Basic Computer Coding:
CSS

2nd Edition

BIBLIOTEX
Digital Library

BASIC COMPUTER CODING: CSS

2nd Edition

BIBLIOTEX
Digital Library

www.bibliotex.com

BASIC COMPUTER CODING: CSS

2ND EDITION

www.bibliotex.com

email: info@bibliotex.com

e-book Edition 2022

ISBN: 978-1-98467-602-3 (e-book)

In Collaboration with 3G E-Learning LLC. Originally Published in printed book format by 3G E-Learning LLC with ISBN 978-1-98465-895-1

EDITORIAL BOARD

Aleksandar Mratinković was born on May 5, 1988 in Arandjelovac, Serbia. He has graduated on Economic high school (2007), The College of Tourism in Belgrade (2013), and also has a master degree of Psychology (Faculty of Philosophy, University of Novi Sad). He has been engaged in different fields of psychology (Developmental Psychology, Clinical Psychology, Educational Psychology and Industrial Psychology) and has published several scientific works.

Dan Piestun (PhD) is currently a startup entrepreneur in Israel working on the interface of Agriculture and Biomedical Sciences and was formerly president-CEO of the National Institute of Agricultural Research (INIA) in Uruguay. Dan is a widely published scientist who has received many honours during his career including being a two-time recipient of the Amit Golda Meir Prize from the Hebrew University of Jerusalem, his areas of expertise includes stem cell molecular biology, plant and animal genetics and bioinformatics. Dan's passion for applied science and technological solutions did not stop him from pursuing a deep connection to the farmer, his family and nature. Among some of his interest and practices counts enjoying working as a beekeeper and onboard fishing.

Hazem Shawky Fouda has a PhD. in Agriculture Sciences, obtained his PhD. From the Faculty of Agriculture, Alexandria University in 2008, He is working in Cotton Arbitration & Testing General Organization (CATGO).

Felecia Killings is the Founder and CEO of LiyahAmore Publishing, a publishing company committed to providing technical and educational services and products to Christian Authors. She operates as the Senior Editor and Writer, the Senior Writing Coach, the Content Marketing Specialist, Editor-in-Chief to the company's quarterly magazine, the Executive and Host of an international virtual network, and the Executive Director of the company's online school for Authors. She is a former high-school English instructor and professional development professor. She possesses a Master of Arts degree in Education and a Bachelor's degree in English and African American studies.

Dr. Sandra El Hajj, Ph.D. in Health Sciences from Nova Southeastern University, Florida, USA is a health professional specialized in Preventive and Global Health. With her 12 years of education obtained from one of the most prominent universities in Beirut, in addition to two leading universities in the State of Florida (USA), Dr. Sandra made sure to incorporate interdisciplinary and multicultural approaches in her work. Her long years of studies helped her create her own miniature world of knowledge linking together the healthcare field with Medical Research, Statistics, Food Technology, Environmental & Occupational Health, Preventive Health and most noteworthy her precious last degree of Global Health. Till today, she is the first and only doctor specialized in Global Health in the Middle East area.

Fozia Parveen has a Dphil in Sustainable Water Engineering from the University of Oxford. Prior to this she has received MS in Environmental Sciences from National University of Science and Technology (NUST), Islamabad Pakistan and BS in Environmental Sciences from Fatima Jinnah Women University (FJWU), Rawalpindi.

Igor Krunic 2003-2007 in the School of Economics. After graduating in 2007, he went on to study at The College of Tourism, at the University of Belgrade where he got his bachelor degree in 2010. He was active as a third-year student representative in the student parliament.Then he went on the Faculty of science, at the University of Novi Sad where he successfully defended his master's thesis in 2013. The crown of his study was the work titled Opportunities for development of cultural tourism in Cacak". Later on, he became part of a multinational company where he got promoted to a deputy director of logistic. Nowadays he is a consultant and writer of academic subjects in the field of tourism.

Dr. Jovan Pehcevski obtained his PhD in Computer Science from RMIT University in Melbourne, Australia in 2007. His research interests include big data, business intelligence and predictive analytics, data and information science, information retrieval, XML, web services and service-oriented architectures, and relational and NoSQL database systems. He has published over 30 journal and conference papers and he also serves as a journal and conference reviewer. He is currently working as a Dean and Associate Professor at European University in Skopje, Macedonia.

Dr. Tanjina Nur finished her PhD in Civil and Environmental Engineering in 2014 from University of Technology Sydney (UTS). Now she is working as Post-Doctoral Researcher in the Centre for Technology in Water and Wastewater (CTWW) and published about eight International journal papers with 80 citations. Her research interest is wastewater treatment technology using adsorption process.

Stephen obtained his PhD from the University of North Carolina at Charlotte in 2013 where his graduate research focused on cancer immunology and the tumor microenvironment. He received postdoctoral training in regenerative and translational medicine, specifically gastrointestinal tissue engineering, at the Wake Forest Institute of Regenerative Medicine. Currently, Stephen is an instructor for anatomy and physiology and biology at Forsyth Technical Community College.

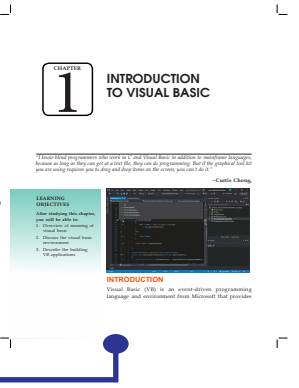
Michelle holds a Masters of Business Administration from the University of Phoenix, with a concentration in Human Resources Management. She is a professional author and has had numerous articles published in the Henry County Times and has written and revised several employee handbooks for various YMCA organizations throughout the United States.

HOW TO USE THE BOOK

This book has been divided into many chapters. Chapter gives the motivation for this book and the use of templates. The text is presented in the simplest language. Each paragraph has been arranged under a suitable heading for easy retention of concept. Keywords are the words that academics use to reveal the internal structure of an author's reasoning. Review questions at the end of each chapter ask students to review or explain the concepts. References provides the reader an additional source through which he/she can obtain more information regarding the topic.

LEARNING OBJECTIVES

See what you are going to cover and what you should already know at the start of each chapter


CHAPTER 1

INTRODUCTION TO VISUAL BASIC

"I know blind programmers who work in C and Visual Basic in addition to mainframe languages, because as long as they can get at a text file, they can do programming. But if the graphical tool kit you are using requires you to drag and drop items on the screen, you can't do it."

–Curtis Chong,

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Overview of meaning of visual basic
2. Discuss the visual basic programming
3. Describe the building VB applications

INTRODUCTION

Visual Basic (VB) is an event-driven programming language and environment from Microsoft that provides

enhancements, including the striking ability of creating web based applications. The extended support for Visual Basic 6.0 was ceased in the month of March in 2008. The basic parts of development environment of Visual Basic 6, however, still run in all the 32-bit Microsoft windows, including Windows 8.1.

After the cessation of mainstream and extended support for Visual Basic 6.0 caused a number of programs to show concern. The community members then created a lobby of users and a petition was signed by them. The basic aim of this petition was to ensure that the product remains alive. However, the petition did not attain its aim effectively.

1.1.2 The Importance of Visual Basic Programming Language

Visual Basic is regarded as the third generation event-drivon programming language. It was released in 1987. Being the first visual development tool from Microsoft, it is considered as one of the most powerful programming languages. As compared to other computer programming languages, such as, C, C++, it is easy to learn and understand, provided that one has determination and dedication to do so.

Visual basic programming language allows programmers to create software interface and codes in an easy to use graphical environment. VB is the combination of different components that are used on forms having specific attributes and actions with the help of those components. On the one hand it allows programmers to develop seldom based applications rapidly; on the other hand, it helps greatly in accessing data bases, using ADO while letting the programmers use ActiveX controls and various objects. While it is intended more to develop applications, it is also useful for games development for particular or limited purposes, unlike C++ that is more suitable for developing games.

As compared to other languages, Visual basic may be slower though, yet it is flexible and it can be rightly said that things that are difficult in other languages are comparatively easier in visual basic programming language. It may also be said that, since it is one of the most popular programming languages, lots of related books and material and other resources are available and can be accessed for developing programming skills at visual basic programming language conveniently.

One of the most important things to be considered with regard to programming in Visual basic is that the structure of VB is designed in a way that allows programmers to create executable code – Exe files. It enables programmers to develop programs that can be used as front end to databases. Besides, it's with the help of visual basic tools, one can change the abstract ideas into programs or into the whole software while it allows revising and modifying the programs fittingly.

ABOUT THIS CHAPTER

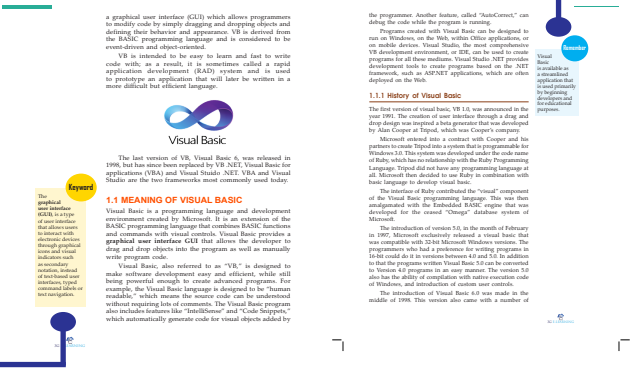
An introduction is a beginning of section which states the purpose and goals of the topics which are discussed in the chapter. It also starts the topics in brief.

REMEMBER

This revitalizes a must read information of the topic.

KEYWORDS

This section contains some important definitions that are discussed in the chapter. A keyword is an index entry that identifies a specific record or document. It also gives the extra information to the reader and an easy way to remember the word definition.


Keyword

The graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to of text-based user interfaces, typed command labels or text navigation.

a graphical user interface (GUI) which allows programmers to modify code by simply dragging and dropping objects and defining their behavior and appearance. VB is derived from the BASIC programming language and is considered to be event-driven and object-oriented.

VB is intended to be easy to learn and fast to write code with; as a result, it is sometimes called a rapid application development (RAD) system and is used to prototype an application that will later be written in a more difficult but efficient language.

Visual Basic

The last version of VB, Visual Basic 6, was released in 1998, but has since been replaced by VB .NET, Visual Basic for applications (VBA) and Visual Studio .NET. VBA and Visual Studio are the two frameworks most commonly used today.

1.1 MEANING OF VISUAL BASIC

Visual Basic is a programming language and development environment created by Microsoft. It is an extension of the BASIC programming language that combines BASIC functions and commands with visual controls. Visual Basic provides a **graphical user interface GUI** that allows the developer to drag and drop objects into the program as well as manually write program code.

Visual Basic, also referred to as "VB," is designed to make software development easy and efficient, while still being powerful enough to create advanced programs. For example, the Visual Basic language is designed to be "human readable," which means the source code can be understood without requiring lots of comments. The Visual Basic program also includes features like "IntelliSense" and "Code Snippets," which automatically generate code for visual objects added by

the programmer. Another feature, called "AutoCorrect," can debug the code while the program is running.

Programs created with Visual Basic can be designed to run on Windows, on the Web, within Office applications, or on mobile devices. Visual Studio, the most comprehensive VB development environment, or IDE, can be used to create programs for all these mediums. Visual Studio .NET provides development tools to create programs based on the .NET framework, such as ASP.NET applications, which are often deployed on the Web.

Remember

Visual Basic is available as a standalone application that is used primarily by beginning developers and for educational purposes.

1.1.1 History of Visual Basic

The first version of visual basic, VB 1.0, was announced in the year 1991. The creation of user interface through a drag and drop design was inspired a beta generator that was developed by Alan Cooper at Tripod, which was Cooper's company.

Microsoft entered into a contract with Cooper and his partners to create Tripod into a system that is programmable for Windows 3.0. This system was developed under the code name of Ruby, which has no relationship with the Ruby Programming Language. Tripod did not have any programming language at all. Microsoft then decided to use Ruby in combination with basic language to develop visual basic.

The interface of Ruby contributed the "visual" component of the Visual Basic programming language. This was then amalgamated with the Embedded BASIC engine that was developed for the ceased "Omega" database system of Microsoft.

The introduction of version 5.0, in the month of February in 1997, Microsoft exclusively released a visual basic that was compatible with 32-bit Microsoft Windows versions. The programmers who had a preference for writing programs in 16-bit could do it in versions between 4.0 and 5.0. In addition to that the programs written Visual Basic 5.0 can be converted to Version 4.0 programs in an easy manner. The version 5.0 also has the ability of compilation with native execution code of Windows, and introduction of custom user controls.

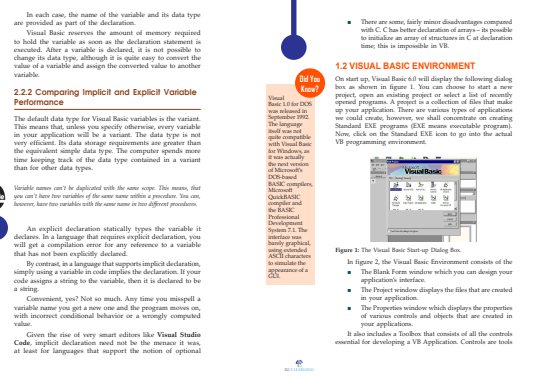
The introduction of Visual Basic 6.0 was made in the middle of 1998. This version also came with a number of

DID YOU KNOW?

This section equip readers the interesting facts and figures of the topic.

EXAMPLE

The book cabinets' examples to illustrate specific ideas in each chapter.



In each case, the name of the variable and its data type are provided as part of the declaration.

Visual Basic reserves the amount of memory required to hold the variable as soon as the declaration statement is executed. After a variable is declared, it is not possible to change its data type, although it is quite easy to convert the value of a variable and assign the converted value to another variable.

2.2.2 Comparing Implicit and Explicit Variable Performance

The default data type for Visual Basic variables is the variant. This means that, unless you specify otherwise, every variable in your application will be a variant. The data type is not very efficient. Its data storage requirements are greater than the equivalent simple data type. The computer spends more time keeping track of the data type contained in a variant than for other data types.

Variable names can't be duplicated with the same scope. This means, that you can't have two variables of the same name within a procedure. You can, however, have two variables with the same name in two different procedures.

An explicit declaration statically types the variable it declares. In a language that requires explicit declaration, you will get a compilation error for any reference to a variable that has not been explicitly declared.

By contrast, in a language that supports implicit declaration, simply using a variable in code implies the declaration. If your code assigns a string to the variable, then it is declared to be a string.

Convenient, yes? Not so much. Any time you misspell a variable name you get a new one and the program moves on, with incorrect conditional behavior or a wrongly computed value.

Given the rise of very smart editors like **Visual Studio Code**, implicit declaration need not be the menace it was, at least for languages that support the notion of optional

- There are some, fairly minor disadvantages compared with C. C has better declaration of arrays – its possible to initialize an array of structures in C at declaration time; this is impossible in VB.

1.2 VISUAL BASIC ENVIRONMENT

Did You Know?

Visual Basic 1.0 for DOS was released in September 1992. The language itself was not quite compatible with Visual Basic for Windows, as it was actually the next version of Microsoft's DOS-based BASIC compilers, Microsoft QuickBASIC compiler and the BASIC Professional Development System 7.1. The interface was barely graphical, using extended ASCII characters to simulate the appearance of a GUI.

On start up, Visual Basic 6.0 will display the following dialog box as shown in figure 1. You can choose to start a new project, open an existing project or select a list of recently opened programs. A project is a collection of files that make up your application. There are various types of applications we could create, however, we shall concentrate on creating Standard EXE programs (EXE means executable program). Now, click on the Standard EXE icon to go into the actual VB programming environment.

Figure 1: The Visual Basic Start-up Dialog Box.

In figure 2, the Visual Basic Environment consists of the

- The Blank Form window which you can design your application's interface.
- The Project window displays the files that are created in your application.
- The Properties window which displays the properties of various controls and objects that are created in your applications.

It also includes a Toolbox that consists of all the controls essential for developing a VB Application. Controls are tools

ROLE MODEL

A biography of someone who has/had acquired remarkable success in their respective field as Role Models are important because they give us the ability to imagine our future selves.

CASE STUDY

This reveals what students need to create and provide an opportunity for the development of key skills such as communication, group working and problem solving.

ROLE MODEL

ALAN COOPER: FATHER OF VISUAL BASIC

Born in San Francisco in 1952 and raised in Marin County, California, Alan Cooper has always taken the path less traveled. A rebellious teenager, he dropped out of high school, but eventually made his way to the College of Marin to pursue his interest in architecture. After an exploratory course in programming, it became clear that his future was in architecture—software architecture. After getting his associate degree and a COBOL programming job, he saw an advertisement for one of the first personal computers and .conceived an idea for a new business venture

In 1976, Cooper founded Structured Systems Group (SSG), a company Fire in the Valley authors Paul Freiberger and Michael Swaine said created "the first serious business software for microcomputers." In four years, Cooper wrote and shipped a dozen application programs. SSG became the archetype for many software startups in the early days of the personal computer revolution.

During the 1980s, after leaving SSG, Cooper invented, wrote, and sold three major software packages to prominent publishers. One of those was the visual programming front-end code named "Ruby," for what became Visual BASIC. Bill Gates purchased it from Cooper in 1988, noting that it would have significant impact across Microsoft's entire product line. Visual BASIC was deemed both a commercial and critical success, earning Cooper the moniker "Father of Visual BASIC." Visual BASIC has influenced integrated development languages ever since.

In 1990 Cooper became fascinated with the challenge of making software products that were easy to use and understand. He and his wife, Susan, founded Cooper Interaction Design (now "Cooper") to assist in what Cooper calls "interaction design." In the design field, Cooper's software development background was unique and, over the next few years, he invented many of the tools and techniques now standard in the user experience industry, including personas and scenarios,

CASE STUDY

FUJITSU FACILITATES SMOOTH MIGRATION TO VB.NET AT AN POST

Fujitsu has an excellent technical team, which works closely with our staff. We have had a good working relationship for many years and Fujitsu has an in-depth knowledge of our mission-critical application gained from several years' development and support work.

Challenge

A Post, one of Ireland's largest companies, is a major commercial organization providing a wide range of postal, communication, retail and financial services. With 9,600 employees throughout its national network of retail, processing and delivery points, the business also provides services to government departments, the National Treasury Management Agency and its own National Lottery Company. A decade ago, A Post implemented a new nationwide time and attendance system to calculate and record staff salary and wages functions. The Staff Remuneration and Administration Management System (STREAMS) is a bespoke, mission-critical application developed by Fujitsu as a reliable, scalable client server system using Microsoft technologies. The STREAMS front-end system gathers information and feeds the data to the company's HR, payroll and financial departments. It primarily creates more efficient processes for A Post to capture data for the weekly payroll run whilst simultaneously minimizing the number of payroll queries by employees. Following deployment, STREAMS improved cost center reporting, significantly lowered the time to record pay details and enhanced the processing of casual staff pay. During this period, Fujitsu provided quality support and maintenance services and application enhancements to increase functionality, ensuring the long-term reliability of STREAMS. For instance, as employee numbers steadily increased to exceed original expectations, Fujitsu boosted system performance by upgrading the infrastructure and optimizing the software. STREAMS originally employed Visual Basic (VB), a third-generation event-driven programming language and integrated development environment (IDE) from Microsoft. IDE provides programmers with comprehensive facilities for software development and comprises a source code editor, a compiler and/or an interpreter, build automation tools and a debugger. However, Microsoft no longer supports VB version 6.0, the edition employed by A Post. Syl Byrne, IT Manager Remuneration Services, A Post, explains: "To ensure that our business-critical application is future-proof, we needed to move to a platform that Microsoft will support for the foreseeable future." A Post therefore decided to migrate STREAMS to the VB.NET platform, an object-orientated programming language. This strategy would protect its investment for the next 10 years by creating a secure, scalable

KNOWLEDGE CHECK

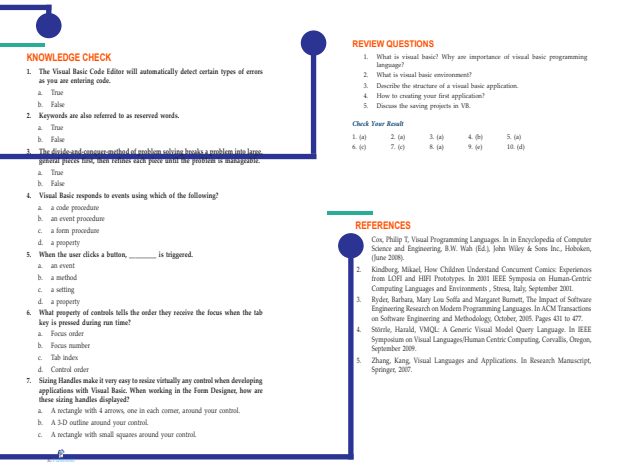
This is given to the students for progress check at the end of each chapter.

REVIEW QUESTIONS

This section is to analyze the knowledge and ability of the reader.

REFERENCES

References refer those books which discuss the topics given in the chapters in almost same manner.



KNOWLEDGE CHECK

1. **The Visual Basic Code Editor will automatically detect certain types of errors as you are entering code.**
   a. True
   b. False
2. **Keywords are also referred to as reserved words.**
   a. True
   b. False
3. **The divide-and-conquer method of problem solving breaks a problem into large general pieces first, then refines each piece until the problem is manageable.**
   a. True
   b. False
4. **Visual Basic responds to events using which of the following?**
   a. a code procedure
   b. an event procedure
   c. a form procedure
   d. a property
5. **When the user clicks a button, _____ is triggered.**
   a. an event
   b. a method
   c. a setting
   d. a property
6. **What property of controls tells the order they receive the focus when the tab key is pressed during run time?**
   a. Focus order
   b. Focus number
   c. Tab index
   d. Control order
7. **Sizing Handles make it very easy to resize virtually any control when developing applications with Visual Basic. When working in the Form Designer, how are these sizing handles displayed?**
   a. A rectangle with 4 arrows, one in each corner, around your control.
   b. A 3-D outline around your control.
   c. A rectangle with small squares around your control.

REVIEW QUESTIONS

1. What is visual basic? Why are importance of visual basic programming language?
2. What is visual basic environment?
3. Describe the structure of a visual basic application.
4. How to creating your first application?
5. Discuss the saving projects in VB.

Check Your Result

1. (a)	2. (a)	3. (a)	4. (b)	5. (a)
6. (c)	7. (c)	8. (a)	9. (e)	10. (d)

REFERENCES

1. Cox, Philip T, Visual Programming Languages. In in Encyclopedia of Computer Science and Engineering, B.W. Wah (Ed.), John Wiley & Sons Inc., Hoboken, (June 2008).
2. Kindborg, Mikael, How Children Understand Concurrent Comics: Experiences from LOFI and HIFI Prototypes. In 2001 IEEE Symposia on Human-Centric Computing Languages and Environments , Stresa, Italy, September 2001.
3. Ryder, Barbara, Mary Lou Soffa and Margaret Burnett, The Impact of Software Engineering Research on Modern Programming Languages. In ACM Transactions on Software Engineering and Methodology, October, 2005. Pages 431 to 477.
4. Störrle, Harald, VMQL: A Generic Visual Model Query Language. In IEEE Symposium on Visual Languages/Human Centric Computing, Corvallis, Oregon, September 2009.
5. Zhang, Kang, Visual Languages and Applications. In Research Manuscript, Springer, 2007.

TABLE OF CONTENTS

PREFACE

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is used to style and lay out your content, whereas HTML is used to define its structure and semantics. CSS, for example, can be used to change the font, color, size, and spacing of your content, divide it into multiple columns, or add animations and other decorative elements.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. Cascading Style Sheets (CSS) are an important way to control how your Web pages look. CSS can control the fonts, text, colors, backgrounds, margins, and layout. CSS is easy to learn and understand but it entails powerful control to the presentation of an HTML document. Usually, CSS is integrated with the markup languages HTML or XHTML.

Organization of the Book

This book is systematically organized into nine chapters in this edition. This edition is packed up with updated information. With this book, you can familiarize yourself with how CSS works, learn how to efficiently work with CSS selectors, and apply what you've learned to create beautifully styled simple web pages.

Chapter 1 introduces the meaning of CSS, and discusses about the CSS selectors. It also explains the inheritance, including with the cascade.

Chapter 2 focuses on using colors in web design. Typically, these are used to set a color either for the foreground of an element (i.e., its text) or else for the background of the element.

Chapter 3 presents an introduction to box sizing and borders property. In this chapter, you will find box-sizing property to include the padding and border in an element's total width and height. The CSS border properties allow you to specify the style, width, and color of an element's border.

Chapter 4 gives an exploration on the appearance of cascading style sheets. In this chapter, you will learn about the several styles of CSS and CSS validation service.

Chapter 5 is aimed to discuss the positioning elements. CSS helps you to position your HTML element. You can put any HTML element at whatever location you like. Fixed positioning allows you to fix the position of an element to a particular spot on the page, regardless of scrolling.

Chapter 6 begins with concept of CSS font-face Rule. It also focuses on CSS Font Property, including with Web Safe Fonts.

Chapter 7 discusses about Lists, Table and Forms. Lists behave like any other text for the most part, but there are some CSS properties specific to lists that you need to know about, and some best practices to consider. Tables are a nice way to organize a lot of data.

Chapter 8 explores the basic concepts of grid layout. Further, it describes the relationship of grid layout to other layout methods. The grid layout gives us a way to create grid structures depicted in CSS, not in HTML. Similar to the table, it enables a user to align the elements into rows and columns.

Chapter 9 shows how to work with images and multimedia. Images are an important part of any web application. Including a lot of images in a web application is generally not recommended, but it is important to use the images wherever they required. CSS helps us to control the display of images in web applications.

CHAPTER

1

INTRODUCTION TO CSS

"Learning HTML and CSS is a lot more challenging than it used to be. Responsive web design adds more layers of complexity to design and develop websites."

– Jacob Lett

LEARNING OBJECTIVES

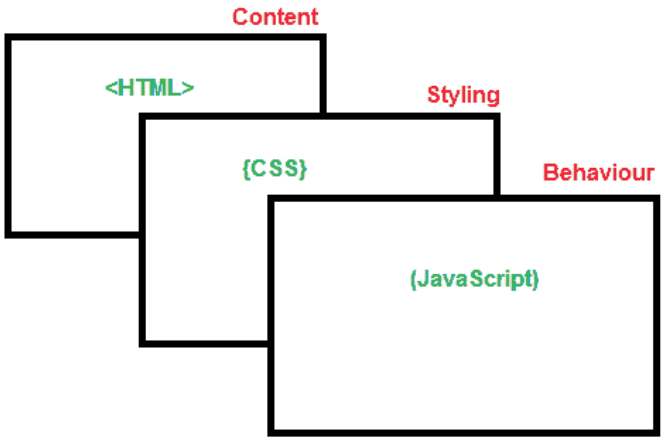
After studying this chapter, you will be able to:

1. Give meaning of CSS
2. Discuss about the CSS selectors
3. Explain the inheritance
4. Explain the cascade

INTRODUCTION

CSS stands for Cascading Style Sheets. Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects

of Web pages that previously could only be defined in a page's HTML. CSS helps Web developers create a uniform look across several pages of a Web site. Instead of defining the style of each table and each block of text within a page's HTML, commonly used styles need to be defined only once in a CSS document. Once the style is defined in cascading style sheet, it can be used by any page that references the CSS file. Plus, CSS makes it easy to change styles across several pages at once. For example, a Web developer may want to increase the default text size from 10pt to 12pt for fifty pages of a Web site. If the pages all reference the same style sheet, the text size only needs to be changed on the style sheet and all the pages will show the larger text.



Cascading Style Sheets

While CSS is great for creating text styles, it is helpful for formatting other aspects of Web page layout as well. For example, CSS can be used to define the cell padding of table cells, the style, thickness, and color of a table's border, and the padding around images or other objects. CSS gives Web developers more exact control over how Web pages will look than HTML does. This is why most Web pages today incorporate cascading style sheets.

1.1 MEANING OF CSS

Cascading Style Sheet (CSS) is a way to design a website, or a group of websites, so that they have a consistent look and feel, and so that their look and feel is easy to change. By using CSS to design a **website**, the web developer gains a greater degree of control over how the site appears.

A web developer can use a CSS file to control the look of a website in three main ways. The first way is called inline, referring to the fact that the code is placed right into the line of the website code. For example, a web developer might want to make a particular sentence appear in bold, red type so that it stands out. She could use CSS to set the style of that sentence to bold and red using inline code. The benefit of this method is that it allows a quick and easy change to a particular part of a web page. Another way that a web developer can use CSS is to make rules for a single web page. In this case, the developer would use what is called embedded CSS. The developer can, for example, make each new paragraph indent and each header in bold. The embedded instructions are usually placed at the top of the web page's code. This allows the developer to change the embedded code once and have the effects take place throughout the entire page. If he decided to put all headers in italics rather than bold text, he could simply change the style coding, and all the headers on that page would change. This has an advantage over the inline method in that it covers the entire web page, and changes can be made to the entire page at once.

The final common type of CSS is what is known as an external CSS. A web developer will write the code to apply to an entire group of web pages, a whole website, or even multiple web sites. These rules can include things like background color, text color, word spacing, and other elements of page layout, just like the previous two examples of CSS. The difference is that these instructions are not for a single section of the page, or just one web page, but for an entire website. The advantage is that the look and feel of an entire website can be changed at one time by making changes to the external style sheet. If the designer wants to try a new background color or a new font for the entire website, she can do so with the change of a few lines in the external code, rather than going to each page individually and making changes there.

Keyword

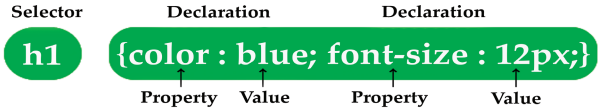
A **website** is a collection of related web pages, including multimedia content, typically identified with a common domain name, and published on at least one web server.

Did You Know?

CSS was first proposed by Håkon Wium Lie on October 10, 1994.

1.1.1 CSS Syntax

A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

In the following example all <p> elements will be center-aligned, with a red text color:

Example

```
p {
    color: red;
    text-align: center;
}
```

1.1.2 History of CSS

Before Cascading Style Sheets (CSS) there was very little that could be done to change the design of a web page. While Hyper Text Markup Language (HTML) creates documents for the World Wide Web, it was specifically designed to hold the content of a web page. Housed in a separate file, CSS adds the style and design to a web page. The term cascading comes from the ability to combine multiple CSS files to determine the style for one page.

As more people started using HTML, the demand grew for more design capabilities, which would allow developers to control how web documents looked. But browsers offered limited capabilities for styling. In 1993 NCSA Mosaic was released, making the web more popular than ever, but it only offered limited capability to change fonts and colors. In October 1994, Tim Berners-Lee formed the World Wide Web Consortium (W3C) at the Massachusetts Institute of Technology Laboratory for Computer Science. The W3C has members that are government entities, businesses, educational institutions and individuals. The W3C creates recommendations that are used to keep the web experience consistent among different browsers.

Hakon Wium Lie released the first draft of "Cascading HTML Style Sheets" in October 1994. Brent Bos was also building Argo, a browser that had style sheets which could be used by other markup languages. In all, nine different style sheet proposals were submitted. The discussions on the www-style mailing list, which was created in May 1995, influenced the development of CSS. Because it was important to be able to use the style sheets for other markup languages, HTML was removed from the name. The W3C set up the HTML Editorial Review Board (ERB) in late 1995 to make HTML specifications.

Although it took 3 years for any browser to come close to fully implementing CSS, August 1996 Microsoft Internet Explorer became the first browser to support CSS. Netscape followed suit in supporting CSS, but also implemented an alternative Javascript Style Sheets, which were never fully completed, and are now deprecated. To this day, there are differences in the way CSS is implemented in different browsers, leading developers to use hacks to make web pages look consistent in different browsers.

The CSS Level 1 W3C Recommendation was made in December 1996. The release of CSS 1 supported: font properties, text attributes, alignment of text, tables, images and more, colors of text and backgrounds, spacing of words, letters and lines, margins, borders, padding and positioning, and unique identification and classification of groups of attributes. It was after this release that the Cascading Style Sheets and Formatting Properties Working Group was formed by the W3C to focus solely on CSS.

In May of 1998, the W3C released CSS 2 which added new capabilities including z-index, media types, bidirectional text, absolute, relative and fixed positioning, and support for aural style sheets. Not long after the release of CSS 2, a new browser, Opera was released which also supported CSS. It wasn't until June 2011 that W3C released CSS 2.1, which fixed errors and aligned the capabilities better with browser functions.

With the release of CSS 3, capabilities were broken into modules. Between June 2011 and June 2012, the following four modules were released as formal recommendations: color, selectors level 3, namespaces and media queries. More modules are have reached the working draft or candidate recommendation phase of release and are considered stable.

Because level 3 was separated into modules, there will be no single CSS 4 release. Some level 4 modules which build on level 3 functions have released. New functionality has been added in other modules. Modules will continue to be released to continue to add and update features.

Although there are some limitations, CSS has added functionality and design to web pages which were once simple documents. The advantages of CSS include content and presentation being in separate files, consistency of format throughout a site, reduction of bandwidth, accessibility, the ability to easily change the style of a site, and the ability to change the design of a web page for different devices. If you take a moment to imagine a website where the only design features are font and color

changes, you will recognize that CSS has contributed greatly to the popularity, ease of use and accessibility of the internet.

1.1.3 Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

External Style Sheet

With an external style sheet, you can change the look of an entire website by changing just one file!

Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section:

Example

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

Here is how the "mystyle.css" looks:

```
body {
    background-color: lightblue;
}
h1 {
    color: navy;
    margin-left: 20px;
}
```

Internal Style Sheet

An internal style sheet may be used if one single page has a unique style.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

Example

```
<head>
<style>
body {
    background-color: linen;
}

h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
```

Inline Styles

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

The example below shows how to change the color and the left margin of a <h1> element:

Example

```
<h1 style="color:blue;margin-left:30px;">This is a heading</h1>
```

1.1.4 Advantages of CSS

- **CSS saves time** – You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** – If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
- **Easy maintenance** – To make a global change, simply change the style, and all elements in all the **web pages** will be updated automatically.
- **Superior styles to HTML** – CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

- **Multiple Device Compatibility** – Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** – Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

1.1.5 Disadvantages of CSS

1. Come in different levels – There's CSS, CSS 1 up to CSS3, which has resulted in confusion among developers and web browsers. One type of CSS should be enough. It would be preferable than having to choose which CSS level to use.
2. Fragmentation – With CSS, what works with one browser may not always work with another. This is why web developers have to test for compatibility, running the program across multiple browsers before a website is set live. If only people use Mozilla or Chrome, but they don't.
3. Lack of security– Because it is an open text-based system, CSS doesn't have the built-in security that will protect it from being overridden. Anyone who has a read/write access to a website can change the CSS file, alter the links or disrupt the formatting, whether by accident or design.

Keyword

A **web page** is a document that is suitable for the World Wide Web and web browsers.

1.2 CSS SELECTORS

A CSS selector is the part of a CSS rule set that actually selects the content you want to style. Let's look at all the different kinds of selectors available, with a brief description of each.

1.2.1 Universal Selector

The *universal selector* works like a wild card character, selecting all elements on a page. Every HTML page is built on content

placed within HTML tags. Each set of tags represents an element on the page. Look at the following CSS example, which uses the universal selector:

```
* {
    color: green;
    font-size: 20px;
    line-height: 25px;
}
```

The three lines of code inside the curly braces (color, font-size, and line-height) will apply to all elements on the HTML page. As seen here, the universal selector is declared using an asterisk. You can also use the universal selector in combination with other selectors.

1.2.2 Element Type Selector

Also referred to simply as a "type selector," this selector must match one or more HTML elements of the same name. Thus, a selector of nav would match all HTML nav elements, and a selector of <ul> would match all HTML unordered lists, or <ul> elements.

The following example uses an element type selector to match all <ul> elements:

```
ul {
    list-style: none;
    border: solid 1px #ccc;
}
```

To put this in some context, here's a section of HTML to which we'll apply the above CSS:

```
<ul>
  <li>Fish</li>
  <li>Apples</li>
  <li>Cheese</li>
</ul>

<div class="example">
  <p>Example paragraph text.</p>
</div>

<ul>
```

```
<li>Water</li>
<li>Juice</li>
<li>Maple Syrup</li>
</ul>
```

There are three main elements making up this part of the page: Two <ul> elements and a . The CSS will apply only to the two <ul> elements, and not to the . Were we to change the element type selector to use instead of <ul>, then the styles would apply to the and not to the two <ul> elements.

Also note that the styles will not apply to the elements inside the <ul> or elements. That being said, some of the styles may be inherited by those inner elements.

1.2.3 ID Selector

An ID selector is declared using a hash, or pound symbol (#) preceding a string of characters. The string of characters is defined by the developer. This selector matches any HTML element that has an ID attribute with the same value as that of the selector, but minus the hash symbol.

Here's an example:

```
#container {
    width: 960px;
    margin: 0 auto;
}
```

This CSS uses an ID selector to match an HTML element such as:

```
<div id="container"></div>
```

The fact that this is a element doesn't matter—it could be any kind of HTML element. As long as it has an ID attribute with a value of container, the styles will apply.

An ID element on a web page should be unique. That is, there should only be a single element on any given page with an ID of container. This makes the ID selector quite inflexible, because the styles used in the ID selector rule set can be used only once per page.

If there happens to be more than one element on the page with the same ID, the styles will still apply, but the HTML on such a page would be invalid from a technical standpoint, so you'll want to avoid doing this.

In addition to the problems of inflexibility, ID selectors also have the problem of very high specificity.

1.2.4 Class Selector

The class selector is the most useful of all CSS selectors. It's declared with a dot preceding a string of one or more characters. Just as is the case with an ID selector, this string of characters is defined by the developer. The class selector also matches all elements on the page that have their class attribute set to the same value as the class, minus the dot.

Take the following rule set:

```
.box {
    padding: 20px;
    margin: 10px;
    width: 240px;
}
```

These styles will apply to the following HTML element:

```
<div class="box"></div>
```

The same styles will also apply to any other HTML elements that have a class attribute with a value of box. Having multiple elements on a single page with the same class attribute is beneficial, because it allows you to reuse styles, and avoid needless repetition. In addition to this, class selectors have very low specificity—again, more on this later.

Another reason the class selector is a valuable ally is that HTML allows multiple classes to be added to a single element. This is done by separating the classes in the HTML class attribute using spaces. Here's an example:

```
<div class="box box-more box-extended"></div>
```

1.2.5 Descendant Combinator

The descendant selector or, more accurately, the descendant combinator lets you combine two or more selectors so you can be more specific in your selection method. For example:

```
#container .box {
    float: left;
    padding-bottom: 15px;
}
```

This declaration block will apply to all elements that have a class of box that are inside an element with an ID of container. It's worth noting that the .box element doesn't have to be an immediate child: there could be another element wrapping .box, and the styles would still apply.

Look at the following HTML:

```
<div id="container">
  <div class="box"></div>
  <div class="box-2"></div>
</div>
<div class="box"></div>
```

If we apply the CSS in the previous example to this section of HTML, the only element that'll be affected by those styles is the first element that has a class of box. The element that has a class of box-2 won't be affected by the styles. Similarly, the second element with a class of box won't be affected because it's not inside an element with an ID of container.

You should be careful when using the descendant combinator in your CSS. This kind of selector, while making your CSS a little easier to read, can unnecessarily restrict your styles to a specific context—in this case, the styles are restricted to boxes inside of #container—which can make your code inflexible.

1.2.6 Child Combinator

A selector that uses the child combinator is similar to a selector that uses a descendant combinator, except it only targets immediate child elements:

```
#container > .box {
   float: left;
   padding-bottom: 15px;
}
```

This is the same code from the descendant combinator example, but instead of a space character, we're using the greater-than symbol (or right angle bracket.)

In this example, the selector will match all elements that have a class of box and that are immediate children of the #container element. That means, unlike the descendant combinator, there can't be another element wrapping .box—it has to be a direct child element.

Here's an HTML example:

```
<div id="container">
  <div class="box"></div>
  <div>
    <div class="box"></div>
  </div>
```

```
</div>
```

In this example, the CSS from the previous code example will apply only to the first element that has a class of box. As you can see, the second element with a class of box is inside another element. As a result, the styles will not apply to that element, even though it too has a class of box.

Again, selectors that use this combinator can be somewhat restricting, but they can come in handy—for example, when styling nested lists.

1.2.7 General Sibling Combinator

A selector that uses a general sibling combinator matches elements based on sibling relationships. That is to say, the selected elements are beside each other in the HTML.

```
h2 ~ p {
   margin-bottom: 20px;
```

This type of selector is declared using the tilde character (~). In this example, all paragraph elements (<p>) will be styled with the specified rules, but only if they are siblings of <h2> elements. There could be other elements in between the <h2> and <p>, and the styles would still apply.

Let's apply the CSS from above to the following HTML:

```
<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<div class="box">
  <p>Paragraph example.</p>
</div>
```

In this example, the styles will apply only to the first three paragraph elements. The last paragraph element is not a sibling of the <h2> element because it sits inside the element.

1.2.8 Adjacent Sibling Combinator

A selector that uses the adjacent sibling combinator uses the plus symbol (+), and is almost the same as the general sibling selector. The difference is that the targeted element must be an immediate sibling, not just a general sibling. Let's see what the CSS code for this looks like:

```
p + p {
```

```
    text-indent: 1.5em;
    margin-bottom: 0;
}
```

This example will apply the specified styles only to paragraph elements that immediately follow other paragraph elements. This means the first paragraph element on a page would not receive these styles. Also, if another element appeared between two paragraphs, the second paragraph of the two wouldn't have the styles applied.

So, if we apply this selector to the following HTML:

```
<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<div class="box">
  <p>Paragraph example.</p>
  <p>Paragraph example.</p>
</div>
```

1.2.9 Attribute Selector

The attribute selector targets elements based on the presence and/or value of HTML attributes, and is declared using square brackets:

```
input[type="text"] {
    background-color: #444;
    width: 200px;
}
```

There should not be a space before the opening square bracket unless you intend to use it along with a descendant combinator. The above CSS would match the following element:

```
<input type="text">
```

But it wouldn't match this one:

```
<input type="submit">
```

The attribute selector can also be declared using just the attribute itself, with no value, like this:

```
input[type] {
    background-color: #444;
```

```
    width: 200px;
}
```

This will match all input elements with an attribute of type, regardless of the value.

You can also use attribute selectors without specifying anything outside the square brackets (thus targeting based on the attribute alone, irrespective of the element). It's also worth noting that, when using values, you have the option to include quotes (single or double,) or not.

1.2.10 Pseudo-class

A pseudo-class uses a colon character to identify a pseudo-state that an element might be in—for example, the state of being hovered, or the state of being activated. Let's look at a common example:

```
a:hover {
    color: red;
}
```

The pseudo-class portion of the selector is the :hover part. Here we've attached this pseudo-class to all anchor elements (elements). This means that when the user hovers their mouse over an element, the color property for that element will change to red. This type of pseudo-class is a dynamic pseudo-class, because it occurs only in response to user interaction—in this case, the mouse moving over the targeted element.

1.2.11 Pseudo-element

Finally, CSS has a selector referred to as a pseudo-element and, used appropriately, it can be very useful. The only caveat is that this selector is quite different from the other examples we've considered. Let's see a pseudo-element in context:

```
.container:before {
    content: «»;
    display: block;
    width: 50px;
    height: 50px;
    background-color: #141414;
}
```

This example uses one kind of pseudo-element, the :before pseudo-element. As the name suggests, this selector inserts an imaginary element into the page, inside the targeted element, before its contents.

1.3 INHERITANCE

Inheritance is the mechanism by which certain properties are passed on from a parent element down to its children, in the same fashion as genetics: if parents have blue eyes, their children will probably also have blue eyes.

Not all CSS properties are inherited, because it does not make sense for some of them to be. For instance, margins and width are not inherited, since it is unlikely that a child element requires the same margins as its parent. Imagine if you set the content block of a site to be 70% of the browser window width, and then all of its children adopted a width of 70% of their parent elements? Designing **page layouts** with CSS would be a nightmare.

Remember

It's important to recognize that these types of selectors do not just select elements; they select elements that are in a particular state.

1.3.1 Why Inheritance is Useful

CSS has an inheritance mechanism because otherwise CSS rules would be redundant. Without inheritance, it would be necessary to specify styles like font family, font size, and text color individually — for every single element type. The code would become bloated and repetitive. Using inheritance, you can specify the font properties for the html or body elements and the styles will be inherited by all other elements. You can specify background and text colors for a specific container element and the text color will automatically be inherited by any child elements in that container. The background color is not inherited, but the initial value for background-color is transparent, which means a parent's background will shine through. The effect is similar to the page's appearance if background colors were inherited.

Note: Consider what would happen if background *images* were inherited. Every child would display the same background image as its parent. The result would look like a jigsaw puzzle put together by someone with a serious drug problem, since the background would be redrawn inside each subsequent child element.

Keyword

Page layout is the part of graphic design that deals in the arrangement of visual elements on a page. It generally involves organizational principles of composition to achieve specific communication objectives.

1.3.2 How Inheritance Works

Every element in an HTML document inherits all inheritable properties from its parent *except* the root element (<html>),

which does not have a parent. Whether or not the inherited properties will have any effect depends on other things, as described later in the section about the cascade. Just as a blue-eyed mother can have a brown-eyed child if the father has brown eyes, inherited properties in CSS can be overridden.

Keyword

Inheritance is the mechanism of basing an object or class upon another object (prototypical inheritance) or class (class-based inheritance), retaining similar implementation.

An example of inheritance

1. Copy the following HTML document into a new file in your favorite text editor and save it as inherit.html.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Inheritance</title>
  </head>
  <body>
   <h1>Heading</h1>
   <p>A paragraph of text.</p>
  </body>
</html>
```

If you open the document in your web browser, you will see a rather boring document displayed according to your browser's default styling.

2. Create a new empty file in your text editor, copy the following CSS rule into it, and save the file as style.css in the same location as the HTML file.

```
} html
;font: 75% Verdana, sans-serif
  }
```

3. Link the style sheet to your HTML document by inserting the following line before the </head> tag:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

4. Save the modified HTML file and reload the document in your browser.

If you do not have Verdana installed on your computer, the text displays in the default sans-serif font specified in your browser settings. The text is also smaller; only three quarters

of its size in the unstyled version. The CSS rule applies only to the <html> element. It does not specify any rules for headings or paragraphs, yet all of the text now displays in Verdana at 75% of its default size. Why? Because of inheritance. The font property is a shorthand property that sets a whole range of font-related properties.

The CSS rule only specified two properties — the font size and the font family — but that rule is equivalent to the following:

```
html {
  font-style: normal;
  font-variant: normal;
  font-weight: normal;
  font-size: 75%;
  line-height: normal;
  font-family: Verdana,sans-serif;
}
```

All of those properties are inherited, so the <body> element will inherit them from the <html> element and then pass them on to its children — the heading and the paragraph text. But notice that there is something unusual occurring with the inheritance of font size.

The <html> element's font size is set to 75%, but 75% of *what*? Surely the font size of the <body> should be 75% of its parent's font size, and the font sizes of the heading and the paragraph be 75% of that of the <body> element? The value inherited is not the specified value — the value typed in the style sheet — but something else called the computed value.

The computed value is, in the case of font size, an absolute value measured in pixels. For the <html> element, which doesn't have a parent element from which to inherit, a percentage value for font size relates to the default font size set in the browser. Most contemporary browsers have a default font size setting of 16px. 75% of 16 is 12, so the computed value for the font size of the <html> element is around 12px.

And *that* is the value that is inherited by <body> and passed on to the heading and the paragraph. (The font size of the heading is larger, because the browser applies some built-in styling of its own.

Consider this example:

1. Add two more declarations to the rule in your CSS style sheet:

```
html {
  font: 75% Verdana,sans-serif;
```

```
  /* Add these */
  background-color: blue;
  color: white;
}
```

2. Save the CSS file and reload the document in your browser. Now the background of the whole document is bright blue, and all of the text is white. The white text color is inherited by the <body> element and passed on to all children of body —the heading and the paragraph. The heading and paragraph do not, however, inherit the background but instead default to transparent, so the net visual result is white text displayed on a blue background.
3. Add another new rule to the style sheet:

```
h1 {
  font-size: 300%;
}
```

4. Save and reload the document: this rule sets the font size of the heading. The percentage is applied to the inherited font size — 12px, as we discussed above — so the heading size will be 300% of 12px, or 36px.

1.3.3 Forcing Inheritance

You can force inheritance — even for properties that are not inherited by default — by using the inherit keyword. Use this strategy with care, however, since Microsoft Internet Explorer versions up to and including version 7 do not support this keyword.

The following rule will make all paragraphs inherit all background properties from their parents:

```
p {
  background: inherit;
}
```

Forcing inheritance is not a common practice. It can be useful for "undoing" a declaration in a conflicting rule, or to avoid hard coding certain values. As an example, consider a typical navigation menu:

```
<ul id="nav">
  <li><a href="/">Home</a></li>
  <li><a href="/news/">News</a></li>
  <li><a href="/products/">Products</a></li>
  <li><a href="/services/">Services</a></li>
```

```
  <li><a href="/about/">About Us</a></li>
</ul>
```

To display this list of links as a horizontal menu, you could use the following CSS rules:

```
#nav {
  background: blue;
  color: white;
  margin: 0;
  padding: 0;
}

#nav li {
  display: inline;
  margin: 0;
  padding: 0 0.5em;
  border-right: 1px solid;
}

#nav li a {
  color: inherit;
  text-decoration: none;
}
```

Here the background color of the whole list is set to blue in the rule for #nav. This also sets the foreground color to white, and this property is inherited by each list item and each link. The rule for the list items sets a right border, but doesn't specify the border color, which means it will use the inherited foreground color, white. For the links, the color:inherit; forces inheritance and overrides the browser's default link color.

Of course, you can specify the border color as white and the link text color as white, but the beauty of letting inheritance do the job is that it is only necessary to update one place to change the colors if you decide to update the color scheme later.

1.4 THE CASCADE

CSS an acronym of Cascading Style Sheets, so it is not a surprise that the cascade is an important concept. It is the mechanism that controls the end result when multiple,

conflicting CSS declarations apply to the same element. There are three main concepts that control the order in which CSS declarations are applied:

- Importance
- Specificity
- Source order

Importance is the most … er … important. If two declarations have the same importance, the specificity of the rules decide which one will apply. If the rules have the same specificity, then source order controls the outcome.

1.4.1 Importance

The importance of a CSS declaration depends on where it is specified. The conflicting declarations will be applied in the following order, with later declarations overriding earlier ones:

- User agent style sheets
- Normal declarations in user style sheets
- Normal declarations in author style sheets
- Important declarations in author style sheets
- Important declarations in user style sheets

User Agent Style Sheets

A user agent style sheet is the built-in style sheet of the browser. Every browser has its default rules for how to display various HTML elements if no style is specified by the user or designer of the page. For instance, unvisited links are usually blue and underlined.

A user style sheet is a style sheet that the *user* has specified. Not all browsers support user style sheets, but they can be very useful, especially for users with certain types of disabilities. For instance, a dyslexic person can have a user style sheet that specifies certain fonts and colors to help them read more easily.

An author style sheet is what we normally refer to when we say "style sheet". It is the style sheet that the author of the document (or, more likely, the website's designer) has written and linked (or included).

Keyword

User agent style sheets are overridden by anything that you set in your own style sheet. They are just the rock bottom: in the absence of any style sheets provided by the page or by the user, the browser still has to render the content somehow, and the user agent style sheet just describes this.

Normal and Important Declarations

Normal declarations are just that: normal declarations.

The opposite is important declarations, which are declarations followed by an !important directive. A dyslexic user might, for instance, want all text to be displayed in Comic Sans MS if he finds that font easier to read. He could then have a user style sheet containing the following rule:

```
* {
  font-family: “Comic Sans MS” !important;
}
```

Important declarations in a user style sheet will trump everything else, which is logical. No matter what the designer specified, and no matter what the browser’s default font family is set to, everything will be displayed in Comic Sans MS. The default browser rendering will only apply if those declarations aren’t overridden by any rules in a user style sheet or an author style sheet, since the **user agent style sheet** has the lowest precedence.

To be honest, most designers don’t have to think too much about importance, since there’s nothing we can do about it. There is no way we could know if a user has a user style sheet defined that will override our CSS. If they do, they probably have a very good reason for doing so, anyway. Still, it’s good to know what importance is and how it may affect the presentation of our documents.

1.4.2 Specificity

Specificity is something every CSS author needs to understand and to think about. It can be thought of as a measure of how specific a rule’s selector is. A selector with low specificity may match many elements (like *, which matches every element in the document), while a selector with high specificity might only match a single element on a page (like #nav, which only matches the element with an id of nav).

The specificity of a selector can easily be calculated, as we shall see below. If two or more declarations conflict for a given element, and all the declarations have the same importance, then the one in the rule with the most specific selector will “win”.

Specificity has four components; let’s call them a, b, c and d. Component “a” is the most distinguishing, “d” the least.

- Component “a” is quite simple: it’s 1 for a declaration in a style attribute (also known as inline styling), otherwise it’s 0.
- Component “b” is the number of id selectors in the selector (those that begin with a #).
- Component “c” is the number of attribute selectors—including class selectors

— and pseudo-classes.

- Component "d" is the number of element types and pseudo-elements in the selector.

After a bit of counting, we can thus string those four components together to get the specificity for any rule. CSS declarations in a style attribute don't have a selector, so their specificity is always 1,0,0,0.

Let's look at a few examples — after this it should be quite clear how this works.

Selector	a	b	c	d	Specificity
h1				1	0,0,0,1
.foo			1		0,0,1,0
#bar		1			0,1,0,0
html>head+body ul#nav *.home a:link		1	2	5	0,1,2,5

Let's look at the last example in some more detail. a = 0 since it's a selector, not a declaration in a style attribute. There is one ID selector (#nav), so b = 1. There is one attribute selector (.home) and one pseudo-class (:link), so c = 2. There are five element types (<html>, <head>, <body>, <ul> and <a>), so d = 5.

The final specificity is thus 0,1,2,5.

Note: Combinators (like **>**, **+** and white space) do not affect a selector's specificity. The universal selector (*****) has no input on specificity, either.

Note #2: There is a huge difference in specificity between an id selector and an attribute selector that happens to refer to an id attribute. Although they match the same element, they have very different specificities. The specificity of #nav is 0,1,0,0 while the specificity of [id="nav"] is only 0,0,1,0.

Let's look at how this works in practice.

1. First of all, start by adding another paragraph to your HTML document.

```
<body>
  <h1>Heading</h1>
  <p>A paragraph of text.</p>

  <!-- Add this -->
  <p>A second paragraph of text.</p>
</body>
```

2. Next, add a rule to your stylesheet to make the paragraph text have a different color:

```
p {
  color: cyan;
}
```

3. Now save both files and reload the document in your browser; there should now be two paragraphs with cyan text.

4. Set an id on the first paragraph so you can target it easily with a CSS selector.

```
<body>
  <h1>Heading</h1>
  <!-- Add the id of "special" to this paragraph -->
  <p id="special">A paragraph of text.</p>
  <p>A second paragraph of text.</p>
</body>
```

5. Carry on by adding a rule for the special paragraph in your style sheet:

```
#special {
  background-color: red;
  color: yellow;
}
```

6. Finally, save both files and reload the document in your browser to see the now rather colorful result.

Let's look at the declarations that apply to the two paragraphs.

The first rule you added sets a text color of cyan for *all* paragraphs. The second rule sets a red background color and a yellow text color for the single element that has the id of special. Your first paragraph matches both of those rules; it is a paragraph and it has an id of special.

The red background isn't a problem, because it's only specified for #special. Both rules contain a declaration of the color property, though, which means there is a conflict. Both rules have the same importance — they are normal declarations in the author style sheet — so you have to look at the specificity of the two selectors.

The selector of the first rule is <p>, which has a specificity of 0,0,0,1 according to the rules above since it contains a single element type. The selector of the second rule is #special, which has a specificity of 0,1,0,0 since it consists of an id selector. 0,1,0,0 is much more specific than 0,0,0,1 so the color: yellow; declaration wins and you get yellow text on a red background.

1.4.3 Source Order

If two declarations affect the same element, have the same importance and the same specificity, the final distinguishing mark is the source order. The declaration that appears later in the style sheets will "win" over those that come before it.

If you have a single external style sheet, then the declarations at the end of the file will override those that occur earlier in the file if there's a conflict. The conflicting declarations could also occur in different style sheets.

In that case, the order in which the style sheets are linked, included or imported controls what declaration will be applied, so if you have two stylesheets linked in a document <head>, the one linked to further down will override the one linked to higher up. Let's look at a practical example of how this works.

1. Add a new rule to your style sheet at the very end of the file, like so:

```
p {
  background-color: yellow;
  color: black;
}
```

2. Save and reload the web page. you will now have *two* rules that match all paragraphs. They have the same importance and the same specificity (since the selector is the same), therefore the final mechanism for choosing which one wins will be the source order. The last rule specifies color:black and that will override color:cyan from the earlier rule.

ROLE MODEL

TIM BERNERS-LEE: INVENTOR OF THE WORLD WIDE WEB

Sir Tim Berners-Lee is a British computer scientist who invented what is undoubtedly one of the most revolutionary inventions of the 20th century—the World Wide Web (WWW). A qualified software engineer who was working at CERN when he came up with the idea of a global network system, Sir Tim is also credited for creating the world's first web browser and editor. He founded the World Wide Web Foundation and directs the World Wide Web Consortium (W3C). Both of his parents worked on the Ferranti Mark I, the first commercial computer, and thus it is not surprising that he too chose the field of computers. But what is surprising is the phenomenal impact his idea of a global network has had on the world of information and technology. An alumnus of the University of Oxford, he realized the need for a global communication network while working at CERN as the researchers from all over the world needed to share their data with each other. By the late 1980s he had drawn up a proposal for creating a global hypertext document system using the internet. A few more years of pioneering work in the field led to the birth of the World Wide Web making Berners-Lee one of the most significant inventors of the modern era.

Childhood and Early Life

He was born on June 8, 1955, as Timothy Berners-Lee to Mary Lee Woods and Conway Berners-Lee. He has three siblings. Both his parents worked on the first commercially-built computer, the Ferranti Mark I and thus Tim was fascinated by computers from a young age.

He received his primary education from Sheen Mount Primary School before moving on to London's independent Emanuel School where he studied from 1969 to 1973.

He enrolled at The Queen's College of the University of Oxford in 1973 and graduated in 1976 with a first-class degree in physics.

Career

- He was appointed as an engineer at the telecommunications company, Plessey in Poole after completing his studies. He remained there for two years, working on distributed transaction systems, message relays, and bar code technology.
- He left Plessey in 1978 and joined D. G. Nash Ltd. In this job he wrote typesetting software for intelligent printers and a multitasking operating system.
- In the late 1970s he began working as an independent consultant and worked for many companies, including CERN where he worked from June to December 1980 as a consultant software engineer.
- While at CERN he wrote a program called "Enquire" for his own personal use. It was a simple hypertext program which laid the conceptual foundation for the development of the World Wide Web in future.
- He started working at John Poole's Image Computer Systems, Ltd. in 1981. For the next three years he worked on the company's technical side which enabled him to gain experience in computer networking. His work included real time control firmware, graphics and communications software, and a generic macro language.
- He returned to CERN in 1984 after receiving a fellowship there. During the 1980s thousands of people were working at CERN and they needed to share information and data with each other. Much of the work was done by email and the scientists had to keep track of different things simultaneously. Tim realized that a simpler and more efficient method of data sharing had to be devised.
- In 1989, he wrote a proposal for a more effective communication system within the organization which eventually led to the conceptualization of the World Wide Web—an information sharing system that could be implemented throughout the world.
- The world's first ever website, Info.cern.ch, was built at CERN and put online on 6th August 1991, ushering in a new era in the field of communication and technology. The site provided information of what the World Wide Web was and how it could be used for information sharing.
- He established the World Wide Web Consortium (W3C) at the Massachusetts Institute of Technology's Laboratory for Computer Science in 1994. The W3C decided that its technologies should be royalty-free so that anyone could adopt them.
- He became a professor in the Computer Science Department at the University of Southampton, UK, in December 2004. There he worked on the Semantic Web.
- In 2006, he became the Co-Director of the Web Science Trust which was launched to analyze the World Wide Web and devise solutions to optimize

its usage and design. He also serves as the Director of the World Wide Web Foundation, started in 2009.

- Along with Professor Nigel Shadbolt, he is one of the key figures behind data.gov.uk, a UK Government project to make non-personal UK government data more accessible to the public.

Major Works

- His invention, the World Wide Web, is counted among the most significant inventions of the 20th century. The web revolutionized the world of information and technology and has opened up several new avenues.

Awards and Achievements

- He was presented with The Software System Award from the Association for Computing Machinery (ACM) in 1995.
- He was named as one of the 100 Most Important People of the 20th century by the Time Magazine in 1999.
- He was made the Commander of the Order of the British Empire (KBE) in the New Year Honours "for services to the global development of the Internet" in 2004.
- In 2013, he became one of five Internet and Web pioneers awarded the inaugural Queen Elizabeth Prize for Engineering.

Personal Life and Legacy

- He met Jane while studying physics at Oxford and married her soon after graduation in 1976. This marriage, however, ended in a divorce.
- While working for CERN he became acquainted with Nancy, an American software engineer. They so fell in love and tied the knot in 1990. This marriage too ended after some years.
- Currently he is married to Rosemary Leith who he wed in June 2014.

SUMMARY

- Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML.
- A web developer will write the code to apply to an entire group of web pages, a whole website, or even multiple web sites.
- An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension
- Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing
- The universal selector works like a wild card character, selecting all elements on a page. Every HTML page is built on content placed within HTML tags.
- An ID selector is declared using a hash, or pound symbol (#) preceding a string of characters. The string of characters is defined by the developer. This selector matches any HTML element that has an ID attribute with the same value as that of the selector, but minus the hash symbol.
- The class selector is the most useful of all CSS selectors. It's declared with a dot preceding a string of one or more characters. Just as is the case with an ID selector, this string of characters is defined by the developer.
- The descendant selector or, more accurately, the descendant combinator lets you combine two or more selectors so you can be more specific in your selection method.
- A selector that uses a general sibling combinator matches elements based on sibling relationships. That is to say, the selected elements are beside each other in the HTML.
- A selector that uses the adjacent sibling combinator uses the plus symbol (+), and is almost the same as the general sibling selector.
- Inheritance is the mechanism by which certain properties are passed on from a parent element down to its children, in the same fashion as genetics: if parents have blue eyes, their children will probably also have blue eyes.
- CSS an acronym of Cascading Style Sheets, so it is not a surprise that the cascade is an important concept. It is the mechanism that controls the end result when multiple, conflicting CSS declarations apply to the same element

KNOWLEDGE CHECK

1. **What does CSS stand for?**
   a. Creative Style Sheets
   b. Colorful Style Sheets
   c. Cascading Style Sheets
   d. Computer Style Sheets
2. **CSS padding property is used for?**
   a. Space
   b. Border
   c. Background color
   d. Margin
3. **The …….. selector is used to specify a style for a single, unique element**
   a. Id
   b. class
   c. text
   d. Bit
4. **Which of the following ways below is correct to write a CSS?**
   a. p {color:red; text-align:center};
   b. p {color:red; text-align:center}
   c. p {color:red; text-align:center;}
   d. p (color:red;text-align:center;)
5. **Which is the correct CSS syntax?**
   a. body {color: black}
   b. {body;color:black}
   c. {body:color=black(body}
   d. body:color=black
6. **Which of the following is the correct syntax for referring the external style sheet?**
   a. <style src = example.css>
   b. <style src = "example.css" >
   c. <stylesheet> example.css </stylesheet>
   d. <link rel="stylesheet" type="text/css" href="example.css">

7. The CSS property used to control the element's font-size is -

a. text-style
b. text-size
c. font-size
d. None of the above

8. The HTML attribute used to define the inline styles is -

a. style
b. styles
c. class
d. None of the above

9. The HTML attribute used to define the internal stylesheet is -

a. <style>
b. style
c. <link>
d. <script>

10. Which of the following syntax is correct in CSS to make each word of a sentence start with a capital letter?

a. text-style : capital;
b. transform : capitalize;
c. text-transform : capital;
d. text-transform : capitalize;

REVIEW QUESTIONS

1. What is **c**ascading **s**tyle **s**heets (CSS)?
2. What are the different variations of CSS ?
3. What are the advantages and disadvantage of CSS?
4. Discuss the CSS selector.
5. Explain universal selector.
6. What is inheritance and Cascade?

Check Your Result

1. (c)	2. (a)	3. (a)	4. (c)	5. (a)
6. (d)	7. (c)	8. (a)	9. (a)	10. (d)

REFERENCES

1. Andrew, Rachel. The New CSS Layout , A Book Apart, 2017.
2. Bartlett, Kynn. Sams Teach Yourself Cascading Style Sheets in 24 Hours, Second Edition, Sams, 2006.
3. Briggs, Owen et al. Cascading Style Sheets: Separating Content from Presentation, Second Edition, APress, 2004.
4. Brown, Tiffany. CSS Master, Sitepoint, 2021.
5. Budd, Andy, et al. CSS Mastery: Advanced Web Standards Solutions, Friends of ED, 2006.
6. Casciano, Chris. The CSS Pocket Guide, Peachpit Press, 2010.
7. Cederholm, Dan. CSS3 For Web Designers, A Book Apart, 2015.
8. Clarke, Andy. Hardboiled Web Design, Five Simple Steps, 2010.
9. Clarke, Andy. Transcending CSS: The Fine Art of Web Design, New Riders, 2006.
10. Collison, Simon. Professional CSS for Web Development: From Novice to Professional, APress, 2006.
11. Croft, Jeff, , et al. ProCSS Techniques, APress, 2006.
12. Debolt, Virginia. Integrated HTML and CSS, Sybex Inc, 2005.
13. Debolt, Virginia. Mastering Integrated HTML and CSS, Sybex Inc, 2007.
14. Freeman, Elisabeth and Freeman, Eric. Head First HTML with CSS & XHTML, O'Reilly & Associates, 2005.
15. Gasston, Peter. Book of CSS3, No Starch Press, 2011.
16. Gillenwater, Zoe, Mickley. Flexible Web Design: Creating Liquid and Elastic Layouts with CSS, New Riders Press, 2008.
17. Gillenwater, Zoe, Mickley. Stunning CSS3: A Project-based Guide to the Latest in CSS, New Riders Press, 2010.
18. Grannell, Craig. The Essential Guide to CSS and HTML Web Design, friends of ED, 2007.
19. Grannell, Craig. Web Designer's Reference, friends of ED, 2004.
20. Griffiths, Patrick. XHTML & CSS: A Web Standards Approach, New Riders, 2005.

CHAPTER

2 COLOR

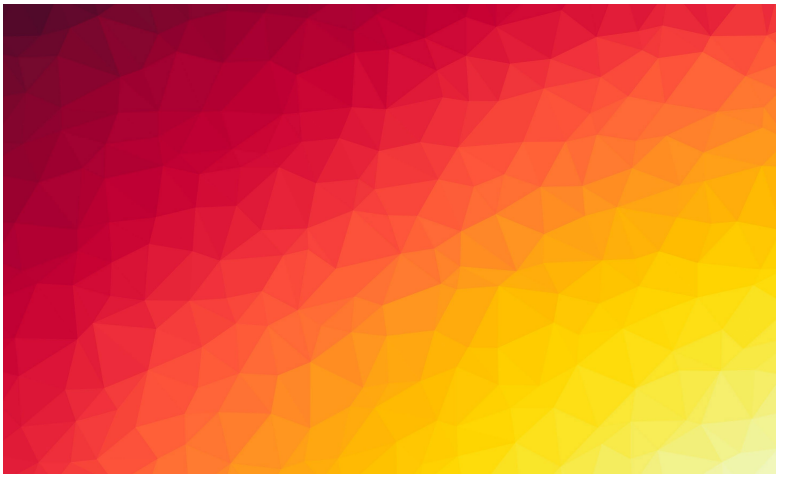
"Color does not add a pleasant quality to design - it reinforces it."

– Pierre Bonnard

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Explain fundamentals of CSS colors
2. Discuss about color properties
3. Use color in CSS
4. Identify different color values



INTRODUCTION

CSS uses the color values to specify the colors. Typically, the color property is used to set the background or font color on the page. You can also use this property for the border-color and other decorative elements.

The hexadecimal is a six-digit representation of the color. It is denoted by the # symbol followed by six characters range from 0 to F. The first two digits of the hexadecimal value represent stand for the color value,red (RR). The next two digits in the value sequence are for the color value, green (GG) followed by the last 2, which are for the blue (BB) color value.

The RGB format is the short form for red, green, and blue. We specify the color value using the rgb( ) property. The color value can be any integer value from 0 to. 255. It can also be a percentage.

When a color tag is used, there are a number of options for specifying color. One is simply to name the color, such as "black" or "blue." Another is to use a six digit hexadecimal code which specifies a particular color, such as #FFFFFF for white, although people should be aware that sometimes these codes render differently in different browsers when it comes to obscure colors. Yet another option is to use an RGB color, in which the levels of red, green, and blue are specified, as in 0,0,0 for black.

2.1 CSS COLORS: FUNDAMENTALS

Colors are widely used in CSS, whether for text color, background color, gradients, shadows, borders.

You can specify your color values in various formats. Following table lists all the possible formats –

Format	Syntax	Example
Hex Code	#RRGGBB	p{color:#FF0000;}
Short Hex Code	#RGB	p{color:#6A7;}
RGB %	rgb(rrr%,ggg%,bbb%)	p{color:rgb(50%,50%,50%);}
RGB Absolute	rgb(rrr,ggg,bbb)	p{color:rgb(0,0,255);}
keyword	aqua, black, etc.	p{color:teal;}

These formats are explained in more detail in the following sections –

2.1.1 CSS Colors - Hex Codes

A hexadecimal is a 6 digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).

A hexadecimal value can be taken from any graphics software like **Adobe Photoshop**, Jasc Paintshop Pro, or even using Advanced Paint Brush.

Each hexadecimal code will be preceded by a pound or hash sign '#'. Following are the examples to use Hexadecimal notation.

Color	Color HEX
	#000000
	#FF0000
	#00FF00
	#0000FF
	#FFFF00
	#00FFFF
	#FF00FF
	#C0C0C0
	#FFFFFF

Keyword

Adobe Photoshop is a raster graphics editor developed and published by Adobe Systems for macOS and Windows. Photoshop was created in 1988 by Thomas and John Knoll.

2.1.2 CSS Colors - Short Hex Codes

This is a shorter form of the six-digit notation. In this format, each digit is replicated to arrive at an equivalent six-digit value. For example: #6A7 becomes #66AA77.

A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Jasc Paintshop Pro, or even using Advanced Paint Brush.

Each hexadecimal code will be preceded by a pound or hash sign '#'. Following are the examples to use Hexadecimal notation.

Color	Color HEX
	#000
	#F00
	#0F0
	#0FF
	#FF0
	#0FF
	#F0F
	#FFF

Remember

To prevent unexpected behavior, such as in a <gradient>, the current CSS spec states that transparent should be calculated in the alpha-premultiplied color space. However, be aware that older browsers may treat it as black with an alpha value of 0.

2.1.3 CSS Colors - RGB Values

This color value is specified using the **rgb( )** property. This property takes three values, one each for red, green, and blue. The value can be an integer between 0 and 255 or a percentage.

Following is the example to show few colors using RGB values.

Remember

All the browsers does not support rgb() property of color so it is recommended not to use it.

Color	Color RGB
	rgb(0,0,0)
	rgb(255,0,0)
	rgb(0,255,0)
	rgb(0,0,255)
	rgb(255,255,0)
	rgb(0,255,255)
	rgb(255,0,255)
	rgb(192,192,192)
	rgb(255,255,255)

2.1.4 Color Model

A color model is a system that uses three primary colors to create a larger range of colors. There are different kinds of color models used for different purposes, and each has a slightly different range of colors they can produce. The whole range of colors that a specific type of color model produces is called a color space. All color results from how our eye processes light waves, but depending on the type of media, creating that color comes from different methods.

The RGB Color Model

There are two basic kinds of color models, additive and subtractive. Let's look at an additive color model first. The most common one is Red/Green/Blue, usually referred to as RGB. This color model uses light to create color, and it is used

for digital media. When you play a game on your smart phone or watch an action movie on TV, you are seeing color in an RGB color space. RGB is called an additive color model because when the three colors of light are shown in the same intensity at the same time, they produce white. If all the lights are out, they create black.

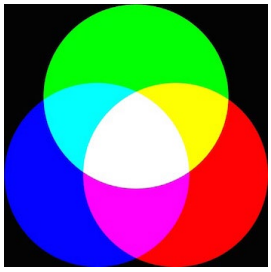


Figure 1: Image of a RGB Color Model. The middle is where all the colors at equal strength form white.

The CMYK Color Model

When printing color images, you can't use colored light, and that means images cannot be printed in RGB. That is where the other color model comes in. A subtractive color model adds pigment in the form of ink or dye that causes an absence of white. The most common subtractive color model is Cyan/Magenta/Yellow/Black, usually referred to as CMYK. It is what printers use, and you will sometimes also see it called process color because it is used in the four color printing process. To print a color image on paper, you have to use ink. Starting with the bright white paper surface, the colors are printed according to a pattern. The more color is applied, the more the white surface is masked. That is why it is called subtractive. But why the addition of black ink? Because when all the colors are mixed, they create a muddy brown. To get rich deep black, you have to use black ink.

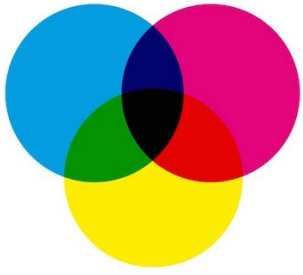


Figure 2: Image of a CMYK Color Model. In this image, the middle represents the addition of black ink for pure deep black.

The RYB Color Model

The RYB color model is a subtractive color model for mixing painting pigments. It is usually the first color model that we learn at an early age, perhaps in kindergarten. Starting with White paper, RYB color pigments when combined together yield Black, similar to the CMYK color model. Secondary colors that result from mixing primary pigments include the following: (1) the combination of Red and Yellow to yield Orange, (2) the combination of Yellow and Blue to yield Green, and (3) the combination of Blue and Red to yield Purple. The RYB color model is used in the arts and arts education. Figure 3 shows the RYB color model.

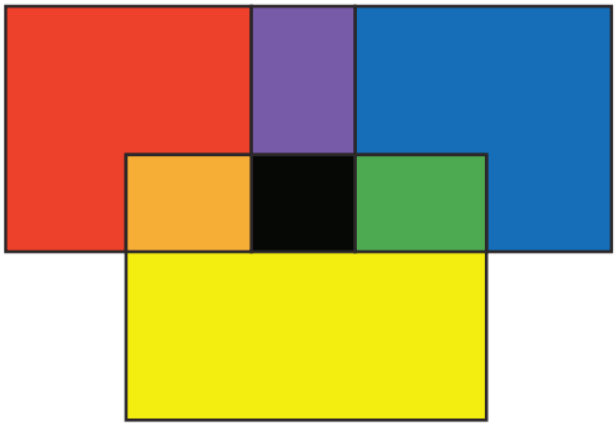


Figure 3: Illustration of the RYB color model.

Newton published his rainbow color map findings in 1704. During that period, production and reproduction of color images was performed with paint pigments on a White or cream canvas. Painters relied on the RYB color model for mixing and understanding colors. The theory of trichromatic (RGB) color vision had not been postulated. Although mirror displays existed, photographic, television, video, computer, and mobile display technologies with Red, Green, and Blue lights had not been developed. The RYB color model of the eighteenth century was the foundation of theories of color vision.

As a result, it was difficult for painters to understand how to incorporate Newton's observation (dispersion of White sunlight into a rainbow of colors) into their working knowledge of the RYB color theory. Therefore, Newton's observations were very misunderstood and frequently challenged by painters and other visual artists in the eighteenth century. It was not until the nineteenth-century developments of RGB color vision principles that the relationship between additive and subtractive color models, was understood.

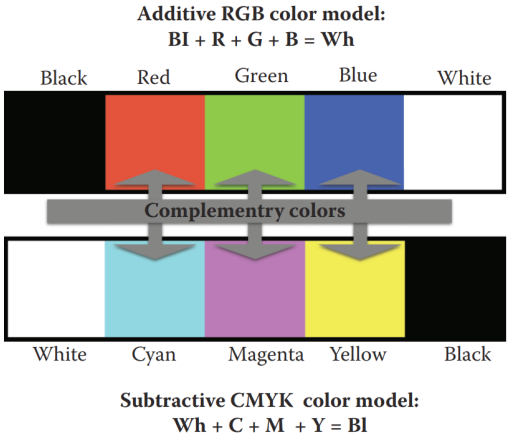


Figure 4: Diagram of the complementary relationship between the RGB color model and CMYK model.

Other Color Models

Both color models cover a range of color, but not all the colors are ones our eyes can see. Because CMYK uses inks rather than light, and more pigments dull the perception .of color, its color space is smaller than that of RGB

2.1.5 Browser Safe Colors

Here is the list of 216 colors which are supposed to be most safe and computer independent colors. These colors vary from hexa code 000000 to FFFFFF. These colors are safe to use because they ensure that all computers would display the colors correctly when running a 256 color palette –

000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF

336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

2.2 COLOR PROPERTIES

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

2.2.1 Foreground Color: The 'color' Property

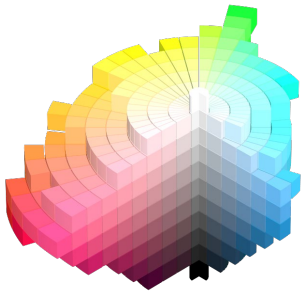
'color' properties allow to set colors for the text content of an element. The following table shows the necessary detail of the 'color ' property.

Name:	**color**
Value:	<color> | inherit
Initial:	depends on user agent
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	■ The computed value for basic color keywords, RGB hex values and extended color keywords is the equivalent triplet of numerical RGB values, e.g. six digit hex value or rgb(...) functional value, with an alpha value of 1. ■ The computed value of the keyword 'transparent' is the quadruplet of all zero numerical RGBA values, e.g. rgba(0,0,0,0). ■ For all other values, the computed value is the specified value.

Keyword

Foreground color usually refers to the text color, and the background color is the page color.

This property describes the **foreground color** of an element's text content. In addition it is used to provide a potential indirect value (currentColor) for any *other* properties that accept color values. If the 'currentColor' keyword is set on the 'color' property itself, it is treated as 'color: inherit'.



There are different ways to specify lime green:

```
em { color: lime }                /* color keyword */
em { color: rgb(0,255,0) }        /* RGB range 0-255  */
```

<color>

A <color> is either a keyword or a numerical specification.

2.2.2. Transparency: The 'opacity' Property

Remember

The color keywords all represent plain, solid colors, without transparency.

Opacity can be thought of as a postprocessing operation. Conceptually, after the element (including its descendants) is rendered into an RGBA offscreen image, the opacity setting specifies how to blend the offscreen rendering into the current composite rendering.

Name:	***opacity***
Value:	<alphavalue> | inherit
Initial:	1
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	The same as the specified value after clipping the <alphavalue> to the range [0.0,1.0].

<alphavalue>

Syntactically a <number>. The uniform opacity setting to be applied across an entire object. Any values outside the range 0.0 (fully transparent) to 1.0 (fully opaque) will be clamped to this range. If the object has children, then the effect is as

if the children were blended against the current background using a mask where the value of each pixel of the mask is <alphavalue>. For SVG, 'has children' is equivalent to being a container element [SVG11].

Since an element with opacity less than 1 is composited from a single offscreen image, content outside of it cannot be layered in z-order between pieces of content inside of it. For the same reason, implementations must create a new stacking context for any element with opacity less than 1. If an element with opacity less than 1 is not positioned, then it is painted on the same layer, within its parent stacking context, as positioned elements with stack level 0. If an element with opacity less than 1 is positioned, the 'z-index' property applies as described in [CSS21], except that if the used value is 'auto' then the element behaves exactly as if it were '0'.

Remember

Alpha indicates opacity and is determined in scale from 0 to 1.

2.3 USING COLOR IN CSS

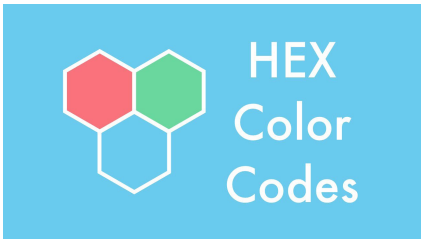
Using color in CSS (short for cascading stylesheets) is actually pretty easy. In this section we will show you how to use all the various color formats in CSS, including Hex color codes, HTML color names, and RGB and HSL values.

2.3.1 Using Hex Color Codes in CSS

Hex color codes are the most common method of adding color to an HTML element using CSS. In your stylesheet, you can use the CSS color property to change the default color of your website's text.

CSS

```
/* In your .css stylesheet */
body { color: #FF0000; }
```



A second option is to include the CSS styles right in the <head> of your HTML document using <style> tags, like below:

HTML

```
<!-- In your HTML document -->
<head>
  <style>
    body { color: #FF0000; }
  </style>
</head>
```

Easy, right? You can use the CSS color property with a Hex code on just about any HTML element, the <body> tag is only one example.

2.3.2 Using HTML Color Names in CSS

HTML color names are another way to style your content in CSS, and can often be easier to understand. You can use a color name in the same way as a Hex color code, setting it as the value for the CSS color property in your **stylesheet**.

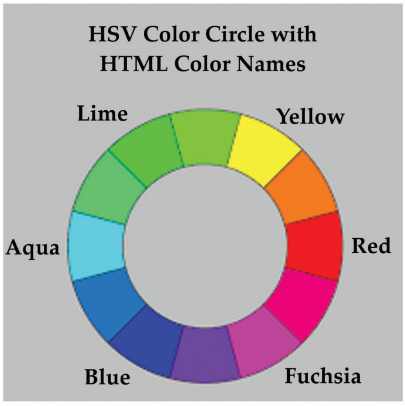
CSS

```
/* In your .css stylesheet */
body { color: red; }
```

Currently 140 color names are supported across all modern browsers.

Keyword

Style Sheet is a collection of style rules that that tells a browser how the various styles are to be applied to the HTML tags to present the document.



2.3.3 Using RGB Color Values in CSS

RGB, which stands for Red, Green and Blue, is a color system commonly found in many design applications and technologies, and more recently has become a go-to for web designers and programmers alike. In CSS, RGB colors can be used by wrapping the values in parentheses and prefixing them with a lowercase 'rgb'.

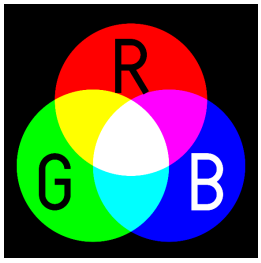
CSS

```
/* In your .css stylesheet */
body { color: rgb(255, 0, 0); }
```

One of the benefits of using RGB in your CSS is the ability to control the opacity in addition to the color. Adding an 'a' to the rgb() prefix enables a fourth value to be assigned which determines transparency of the color on a scale of 0 to 1. In this example, HTML page text would render at 50% opacity since 0.5 is halfway between 0 and 1.

CSS

```
/* In your .css stylesheet */
body { color: rgba(255, 0, 0, 0.5); }
```



2.3.4 Using HSL Color Values in CSS

You may have heard of HSL, which stands for Hue, Saturation and Lightness, another color system used in many applications. Hue is measured in degrees from 0 – 360, while Saturation and Lightness use a scale of 0% – 100%. In CSS, HSL can be easily implemented following a syntax similar to RGB but prefixed instead with 'hsl'.

CSS

```
/* In your .css stylesheet */
body { color: hsl(0, 100%, 50%); }
```

Did You Know?

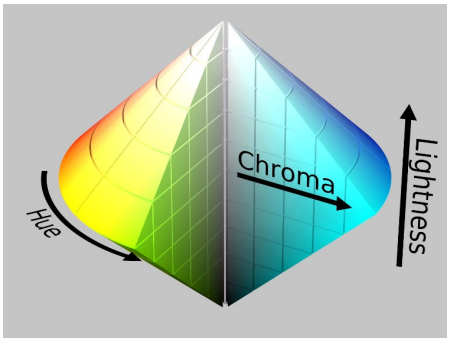
The CIE 1931 color space standard defines both the CIE RGB space, which is an RGB color space with monochromatic primaries, and the CIE XYZ color space, which works like an RGB color space except that it has non-physical primaries that cannot be said to be red, green, and blue.

Likewise, HSL also supports an alpha channel to control the transparency of the color. Its use is identical to RGB, with a fourth value between 0 and 1 enabled when using the 'hsla' prefix.

CSS

```
/* In your .css stylesheet */
body { color: hsla(0, 100%, 50%, 0.5); }
```

Note, rgba(), hsl() and hsla() are relatively new additions to CSS, and are not supported by some older browsers. Depending on your situation, you may want to select different methods for manipulating the opacity of your colors.



2.4 COLOR VALUES

In HTML, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values:

Same as color name "Tomato":

rgb(255, 99, 71)

#ff6347

hsl(9, 100%, 64%)

Same as color name "Tomato", but 50% transparent:

rgba(255, 99, 71, 0.5)

hsla(9, 100%, 64%, 0.5)

Example:

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
<h1 style="background-color:#ff6347;">...</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>
<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

2.4.1 RGB Value

In HTML, a color can be specified as an RGB value, using this formula:

rgb(*red, green, blue*)

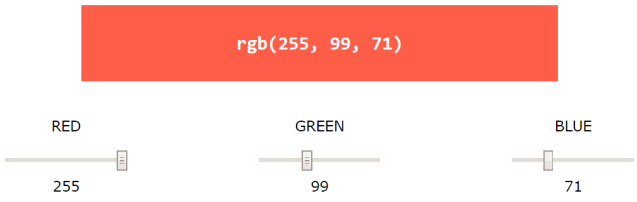
Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.

To display the color black, all color parameters must be set to 0, like this: rgb(0, 0, 0).

To display the color white, all color parameters must be set to 255, like this: rgb(255, 255, 255).

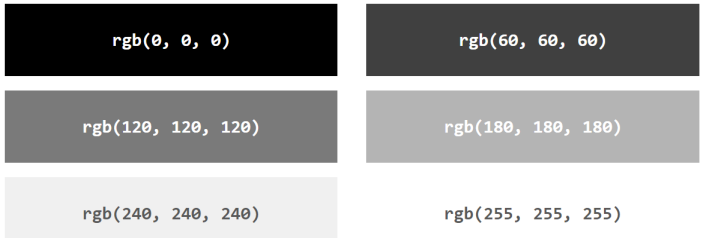
Experiment by mixing the RGB values below:



Example:

Shades of gray are often defined using equal values for all the 3 light sources:

Example:



2.4.2 HEX Value

In HTML, a color can be specified using a hexadecimal value in the form:

#rrggbb

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

#ff0000 is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).

Example:

Shades of gray are often defined using equal values for all the 3 light sources:

Example:

2.4.3 HSL Value

In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form:

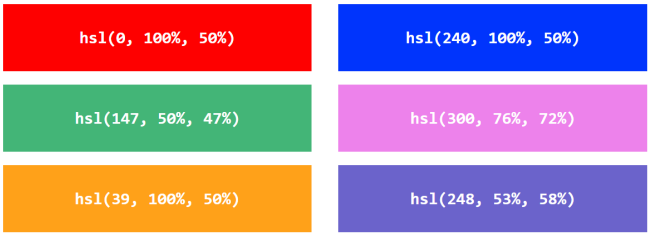
hsl(*hue, saturation, lightness*)

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white

Example:



Saturation

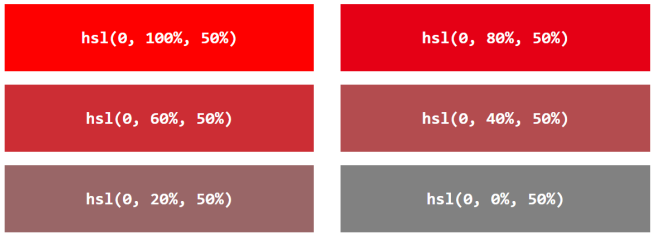
Saturation can be describe as the intensity of a color.

100% is pure color, no shades of gray

50% is 50% gray, but you can still see the color.

.is completely gray, you can no longer see the color 0%

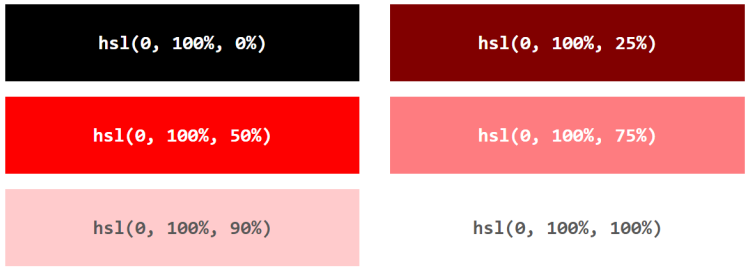
Example:



Lightness

The lightness of a color can be described as how much light you want to give the color, where 0% means no light (black), 50% means 50% light (neither dark nor light) 100% means full lightness (white).

Example:



Keyword

Hue is one of the main properties of a color, defined technically, as "the degree to which a stimulus can be described as similar to or different from stimuli that are described as red, green, blue, and yellow".

Shades of gray are often defined by setting the **hue** and saturation to 0, and adjust the lightness from 0% to 100% to get darker/lighter shades:

Example:

2.4.4 RGBA Value

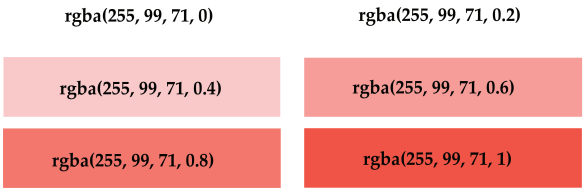
RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with:

rgba(*red, green, blue, alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Example:



Remember

If RGBA values are not supported by a user agent, they should be treated like unrecognized values per the CSS forward compatibility parsing rules. RGBA values must not be treated as simply an RGB value with the opacity ignored.

2.4.5 HSLA Value

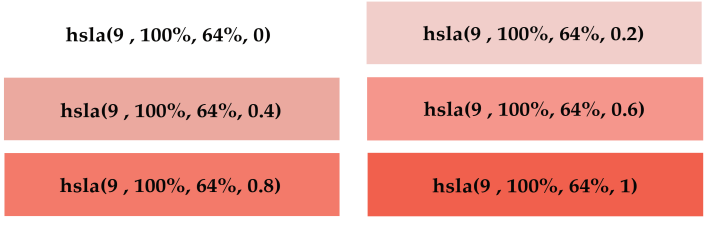
HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with:

hsla(*hue, saturation, lightness, alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Example:



ROLE MODEL

JACOB LE BLON: INVENTOR OF THREE- AND FOUR-COLOR PROCESS USING AN RYBK COLOR MODEL

Jacob Christoph Le Blon, or Jakob Christoffel Le Blon, (2 May 1667 – 16 May 1741) was a painter and engraver from Frankfurt who invented the system of three- and four-colour printing, using an RYBK color model similar to the modern CMYK system. He used the mezzotint method to engrave three or four metal plates (one each per printing ink) to make prints with a wide range of colours. His methods helped form the foundation for modern colour printing.

Biography

On his father's side Le Blon descended from Huguenots fleeing France in 1576, having settled in Frankfurt. His grandfather Christof Le Blon married Susanna Barbara Merian daughter of the artist and engraver Matthäus Merian (1593–1650). Le Blon is reported to have received training as a young man from the Swiss painter and engraver Conrad Ferdinand Meyer (1618–1689) in Zurich but there is no documentary evidence. It is generally agreed that Le Blon had an extended stay sometime between 1696 and 1702 in Rome where he is reported to have studied art under the painter Carlo Maratta (1625–1713). There he became acquainted with the Dutch painter and engraver Bonaventura van Overbeek who created an extensive work of views of the antiquities of Rome, published posthumously in 1708 and containing a portrait of Overbeek attributed to Le Blon. Encouraged by van Overbeek, Le Blon moved to Amsterdam, presumably in 1702, where he worked as a miniature painter and engraver. In 1705 he married Gerarda Vloet with whom he had two sons that appear to have died in infancy.

In 1707 Le Blon issued a short publication in Dutch on the forms of the human body. In 1708 and 1709 he is known to have made colorant mixing experiments in Amsterdam and in 1710 he made his first color prints with yellow, red, and blue plates. While in Amsterdam he became acquainted

with Arnold Houbraken, who quoted him as a source of information on German painters for his *Schouburg*, later published after Houbraken's death in 1718. Le Blon's wife died in 1716 and in 1717 he moved to London where he received royal patents for the three-color printing process and a three-color tapestry weaving process. The tapestry process involved using white, yellow, red, blue, and black fibers to create images. The printing process involved using three different intaglio plates, inked in different colours. In 1722 he published a small book, Coloritto, in French and English. In it he stated that "Painting can represent all visible objects with three colors, yellow, red, and blue" (pp. 6, 7). During his stay in England he produced several dozen of three- and four-colored images in multiple copies that initially sold well in England and on the continent. In the long run his enterprise did not succeed, however, and Le Blon left England in 1735, moving to Paris where he continued producing prints by his method. During his last years several sequences of prints were produced and sold showing the different steps of his printing process, such as a portrait of the French Cardinal de Fleury. In 1740 he began work on a collection of anatomical prints for which he had a solid list of subscribers. When he died in 1741 in Paris he left a 4½ year-old daughter, Margueritte, as sole heir. A detailed description of Le Blon's work was published in 1756 by Antoine Gautier de Montdorge who befriended him during his final years in Paris.

Le Blon's method required experience in deconstructing a colored image into its presumed primary chromatic components and understanding the effects of superimposing printing inks in certain areas, for which extensive trial and error work was required. Le Blon's process was practiced in France after his death and progressed in the early-mid-19th century into chromolithography. What was required, however, is a methodology to break images objectively into color components which became possible with the invention of color photography in the second half of the 19th century and the invention of half-tone printing in the late 19th century.

SUMMARY

- CSS uses color values to specify a color. Typically, these are used to CSS uses the color values to specify the colors. Typically, the color property is used to set the background or font color on the page. You can also use this property for the border-color and other decorative elements.
- A hexadecimal is a 6 digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).
- A color model is a system that uses three primary colors to create a larger range of colors. There are different kinds of color models used for different purposes, and each has a slightly different range of colors they can produce.
- The RYB color model is a subtractive color model for mixing painting pigments. It is usually the first color model that we learn at an early age, perhaps in kindergarten.
- Hex color codes are the most common method of adding color to an HTML element using CSS. In your stylesheet, you can use the CSS color property to change the default color of your website's text.
- RGB, which stands for Red, Green and Blue, is a color system commonly found in many design applications and technologies, and more recently has become a go-to for web designers and programmers alike.
- The lightness of a color can be described as how much light you want to give the color, where 0% means no light (black), 50% means 50% light (neither dark nor light) 100% means full lightness (white).
- RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.
- HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.
- alpha channel - which specifies the opacity for a color.

KNOWLEDGE CHECK

1. **Which of the following is correct about Hex Code format of CSS colors?**
   a. The first two digits(RR) represent a red value.
   b. The next two are a green value(GG).
   c. The last are the blue value(BB).
   d. All of the above.
2. **Which of the following CSS3 Color Feature can be used as a macro for whatever the current color is?**
   a. CurrentColor keyword
   b. HSLa Color
   c. HSL Color
   d. RGB Color
3. **Which of the following CSS3 Color Feature like RGB color but adds an alpha channel value to specify the opacity of the color?**
   a. RGB
   b. RGBa
   c. RGBaplha
   d. AlphaRGB
4. **Which of the following Color Format is a CSS's six-digit hexadecimal format is the same as color defined in (X)HTML?**
   a. 6-Hex Color
   b. 3-Hex Color
   c. RGBS
   d. RGBa
5. **Which of the following Color Format is a CSS2 introduced named color keywords which allows Web page colors to be matched to an operating system's color use?**
   a. HSLa Color
   b. Commonly defined named colors
   c. System Color Names
   d. Specificationdefined named colors
6. **Which of the following Color Format are a 17 defined colors under CSS 2.1?**
   a. HSLa Color
   b. Commonly defined named colors

c. System Color Names

d. Specificationdefined named colors

7. Which of the following Color Format is a CSS colors can also be defined using the keyword rgb, followed by three numbers between 0 and 255, contained in parentheses and separated by commas, with no spaces between them?

a. RGB Color

b. RGBa Color

c. HSL Color

d. HSLa Color

8. Which of the following Color Format is a CSS3 HSL value with a fourth value to set the alpha channel value for the color to define the opacity of the element?

a. RGB Color

b. RGBa Color

c. HSL Color

d. HSLa Color

9. The property in CSS used to change the text color of an element is -

a. bgcolor

b. color

c. background-color

d. All of the above

10. Which of the following is the correct syntax to make the background-color of all paragraph elements to yellow?

a. p {background-color : yellow;}

b. p {background-color : #yellow;}

c. all {background-color : yellow;}

d. all p {background-color : #yellow;}

REVIEW QUESTIONS

1. What are Hex Codes? Explain with suitable examples.
2. Compare RGB values with Hexadecimal color codes?
3. Which CSS property is used to change the text color of an element?
4. Which property is used to change the background color?
5. What does HSL stands for?
6. What is the use of CSS Opacity?

Check Your Result

1. (d)	2. (a)	3. (b)	4. (a)	5. (c)
6. (d)	7. (a)	8. (d)	9. (b)	10. (a)

REFERENCES

1. About Dynamic Properties. Msdn.microsoft.com. Archived from the original on 2017-10-14. Retrieved 2009-06-20.
2. Antti, Hiljá. "OOCSS, ACSS, BEM, SMACSS: what are they? What should I use?". clubmate.fi. Hiljá. Archived from the original on 2 June 2015. Retrieved 2 June 2015.
3. Cederholm, Dan; Ethan Marcotte (2009). Handcrafted CSS: More Bulletproof Web Design. New Riders. p. 114. ISBN 978-0-321-64338-4. Archived from the original on 20 December 2012. Retrieved 19 June 2010.
4. CSS Multi-column Layout Module. World Wide Web Consortium. Archived from the original on 2011-04-29. Retrieved 2011-05-01.
5. Selectors Level 4 – Determining the Subject of a Selector. W3.org. Archived from the original on 2013-08-17. Retrieved 2013-08-13.
6. Snook, Jonathan (October 2010). "Why we don't have a parent selector". snook.ca. Archived from the original on 2012-01-18. Retrieved 2012-01-26.

CHAPTER

3

BOXES: SIZE AND BORDER

"If there's one thing you learn by working on a lot of different Web sites, it's that almost any design idea--no matter how appallingly bad--can be made usable in the right circumstances, with enough effort."

– Steve Krug

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Explain the box sizing
2. Describe the box borders

INTRODUCTION

The box-sizing CSS property sets how the total width and height of an element is calculated.

The width and height you assign to an element is applied only to the element's content box. If the element has any border or padding, this is then added to the width and height to arrive at the size of the box that's rendered on the screen. This means that when you set width and height, you have to adjust the value you give to allow for any border or padding that may be added. For example, if you have four boxes with width: 25%;, if any has left or right padding or a left or right border, they will not by default fit on one line within the constraints of the parent container.

The box-sizing property can be used to adjust this behavior:

- content-box gives you the default CSS box-sizing behavior. If you set an element's width to 100 pixels, then the element's content box will be 100 pixels wide, and the width of any border or padding will be added to the final rendered width, making the element wider than 100px.
- border-box tells the browser to account for any border and padding in the values you specify for an element's width and height. If you set an element's width to 100 pixels, that 100 pixels will include any border or padding you added, and the content box will shrink to absorb that extra width. This typically makes it much easier to size elements. box-sizing: border-box is the default styling that browsers use for the <table>, <select>, and <button> elements, and for <input> elements whose type is radio, checkbox, reset, button, submit, color, or search.

3.1 BOX SIZING

Box sizing property is using to change the height and width of element.

Since css2, the box property has worked like as shown below –

width + padding + border = actual visible/rendered width of an element's box

height + padding + border = actual visible/rendered height of an element's box

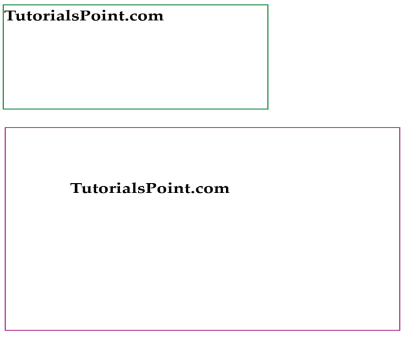
Means when you set the height and width, it appears little bit bigger then given size cause element boarder and padding added to the element height and width.

3.1.1 CSS2 sizing property

```
<html>
  <head>
    <style>
      .div1 {
        width: 200px;
        height: 100px;
        border: 1px solid green;
```

```
        }
        .div2 {
            width: 200px;
            height: 100px;
            padding: 50px;
            border: 1px solid pink;
        }
    </style>
</head>

<body>
    <div class = "div1">TutorialsPoint.com</div><br />
    <div class = "div2">TutorialsPoint.com</div>
</body>
</html>
```

It will produce the following result –



Above image is having same width and **height** of two elements but giving result is different, cause second one is .included padding property

3.1.2 CSS3 box sizing property

```
<html>
    <head>
        <style>
```

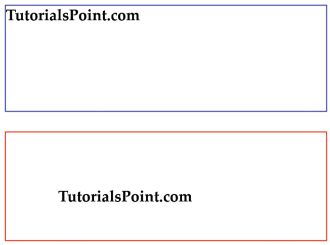
Keyword

Height is the measure of vertical distance, either how "tall" something or someone is, or how "high" the position is.

```
} div1.
            width: 300px;
            height: 100px;
            border: 1px solid blue;
            box-sizing: border-box;
         }
         .div2 {
            width: 300px;
            height: 100px;
            padding: 50px;
            border: 1px solid red;
            box-sizing: border-box;
         }
      </style>
   </head>
   <body>
      <div class = “div1”>TutorialsPoint.com</div><br />
      <div class = “div2”>TutorialsPoint.com</div>
   </body>
</html>
```

Above sample is having same height and width with box-sizing: border-box. here result is shown below.

It will produce the following result –



Above elements are having same height and width with box-sizing: border-box so result is always same for both elements as shown above.

3.1.3 Without the CSS box-sizing Property

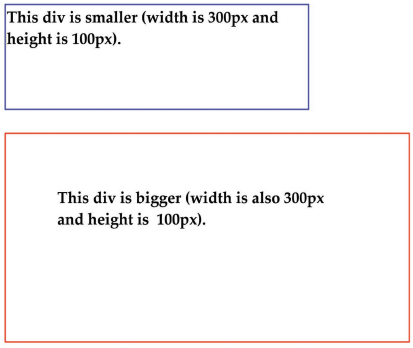
By default, the width and height of an element is calculated like this:

width + padding + border = actual width of an element

height + padding + border = actual height of an element

This means: When you set the width/height of an element, the element often appears bigger than you have set (because the element's border and padding are added to the element's specified width/height).

The following illustration shows two elements with the same specified width and height:



The two elements above end up with different sizes in the result (because div2 has a padding specified):

Example

```
.div1 {
        width: 300px;
        height: 100px;
        border: 1px solid blue;
    }
    .div2 {
        width: 300px;
        height: 100px;
        padding: 50px;
;border: 1px solid red
    }
```

Run

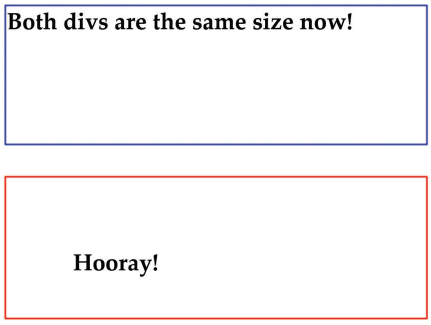
```
<!DOCTYPE html>
<html>
<head>
<style>
.div1 {
    width: 300px;
    height: 100px;
    border: 1px solid blue;
}

.div2 {
    width: 300px;
    height: 100px;
    padding: 50px;
    border: 1px solid red;
}
</style>
</head>
<body>
<div class="div1">This div is smaller (width is 300px and height is 100px).</div>
<br>
<div class="div2">This div is bigger (width is also 300px and height is 100px).</div>
</body>
</html>
```

The box-sizing property solves this problem.

With the CSS box-sizing Property

The box-sizing property allows us to include the padding and border in an element's total width and height.

If you set box-sizing: border-box; on an element padding and border are included in the width and height:



Here is the same example as above, with box-sizing: border-box; added to both elements:

Example

```
.div1 {
    width: 300px;
    height: 100px;
    border: 1px solid blue;
    box-sizing: border-box;
}
.div2 {
    width: 300px;
    height: 100px;
    padding: 50px;
    border: 1px solid red;
    box-sizing: border-box;
}
```

Since the result of using the box-sizing: border-box; is so much better, many developers want all elements on their pages to work this way.

The code below ensures that all elements are sized in this more intuitive way. Many browsers already use box-sizing: border-box; for many form elements (but not all - which is why inputs and text areas look different at width: 100%;).

Applying this to all elements is safe and wise:

Did You Know?

CSS was first proposed by Håkon Wium Lie on October 10, 1994. At the time, Lie was working with Tim Berners-Lee at CERN. Several other style sheet languages for the web were proposed around the same time, and discussions on public mailing lists and inside World Wide Web Consortium resulted in the first W3C CSS Recommendation (CSS1) being released in 1996.

Example

```
* {
    box-sizing: border-box;
}
```

3.2 BORDERS

CSS Border Properties

The CSS border properties allow you to specify the style, width, and color of an element's border.

I have borders on all sides.

I have a red bottom border.

I have rounded borders.

I have a blue left border.

3.2.1 Border Style

The border-style property specifies what kind of border to display.

The following values are allowed:

dotted - Defines a dotted border

dashed - Defines a dashed border

solid - Defines a solid border

double - Defines a double border

groove - Defines a 3D grooved border. The effect depends on the border-color value

ridge - Defines a 3D ridged border. The effect depends on the border-color value

inset - Defines a 3D inset border. The effect depends on the border-color value

outset - Defines a 3D outset border. The effect depends on the border-color value

none - Defines no border

hidden - Defines a hidden border

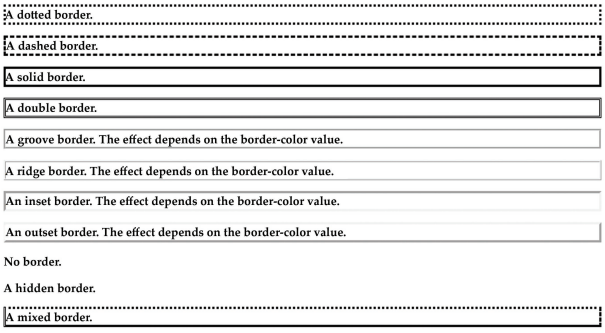
The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
```

```
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

Result:



Remember

None of the OTHER CSS border properties described below will have ANY effect unless the border-style property is set!

Border Width

The border-width property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.

The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

```
p.one {
    border-style: solid;
    border-width: 5px;
}
p.two {
    border-style: solid;
    border-width: medium;
}
p.three {
    border-style: solid;
    border-width: 2px 10px 4px 20px;
}
```

Example

Set the width of the borders to medium:

```
div {border-width: medium;}
```

Example

Set the width of the borders to thick:

```
div {border-width: thick;}
```

Example

Set the width of the borders to 1px:

```
div {border-width: 1px;}
```

Example

Set the width of the borders to 15px:

```
div {border-width: 15px;}
```

Example

Set the width of the top and bottom borders to 10px, and the width of the left and right borders to 1px:

```
div {border-width: 10px 1px;}
```

Border Color

The border-color property is used to set the color of the four borders.

The color can be set by:

name - specify a color name, like “red”

Hex - specify a hex value, like “#ff0000”

RGB - specify a RGB value, like “rgb(255,0,0)”

transparent

The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border).

If border-color is not set, it inherits the color of the element.

Example

```
p.one {
    border-style: solid;
    border-color: red;
}

p.two {
    border-style: solid;
    border-color: green;
}

p.three {
    border-style: solid;
    border-color: red green blue yellow;
}
```

Examples

Set a color for the border with a HEX value:

```
div {border-color: #92a8d1;}
```

Example

Set a color for the border with an RGB value:

```
{;(div {border-color: rgb(201, 76, 76
```

Example

Set a color for the border with an RGBA value:

```
div {border-color: rgba(201, 76, 76, 0.3);}
```

Example

Set a color for the border with a HSL value:

```
div {border-color: hsl(89, 43%, 51%);}
```

Keyword

CSS is a style sheet language used for describing the presentation of a document written in a markup language like HTML.

Example

Set a color for the border with a HSLA value:

```
div {border-color: hsla(89, 43%, 51%, 0.3);}
```

Example

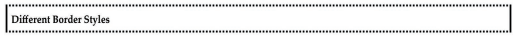
Set a different border-color for each side of an element:

```
div.ex1 {border-color: #0000ff;}
div.ex2 {border-color: #ff0000 #0000ff;}
div.ex3 {border-color: #ff0000 #00ff00 #0000ff;}
div.ex4 {border-color: #ff0000 #00ff00 #0000ff rgb(250,0,255);}
```

3.2.2 Border - Individual Sides

From the examples above you have seen that it is possible to specify a different border for each side.

In **CSS**, there are also properties for specifying each of the borders (top, right, bottom, and left):



Example

```
p {
    border-top-style: dotted;
    border-right-style: solid;
    border-bottom-style: dotted;
    border-left-style: solid;
}
```

The example above gives the same result as this:

Example

```
p {
    border-style: dotted solid;
}
```

So, here is how it works:

If the border-style property has four values:

border-style: dotted solid double dashed;

top border is dotted

right border is solid

bottom border is double

left border is dashed

If the border-style property has three values:

border-style: dotted solid double;

top border is dotted

right and left borders are solid

bottom border is double

If the border-style property has two values:

border-style: dotted solid;

top and bottom borders are dotted

right and left borders are solid

If the border-style property has one value:

border-style: dotted;

all four borders are dotted

The border-style property is used in the example above. However, it also works with border-width and border-color.

Example

A dashed border:

```
div {border-style: dashed;}
```

Example

A solid border:

```
div {border-style: solid;}
```

Example

A double border:

```
div {border-style: double;}
```

Example

A groove border:

```
div {
    border-style: groove;
    border-color: coral;
    border-width: 7px;
}
```

Example

A ridge border:

```
div {
    border-style: ridge;
    border-color: coral;
    border-width: 7px;
}
```

Example

An inset border:

```
div {
    border-style: inset;
    border-color: coral;
    border-width: 7px;
}
```

Example

An outset border:

```
div {
    border-style: outset;
    border-color: coral;
    border-width: 7px;
}
```

Set different borders on each side of an element:

p.one {border-style: dotted solid dashed double;}

p.two {border-style: dotted solid dashed;}

p.three {border-style: dotted solid;}

p.four {border-style: dotted;}

3.2.3 Border - Shorthand Property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The border property is a shorthand property for the following individual border properties:

- border-width
- border-style (required)
- border-color

Example

```
p {
    border: 5px solid red;
}
```

Result:

You can also specify all the individual border properties for just one side:

Left Border

```
p {
    border-left: 6px solid red;
    background-color: lightgrey;
}
```

Result:

Bottom Border

```
p {
    border-bottom: 6px solid red;
    background-color: lightgrey;
}
```

Result:

Rounded Borders

The border-radius property is used to add rounded borders to an element:

Normal border

Round border

Rounder border

Roundest border

Example

```
p {
    border: 2px solid red;
    border-radius: 5px;
}
```

Border Images

With the CSS border-image property, you can set an image to be used as the border around an element.

3.2.4 Border-Image Property

The CSS border-image property allows you to specify an image to be used instead of the normal border around an element.

The property has three parts:

- The image to use as the border
- Where to slice the image
- Define whether the middle sections should be repeated or stretched

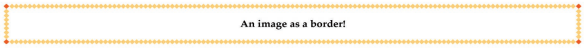
We will use the following image (called "border.png"):

The border-image property takes the image and slices it into nine sections, like a **tic-tac-toe** board. It then places the corners at the corners, and the middle sections are repeated or stretched as you specify.

Keyword

URL is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

Note: For border-image to work, the element also needs the border property set!
Here, the middle sections of the image are repeated to create the border:

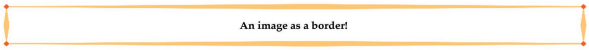


Here is the code:

Example

```
#borderimg {
    border: 10px solid transparent;
    padding: 15px;
    border-image: url(border.png) 30 round;
}
```

Here, the middle sections of the image are stretched to create the border:



Here is the code:

Example

```
#borderimg {
    border: 10px solid transparent;
    padding: 15px;
    border-image: url(border.png) 30 stretch;
}
```

URL is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

border-image - Different Slice Values

:ifferent slice values completely changes the look of the border
:Example 1

;border-image: url(border.png) 50 round
:Example 2

Example 3:

:Here is the code

Example

} borderimg1#
;border: 10px solid transparent
;padding: 15px
;border-image: url(border.png) 50 round
{
} borderimg2#
border: 10px solid transparent;

;padding: 15px
;border-image: url(border.png) 20% round
{
} borderimg3#
;border: 10px solid transparent
;padding: 15px
border-image: url(border.png) 30% round;

{

3.2.5 Border-Right-Color Property

This CSS explains how to use the CSS property called **border-right-color** with syntax and examples.

Description

The CSS border-right-color property defines the color of the right border of a box.

Syntax

The syntax for the border-right-color CSS property is:

;border-right-color: value

Parameters or Arguments

value

:The color of the right border. It can be one of the following

Value	Description
#RRGGBB	Hexadecimal representation of the border-right-color div { border-right-color: #000000; }
rgb()	RGB representation of the border-right-color div { border-right-color: rgb(0,0,0); }
name	Name of the border-right-color (ie: red, blue, black, white) div { border-right-color: black; }
transparent	Border is not displayed but still takes up space on the page div { border-right-color: transparent; }
inherit	Element will inherit the border-right-color from its parent element div { border-right-color: inherit; }

Note

- Be sure to set the border-right-style property as well to make sure that your right border appears.
- See also border-right-style, border-right-width, and border-right properties.
- Need to convert your color value to a different representation? Try this online tool to convert your color value between hexadecimal and RGB.

Example

We will discuss the border-right-color property below, exploring examples of how to use this property in CSS.

Using Hexadecimal

Let's look at a CSS border-right-color example where we have provided a hexadecimal value for the border-right-color property.

div { border-right-color: #0000FF; border-right-style: solid; }

In this border-right-color example, we have provided a hexadecimal value of #0000FF which would display a blue right border for the tag. Be sure to apply a border-right-style so that the border appears.

Using RGB

We can also provide the border-right-color value using rgb.

div {border-right-color: rgb(0,0,255); border-right-style: solid; }

In this border-right-color example, rgb(0,0,255) would also display the right border for the tag in blue.

Using Color Name

Let's look at a CSS border-right-color example where we have provided the value as a named color.

div {border-right-color: blue; border-right-style: solid;}

In this CSS border-right-color example, we have provided the name "blue" which would also set the right border to blue.

Keyword

Parameter is any characteristic that can help in defining or classifying a particular system

3.2.6 Border-Right-Style Property

His CSS explains how to use the CSS property called **border-right-style** with syntax and examples.

Description

The CSS border-right-style property defines the line style of the right border of a box.

Syntax

The syntax for the border-right-style CSS property is:

border-right-style: value;

Parameters or Arguments

value

The line style to use for the right border. It can be one of the following:

Value	Description
none	No border (This is the default) div { border-right-style: none; }
solid	Single, straight, solid line div { border-right-style: solid; }
dotted	Series of dots div { border-right-style: dotted; }
dashed	Series of short dashes div { border-right-style: dashed; }
double	Two straight lines that total the pixel amount defined by border-right-width div { border-right-style: double; }
groove	Carved effectdiv { border-right-style: groove; }
ridge	3D appearance where border looks like it is "coming out" (opposite of groove) div { border-right-style: ridge; }
inset	Embedded appearance div { border-right-style: inset; }
outset	Embossed appearance (opposite of inset) div { border-right-style: outset; }
hidden	Border is hidden div { border-right-style: hidden; }
inherit	Element will inherit the border-right-style from its parent element div { border-right-style: inherit; }

Note

- Since the default value for CSS border-right-style is *none*, you must set a CSS border-right-style value for your right border to appear.
- See also the border-right-color, border-right-width, and border-right properties.

Example

We will discuss the border-right-style property below, exploring examples of how to use this property in CSS.

Let's look at an example where we set a solid right border.

div { border-right-style: solid; }

In this CSS border-right-style example, we have set the style of the right border to solid.

Next, let's set our right border style and the right border color.

div { border-right-style: dashed; border-right-color: blue; }

In this border-right-style example, we have set the right border style to a blue dashed line.

3.2.7 Border-Right-Width Property

This CSS explains how to use the CSS property called border-right-width with syntax and examples.

Description

The CSS border-right-width property defines the width of the right border of a box.

Syntax

The syntax for the border-right-width CSS property is:

border-right-width: value;

Parameters or Arguments

value

The width of the right border of a box. It can be one of the following:

Value	Description
fixed	Fixed value expressed in px, em, ... div { border-right-width: 5px; }
thin	Thin border width, which might be 1px or 2px depending on the browser div { border-right-width: thin; }
medium	Medium border width, which might be 3px or 4px depending on the browser div { border-right-width: medium; }

thick	Thick border width, which might be 5px or 6px depending on the browser div { border-right-width: thick; }

Note

- Be sure to set the border-right-style property as well to make sure that your right border appears.
- See also the border-right-color, border-right-style, and border-right properties.

Example

We will discuss the border-right-width property below, exploring examples of how to use this property in CSS.

Let's look at an example where we set the right border to a fixed width.

div { border-right-width: 0.2em; border-right-style: solid; }

In this CSS border-right-style example, we have set the width of the right border to 0.2em. Be sure to apply a border-right-style so that the border appears.

Next, let's try using one of the width keywords (thin, medium, thick) to set our right border width.

div { border-right-width: medium; border-right-style: solid; }

In this border-right-style example, we have set the right border to a medium line.

3.2.8 Border-Style Property

This CSS explains how to use the CSS property called **border-style** with syntax and examples.

Description

The CSS border-style property defines the line style of the border of a box.

Syntax

The CSS border-style property can be expressed with one, two, three or four values provided.

One Value

The syntax for the CSS border-style property (with 1 value) is:

border-style: all;

When one single value is provided, the border-style value will apply to all four sides of the box (ie: top, right, bottom, left).

Two Values

The syntax for the CSS **border-style property** (with 2 values) is:

border-style: top_bottom left_right;

When two values are provided, the first value will apply to the top and bottom of the box. The second value will apply to the left and right sides of the box.

Three Values

The syntax for the CSS **border-style property** (with 3 values) is:

border-style: top right_left bottom;

When three values are provided, the first value will apply to the top of the box. The second value will apply to the right and left sides of the box. The third value will apply to the bottom of the box.

Four Values

The syntax for the CSS **border-style property** (with 4 values) is:

border-style: top right bottom left;

When four values are provided, the first value will apply to the top of the box. The second value will apply to the right side of the box. The third value will apply to the bottom of the box. The fourth value will apply to the left side of the box.

The top, right, bottom, and left border-style values can be one of the following:

Value	Description
none	No border (This is the default) div { border-style: none; }
solid	Single, straight, solid line div { border-style: solid; }
dotted	Series of dots div { border-style: dotted; }

dashed	Series of short dashes div { border-style: dashed; }
double	Two straight lines that total the pixel amount defined by border-width div { border-style: double; }
groove	Carved effect div { border-style: groove; }
ridge	3D appearance where border looks like it is "coming out" (opposite of groove) div { border-style: ridge; }
inset	Embedded appearance div { border-style: inset; }
outset	Embossed appearance (opposite of inset) div { border-style: outset; }
hidden	Border is hidden div { border-style: hidden; }
inherit	Element will inherit the border-style from its parent element div { border-style: inherit; }

Note

- In the CSS border-style property, you can combine different border-style values for *top*, *right*, *bottom*, and *left* sides of the box.
- Since the default value for CSS border-style is *none*, you must set a CSS border-style value for your border to appear.
- See also the border, border-color, and border-width properties.

Example

We will discuss the border-style property below, exploring examples of how to use this property in CSS with 1, 2, 3, or 4 values.

div { border-style: solid; }

In this CSS border-style example, we have provided one value of *solid* which would apply to all 4 sides of the box.

Next, we'll look at a CSS border-style example where we provide two values.

div { border-style: dashed dotted; }

In this CSS border-style example, we have provided two values. The first value of *dashed* would apply to the top and bottom of the box. The second value of *dotted* would apply to the left and right sides of the box.

Next, we'll look at a CSS border-style example where we provide three values.

div { border-style: none solid double; }

In this CSS border-style example, we have provided three values. The first value of *none* would apply to the top of the box. The second value of *solid* would apply to the right and left sides of the box. The third value of *double* would apply to the bottom of the box.

Next, we'll look at a CSS border-style example where we provide four values.

div { border-style: inset outset solid solid; }

In this CSS border-style example, we have provided four values. The first value of *inset* would apply to the top of the element. The second value of *outset* would apply to the right side of the box. The third value of *solid* would apply to the bottom of the box. The fourth value of *solid* would apply to the left side of the box.

3.2.9 Border-top Property

This CSS explains how to use the CSS property called **border-top** with syntax and examples.

Description

The CSS border-top property defines the width, line style, and color of the top border of a box. It is a shorthand property for setting the border-top-width, border-top-style, and border-top-color CSS properties.

Syntax

The syntax for the border-top CSS property is:

border-top: border-top-width border-top-style border-top-color;

Parameters or Arguments

border-top-width is the top border width of a box and can be one of the following: (If *border-top-width* is not provided, the default is *medium*)

Value	Description
fixed	Fixed value expressed in px, em, ... div { border-top: 2px; }

thin	Thin border-top width, which might be 1px or 2px depending on the browser div { border-top: thin; }
medium	Medium border-top width, which might be 3px or 4px depending on the browser div { border-top: medium; }
thick	Thick border-top width, which might be 5px or 6px depending on the browser div { border-top: thick; }

border-top-style is the line style of the top border of a box and can be one of the following: (If *border-top-style* is not provided, the default is *none*)

Value	Description
none	No border-top (This is the default) div { border-top: none; }
solid	Single, straight, solid line div { border-top: solid; }
dotted	Series of dots div { border-top: dotted; }
dashed	Series of short dashes div { border-top: dashed; }
double	Two straight lines that total the pixel amount defined by border-top-width div { border-top: double; }
groove	Carved effect div { border-top: groove; }
ridge	3D appearance where border-top looks like it is "coming out" (opposite of groove) div { border-top: ridge; }
inset	Embedded appearance div { border-top: inset; }
outset	Embossed appearance (opposite of inset) div { border-top: outset; }
hidden	Border is hidden div { border-top: hidden; }
inherit	Element will inherit the border-top-style from its parent element div { border-top: inherit; }

border-top-color is the color of the top border of a box and can be one of the following:

Value	Description
#RRGGBB	Hexadecimal representation of the border-top-color div { border-top: #FF0000; }
rgb()	RGB representation of the border-top-color div { border-top: rgb(255,0,0); }
name	Name of the border-top-color (ie: red, blue, black, white) div { border-top: red; }
transparent	Border is not displayed but still takes up space on the page div { border-top: transparent; }
inherit	Element will inherit the border-top-color from its parent element div { border-top: inherit; }

Note

- When using the border-top property, you can provide one or all of the values (border-top-width, border-top-style, and border-top-width values) and they can be provided in any order.
- You must provide a value for the border-top-style for the top border to appear.
- See also the border-top-color, border-top-style, and border-top-width properties.
- Need to convert your color value to a different representation? Try this online tool to convert your color value between hexadecimal and RGB.

Example

We will discuss the border-top property below, exploring examples of how to use this property in CSS.

For example:

div {border-top: solid;}

In this CSS border-top example, we have set the line style of the top border to *solid*. You must set the style for the top border or the top border will not appear.

Next, we'll look at a CSS border-top example where we provide the *border-top-width*, *border-top-style*, and *border-top-color*values.

div { border-top: 2px solid red; }

In this CSS border-top example, we have set the *border-top-width* to 2px, the *border-top-style* to solid, and the *border-top-color* to red.

We could rewrite this example using the hexadecimal value for red as follows:

div { border-top: 2px solid #FF0000; }

Or we could rewrite this example using the rgb() value for red as follows:

div { border-top: 2px solid rgb(255,0,0); }

3.2.10 Border-top-Color Property

This CSS explains how to use the CSS property called **border-top-color** with syntax and examples.

Description

The CSS border-top-color property defines the color of the top border of a box.

Syntax

The syntax for the border-top-color CSS property is:

border-top-color: value;

Parameters or Arguments

value

The color of the top border. It can be one of the following:

Value	Description
#RRGGBB	Hexadecimal representation of the border-top-color p { border-top-color: #FF0000; }
rgb()	RGB representation of the border-top-color p { border-top-color: rgb(255,0,0); }
name	Name of the border-top-color (ie: red, blue, black, white) p { border-top-color: red; }
transparent	Border is not displayed but still takes up space on the page p { border-top-color: transparent; }
inherit	Element will inherit the border-top-color from its parent element p { border-top-color: inherit; }

Note

- Be sure to set the border-top-style property as well to make sure that your top border appears.
- See also border-top-style, border-top-width, and border-top properties.
- Need to convert your color value to a different representation? Try this online tool to convert your color value between hexadecimal and RGB.

Example

We will discuss the border-top-color property below, exploring examples of how to use this property in CSS.

Using Hexadecimal

Let's look at a CSS border-top-color example where we have provided a hexadecimal value for the border-top-color property.

p { border-top-color: #FF0000; border-top-style: solid; }

In this border-top-color example, we have provided a hexadecimal value of #FF0000 which would display a red top border for the <p> tag. Be sure to apply a border-top-style so that the border appears.

Using RGB

We can also provide the border-top-color value using rgb.

p { border-top-color: rgb(255,0,0); border-top-style: solid; }

In this border-top-color example, rgb(255,0,0) would also display the top border for the <p> tag in red.

Using Color Name

Let's look at a CSS border-top-color example where we have provided the value as a named color.

p {border-top-color: red; border-top-style: solid;}

In this CSS border-top-color example, we have provided the name "red" which would also set the top border to red.

SUMMARY

- The box-sizing CSS property sets how the total width and height of an element is calculated. The width and height you assign to an element is applied only to the element's content box. If the element has any border or padding, this is then added to the width and height to arrive at the size of the box that's rendered on the screen.
- The box-sizing property allows us to include the padding and border in an element's total width and height.
- The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.
- The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border).
- The CSS border-image property allows you to specify an image to be used instead of the normal border around an element.
- The border-image property takes the image and slices it into nine sections, like a tic-tac-toe board. It then places the corners at the corners, and the middle sections are repeated or stretched as you specify.
- The CSS border-style property defines the line style of the border of a box.
- The syntax for the CSS border-style property (with 2 values) is: border-style: top_bottom left_right;
- The CSS border-top property defines the width, line style, and color of the top border of a box. It is a shorthand property for setting the border-top-width, border-top-style, and border-top-color CSS properties.

KNOWLEDGE CHECK

1. **Which of the following CSS property defines the different properties of all four sides of an element's border in a single declaration?**
   a. border
   b. padding
   c. border-collapse
   d. border-width
2. **Identify the CSS property defining bottom-left corner shape of the border?**
   a. border-radius
   b. border-corner-radius
   c. border-bottom-left-radius
   d. border-left-radius
3. **Select the CSS property that sets the width of an element's bottom border?**
   a. border-width
   b. border-bottom
   c. border-width-down
   d. border-bottom-width
4. **Which of the following CSS property border-color property sets the color of an element's four borders?**
   a. border-background
   b. border-background-color
   c. border-color
   d. all of the mentioned
5. **Choose the CSS property that can be used for collapsing the borders between table cells?**
   a. border
   b. collapse-border
   c. border-collapse
   d. border-cell
6. **Which of the following box-sizing property value is described by width and height include content, padding, and borders?**
   a. content-box
   b. content-box

c. border-box

d. All of above

7. Which of the following property sets a consistent margin on all four sides of the affected element?

a. padding

b. boder

c. margin

d. All of above

e. `None of these

8. Which of the following property defines the style for the bottom border of an element?

a. border-bottom-style

b. border-collapse

c. border-style-bottom

d. none of the mentioned

9. Which of the following property controls the horizontal overflow of a block or inline block?

a. overflow-x

b. overflow

c. overflow-y

d. overflow-k

10. Which of the following property sets a consistent margin on all four sides of the affected element?

a. boder

b. margin

c. padding

d. none fof the mentioned

REVIEW QUESTIONS

1. Write a program without the CSS box-sizing property.
2. Write a program border - individual sides.
3. Write a program border - shorthand property.
4. Write a program border-right-color property.
5. Write a program border-right-width property.
6. Write a program border-top property.

Check Your Result

1. (d)	2. (c)	3. (d)	4. (c)	5. (c)
6. (c)	7. (d)	8. (a)	9. (a)	10. (b)

REFERENCES

1. 5. Distance Units: the <length> type. CSS Values and Units Module Level 3. 6 June 2019. Archived from the original on 7 June 2019. Retrieved 20 June 2019.
2. Andrew Hunt and David Thomas. The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley, Reading, MA, 2000.
3. CSS Color. Mozilla Developer Network. 2016-06-28. Archived from the original on 2016-09-21. Retrieved 2016-08-23.
4. Jeffrey Zeldman. Designing with Web Standards. New Riders Press, Upper Saddle River, NJ, 2009.
5. Jon Duckett. HTML and CSS: Design and Build Websites. John Wiley & Sons, New York, NY, 2-11.
6. Meyer, Eric A. (2006). Cascading Style Sheets: The Definitive Guide (3rd ed.). O'Reilly Media, Inc. ISBN 0-596-52733-0. Archived from the original on 2014-02-15. Retrieved 2014-02-16.
7. W3C CSS validation service. Archived from the original on 2011-02-14. Retrieved 2012-06-30.
8. W3C CSS2.1 specification for pseudo-elements and pseudo-classes. World Wide Web Consortium. 7 June 2011. Archived from the original on 30 April 2012. Retrieved 30 April 2012.
9. W3C CSS2.1 specification for rule sets, declaration blocks, and selectors. World Wide Web Consortium. 7 June 2011. Archived from the original on 28 March 2008. Retrieved 2009-06-20.
10. W3C HTML Working Group. "HTML 5. A vocabulary and associated APIs for HTML and XHTML". World Wide Web Consortium. Archived from the original on 15 July 2014. Retrieved 28 June 2014.
11. Web-based Mobile Apps of the Future Using HTML 5, CSS and JavaScript". HTMLGoodies. 23 July 2010. Archived from the original on 2014-10-20. Retrieved 2014-10-16.

CHAPTER

4

APPEARANCE OF CASCADING STYLE SHEETS

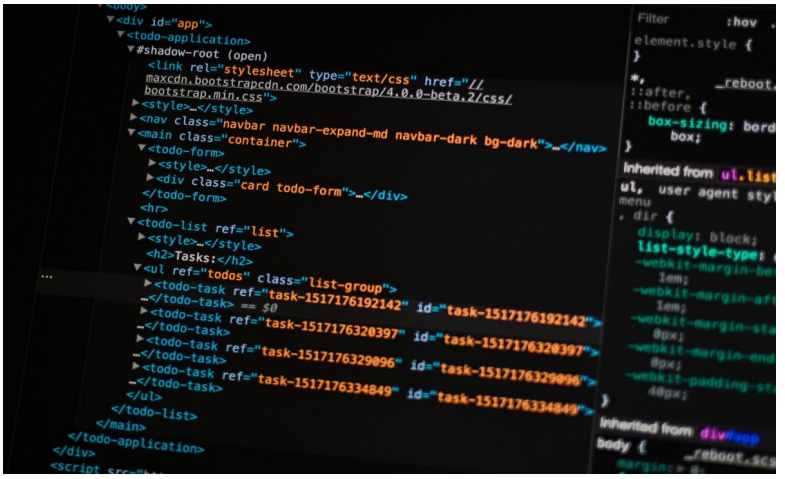
"Anyone can dream up great ideas, but an idea is nothing until it's realized, be it as a website, a physical product, an app, or a user interface."

– Jens Martin Skibsted

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Learn about the several style of CSS
2. Discuss on W3C CSS validation service



INTRODUCTION

Cascading style sheets (CSS) are used to specify the appearance of web pages written in HTML. Appearance includes colour, dimensions, position, and behaviour of

elements. The word "cascading" has a dual meaning: (1) the hierarchy of specifications, and (2) a file of type .css.

CSS file syntax is specified by the W3 Consortium. The file type was first proposed by Dr. Håkum Wium Lie, a web developer from Norway who was formerly the chief technology officer of Opera Software, publishers of the Opera browser.

The appearance CSS property is used to display an element using platform-native styling, based on the operating system's theme. The -moz-appearance and -webkit-appearance properties are non-standard versions of this property, used (respectively) by Gecko (Firefox) and by WebKit-based (e.g., Safari) and Blink-based (e.g., Chrome, Opera) browsers to achieve the same thing. Note that Firefox and Edge also support -webkit-appearance, for compatibility reasons.

4.1 SEVERAL STYLE OF CSS

A cascading style sheet (CSS) contains style definitions that are applied to elements in an HTML document. CSS styles define how elements are displayed and where they are positioned on your page. Instead of assigning attributes to each element on your page individually, you can create a general rule that applies attributes whenever a Web browser encounters an instance of an element, or an element assigned to a certain style CLASS.

CSS styles can be placed inline within a single HTML element, grouped in a <STYLE> block in the HEAD portion of a Web page, or imported from a separate CSS style sheet file. The same external style sheet file can be linked to many Web pages, thus giving a common appearance to an entire Web site.

4.1.1 Inline Styles

A Web developer can declare document styles in many ways. This section presents inline styles that declare an individual element's format using the XHTML attribute style. Inline styles override any other styles applied using the techniques. Figure 1 applies inline styles to p elements to alter their font size and color.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.1: inline.html -->
6  <!-- Using inline styles   -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Inline Styles</title>
```

```
11      </head>
12
13      <body>
14
15         <p>This text does not have any style applied to it.</p>
16
17         <!-- The style attribute allows you to declare -->
18         <!-- inline styles. Separate multiple styles   -->
19         <!-- with a semicolon.                         -->
20         <p style = "font-size: 20pt">This text has the
21         <em>font-size</em> style applied to it, making it 20pt.
22         </p>
23
24         <p style = "font-size: 20pt; color: #0000ff">
27         20pt. and blue.</p>
28
29      </body>
30   </html>
```

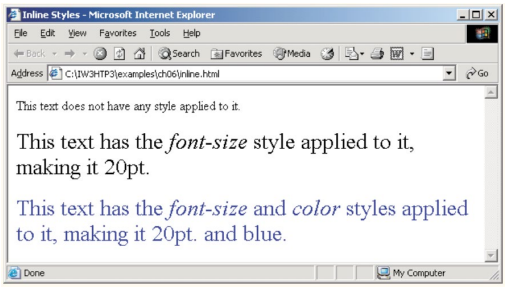


Figure 1. Inline styles.

The first inline style declaration appears in line 20. Attribute style specifies the style for an element. Each CSS property (the font-size property in this case) is followed by a colon and a value. In line 20, we declare this particular p element to use 20-point font size. Line 21 uses element em to "emphasize" text, which most browsers do by making the font italic. Line 24 specifies the two properties, font-size and color, separated by a semicolon. In this line, we set the given paragraph's color to blue, using the hexadecimal code #0000ff. Color names may be used in place of hexadecimal codes, as we demonstrate in the next example.

4.1.2 Embedded Style Sheets

A second technique for using style sheets is embedded style sheets. Embedded style sheets enable a Web-page author to embed an entire CSS document in an XHTML document's head section. Figure 2 creates an embedded style sheet containing four styles.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.2: declared.html                          -->
6  <!-- Declaring a style sheet in the header section. -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Style Sheets</title>
11
12       <!-- this begins the style sheet section -->
13       <style type = "text/css">
14
15          em       { background-color: #8000ff;
16                     color: white }
17
18          h1       { font-family: arial, sans-serif }
19
20          p        { font-size: 14pt }
21
22        .special { color: blue }
23
24       </style>
25    </head>
26
27    <body>
28
29       <!-- this class attribute applies the .special style -->
30       <h1 class = "special">Deitel & Associates, Inc.</h1>
31
32       <p>Deitel &amp; Associates, Inc. is an internationally
33       recognized corporate training and publishing organization
34       specializing in programming languages, Internet/World
35       Wide Web technology and object technology education.
36       Deitel &amp; Associates, Inc. is a member of the World Wide
37       Web Consortium. The company provides courses on Java,
38       C++, Visual Basic, C, Internet and World Wide Web
39       programming, and Object Technology.</p>
40
41       <h1>Clients</h1>
42       <p class = "special"> The company's clients include many
43       <em>Fortune 1000 companies</em>, government agencies,
44       branches of the military and business organizations.
45       Through its publishing partnership with Prentice Hall,
46       Deitel &amp; Associates, Inc. publishes leading-edge
47       programming textbooks, professional books, interactive
48        CD-ROM-based multimedia Cyber Classrooms, satellite
49        courses and World Wide Web courses.</p>
50
51    </body>
52 </html>
```

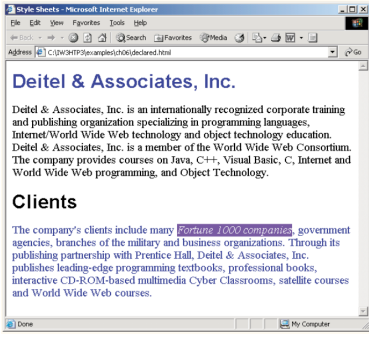


Figure 2. Embedded style sheets.

The style element (lines 13–24) defines the embedded style sheet. Styles placed in the head apply to matching elements wherever they appear in the entire document. The style element's type attribute specifies the Multipurpose Internet Mail Extensions (MIME) type that describes a file's content. CSS documents use the MIME type text/css. Other MIME types include image/gif (for GIF images) and text/javascript. The body of the style sheet (lines 15–22) declares the CSS rules for the style sheet. We

declare rules for em (lines 15–16), h1 (line 18) and p (line 20) elements. When the browser renders this document, it applies the properties defined in these rules to every element to which the rule applies. For example, the rule in lines 15–16 will be applied to all em elements (in this example, there is one in line 43). The body of each rule is enclosed in curly braces ({ and }).

Line 22 declares a style class named special. Style classes define styles that can be applied to any type of element. In this example, we declare class special, which sets color to blue. We can apply this style to elements of any type, whereas the other rules in this style sheet apply only to specific element types (i.e., em, h1 or p). Style class declarations are preceded by a period. We will discuss how to apply a style class momentarily.

Multiple properties are separated by semicolons (;). In the rule for em elements, the color property specifies the color of the text, and property background-color specifies the background color of the element. The font-family property (line 18) specifies the name of the font to use. In this case, we use the Arial font. The second value, sans-serif, is a generic font family. Not all users have the same fonts installed on their computers, so Web-page authors often specify a comma-separated list of fonts to use for a particular style. The browser attempts to use the fonts in the order they appear in the list. Many Web-page authors end a font list with a generic font family name in case the other fonts are not installed on the user's computer. In this example, if the arial font is not found on the system, the browser instead will display a generic sans-serif font, such as helvetica or verdana. Other generic font families include serif (e.g., times new roman, Georgia), cursive (e.g., script), fantasy (e.g., critter) and monospace (e.g., courier, fixedsys).

The font-size property (line 20) specifies a 14-point font. Other possible measurements in addition to pt (point). Relative values— xxsmall, x-small, small, smaller, medium, large, larger, x-large and xx-large— also can be used. Generally, relative values for font-size are preferred over point sizes because an author does not know the specific measurements of the display for each client. Relative font-size values permit more flexible viewing of Web pages.

Remember

CSS rules in embedded style sheets use the same syntax as inline styles; the property name is followed by a colon (:) and the value of the property.

A user may wish to view a Web page on a handheld device with a small screen. Specifying an 18- point font size in a style sheet will prevent such a user from seeing more than one or two characters at a time. However, if a relative font size is specified, such as large or larger, the actual size is determined by the browser .that displays the font

Using relative sizes also makes pages more accessible to users with disabilities. Users with impaired vision, for example, may configure their browser to use a larger default font, upon which all relative sizes are based. Text that the author specifies to be smaller than the main text still displays in a .smaller size font, yet it is clearly visible to each user

Line 30 uses attribute class in an h1 element to apply a style class—in this case class special (declared as .special in the style sheet). When the browser renders the h1 element, note that the text appears on screen with the properties of both an h1 element (Arial or sans-serif font defined in line 18) and the .special style class applied (the color blue defined in line 22). The formatting for the p element and the .special class are applied to the text in lines 42–49. All the styles applied to an element (the parent or ancestor element) also apply to the element's nested elements (child or descendant elements). The em element nested in the p element in line 43 inherits the style from the p element (namely, the 14-point font size in line 20), but retains its italic style. The em element has its own color property, so it overrides the color property of the special class.

4.1.3 Conflicting Styles

Cascading style sheets are "cascading" because styles may be defined by a user, an author or a user agent (e.g., a Web browser). Styles "cascade," or flow together, such that the ultimate appearance of elements on a page results from combining styles defined in several ways. Styles defined by the user take precedence over styles defined by the user agent, and styles defined by authors take precedence over styles defined by the user. Styles defined for parent elements are also inherited by child (nested) elements. In this section, we discuss the rules for resolving conflicts between styles defined for elements and styles inherited from parent and ancestor elements.

Figure 2 presented an example of inheritance in which a child em element inherited the font-size property from its parent p element. However, in Fig. 2, the child em element had a color property that conflicted with (i.e., had a different value than) the color property of its parent p element. Properties defined for child and descendant elements have a greater specificity than properties defined for parent and ancestor elements. According to the W3C CSS Recommendation, conflicts are resolved in favor of properties with a higher specificity. In other words, the styles explicitly defined for a child element are more specific than the styles defined for the child's parent element; therefore, the child's styles take precedence. Figure 3 illustrates examples of **inheritance** and specificity.

Keyword

Inheritance is the mechanism of basing an object or class upon another object (prototypical inheritance) or class (class-based inheritance), retaining similar implementation.

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig 6.3: advanced.html     -->
<!-- More advanced style sheets -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>More Styles</title>

      <style type = "text/css">

         a.nodec  { text-decoration: none }

         a:hover  { text-decoration: underline;
                    color: red;
                    background-color: #ccffcc }

         li em    { color: red;
                    font-weight: bold }

         ul       { margin-left: 75px }

         ul ul    { text-decoration: underline;
                    margin-left: 15px }

      </style>
   </head>

   <body>

      <ul>
         <li>Milk</li>
         <li>Bread
            <ul>
               <li>White bread</li>
               <li>Rye bread</li>
               <li>Whole wheat bread</li>
            </ul>
         </li>
         <li>Rice</li>
         <li>Potatoes</li>
         <li>Pizza <em>with mushrooms</em></li>
      </ul>

      <p><a class = "nodec" href = "http://www.food.com">
      Go to the Grocery store</a></p>

   </body>
</html>
```

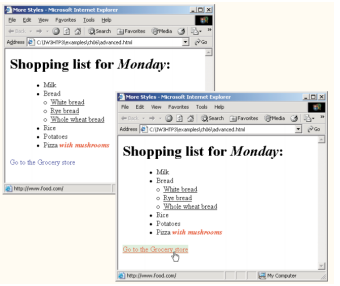


Figure 3. Inheritance in style sheets.

Line 14 applies property text-decoration to all a elements whose class attribute is set to nodec. The text-decoration property applies decorations to text within an element. By default, browsers underline the text of an a (anchor) element. Here, we set the text-decoration property to none to indicate that the browser should not underline hyperlinks. Other possible values for text-decoration include overline, linethrough, underline and blink. [Note: blink is not supported by Internet Explorer.] The .nodec appended to a is an extension of class styles; this style will apply only to a elements that specify nodec in their class attribute.

Lines 16–18 specify a style for hover, which is a pseudoclass. Pseudoclasses give the author access to content not specifically declared in the document. The hover pseudoclass is activated dynamically when the user moves the mouse cursor over an element. Note that pseudoclasses are separated by a colon (with no surrounding spaces) from the name of the element to which they are applied.

Lines 20–21 declare a style for all em elements that are children of li elements. In the screen output of Fig. 3, note that Monday (which line 33 contains in an em element) does not appear in bold red, because the em element is not in an li element. However, the em element containing with mushrooms (line 46) is nested in an li element; therefore, it is formatted in bold red. The syntax for applying rules to multiple elements is similar.For example, to apply the rule in lines 20–21 to all li and em elements, you would separate the elements with commas, as follows:

```
li, em  { color: red; font-weight: bold }
```

Lines 25–26 specify that all nested lists (ul elements that are descendants of ul elements) are to be underlined and have a left-hand margin of 15 pixels. A pixel is a relativelength measurement—it varies in size, based on screen resolution. Other relative lengths are em (the so-called M-height of the font, which is usually set to the height of an uppercase M), ex (the so-called x-height of the font, which is usually set to the height of a lowercase x) and percentages (e.g., margin-left: 10%). To set an element to display text at 150% of its default text size, the author could use the syntax

```
font-size: 1.5em
```

Other units of measurement available in CSS are absolute-length measurements—i.e., units that do not vary in size based on the system. These units are in (inches), cm (centimeters), mm (millimeters), pt (points; 1 pt=1/72 in) and pc (picas—1 pc = 12 pt).

In Fig. 3, the entire list is indented because of the 75-pixel left-hand margin for toplevel ul elements. However, the nested list is indented only 15 pixels more (not another 75 pixels) because the child ul element's margin-left property (in the ul ul rule in line 25) overrides the parent ul element's margin-left property.

4.1.4 Linking External Style Sheets

Style sheets are a convenient way to create a document with a uniform theme. With external style sheets (i.e., separate documents that contain only CSS rules), Web-page authors can provide a uniform look and feel to an entire Web site. Different pages on a site can all use the same style sheet. When changes to the styles are required, the Web-page author needs to modify only a single CSS file to make style changes across the entire Web site. Figure 4 presents an external style sheet. Lines 1–2 are CSS comments. Like XHTML comments, CSS comments describe the content of a CSS document. Comments may be placed in any type of CSS code (i.e., inline styles, embedded style sheets and external style sheets) and always start with /* and end with */. Text between these delimiters is ignored by the browser.

```
1  /* Fig. 6.4: styles.css   */
2  /* An external stylesheet */
3
4  a       { text-decoration: none }
5
6  a:hover { text-decoration: underline;
7            color: red;
8            background-color: #ccffcc }
9
10 li em   { color: red;
11           font-weight: bold;
12           background-color: #ffffff }
13
14 ul       { margin-left: 2cm }
15
16 ul ul    { text-decoration: underline;
17           margin-left: .5cm }
```

Figure 4. External style sheet (styles.css).

Keyword

Style Sheet is a collection of style rules that that tells a browser how the various styles are to be applied to the HTML tags to present the document.

Figure 5 contains an XHTML document that references the external **style sheet** in Fig. 4. Lines 11–12 (Fig. 5) show a link element that uses the rel attribute to specify a relationship between the current document and another document. In this case, we declare

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.5: external.html        -->
6  <!-- Linking external style sheets  -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Linking External Style Sheets</title>
11       <link rel = "stylesheet" type = "text/css"
12          href = "styles.css" />
13    </head>
14
15    <body>
16
17       <h1>Shopping list for <em>Monday</em>:</h1>
18       <ul>
19          <li>Milk</li>
20          <li>Bread
21           <ul>
22              <li>White bread</li>
23              <li>Rye bread</li>
24              <li>Whole wheat bread</li>
25           </ul>
26        </li>
27        <li>Rice</li>
28        <li>Potatoes</li>
29        <li>Pizza <em>with mushrooms</em></li>
30     </ul>
31
32     <p>
33     <a href = "http://www.food.com">Go to the Grocery store</a>
34     </p>
35
36  </body>
37 </html>
```

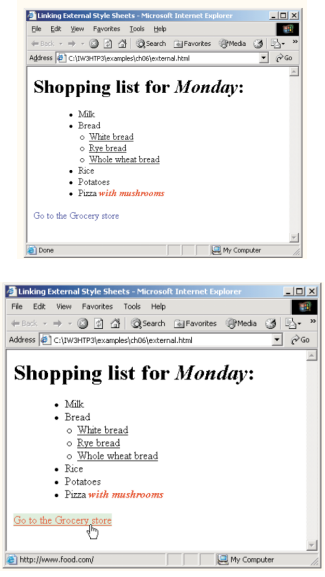


Figure 5. Linking an external style sheet.

4.2 W3C CSS VALIDATION SERVICE

The W3C provides a validation service that validates external CSS documents to ensure that they conform to the W3C CSS Recommendation. Like XHTML validation, CSS

validation ensures that style sheets are syntactically correct. The validator provides the option of either entering the CSS document's URL, pasting the CSS document's contents into a text area or uploading a CSS document. Figure 6 illustrates uploading a CSS document, using the file upload feature available at jigsaw.w3.org/css-validator/validator-upload.html. To validate the document, click the Browse… button to locate the file on your computer. Like many W3C Recommendations, the CSS Recommendation is being developed in stages (or versions). The current version under development is Version 3, so select CSS version 3 in the Profile drop-down list. This field indicates to the validator the CSS Recommendation against which the uploaded file should be validated. Click Submit this CSS file for validation to upload the file for validation. Figure 7 shows the results of validating styles.css (Fig. 4).

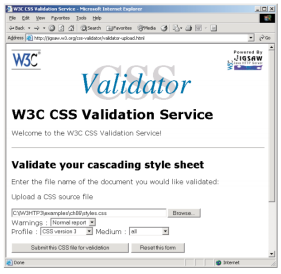


Figure 6. Validating a CSS document. (Courtesy of World Wide Web Consortium (W3C).)

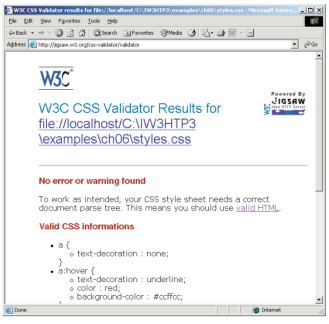


Figure 7. CSS validation results.

4.2.1 Positioning Elements

Before CSS, controlling the positioning of elements in an XHTML document was difficult—the browser determined positioning. CSS introduced the position property and a capability called absolute positioning, which gives authors greater control over how document elements are displayed. Figure 8 demonstrates absolute positioning.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig 6.8: positioning.html        -->
6  <!-- Absolute positioning of elements -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Absolute Positioning</title>
11    </head>
12
13    <body>
14
15       <p><img src = "i.gif" style = "position: absolute;
16          top: 0px; left: 0px; z-index: 1"
17          alt = "First positioned image" /></p>
18       <p style = "position: absolute; top: 50px; left: 50px;
19          z-index: 3; font-size: 20pt">Positioned Text</p>
20       <p><img src = "circle.gif" style = "position: absolute;
21          top: 25px; left: 100px; z-index: 2" alt =
22          "Second positioned image" /></p>
23
24    </body>
25 </html>
```

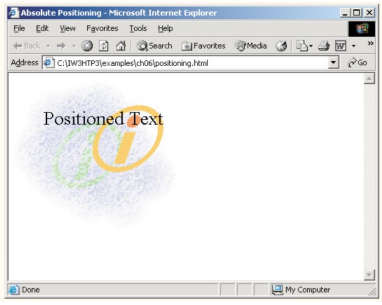


Figure 8. Absolute positioning of elements with CSS.

Keyword

Block-level elements, have a rectangular structure. By default, these elements will span the entire width of its parent element, and will thus not allow any other element to occupy the same horizontal space as it is placed on.

Lines 15–17 position the first img element (i.gif) on the page. Specifying an element's position as absolute removes the element from the normal flow of elements on the page, instead positioning it according to the distance from the top, left, right or bottom margins of its containing **block-level element** (i.e., an element such as body or p). Here, we position the element to be 0 pixels away from both the top and left margins of the p element (lines 15–17).

The z-index attribute allows you to layer overlapping elements properly. Elements that have higher z-index values are displayed in front of elements with lower z-index values. In this example, i.gif has the lowest z-index (1), so it displays in the background. The img element in lines 20–22 (circle. gif) has a z-index of 2, so it displays in front of i.gif. The p element in lines 18–19 (Positioned Text) has a z-index of 3,

so it displays in front of the other two. If you do not specify a z-index or if elements have the same z-index value, the elements are placed from background to foreground in the order they are encountered in the document.

Absolute positioning is not the only way to specify page layout. Figure 9 demonstrates relative positioning, in which elements are positioned relative to other elements.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.9: positioning2.html       -->
6  <!-- Relative positioning of elements  -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Relative Positioning</title>
11
12       <style type = "text/css">
13
14          p           { font-size: 1.3em;
15                        font-family: verdana, arial, sans-serif }
16
17          span        { color: red;
18                        font-size: .6em;
19                        height: 1em }
20
21          .super      { position: relative;
22                        top: -1ex }
23
24          .sub        { position: relative;
25                        bottom: -1ex }
26
27          .shiftleft  { position: relative;
28                        left: -1ex }
29
30          .shiftright { position: relative;
31                        right: -1ex }
32
33       </style>
34    </head>
35
36    <body>
37
38       <p>The text at the end of this sentence
39       <span class = "super">is in superscript</span>.</p>
40
41       <p>The text at the end of this sentence
42       <span class = "sub">is in subscript</span>.</p>
43
44       <p>The text at the end of this sentence
45       <span class = "shiftleft">is shifted left</span>.</p>
46
47       <p>The text at the end of this sentence
48       <span class = "shiftright">is shifted right</span>.</p>
49
50    </body>
51 </html>
```

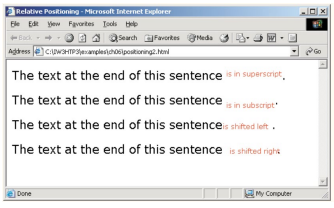


Figure 9. Relative positioning of elements with CSS.

Setting the position property to relative, as in class super (lines 21–22),lays out the element on the page and offsets it by the specified top, bottom, left or right value. Unlike absolute positioning, relative positioning keeps elements in the general flow of elements on the page, so positioning is relative to other elements in the flow. Recall

that ex (line 22) is the x-height of a font, a relative length measurement typically equal to the height of a lowercase x.

We introduce the span element in line 39. Element span is a grouping element—it does not apply any inherent formatting to its contents. Its primary purpose is to apply CSS rules or id attributes to a block of text. Element span is an inline-level element—it is displayed inline with other text and with no line breaks. Lines 17–19 define the CSS rule for span. A similar element is the div element, which also applies no inherent styles but is displayed on its own line, with margins above and below (a block-level element).

4.2.2 Backgrounds

CSS provides control over the element backgrounds. We introduced the background-color property. CSS also can add background images to documents. Figure 10 adds a corporate logo to the bottom-right corner of the document. This logo stays fixed in the corner even when the user scrolls up or down the screen.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.10: background.html                -->
6  <!-- Adding background images and indentation -->
7
8  <html xmlns = "http://www.w3 .org/1999/xhtml">
9     <head>
10       <title>Background Images</title>
11
12       <style type = "text/css">
13
14          body  { background-image: url(logo.gif);
15                  background-position: bottom right;
16                  background-repeat: no-repeat;
17                  background-attachment: fixed; }
18
19          p     { font-size: 18pt;
20                  color: #aa5588;
21                  text-indent: 1em;
22                  font-family: arial, sans-serif; }
23
24          .dark { font-weight: bold }
25
26       </style>
27    </head>
28
29    <body>
30
31       <p>
32       This example uses the background-image,
33       background-position and background-attachment
34       styles to place the <span class = "dark">Deitel
35       &amp;amp; Associates, Inc.</span> logo in the bottom,
36       right corner of the page. Notice how the logo
37       stays in the proper position when you resize the
38       browser window.
39       </p>
40
41    </body>
42 </html>
```

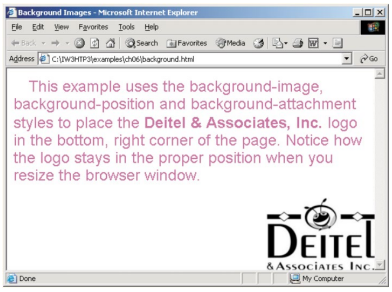


Figure 10. Background image added with CSS

The background-image property (line 14) specifies the image URL for the image logo.gif in the format url(fileLocation). The Web-page author also can set the background-color property in case the image is not found. The background-position property (line 15) places the image on the page. The keywords top, bottom, center, left and right are used individually or in combination for vertical and horizontal positioning. An image can be positioned using lengths by specifying the horizontal length followed by the vertical length. For example, to position the image as horizontally centered (positioned at 50% of the distance across the screen) and 30 pixels from the top, use

```
background-position: 50% 30px;
```

The background-repeat property (line 16) controls the tiling of the background image. Tiling places multiple copies of the image next to each other to fill the background. Here, we set the tiling to no-repeat to display only one copy of the background image. The background-repeat property can be set to repeat (the default) to tile the image vertically and horizontally, repeat-x to tile the image only horizontally or repeat-y to tile the image only vertically. The final property setting, background-attachment: fixed (line 17), fixes the image in the position specified by background-position. Scrolling the browser window will not move the image from its position. The default value, scroll, moves the image as the user scrolls through the document. Line 21 indents the first line of text in the element by the specified amount, in this case 1em. An author might use this property to create a **Web page** that reads more like a novel, in which the first line of every paragraph is indented.

Keyword

Web page is a document that is suitable for the World Wide Web and web browsers. A web browser displays a web page on a monitor or mobile device.

Line 24 uses the font-weight property to specify the "boldness" of text. Possible values are bold, normal (the default), bolder (bolder than bold text) and lighter (lighter than normal text). Boldness also can be specified with multiples of 100, from 100 to 900 (e.g., 100, 200, …, 900). Text specified as normal is equivalent to 400, and bold text is equivalent to 700.

4.2.3 Element Dimensions

In addition to positioning elements, CSS rules can specify the actual dimensions of each page element. Figure 11 demonstrates how to set the dimensions of elements.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.11: width.html                      -->
6  <!-- Setting box dimensions and aligning text  -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Box Dimensions</title>
11
12       <style type = "text/css">
13
14          div { background-color: #ffccff;
15                margin-bottom: .5em }
16       </style>
17
18    </head>
19
20    <body>
21
22       <div style = "width: 20%">Here is some
23       text that goes in a box which is
24       set to stretch across twenty percent
25       of the width of the screen.</div>
26
27       <div style = "width: 80%; text-align: center">
28       Here is some CENTERED text that goes in a box
29       which is set to stretch across eighty percent of
30       the width of the screen.</div>
31
32       <div style = "width: 20%; height: 30%; overflow: scroll">
33       This box is only twenty percent of
34       the width and thirty percent of the height.
35       What do we do if it overflows? Set the
36       overflow property to scroll!</div>
37
38    </body>
39 </html>
```

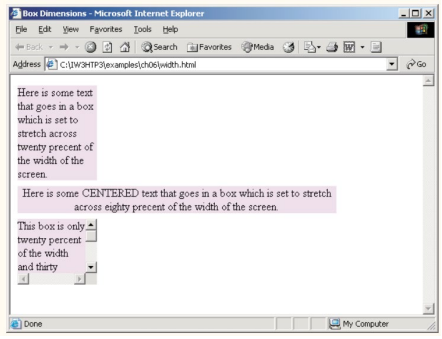


Figure 11. Element dimensions and text alignment.

The inline style in line 22 illustrates how to set the width of an element on screen; here, we indicate that the div element should occupy 20% of the screen width. Most elements are left-aligned by default; however, this alignment can be altered to position the element elsewhere. The height of an element can be set similarly, using the height property. The height and width values also can be specified as relative or absolute lengths. For example

```
width: 10em
```

sets the element's width to be equal to 10 times the font size. Line 27 sets text in the element to be center aligned; other values for the text-align property include left and right. One problem with setting both dimensions of an element is that the content inside the element can exceed the set boundaries, in which case the element is simply made large enough for all the content to fit. However, in line 32, we set the overflow property to scroll, a setting that adds scrollbars if the text .overflows the boundaries

Remember

Many systems do not have fonts that can scale with this level of precision, so using the values from 100 to 900 might not display the desired effect. Another CSS property that formats text is the font-style property, which allows the developer to set text to none, italic or oblique (oblique will default to italic if the system does not support oblique text).

4.2.4 User Style Sheets

Users can define their own user style sheets to format pages based on their preferences. For example, people with visual impairments may want to increase the page's text size. Webpage authors need to be careful not to inadvertently override user preferences with defined styles. This section discusses possible conflicts between author styles and user styles. Figure 12 contains an author style. The font-size is set to 9pt for all

tags that have class note applied to them.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.16: user_absolute.html    -->
6  <!-- User styles                      -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>User Styles</title>
11
12       <style type = "text/css">
13
14          .note { font-size: 9pt }
15
16       </style>
17    </head>
18
19    <body>
20
21       <p>Thanks for visiting my Web site. I hope you enjoy it.
22       </p><p class = "note">Please Note: This site will be
23       moving soon. Please check periodically for updates.</p>
24
25    </body>
26 </html>
```

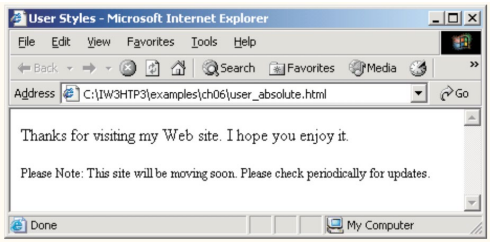


Figure 12. pt measurement for text size.

User style sheets are external style sheets. Figure 13 shows a user style sheet that sets the body's font-size to 20pt, color to yellow and background-color to #000080.

```
1  /* Fig. 6.17: userstyles.css */
2  /* A user stylesheet         */
3
4  body     { font-size: 20pt;
5             color: yellow;
6             background-color: #000080 }
```

Figure 13. User style sheet.

User style sheets are not linked to a document; rather, they are set in the browser's options. To add a user style sheet in Internet Explorer 6, select Internet Options..., located in the Tools menu. In the Internet Options dialog (Fig. 14) that appears, click Accessibility..., check the Format documents using my style sheet checkbox, and type the location of the user style sheet. **Internet Explorer** 6 applies the user style sheet to any document it loads.

Keyword

Internet Explorer is a series of graphical web browsers developed by Microsoft and included in the Microsoft Windows line of operating systems.

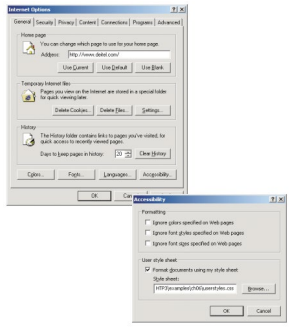


Figure 14. User style sheet in Internet Explorer 6.

The Web page from Fig. 12 is displayed in Fig. 15, with the user style sheet from Fig. 13 applied.

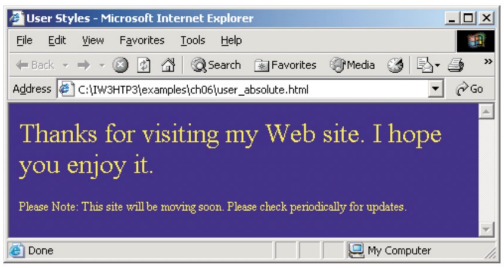


Figure 15. User style sheet applied with pt measurement.

In this example, if users define their own font-size in a user style sheet, the author style has a higher precedence and overrides the user style. The 9pt font specified in the author style sheet overrides the 20pt font specified in the user style sheet. This small font may make pages difficult to read, especially for individuals with visual impairments. A developer can avoid this problem by using relative measurements (e.g., em or ex) instead of absolute measurements, such as pt. Figure 16 changes the font-size property to use a relative measurement (line 14) that does not override the user style set in Fig. 13. Instead, the font size displayed is relative to the one specified in the user style sheet. In this case, text enclosed in the tag displays as 20pt ,and tags that have class note applied to them are displayed in 15pt (.75 times 20pt).

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.20: user_relative.html   -->
6  <!-- User styles                     -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>User Styles</title>
11
12       <style type = "text/css">
13
14          .note { font-size: .75em }
15
16       </style>
17    </head>
18
19    <body>
20
21       <p>Thanks for visiting my Web site. I hope you enjoy it.
22       </p><p class = "note">Please Note: This site will be
23       moving soon. Please check periodically for updates.</p>
24
25    </body>
26 </html>
```

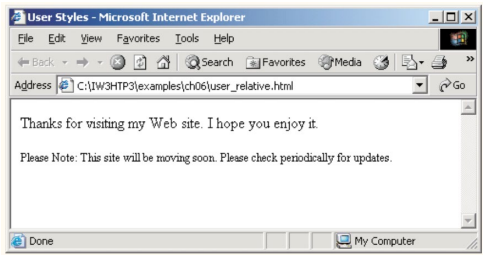


Figure 16. em measurement for text size.

Figure 17 displays the Web page from Fig. 16 with the user style sheet from Fig. 12 applied. Note that the second line of text displayed is larger than the same line of text in Fig. 15.

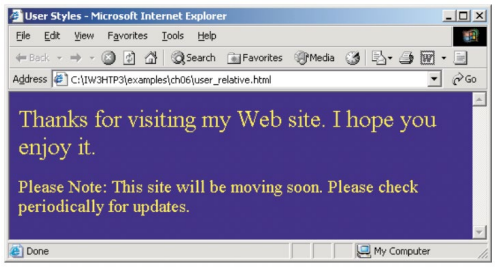


Figure 17. User style sheet applied with em measurement.

Did You Know?

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

CASE STUDY

WEB DEVELOPMENT

UPGRADING TABLES TO DIVS: BY MICHAEL ROHDE

It's hard to imagine that some web developers are still using table tags for web page layouts. But there are still some out there that do. I was recently asked by the Principal Consultant of the firm that I work at to take a recently developed online tool and "make it look better." These projects are always fun, because making it look better can mean a myriad of different things to different people. I undertook the project with an open mind and was careful not to step on the original developer's toes. Here's how it looked when I originally received it.

Looking at the code, the first thing that I noticed was that the entire page was coded in table tags. I reworked the code so that instead of table tags, it used div tags in the CSS.

Here's the code in tables for the row of logos along the bottom of the page, which were ultimately moved to the top:

<table cellspacing='0' cellpadding='0' border='0' align='center' style='padding-top:50px' width='900'>

<tr>

<th><img src='fsclogo.jpg' alt='Freeman, Sullivan'></th>

<th><img src='berkeleylabslogo.jpg' alt='Berkeley Labs'></th>

<th><img src='doelogo.jpg' alt='US Department of Energy'></th>

</tr>

</table>

</th>

</tr>

</table>

I reworked the table tags into div tags. Here's the new code for the HTML:

<img src='berkeleylabslogo.jpg' alt='Berkeley Labs' align="left">

<img src='doelogo.jpg' alt='US Department of Energy' align="left"

<img src='fsclogo.jpg' alt='Freeman, Sullivan' align="right">

Here's the CSS code:

#logos {

width: 900px;

}

The amount of written code went from 447 characters to 361. The new code for the header was designed in a similar fashion but I used nested div tags to help with the spacing. Here is the HTML for the header that appears below the logos in the final version:

<img src='powerlines.jpg' align="left">

<h1 align="left"> ICECalculator.com</h1>

<h2 align="left"> Interruption Cost Estimate </br>   ;Calculator</h2>

Yes, I admit I used hard space code to center the word 'Calculator,' but at the time I was under deadline and I had to work quickly. And the end result works.

And here's the CSS:

#header {

background-color: #3399FF;

-moz-border-radius: 15px;

-webkit-border-radius: 15px;

border: 2px solid #0033FF;

padding: 0px;

width: 900px;

float: left;

```
}
#header #text1 {
width: 440px;
float: left;
}
#header #text2 {
width: 310px;
float: right;
text-align: left;
vertical-align: middle;
}
```

You'll notice in the first div there's vendor-specific code, which means the rounded corners will not appear in Internet Explorer.

The original HTML code for the Main Menu consisted of over 3,000 characters using table tags. In the interest of saving space, I won't repeat the code here. Instead, I'll provide the HTML and CSS for the rewritten code. Here's the HTML for both the side menu and the main window:

```
<div id="sidemenu">
Home</br></br>
<a href="about.htm">About the Calculator</a></br></br>
<a href="disclaimer.htm">Disclaimer</a></br></br>
<a href="relevant-reports.htm">Relevant Reports</a></br></br>
<a href='mailto:webmaster@fscgroup.com'>Contact Us</a>
</div>
<div id="mainwindow">
<a href="1">Estimate Interruption Costs</a></br>
Estimate the cost per event, cost per average kW, cost per unserved kWh and the total cost of
unreliability. </br></br>
<a href="2">Estimate Value of Reliability Improvement in a Static Environment</
a></br>
Estimate the value associated with a given reliability improvement. The environment is "static" because the expected reliability with and without the improvement does not change over time.</br></br>
```

Estimate Value of Reliability Improvement in a Dynamic Environment</br>

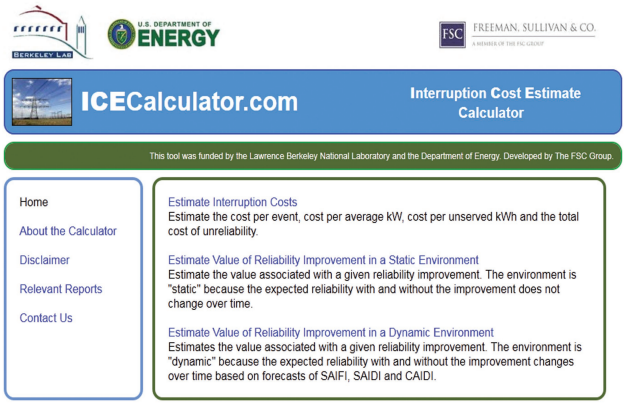
Estimates the value associated with a given reliability improvement. The environment is "dynamic" because the expected reliability with and without the improvement changes over time based on forecasts of SAIFI, SAIDI and CAIDI.

And the CSS for the side menu and the main window:

```
#sidemenu {
background-color: #ffffff;
-moz-border-radius: 15px;
-webkit-border-radius: 15px;
border: 4px solid #6495ed;
padding: 20px;
width: 160px;
height: 250px;
float: left;
text-align:left;
}
#mainwindow {
background-color: #ffffff;
-moz-border-radius: 15px;
-webkit-border-radius: 15px;
border: 4px solid #556b2f;
padding: 20px;
width: 630px;
height: 250px;
float: right;
text-align:left;
}
```

Between the HTML and the CSS, the total character count comes to a little over 1,100. The difference between using tables and div tags here comes close to 2,000 characters, which is a world of difference.

And here's the final output:


BERKELEY LAB
U.S. DEPARTMENT OF ENERGY
FSC
FREEMAN, SULLIVAN & CO.
A MEMBER OF THE FSC GROUP
ICECalculator.com
Interruption Cost Estimate Calculator
This tool was funded by the Lawrence Berkeley National Laboratory and the Department of Energy. Developed by The FSC Group.
Home
About the Calculator
Disclaimer
Relevant Reports
Contact Us
Estimate Interruption Costs
Estimate the cost per event, cost per average kW, cost per unserved kWh and the total cost of unreliability.
Estimate Value of Reliability Improvement in a Static Environment
Estimate the value associated with a given reliability improvement. The environment is "static" because the expected reliability with and without the improvement does not change over time.
Estimate Value of Reliability Improvement in a Dynamic Environment
Estimates the value associated with a given reliability improvement. The environment is "dynamic" because the expected reliability with and without the improvement changes over time based on forecasts of SAIFI, SAIDI and CAIDI.

SUMMARY

- A cascading style sheet (CSS) contains style definitions that are applied to elements in an HTML document. CSS styles define how elements are displayed and where they are positioned on your page.
- CSS styles can be placed inline within a single HTML element, grouped in a <STYLE> block in the HEAD portion of a Web page, or imported from a separate CSS style sheet file.
- Inline styles override any other styles applied using the techniques. The first inline style declaration appears in line 20. Attribute style specifies the style for an element.
- A second technique for using style sheets is embedded style sheets. Embedded style sheets enable a Web-page author to embed an entire CSS document in an XHTML document's head section.
- Cascading style sheets are "cascading" because styles may be defined by a user, an author or a user agent (e.g., a Web browser).
- Style sheets are a convenient way to create a document with a uniform theme. With external style sheets (i.e., separate documents that contain only CSS rules), Web-page authors can provide a uniform look and feel to an entire Web site.
- The W3C provides a validation service that validates external CSS documents to ensure that they conform to the W3C CSS Recommendation.
- CSS provides control over the element backgrounds. We introduced the background-color property. CSS also can add background images to documents.
- The Web-page author also can set the background-color property in case the image is not found. The background-position property places the image on the page.

KNOWLEDGE CHECK

1. **Which of the following selector selects all elements of E that have the attribute attr that end with the given value?**
   a. E[attr^=value]
   b. E[attr$=value]
   c. E[attr*=value]
   d. none of the mentioned
2. **Which of the following selector selects the elements that are checked?**
   a. E ~ F
   b. ::after
   c. :checked
   d. none of the mentioned
3. **Which of the following selector selects the elements that are the default among a set of similar elements?**
   a. :default
   b. :%
   c. :disabled
   d. none of the mentioned
4. **Which of the following selector selects an element that has no children?**
   a. :empty
   b. :nochild
   c. :inheritance
   d. :no-child
5. **Which of the following selector selects the elements that are currently enabled?**
   a. :element
   b. :empty
   c. :enabled
   d. none of the mentioned
6. **Suppose we want to arragnge five nos. of DIVs so that DIV4 is placed above DIV1. Now, which css property will we use to control the order of stack?**
   a. d-index
   b. s-index

c. x-index

d. z-index

7. If we want to wrap a block of text around an image, which css property will we use?

a. wrap

b. push

c. float

d. align

8. CSS is created and maintained through a group of people within the W3C called the?

a. CSS Class Tag

b. CSS Class Group

c. CSS Working Group

d. CSS Working Id

9 ..... has introduced text, list, box, margin, border, color, and background properties.

a. css

b. html

c. ajax

d. php

10. Cascading Style Sheets level 1 (CSS1) came out of W3C as a recommendation in?

a. 35400

b. 35431

c. 35462

d. 35490

REVIEW QUESTIONS

1. Discuss about the inline styles of CSS.
2. How to manage embedded style sheets in CSS?
3. What is linking external style sheets?
4. How positioning the HTML elements in CSS?
5. Explain the view of element dimensions.

Check Your Result

1. (b)	2. (c)	3. (a)	4. (a)	5. (c)
6. (d)	7. (c)	8. (c)	9. (a)	10. (a)

REFERENCES

1. Archive of W3C News in 2007. World Wide Web Consortium. Archived from the original on 2011-06-28. Retrieved 2011-02-16.
2. Bos, Bert; et al. (7 December 2010). "9.3 Positioning schemes". Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. W3C. Archived from the original on 18 February 2011. Retrieved 16 February 2011.
3. Byrne, Jim. 60 hot to touch Accessible Web Design tips – the tips no web developer can live without!, Jim Byrne, 2006, (ISBN: 978-1-4116-6729-7).
4. Cascading Style Sheets, level 1. World Wide Web Consortium. Archived from the original on 2014-04-09. Retrieved 2014-03-07.
5. Cascading Style Sheets. University of Oslo. Archived from the original on 2006-09-06. Retrieved 3 September 2014.
6. Chisholm, and May. Universal Design for Web Applications: Web Applications That Reach Everyone, O'Reilly Media, 2008.
7. Duckett, Jon. Accessible XHTML and CSS Web Sites Problem Design Solution, Wrox, 2005.
8. Feingold , Lainey.Structured Negotiation: A Winning Alternative to Lawsuits , American Bar Association, 2016.
9. McBride, Don (27 November 2009). "File Types". Archived from the original on 29 October 2010. Retrieved 20 June 2010.
10. Nielsen, Henrik Frystyk (7 June 2002). "Libwww Hackers". World Wide Web Consortium. Archived from the original on 2 December 2009. Retrieved 6 June 2010.
11. Nitot, Tristan (18 March 2002). "Incorrect MIME Type for CSS Files". Mozilla Developer Center. Mozilla. Archived from the original on 2011-05-20. Retrieved 20 June 2010.
12. Shorthand properties. Tutorial. Mozilla Developers. 2017-12-07. Archived from the original on 2018-01-30. Retrieved 2018-01-30.
13. The W3C Team: Technology and Society. World Wide Web Consortium. 18 July 2008. Archived from the original on 28 May 2010. Retrieved 22 January 2011. https://www.w3.org/TR/CSS2/css2.pdf

CHAPTER

5

POSITIONING ELEMENTS

"CSS is the design language of the web, and it is not as easy to use as it ought to be, and it can be confusing, especially if you are new to it."

– Jeffrey Zeldman

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Describe the types of position property
2. Explain how to create multiple columns

INTRODUCTION

The CSS position is how we position each element in a document. This property is a single keyword, and we attach a value to it to set the specific position of an element.

There are five main values for the position property. We will define these in detail below:

- static
- relative
- absolute
- fixed
- sticky

The element() CSS function defines an <image> value generated from an arbitrary HTML element. This image is live, meaning that if the HTML element is changed, the CSS properties using the resulting value are automatically updated.

Did You Know?

CSS 2.1 first became a Candidate Recommendation on February 25, 2004, but it was reverted to a Working Draft on June 13, 2005 for further review. It returned to Candidate Recommendation on 19 July 2007 and then updated twice in 2009. However, because changes and clarifications were made, it again went back to Last Call Working Draft on 7 December 2010.

5.1 THE POSITION PROPERTY

The *position* property in CSS tells about the method of positioning for an element or an HTML entity. There are five different types of position property available in CSS:

- Fixed
- Static
- Relative
- Absolute
- Sticky

The positioning of an element can be done using the *top*, *right*, *bottom* and *left* property. These specify the distance of an HTML element from the edge of the viewport. To set the position by these four properties, we have to declare the positioning method.

5.1.1 Fixed Positioning

Fixed positioning allows you to fix the position of an element to a particular spot on the page, regardless of scrolling. Specified coordinates will be relative to the **browser window**.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

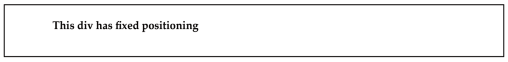
- Move Left - Use a negative value for *left*.
- Move Right - Use a positive value for *left*.

- Move Up - Use a negative value for *top*.
- Move Down - Use a positive value for *top*.

NOTE – You can use *bottom* or *right* values as well in the same way as *top* and *left*.

```
<html>
   <head>
   </head>
   <body>
      <div style = "position:fixed; left:80px; top:20px; background-color:yellow;">
         This div has fixed positioning.
      </div>
   </body>
</html>
```

It will produce the following result –



Any HTML element with **position: fixed** property will be positioned relative to the viewport. An element with fixed positioning allows it to remain at the same position even we scroll the page. We can set the position of the element using the top, right, bottom, left.

```
<!-->html code<-->
<body>
   <div class="fixed">This div has <span>position: fixed;</span></div>
      <pre>
         Lorem ipsum dolor sits amet, consectetur adipiscing elit.
         Nunc eget mauris at urna hendrerit iaculis sit amet et ipsum.
         Maecenas nec mi eget leo malesuada vehicula.
         Nam eget velit maximus, elementum ante pretium, aliquet felis.
```

Keyword

Browser window is the space on your Chrome, Safari or Firefox where websites are being displayed.

Remember

In case of the print media type, the fixed positioned element is rendered on every page, and is fixed with respect to the page box (even in print-preview). IE7 and IE8 support the fixed value only if a <!DOCTYPE> is specified.

```
            Aliquam quis turpis laoreet, porttitor lacus at, posuere massa.
        </pre>
</body>
```

Below is the CSS code to illustrate the fixed property:

```
// css code
body
{
    margin: 0;
    padding: 20px;
    font-family: sans-serif;
    background: #efefef;
}

.fixed
{
    position: fixed;
    background: #cc0000;
    color: #ffffff;
    padding: 30px;
    top: 50;
    left: 10;
}

span
{
    padding: 5px;
    border: 1px #ffffff dotted;
}
```

5.1.2 Static Positioning

This method of positioning is set by default. If we do not mention the method of positioning for any element, the element has the **position:static** method by default. By defining Static, the top, right, bottom and left will not have any control over the

element. The element will be positioned with the **normal flow** of the page.

```
<!-->html code<-->
<body>
  <div class="static">This div has <span>position: static;</span></div>
    <pre>
      Lorem ipsum dolor sits amet, consectetur adipiscing elit.
      Nunc eget mauris at urna hendrerit iaculis sit amet et ipsum.
      Maecenas nec mi eget leo malesuada vehicula.
      Nam eget velit maximus, elementum ante pretium, aliquet felis.
      Aliquam quis turpis laoreet, porttitor lacus at, posuere massa.
    </pre>
</body>
```

Below is the CSS code to illustrate the static property:

```
// css code
body
{
    margin: 0;
    padding: 20px;
    font-family: sans-serif;
    background: #efefef;
}
.static
{
    position: static;
    background: #cc0000;
    color: #ffffff;
    padding: 30px;
}
```

Keyword

Normal flow is the way that elements are displayed in a web page in most circumstances. All elements in HTML are inside boxes which are either inline boxes or block boxes.

```
span
{
    padding: 5px;
    border: 1px #ffffff dotted;
}
```

Keyword

HTML element is an individual component of an HTML (Hypertext Markup Language) document or web page, once this has been parsed into the Document Object Model.

5.1.3 Relative Positioning

Relative positioning changes the position of the **HTML element** relative to where it normally appears. So "left:20" adds 20 pixels to the element's LEFT position.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

- Move Left - Use a negative value for *left*.
- Move Right - Use a positive value for *left*.
- Move Up - Use a negative value for *top*.
- Move Down - Use a positive value for *top*.

NOTE – You can use *bottom* or *right* values as well in the same way as *top*and *left*.

Example

```
<html>
  <head>
  </head>

  <body>
      <div style = "position:relative; left:80px; top:2px; background-color:yellow;">
      This div has relative positioning.
    </div>
  </body>
  </html>
```

An element with **position: relative** is positioned relatively

with the other elements which are sitting at top of it. If we set its top, right, bottom or left, other elements will not fill up the gap left by this element.

```
<!-->html code<-->
<body>
<div class="relative">This div has
    <span>position: relative;</span></div>
<pre>
    Lorem ipsum dolor sits amet, consectetur adipiscing elit.
    Nunc eget mauris at urna hendrerit iaculis sit amet et ipsum.
    Maecenas nec mi eget leo malesuada vehicula.
    Nam eget velit maximus, elementum ante pretium, aliquet felis.
    Aliquam quis turpis laoreet, porttitor lacus at, posuere massa.
</pre>
</body>
```

Below is the CSS code to illustrate the relative property:

```
// css code
body
{
    margin: 0;
    padding: 20px;
    font-family: sans-serif;
    background: #efefef;
}

.relative
{
    position: relative;
    background: #cc0000;
    color: #ffffff;
```

Remember

A relatively positioned element can be moved and overlap other elements, but it keeps the space originally reserved for it in the normal flow.

```
    padding: 30px;
}

span
{
    padding: 5px;
    border: 1px #ffffff dotted;
}
```

5.1.4 Absolute Positioning

An element with **position: absolute** is positioned at the specified coordinates relative to your screen top-left corner.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

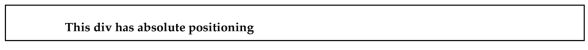
- Move Left - Use a negative value for *left*.
- Move Right - Use a positive value for *left*.
- Move Up - Use a negative value for *top*.
- Move Down - Use a positive value for *top*.

NOTE – You can use *bottom* or *right* values as well in the same way as top and left.

Here is an example –

```
<html>
   <head>
   </head>

   <body>
    <div style = "position:absolute; left:80px; top:20px; background-color:yellow;">
        This div has absolute positioning.
      </div>
   </body>
</html>
```

It will produce the following result –



An element with **position: absolute** will be positioned with respect to its parent. Positioning of this element does not depend upon its siblings or the elements which are at same level.

```
<!-->html code<-->
<body>
    <pre>
      Lorem ipsum dolor sits amet, consectetur adipiscing elit.
      Nunc eget mauris at urna hendrerit iaculis sit amet et ipsum.
      Maecenas nec mi eget leo malesuada vehicula.
      <div class=”relative”>
        <p>This div has <span><strong>position: relative;</strong>
                                                    </span></p>
        <div class=”absolute”>
          This div has <span><strong>position:
                        absolute;</strong></span>
        </div>
      </div>
      Nam eget velit maximus, elementum ante pretium, aliquet felis.
      Aliquam quis turpis laoreet, porttitor lacus at, posuere massa.
    </pre>
</body>
```

Below is the CSS code to illustrate the absolute property:

```
// css code
body
{
    margin: 0;
    padding: 20px;
    font-family: sans-serif;
    background: #efefef;
}

.absolute
{
    position: absolute ;
```

```
    background: #cc0000;
    color: #ffffff;
    padding: 30px;
    font-size: 15px;
    bottom: 20px;
    right: 20px;
}

.relative
{
    position: relative;
    background: #aad000;
    height: 300px;
    font-size: 30px;
    border: 1px solid #121212;
    text-align: center;
}

span
{
    padding: 5px;
    border: 1px #ffffff dotted;
}

pre
{
        padding: 20px;
        border: 1px solid #000000;
}
```

5.1.5 Sticky Positioning

Element with **position: sticky** and **top: 0** played a role between **fixed & relative** based on the position where it is placed. If the element is placed at the middle of the

document then when user scrolls the document, the sticky element starts scrolling until it touch the top. When it touches the top, it will be fixed at that place inspite of further scrolling. We can stick the element at bottom, with the **bottom** property.

```
<!-->html code<-->
<body>
    <pre>
     Lorem ipsum dolor sits amet, consectetur adipiscing elit.
      Nunc eget mauris at urna hendrerit iaculis sit amet et ipsum.
       Maecenas nec mi eget leo malesuada vehicula.
            <div class="sticky">
             This div has <span>position: sticky;</span>
            </div>
        Nam eget velit maximus, elementum ante pretium, aliquet felis.
          Aliquam quis turpis laoreet, porttitor lacus at, posuere massa.
    </pre>
</body>
```

Below is the CSS code to illustrate the sticky property:

```
// css code
body
{
    margin: 0;
    padding: 20px;
    font-family: sans-serif;
    background: #efefef;
}

.sticky
{
    position: sticky;
    background: #cc0000;
```

Keyword

Document is a written, drawn, presented, or memorialized representation of thought.

```
    color: #ffffff;
    padding: 30px;
        top: 10px;
        right: 50px;
}

span
{
    padding: 5px;
    border: 1px #ffffff dotted;
}

pre
{
        padding: 20px;
        border: 1px solid #000000;
}
```

Keyword

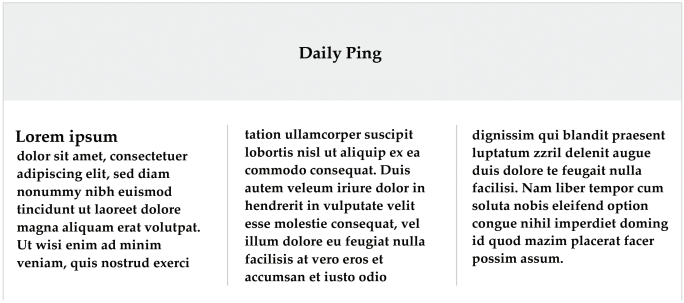
Column is one or more vertical blocks of content positioned on a page, separated by gutters (vertical whitespace) or rules (thin lines, in this case vertical).

5.2 MULTIPLE COLUMNS

Multi-column Layout is a module of CSS that adds support for multi-column layouts. Support is included for establishing the number of columns in a layout, as well as how content should flow from column to column, gap sizes between columns, and **column** dividing lines (known as column rules) along with their appearance.

5.2.1 Multi-column Layout

The CSS multi-column layout allows easy definition of multiple columns of text - just like in newspapers:



Browser Support

The numbers in the table specify the first browser version that fully supports the property.

Numbers followed by -webkit- or -moz- specify the first version that worked with a prefix.

Property					
column-count	50.0 4.0 - webkit-	10.0	52.0 2.0 -moz-	9.0 3.1 -webkit	37.0 15.0 -webkit 11.1
column-gap	50.0 4.0 - webkit-	10.0	52.0 2.0 -moz-	9.0 3.1 -webkit	37.0 15.0 -webkit 11.1
column-rule	50.0 4.0 - webkit-	10.0	52.0 2.0 -moz-	9.0 3.1 -webkit	37.0 15.0 -webkit 11.1
column-rule-color	50.0 4.0 - webkit-	10.0	52.0 2.0 -moz-	9.0 3.1 -webkit	37.0 15.0 -webkit 11.1
column-rule-style	50.0 4.0 - webkit-	10.0	52.0 2.0 -moz-	9.0 3.1 -webkit	37.0 15.0 -webkit 11.1
column-rule-width	50.0 4.0 - webkit-	10.0	52.0 2.0 -moz-	9.0 3.1 -webkit	37.0 15.0 -webkit 11.1
column-span	50.0 4.0 - webkit-	10.0	Not supported	9.0 3.1 -webkit	37.0 15.0 -webkit 11.1
column-width	50.0 4.0 - webkit-	10.0	52.0 2.0 -moz-	9.0 3.1 -webkit	37.0 15.0 -webkit 11.1

5.2.2 Multi-column Properties

In this section you will learn about the following multi-column properties:

- column-count

- column-gap
- column-rule-style
- column-rule-width
- column-rule-color
- column-rule
- column-span
- column-width

Create Multiple Columns

The column-count property specifies the number of columns an element should be divided into.

The following example will divide the text in the element into 3 columns:

Example

```
div {
    column-count: 3;
}
```

Specify the Gap between Columns

The column-gap property specifies the gap between the columns.

The following example specifies a 40 pixels gap between the columns:

Example

```
div {
    column-gap: 40px;
}
```

Column Rules

The column-rule-style property specifies the style of the rule between columns:

Example

```
div {
    column-rule-style: solid;
```

```
}
```

The column-rule-width property specifies the width of the rule between columns:

Example

```
div {
    column-rule-width: 1px;
}
```

The column-rule-color property specifies the color of the rule between columns:

Example

```
div {
    column-rule-color: lightblue;
}
```

The column-rule property is a shorthand property for setting all the column-rule-* properties above.

The following example sets the width, style, and color of the rule between columns:

Example

```
div {
    column-rule: 1px solid lightblue;
}
```

Remember

The width of column rule does not affect the width of column boxes, but if a column rule is wider than the gap, the adjacent column boxes will overlap the rule.

Specify How Many Columns an Element Should Span

The column-span property specifies how many columns an element should span across.

The following example specifies that the <h2> element should span across all columns:

Example

```
h2 {
    column-span: all;
}
```

Specify the Column Width

The column-width property specifies a suggested, optimal width for the columns.

The following example specifies that the suggested, optimal width for the columns :should be 100px

Example

```
div {
    column-width: 100px;
}
```

The following table lists all the multi-columns properties:

Property	Description
column-count	Specifies the number of columns an element should be divided into
column-fill	Specifies how to fill columns
column-gap	Specifies the gap between the columns
column-rule	A shorthand property for setting all the column-rule-* properties
column-rule-color	Specifies the color of the rule between columns
column-rule-style	Specifies the style of the rule between columns
column-rule-width	Specifies the width of the rule between columns
column-span	Specifies how many columns an element should span across
column-width	Specifies a suggested, optimal width for the columns
columns	A shorthand property for setting column-width and column-count

CASE STUDY

POSITIONING IN CSS

Most of the work we have done with CSS is fairly simple: changing fonts, colours, spacing, and borders. All of those changes are useful and necessary, but not the *hard* part of CSS.

Things become more interesting when you start moving stuff around the page.

The float Property

The easiest way to move content is the **float** property. It will take content and move it to the left or right sides of the page.

When floated content moves, whatever content follows it will move up, and flow around it. That means that the floated content will move to the side of the page, and other text will avoid overlapping with it.

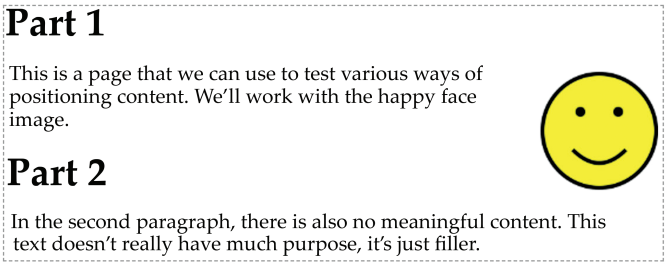
As an example of **float** (and later CSS properties in this topic), we will use an HTML page structured like this:

```
<section id="p1">
<h2>Part 1</h2>
<figure id="happy">…</figure>
<p>…</p>
</section>
<section id="p2">
<h2>Part 2</h2>
<p>…</p>
</section>
```

To illustrate how **float** works, we will apply this CSS:

```
#happy {
  float: right;
}
```

The resulting page will look like this:



You can see this example in your browser if you like. You can select the different stylesheets described here, and experiment with the width of the content to see how that affects the layout.

As you can see, the **<figure>** is moved to the right of the page, and text that comes after it avoids overlapping it, but below the figure, it extends back to the right side of the page.

Also note: the vertical position of the figure hasn't changed: it is immediately below the first **<h2>**, where you'd expect it. It only moved to the right. The following paragraph moved up and had its lines re-wrapped.

The clear Property

When floating some content like that, it's common to not necessarily know exactly what will follow. If you have a large floating block, it may be difficult to predict how it will interact with the content that follows.

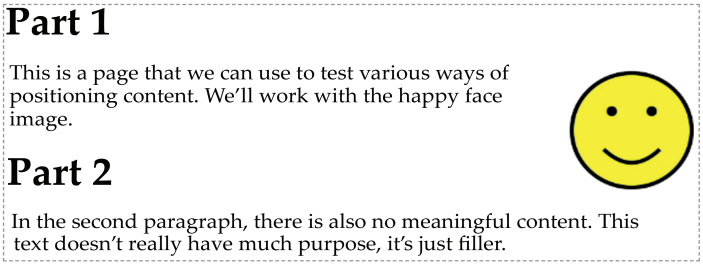
In the above example, you can see that the second section starts while the floating content is still to its right. It would be unusual to start another section of the page while the content from the previous section is still there: it would be nice to move the following content down below the floating figure.

The **clear** property does just this: it tells the browser that you want to move a piece of content *down* until there is nothing floating beside it: either to the left, right, or both sides.

To solve the problem of starting a new section with content floating content still beside it, we can do something like this in CSS:

```
#happy {
  float: right;
}
section {
  clear: both;
}
```

It will look like this in the browser, with the following **<section>** moved down until there is nothing floating on either its left or right:



We could have also specified **clear: right** in this case, since there is nothing floating on the left. It's probably more complete to specify **both**: we wouldn't want to start a section with anything floating on the left if we did that later on some other page. You can experiment with this example as well.

The **float** and **clear** properties are good at what they do. There are, however, many visual design ideas that you could want to implement which are very difficult to map onto float operations. In those cases, we need to reach for a more powerful tool…

The position Property

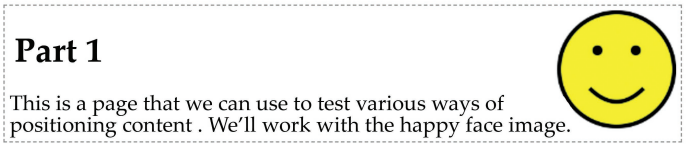
The **position** property is both more powerful and more dangerous than **float** and **clear**.

Setting **position: absolute** on an element lets you use the CSS properties **top**, **bottom**, **left**, and **right** to move the element around the page to exactly where you want it. For example, setting **top: 1em** move the element so its top is **1em** from the top of the page.

This sounds easy. For example, we can move the image to the top-right of the page like this:

```
#happy {
  position: absolute;
  top: 0;
  right: 0;
}
```

That will display like this. You can experiment with this example and the following ones as well.

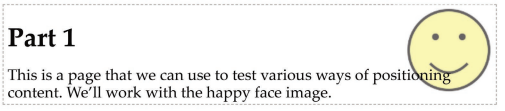


You can see the problem with **position**: as soon as you start positioning content, it is taken out of the normal flow of content on the page and will overlap other content. This is something you have to be **very** aware of when using **position**: you need to think of all of the possible ways your content can overlap on different devices and window sizes.

There are still some ways absolute positioning can be useful. For example, we could allow overlap, but make it look okay with **opacity** to make the content translucent.

```
#happy {
  position: absolute;
  top: 0;
  right: 0;
  opacity: 0.33;
  z-index: -1;
}
```

The negative **z-index** value moves the figure *behind* the other content. That seems reasonable: we can allow content to overlap, but it's okay because of the other styling done to it:

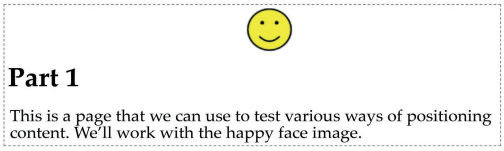


Another good use of absolute positioning: moving content around when you know *exactly* how much space you have to work with. For example, here we will move the figure up to the top of the page: this technique might be useful for a menu or page header (with page title, logo, and menu).

```
#happy {
  position: absolute;
  top: 0.5em;
  left: 0;
  width: 100%;
```

```
  margin: 0;
  text-align: center;
}
#happy img {
  width: 3em;
  height: 3em;
}
main>section:first-child {
  margin-top: 4em;
}
```

Here, the figure is at the top of the page, taking its entire width. The image (only content in it) is sized it's exactly **3em** high.



There will be no overlap as long as we make sure the following content has a top margin **3em** or larger. Selecting that first **<section>** was tricky because of the way the HTML was structured, but it does express the intention here.

There are other values that can be given to **position**. The default is **static**. Using **position: relative** works like **static** but you can adjust the position of the element relative to where it would have naturally been.

Setting **position: fixed** is like **absolute**, but relative to the window: fixed things won't move as the user scrolls. There is even more opportunity for unwanted content overlap that way.

The position you give (with **top**, **left**, etc.) for an absolutely positioned element is usually relative to the entire page: the top of the entire page, left of the page, etc. This is not true if any element that contains the positioned one is itself positioned. This trick has been used on the example page to get things floating/positioned within the example: the **<main>** has **position: relative** applied, not to move it but to make it contain floated/positioned thing within it.

Positioning Difficulties

As you may have noticed in this section (and while doing the assignments), positioning things the way you want in CSS can be difficult.

Using **float** is easy enough if you're doing what it was designed for: floating figures beside other content. If you are lucky enough to know exactly how big your .content is, then **position** can be used without too much trouble

There are many other ways you might want to arrange content though, where isn't not immediately obvious how to achieve them with float or position: a menu in a column to the left of other content; a photo gallery with three columns of images (but 2 on smaller screens); links styled like "tabs" above a section of content; and .many more

SUMMARY

- The position property in CSS tells about the method of positioning for an element or an HTML entity. There are five different types of position property available in CSS: Fixed, Static, Relative, Absolute and Sticky.
- The positioning of an element can be done using the top, right, bottom and left property. These specify the distance of an HTML element from the edge of the viewport.
- Fixed positioning allows you to fix the position of an element to a particular spot on the page, regardless of scrolling.
- Relative positioning changes the position of the HTML element relative to where it normally appears. So "left:20" adds 20 pixels to the element's LEFT position.
- An element with position: absolute is positioned at the specified coordinates relative to your screen top-left corner.
- Multi-column Layout is a module of CSS that adds support for multi-column layouts. Support is included for establishing the number of columns in a layout, as well as how content should flow from column to column, gap sizes between columns, and column dividing lines (known as column rules) along with their appearance.

KNOWLEDGE CHECK

1. **Which of the following CSS Property controls how an element is positioned?**
   a. position
   b. set
   c. static
   d. fix
2. **Which of the following CSS Property specifies the top offset of a positioned element?**
   a. top
   b. up
   c. reverse
   d. fix
3. **Which of the following CSS Property Specifies the left offset of a positioned element?**
   a. right
   b. left
   c. bottom
   d. up
4. **Which of the following CSS Property sets the stacking order of positioned elements?**
   a. x-index
   b. y-index
   c. z-index
   d. all of the mentioned
5. **Which of the following CSS Property Defines the area of an absolutely positioned element that remains visible?**
   a. clamp
   b. clip
   c. visibility
   d. static
6. **How many types of position values are there in CSS?**
   a. 2
   b. 3

c. 4

d. 5

7. Positioning attributes top, bottom, left and right are used to determine the exact location

a. TRUE

b. FALSE

c. Can be true or false

d. Can not say

8. In Absolute position, If there is no parent element, these elements are set to be ..... to the page.

a. absolute

b. static

c. fixed

d. relative

9. Reverse CSS Property specifies the top offset of a positioned element.

a. TRUE

b. FALSE

c. Can be true or false

d. Can not say

10. The default for any page element is?

a. Fixed

b. relative

c. absolute

d. Static

REVIEW QUESTIONS

1. Describe the different types of position property.
2. How to do absolute positioning with CSS?
3. How to create multiple columns layout? Discuss.
4. Specify the gap between columns.
5. Specify how many columns an element should span.

Check Your Result

1. (a)	2. (a)	3. (b)	4. (c)	5. (b)
6. (c)	7. (a)	8. (d)	9. (b)	10. (d)

REFERENCES

1. Bos, Bert (18 February 2011). "Descriptions of all CSS specifications". World Wide Web Consortium. Archived from the original on 31 March 2011. Retrieved 3 March 2011.
2. Bos, Bert (26 February 2011). "CSS current work". World Wide Web Consortium. Archived from the original on 3 March 2011. Retrieved 3 March 2011.
3. Bos, Bert (26 February 2011). "CSS current work". World Wide Web Consortium. Retrieved 3 March 2011.
4. Dan Cederholm (2009): Web Standards Solutions, The Markup and Style Handbook, Friends of Ed, ISBN 978- 1430219200 (paperback) (Author's site)
5. Etemad, Elika J. (12 December 2010). "Cascading Style Sheets (CSS) Snapshot 2010". World Wide Web Consortium. Archived from the original on 16 March 2011. Retrieved 3 March 2011.
6. Etemad, Elika J. (12 December 2010). "Cascading Style Sheets (CSS) Snapshot 2010". World Wide Web Consortium. Retrieved 3 March 2011.
7. Keith Schengili-Roberts (2003): Core CSS, 2nd Edition, Prentice Hall, ISBN 0-13-009278-9
8. Meyer, Eric A. (2001) Cascading Style Sheets 2.0 Programmer's Reference, McGraw-Hill Osborne Media, ISBN 0-07-213178-0
9. Nitot, Tristan (18 March 2002). "Incorrect MIME Type for CSS Files". Mozilla Developer Center. Mozilla. Archived from the original on 2011-05-20. Retrieved 20 June 2010.
10. W3C: Cascading Style Sheets level 1 specification Archived 2011-02-11 at the Wayback Machine CSS level 1 specification
11. W3C: Cascading Style Sheets, level 1 Archived 2011-02-09 at the Wayback Machine CSS 1 specification
12. W3C: Cascading Style Sheets, level 2 Archived 2011-01-16 at the Wayback Machine CSS 2 specification (1998 recommendation)
13. W3C:Cascading Style Sheets, level 2 revision 1 Archived 2011-11-09 at the Wayback Machine CSS 2.1 specification (W3C Proposed Recommendation)

CHAPTER

6

CSS TEXT: TYPEFACE AND FONTS

"Make it as simple as possible, but not simpler."

– Albert Einstein

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Explain the CSS font-face Rule
2. Focus on CSS Font Property
3. Discuss about Web Safe Fonts

INTRODUCTION

CSS Text is a module of CSS that defines how to perform text manipulation, like line breaking, justification and alignment, white space handling, and text transformation.

Text and font properties in the Cascading Style Sheets serve to set the appearance of individual characters in a word or line of text. The main problems are unavailable fonts and mapping of (variant) glyphs to character positions.

Font properties			
property	**type**	**default**	**comments**
font	*URI*	inherited (-)	overrides the next 6 below
font.family	*string*+	inherited (-)	
font.weight	..., -2, -1, 0, 1, 2,...	inherited (0)	1=bold, 2=extra-bold, etc.
font.expansion	..., -2, -1, 0, 1, 2,...	inherited (0)	0=normal, 1=expanded, etc.
font.style	(italic | slanted | roman | small-caps)+	inherited (roman)	
font.size	..., -2, -1, 0, 1, 2,... | *length*	0 (-)	*n*=1.2^*n* times parent's size; *length*in pt
encoding	*encoding*	inherited (none)	Postscript-like encoding vector

Selecting a font can either be done with the font property or with the six partial font properties. If font is present, it overrides the other font properties and the application is expected to extract the proper values for them from the single font as well as it can.

Where the column default has a value in parentheses (), it is the value that is the default for the root element (which cannot inherit anything). A value (-) means that the default value is application dependent.

The table says that font.size is not inherited, but the default value, 0, effectively means that the size is the same as the parent's font size. So, the relative size is not inherited, but the absolute size is.

The application is should do its best to find fonts that closely match the given properties, but it will not always succeed perfectly. How fonts and/or styles are substituted depends on the font, the user's preferences and the applications intelligence.

The text.encoding property holds a list of zero or more mappings from characters (or character codes, i.e., numbers) to glyph identifiers. The glyph identifiers are the Adobe/Postscript glyph names (when available, otherwise AFII numbers). All characters not mentioned in this encoding have their default mapping, which depends on the font.

The font.family and font properties can be lists of one or more fonts. The meaning of this is twofold: (1) when a font is unavailable, the next font in the list is tried, and (2) when a character is not available in one font, the next font is tried. This is particalarly important with multilingual documents (which use Unicode as character set), for many fonts will only contain a small set of glyphs

6.1 CSS FONT-FACE RULE

With the @font-face rule, web designers do not have to use one of the "web-safe" fonts anymore. In the @font-face rule you must first define a name for the font (e.g. myFirstFont), and then point to the font file.

Tip: Use lowercase letters for the font URL. Uppercase letters can give unexpected results in IE! To use the font for an **HTML element**, refer to the name of the font (myFirstFont) through the font-family property:

```
div {
    font-family: myFirstFont;
}
```

Browser Support

The @font-face rule is supported in Internet Explorer, Firefox, Opera, Chrome, and Safari.

The numbers in the table specifies the first browser version that fully supports the font format

Keyword

An **HTML element** is an individual component of an HTML (Hypertext Markup Language) document or web page, once this has been parsed into the Document Object Model.

Font format					
TTF/OTF	4.0	9.0*	3.5	3.1	10.0
WOFF	5.0	9.0	3.6	5.1	11.1
WOFF2	36.0	Not supported	35.0*	Not supp orted	26.0
SVG	4.0	Not supported	Not supp orted	3.2	9.0
EOT	Not supp orted	6.0	Not supp orted	Not supp orted	Not sup ported

*Edge and IE: The font format only works when set to be "installable".

*Firefox: Disabled by default, but can be enabled (need to set a flag to "true" to use WOFF2).

Syntax

```
@font-face {
    font-properties
}
```

Font descriptor	Values	Description
font-family	*name*	Required. Defines the name of the font.
src	*URL*	Required. Defines the URL(s) where the font should be downloaded from
font-stretch	normal condensed ultra-condensed extra-condensed semi-condensed expanded semi-expanded extra-expanded ultra-expanded	Optional. Defines how the font should be stretched. Default value is "normal"
font-style	normal italic oblique	Optional. Defines how the font should be styled. Default value is "normal"
font-weight	normal bold 100 200 300 400 500 600 700 800 900	Optional. Defines the boldness of the font. Default value is "normal"
unicode-range	*unicode-range*	Optional. Defines the range of unicode characters the font supports. Default value is "U+0-10FFFF"

More Examples

Example

You must add another @font-face rule containing descriptors for bold text:

```
@font-face {
    font-family: myFirstFont;
    src: url(sansation_bold.woff);
    font-weight: bold;
}
```

The file "sansation_bold.woff" is another font file that contains the bold characters for the Sansation font.

Browsers will use this whenever a piece of text with the font-family "myFirstFont" should render as bold.

This way you can have many @font-face rules for the same font.

Difference between Serif and Sans-serif Fonts

6.1.1 CSS Font Families

In CSS, there are two types of font family names:

- **generic family** - a group of font families with a similar look (like «Serif» or «Monospace»)
- **font family** - a specific font family (like «Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

Note: On computer screens, sans-serif fonts are considered easier to read than serif fonts.

6.1.2 Font-Family

The **font-family** CSS property specifies a prioritized list of one or more font family names and/or generic family names for the selected element.

font-family: Georgia, serif;

Copy to Clipboard

font-family: “Gill Sans”, sans-serif;

font-family: sans-serif;

font-family: serif;

font-family: cursive;

font-family: system-ui;

Values are separated by commas to indicate that they are alternatives. The browser will select the first font in the list that is installed or that can be downloaded using a @ font-face at-rule.

It is often convenient to use the shorthand property font to set font-size and other font related properties all at once.

You should always include at least one **generic family** name in a font-family list, since there’s no guarantee that any given font is available. This lets the browser select an acceptable fallback font when necessary.

The font-family property specifies a list of fonts, from highest priority to lowest. Font selection does not simply stop at the first font in the list that is on the user’s system. Rather, font selection is done one character at a time, so that if an available font does not have a glyph for a needed character, the latter fonts are tried. (However, this doesn’t work in Internet Explorer 6 or earlier.) When a font is only available in some styles, variants, or sizes, those properties may also influence which font family is chosen.

Keyword

Generic-family: The name of a generic-family, like “serif”, “sans-serif”, “cursive”, “fantasy”, “monospace”.

Syntax

/* A font family name and a generic family name */

font-family: Gill Sans Extrabold, sans-serif;

font-family: “Goudy Bookletter 1911”, sans-serif;

/* A generic family name only */

font-family: serif;

font-family: sans-serif;

font-family: monospace;

font-family: cursive;

font-family: fantasy;

font-family: system-ui;

/* Global values */

font-family: inherit;

font-family: initial;

font-family: unset;

The font-family property lists one or more font families, separated by commas. Each font family is specified as either a <family-name> or a <generic-name> value.

The example below lists two font families, the first with a <family-name> and the second with a <generic-name>:

font-family: Gill Sans Extrabold, sans-serif;

Values

<family-name>

The name of a font family. For example, "Times" and "Helvetica" are font families. Font family names containing whitespace should be quoted.

<generic-name>

Generic font families are a fallback mechanism, a means of preserving some of the style sheet author's intent when none of the specified fonts are available. Generic family names are keywords and must not be quoted. A generic font family should be the last item in the list of font family names.

The following keywords are defined:

serif

Glyphs have finishing strokes, flared or tapering ends, or have actual serifed endings.

E.g. Lucida Bright, Lucida Fax, Palatino, "Palatino Linotype", Palladio, "URW Palladio", serif.

sans-serif

Glyphs have stroke endings that are plain.

E.g. "Open Sans", "Fira Sans", "Lucida Sans", "Lucida Sans Unicode", "Trebuchet MS", "Liberation Sans", "Nimbus Sans L", sans-serif.

monospace

All glyphs have the same fixed width.

E.g. "Fira Mono", "DejaVu Sans Mono", Menlo, Consolas, "Liberation Mono", Monaco, "Lucida Console", monospace.

cursive

Glyphs in cursive fonts generally have either joining strokes or other cursive characteristics beyond those of italic typefaces. The glyphs are partially or completely connected, and the result looks more like handwritten pen or brush writing than printed letter work.

E.g. "Brush Script MT", "Brush Script Std", "Lucida Calligraphy", "Lucida Handwriting", "Apple Chancery", cursive.

fantasy

Fantasy fonts are primarily decorative fonts that contain playful representations of characters.

E.g. Papyrus, Herculanum, Party LET, Curlz MT, Harrington, fantasy.

system-ui

Glyphs are taken from the default user interface font on a given platform. Because typographic traditions vary widely across the world, this generic is provided for typefaces that don't map cleanly into the other generics.

Remember

If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".

More than one font family is specified in a comma-separated list:

Example

```
p {
font-family: "Times New Roman", Times, serif;}
}
```

Font Style

The font-style property is mostly used to specify italic text.

This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Example

```
p.normal {
font-style: normal;
}
```

```
p.italic
{
 font-style: italic;
}
p.oblique {
 font-style: oblique;
}
```

6.1.3 Font Size

The font-size property sets the size of the text.

Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.

The font-size value can be an absolute, or relative size.

:Absolute size

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in **browsers**

Set Font Size with Pixels

Setting the text size with pixels gives you full control over the text size:

Example

```
h1 {
    font-size: 40px;
}
```

Keyword

A **browser** is software that is used to access the internet. A browser lets you visit websites and do activities within them like login, view multimedia, link from one site to another, visit one page from another, print, send and receive email, among many other activities.

```
h2 {
    font-size: 30px;
}

p {
    font-size: 14px;
}
```

Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.

The em size unit is recommended by the W3C.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: *pixels*/16=*em*

Example

```
h1 {
    font-size: 2.5em; /* 40px/16=2.5em */
}
h2 {
    font-size: 1.875em; /* 30px/16=1.875em */
}
p {
    font-size: 0.875em; /* 14px/16=0.875em */
}
```

In the example above, the text size in em is the same as the example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

Unfortunately, there is still a problem with older versions of IE. The text becomes larger than it should when made larger, and smaller than it should when made smaller.

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

Example

```
body {
    font-size: 100%;
}

h1 {
    font-size: 2.5em;
}

h2 {
    font-size: 1.875em;
}

p {
    font-size: 0.875em;
}
```

Our code now works great! It shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

Font Weight

The font-weight property specifies the weight of a font:

Example

```
p.normal {
    font-weight: normal;
}

p.thick {
    font-weight: bold;
}
```

Responsive Font Size

The text size can be set with a vw unit, which means the «viewport width».

That way the text size will follow the size of the browser window:

Hello World

Resize the browser window to see how the font size scales.

Example

```
<h1 style="font-size:10vw">Hello World</h1>
```

Font Variant

The font-variant property specifies whether or not a text should be displayed in a small-caps font. In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

Example

```
p.normal {
    font-variant: normal;
}
p.small {
    font-variant: small-caps;
}
```

All CSS Font Properties

Property	Description
font	Sets all the font properties in one declaration
font-family	Specifies the font family for text
font-size	Specifies the font size of text
font-style	Specifies the font style for text
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-weight	Specifies the weight of a font

6.2 CSS FONT PROPERTY

Example

Set some font properties with the shorthand declaration:

```
p.a {
    font: 15px arial, sans-serif;
}
p.b {
    font: italic bold 12px/30px Georgia, serif;
}
```

More "Try it Yourself" examples below.

Definition and Usage

The font property is a shorthand property for:

- font-style
- font-variant
- font-weight
- font-size/line-height
- font-family

The font-size and font-family values are required. If one of the other values is missing, their default value are used.

Default value:	*The default value of the font properties*
Inherited:	yes
Animatable:	yes, *see individual properties*. Read about *animatable*Try it
Version:	CSS1
JavaScript syntax:	*object*.style.font="italic small-caps bold 12px arial,sans-serif"

Browser Support

The numbers in the table specify the first browser version that fully supports the property.

Property					
font	1.0	4.0	1.0	1.0	3.5

CSS Syntax

font: *font-style font-variant font-weight font-size/line-height font-family* | caption | icon | menu | message-box | small-caption | status-bar | initial | inherit;

Property Values

Property/Value	**Description**
font-style	Specifies the font style. Default value is "normal"
font-variant	Specifies the font variant. Default value is "normal"
font-weight	Specifies the font weight. Default value is "normal"
font-size/line-height	Specifies the font size and the line-height. Default value is "normal"
font-family	Specifies the font family. Default value depends on the browser
caption	Uses the font that are used by captioned controls (like buttons, drop-downs, etc.)
icon	Uses the font that are used by icon labels
menu	Uses the fonts that are used by dropdown menus
message-box	Uses the fonts that are used by dialog boxes
small-caption	A smaller version of the caption font
status-bar	Uses the fonts that are used by the status bar
initial	Sets this property to its default value. Read about *initial*
inherit	Inherits this property from its parent element.

More Examples

Example

A demonstration of some of the other font property values.

```
<p style="font:caption">The browser font used in captioned controls.</p>
<p style="font:icon">The browser font used in icon labels.</p>
<p style="font:menu">The browser font used in dropdown menus.</p>
<p style="font:message-box">The browser font used in dialog boxes.</p>
<p style="font:small-caption">A smaller version of the caption font.</p>
<p style="font:status-bar">The browser font used in the status bar.</p>
```

6.2.1 CSS font-style Property

Example

Set different font styles for three paragraphs:

```
p.a {
    font-style: normal;
}

p.b {
    font-style: italic;
}

p.c {
    font-style: oblique;
}
```

Definition and Usage

The font-style property specifies the font style for a text.

Default value:	normal
Inherited:	yes
Animatable:	no. Read about *animatable*
Version:	CSS1
JavaScript syntax:	*object*.style.fontStyle="italic"

Browser Support

The numbers in the table specify the first browser version that fully supports the property.

Property					
font	1.0	4.0	1.0	1.0	7.0

CSS Syntax

font-style: normal | italic | oblique | initial | inherit;

Property Values

Value	Description	Play it
normal	The browser displays a normal font style. This is default	Play it »
italic	The browser displays an italic font style	Play it »
oblique	The browser displays an oblique font style	Play it »
initial	Sets this property to its default value. Read about *initial*	Play it »
inherit	Inherits this property from its parent element.	

The font property in CSS is a shorthand property that combines all the following sub-properties in a single declaration.

```
body {
  font: normal small-caps normal 16px/1.4 Georgia;
}

/* is the same as:

body {
  font-family: Georgia;
  line-height: 1.4;
  font-weight: normal;
  font-stretch: normal;
```

```
  font-variant: small-caps;
  font-size: 16px;
}
 */
```

There are seven font sub-properties, including:

- font-stretch: this property sets the font width, such as condensed or expanded.
  - normal
  - ultra-condensed
  - extra-condensed
  - condensed
  - semi-condensed
  - semi-expanded
  - expanded
  - extra-expanded
  - ultra-expanded
- font-style: makes the text appear italicized or oblique.
  - normal
  - italic
  - oblique
  - inherit
- font-variant: changes target text to small caps.
  - normal
  - small-caps
  - inherit
- font-weight: sets the weight or the thickness of the font.
  - normal
  - bold
  - bolder
  - lighter
  - 100, 200, 300, 400, 500, 600, 700, 800, 900
  - inherit
- font-size: sets the height of the font.
  - xx-small

- x-small
- small
- medium
- large
- x-large
- xx-large
- smaller, larger
- percentage
- inherit

- line-height: defines the amount of space above and below inline elements.
  - normal
  - number (font-size multiplier)
  - percentage
- font-family: definies the font that is applied to the element.
  - sans-serif
  - serif
  - monospace
  - cursive
  - fantasy
  - caption
  - icon
  - menu
  - **message-box**
  - small-caption
  - status-bar
  - "string"

Keyword

The **message box** returns an integer value that indicates which button the user clicked.

Font Shorthand Gotchas

The font property is not as straightforward as other shorthand properties, partly due to the syntax requirements for it, and partly due to inheritance issues.

Here is a summary of some of the things you should know when using this shorthand property.

Mandatory Values

Two of the values in font shorthand are mandatory: font-size and font-family. If either of these is not included, the entire declaration will be ignored.

Also, font-family must be declared last of all values, otherwise, again, the entire declaration will be ignored.

Optional Values

All five of the other values are optional. If you include any of font-style, font-variant, and font-weight, they must come before font-size in the declaration. If they aren't, they will be ignored and may also cause the mandatory values to be ignored.

```
body {
    font: italic small-caps bold 44px Georgia, sans-serif;
}
```

In the above example, three optional are included. As long as these are defined before font-size, they can be placed in any order.

Including line-height is likewise optional but may be declared only after font-size and only following a forward slash:

```
body {
    font: 44px/20px Georgia, sans-serif;
}
```

In the above example, the line-height is "20px". If you omit line-height, you must also omit the slash, otherwise the entire line will be ignored.

Using font-stretch

The font-stretch property is new in CSS3 so if it is used in an older browser that doesn't support font-stretch in font shorthand, it will cause the entire line to be ignored.

The spec recommends including a fallback without font-stretch, like this:

```
body {
   font: italic small-caps bold 44px Georgia, sans-serif; /* fallback for older browsers */
    font: ultra-condensed italic small-caps bold 44px Georgia, sans-serif;
}
```

Inheritance for optional

If you omit any of the optional values (including line-height), the omitted optional will not inherit values from their parent element, as is often the case with typographical properties. Instead, they will be reset to their initial state.

For example:

```
body {
   font: italic small-caps bold 44px/50px Georgia, sans-serif;
}

p {
    font: 30px Georgia, sans-serif;
}
```

Keyword

The usage of the **initial value** depends on whether a property is inherited or not: For inherited properties, the initial value is used on the root element only, as long as no specified value is supplied.

The optional values (italic, small-caps, and bold) are placed on the font declaration on the <body> element. These will also apply to most child elements.

However, because we've declared the font property on the paragraph elements, all the optional will be reset on the paragraphs, causing the style, variant, weight, and line-height to revert to their **initial values**.

Keywords for Defining System Fonts

In addition to the above syntax, the font property also allows use of keywords as values. They are:

- caption
- icon
- menu
- message-box
- small-caption
- status-bar

These keyword values set the font to the one that is used on the user's operating system for that particular category. For example, defining "caption" will set the font on that element to use the same font that is used on the operating system for captioned controls (buttons, drop-downs, etc).

A single keyword comprises the entire value:

```
body {
    font: menu;
}
```

The other properties mentioned earlier are not valid in conjunction with these keywords. These keywords can only be used with font shorthand and cannot be declared using any of the individual longhand properties.

The font-style property allows you to make text appear italicized (i.e. sloped, or slanted).

```
em {
  font-style: italic;
}
```

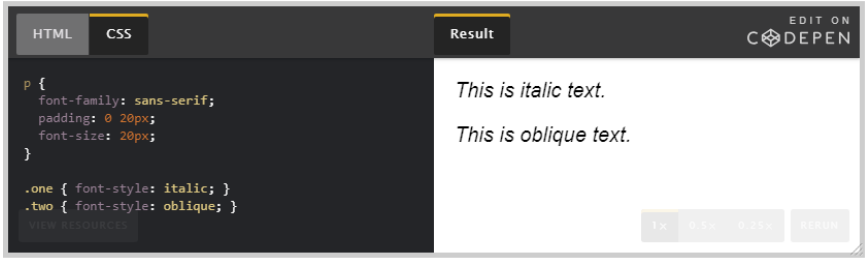
This property accepts one of three possible values: normal, italic, and oblique, If a given font family has an italic or oblique face embedded, the browser will select that face. If no italic or oblique face is available, the browser will mimic the sloping effect. If italic is defined and there is no italic face available, the browser will look for an oblique face before it tries to mimic the italic. The same applies to oblique; it too will look first for an italic face.

To prevent the browser from adding a mimicked, or synthesized, version of the italic or oblique face, you may use the font-synthesis property (if supported).

Italic vs. Oblique

According to the spec, "Italic forms are generally cursive in nature while oblique faces are typically sloped versions of the regular face." However, if the font being used does not have italic or oblique faces available, in most cases there is little, if any, difference between italic and oblique.

The demo below demonstrates both italic and oblique:



For Examples

```
/* Set the font size to 12px and the line height to 14px.
   Set the font family to sans-serif */
p { font: 12px/14px sans-serif }
/* Set the font size to 80% of the parent element
   or default value (if no parent element present).
   Set the font family to sans-serif */
p { font: 80% sans-serif }
/* Set the font weight to bold,
   the font-style to italic,
   the font size to large,
   and the font family to serif. */
p { font: bold italic large serif }
/* Use the same font as the status bar of the window */
p { font: status-bar }
```

6.3 WEB SAFE FONTS

Speaking of font availability, there are only a certain number of fonts that are generally available across all systems, and can therefore be used without much worry. These are the so-called **web safe fonts**.

Most of the time, as web developers we want to have more specific control over the fonts used to display our text content. The problem is to find a way to know which font is available on the computer used to see our web pages. There is no way to know this in every case, but the web safe fonts are known to be available on nearly all instances of the most used operating systems (Windows, Mac, the most common Linux distributions, Android, and iOS.)

The list of actual web safe fonts will change as operating systems evolve, but it's okay to consider the following fonts web safe, at least for now (many of them have been popularized thanks to the Microsoft *Core fonts for the Web* initiative in the late 90s and early 2000s):

Name	Generic type	Notes
Arial	sans-serif	It's often considered best practice to also add *Helvetica* as a preferred alternative to *Arial* as, although their font faces are almost identical, *Helvetica* is considered to have a nicer shape, even if *Arial* is more broadly available.
Courier New	monospace	Some OSes have an alternative (possibly older) version of the *Courier New* font called *Courier*. It's considered best practice to use both with *Courier New* as the preferred alternative.
Georgia	serif	
Times New Roman	serif	Some OSes have an alternative (possibly older) version of the *Times New Roman* font called *Times*. It's considered best practice to use both with *Times New Roman* as the preferred alternative.
Trebuchet MS	sans-serif	You should be careful with using this font — it isn't widely available on mobile OSes.
Verdana	sans-serif	

Font embedding services (like Google Web Fonts or Typekit) sprung up as an alternative, giving your designs something .new, fresh, and unexpected

They're also super easy to use.

Take Google for example:

Choose any font like Open Sans or Droid Serif or Lato. Generate the code and paste in your document's <Head>. And you're all set to reference it in CSS!

That took 60 seconds. And it was completely free. (Thanks, Google!)

What could go wrong, right?

Not everyone will have access to that same font. Which means you will have a problem. That beautiful font you just chose is going to show up as something random for your visitors.

Not if you create a fallback with a web safe alternative!

Here's how it works.

Remember

Among various resources, the cssfontstack.com website maintains a list of web safe fonts available on Windows and macOS operating systems, which can help you make your decision about what you consider safe for your usage.

Why Does 'Web Safe' Matter?

Each device comes with its own pre-installed font selection. The selection is based largely on its operating system.

The problem is that every system differs a bit.

Windows-based devices might have one group. MacOS ones pull from another. Google's own Android system uses their own as well.

Now pull up a website. Even this one would work. The font you see may not be the one original one intended.

Meaning: Let's say the designer picked some obscure, paid font family for this site's design. If you don't have that font already installed and it's not pulling from a web-friendly place – more on that later – the font *you see* would default back to some basic variation like Times New Roman.

You, as the visitor, wouldn't necessarily know that this is what has happened, though. For you, it might just look plain ugly.

Did You Know?

On December 17, 1996, W3C published the first standard for CSS. And thus from December 17, 2016 until one year later, CSS is 20 years old.

The 'Web safe' ones, appear across all operating systems. They're the small collection of fonts that overlap from Windows to Mac to Google (even UNIX or Linux ones too).

They give designers (and website owners) the ability to specify which fonts to *fall back to* if needed. That way, you can control what shows up (no matter what) across all devices. And you can pick something that's still *kinda* close to the original font (so that what your users wouldn't see something random or out of place).

It's a plan B, the 'just-in-case' version. An emergency system to save the world from bad font selections.

Got it? Good! Let's take a look at the most popular web safe fonts to choose from.

6.3.1 Best Web Safe Fonts

There might be a few more.

But these are the best 15 web safe fonts to choose from. Select one of these and you can't go wrong.

1. Arial

Arial Header

Arial body copy.

Arial is like the de facto standard for most.

It's one of the most widely used sans-serif fonts (which means no little curls on the end of each letter). It's often substituted on Windows devices for other interesting (read: more beautiful) font choices.

2. Helvetica

Helvetica Header

Helvetica body copy.

Helvetica is usually the designers' go-to sans serif font. You can almost never go wrong with Helvetica (or at least using it as a fallback for most other choices).

3. Times New Roman

Times New Roman Header

Times New Roman body copy.

Times New Roman is to serif what Arial is to sans serif.

It's among the most popular on Windows devices and is a new variation on the old Times font.

4. Times

Times Header

Times body copy.

The Times font probably looks familiar. It's the old newspaper print that you're used to seeing in a small size in narrow columns. It's about as traditional as it gets.

5. Courier New

Courier New Header

Courier New body copy.

Keyword

Times New Roman is a serif typeface designed for legibility in body text.

6. Courier

Courier Header

Courier body copy.

Courier is the old moonscape stand-by available on almost all devices and operating systems.

7. Verdana

Verdana Header

Verdana body copy.

Verdana is a true web font because (1) the simple sans serif lines and (2) it's super large size. The letters are almost elongated, which makes it easy to read online.

8. Georgia

Georgia Header

Georgia body copy.

Georgia is similar to Verdana in size and stature (with bigger-than-usual letters compared with fonts of the same size). So while it's great for certain circumstances, make sure to avoid pairing this serif font with others (like Times New Roman) which might look minuscule in comparison

9. Palatino

Palatino Header

Palatino body copy.

Palatino dates back to the Renaissance. Seriously! It's another large font that makes it perfect for the web, traditionally used for headings and print-style ads.

10. Garamond

Garamond Header

Garamond body copy.

Garamond is another old-school font that dates back to styles used in 16th century Paris. This new and improved version was introduced and bundled on most Windows devices (and has been adopted by others since).

11. Bookman

Bookman Header

Bookman body copy.

Bookman (or Bookman Old Style) is another perfect headline option that maintains legibility (or readability) even when used in a small size.

12. Comic Sans MS

Comic Sans MS Header

Comic Sans MS body copy.

Comic Sans MS is a playful, whimsical alternative to other sans serif options.

13. Trebuchet MS

Trebuchet MS Header

Trebuchet MS body copy.

Trebuchet MS is a medieval-themed font originally designed by Microsoft in the mid-nineties. It was used on the XP version, and today commonly appears as body copy on the 'net.

14. Arial Black

Arial Black Header

Arial Black body copy.

Arial Black is the bigger, bolder, badder version of your basic Arial. Funny enough, it also shares proportions with Helvetica. Why is that important? So that they could originally use it to replace Helvetica and print things without paying for the license.

15. Impact

Impact Header

Impact body copy.

Impact is another bold headline choice that looks great in a few short words, and absolutely terrible in a sentence or longer.

What is involved in styling Text in CSS?

As you'll have already experienced in your work with HTML and CSS, text inside an element is laid out inside the element's content box. It starts at the top left of the content area (or the top right, in the case of RTL language content), and flows towards the end of the line. Once it reaches the end, it goes down to the next line and continues, then the next line, until all the content has been placed in the box. Text content effectively behaves like a series of inline elements, being laid out on lines adjacent to one another, and not creating line breaks until the end of the line is reached, or unless you force a line break manually using the
 element.

The CSS properties used to style text which we'll look at separately:

- **Font styles**: Properties that affect the font that is applied to the text, affecting what font is applied, how big it is, whether it is bold, italic, etc.
- **Text layout styles**: Properties that affect the spacing and other layout features of the text, allowing manipulation of, for example, the space between lines and letters, and how the text is aligned within the content box.

```
<h1>Tommy the cat</h1>

<p>I remember as if it were a meal ago...</p>

<p>Said Tommy the Cat as he reeled back to clear whatever foreign matter
 may have nestled its way into his mighty throat. Many a fat alley rat
had met its demise while staring point blank down the cavernous barrel of
 this awesome prowling machine. Truly a wonder of nature this urban
predator – Tommy the cat had many a story to tell. But it was a rare
occasion such as this that he did.</p>
```

Default Fonts

CSS defines five generic names for fonts: serif, sans-serif, monospace, cursive and fantasy. Those are very generic and the exact font face used when using those generic names is up to each browser and can vary for each **operating system** they are running on. It represents a worst case scenario where the browser will try to do its best to provide at least a font that looks appropriate. Serif, sans-serif and monospace are quite predictable and should provide something reasonable. On the other hand, cursive and fantasy are less predictable and we recommend using them very carefully, testing as you go.

The five names are defined as follows:

Term	Definition	Example
serif	Fonts that have serifs (the flourishes and other small details you see at the ends of the strokes in some typefaces)	My big red elephant
sans-serif	Fonts that don't have serifs.	My big red elephant
monospace	Fonts where every character has the same width, typically used in code listings.	My big red elephant
cursive	Fonts that are intended to emulate handwriting, with flowing, connected strokes.	My big red elephant
fantasy	Fonts that are intended to be decorative.	My big red elephant

Keyword

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs.

Font stacks

Since you can't guarantee the availability of the fonts you want to use on your webpages (even a web font *could* fail for some reason), you can supply a **font stack** so that the browser has multiple fonts it can choose from. This simply involves a font-family value consisting of multiple font names separated by commas, e.g.

```
p {
  font-family: "Trebuchet MS", Verdana, sans-serif;
}
```

In such a case, the browser starts at the beginning of the list and looks to see if that font is available on the machine. If it is, it applies that font to the selected elements. If not, it moves on to the next font, and so on.

It is a good idea to provide a suitable generic font name at the end of the stack, so that if none of the listed fonts are available, the browser can at least provide something approximately suitable. To emphasise this point, paragraphs are given the browser's default serif font if no other option is available — which is usually Time New Roman — this is no good for a sans-serif font!

SUMMARY

- CSS Text is a module of CSS that defines how to perform text manipulation, like line breaking, justification and alignment, white space handling, and text transformation.
- The font-family CSS property specifies a prioritized list of one or more font family names and/or generic family names for the selected element.
- The font-family property specifies a list of fonts, from highest priority to lowest. Font selection does not simply stop at the first font in the list that is on the user's system.
- The font-variant property specifies whether or not a text should be displayed in a small-caps font. In a small-caps font, all lowercase letters are converted to uppercase letters.
- The font property is not as straightforward as other shorthand properties, partly due to the syntax requirements for it, and partly due to inheritance issues.
- Two of the values in font shorthand are mandatory: font-size and font-family. If either of these is not included, the entire declaration will be ignored.
- The font-stretch property is new in CSS3 so if it is used in an older browser that doesn't support font-stretch in font shorthand, it will cause the entire line to be ignored.
- The optional values (italic, small-caps, and bold) are placed on the font declaration on the <body> element. These will also apply to most child elements.
- Speaking of font availability, there are only a certain number of fonts that are generally available across all systems, and can therefore be used without much worry. These are the so-called web safe fonts.
- CSS defines five generic names for fonts: serif, sans-serif, monospace, cursive and fantasy. Those are very generic and the exact font face used when using those generic names is up to each browser and can vary for each operating system they are running on.

KNOWLEDGE CHECK

1. **Which of the following is not a appropriate value for font-variant property?**
   a. inherit
   b. default
   c. large-caps
   d. small-caps
2. **Which of the following property adjusts the font-size of the fallback fonts defined with font-family, so that the x-height is the same no matter what font is used?**
   a. default
   b. font-size-fallback
   c. font-adjust
   d. font-size-adjust
3. **Which of the following allows you to expand or condense the widths for a normal, condensed, or expanded font face?**
   a. font-style
   b. font-stretch
   c. font-expand
   d. none of the mentioned
4. **Which of the following font-size-adjust is value used in calculating the size of the fallback fonts?**
   a. auto
   b. number
   c. count
   d. none
5. **"font-style comes first than font-weight in font attribute".State true or false.**
   a. True
   b. False
6. **Which of the following value is supposed to be a slightly bolder weight that standard bold in font attribute?**
   a. empasize
   b. light
   c. lighter
   d. dark

7. **Which of the following property allows contextual adjustment of inter-glyph spacing, i.e. the spaces between the characters in text?**
   a. font-style
   b. font-family
   c. font-kerning
   d. font-variant
8. **Which of the following is not a value for font-style property?**
   a. normal
   b. italic
   c. oblique
   d. none of the above
9. **Which of the following selects a normal, or small-caps face from a font family?**
   a. font-weight
   b. font-synthesis
   c. font-kerning
   d. font-variant
10. **How many type faces inside property "font-family" ?**
   a. 1
   b. Maximum 2
   c. No Limit
   d. Maximum 3

REVIEW QUESTIONS

1. What fonts are available in CSS?
2. How do you change the color of your text?
3. What is font family in CSS?
4. How do you make the text bold?
5. What is involved in styling Text in CSS?

Check Your Result

1. (c)	2. (d)	3. (d)	4. (b)	5. (b)
6. (d)	7. (c)	8. (d)	9. (d)	10. (c)

REFERENCES

1. All CSS specifications. W3.org. 2014-05-22. Archived from the original on 2014-05-30. Retrieved 2014-05-30.
2. Cascading Style Sheets (CSS) Snapshot 2010. 12 May 2011. Archived from the original on 16 March 2011. Retrieved 3 March 2011.
3. CSS Flexible Box Layout Module Level 1. W3C. 19 November 2018. Archived from the original on 19 October 2012. Retrieved 18 October 2012.
4. CSS Snapshot 2017. 31 January 2017. Archived from the original on 13 February 2017. Retrieved 13 February 2017.
5. CSS Snapshot 2018. 15 November 2018. Archived from the original on 1 February 2019. Retrieved 2 January 2019.
6. CSS3 Solutions for Internet Explorer – Smashing Magazine. Smashing Magazine. 2010-04-28. Archived from the original on 2016-10-12. Retrieved 2016-10-12
7. CSS4 Community Group. Archived from the original on 2020-02-27. Retrieved 2020-02-27.
8. Molly Holzschlag (January 2012). "Seven Things Still Missing from CSS". .net Magazine. Archived from the original on 2017-03-05. Retrieved 2017-03-04.
9. Using Feature Queries in CSS Mozilla Hacks – the Web developer blog. hacks.mozilla.org. Archived from the original on 2016-10-11. Retrieved 2016-10-12.

CHAPTER

7

LISTS, TABLE AND FORMS

"The success of every website now depends on search engine optimisation and digital marketing strategy. If you are on first page of all major search engines, then you are ahead among your competitors in terms of online sales."

– Dr. Christopher Dayagdag

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Discuss the lists
2. Describe the table
3. Explain the forms

INTRODUCTION

Lists are very helpful in conveying a set of either numbered or bullet points. CSS list properties you can apply on HTML list element like CSS list-style-type, CSS list-style-

image and CSS list-style-position property. Table layout can be used to represent tabular relationships between data. A table consists of an optional caption and any number of rows of cells. The table model is said to be "row primary" since authors specify rows, not columns, explicitly in the document language. Columns are derived once all the rows have been specified -- the first cell of each row belongs to the first column, the second to the second column, etc.). Rows and columns may be grouped structurally and this grouping reflected in presentation (e.g., a border may be drawn around a group of rows).

CSS form is used to create interactive form for user. It provides many ways to set the style.

7.1 LISTS

The w3-ul class is used to display a basic list:

- Jill
- Eve
- Adam

Example

```
<ul class="w3-ul">
<li>Jill</li>
<li>Eve</li>
<li>Adam</li>
</ul>
```

7.1.1 Bordered List

The w3-border class adds a border around the list:

- Jill
- Eve
- Adam

Example

```
<ul class="w3-ul w3-border">
  <li>Jill</li>
  <li>Eve</li>
```

```
  <li>Adam</li>
</ul>
```

7.1.2 List Header

An example of how to add a heading element inside the list item:

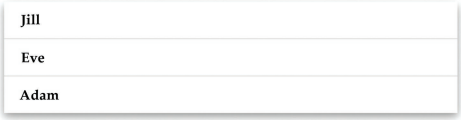
- Names
- Jill
- Eve
- Adam

Example

```
  <ul class="w3-ul w3-border">
<li><h2>Names</h2></li>
<li>Jill</li>
<li>Eve</li>
<li>Adam</li>
  </ul>
```

7.1.3 List as a Card

The w3-card-number classes can be used to show a list as a card:



Example

```
<ul class="w3-ul w3-card-4" style="width:50%">
  <li>Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

7.1.4 Centered List

The w3-center class can be used to center the list items in a list:

Example

```
<ul class="w3-ul w3-center">
  <li>Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

7.1.5 Colored List

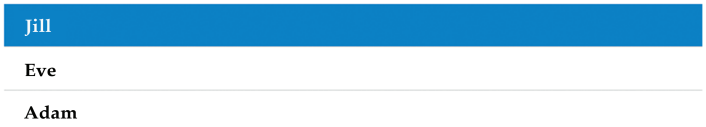
The w3-color classes can be used to add a color to the list:

Example

```
<ul class="w3-ul w3-red">
  <li>Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

7.1.6 Colored List Item

The w3-color classes can be used to add a color to the list item:



Example

```
<ul class="w3-ul">
  <li class="w3-blue">Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

7.1.7 Hoverable List

The w3-hoverable class adds a grey background color to each list item on mouse-over:

Example

```
<ul class="w3-ul w3-hoverable">
  <li>Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

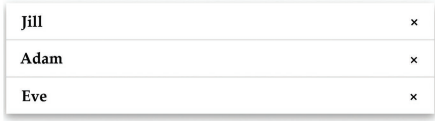
If you want a specific hover color, add any of the w3-hover-color classes to each <li> element:

Example

```
<ul class="w3-ul">
  <li class="w3-hover-red">Jill</li>
  <li class="w3-hover-blue">Eve</li>
  <li class="w3-hover-green">Adam</li>
</ul>
```

7.1.8 Closable List Item

Click on the "x" to close/hide a list item:



Example

```
<li class="w3-display-container">Jill
  <span onclick="this.parentElement.style.display='none'"
  class="w3-button w3-display-right">&times;</span>
</li>
```

Tip: The **HTML** × entity is the preferred icon for close buttons (rather than the letter "X").

Keyword

HTML is the standard markup language for creating web pages and web applications.

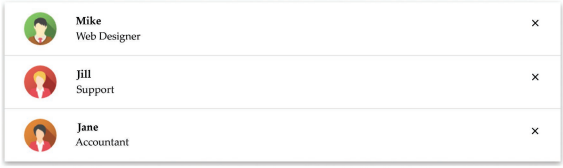
7.1.9 List with Padding

The w3-padding classes can be used to add padding to list items:

Example

```
<ul class="w3-ul">
```

```
  <li class="w3-padding-small">Jill</li>
  <li class="w3-padding-small">Eve</li>
  <li class="w3-padding-small">Adam</li>
</ul>
```



Example

```
<li class="w3-bar">
  <span onclick="this.parentElement.style.display='none'"
  class="w3-bar-item w3-button w3-xlarge w3-right">&times;</span>
  <img src="img_avatar2.png" class="w3-bar-item w3-circle" style="width:85px">
  <div class="w3-bar-item">
    <span class="w3-large">Mike</span><br>
    <span>Web Designer</span>
  </div>
</li>
```

7.1.10 List Width

Lists have a 100% width by default. Use the width property to change this.

Example

```
<ul class="w3-ul" style="width:30%">
  <li>Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

30% width:

Jill
Adam

50% width:

Jill
Adam

80% width:

Jill
Adam

7.1.11 Tiny List

Use the w3-tiny class to display a tiny list:

Jill
Eve
Adam

Example

```
<ul class="w3-ul w3-tiny">
  <li>Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

7.1.12 Small List

:Use the w3-small class to display a small list

Jill
Eve
Adam

Example

```
<ul class="w3-ul w3-small">
  <li>Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

7.1.13 Large List

Use the w3-large class to display a large list:

Jill

Eve

Adam

Example

```
<ul class="w3-ul w3-large">
  <li>Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

7.1.14 XLarge List

Use the w3-xlarge class to display an extra-large list:

Jill

Eve

Adam

Example

```
<ul class="w3-ul w3-xlarge">
  <li>Jill</li>
  <li>Eve</li>
```

```
  <li>Adam</li>
</ul>
```

7.1.15 XXLarge List

Use the w3-xxlarge class to display an XXLarge list:

- Jill
- Eve
- Adam

Example

```
<ul class="w3-ul w3-xxlarge">
  <li>Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

7.1.16 XXXLarge List

Use the w3-xxxlarge class to display an XXXLarge list:

- Jill
- Eve
- Adam

Example

```
<ul class="w3-ul w3-xxxlarge">
  <li>Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

7.1.17 Jumbo List

Use the w3-jumbo class to display an enormous "jumbo" list:

- Jill
- Eve
- Adam

Example

```
<ul class="w3-ul w3-jumbo">
  <li>Jill</li>
  <li>Eve</li>
  <li>Adam</li>
</ul>
```

7.2 TABLES

Teach you how to set different properties of an HTML table using CSS. You can set following properties of a table

- The **border-collapse** specifies whether the browser should control the appearance of the adjacent borders that touch each other or whether each cell should maintain its style.
- The **border-spacing** specifies the width that should appear between table cells.
- The **caption-side** captions are presented in the <caption> element. By default, these are rendered above the table in the document. You use the *caption-side* property to control the placement of the table caption.
- The **empty-cells** specifies whether the border should be shown if a cell is empty.
- The **table-layout** allows browsers to speed up layout of a table by using the first width properties it comes across for the rest of a column rather than having to load the whole table before rendering it.

Did You Know?

In 1980, physicist Tim Berners-Lee, a contractor at CERN, proposed and prototyped ENQUIRE, a system for CERN researchers to use and share documents. In 1989, Berners-Lee wrote a memo proposing an Internet-based hypertext system.

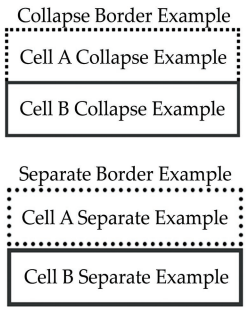
7.2.1 The border-collapse Property

This property can have two values *collapse* and *separate*. The following example uses both the values –

```
<html>
   <head>
      <style type = "text/css">
         table.one {border-collapse:collapse;}
         table.two {border-collapse:separate;}

         td.a {
            border-style:dotted;
            border-width:3px;
            border-color:#000000;
            padding: 10px;
         }
         td.b {
            border-style:solid;
            border-width:3px;
            border-color:#333333;
            padding:10px;
         }
      </style>
   </head>

   <body>
      <table class = "one">
         <caption>Collapse Border Example</caption>
         <tr><td class = "a"> Cell A Collapse Example</td></tr>
         <tr><td class = "b"> Cell B Collapse Example</td></tr>
      </table>
      <br />

      <table class = "two">
```

```
        <caption>Separate Border Example</caption>
        <tr><td class = "a"> Cell A Separate Example</td></tr>
        <tr><td class = "b"> Cell B Separate Example</td></tr>
      </table>
    </body>
</html>
```

It will produce the following result –



7.2.2 The border-spacing Property

The border-spacing property specifies the distance that separates adjacent cells'. borders. It can take either one or two values; these should be units of length.

If you provide one value, it will applies to both vertical and horizontal borders. Or you can specify two values, in which case, the first refers to the horizontal spacing and the second to the vertical spacing –

NOTE – Unfortunately, this property does not work in Netscape 7 or IE 6.

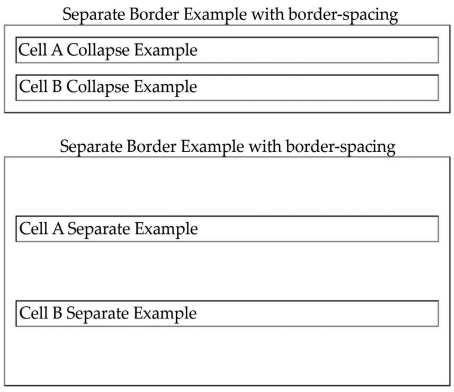
```
<style type="text/css">
  /* If you provide one value */
  table.example {border-spacing:10px;}
  /* This is how you can provide two values */
  table.example {border-spacing:10px; 15px;}
</style>
```

Now let's modify the previous example and see the effect –

```
<html>
  <head>
    <style type = "text/css">
```

```
        table.one {
            border-collapse:separate;
            width:400px;
            border-spacing:10px;
        }
        table.two {
            border-collapse:separate;
            width:400px;
            border-spacing:10px 50px;
        }
    </style>
</head>

<body>

    <table class = "one" border = "1">
        <caption>Separate Border Example with border-spacing</caption>
        <tr><td> Cell A Collapse Example</td></tr>
        <tr><td> Cell B Collapse Example</td></tr>
    </table>
    <br />

    <table class = "two" border = "1">
        <caption>Separate Border Example with border-spacing</caption>
        <tr><td> Cell A Separate Example</td></tr>
        <tr><td> Cell B Separate Example</td></tr>
    </table>

</body>
</html>
```

It will produce the following result –



7.2.3 The caption-side Property

The caption-side property allows you to specify where the content of a <caption> element should be placed in relationship to the table. The table that follows lists the possible values.

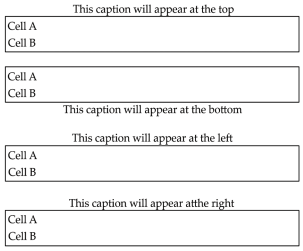
This property can have one of the four values *top*, *bottom*, *left* or *right*. The following example uses each value.

NOTE – These properties may not work with your IE Browser.

```
<html>
   <head>
      <style type = "text/css">
         caption.top {caption-side:top}
         caption.bottom {caption-side:bottom}
         caption.left {caption-side:left}
         caption.right {caption-side:right}
      </style>
   </head>
   <body>

      <table style = "width:400px; border:1px solid black;">
         <caption class = "top">
            This caption will appear at the top
         </caption>
         <tr><td > Cell A</td></tr>
```

```
      <tr><td > Cell B</td></tr>
   </table>
   <br />

   <table style = "width:400px; border:1px solid black;">
      <caption class = "bottom">
         This caption will appear at the bottom
      </caption>
      <tr><td > Cell A</td></tr>
      <tr><td > Cell B</td></tr>
   </table>
   <br />

   <table style = "width:400px; border:1px solid black;">
      <caption class = "left">
         This caption will appear at the left
      </caption>
      <tr><td > Cell A</td></tr>
      <tr><td > Cell B</td></tr>
   </table>
   <br />

   <table style = "width:400px; border:1px solid black;">
      <caption class = "right">
         This caption will appear at the right
      </caption>
      <tr><td > Cell A</td></tr>
      <tr><td > Cell B</td></tr>
   </table>

  </body>
</html>
```

It will produce the following result –



7.2.4 The empty-cells Property

The empty-cells property indicates whether a cell without any content should have a border displayed.

This property can have one of the three values - *show, hide* or *inherit*.

Here is the empty-cells property used to hide borders of empty cells in the <table> element.

```
<html>
  <head>
    <style type = "text/css">
      table.empty {
        width:350px;
        border-collapse:separate;
        empty-cells:hide;
      }
      td.empty {
        padding:5px;
        border-style:solid;
        border-width:1px;
        border-color:#999999;
      }
    </style>
  </head>

  <body>

    <table class = "empty">
```

```
        <tr>
            <th></th>
            <th>Title one</th>
            <th>Title two</th>
        </tr>

        <tr>
            <th>Row Title</th>
            <td class = "empty">value</td>
            <td class = "empty">value</td>
        </tr>

        <tr>
            <th>Row Title</th>
            <td class = "empty">value</td>
            <td class = "empty"></td>
        </tr>
    </table>

  </body>
</html>
```

It will produce the following result –

	Title one	Title two
Row Title	value	value
Row Title	value	

Keyword

Browser is a software application for accessing information on the World Wide Web.

7.2.5 The table-layout Property

The table-layout property is supposed to help you control how a browser should render or lay out a table.

This property can have one of the three values: *fixed, auto* or *inherit*.

The following example shows the difference between these properties.

NOTE – This property is not supported by many **browsers** so do not rely on this property.

```
<html>
   <head>
      <style type = “text/css”>
         table.auto {
            table-layout: auto
         }
         table.fixed {
            table-layout: fixed
         }
      </style>
   </head>

   <body>

      <table class = “auto” border = “1” width = “100%”>
         <tr>
            <td width = “20%”>10000000000000000000000000000</td>
            <td width = “40%”>10000000</td>
            <td width = “40%”>100</td>
         </tr>
      </table>
      <br />

      <table class = “fixed” border = “1” width = “100%”>
         <tr>
            <td width = “20%”>10000000000000000000000000000</td>
            <td width = “40%”>10000000</td>
            <td width = “40%”>100</td>
         </tr>
      </table>
```

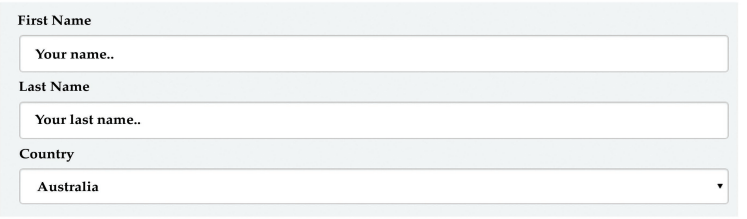
```
    </body>
</html>
```

It will produce the following result –

10000000000000000	10000000	100
100000000000	10000000	100

7.3 FORMS

The look of an HTML form can be greatly improved with CSS:



7.3.1 Styling Input Fields

Use the width property to determine the width of the input field:

Example

```
input {
    width: 100%;
    }
```

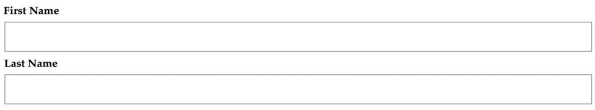
The example above applies to all <input> elements. If you only want to style a specific input type, you can use attribute selectors:

- input[type=text] - will only select text fields
- input[type=password] - will only select password fields

- input[type=number] - will only select number fields
- etc..

7.3.2 Padded Inputs

Use the padding property to add space inside the text field.



Remember

When you have many inputs after each other, you might also want to add some margin, to add more space outside of them:

Example

```
input[type=text] {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    box-sizing: border-box;
}
```

7.3.3 Bordered Inputs

Use the border property to change the border size and color, and use the border-radius property to add rounded corners:

Example

```
input[type=text] {
    border: 2px solid red;
    border-radius: 4px;
}
```

If you only want a bottom border, use the border-bottom property:

First Name

Example

```
input[type=text] {
    border: none;
    border-bottom: 2px solid red;
}
```

7.3.4 Colored Inputs

Use the background-color property to add a background color to the input, and the color property to change the text color:

Example

```
input[type=text] {
    background-color: #3CBC8D;
    color: white;
}
```

7.3.5 Focused Inputs

By default, some browsers will add a blue outline around the input when it gets focus (clicked on). You can remove this behavior by adding outline: none; to the input.

Use the: focus selector to do something with the input field when it gets focus:

Example

input[type=text]:focus {

```
background-color: lightblue;
}
```


Example

```
input[type=text]:focus {
    border: 3px solid #555;
}
```

Example

Given the following style sheet:

```
h1 {
color: pink;
}
```

Suppose there is an h1 element with an emphasizing element (em) inside:

```
<h1>
This is to <em>illustrate</em> inheritance
</h1>
```

Input with icon/image 7.3.6

If you want an icon inside the input, use the background-image property and position it with the background-position property. Also notice that we add a large left padding to reserve the space of the icon:

Example

```
input[type=text] {
    background-color: white;
    background-image: url('searchicon.png');
    background-position: 10px 10px;
    background-repeat: no-repeat;
    padding-left: 40px;
}
```

7.3.7 Animated Search Input

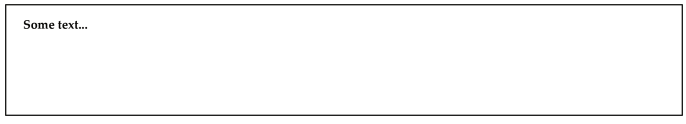
In this example we use the CSS transition property to animate the width of the search input when it gets focus.

Example

```
input[type=text] {
    -webkit-transition: width 0.4s ease-in-out;
    transition: width 0.4s ease-in-out;
}
input[type=text]:focus {
    width: 100%;
}
```

7.3.8 Styling Text areas

Use the resize property to prevent text areas from being resized (disable the "grabber" in the bottom right corner):



Example

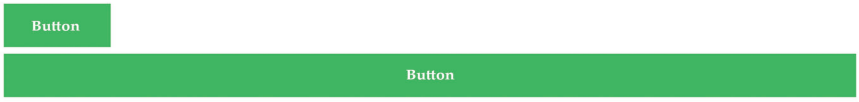
```
textarea {
    width: 100%;
    height: 150px;
    padding: 12px 20px;
    box-sizing: border-box;
    border: 2px solid #ccc;
    border-radius: 4px;
    background-color: #f8f8f8;
    resize: none;
}
```

7.3.9 Styling Select Menus

Example

```
select {
    width: 100%;
    padding: 16px 20px;
    border: none;
    border-radius: 4px;
    background-color: #f1f1f1;
}
```

7.3.10 Styling Input Buttons



Example

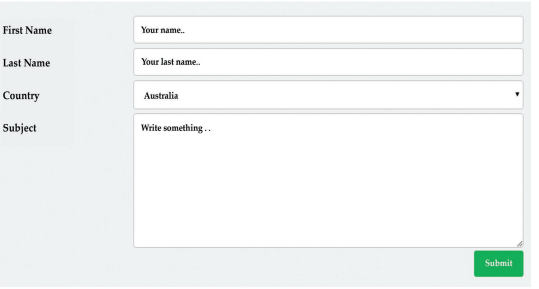
```
input[type=button], input[type=submit], input[type=reset]
{
    background-color: #4CAF50;
    border: none;
    color: white;
    padding: 16px 32px;
    text-decoration: none;
    margin: 4px 2px;
    cursor: pointer;
}
/* Tip: use width: 100% for full-width buttons */
```

Keyword

Windows is a group of several graphical operating system families, all of which are developed, marketed, and sold by Microsoft.

7.3.11 Responsive Form

Resize the browser **window** to see the effect. When the screen is less than 600px wide, make the two columns stack on top of each other instead of next to each other.



CASE STUDY

CSS LEVEL 1

CSS was created to replace all the formatting markup — all the FONT tags and BR tags — with a simple style sheet language. Using CSS, you could control the style of an entire website with a few lines of code in a single file, and that, at the time, was totally radical.

CSS Level 1

width & height
float & clear
margins & padding

backgrounds & borders
fonts
liststyles
alignment & white-space

Now CSS Level 1 didn't have any real layout tools. It had width and height properties, which replaced their namesake HTML attributes; and it had floats, which replaced the align attribute on IMG and TABLE. This was not intended—nor, at that time, used—for coarse layout but for, well, floats. Y'know, with text wrapping around them. Tables were still the accepted way of handling 2D layout; there was nothing else. And of course CSS margins and padding provided a much-needed relief from spacer gifs and BR tags. These things seem very simple to us now, but back then they were revolutionary. :)

SUMMARY

- Lists are very helpful in conveying a set of either numbered or bullet points. CSS list properties you can apply on HTML list element like CSS list-style-type, CSS list-style-image and CSS list-style-position property.
- The border-collapse specifies whether the browser should control the appearance of the adjacent borders that touch each other or whether each cell should maintain its style.
- The caption-side captions are presented in the element. By default, these are rendered above the table in the document. You use the caption-side property to control the placement of the table caption.
- The table-layout allows browsers to speed up layout of a table by using the first width properties it comes across for the rest of a column rather than having to load the whole table before rendering it.
- The border-spacing property specifies the distance that separates adjacent cells'. borders. It can take either one or two values; these should be units of length.
- The caption-side property allows you to specify where the content of a <caption> element should be placed in relationship to the table. The empty-cells property indicates whether a cell without any content should have a border displayed. This property can have one of the three values - show, hide or inherit.
- The table-layout property is supposed to help you control how a browser should render or lay out a table. This property can have one of the three values: fixed, auto or inherit.
- Resize the browser window to see the effect. When the screen is less than 600px wide, make the two columns stack on top of each other instead of next to each other.

KNOWLEDGE CHECK

1. **Which of the following property is used to control the space between the border and content in a table?**
   a. border
   b. margin
   c. padding
   d. resize
2. **Which of the following property is used to specify table borders in CSS?**
   a. table
   b. border
   c. table: border
   d. none of the mentioned
3. **Which of the following property sets whether the table borders are collapsed into a single border or separated:?**
   a. border
   b. border-collapse
   c. collapse
   d. table-border
4. **Which of the following property is used to change the width of table?**
   a. width
   b. table
   c. table-width
   d. resize
5. **Which of the following property sets the vertical alignment?**
   a. align
   b. vertical
   c. vertical-align
   d. vertical-alignment
6. **By which tag, an unordered list is represented?**
   a. <u>
   b. <I>
   c.
   d.

7. **By which tag, an ordered list is represented?**
   a. <u>
   b. <I>
   c.
   d.
8. **In order to start a list from 10, what attribute should be added in the opening tag of ordered list?**
   a. begin= "10"
   b. start= "10"
   c. style= "begin:10"
   d. style= "start:10"
9. **Choose the correct option.**
   a. <th> is used for defining the heading of a table.
   b. By default, contents written between <th> and </th> are bold and centered.
   c. Both A and B
   d. None of the above
10. **By using which of the following options, the border of table can be collapsed?**
   a. border-collapse:collapse
   b. table-border:collapse
   c. border:collapse
   d. table-border-collapse:collapse

REVIEW QUESTIONS

1. How to style an unordered list with CSS?
2. Write a program centered list.
3. What does border-collapse property do?
4. What is the difference between caption-side property in CSS?
5. Explain the styling input fields.
6. Write a program styling select menus.

Check Your Result

1. (c)	2. (b)	3. (b)	4. (a)	5. (b)
6. (c)	7. (d)	8. (b)	9. (c)	10. (a)

REFERENCES

1. C. M. Sperberg-McQueen; et al. Extensible Markup Language (XML) 1.0 (Fifth Edition). 26 November 2008.
2. Christensen, Mathias Biilmann (2015-11-16). "Static Website Generators Reviewed: Jekyll, Middleman, Roots, Hugo". Smashing Magazine. Retrieved 2016-10-26.
3. Elika J. Etemad. CSS Namespaces Module Level 3. 29 September 2011.
4. Gavin Nicol; et al. Document Object Model (DOM) Level 3 Core Specification. 7 April 2004.
5. Håkon Wium Lie; Bert Bos. Cascading Style Sheets (CSS1) Level 1 Specification. 11 April 2008.
6. Oleksy, Walter (2001). Careers in Web Design. New York: The Rosen Publishing Group, Inc. pp. 9–11.
7. Porter, Joshua (24 April 2006), Prioritizing Design Time: A Long Tail Approach, User Interface Engineering, archived from the original on 14 May 2013
8. Spool, Jared (6 August 2007), Usability Tools Podcast: Home Page Design, archived from the original on 29 April 2013

CHAPTER

8

CSS GRID

"The grid system is an aid, not a guarantee. It permits a number of possible uses and each designer can look for a solution appropriate to his personal style. But one must learn how to use the grid; it is an art that requires practice."

– Josef Müller-Brockmann

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Understand the basic concepts of grid layout
2. Describe the relationship of grid layout to other layout methods
3. Explain grid template areas
4. Define layout using named grid lines

INTRODUCTION

A grid can be defined as an intersecting set of vertical and horizontal lines. Grid property offers a grid-based layout system having rows and columns. It makes the designing

of web pages easy without positioning and floating. The grid layout gives us a way to create grid structures depicted in CSS, not in HTML.

Similar to the table, it enables a user to align the elements into rows and columns. But compare to tables, it is easy to design layout with the CSS grid. We can define columns and rows on the grid by using grid-template-rows and grid-template-columns properties.

A grid container can be created by declaring the display: grid or display: inline-grid on an element. Grid container contains the items of a grid that are placed inside the rows and columns.

8.1 BASIC CONCEPTS OF GRID LAYOUT

CSS Grid Layout introduces a two-dimensional grid system to CSS. Grids can be used to lay out major page areas or small user interface elements.

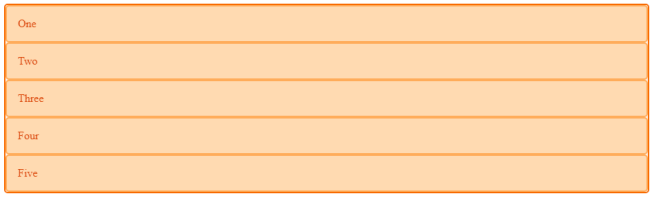
8.1.1 The Grid Container

We create a grid container by declaring display: grid or display: inline-grid on an element. As soon as we do this, all direct children of that element become grid items.

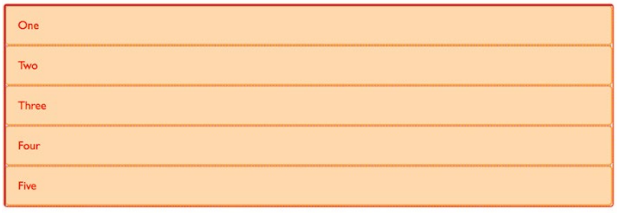
```
<div class=”wrapper”>
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

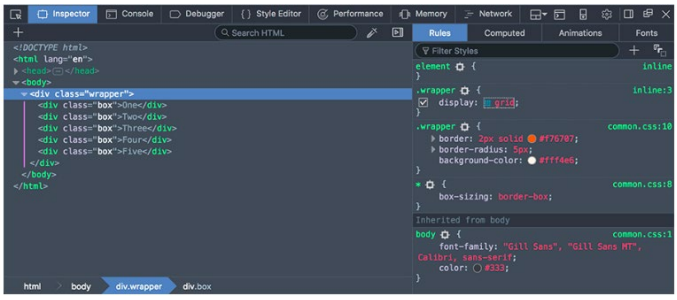
I make the .wrapper a grid container.

```
.wrapper {
  display: grid;
}
```



All the direct children are now grid items. In a web browser, you won't see any difference to how these items are displayed before turning them into a grid, as grid has created a single column grid for the items. At this point, you may find it useful to work with the Grid Inspector, available as part of Firefox's Developer Tools. If you view this example in Firefox and inspect the grid, you will see a small icon next to the value grid. Click this and then the grid on this element will be overlaid in the browser window.





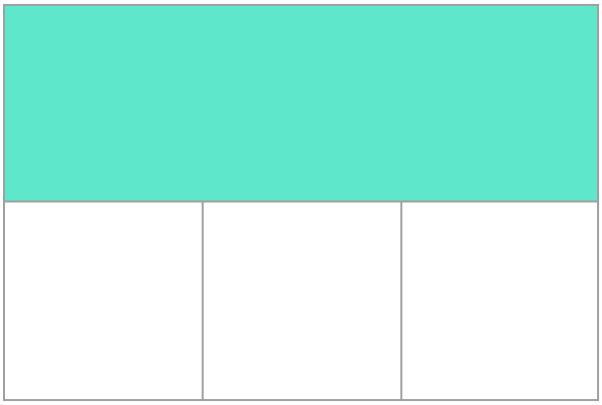
As you learn and then work with the CSS Grid Layout this tool will give you a better idea of what is happening with your grids visually.

Keyword

Grid Inspector allows you to examine CSS Grid Layouts using the Firefox DevTools, discovering grids present on a page, examining and modifying them, debugging layout issues, and more.

8.1.2 Grid Tracks

We define rows and columns on our grid with the grid-template-columns and grid-template-rows properties. These define grid tracks. A grid track is the space between any two lines on the grid. In the below image you can see a track highlighted – this is the first row track in our grid.



The fr unit

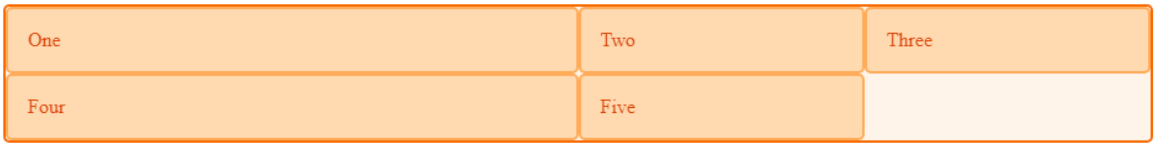
Tracks can be defined using any length unit. Grid also introduces an additional length unit to help us create flexible grid tracks. The new fr unit represents a fraction of the available space in the grid container. The next grid definition would create three equal width tracks that grow and shrink according to the available space.

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>

.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

Unequal Sizes

In this next example, we create a definition with a 2fr track then two 1fr tracks. The available space is split into four. Two parts are given to the first track and one part each to the next two tracks.

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
.wrapper {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
}
```

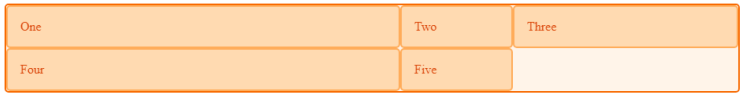


Mixing flexible and Absolute Sizes

In this final example, we mix absolute sized tracks with fr units. The first track is 500 pixels, so the fixed width is taken away from the available space. The remaining space is divided into three and assigned in proportion to the two flexible tracks.

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

```
.wrapper {
  display: grid;
  grid-template-columns: 500px 1fr 2fr;
}
```



Track listings with repeat() notation

Large grids with many tracks can use the repeat() notation, to repeat all or a section of the track listing. For example the grid definition:

```
.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

Can also be written as:

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

Repeat notation can be used for a part of the track listing. In this next example I have created a grid with an initial 20-pixel track, then a repeating section of 6 1fr tracks then a final 20-pixel track.

```
.wrapper {
  display: grid;
  grid-template-columns: 20px repeat(6, 1fr) 20px;
}
```

Repeat notation takes the track listing, and uses it to create a repeating pattern of tracks. In this next example, my grid will consist of 10 tracks, a 1fr track, and then followed by a 2fr track. This pattern will be repeated five times.

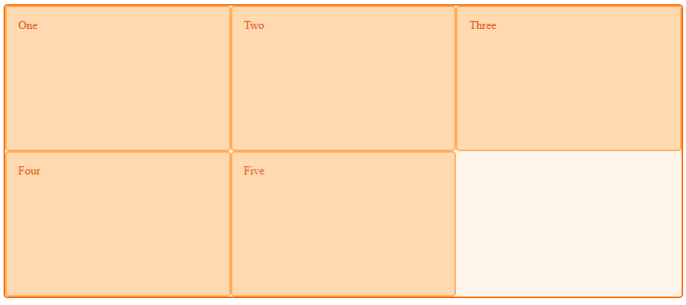
```
.wrapper {
  display: grid;
  grid-template-columns: repeat(5, 1fr 2fr);
}
```

The Implicit and Explicit Grid

When creating our example grid we specifically defined our column tracks with the grid-template-columns property, but the grid also created rows on its own. These rows are part of the implicit grid. Whereas the explicit grid consists of any rows and columns defined with grid-template-columns or grid-template-rows. If you place something outside of the defined grid—or due to the amount of content, more grid tracks are needed—then the grid creates rows and columns in the implicit grid. These tracks will be auto-sized by default, resulting in their size being based on the content that is inside them.

You can also define a set size for tracks created in the implicit grid with the grid-auto-rows and grid-auto-columns properties. In the below example we use grid-auto-rows to ensure that tracks created in the implicit grid are 200 pixels tall.

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 200px;
}
```



Did You Know?

All modern browsers support CSS grid except for Opera Mini and brave Browser. Web developers targeting older browsers can utilize Modernizr 3.5.0 to detect and gracefully degrade the webpage as needed.

Track Sizing and Minmax

When setting up an explicit grid or defining the sizing for automatically created rows or columns we may want to give tracks a minimum size, but also ensure they expand to fit any content that is added. For example, I may want my rows to never collapse smaller than 100 pixels, but if my content stretches to 300 **pixels** in height, then I would like the row to stretch to that height.

Keyword

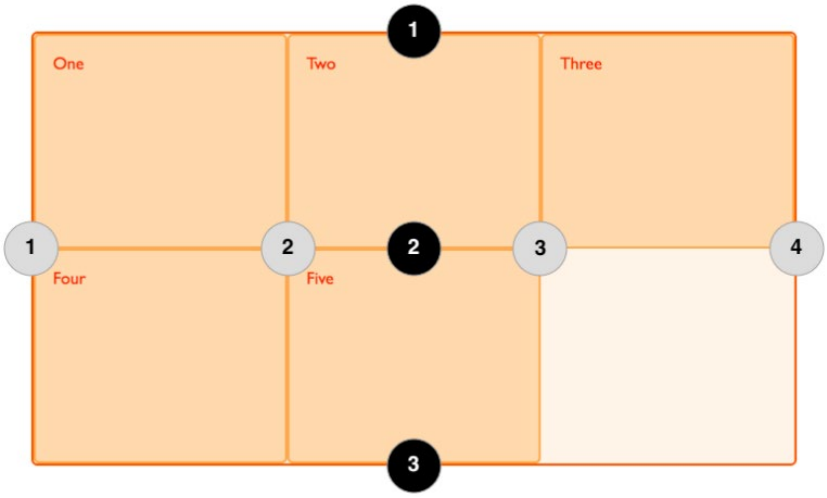
Pixel, pel, or picture element is a smallest addressable element in a raster image, or the smallest addressable element in an all points addressable display device; so it is the smallest controllable element of a picture represented on the screen.

Grid has a solution for this with the minmax() function. In this next example I am using minmax() in the value of grid-auto-rows. This means automatically created rows will be a minimum of 100 pixels tall, and a maximum of auto. Using auto means that the size will look at the content size and will stretch to give space for the tallest item in a cell, in this row.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: minmax(100px, auto);
}
<div class="wrapper">
  <div>One</div>
  <div>Two
    <p>I have some more content in.</p>
    <p>This makes me taller than 100 pixels.</p>
  </div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

8.1.3 Grid Lines

It should be noted that when we define a grid we define the grid tracks, not the lines. Grid then gives us numbered lines to use when positioning items. In our three column, two row grid we have four column lines.



Lines are numbered according to the writing mode of the document. In a left-to-right language, line 1 is on the left-hand side of the grid. In a right-to-left language, it is on the right-hand side of the grid. Lines can also be named, and we will look at how to do this in a later guide in this series.

Positioning Items Against Lines

The following example demonstrates doing this in a simple way. When placing an item, we target the line – rather than the track.

In the following example I am placing the first two items on our three column track grid, using the grid-column-start, grid-column-end, grid-row-start and grid-row-end properties. Working from left to right, the first item is placed against column line 1, and spans to column line 4, which in our case is the far-right line on the grid. It begins at row line 1 and ends at row line 3, therefore spanning two row tracks.

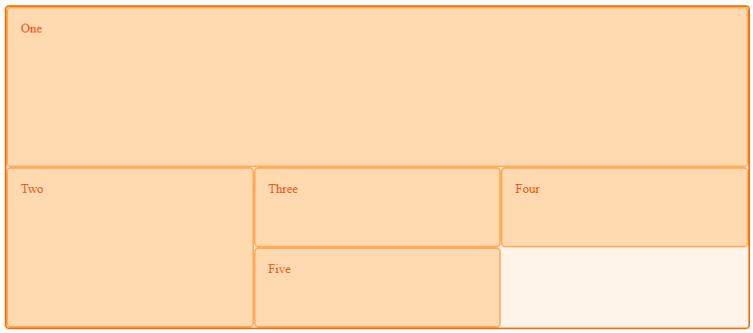
The second item starts on grid column line 1, and spans one track. This is the default so we do not need to specify the end line. It also spans two row tracks from row line 3 to row line 5. The other items will place themselves into empty spaces on the grid.

```
<div class=”wrapper”>
  <div class=”box1”>One</div>
  <div class=”box2”>Two</div>
  <div class=”box3”>Three</div>
  <div class=”box4”>Four</div>
  <div class=”box5”>Five</div>
```

```
</div>
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}
.box2 {
  grid-column-start: 1;
  grid-row-start: 3;
  grid-row-end: 5;
}
```

Remember

Don't forget that you can use the Grid Inspector in Firefox Developer Tools to see how the items are positioned against the lines of the grid.



Line-Positioning Shorthands

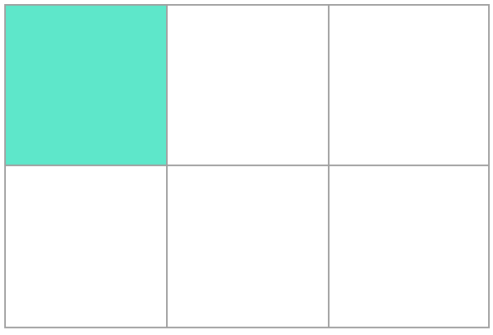
The longhand values used above can be compressed onto one line for columns with grid-column, and one line for rows with grid-row. The following example would give the same positioning as in the previous code, but with far less CSS. The value before the forward slash character (/) is the start line, the value after the end line.

You can omit the end value if the area only spans one track.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}
.box1 {
  grid-column: 1 / 4;
  grid-row: 1 / 3;
}
.box2 {
  grid-column: 1 / 3;
  grid-row: 5;
}
```

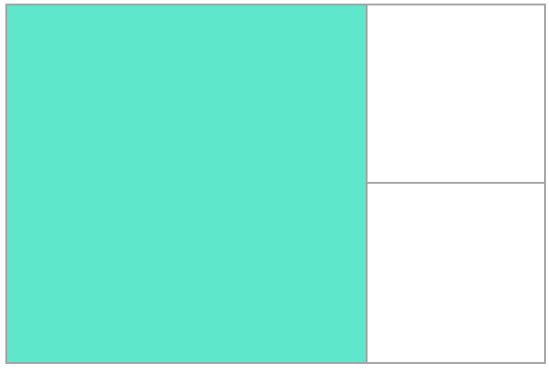
8.1.4 Grid Cells

A grid cell is the smallest unit on a grid. Conceptually it is like a table cell. As we saw in our earlier examples, once a grid is defined as a parent the child items will lay themselves out in one cell each of the defined grid. In the below image:



8.1.5 Grid Areas

Items can span one or more cells both by row or by column, and this creates a grid area. Grid areas must be rectangular – it isn't possible to create an L-shaped area for example. The highlighted grid area spans two row and two column tracks.



8.1.6 Gutters

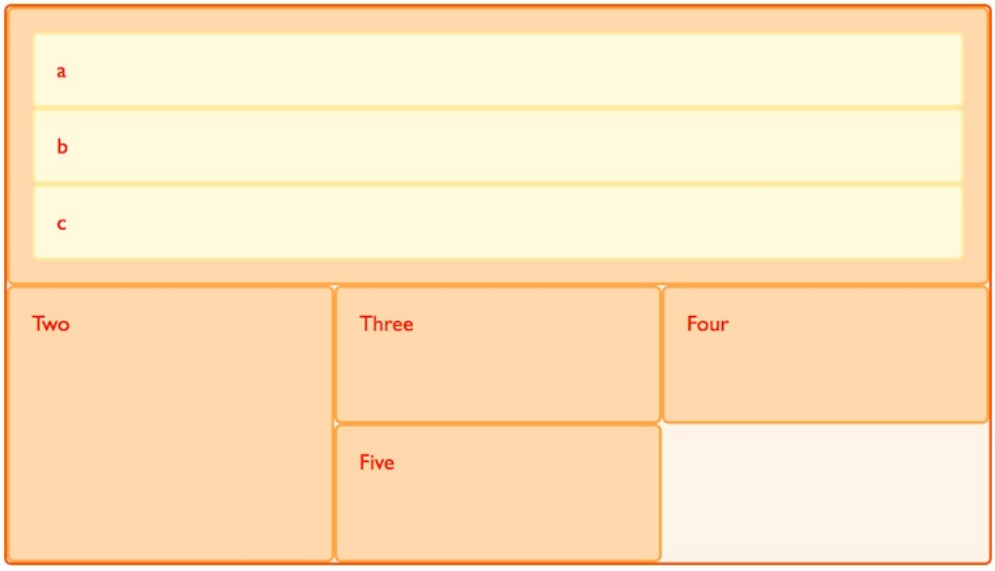
Gutters or alleys between grid cells can be created using the column-gap and row-gap properties, or the shorthand gap. In the below example, I am creating a 10-pixel gap between columns and a 1em gap between rows.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  column-gap: 10px;
  row-gap: 1em;
}

<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

Any space used by gaps will be accounted for before space is assigned to the flexible length fr tracks, and gaps act for sizing purposes like a regular grid track, however you cannot place anything into a gap. In terms of line-based positioning, the gap acts like a thick line.

8.1.7 Nesting Grids

A grid item can become a grid container. In the following example, I have the three-column grid that I created earlier, with our two positioned items. In this case the first item has some sub-items. As these items are not direct children of the grid they do not participate in grid layout and so display in a normal document flow.



Nesting without Subgrid

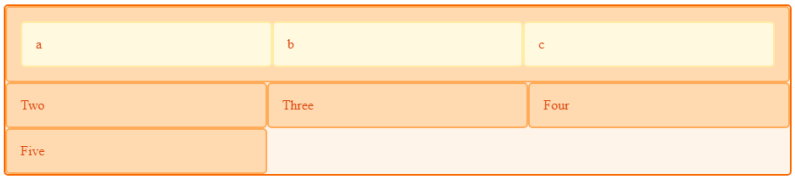
If I set box1 to display: grid I can give it a track definition and it too will become a grid. The items then lay out on this new grid.

```
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  display: grid;
```

```
  grid-template-columns: repeat(3, 1fr);
}
```

Copy to Clipboard

```
* {box-sizing: border-box;}
.wrapper {
  border: 2px solid #f76707;
  border-radius: 5px;
  background-color: #fff4e6;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
.box {
  border: 2px solid #ffa94d;
  border-radius: 5px;
  background-color: #ffd8a8;
  padding: 1em;
  color: #d9480f;
}
.box1 {
  grid-column: 1 / 4;
}
.nested {
  border: 2px solid #ffec99;
  border-radius: 5px;
  background-color: #fff9db;
  padding: 1em;
}
```



In this case the nested grid has no relationship to the parent. As you can see in the example it has not inherited the gap of the parent and the lines in the nested grid do not align to the lines in the parent grid.

Subgrid

In the working draft of the Level 2 Grid specification there is a feature called subgrid, which would let us create nested grids that use the track definition of the parent grid.

In the current specification, we would edit the above nested grid example to change the track definition of grid-template-columns: repeat(3, 1fr), to grid-template-columns: subgrid. The nested grid will then use the parent grid tracks to layout items.

```
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  display: grid;
  grid-template-columns: subgrid;
}
```

8.1.8 Layering Items with z-Index

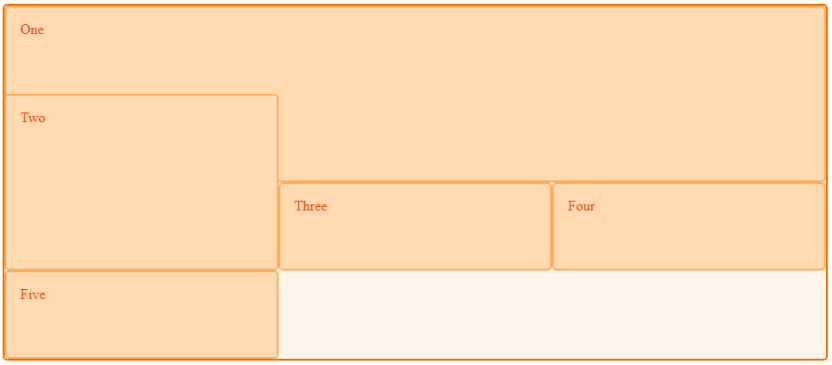
Grid items can occupy the same cell, and in this case we can use the z-index property to control the order in which overlapping items stack.

Overlapping without z-index

If we return to our example with items positioned by line number, we can change this to make two items overlap.

```
<div class="wrapper">
  <div class="box box1">One</div>
  <div class="box box2">Two</div>
  <div class="box box3">Three</div>
  <div class="box box4">Four</div>
  <div class="box box5">Five</div>
</div>
.wrapper {
```

```
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}
.box2 {
  grid-column-start: 1;
  grid-row-start: 2;
  grid-row-end: 4;
}
```



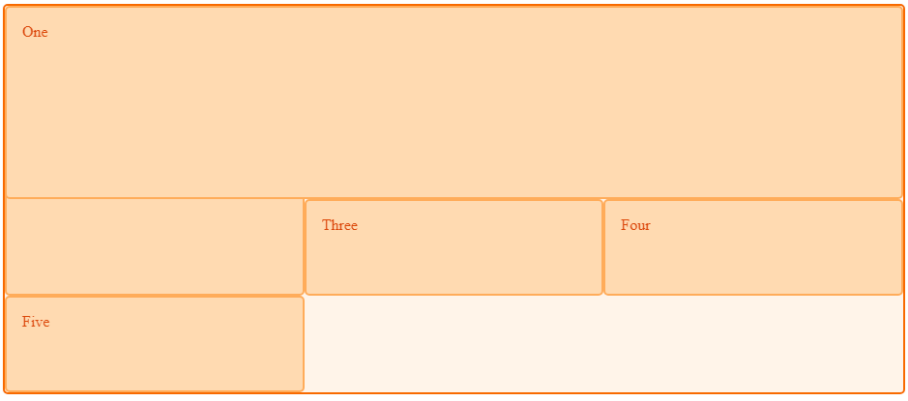
The item box2 is now overlapping box1, it displays on top as it comes later in the source order.

Controlling the order

We can control the order in which items stack up by using the z-index property - just like positioned items. If we give box2 a lower z-index than box1 it will display below box1 in the stack.

```
.wrapper {
  display: grid;
```

```
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  z-index: 2;
}
.box2 {
  grid-column-start: 1;
  grid-row-start: 2;
  grid-row-end: 4;
  z-index: 1;
}
```



8.2 RELATIONSHIP OF GRID LAYOUT TO OTHER LAYOUT METHODS

CSS Grid Layout has been designed to work alongside other parts of CSS, as part of a complete system for doing the layout.

8.2.1 Grid and Flexbox

The basic difference between CSS Grid Layout and CSS Flexbox Layout is that **flexbox** was designed for layout in one dimension - either a row or a column. Grid was designed for two-dimensional layout - rows, and columns at the same time. The two specifications share some common features, however, and if you have already learned how to use flexbox, the similarities should help you get to grips with Grid.

Keyword

Flexbox is a one-dimensional layout method for arranging items in rows or columns.

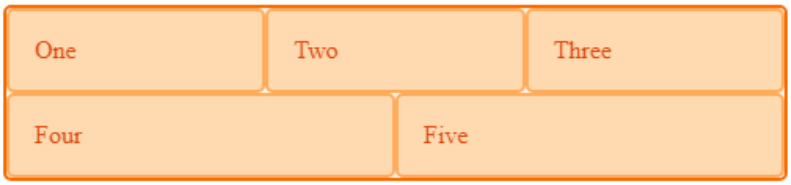
One-dimensional versus two-dimensional layout

A simple example can demonstrate the difference between one- and two-dimensional layouts.

In this first example, I am using flexbox to lay out a set of boxes. We have five child items in my container, and we have given the flex properties values so that they can grow and shrink from a flex-basis of 150 pixels.

We have also set the flex-wrap property to wrap, so that if the space in the container becomes too narrow to maintain the flex basis, items will wrap onto a new row.

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
.wrapper {
  width: 500px;
  display: flex;
  flex-wrap: wrap;
}
.wrapper > div {
  flex: 1 1 150px;
}
```



In the image, you can see that two items have wrapped onto a new line. These items are sharing the available space and not lining up underneath the items above. This is because when you wrap flex items, each new row (or column when working by column) is an independent flex line in the flex container. Space distribution happens across the flex line.

A common question then is how to make those items line up. This is where you want a two-dimensional layout method: You want to control the alignment by row and column, and this is where grid comes in.

The Same Layout with CSS Grids

In this next example, we create the same layout using Grid. This time we have three 1fr column tracks. We do not need to set anything on the items themselves; they will lay themselves out one into each cell of the created grid. As you can see they stay in a strict grid, lining up in rows and columns. With five items, we get a gap on the end of row two.

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

A simple question to ask yourself when deciding between grid or flexbox is:

- do I only need to control the layout by row *or* column – use a flexbox
- do I need to control the layout by row *and* column – use a grid

Content out or layout in?

In addition to the one-dimensional versus two-dimensional distinction, there is another way to decide if you should use flexbox or grid for a layout. Flexbox works from the content out. An ideal use case for flexbox is when you have a set of items and want to space them out evenly in a container. You let the size of the content decide how much individual space each item takes up. If the items wrap onto a new line, they will work out their spacing based on their size and the available space *on that line*.

Grid works from the layout in. When you use CSS Grid Layout you create a layout and then you place items into it, or you allow the auto-placement rules to place the items into the grid cells according to that strict grid. It is possible to create tracks that respond to the size of the content, however, they will also change the entire track.

If you are using flexbox and find yourself disabling some of the flexibility, you probably need to use CSS Grid Layout. An example would be if you are setting a percentage width on a flex item to make it line up with other items in a row above. In that case, a grid is likely to be a better choice.

Box Alignment

The feature of flexbox that was most exciting to many of us was that it gave us proper alignment control for the first time. It made it easy to center a box on the page. Flex items can stretch to the height of the flex container, meaning that equal height columns were possible. These were things we have wanted to do for a very long time, and have come up with all kinds of hacks to accomplish, at least visually.

The alignment properties from the flexbox specification have been added to a new specification called Box Alignment Level 3. This means that they can be used in other specifications, including Grid Layout. In the future, they may well apply to other layout methods as well.

In a later guide in this series, I'll be taking a proper look at Box Alignment and how it works in Grid Layout. For now, here is a comparison between simple examples of flexbox and grid. In the first example, which uses flexbox, we have a container with three items inside. The wrapper min-height is set, so it defines the height of the flex container. We have set align-items on the flex container to flex-end so the items will line up at the end of the flex container. We have also set the align-self property on box1 so it will override the default and stretch to the height of the container and on box2 so it aligns to the start of the flex container.

```
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
</div>
.wrapper {
  display: flex;
  align-items: flex-end;
  min-height: 200px;
}
.box1 {
  align-self: stretch;
}
.box2 {
  align-self: flex-start;
}
```

Alignment in CSS Grids

This second example uses a grid to create the same layout. This time we are using the box alignment properties as they apply to a grid layout. So we align to start and end rather than flex-start and flex-end. In the case of a grid layout, we are aligning the items inside their grid area. In this case that is a single grid cell, but it could be an area made up of several grid cells.

```
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
```

```
</div>
.wrapper {
  display: grid;
  grid-template-columns: repeat(3,1fr);
  align-items: end;
  grid-auto-rows: 200px;
}
.box1 {
  align-self: stretch;
}
.box2 {
  align-self: start;
}
```

The fr unit and flex-basis

We have already seen how the fr unit works to assign a proportion of available space in the grid container to our grid tracks. The fr unit, when combined with the minmax() function can give us very similar behavior to the flex properties in flexbox while still enabling the creation of a layout in two dimensions.

If we look back at the example where I demonstrated the difference between one and two-dimensional layouts, you can see there is a difference between the way of the two layouts work responsively. With the flex layout, if we drag our window wider and smaller, the flexbox does a nice job of adjusting the number of items in each row according to the available space. If we have a lot of space all five items can fit on one row. If we have a very narrow container we may only have space for one.

In comparison, the grid version always has three column tracks. The tracks themselves will grow and shrink, but there are always three since we asked for three when defining our grid.

Auto-filling grid tracks

We can create a similar effect to flexbox, while still keeping the content arranged in strict rows and columns, by creating our track listing using repeat notation and the auto-fill and auto-fit properties.

In this next example, we have used the auto-fill keyword in place of an integer in the repeat notation and set the track listing to 200 pixels. This means that grid will create as many 200 pixels column tracks as will fit in the container.

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
</div>
.wrapper {
  display: grid;
  grid-template-columns: repeat(auto-fill, 200px);
}
```

A Flexible Number of Tracks

This isn't quite the same as flexbox. In the flexbox example, the items are larger than the 200 pixel basis before wrapping. We can achieve the same in grid by combining auto-fit and the minmax() function. In this next example, I create auto filled tracks with minmax. I want my tracks to be a minimum of 200 pixels, so I set the maximum to be 1fr. Once the browser has worked out how many times 200 pixels will fit into the container–also taking account of grid gaps–it will treat the 1fr maximum as an instruction to share out the remaining space between the items.

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
</div>
.wrapper {
  display: grid;
```

```
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
}
```

8.2.2 Grid and Absolutely Positioned Elements

Grid interacts with absolutely positioned elements, which can be useful if you want to position an item inside a grid or grid area. The specification defines the behavior when a grid container is a containing block and a parent of the absolutely positioned item.

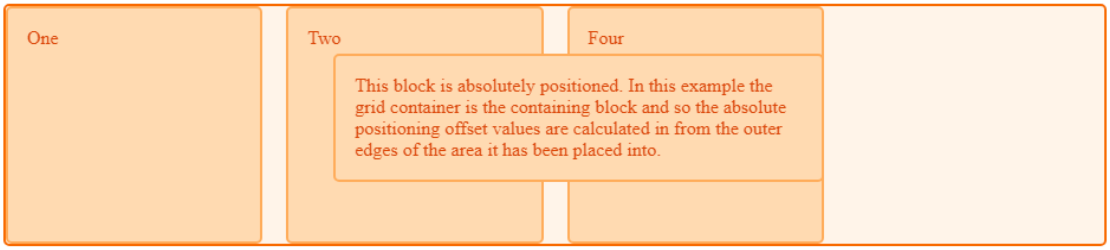
A Grid Container As Containing Block

To make the grid container a containing block you need to add the position property to the container with a value of relative, just as you would make a containing block for any other absolutely positioned items. Once you have done this, if you give a grid item position: absolute it will take as its containing block the grid container or, if the item also has a grid position, the area of the grid it is placed into.

In the below example we have a wrapper containing four child items. Item three is absolutely positioned and also placed on the grid using line-based placement. The grid container has position: relative and so becomes the positioning context of this item.

```
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">
    This block is absolutely positioned. In this example the grid container is the containing block and so the absolute positioning offset values are calculated in from the outer edges of the area it has been placed into.
  </div>
  <div class="box4">Four</div>
</div>
.wrapper {
  display: grid;
  grid-template-columns: repeat(4,1fr);
  grid-auto-rows: 200px;
  gap: 20px;
```

```
  position: relative;
}
.box3 {
  grid-column-start: 2;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  position: absolute;
  top: 40px;
  left: 40px;
}
```

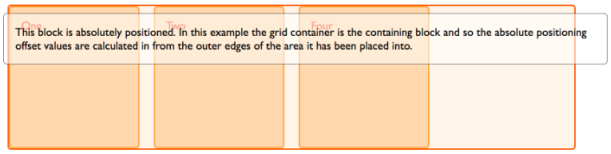


You can see that the item is taking the area from grid column line 2 to 4, and starting after line 1. Then it is offset in that area using the top and left properties. However, it has been taken out of flow as is usual for absolutely positioned items and so the auto-placement rules now place items into the same space. The item also doesn't cause the additional row to be created to span to row line 3.

If we remove position: absolute from the rules for .box3 you can see how it would display without the positioning.

A Grid Container as Parent

If the absolutely positioned child has a grid container as a parent but that container does not create a new positioning context, then it is taken out of flow as in the previous example. The positioning context will be whatever element creates a positioning context as is common to other layout methods. In our case, if we remove position: relative from the wrapper above, positioning context is from the viewport, as shown in this image.



Once again the item no longer participates in the **grid layout** in terms of sizing or when other items are auto-placed.

Keyword

Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

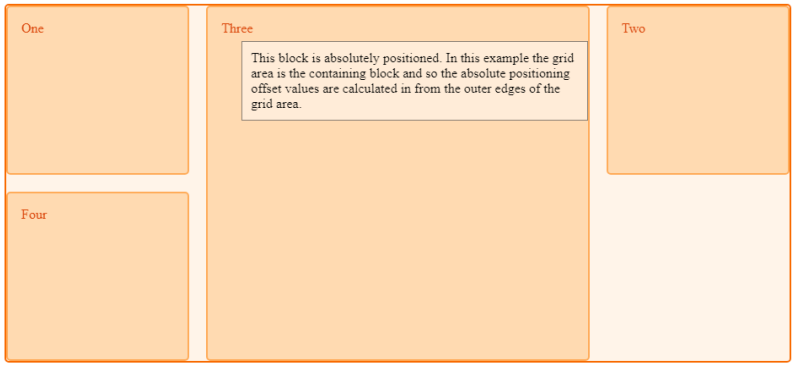
With a grid area as the parent

If the absolutely positioned item is nested inside a grid area then you can create a positioning context on that area. In the below example we have our grid as before but this time we have nested an item inside .box3 of the grid.

We have given .box3 position relative and then positioned the sub-item with the offset properties. In this case, the positioning context is the grid a

```
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three
    <div class="abspos">
      This block is absolutely positioned. In this example the grid area is the containing block and so the absolute positioning offset values are calculated in from the outer edges of the grid area.
    </div>
  </div>
  <div class="box4">Four</div>
</div>
.wrapper {
  display: grid;
  grid-template-columns: repeat(4,1fr);
  grid-auto-rows: 200px;
  gap: 20px;
```

```
}
.box3 {
  grid-column-start: 2;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  position: relative;
}
.abspos {
  position: absolute;
  top: 40px;
  left: 40px;
  background-color: rgba(255,255,255,.5);
  border: 1px solid rgba(0,0,0,0.5);
  color: #000;
  padding: 10px;
}
```



8.3 GRID TEMPLATE AREAS

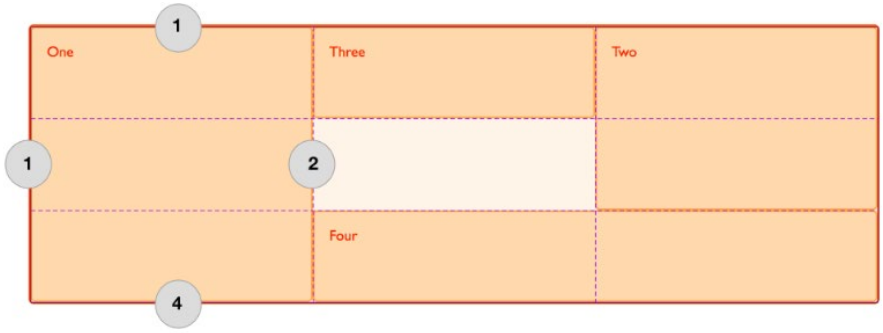
When you use CSS Grid Layout you always have lines, and this can be a straightforward way to place items on your grid. However, there is an alternate method to use for positioning items on the grid which you can use alone or in combination with line-based placement. This method involves placing our items using named template areas, and we will find out exactly how this method works.

8.3.1 Naming a Grid Area

You have already encountered the grid-area property. This is the property that can take as a value all four of the lines used to position a grid area.

```
.box1 {
   grid-area: 1 / 1 / 4 / 2;
}
```

What we are doing here when defining all four lines, is defining the area by specifying the lines that enclose that area.



We can also define an area by giving it a name and then specify the location of that area in the value of the grid-template-areas property. You can choose what you would like to name your area. For example, if I wish to create the layout shown below I can identify four main areas.

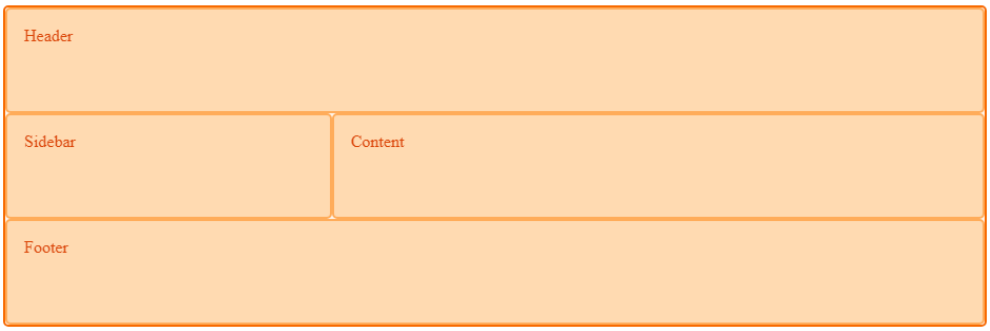
- a header
- a footer
- a sidebar
- the main content

With the grid-area property we can assign each of these areas a name. This will not yet create any layout, but we now have named areas to use in a layout.

```
.header {
    grid-area: hd;
}
.footer {
    grid-area: ft;
}
.content {
    grid-area: main;
}
.sidebar {
    grid-area: sd;
}
```

This time, instead of placing my items using line numbers specified on the items themselves, we create the whole layout on the grid container.

```
.wrapper {
    display: grid;
    grid-template-columns: repeat(9, 1fr);
    grid-auto-rows: minmax(100px, auto);
    grid-template-areas:
      "hd hd hd hd  hd  hd  hd  hd  hd"
      "sd sd sd main main main main main main"
"ft ft ft ft  ft  ft  ft  ft  ft";
}
<div class="wrapper">
    <div class="header">Header</div>
    <div class="sidebar">Sidebar</div>
    <div class="content">Content</div>
    <div class="footer">Footer</div>
</div>
```



Using this method we do not need to specify anything at all on the individual grid items, everything happens on our grid container. We can see the layout described as the value of the grid-template-areas property.

8.3.2 Leaving a Grid Cell Empty

We have completely filled our grid with areas in this example, leaving no white space. However you can leave grid cells empty with this method of layout. To leave a cell empty use the full stop character, '.'. If I want to only display the footer directly under the main content we would need to leave the three cells underneath the sidebar empty.

```
.header {
    grid-area: hd;
}
.footer {
    grid-area: ft;
}
.content {
    grid-area: main;
}
.sidebar {
    grid-area: sd;
}
.wrapper {
    display: grid;
    grid-template-columns: repeat(9, 1fr);
    grid-auto-rows: minmax(100px, auto);
    grid-template-areas:
```

```
        "hd hd hd hd   hd   hd   hd   hd   hd"
        "sd sd sd main main main main main main"
        ".  .  .  ft   ft   ft   ft   ft   ft";
}
<div class="wrapper">
    <div class="header">Header</div>
    <div class="sidebar">Sidebar</div>
    <div class="content">Content</div>
    <div class="footer">Footer</div>
</div>
```

In order to make the layout neater I can use multiple . characters. As long as there is no white space between the full stops it will be counted as one cell. For a complex layout there is a benefit to having the rows and columns neatly aligned. It means that you can actually see, right there in the CSS, what this layout looks like.

8.3.3 Spanning Multiple Cells

In our example each of the areas spans multiple grid cells and we achieve this by repeating the name of that grid area multiple times with white space between. You can add extra white space in order to keep your columns neatly lined up in the value of grid-template-areas. You can see that we have done this in order that the hd and ft line up with main.

The area that you create by chaining the area names must be rectangular, at this point there is no way to create an L-shaped area. The specification does note that a future level might provide this functionality. You can however span rows just as easily as columns. For example we could make our sidebar span down to the end of the footer by replacing the . with sd.

```
.header {
    grid-area: hd;
}
.footer {
    grid-area: ft;
}
.content {
    grid-area: main;
}
.sidebar {
    grid-area: sd;
}
.wrapper {
    display: grid;
    grid-template-columns: repeat(9, 1fr);
    grid-auto-rows: minmax(100px, auto);
    grid-template-areas:
      "hd hd hd hd   hd   hd   hd   hd   hd"
      "sd sd sd main main main main main main"
      "sd sd sd  ft  ft   ft   ft   ft   ft";
}
```

The value of grid-template-areas must show a complete grid, otherwise it is invalid (and the property is ignored). This means that you must have the same number of cells for each row, if empty with a full stop character demonstrating that the cell is to be left empty. You will also create an invalid grid if your areas are not rectangular.

8.3.4 Redefining the Grid Using Media Queries

As our layout is now contained in one part of the CSS, this makes it very easy to make changes at different breakpoints. You can do this by redefining the grid, the position of items on the grid, or both at once.

When doing this, define the names for your areas outside of any **media queries**. That way the content area would always be called main no matter where on the grid it is placed.

For our layout above, we might like to have a very simple layout at narrow widths, defining a single column grid and stacking up our items.

```
.header {
    grid-area: hd;
}
.footer {
    grid-area: ft;
}
.content {
    grid-area: main;
}
.sidebar {
    grid-area: sd;
}
.wrapper {
    display: grid;
    grid-auto-rows: minmax(100px, auto);
    grid-template-columns: 1fr;
    grid-template-areas:
      "hd"
      "main"
      "sd"
      "ft";
}
```

Keyword

Media queries is a feature of CSS 3 allowing content rendering to adapt to different conditions such as screen resolution.

We can then redefine that layout inside media queries to go to our two columns layout, and perhaps take it to a three column layout if the available space is even wider.

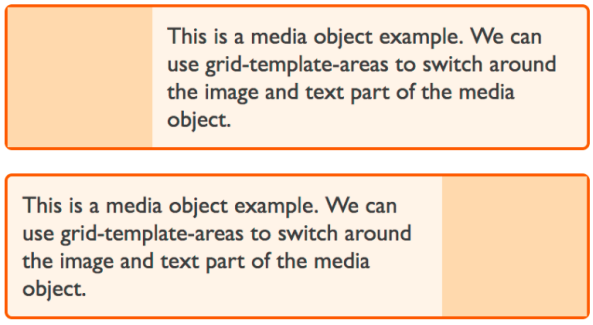
```
@media (min-width: 500px) {
    .wrapper {
        grid-template-columns: repeat(9, 1fr);
        grid-template-areas:
          "hd hd hd hd  hd  hd  hd  hd  hd"
          "sd sd sd main main main main main main"
          "sd sd sd  ft  ft  ft  ft  ft  ft";
    }
}
@media (min-width: 700px) {
    .wrapper {
        grid-template-areas:
          "hd hd hd  hd  hd  hd  hd  hd hd"
          "sd sd main main main main main ft ft";
    }
}
```

8.3.5 Using Grid-Template-Areas for UI Elements

Many of the grid examples you will find online make the assumption that you will use grid for main page layout, however grid can be just as useful for small elements as those larger ones. Using grid-template-areas can be especially nice as it is easy to see in the code what your element looks like.

Media object Example

As a very simple example we can create a "media object". This is a component with space for an image or other media on one side and content on the other. The image might be displayed on the right or left of the box.



Our grid is a two-column track grid, with the column for the image sized at 1fr and the text 3fr. If you wanted a fixed width image area, then you could set the image column as a pixel width, and assign the text area 1fr. A single column track of 1fr would then take up the rest of the space.

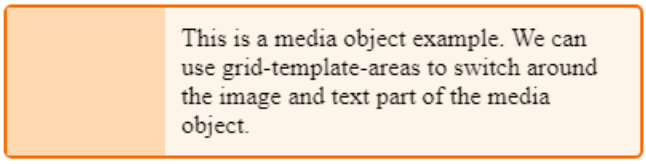
We give the image area a grid area name of img and the text area content, then we can lay those out using the grid-template-areas property.

```
* {box-sizing: border-box;}
.media {
    border: 2px solid #f76707;
    border-radius: 5px;
    background-color: #fff4e6;
    max-width: 400px;
}
.media {
    display: grid;
    grid-template-columns: 1fr 3fr;
    grid-template-areas: "img content";
    margin-bottom: 1em;
}
.media .image {
    grid-area: img;
```

```
    background-color: #ffd8a8;
}
.media .text {
    grid-area: content;
    padding: 10px;
}
<div class="media">
    <div class="image"></div>
    <div class="text">This is a media object example.
```

We can use grid-template-areas to switch around the image and text part of the media object.

```
    </div>
</div>
```



Displaying the Image on the Other Side of the Box

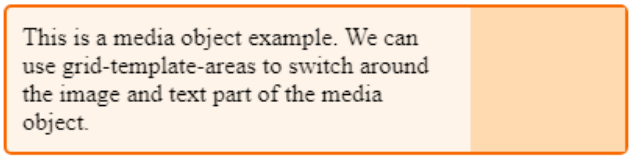
We might want to be able to display our box with the image the other way around. To do this, we redefine the grid to put the 1fr track last, and flip the values in grid-template-areas.

```
* {box-sizing: border-box;}
.media {
    border: 2px solid #f76707;
    border-radius: 5px;
    background-color: #fff4e6;
    max-width: 400px;
}
.media {
    display: grid;
    grid-template-columns: 1fr 3fr;
```

```
    grid-template-areas: “img content”;
    margin-bottom: 1em;
}
.media.flipped {
    grid-template-columns: 3fr 1fr;
    grid-template-areas: “content img”;
}
.media .image {
    grid-area: img;
    background-color: #ffd8a8;
}
.media .text {
    grid-area: content;
    padding: 10px;
}
<div class=”media flipped”>
    <div class=”image”></div>
    <div class=”text”>This is a media object example.
```

We can use grid-template-areas to switch around the image and text part of the media object.

```
    </div>
</div>
```



8.4 LAYOUT USING NAMED GRID LINES

Line naming is incredibly useful, but some of the more baffling looking grid syntax comes from this combination of names and track sizes. Once you work through some examples it should become clearer and easier to work with.

8.4.1 Naming Lines when defining a Grid

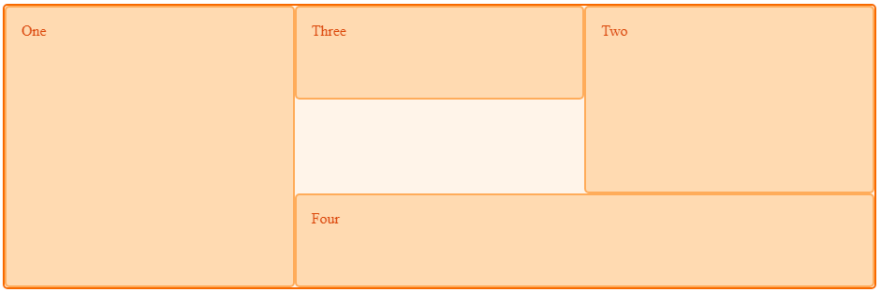
You can assign some or all of the lines in your grid a name when you define your grid with the grid-template-rows and grid-template-columns properties. Those names can be anything you like. We have defined a name for the start and end of the container, both for rows and columns. Then defined the center block of the grid as content-start and content-end again, both for columns and rows although you do not need to name all of the lines on your grid. You might choose to name just some key lines for your layout.

```
.wrapper {
  display: grid;
 grid-template-columns: [main-start] 1fr [content-start] 1fr [content-end] 1fr [main-end];
  grid-template-rows: [main-start] 100px [content-start] 100px [content-end] 100px [main-end];
}
```

Once the lines have names, we can use the name to place the item rather than the line number.

```
.box1 {
  grid-column-start: main-start;
  grid-row-start: main-start;
  grid-row-end: main-end;
}
.box2 {
  grid-column-start: content-end;
  grid-row-start: main-start;
  grid-row-end: content-end;
}
.box3 {
  grid-column-start: content-start;
  grid-row-start: main-start;
}
.box4 {
  grid-column-start: content-start;
  grid-column-end: main-end;
```

```
  grid-row-start: content-end;
}
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
  <div class="box4">Four</div>
</div>
```



Everything else about line-based placement still works in the same way and you can mix named lines and line numbers. Naming lines is useful when creating a responsive design where you redefine the grid, rather than then needing to redefine the content position by changing the line number in your media queries, you can ensure that the line is always named the same in your definitions.

Giving lines multiple names

You may want to give a line more than one name, perhaps it denotes the sidebar-end and the main-start for example. To do this add the names inside the square brackets with whitespace between them [sidebar-end main-start]. You can then refer to that line by either of the names.

8.4.2 Implicit Grid Areas from Named Lines

The name is a custom ident, an author-defined name. When choosing the name you need to avoid words that might appear in the specification and be confusing - such as span. Idents are not quoted. While you can choose any name, if you append -start and -end to the lines around an area, as we have in the example above, grid will create you a named area of the main name used. Taking the above example, we have content-start and content-end both for rows and for columns. This means I get a grid area named content, and could place something in that area should I wish to.

```
.wrapper {
  display: grid;
 grid-template-columns: [main-start] 1fr [content-start] 1fr [content-end] 1fr [main-end];
  grid-template-rows: [main-start] 100px [content-start] 100px [content-end] 100px [main-end];
}
.thing {
  grid-area: content;
}

<div class="wrapper">
  <div class="thing">I am placed in an area named content.</div>
</div>
```

8.4.3 Implicit Grid Lines from Named Areas

We have seen how named lines create a named area, and this also works in reverse. Named template areas create named lines that you can use to place your items. If we take the layout created in the guide to Grid Template Areas, we can use the lines created by our areas to see how this works.

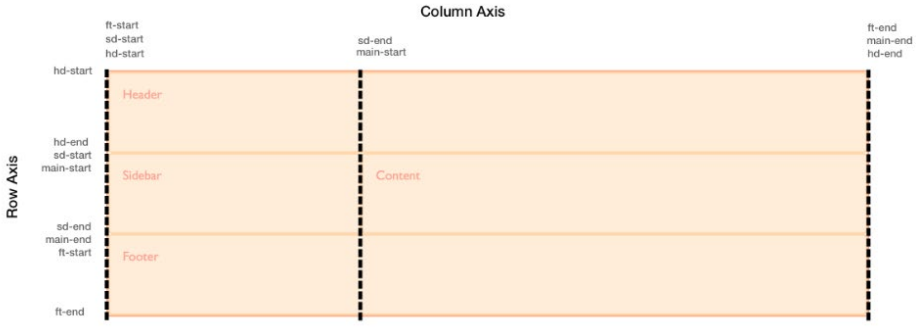
In this example we have added an extra div with a class of overlay. We have named areas created using the grid-area property, then a layout created in grid-template-areas. The area names are:

- hd
- ft
- main
- sd

This gives us column and row lines:

- hd-start
- hd-end
- sd-start
- sd-end
- main-start
- main-end
- ft-start
- ft-end

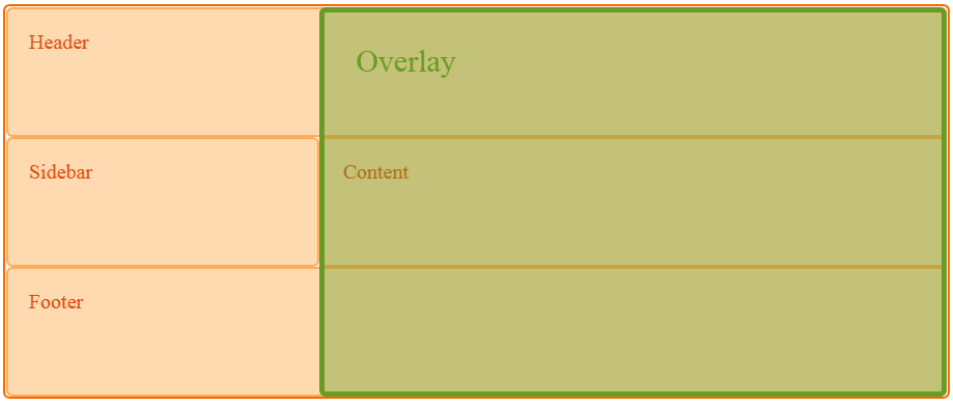
You can see the named lines in the image, note that some lines have two names - for example sd-end and main-start refer to the same column line.



To position overlay using these implicit named lines is the same as positioning an item using lines that we have named.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(9, 1fr);
  grid-auto-rows: minmax(100px, auto);
  grid-template-areas:
    "hd hd hd hd  hd  hd  hd  hd  hd"
    "sd sd sd main main main main main main"
```

```
    "ft ft ft ft   ft   ft   ft   ft   ft";
}
.header {
  grid-area: hd;
}
.footer {
  grid-area: ft;
}
.content {
  grid-area: main;
}
.sidebar {
  grid-area: sd;
}
.wrapper > div.overlay {
  z-index: 10;
  grid-column: main-start / main-end;
  grid-row: hd-start / ft-end;
  border: 4px solid rgb(92,148,13);
  background-color: rgba(92,148,13,.4);
  color: rgb(92,148,13);
  font-size: 150%;
}
<div class="wrapper">
  <div class="header">Header</div>
  <div class="sidebar">Sidebar</div>
  <div class="content">Content</div>
  <div class="footer">Footer</div>
  <div class="overlay">Overlay</div>
</div>
```



Given that we have this ability to position create lines from named areas and areas from named lines it is worth taking a little bit of time to plan your naming strategy when starting out creating your grid layout. By selecting names that will make sense to you and your team you will help everyone to use the layouts you create more easily.

8.4.4 Multiple Lines With The Same Name with repeat()

If you want to give all of the lines in your grid a unique name then you will need to write out the track definition long-hand rather than using the repeat syntax, as you need to add the name in square brackets while defining the tracks. If you do use the repeat syntax you will end up with multiple lines that have the same name, however this can be very useful too.

Twelve-Column Grid Using Repeat()

Before defining the 1fr size of the column track I am also defining a line name of [col-start]. This means that we will end up with a grid that has 12 column lines all named col-start before a 1fr width column.

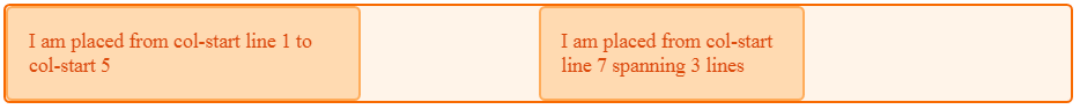
```
.wrapper {
  display: grid;
  grid-template-columns: repeat(12, [col-start] 1fr);
}
```

Once you have created the grid you can place items onto it. As we have multiple lines named col-start if you place an item to start after line col-start grid uses the first line named col-start, in our case that will be the far left line. To address another line use the name, plus the number for that line. To place our item from the first line named col-start to the 5th, we can use:

```
.item1 {
  grid-column: col-start / col-start 5
}
```

You can also use the span keyword here. My next item will be placed from the 7th line named col-start and span 3 lines.

```
.item2 {
  grid-column: col-start 7 / span 3;
}
<div class="wrapper">
  <div class="item1">I am placed from col-start line 1 to col-start 5</div>
  <div class="item2">I am placed from col-start line 7 spanning 3 lines</div>
</div>
```



If you take a look at this layout in the Firefox Grid Highlighter you can see how the column lines are shown, and how our items are placed against these lines.

Defining named lines with a track list

The repeat syntax can also take a track list, it doesn't just need to be a single track size that is being repeated. The code below would create an eight track grid, with a narrower 1fr width column named col1-start followed by a wider 3fr column named col2-start.

```
.wrapper {
  grid-template-columns: repeat(4, [col1-start] 1fr [col2-start] 3fr);
}
```

If your repeating syntax puts two lines next to each other then they will be merged, and create the same result as giving a line multiple names in a non-repeating track

definition. The following definition, creates four 1fr tracks, which each have a start and end line.

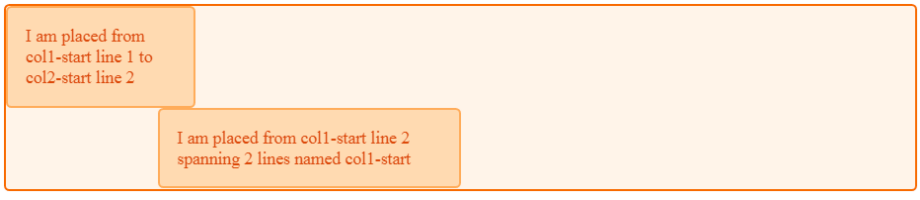
```
.wrapper {
  grid-template-columns: repeat(4, [col-start] 1fr [col-end] );
}
```

If we write this definition out without using repeat notation it would look like this.

```
.wrapper {
  grid-template-columns: [col-start] 1fr [col-end col-start] 1fr [col-end col-start] 1fr [col-end col-start] 1fr [col-end];
}
```

If you have used a track list then you can use the span keyword not just to span a number of lines but also to span a number of lines of a certain name.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(6, [col1-start] 1fr [col2-start] 3fr);
}
.item1 {
  grid-column: col1-start / col2-start 2
}
.item2 {
  grid-row: 2;
  grid-column: col1-start 2 / span 2 col1-start;
}
<div class="wrapper">
  <div class="item1">I am placed from col1-start line 1 to col2-start line 2</div>
  <div class="item2">I am placed from col1-start line 2 spanning 2 lines named col1-start</div>
</div>
```



Twelve-Column Grid Framework

Over the last three guides you have discovered that there are a lot of different ways to place items using grid. This can seem a little bit overcomplicated at first, but remember you don't need to use all of them. In practice we find that for straightforward layouts, using named template areas works well, it gives that nice visual representation of what your layout looks like, and it is then easy to move things around on the grid.

If working with a strict multiple column layout for example the named lines demonstration in the last part of this guide works very well. If you consider grid systems such as those found in frameworks like Foundation or **Bootstrap**, these are based on a 12 column grid. The framework then imports the code to do all of the calculations to make sure that the columns add up to 100%. With grid layout the only code we need for our grid "framework" is:

Keyword

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development.

```
.wrapper {
  display: grid;
  gap: 10px;
  grid-template-columns: repeat(12, [col-start] 1fr);
}
```

We can then use that framework to layout our page. For example, to create a three column layout with a header and footer, we might have the following markup.

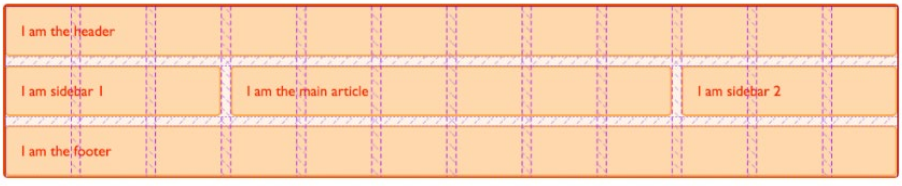
```
<div class="wrapper">
  <header class="main-header">I am the header</header>
  <aside class="side1">I am sidebar 1</aside>
  <article class="content">I am the main article</article>
  <aside class="side2">I am sidebar 2</aside>
  <footer class="main-footer">I am the footer</footer>
</div>
```

We could then place this on my grid layout framework like this.

```
.main-header,
.main-footer  {
```

```
  grid-column: col-start / span 12;
}
.side1 {
  grid-column: col-start / span 3;
  grid-row: 2;
}
.content {
  grid-column: col-start 4 / span 6;
  grid-row: 2;
}
.side2 {
  grid-column: col-start 10 / span 3;
  grid-row: 2;
}
```

Once again, the grid highlighter is helpful to show us how the grid we have placed our items on works.



We don't need to do any calculations, grid automatically removed my 10 pixel gutter track before assigning the space to the 1fr column tracks. As you start to build out your own layouts, you will find that the syntax becomes more familiar and you choose the ways that work best for you and the type of projects you like to build. Try building some common patterns with these various methods, and you will soon find your most productive way to work. Then, in the next guide we will look at how grid can position items for us - without us needing to use placement properties at all!

SUMMARY

- CSS Grid layout separates a page into major sections. Grid property offers a grid-based layout system having rows and columns. It makes the designing of web pages easy without positioning and floating. The grid layout gives us a way to create grid structures depicted in CSS, not in HTML.
- CSS Grid Layout introduces a two-dimensional grid system to CSS. Grids can be used to lay out major page areas or small user interface elements.
- Lines are numbered according to the writing mode of the document. In a left-to-right language, line 1 is on the left-hand side of the grid. In a right-to-left language, it is on the right-hand side of the grid.
- The longhand values used above can be compressed onto one line for columns with grid-column, and one line for rows with grid-row.
- A grid cell is the smallest unit on a grid. Conceptually it is like a table cell. As we saw in our earlier examples, once a grid is defined as a parent the child items will lay themselves out in one cell each of the defined grid.
- Gutters or alleys between grid cells can be created using the column-gap and row-gap properties, or the shorthand gap.
- Grid interacts with absolutely positioned elements, which can be useful if you want to position an item inside a grid or grid area. The specification defines the behavior when a grid container is a containing block and a parent of the absolutely positioned item.

KNOWLEDGE CHECK

1. **The CSS property used to specify the order of flex item in the grid container is -**
   a. order property
   b. float property
   c. overflow property
   d. None of the above
2. **Which of the following property defines the style of the divider rule between columns in a multicolumn text flow?**
   a. columns
   b. column-flow
   c. column-style
   d. none of the mentioned
3. **Which of the following property defines the color of any rules between columns in a multicolumn text flow?**
   a. column-color
   b. column-rule-color
   c. column-rule-style
   d. column-rule-decoration
4. **Which of the following property is a shorthand definition of the number of columns and their widths in a multicolumn text flow?**
   a. column-width
   b. column-style
   c. columns
   d. filter
5. **How many columns and rows are defined in the following code?**

```
section {
display: grid;
grid-template-columns: 100px 1fr 1fr 250px;
grid-rows: 50px 1fr 1fr ;
}

section header {
grid-column: 1 / 4;
```

```
grid-row: 1;
}

section nav {
grid-column: 1;
grid-row: 2 / 3;
}

section article {
grid-column: 2;
grid-row: 2;
}

section aside {
grid-column: 3;
grid-row: 2;
}

section footer {
grid-column: 1 / 4;
grid-row: 3;
}
```

a. 1 rows and 2 columns
b. 2 rows and 3 columns
c. 3 rows and 2 columns
d. 3 rows and 4 columns

6. The CSS property used to specify the transparency of an element is -

a. Hover
b. Opacity
c. clearfix
d. overlay

REVIEW QUESTIONS

1. What is grid container?
2. What is grid lines?
3. Differentiate between CSS Grid Layout and CSS Flexbox Layout.
4. What is use grid-template-areas for UI elements?
5. What is media queries?

Check Your Result

1. (a) 2. (d) 3. (b) 4. (c) 5. (a) 6. (b)

REFERENCES

1. Anderson, Kareem (13 September 2017). "Microsoft's newest browser gets a significant boost with EdgeHTML 16". Retrieved 7 October 2017.
2. Atkins Jr., Tab; J. Etemad, Elika; Atanassov, Rossen; Brufau, Oriol; Mogilevsky, Alex; Cupp, Phil; Mielke, Markus, eds. (2021-03-11). "CSS Grid Layout Module Level 2". W3C. CSS Working Group. Retrieved 2021-04-09.
3. CSS Grid – Table layout is back. Be there and be square. Google Developers. January 2017. Retrieved 2021-04-09.
4. Goetter, Raphael (2017-02-16). "Flexbox grids and frameworks". GitHub Gist. Archived from the original on 2017-02-16. Retrieved 2021-04-09.
5. Marcotte, Ethan (2018-07-18). "Fractional. — Ethan Marcotte". Ethan Marcotte. Retrieved 2021-04-09.
6. Protalinski, Emil (9 March 2017). "Chrome 57 arrives with CSS Grid Layout and API improvements | VentureBeat". VentureBeat. Retrieved 7 October 2017.
7. Rendle, Robin (2017-06-12). "An Introduction to the `fr` CSS unit". CSS-Tricks. Retrieved 2021-04-09.

CHAPTER

9

WORKING WITH IMAGES AND MULTIMEDIA

"Multimedia is not more media, but the employment of various kinds of media (and hybrid media) for what they each offer to advance the narrative."

– Fred Ritchin

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Understand web colors
2. Identify how to choose graphics software
3. Prepare photographic images
4. Discus the create animated web graphics
5. Define integrate multimedia into your website

INTRODUCTION

Images are an important part of any web application. Including a lot of images in a web application is generally not recommended, but it is important to use the images

wherever they required. CSS helps us to control the display of images in web applications.

CSS plays a good role to control image display. You can set the following image properties using CSS.

- The border property is used to set the width of an image border.
- The height property is used to set the height of an image.
- The width property is used to set the width of an image.
- The -moz-opacity property is used to set the opacity of an image.

Multimedia comes in many different formats. It can be almost anything you can hear or see, like images, music, sound, videos, records, films, animations, and more.

9.1 UNDERSTANDING WEB COLORS

Specifying a background color other than white for a web page is easier than you probably realize. For example, to specify blue as the background color for a page, put style="background-color:blue" inside the tag or in the style sheet rule for the body element. Of course, you can use many colors other than blue. In fact, there are 16 colors listed in the W3C standards: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. Obviously there are many more colors displayed on the Web than just those 16. In fact, there are 140 color names that you can use with assurance that all browsers will display these colors similarly. Here's a partial list of the 140 descriptive color names: azure, bisque, cornflowerblue, darksalmon, firebrick, honeydew, lemonchiffon, papayawhip, peachpuff, saddlebrown, thistle, tomato, wheat, and whitesmoke. But names are subjective—for instance, if you look at the color chart of 140 cross-browser color names, you will not be able to distinguish between fuchsia and magenta. You will then realize that the associated hexadecimal color values for those two terms, fuchsia and magenta, are exactly the same: #FF00FF.

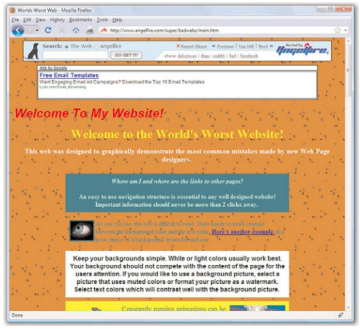


Figure 1. A partial screenshot of "The World's Worst Website."

There are, in fact, 16 million colors made possible with hexadecimal color codes. However, most modern computer displays can display "only" 16,384. But 16,384 is still a lot more than 140, or 16.

You should be aware that not all computer monitors display colors in the same hues. What might appear as a beautiful light blue background color on your monitor might be more of a purple hue on another user's monitor. Neutral, earth-tone colors (such as medium gray, tan, and ivory) can produce even more unpredictable results on many computer monitors. These colors might even seem to change color on one monitor depending on lighting conditions in the room or the time of day

In addition to changing the background of your pages to a color other than white, you can change the color of text links, including various properties of links. You can also set the background color of container elements (such as paragraphs, divs, blockquotes, and table cells), and you can use colors to specify the borders around those elements.

9.1.1 Using Hexadecimal Values for Colors

To remain standards-compliant, as well as to retain precise control over the colors in your website, you can reference colors by their hexadecimal value. The hexadecimal value of a color is an indication of how much red, green, and blue light should be mixed into each color. It works a little bit like Play-Doh—just mix in the amounts of red, blue, and green you want to get the appropriate color.

The hexadecimal color format is #rrggbb, in which rr, gg, and bb are twodigit hexadecimal values for the red (rr), green (gg), and blue (bb) components of the color. If you're not familiar with hexadecimal numbers, don't sweat it. Just remember that FF is the maximum and 00 is the minimum. Use one of the following codes for each component:

- FF means full brightness.
- CC means 80% brightness.
- 99 means 60% brightness.
- 66 means 40% brightness.
- 33 means 20% brightness.
- 00 means none of this color component.

For example, bright red is #FF0000, dark green is #003300, bluish-purple is #660099, and medium-gray is #999999. To make a page with a red background and dark green text, the HTML code would look like the following:

<body style="background-color:#FF0000; color:#003300">

Although only six examples of two-digit hexadecimal values are shown here, there are actually 225 combinations of two-digit hexadecimal values—0 through 9 and A through F, paired up. For example, F0 is a possible hex value (decimal value 240),

62 is a possible hex value (decimal value 98), and so on. The rr, gg, and bb in the #rrggbb hexadecimal color code format stand for the red, green, and blue components of the color. Each of those components has a decimal value ranging from 0 (no color) to 255 (full color).

So, white (or #FFFFFF) translates to a red value of 255, a green value of 255, and a blue value of 255. Similarly, black (#000000) translates to a red value of 0, a green value of 0, and a blue value of 0. True red is #FF0000 (all red, no green, and no blue), true green is #00FF00 (no red, all green, no blue), and true blue is #0000FF (no red, no green, and all blue). All other hexadecimal notations translate to some variation of the 255 possible values for each of the three colors. The cross-browser compatible color name cornflowerblue is associated with the hexadecimal notation #6495ED—a red value of 40, a green value of 149, and a blue value of 237 (almost all of the available blue values).

When picking colors, either through a graphics program or by finding something online that you like, you might see the color notion in hexadecimal or decimal. If you type hexadecimal color converter in your search engine, you will find numerous options to help you convert color values into something you can use in your style sheets.

9.1.2 Using CSS to Set Background, Text, and Border Colors

When using CSS, there are three instances in which color values can be used: when specifying the background color, the text color, or the border color of elements.

Figure 2 shows an example of color usage that could easily go into a web design Hall of Shame. I can't imagine ever using these combinations of colors and styles in a serious website, but it serves here as an example of how color style could be applied to various elements.

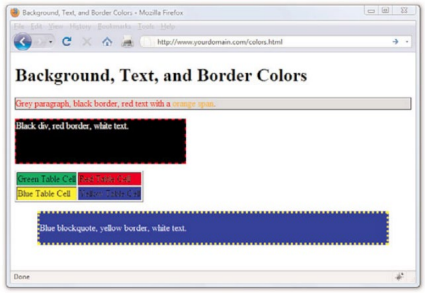


Figure 2. Background, text, and border colors can all be set using CSS.

Listing 1 shows the XHTML and CSS styles used to produce Figure 2.

Listing 1. Using Styles to Produce Background, Text, and Border Colors

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>Background, Text, and Border Colors</title>
</head>
<body>
<h1>Background, Text, and Border Colors</h1>
<p style="background-color:#CCCCCC;
border:1px solid #000000; color:#FF0000">
Grey paragraph, black border, red text with a
<span style="color:#FFA500">orange span</span>.</p>
<div style="width:300px; height:75px; margin-bottom: 12px;
background-color:#000000; border:2px dashed #FF0000;
color: #FFFFFF">
Black div, red border, white text. </div>
<table border="1">
<tr>
<td style="background-color: #00FF00">Green Table Cell</td>
<td style="background-color: #FF0000">Red Table Cell</td>
</tr>
<tr>
<td style="background-color: #FFFF00">Blue Table Cell</td>
<td style="background-color: #0000FF">Yellow Table Cell</td>
</tr>
</table>
<blockquote style="background-color:#0000FF;
border:4px dotted #FFFF00; color:#FFFFFF"><p>Blue blockquote,
yellow border, white text.</p></blockquote>
</body>
</html>
```

Looking at the styles used in Listing 1, you should be able to figure out almost everything except some of the border styles. In CSS, borders can't be designated as a color without also having a width and type; in the first example shown in Listing 1, the border width is 1px, and the border type is solid. In the second example shown in Listing 1, the border width is 2px, and the border type is dashed. In the fourth example shown in Listing 1, the border width is 4px, and the border type is dotted.

When picking colors for your website, remember that a little bit goes a long way—if you really like a bright and spectacular color, use it as an accent color and not throughout the primary design elements. For readability, remember that light backgrounds with dark text are much easier to read than dark backgrounds with light text.

9.2 CHOOSING GRAPHICS SOFTWARE

You can use almost any graphics program to create and edit images for your website, from the simple paint program that typically comes free with your computer's operating system to an expensive professional program such as Adobe Photoshop. Similarly, if you have a digital camera or scanner attached to your computer, it probably came with some graphics software capable of creating web page graphics. There are also several free image editors available for download—or even online as a web application—that deal just with the manipulation of photographic elements.

9.2.1 The Least You Need to Know About Graphics

Two forces are always at odds when you post graphics and multimedia on the Internet. The users' eyes and ears want all your content to be as detailed and accurate as possible, and they also want that information displayed immediately. Intricate, colorful graphics mean big file sizes, which increase the transfer time even over a fast connection. How do you maximize the quality of your presentation while minimizing file size? To make these choices, you need to understand how color and resolution work together to create a subjective sense of quality.

The resolution of an image is the number of individual dots, or pixels, that make up an image. Large, high-resolution images generally take longer to transfer and display than small, low-resolution images. Resolution is usually specified as the width times the height of the image, expressed in pixels; a 300×200 image, for example, is 300 pixels wide and 200 pixels high.

You might be surprised to find that resolution isn't the most significant factor determining an image file's storage size (and transfer time). This is because images used on web pages are always stored and transferred in compressed form. Image compression is the mathematical manipulation that images are put through to squeeze

out repetitive patterns. The mathematics of image compression is complex, but the basic idea is that repeating patterns or large areas of the same color can be squeezed out when the image is stored on a disk. This makes the image file much smaller and allows it to be transferred faster over the Internet. The web browser then restores the original appearance of the image when the image is displayed.

The techniques you'll use to accomplish this depend on the contents and purpose of each image. There are as many uses for web graphics as there are web pages, but four types of graphics are by far the most common:

- Photos of people, products, or places
- Graphical banners and logos
- Buttons or icons to indicate actions and provide links
- Background textures for container elements

9.3 PREPARING PHOTOGRAPHIC IMAGES

To put photos on your web pages, you need to convert your print-based photos to digital images or create photos digitally by using a digital camera. You might need to use the custom software that comes with your scanner or camera to save pictures onto your hard drive, or you can just drag and drop files from your camera to your hard drive. After you transfer the digital image files to your computer, you can use your graphics program to crop, resize, color-correct, and compress to get them ready for use in your website.

9.3.1 Cropping an Image

Because you want web page graphics to be as compact as possible, you'll usually need to crop your digital photos. When you crop a photo, you select the area you want to display, and you crop away the rest.

The GIMP toolbox offers quick access to the crop tool and its possible attributes. Find an image file—either a digital image you have taken with your camera and stored on your hard drive or an image you have found online. After opening the image in GIMP, perform the following steps to crop it in GIMP:

1. In the GIMP toolbox, click the crop tool (see Figure 3). Depending on the tool you select, there might be additional attributes you can select. For example, Figure 3 shows the attributes for the cropping tool (such as the aspect ratio, position, size, and so on).

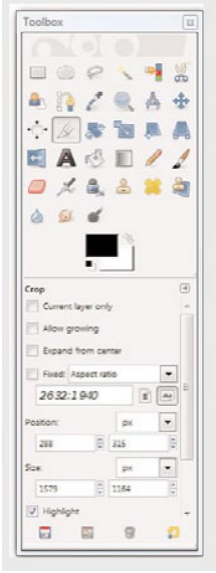


Figure 3. Select the crop tool from the toolbox.

2. In the image you want to crop, draw a box around the selection by clicking the upper-left corner of the portion of the image you want to keep and holding the left mouse button while you drag down to the lowerright corner (see Figure 4).

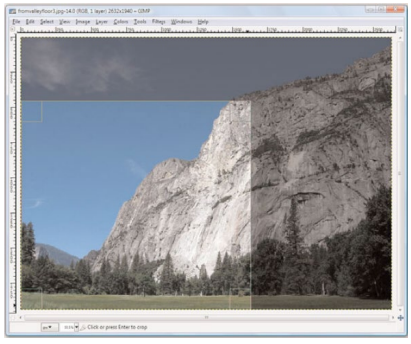


Figure 4. Select the area of the image that you want to display.

3. Click one of the corners of the selection to apply the cropping.

   Even after your image has been cropped, it might be larger than it needs to be for a web page. Depending on the design of a specific web page, you might want to limit large images to no more than 800×600 pixels (if it is shown on a page by itself, such as an item catalog) or even 640×480 pixels or smaller.

When shown alongside text, images tend to be in the 250 to 350 pixel range for width, so there's just enough room for the text as well. In some cases, you might want to also provide a thumbnail version of the image that links to a larger version, in which case you'll probably stick closer to 100 pixels in the larger dimension for the thumbnail.

9.3.2 Resizing an Image

The exact tool necessary to change an image's size will depend on the program you are using. In GIMP, go to the Image menu and click Scale Image to open the Scale Image dialog box (see Figure 5).

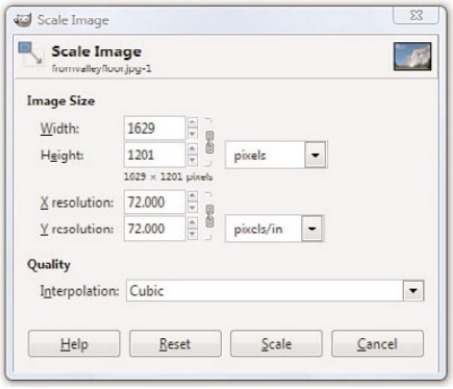


Figure 5. Use the Scale Image dialog box to change the size of an image.

You'll almost always want to resize using the existing aspect ratio, meaning that when you enter the width you'd like the image to be, the height will be calculated automatically (and vice versa) to keep the image from squishing out of shape. In GIMP, the aspect ratio is locked by default, as indicated by the chain link displayed next to the Width and Height options shown in Figure 5. Clicking once on the chain will unlock it, enabling you to specify pixel widths and heights of your own choosing—squished or not.

In most, if not all, graphics programs, you can also resize the image based on percentages instead of providing specific pixel dimensions. For example, if my image started out as 1629×1487, and I didn't want to do the math to determine the values necessary to show it as half that width, I could simply select Percent (in this instance from the drop-down next to the pixel display shown in Figure 5) and change the default setting (100) to 50.

9.3.3 Tweaking Image Colors

If you are editing photographic images rather than creating your own graphics, you might need to use some color-correction tools to get the photo just right. Like many image editing programs, GIMP offers several options for adjusting an image's brightness, contrast, and color balance, as well as a filter to reduce the dreaded red-eye. To remove red-eye using GIMP, go to Filters, click Enhance, and then click Red Eye Removal.

Most of these options are pretty intuitive. If you want the image to be brighter, adjust the brightness. If you want more red in your image, adjust the color balance. In GIMP, the Colors menu gives you access to numerous tools. Each tool displays a dialog box in the foreground of your workspace. As you adjust the colors, the image reflects those changes. This preview function is a feature included in most image editing software.

Figure 6 shows the Adjust Hue/Lightness/Saturation tool, one of the many tools provided on the Colors menu. As shown in the figure, many color-related changes occur by using various sliders in dialog boxes to adjust the values you are working with. The Preview feature enables you to see what you are doing as you are doing it. The Reset Color button returns the image to its original state without any changes applied.

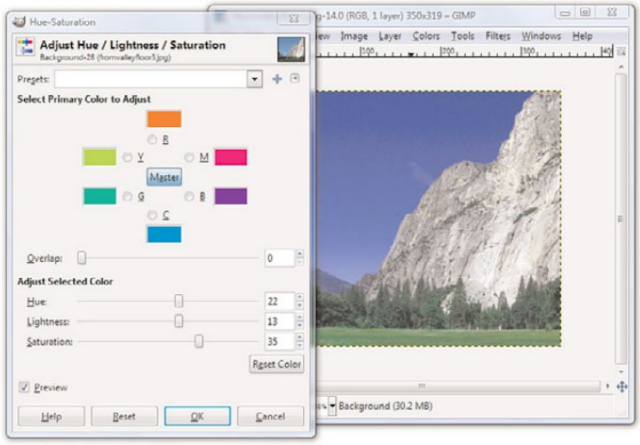


Figure 6. The Adjust Hue/Lightness/ Saturation tool is one of many slider-based color modification tools available in GIMP.

Because of the numerous tools available to you and the preview function available with each tool, a little playful experimentation is the best way to find out what each tool does.

9.3.4 Controlling JPEG Compression

Photographic images on the Web work best when saved in the JPEG file format rather than GIF; JPEG enables you to retain the number of colors in the file while still keeping the overall file size to a manageable level. When you're finished adjusting the size and appearance of your photo, select File, Save As and choose JPEG as the file type. Your graphics program will likely provide you with another dialog box through which you can control various JPEG options, such as compression.

Figure 7 shows the Save as JPEG dialog box you'll see when you save a JPEG in GIMP. You can see here that you can control the compression ratio for saving JPEG files by adjusting the Quality slider between 1 (low quality, small file size) and 100 (high quality, large file size).

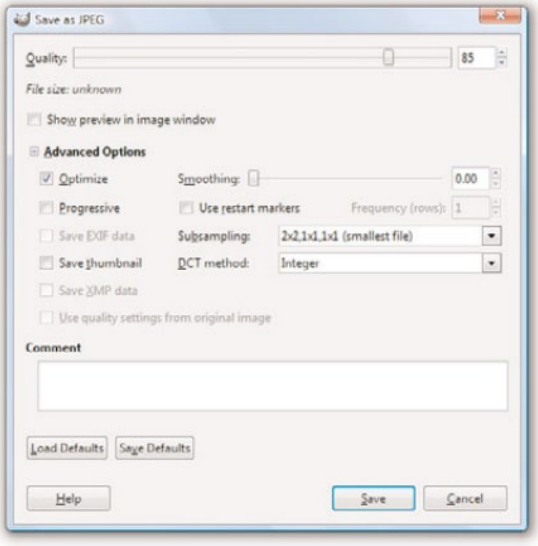


Figure 7. GIMP enables you to reduce file size while still retaining image quality by saving in the JPEG format.

You might want to experiment a bit to see how various JPEG compression levels affect the quality of your images, but 85% quality (or 15% compression) is generally a good compromise between file size (and therefore download speed) and quality for most photographic images.

9.3.5 Creating Banners and Buttons

Graphics that you create from scratch, such as banners and buttons, require you to make considerations uniquely different from photographs.

The first decision you need to make when you produce a banner or button is how big it should be. Most people accessing the web now have a computer with a screen that is at least 1024×768 pixels in resolution, if not considerably larger. For example, my screen is currently set at 1440×900 pixels. You should generally plan your graphics so that they will always fit within smaller screens (1024×768), with room to spare for scrollbars and margins. The crucial size constraint is the horizontal width of your pages because scrolling a page horizontally is a huge hassle and a source of confusion for web users. Vertically scrolling a page is much more acceptable, so it's okay if your pages are taller than the minimum screen sizes.

Assuming that you target a minimum resolution of 800×600 pixels, full-sized banners and title graphics should be no more than 770 pixels wide by 430 pixels tall, which is the maximum viewable area of the page after you've accounted for scrollbars, toolbars, and other parts of the browser window. Within a page, normal photos and artwork should be from 100 to 300 pixels in each dimension, and smaller buttons and icons should be 20 to 100 pixels tall and wide. Obviously, if you design for the 1024×768 resolution, you have more screen "real estate" to work with, but the size guidelines for banners, buttons, and other supplementary graphics are still in effect.

Did You Know?

For many years, designing for 800×600 screen resolution has been the norm. Still keep that low number in mind, as many people do not open applications in full-screen mode. However, designing for a baseline 1,024×768 screen resolution is not a bad idea.

To create a new image in GIMP, go to File and choose New. The Create a New Image dialog box displays (see Figure 8). If you aren't sure how big the image needs to be, just accept the default size of a 640×480. Or you can choose one of the other pre-determined sizes in the Template drop-down, such as Web banner common 468×60 or Web banner huge 728×90. Those two settings are indeed considered "common" and "huge" for website banners. Otherwise, enter the width and height of the new image.

For the image's background color, you should usually choose white to match the background that most web browsers use for web pages. When you know that you'll be creating a page with a background other than white, you can choose a different background color for your image. Or you might want to create an image with no background at all, in which case you'll select Transparency as the background color. When

an image's background is transparent, the web page behind the image is allowed to show through those areas of the image. In GIMP, select the background color for your new image by opening the Advanced Options in the Create a New Image dialog box.

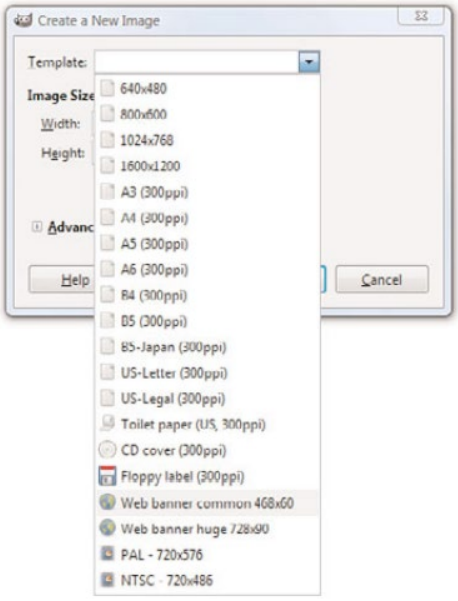


Figure 8. You need to decide on the size of an image before you start working on it.

9.3.6 Reducing the Number of Colors in an Image

One of the most effective ways to reduce the size of, and therefore the download time for, an image is to reduce the number of colors used in the image. This can drastically reduce the visual quality of some photographic images, but it works great for most banners, buttons, and other icons.

You'll be glad to know that there is a special file format for images with a limited number of colors; it's called the Graphics Interchange Format (GIF). When you save an image as a GIF, you might be prompted to flatten layers or reduce the number of colors by converting to an indexed image, as those are requirements for GIFs, as shown in Figure 9. The dialog box will simply ask you to confirm these changes that the save process will do for you—do not concern yourself with understanding these options at this time, but read your software's help file regarding layers and indexed colors for a full understanding.

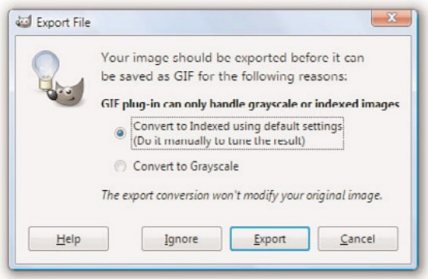


Figure 9. When saving an image as a GIF, you might be prompted to convert it to an indexed color palette.

Remember

The GIF image format is designed for images that contain areas of solid colors, such as web page titles and other illustrated graphics; the GIF format is not ideal for photographs.

PNG (pronounced "ping") is another useful file format that is supported in all major web browsers. Although the GIF image format enables you to specify a single transparent color, which means that the background of the web page will show through those areas of an image, the PNG format takes things a step further by enabling you to specify varying degrees of transparency.

9.3.7 Working with Transparent Images

You might have seen websites that use background colors or images in their container elements, but also have images present in the foreground that allow the background to show through parts of the foreground graphics. In these cases, the images in the foreground have portions which are transparent, so that the images themselves—which are always on a rectangular canvas—do not show the areas of the canvas in which the design does not occur. You'll often want to use these types of partially transparent images to make graphics look good over any background color or background image you might have in place.

To make part of an image transparent, the image must be saved in the GIF or PNG file format. Most graphics programs that support the GIF format enable you to specify one color to be transparent, whereas PNG images allow for a range of transparency. Largely because of this transparency range, the PNG format is superior to GIF. All the latest web browsers already support PNG images.

9.3.8 Creating Tiled Backgrounds

Any GIF or JPEG image can be used as a background tile within a container element. However, before you go off and create a tiled background, especially a highly patterned tiled background, ask yourself what that tiled background adds to the overall look and feel of your website, and—more importantly—ask yourself if the text of the site can be read easily when placed over that pattern. Think about the websites you frequent every day and consider the fact that few use tiled, patterned backgrounds on their entire page. If you restrict your browsing to websites for companies, products, sports teams, or other sites in which information (primarily text) is privileged, the number of sites with tiled, patterned backgrounds will decrease even further. Although the Web affords everyone the right of individuality in design, if you are creating a site for your business, you might want to avoid a highly patterned background with contrasting colored text. If you do use a tiled, patterned background image for your entire site, remember that tiled images look best when you can't tell they're tiled images. In other words, you know you have a good image when the top edge of a background tile matches seamlessly with the bottom edge, and the left edge matches with the right. Figures 10 and 11 show background tiles in use, both with seamless background, but with varying degrees of effectiveness.

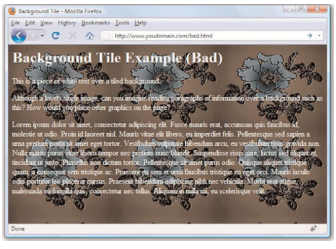


Figure 10. This is an example of a seamless background image whereby you can tell the background is tiled because you can see six identical shapes.

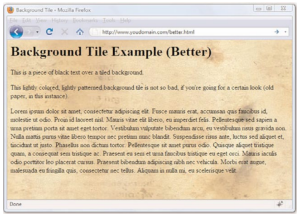


Figure 11. This is also an example of a seamless background image, only you can't tell that it is tiled.

9.4 CREATING ANIMATED WEB GRAPHICS

The GIF image format enables you to create animated images that can be used to add some motion that will spice up any web page. Animated GIF images also transfer much faster than most of the video or multimedia files that are often used for similar effect. With GIMP, you can create animated GIFs by creating multiple layers within an image, and then modifying the Animated GIF options when saving the file.

The first step in creating a GIF animation is to create a series of images to be displayed one after the other—or a series of layers, depending on your particular software program. Each of these images is called a frame.

9.4.1 Placing Images on a Web Page

To put an image on a web page, first move the image file into the same folder as the HTML file or in a directory named images for easy organization.

Insert the following HTML tag at the point in the text where you want the image to appear. Use the name of your image file instead of myimage.gif:

If your image file were in the images directory below the document root, you would use the following:

Remember

It doesn't matter to the web server, web browser, or end user where you put your images, as long as you know where they are and as long as you use the correct paths in your HTML code

Both the src and the alt attributes of the <img /> tag are required in XHTML web pages. The src attribute identifies the image file, and the alt attribute enables you to specify descriptive text about the image, the latter of which is intended to serve as an alternative to the image in the event that a user is unable to view the image.

As an example of how to use the <img /> tag, Listing 2 inserts an image at the top of the page, before a paragraph of text. Whenever a web browser displays the HTML file in Listing 2, it automatically retrieves and displays the image file, as shown in Figure 12.

Listing 2. Using the <img /> Tag to Place Images on a Web Page

```
<?xml version= “1.0” encoding= “UTF-8”?>
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.1//EN”
“http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd”>
<html xmlns= “http://www.w3.org/1999/xhtml” xml:lang= “en”>
<head>
<title>A Spectacular Yosemite View</title>
</head>
<body>
<h1>A Spectacular Yosemite View</h1>
<p><img src= “hd.jpg” alt= “Half Dome” /></p>
<p><strong>Half Dome</strong> is a granite dome in Yosemite National
➥Park,
located in northeastern Mariposa County, California, at the eastern
end of Yosemite Valley. The granite crest rises more than 4,737 ft
(1,444 m) above the valley floor.</p>
<p>This particular view is of Half Dome as seen from Washburn
Point.</p>
</body>
</html>
```

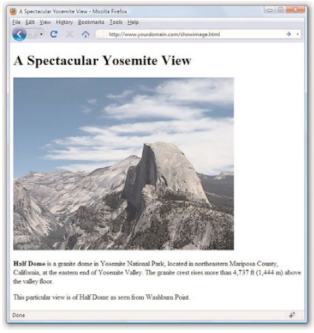


Figure 12. When a web browser displays the HTML code shown in Listing 2, it renders the hd.jpg image.

If you guessed that img stands for image, you’re right. And src stands for source, which is a reference to the location of the image file. An image is always stored in a file separate from the text, even though it appears to be part of the same page when viewed in a browser.

Just as with the <a href> tag used for hyperlinks, you can specify any complete Internet address in the src attribute of the <img /> tag. Alternatively, if an image is located in the same folder as the HTML file, you can specify just the filename. You can also use relative addresses, such as /images/birdy.jpg or ../smiley.gif.

9.4.2 Describing Images with Text

Each <img /> tag in Listing 2 includes a short text message, such as alt="Half Dome". The alt stands for alternate text, which is the message that appears in place of the image itself if it does not load. An image might not load if its address is incorrect, if the user has turned off automatic image downloading in her web browser preferences, or if the Internet connection is very slow and the data has yet to transfer. Figure 13 shows an example of alt text used in place of an image.

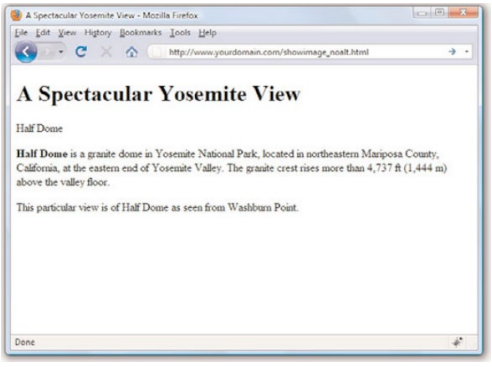


Figure 13. Users will see alt messages when images do not appear.

Even when graphics have fully loaded and are visible in the web browser, the alt message typically appears in a little box (known as a tool tip) whenever the mouse pointer passes over an image. The alt message also helps any user who is visually impaired (or is using a voice-based interface to read the web page).

You must include a suitable alt attribute in every <img /> tag on your web page, keeping in mind the variety of situations in which people might see that message. A brief description of the image is usually best, but web page authors sometimes put short advertising messages or subtle humor in their alt messages; too much humor and not enough information is frowned upon, however. For small or unimportant images, it's tempting to omit the alt message altogether, but it is a required attribute of the <img /> tag. This doesn't mean your page won't display properly, but it does mean you'll be in violation of the latest XHTML standards. I recommend assigning an empty text message to alt if you absolutely don't need it (`alt=""`), which is sometimes the case with small or decorative images.

The title attribute is not required by the <img /> tag, but it functions similarly to the alt attribute; in fact, the title attribute supersedes the alt attribute for tool tips if both attributes are present. Knowing this, the best approach for describing images via text is to use both the alt attribute and the title attribute to provide relevant notation or helpful hints about the image that you think might be useful when viewed as a tool tip or via screen reader software.

9.4.3 Specifying Image Height and Width

Because text moves over the Internet much faster than graphics, most web browsers display the text on a page before they display images. This gives users something to read while they're waiting to see the pictures, which makes the whole page seem to load faster. You can make sure that everything on your page appears as quickly as possible and in the right places by explicitly stating each image's height and width. That way, a web browser can immediately and accurately make room for each image as it lays out the page and while it waits for the images to finish transferring.

For each image you want to include in your site, you can use your graphics program to determine its exact height and width in pixels. You might also be able to find these image properties by using system tools. For example, in Windows, you can see an image's height and width by right clicking on the image, selecting Properties, and then selecting Details. After you know the height and width of an image, you can include its dimensions in the <img /> tag, like this:

```
<img src="myimage.gif" alt="Fancy Picture" width="200" height="100" />
```

9.4.4 Aligning Images

Just as you can align text on a page, you can align images on the page using special attributes. Not only can you align images horizontally, you also can align them vertically with respect to text and other images that surround them.

Horizontal Image Alignment

"Working with Fonts, Text Blocks, and Lists," you can use , and to align an element to the center, to the right margin, or to the left margin. These style settings affect both text and images and can be used within the <p> tag as well.

Like text, images are normally lined up with the left margin unless a style= "text-align:center" or style= "text-align:right" setting indicates that they should be centered or right-justified. In other words, left is the default value of the text-align style property.

You can also wrap text around images by using the float style property directly within the <img /> tag In Listing 3, aligns the first image to the left and wraps text around the right side of it, as you might expect. Similarly, aligns the second image to the right and wraps text around the left side of it. Figure 14 shows how these images align on a web page. There is no concept of floating an image to the center because there would be no way to determine how to wrap text on each side of it.

Listing 3. Using text-align Styles to Align Images on a Web Page

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>More Spectacular Yosemite Views</title>
</head>
<body>
<h1>More Spectacular Yosemite Views</h1>
<p><img src="elcap_sm.jpg" alt="El Capitan" width="100"
height="75" style="float:left; padding: 6px;"/><strong>El
Capitan</strong> is a 3,000-foot (910 m) vertical rock formation
in Yosemite National Park, located on the north side of Yosemite
Valley, near its western end. The granite monolith is one of the
world's favorite challenges for rock climbers. The formation was
named "El Capitan" by the Mariposa Battalion when it explored the
valley in 1851.</p>
<p><img src="tunnelview_sm.jpg" alt="Tunnel View" width="100"
height="80" style="float:right; padding: 6px;"/><strong>Tunnel
View</strong> is a viewpoint on State Route 41 located directly east
of the Wawona Tunnel as one enters Yosemite Valley from the south.
The view looks east into Yosemite Valley including the southwest face
of El Capitan, Half Dome, and Bridalveil Falls. This is, to many, the
first views of the popular attractions in Yosemite.</p>
</body>
</html>
```

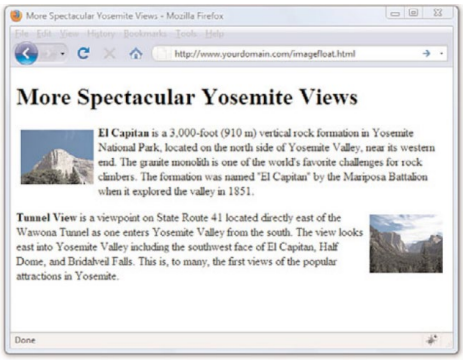


Figure 14. Showing the image alignment from Listing 3.

Vertical Image Alignment

Sometimes, you might want to insert a small image in the middle of a line of text; or you might like to put a single line of text next to an image as a caption. It would be handy to have some control over how the text and images line up vertically. Should the bottom of the image line up with the bottom of the letters, or should the text and images all be arranged so that their middles line up? You can choose between these and several other options:

- To line up the top of an image with the top of the tallest image or letter on the same line, use .
- To line up the bottom of an image with the bottom of the text, use .
- To line up the middle of an image with the overall vertical center of everything on the line, use .
- To line up the bottom of an image with the baseline of the text, use .

All four of these options are used in Listing 4 and displayed in Figure 15. Four thumbnail images are now listed vertically down the page, along with descriptive text next to each image. Various settings for the vertical-align style property are used to align each image and its relevant text.

Listing 4. Using vertical-align Styles to Align Text with Images

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>Small But Mighty Spectacular Yosemite Views</title>
</head>
<body>
<h1>Small But Mighty Yosemite Views</h1>
<p><img src="elcap_sm.jpg" alt="El Capitan" width="100"
height="75" style="vertical-align:text-top;"/><strong>El
Capitan</strong> is a 3,000-foot (910 m) vertical rock formation
in Yosemite National Park.</p>
<p><img src="tunnelview_sm.jpg" alt="Tunnel View" width="100"
height="80" style="vertical-align:text-bottom;"/><strong>Tunnel
View</strong> looks east into Yosemite Valley.</p>
<p><img src="upperyosefalls_sm.jpg" alt="Upper Yosemite Falls"
width="87" height="100" style="vertical-align:middle;"/><strong>Upper
Yosemite Falls</strong> are 1,430 ft and are among the twenty highest
waterfalls in the world. </p>
<p><img src="hangingrock_sm.jpg" alt="Hanging Rock" width="100"
height="75" style="vertical-align:baseline;"/><strong>Hanging
Rock</strong>, off Glacier Point, used to be a popular spot for people
to, well, hang from. Crazy people.</p>
</body>
</html>
```

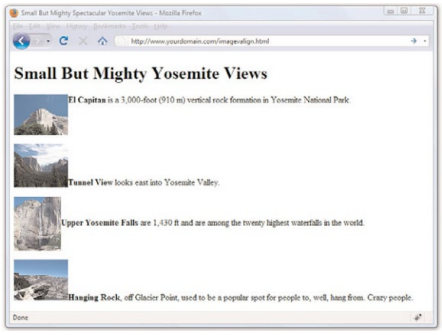


Figure 15. Showing the vertical image alignment options used in Listing 4.

9.4.5 Turning Images into Links

You probably noticed in Figure 12 that the image on the page is quite large, which is fine in this particular example but isn't ideal when you're trying to present multiple images. It makes more sense in this case to create smaller image thumbnails that link to larger versions of each image. Then, you can arrange the thumbnails on the page so that visitors can easily see all the content, even if they see only a smaller version of the actual (larger) image. Thumbnails are one of the many ways you can use image links to spice up your pages.

To turn any image into a clickable link to another page or image, you can use the <a href> tag that you used previously to make text links. Listing 5 contains the code for a modification of Listing 3—which already used thumbnails—to provide links to larger versions of the images. To ensure that the user knows to click the thumbnails, the image and some helper text is enclosed in a , as shown in Figure 16.

Listing 5. Using Thumbnails for Effective Image Links

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>More Spectacular Yosemite Views</title>
<style type="text/css">
div.imageleft {
float:left;
clear: all;
text-align:center;
font-size:9px;
font-style:italic;
}
div.imageright {
float:right;
clear: all;
text-align:center;
font-size:9px;
font-style:italic;
}
```

```
img {
padding: 6px;
border: none;
}
</style>
</head>
<body>
<h1>More Spectacular Yosemite Views</h1>
<p><div class="imageleft">
<a href="http://www.flickr.com/photos/nofancyname/614253439/"><img
src="elcap_sm.jpg" alt="El Capitan" width="100" height="75"/></a>
<br/>click image to enlarge</div><strong>El Capitan</strong>
is a 3,000-foot (910 m) vertical rock formation in Yosemite National
Park, located on the north side of Yosemite Valley, near its western
end. The granite monolith is one of the world's favorite challenges
for rock climbers. The formation was named "El Capitan" by the
Mariposa Battalion when it explored the valley in 1851.</p>
<p><div class="imageright">
<a href="http://www.flickr.com/photos/nofancyname/614287355/"><img
src="tunnelview_sm.jpg" alt="Tunnel View" width="100" height="80"/></a>
<br/>click image to enlarge</div><strong>Tunnel View</strong> is a
viewpoint on State Route 41 located directly east of the Wawona Tunnel
as one enters Yosemite Valley from the south. The view looks east into
Yosemite Valley including the southwest face of El Capitan, Half Dome,
and Bridalveil Falls. This is, to many, the first views of the popular
attractions in Yosemite.</p>
</body>
</html>
```

The code in Listing 5 uses additional styles, but you should be able to figure out the basics:

- The <a> tags link these particular images to larger versions, which in this case are stored on an external server (at Flickr).
- The tags, and their styles, are used to align those sets of graphics and caption text (and also include some padding).

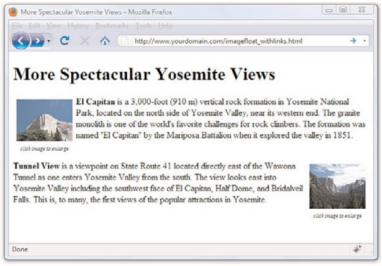


Figure 16. Using thumbnails as links improves the layout of a page that uses large images.

Unless instructed otherwise, web browsers display a colored rectangle around the edge of each image link. Like text links, the rectangle usually appears blue for links that haven't been visited recently or purple for links that have been visited recently—unless you specify different colored links in your style sheet. Because you seldom, if ever, want this unsightly line around your linked images, you should usually include style="border:none" in any <img /> tag within a link. In this instance, the border:none style is made part of the style sheet entry for the img element because we use the same styles twice.

When you click one of the thumbnail images on the sample page shown, the link is opened in the browser, as shown in Figure 17.

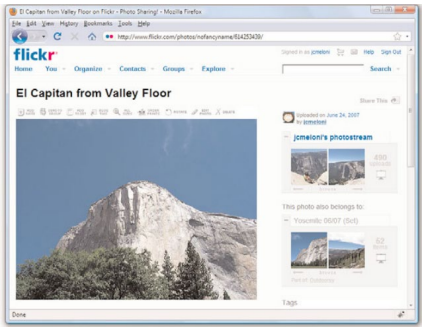


Figure 17. Clicking a linked thumbnail image opens the target of the link.

9.4.6 Using Background Images

You can use background images to act as a sort of wallpaper in a container element, so that the text or other images appear on top of this underlying design. The basic style properties that work together to create a background are as follows:

- background-color—Specifies the background color of the element. Although not image-related, it is part of the set of background-related properties.
- background-image—Specifies the image to use as the background of the element using the following syntax: url('imagename.gif').
- background-repeat—Specifies how the image should repeat, both horizontally and vertically. By default (without specifying anything), background images will repeat both horizontally and vertically. Other options are repeat (same as default), repeat-x (horizontal), repeat-y (vertical), and no-repeat (only one appearance of the graphic).
- background-position—Specifies where the image should be initially placed relative to its container. Options include top-left, top-center, top-right, center-left, center-center, center-right, bottom-left, bottom-center, bottom-right, and specific pixel and percentage placements.

When specifying a background image, you can put all of these specifications together into one property, like so:

```
body {
background: #ffffff url('imagename.gif') no-repeat top right;
}
```

In the previous style sheet entry, the body element of the web page will be white and include a graphic named imagename.gif on the top right. Another use for the background property is the creation of custom bullets for your unordered lists. To use images as bullets, first define the style for the <ul> tag as follows:

```
ul {
list-style-type: none;
padding-left: 0;
margin-left: 0;
}
```

Next, change the declaration for the <li> tag to:

```
li {
background: url(mybullet.gif) left center no-repeat
}
```

Make sure that mybullet.gif (or whatever you name your graphic) is on the web server and accessible; in that case, all unordered list items will show your custom image rather than the standard-filled disc bullet.

9.4.7 Using Imagemaps

Sometimes you might want to use an image as navigation, but beyond the simple button-based or link-based navigation that you often see in websites. For example, perhaps you have a website with medical information, and you want to show an image of the human body that links to pages that provide information about various body parts. Or, you have a website that provides a world map users can click to access information about countries. You can divide an image into regions that link to different documents, depending on where users click within that image. This is called an imagemap, and any image can be made into an imagemap.

Why Imagemaps Aren't Always Necessary

The first thing you should know about imagemaps is that you probably won't need to use them except in very special cases. It's almost always easier and more efficient to use several ordinary images that are placed directly next to one another and provide a separate link for each image.

For example, see Figure 18. This is a web page that shows 12 different corporate logos; this example is a common type of web page in the business world, one in which you give a little free advertisement to your partners. You could present these logos as one large image and create an imagemap that provides links to each of the 12 companies. Users could click each logo in the image to visit each company's site. Or, you could display the images on the page as in this example and use 12 separate images—one for each company—with each image including a link to that particular company.

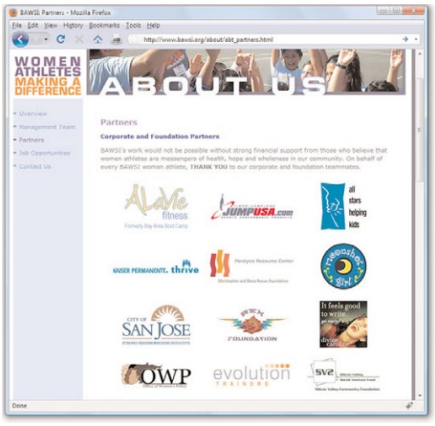


Figure 18. Web page with 12 different logos; this could be presented as a single imagemap or divided into 12 separate pieces.

An imagemap is the best choice for an image that has numerous parts, is oddly arranged, or the design of the image itself might be too complicated to divide into separate images. Figure 19 shows an image that is best suited as an imagemap.

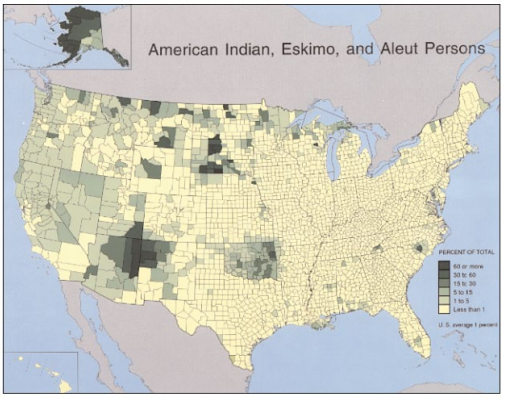


Figure 19. An image that wouldn't take well to being sliced up—better make it an imagemap.

Mapping Regions within an Image

To create any type of imagemap, you need to figure out the numerical pixel coordinates of each region within the image that you want to turn into a clickable link. These clickable links are also known as areas. Your graphics program might provide you with an easy way to find these coordinates. Or you might want to use a standalone imagemapping tool such as Mapedit. In addition to helping you map the coordinates, these tools also provide the HTML code necessary to make the maps work.

Using an imagemapping tool is often as simple as using your mouse to draw a rectangle (or a custom shape) around the area you want to be a link. Figure 20 shows the result of one of these rectangular selections as well as the interface for adding the URL and the title or alternate text for this link. Several pieces of information are necessary to creating the HTML for your imagemap: coordinates, target URL, title of link, and alternative text for the link.

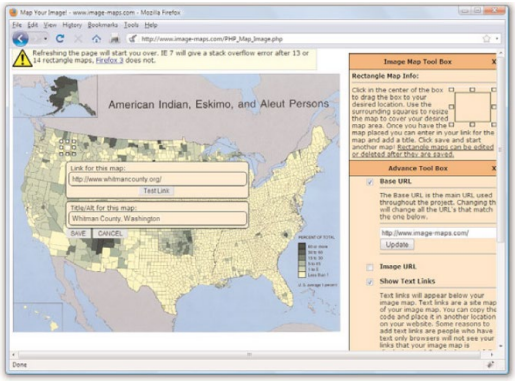


Figure 20. Using an imagemapping tool to create linked areas of a single graphic.

9.4.8 Creating the HTML for an Imagemap

If you use an imagemap generator, you will already have the HTML necessary for creating the imagemap. However, it is a good idea to understand the parts of the code so that you can check it for accuracy. The following HTML code is required to start any imagemap:

<map name= “mapname”>

Keep in mind that you can use whatever name you want for the name of the <map> tag, although it helps if you make it as descriptive as possible. Next, you’ll need an <area /> tag for each region of the image. Following is an example of a single <area /> tag that is used in the mapping-a-map imagemap example:

<area shape=”rect” coords=”100,136,116,152”

href=”http://www.whitmancounty.org/”

alt=”Whitman County, WA”

title=”Whitman County, WA” />

This <area /> tag has five attributes, which you will use with every area you describe in an imagemap:

- shape indicates whether the region is a rectangle (shape=”rect”), a circle (shape=”circle”), or an irregular polygon (shape=”poly”).
- coords gives the exact pixel coordinates for the region. For rectangles, give the x,y coordinates of the upper-left corner followed by the x,y coordinates of the lower-right corner. For circles, give the x,y center point followed by the radius in pixels. For polygons, list the x,y coordinates of all the corners in a connect-the-dots order.

- href specifies the page to which the region links. You can use any address or filename that you would use in an ordinary <a href> link tag.
- alt enables you to provide a piece of text that is associated with the shape. Most browsers (Firefox excluded) display this text in a small box when a user hovers his mouse over the shape. This text adds a subtle but important visual cue to users who might not otherwise realize that they are looking at an imagemap. Firefox correctly uses the title attribute in addition to the alt attribute to provide a visual cue, which is why, you should use both attributes.

Each distinct clickable region in an imagemap must be described as a single area, which means a typical imagemap consists of a list of areas. After coding the <area /> tags, you are done defining the imagemap, so wrap things up with a closing </map> tag.

The last step in creating an imagemap is wiring it to the actual map image. The map image is placed on the page using an ordinary <img /> tag. However, there is an extra usemap attribute that is coded like this:

```
<img src="map.png" usemap="#countymap"
style="border:none; width:650px; height:509px"
alt="Native Peoples Census Map" />
```

When specifying the value of the usemap attribute, use the name you put in the id of the <map> tag (and don't forget the # symbol). Also include the style attribute to specify the height and width of the image and to turn off the border around the imagemap, which you might or might not elect to keep in imagemaps of your own.

Listing 6 shows the complete code for a sample web page containing the map graphic, its imagemap, and a few mapped areas.

Listing 6. Defining the Regions of an Imagemap with and Tags

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>Testing an Imagemap</title>
</head>
<body>
<h1>Testing an Imagemap</h1>
<p style="text-align:center">Click on a logo to go to the
county's web site.<br/>
<img src="map.png" usemap="#countymap"
```

```
style="border:none; width:650px;height:509px"
alt="Native Peoples Census Map" /></p>
<map name="countymap" id="countymap">
<area shape="rect" coords="100,136,116,152"
href="http://www.whitmancounty.org/"
alt="Whitman County, WA" title="Whitman County, WA" />
<area shape="rect" coords="29,271,42,283"
href="http://www.sccgov.org/" alt="Santa Clara County, CA"
title="Santa Clara County, CA" />
<area shape="rect" coords="535,216,548,228"
href="http://visitingmifflincounty.com/"
alt="Mifflin County, PA" title="Mifflin County, PA" />
</map>
</body>
</html>
```

Figure 21 shows the imagemap in action. Notice in the bottom of your browser window that your browser (in this example, Firefox) displays the link address for whatever area the mouse is hovering over. Additionally, when you hover the mouse over an area, the alt or title text for that area—in this example, Whitman County—is displayed on the imagemap.

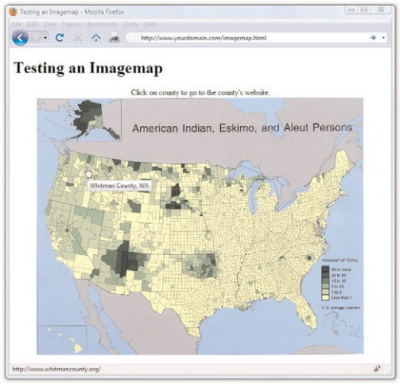


Figure 21. The imagemap defined in Listing 6 as it displays on the web page.

9.5 INTEGRATING MULTIMEDIA INTO YOUR WEBSITE

The term multimedia encompasses everything we see and hear on a web page: audio, video, and animation, as well as static images and text. You won't learn how to create any particular audio, video, or animation, but you will learn how to include such files in your site, through either linking or embedding the content.

Remember, though, that not every user has devices that will play your media type, nor do all users have broadband Internet connections which allow these large files to transfer quickly. Always warn your visitors that the links they click will take them to multimedia files and offer them the choice to view or listen to the content—don't force the files upon them.

Creating multimedia of any kind can be a challenging and complicated task. If you're planning to create your own content from scratch. For those of us who are artistically challenged, several alternative ways to obtain useful multimedia assets are available. Aside from the obvious (such as hiring an artist), here are a few suggestions:

- Much of the material on the Internet is free. Of course, it's still a good idea to double-check with the author or current owner of the content; you don't want to be sued for copyright infringement. In addition, various offices of the U.S. government generate content which, by law, belongs to all Americans. (For example, any NASA footage found online is free for your use.)
- Many search engines (google.com, yahoo.com, bing.com, and so on) have specific search capabilities for finding multimedia files. As long as you are careful about copyright issues, this can be an easy way to find multimedia related to a specific topic. A simple search for sample Flash animations, sample QuickTime movie, or sample audio files will produce more results than you can handle.
- If you are creatively inclined, determine the medium you like most— for some of you it might be video production, others may enjoy audio production, and still others might want to dabble in animation. After you have determined a starting point, look into the various types of software which will enable you to create such artistic masterpieces.

9.5.1 Linking to Multimedia Files

The simplest and most reliable option for incorporating a video or audio file into your website is to simply link it in with , exactly as you would link to another HTML file.

For example, the following line could be used to offer a MOV video of a hockey game:

View the hockey video clip.

When the user clicks the words View the hockey video clip., the hockey.mov QuickTime video file is transferred to her computer from your web server. Whichever helper application or plug-in she has installed automatically starts as soon as the file has finished downloading. If no compatible helper or plug-in can be found, the web browser will offer her a chance to download the appropriate plug-in or save the video on her hard drive for later viewing. The click action results in the browser either playing the video with the help of a plug-in (if one is found that can play the clip) or deferring to a suitable helper application. If you change the link from pond.wmv (Windows Media) to pond.mov (QuickTime), your browser handles the link differently. Instead of launching another program, the QuickTime plug-in enables you to view the movie clip directly in the browser window.

As you might have guessed, this approach of using a simple link to play multimedia files offers the best backward compatibility because the browser bears all the responsibility of figuring out how to play a multimedia clip. The downside to this is that you don't have much control over how a clip is played, and you definitely can't play a clip directly in the context of a page.

9.5.2 Embedding Multimedia Files

XHTML contains a standard <object> tag that is the preferred way to embed multimedia of any kind in a web page. This tag is used instead of the old <embed /> tag that you might still see in some HTML source code.

Embedding a multimedia file into a page produces a set of software controls that allow the file to be played directly—no secondary window is necessary, and there's no need to navigate away from the page you are on.

Following is code to embed the pond video using the <object> tag by itself:

```
<object classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6"
width="320" height="305">
<param name="type" value="video/x-ms-wmv" />
<param name="URL" value="pond.wmv" />
<param name="uiMode" value="full" />
<param name="autoStart" value="false" />
</object>
```

This code isn't too terribly complicated when you consider that it literally embeds a video directly into your web page (see Figure 22). The messiest part of the code is the classid attribute of the <object> tag, which is set to a long alphanumeric code. This code is the global ID for Windows Media Player, which means that you're telling the <object> tag to embed Windows Media Player on the page to play the video clip. You can just copy and paste this code into your own web pages.

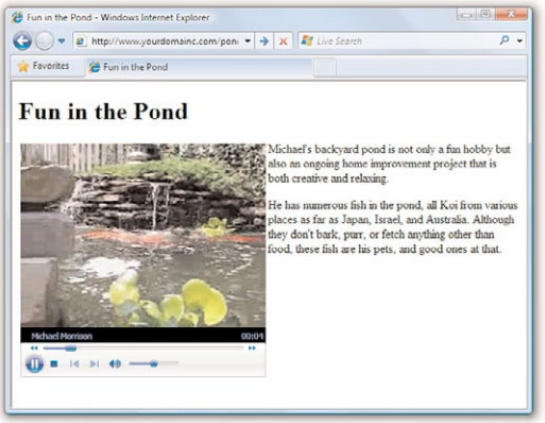


Figure 22. The <object> tag enables you to embed a video clip on a web page while specifying which media player is to play it.

The width and height attributes of the <object> tag determine the size of the embedded Windows Media Player window. Some browsers will automatically size the embedded player to fit the content if you leave these attributes off, whereas others won't show anything at all. Play it safe by setting them to a size that suits the multimedia content being played.

There are four <param> tags within the <object> tag that are responsible for additional details about how the clip is to be played. Each tag has two attributes, name and value, which are responsible for associating data (value) with a particular setting (name). In this example, the URL for the media clip is set to pond.wmv. The third parameter, uiMode, determines which buttons and user interface options are made available by Windows Media Player—full indicates that all user interface features are enabled, such as the control buttons and volume slider. Finally, the autoStart parameter is set to false so that the video clip does not automatically start playing when the page is opened in a browser.

The type parameter is perhaps the trickiest. It identifies the type of media being displayed, which in this case is a Windows Media Video (WMV) file. This media type must be specified as one of the standard Internet MIME types. A MIME type is an identifier for uniquely identifying different types of media objects on the Internet. MIME stands for Multipurpose Internet Mail Extensions, and this name comes from the fact that MIME types were originally used to identify email attachments. These MIME types should be used in the type attribute of the

Following are the MIME types for several popular sound and video formats you might want to use in your web pages:

- WAV Audio—audio/x-wav
- AU Audio—audio/basic

- MP3 Audio—audio/mpeg
- MIDI Audio—audio/midi
- WMA Audio—audio/x-ms-wma
- RealAudio—audio/x-pn-realaudio-plugin
- AVI—video/x-msvideo
- WMV—video/x-ms-wmv
- MPEG Video—video/mpeg
- QuickTime—video/quicktime

Listing 7 shows the relevant code for the pond web page, where you can see the <object> tag as it appears in context.

Listing 7. Using an <object> Tag to Directly Embed a WMV Video Clip

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>Fun in the Pond</title>
</head>
<body>
<h1>Fun in the Pond</h1>
<div style="float:left; padding:3px">
<object classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6"
width="320" height="305">
<param name="type" value="video/x-ms-wmv" />
<param name="URL" value="pond.wmv" />
<param name="uiMode" value="full" />
<param name="autoStart" value="false" />
<embed width="320" height="305" type="video/x-ms-wmv"
src="pond.wmv" controls="All" loop="false" autostart="false"
pluginspage="http://www.microsoft.com/windows/windowsmedia/" />
</object>
</div>
<p>Michael's backyard pond is not only a fun hobby but also
```

```
an ongoing home improvement project that is both creative and
relaxing.</p>
<p>He has numerous fish in the pond, all Koi from various places
as far as Japan, Israel, and Australia. Although they don't bark,
purr, or fetch anything other than food, these fish are his pets,
and good ones at that.</p>
</body>
</html>
```

You might notice that there's some extra code that didn't appear in the earlier <object> tag example. For this reason, it is necessary to include an <embed /> tag within the <object> tag to account for browser inconsistencies. This isn't an ideal solution, but it's all we have while browser vendors continue to lag behind prevailing standards. If you pay close attention, you'll notice that the <embed /> tag contains all the same information as the <object> tag.

The <object> tag is a bit more complex than what is revealed here. However, you don't need to know how to use the more advanced facets of the <object> tag just to play multimedia content. In other words, it isn't important for you to become a multimedia guru to share some multimedia clips on your web pages.

SUMMARY

- Multimedia comes in many different formats. It can be almost anything you can hear or see, like images, music, sound, videos, records, films, animations, and more.
- The hexadecimal value of a color is an indication of how much red, green, and blue light should be mixed into each color. It works a little bit like Play-Doh—just mix in the amounts of red, blue, and green you want to get the appropriate color.
- Two forces are always at odds when you post graphics and multimedia on the Internet. The users' eyes and ears want all your content to be as detailed and accurate as possible, and they also want that information displayed immediately.
- Resolution is usually specified as the width times the height of the image, expressed in pixels; a 300×200 image, for example, is 300 pixels wide and 200 pixels high.
- The GIF image format enables you to create animated images that can be used to add some motion that will spice up any web page. Animated GIF images also transfer much faster than most of the video or multimedia files that are often used for similar effect.
- An imagemap is the best choice for an image that has numerous parts, is oddly arranged, or the design of the image itself might be too complicated to divide into separate images.
- The term multimedia encompasses everything we see and hear on a web page: audio, video, and animation, as well as static images and text. You won't learn how to create any particular audio, video, or animation, but you will learn how to include such files in your site, through either linking or embedding the content.

KNOWLEDGE CHECK

1. **Which of the following CSS property is used to set the background image of an element?**
   a. background-attachment
   b. background-image
   c. background-color
   d. None of the above
2. **Which of the following display property value is described by used internally by browsers to create list items. Of no practical value to web designers?**
   a. inline-block
   b. list-item
   c. group
   d. none
3. **Which of the following function creates a CSS gradient image that can be used anywhere an image URL is required, including background-image, border-image, and list-style properties?**
   a. canvas
   b. Gradient
   c. Animation
   d. color
4. **How to insert a background image in HTML?**
   a. <body background = “img.png”>
   b. <img background = “img.png”>
   c. <bg-image = “img.png”>
   d. None of the above
5. **Which of the following tag is used to create a combo box (or drop-down box)?**
   a. <list>
   b. <select>
   c. <input type = “dropdown”>
   d.

REVIEW QUESTIONS

1. You have a scanned picture of a horse that you need to put on a web page. How big should you make it? In what file format should you save it?
2. How would you insert an elephant.jpg image file at the top of a web page?
3. What is hexadecimal values for colors?
4. What is animated web graphics?
5. How to set background images?

Check Your Result

1. (b) 2. (b) 3. (b) 4. (a) 5. (b)

REFERENCES

1. Cao, Yingxia; Ajjan, Haya; Hong, Paul (July 2013). "Using social media applications for educational outcomes in college teaching: A structural equation analysis: Social media use in teaching". British Journal of Educational Technology. 44 (4): 581–593.
2. Chang, Morris (2005). "Computer Architecture". The Electrical Engineering Handbook. pp. 323–334.
3. Irby, Beverly J; Brown, Genevieve; Lara-Alecio, Rafael; Jackson, Shirley, eds. (2013). The Handbook of Educational Theories.
4. Pierce, Glenn L.; Cleary, Paul F. (July 2016). "The K-12 educational technology value chain: Apps for kids, tools for teachers and levers for reform". Education and Information Technologies. 21 (4): 863–880
5. Pun, Min (23 May 2014). "The Use of Multimedia Technology in English Language Teaching: A Global Perspective". Crossing the Border: International Journal of Interdisciplinary Studies. 1 (1): 29–38.
6. Wingfield, Nick; Isaac, Mike (11 July 2016). "Pokémon Go Brings Augmented Reality to a Mass Audience". The New York Times. Retrieved 24 January 2021.

Index

Level: Beginner to Advanced
Subject: Computer and Information Science

Basic Computer Coding: CSS

2nd Edition

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is used to style and lay out your content, whereas HTML is used to define its structure and semantics. CSS, for example, can be used to change the font, color, size, and spacing of your content, divide it into multiple columns, or add animations and other decorative elements.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. Cascading Style Sheets (CSS) are an important way to control how your Web pages look. CSS can control the fonts, text, colors, backgrounds, margins, and layout. CSS is easy to learn and understand but it entails powerful control to the presentation of an HTML document. Usually, CSS is integrated with the markup languages HTML or XHTML.

This book is systematically organized into nine chapters in this edition. This edition is packed up with updated information. With this book, you can familiarize yourself with how CSS works, learn how to efficiently work with CSS selectors, and apply what you've learned to create beautifully styled simple web pages.

BIBLIOTEX
Digital Library

www.3ge-learning.com
email: info@3ge-learning.com

ISBN 978-1-98467-602-3