

WEB SITE DESIGN



Web Site Design - HTML

This text is disseminated via the Open Education Resource (OER) LibreTexts Project (<https://LibreTexts.org>) and like the thousands of other texts available within this powerful platform, it is freely available for reading, printing, and "consuming."

The LibreTexts mission is to bring together students, faculty, and scholars in a collaborative effort to provide an accessible, and comprehensive platform that empowers our community to develop, curate, adapt, and adopt openly licensed resources and technologies; through these efforts we can reduce the financial burden born from traditional educational resource costs, ensuring education is more accessible for students and communities worldwide.

Most, but not all, pages in the library have licenses that may allow individuals to make changes, save, and print this book. Carefully consult the applicable license(s) before pursuing such effects. Instructors can adopt existing LibreTexts texts or Remix them to quickly build course-specific resources to meet the needs of their students. Unlike traditional textbooks, LibreTexts' web based origins allow powerful integration of advanced features and new technologies to support learning.



LibreTexts is the adaptable, user-friendly non-profit open education resource platform that educators trust for creating, customizing, and sharing accessible, interactive textbooks, adaptive homework, and ancillary materials. We collaborate with individuals and organizations to champion open education initiatives, support institutional publishing programs, drive curriculum development projects, and more.

The LibreTexts libraries are Powered by [NICE CXone Expert](#) and was supported by the Department of Education Open Textbook Pilot Project, the California Education Learning Lab, the UC Davis Office of the Provost, the UC Davis Library, the California State University Affordable Learning Solutions Program, and Merlot. This material is based upon work supported by the National Science Foundation under Grant No. 1246120, 1525057, and 1413739.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation nor the US Department of Education.

Have questions or comments? For information about adoptions or adaptations contact info@LibreTexts.org or visit our main website at <https://LibreTexts.org>.

This text was compiled on 05/15/2026

TABLE OF CONTENTS

Licensing

1: Creating Web Pages

- 1.1: Internet History and Usage
- 1.2: Serving Web Pages
- 1.3: HTML Tags and CSS Styles
- 1.4: HTML Tags and CSS Styles
- 1.5: Creating Your First Web Page

2: Web Site Design

- 2.1: Goal identification
- 2.2: Scope definition
- 2.3: Sitemap and wireframe
- 2.4: Content Creation
- 2.5: Visual elements
- 2.6: Testing
- 2.7: Web Site Launch

3: Basic Document Layout

- 3.1: Structuring Page Content
- 3.2: Document Structure Tags
- 3.3: Paragraphs
- 3.4: Line Breaks
- 3.5: Headings
- 3.6: Horizontal Rules
- 3.7: List Structures
- 3.8: Linking Pages
- 3.9: Displaying Pictures

4: Basic Document Styling

- 4.1: Applying Styles
- 4.2: Applying Styles
- 4.3: Alignment Styles
- 4.4: Width and Height Styles
- 4.5: Basic Color Styles
- 4.6: Basic Color Styles
- 4.7: ID and Class Selectors

5: Text Formatting

- 5.1: Font Settings
- 5.2: Text Properties
- 5.3: Font Style Tags
- 5.4: Text and Background Colors
- 5.5: Special Characters

6: Graphic Images

- 6.1: Graphic Formats
- 6.2: Displaying Images
- 6.3: Background Images
- 6.4: Image Transparency

7: Applying Special Styles

- 7.1: Styling Lists
- 7.2: Borders and Padding
- 7.3: Sizing Elements
- 7.4: Positioning Elements

8: Linking Pages

- 8.1: Text Links
- 8.2: Graphic Links
- 8.3: Image Maps
- 8.4: Redirecting Pages
- 8.5: Traversing Folders

9: Using Tables

- 9.1: Creating Tables
- 9.2: Table Borders
- 9.3: Table Captioning
- 9.4: Table Colors and Backgrounds
- 9.5: Table Size and Alignment
- 9.6: Aligning Text and Cells
- 9.7: Column and Row Spanning
- 9.8: Table Groupings

10: Playing Multimedia

- 10.1: Audio and Video Formats
- 10.2: Playing Audio
- 10.3: Playing Video
- 10.4: Third Party Content

11: Creating Forms

- 11.1: Form Controls
- 11.2: Creating Forms
- 11.3: Textbox Controls
- 11.4: Textarea Control
- 11.5: Radio Buttons
- 11.6: Checkboxes
- 11.7: Drop-Down Lists
- 11.8: Submit and Reset Buttons
- 11.9: Other Buttons
- 11.10: Grouping and Labels
- 11.11: Submitting Forms

- [11.12: Forms and CSS](#)

[12: HTML/CSS Appendix](#)

- [12.1: New HTML5 Tags](#)
- [12.2: HTML Reference](#)
- [12.3: CSS Reference](#)
- [12.4: HTML Special Characters](#)
- [12.5: HTML Colors](#)

[Index](#)

[Glossary](#)

[Detailed Licensing](#)

Licensing

A detailed breakdown of this resource's licensing can be found in [Back Matter/Detailed Licensing](#).

CHAPTER OVERVIEW

1: Creating Web Pages

- [1.1: Internet History and Usage](#)
- [1.2: Serving Web Pages](#)
- [1.3: HTML Tags and CSS Styles](#)
- [1.4: HTML Tags and CSS Styles](#)
- [1.5: Creating Your First Web Page](#)

[1: Creating Web Pages](#) is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

1.1: Internet History and Usage

1.1: Internet History and Usage is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

1.2: Serving Web Pages

1.2: Serving Web Pages is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

1.3: HTML Tags and CSS Styles

1.3: HTML Tags and CSS Styles is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

1.4: HTML Tags and CSS Styles

1.4: HTML Tags and CSS Styles is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

1.5: Creating Your First Web Page

1.5: Creating Your First Web Page is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

2: Web Site Design

- 2.1: Goal identification
- 2.2: Scope definition
- 2.3: Sitemap and wireframe
- 2.4: Content Creation
- 2.5: Visual elements
- 2.6: Testing
- 2.7: Web Site Launch

2: Web Site Design is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

2.1: Goal identification

Know the website's purpose

To begin with, you need to identify the website's end goal, usually in close collaboration with the client or other stakeholders. Questions to explore and answer in this stage of the design and development process include:

- Who is the site for?
- What do they expect to find or do there?
- Is this website's primary aim to inform, to sell (ecommerce, anyone?), or to amuse?
- Does the website need to clearly convey a brand's core message, or is it part of a wider branding strategy with its own unique focus?
- What competitor sites, if any, exist, and how should this site be inspired by/different than, those competitors?

Determining a website's purpose is the most important part of any web design process. If these questions aren't all completely answered, the whole project can get off headed in the wrong direction. Make sure you understand the website's target audience, and develop a working knowledge of the competition.

Whether you're starting from the very beginning, or you're doing a total redesign of an existing website, you need to know what you want a user to take away from your new site. What do you want to communicate with the content? What calls to action need to be woven in? Understand what these focal points are so you create a design that magnifies them.

A website should not be a stand alone site, but should fit into a company's broader strategy. It needs to complement the company's strategy as well as add value of its own. It's fine if the website just provides a slice of fluffy entertainment — if that is really what the client wants. But ultimately, a website should help fill the gap between what customers know, and what your client wishes their customers knew.

2.1: Goal identification is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

2.2: Scope definition

What is Project Scope?

The easiest way to define project scope is to say that it is a blueprint where all necessary questions to be answered and problems to be solved are written out as well as a plan of action. Well-built websites are made with a lot of planning in order to function as needed, reach the right people, and accomplish all business goals.

All information, documented for easy reference to keep everyone headed in the same direction is what project scope is about. It is similar to a website business plan but with a bit more purpose to help designers get final decisions from clients and keep development work organized. With a good record of scope, time and effort can be saved and communication problems minimized, enabling designers to actually build a website.

Defining Project Scope

The most important questions that must be asked – and answered – when attempting to define project scope is to look at what the finished website will be and will use it. A good website is user-friendly to the right audience, easy for that audience to find, and provides information the targeted audience wants and needs. If these things can be stated and planned out, having good project scope is halfway defined.

Everything else that should be documented in project scope is related to the actual process of completing the job. It means knowing the client's time frame, budget and job requirements. It also involves documenting an approval process, who will provide the approvals throughout the job, and at what point in development the client wants to see progress.

Essentially, these are part of a script for both designer and client which, when recognized, prevents “scope creep” – when clients stray from the documented plan and ask for changes or other things that were never written into the agreed-upon plan or included in a price quote. The main reason why designers struggle with both time frame and clear communication is scope creep. The rules **must** be agreed upon and signed; changes and additions can come later, after the site is finished according to the planned project. Anything other than that would be an ‘upgrade’ to the project and require cost redefinition.

Though it may seem like a minor thing, professional designers strongly suggest the use of a project scope that both parties must agree with and sign to acknowledge such an agreement. Then everyone knows what to expect and what should be delivered – and communication stays much easier. By carefully explaining the purpose of having such a plan of action – that it is a tool to keep projects on schedule and designed as requested – website clients can take an active part in website design without overstepping the project's plan and designer with changing information every day – causing delays and cost overrides!

2.2: Scope definition is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

2.3: Sitemap and wireframe

Sitemaps

Once we have determined the goals and the scope of a project, we need to create a sitemap and a wireframe. If we simply start on a project with only a general idea of the project's direction it will complicate the process. The sitemap and wireframe will give you a guide to your website's organization and a targeted outline for creating content.

A sitemap for a website acts like a blueprint, listing all its pages, images, and videos. Not only does a sitemap help organize the website, but it also benefits your search engine optimization (SEO) efforts.

That is because web crawlers use sitemaps to discover all pages on the site, speeding up the indexing process and consequently boosting the site's visibility in search results.

If you're not sure where to start, we will guide you through the entire process of making a sitemap and submitting it to major search engines.

What Is a Sitemap?

A sitemap is a file that contains the website's structure, it provides:

- provides a diagram of the site's hierarchy and shows where each page fits into this hierarchy
- helps the creators and the client understand the connection between web pages, identifying the most important pages

What Does a Successful Sitemap Accomplish?

A website's sitemap will provide a picture for both the web development team and the website owner:

- The page hierarchy is defined, which shows which pages are the most important and where other pages and content need to live.
- A sitemap improves the user experience, as it is created with the specific site's goals and objectives in mind.
- All the pages that need to be created are clearly shown for both the developer and the website owner to see.
- A sitemap gives a visual representation of all the pages that need to be created.

Visual Sitemap

When making a website from scratch, web designers use visual sitemaps to show the proposed page hierarchy and navigation flow. This process helps them deliver better user experience (UX) design to their client.

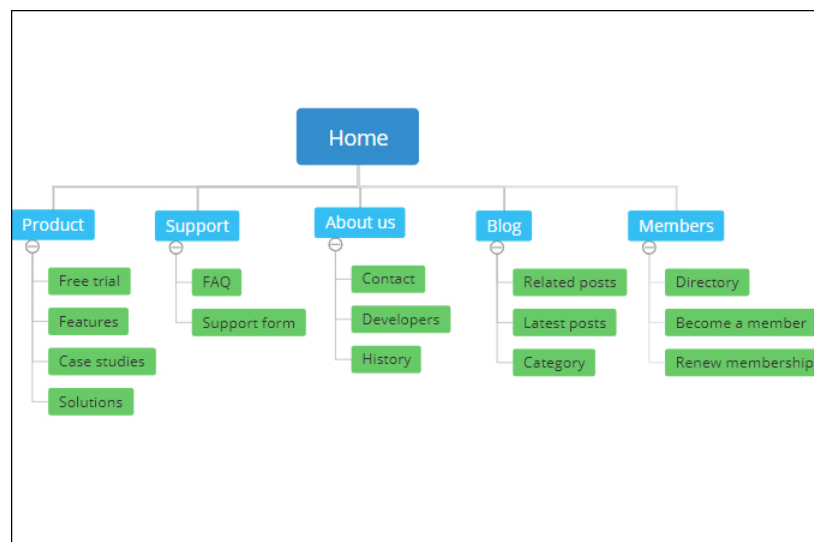


Figure 2.3.1: Example of a Website Sitemap page.

Creating a sitemap

The design of a sitemap can be approached in several different ways. Whatever method is chosen should be based on two qualities:

1. Ability to capture the information you need to capture
2. Ability to convey the information you need to convey

In other words, it needs to be capable of conveying the info you want to convey in a manner that's clear to your audience.

The horizontal diagram more clearly indicates paths between pages/experiences, as can be seen in Figure 2.3.1 above.

The Advantages of a Web Sitemap

Now that we have an understanding of what a sitemap is for, what are the benefits we expect to see from creating the sitemap?

- Improve the website's search engine optimization
- Fewer mistakes are made with the structure of the website
- Create navigation that is easy for users to follow
- Confidence that the correct web pages are being created
- Clarity for the team on the scope of the site
- Builds a roadmap to show what new pages might be added in the future

Wireframe

Imagine you're building a house. Do you go straight to constructing the building without any preparation? No! You start by going to the drawing boards and designing a house plan. By creating a layout that defines the construction of the building.

The same can be said about a wireframe. It helps keep you grounded and craft your layouts and design according to a properly laid out plan.

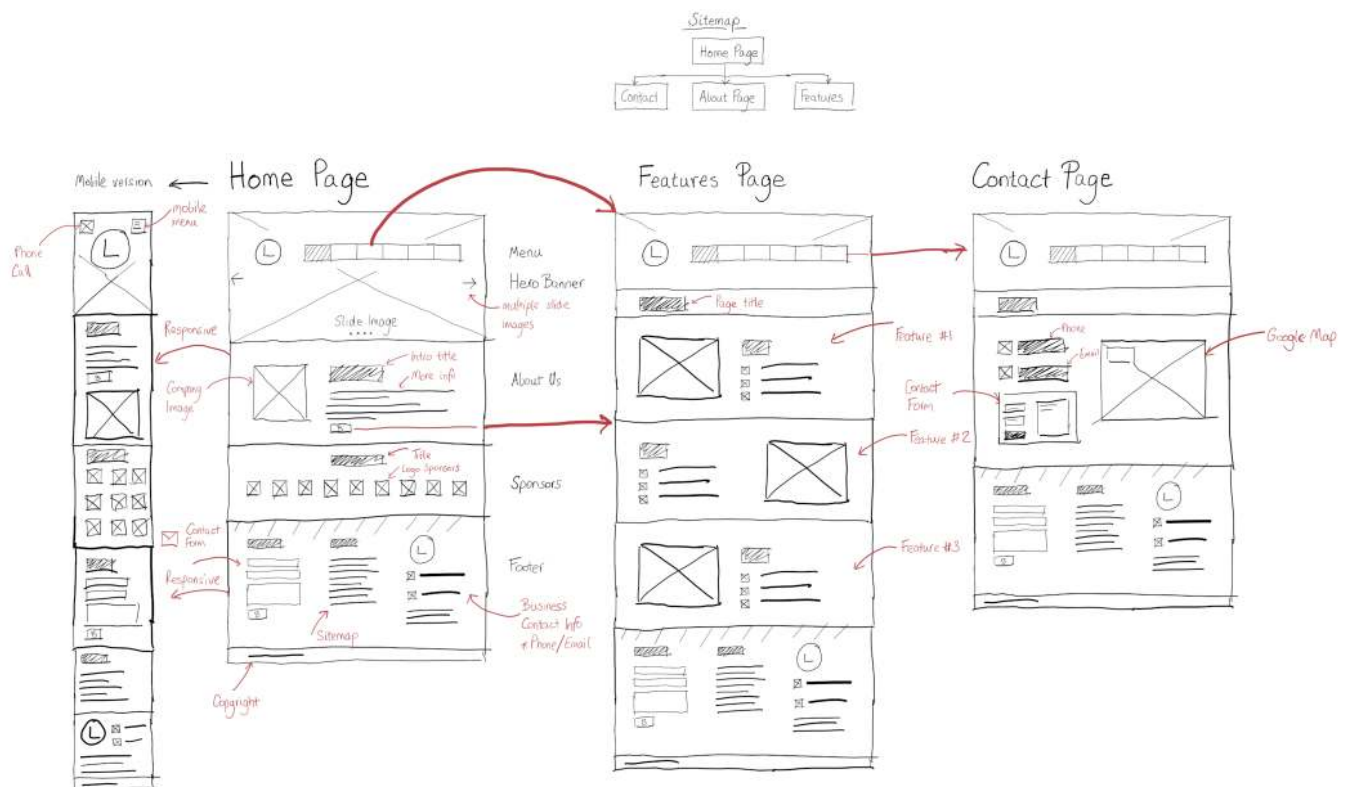


Figure 2.3.1: Example of a hand-drawn wireframe. (Copyright; author via source)

There are many different types of wireframe. In user interface design, wireframe are used to create layouts for websites, web apps, mobile apps, and more. Wireframe are even used in print designs such as posters and infographics to help improve the user

experience.

Why Wireframe Are Important

When it comes to user interface design, most designers often neglect the importance of wire framing. They go straight to the building process because they're already familiar with the usual structure of a design.

When you've seen enough website designs and apps to know where the header, menu, and buttons are, you can go straight to designing without a wireframe. And that's what gives birth to disastrous websites and apps that generate very low user interactions by ruining the user experience.

Wireframes help prevent this by allowing you to understand the outline or the frame of the design beforehand.

There is [a really good article](#) that discusses many thoughts about wireframes....you really should read this article.

2.3: Sitemap and wireframe is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

2.4: Content Creation

Content creation is the foundation of all modern digital marketing efforts. It's defined as the process of conducting research, generating strategic ideas, crafting those ideas into high-value collateral and then promoting those pieces to a target audience. Digital content takes many forms, including web pages, blogs, infographics, videos and social media posts. Content creation is the single most time-consuming responsibility for today's content marketers. When you break down our day-to-day workflow, creating great content is our chief responsibility, and the best way for us to engage with our audience and influence revenue. Not to mention that's how we earn our keep.

The main objective of content creation is to attract new and returning customers. Inbound marketing strategies give away high-value digital media for free. This content establishes your brand's thought leadership while providing readers with the information they need to make a purchasing decision.

Brands use different types of content at each stage of the marketing funnel. At the top of the funnel, search engine optimized (SEO) blogs organically attract new visitors. Mid-funnel landing pages keep readers engaged, and highly detailed content like white papers and eBooks drive conversions.

What is content creation?

Content creation is the process of identifying a new topic you want to write about, deciding which form you want the content to take, formalizing your strategy (keyword or otherwise), and then actually producing it.

In addition, most content creation processes involve thorough rounds of edits with other stakeholders before content is ready for publish.

Because content can take many forms—blog posts, videos, eBooks, Tweets, infographics, and advertisements to name a few—the content creation process is nuanced and not always as simple as it might seem. But doing it well can truly impact your business. In fact, recent research proves that creating quality educational content makes customers 131% more likely to buy from your business.

Creating great content starts with a well-established process. We'll walk you through the content creation process from start to finish, and demonstrate how creating great content can help your audiences and customers find solutions and answers to their problems. So where do we start?

What is a content creator?

Before we dive into the content creation process, let's start with the basics, like the content creator definition. **A content creator refers to someone who is responsible for the ideation, creation, and distribution of content that connects a brand to its target audience.** The goal of content creators is to create appealing and engaging content that captures the attention of users to drive website traffic, conversions, and interactions with your brand on external platforms like social. It can be informational content, but it doesn't have to be. It can also serve to entertain in order to increase brand awareness, for example.

Digital content creators produce content across any platform or channel. Having dedicated content creators is a must for any enterprise organization. These are the content marketers that will help bring ideas to life through quality content that provides the highest possibility of ranking in SERP and increasing the number of visitors to your site.

Why is Great Content So Important?

Millions of pieces of content are published online every single day. To stand out and make an impact on your customers (and potential customers), your content needs to be of a high quality. That means it's not only written and designed well, but also that it's developed with the needs of your audience in mind. The difference between high-quality and poor-quality content is stark — great content rises to the surface while poor content disappears into the void of the internet.

The amount of effort you put into your brand's content is directly proportional to the value you get from it. High-quality content will help you:

- Stand out as an authoritative thought leader.
- Grow your subscribers and email list.
- Increase your website traffic.

- Keep your customers engaged.
- Generate revenue.

2.4: Content Creation is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

2.5: Visual elements

Finally, it's time to create the visual style for the site. This part of the design process will often be shaped by existing branding elements, colour choices, and logos, as stipulated by the client. But it's also the stage of the web design process where a good web designer can really shine. Images are taking on a more significant role in web design now than ever before. Not only do high-quality images give a website a professional look and feel, but they also communicate a message, are mobile-friendly, and help build trust.

The modern web design process

Visual content is known to increase clicks, engagement, and revenue. But more than that, people want to see images on a website. Not only do images make a page feel less cumbersome and easier to digest, but they also enhance the message in the text, and can even convey vital messages without people even needing to read.

Most designers recommend using a professional photographer to get the images right. Just keep in mind that massive, beautiful images can seriously slow down a site. It is best to use some method to compress images without losing quality, saving on page-load times. You'll also want to make sure your images are as responsive as your site.

The visual design is a way to communicate and appeal to the site's users. Get it right, and it can determine the site's success. Get it wrong, and you're just another web address.

There is an awesome article titled, "[20 Web Design Principles to Follow](#)" that you should read if you are serious about creating web sites. It covers a wide variety of topics about many different aspects of a website.

2.5: Visual elements is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

2.6: Testing

2.6: Testing is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

2.7: Web Site Launch

2.7: Web Site Launch is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

3: Basic Document Layout

[3.1: Structuring Page Content](#)

[3.2: Document Structure Tags](#)

[3.3: Paragraphs](#)

[3.4: Line Breaks](#)

[3.5: Headings](#)

[3.6: Horizontal Rules](#)

[3.7: List Structures](#)

[3.8: Linking Pages](#)

[3.9: Displaying Pictures](#)

[3: Basic Document Layout](#) is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

3.1: Structuring Page Content

In this chapter we will look at some of the HTML tags that help define the **template** of a page. The template is an HTML structure that defines how your site looks across many pages. Each page in a website will be different but each page should follow a **common brand** as the other pages in the website. By defining and committing to a brand, users will be able to distinguish your website from other websites. If a page in your website does not follow the common brand users may feel inclined to leave your website. Think about it. If you were reading your local news site and you go to a page that doesn't look like it belongs to them, what do you do? Most would probably close the page and maybe leave your website altogether.

HTML describes and structures, CSS styles

The primary purpose of HTML tags is to structure and describe the text and graphic content of a Web page. The function of HTML tags is not styling and decorating content, but rather presenting content in such a logical and organized manner that allows the browser to understand and to recognize the information. Browsers render many HTML elements with a default style that proves useful in arranging content on a page; however, these default styles may be changed by using CSS.

In addition to structuring the content of a page, visual styles may also be applied. Different sizes, colors, or text fonts are some of the visual changes that can be made. Graphic images can also be resized and decorated. These tutorials will introduce styling methods that add such kinds of visual appeal to text and graphic elements. Styling elements is primarily done through CSS instead of through HTML. Before covering CSS and advanced styling, it is necessary to become familiar with basic HTML tags that can describe content and be used by browsers to provide some degree of arrangement and style to a page.

Editor versus Browser Displays

Web page design begins with a standard document setup that gives the page its overall structure. The image below reproduces the standard page template.

The ideal gas law is easy to remember and apply in solving problems, as long as you get the **proper values a**

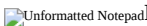
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title> page title goes here </title>
  </head>
  <body>
    *
    page content goes here
    *
  </body>
</html>
```

Listing 3.1.1 A web page template

Every page designed begins with the coding represented in the image above. The title of the page must be coded within the `<title>` tag. When the page is opened, the text will appear in the browser's "Title Bar." Informative content should be added

inside the `<body>` tag. Only the information appearing in the body of the document shows up in the browser window. Text and graphic images are placed inside the `<body>` tag, and are then surrounded by HTML tags to order and arrange them.

It is important to understand that formatting Web pages can not take place without the usage of HTML tags. To emphasize this point consider the heading and three paragraphs coded in the Notepad editor in Figure 2-1. The information in the browser should be displayed in this same general format.

 Figure 3.1.1: Unformatted web document appearing in the Notepad editor ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials. 4](#), [GALILEO Open Learning Materials](#) is licensed under [CC BY-SA 4.0](#))

As shown above, a heading appears at the top of the document, and all subsequent paragraphs are separated from each other by single blank lines. When entering information into the editor, care must always be taken to ensure that the information entered will appear in the browser in an orderly and readable fashion. Editor formatting has little to do with how the entered information appears as a Web page, but it is still a valuable practice to develop to avoid such careless mistakes as interchanging words like deprecating and defecating.

When the document exemplified above is saved as an HTML file (.htm or .html) and opened in the browser, it is displayed as shown in the below Figure 2-2.

 Unformatted Browser View

Figure 3.1.2: Unformatted Web page appearing in the browser. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials. 4](#), [GALILEO Open Learning Materials](#) is licensed under [CC BY-SA 4.0](#))

The appearance of the information is very different from how it is displayed in the editor. Of course, the reason for this is that no HTML tags appear in the document to guide the browser in laying out the text on the page. Therefore, the browser treats the information as one continuous string of text. All blank spaces and line breaks appearing in the editor are collapsed to single blank spaces separating the words. The result is a single block of text on the page.

Formatted Text Content

It is an easy step to return to the Notepad editor and include some basic HTML tags to arrange the text for display in the browser. In this case, as shown in Figure 2-3, paragraph tags (coded as `<p>` container tags) surround the separate paragraphs and a header tag (coded as an `<h2>` container tag) surrounds the heading line.

 Formatted Notepad

Figure 3.1.2: Formatted web document appearing in the Notepad editor. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials. 4](#), [GALILEO Open Learning Materials](#) is licensed under [CC BY-SA 4.0](#))

Now, when the page is opened in a browser, these tags cause the text blocks to be displayed as separate paragraphs as shown in Figure 2-4. The formatting is far from fancy, but it does illustrate the point that the layout of page elements is exclusively controlled by the HTML tags that appear in the document.

 Formatted Browser View

Figure 3.1.3: Formatted Web page appearing in the browser. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials. 4](#), [GALILEO Open Learning Materials](#) is licensed under [CC BY-SA 4.0](#))

The next several topics introduce a basic set of HTML tags that bring order and structural formatting to page content. The initial focus is on text arrangement. Later topics will introduce basic tags for adding graphic images and links. For the time being, learn the structural aspects of page design, and begin becoming comfortable with the styling techniques introduced in the following tutorial.

3.1: Structuring Page Content is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

3.2: Document Structure Tags

HTML 5 includes a new set of semantic document structural tags to define different parts or structure of a web page. A semantic element clearly describes its meaning to both the browser and the web developer.

 Simple Structure of a Page

Figure 3.2.1: A simple template "MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials](#), 4, [GALILEO Open Learning Materials](#) is licensed under [CC BY-SA 4.0](#)

Some of the most common HTML 5 structural elements are described below.

The `<header>` tag

The `<header>` element is used as a container for page heading content. The tag typically contains heading tags (`<h1>` - `<h6>`), a logo image, or other content that introduces the content. It is possible to have more than one `<header>` tag per page; however, the `<header>` cannot be coded within a `<footer>` or another `<header>` tag.

The `<nav>` tag

The `<nav>` tag is a block level element used to define major blocks of navigation links or navigation menus.

The `<main>` tag

The `<main>` tag is used to serve a container for major page content that is not repeated in other parts of the page. A web page can include only one `<main>` element. Additionally, it cannot be included as a child element within a `<header>` , `<nav>` , `<footer>` , `<article>` , or `<aside>` tag.

The `<section>` tag

The `<section>` tag defines generic sections in a web page such as headers, footers, or any other sections of the document. In this context, a section is a thematic grouping of content.

The `<article>` tag

The `<article>` element represents a complete, or self-contained, composition in a document, page, application, or site that is, in principle, independently distributable or reusable.

The `<aside>` tag

The `<aside>` tag is a block level element that defines content aside from the content it is placed in. The aside content should be related to the surrounding content.

The `<footer>` tag

The `<footer>` tag is a block level element that defines footer information for an entire web page or section of the document. The content of the `<footer>` typically consists of contact information, copyright, links, or logos. In some cases, a web page may contain multiple `<footer>` tags.

The `<figure>` tag

The `<figure>` tag specifies self-contained content, like illustrations, diagrams, photos, or code sections. By default, this is a block level element with 40px right and left margin settings.

The `<figcaption>` tag

The `<figcaption>` is a block level tag that defines a caption for a `<figure>` element discussed in the previous section. The `<figcaption>` tag is normally coded as the first or last child element of the `<figure>` tag.

3.2: Document Structure Tags is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

3.3: Paragraphs

The `<p>` Tag

Most Web pages are text documents. In written languages such as English, letters are grouped into words, words are organized into sentences, and sentences are structured into cohesive paragraphs. HTML provides a way for content to be marked as a paragraph. By default, browsers render text marked as a paragraph with a single blank line preceding and following the text. Text inside a paragraph is also aligned to the left margin of the page and is word-wrapped inside the browser window.

In order to format a block of text as a paragraph, the `<p>` (paragraph) tag surrounds the enclosed text. The general format for this tag is given shown below.

```
<p> Text </p>
```

Any amount of text can appear between the opening and closing tags. When displayed in the browser, the enclosed text is single spaced and preceded and followed by a single blank line to separate it from surrounding page content. The `<p>` tag is classified as a container and block-level element.

You can see the effects of `<p>` tags in the Web page shown above which displays text as three separate paragraphs. These tags are added in the editor as shown below.

```
<p>
  A Web page begins with preliminary sections of code th
  HTML5 document. Also included is a title which appears
  Title Bar.
</p>

<p>
  All content that appears in the browser window is code
  of the page. Text information, graphic images, links,
  are added to the page and are surrounded by HTML tags
  them for presentation.
</p>

<p>
  It is important to note that content is arranged on th
  HTML tags coded in the document. Otherwise, all conten
  unformatted string of text.
</p>
```

The text is now easily readable because organization is brought to its information content. Note that back-to-back paragraphs are separated by a single blank line even though each `<p>` tag itself produces a blank line before and after a paragraph. The browser collapses into a single blank line: the one that follows a paragraph and the one that precede the next.

Note that on this and subsequent pages complete coding for a Web page may not be shown in order to focus on the code being discussed. Still, it is assumed that the code shown is surrounded by the page preliminaries down through the `<body>` tag and by the closing `</body>` and `</html>` tags.)

The `<blockquote>` Tag

The `<blockquote>` tag offers a way to indent a block of text, to offset it from surrounding content for special emphasis as would be done for a quoted citation from which the tag gets its name. This container tag indents its enclosed text a fixed number of pixels (approximately 40 pixels) from the left and right margins. Its enclosed text is word-wrapped and preceded and followed by a blank line. The general format for the `<blockquote>` tag is shown below.

```
<blockquote>
  text
</blockquote>
```

According to HTML 5 standards, the `<blockquote>` is used to mark content that is quoted from another source. In practice, the `<blockquote>` should be used only in the correct context to mark a block quotation. For our purposes, it currently serves as a convenient way to indent any block of text.

In the following example, a `<blockquote>` tag offsets the middle paragraph from surrounding text. Browser output is shown in Figure 3.3.1.

```
<p>
  Here are three blocks of text. This first paragraph is
  standard paragraph tag and is blocked at the left marg
</p>

<blockquote>
  This second block of text is formatted with a blockqu
  left and right margins approximately 40 pixels from th
</blockquote>

<p>
  This final block of text is coded like the first one.
  standard paragraph tag and is blocked at the left marg
</p>
```

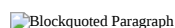
 Blockquoted Paragraph

Figure 3.3.1: Applying the `<blockquote>` tag to a paragraph. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials. 4](#), GALILEO Open Learning Materials is licensed under [CC BY-SA 4.0](#))

Similar to the way it appears in the browser, the above editor coding shows the blockquoted text offset with blank lines and indented. As you know, this editor formatting is not required and has no effect on browser output that is controlled solely by HTML tags. Nevertheless, it is good practice to arrange editor coding to resemble browser output the best way possible for visual indication of the display relationships among sections of page content. It makes it easier to spot missing or miscoded tags when editor coding is well spaced and indented.

Nested Blockquotes

In the following example, `<blockquote>` tags are used to offset three blocks of text. The middle one is further indented inside its outer blockquoted paragraphs. Browser output is shown in Figure 3.3.2.

```
<p>
  Here are five blocks of text. This first block is format
  paragraph tag and is blocked at the left margin of the p
</p>

<blockquote>
  This second block of text is formatted with a blockquoc
  left and right margins approximately 40 pixels from th

    <blockquote>
      This block of text is also surrounded by blockquote
      indented within the margins produced by its outer bl
    </blockquote>

    This block of text is aligned like the second block of
    inside the outermost blockquote tag.
  </blockquote>

<p>
  This fifth block of text is coded like the first one. It
  a standard paragraph tag and blocked at the left margin.
</p>
```

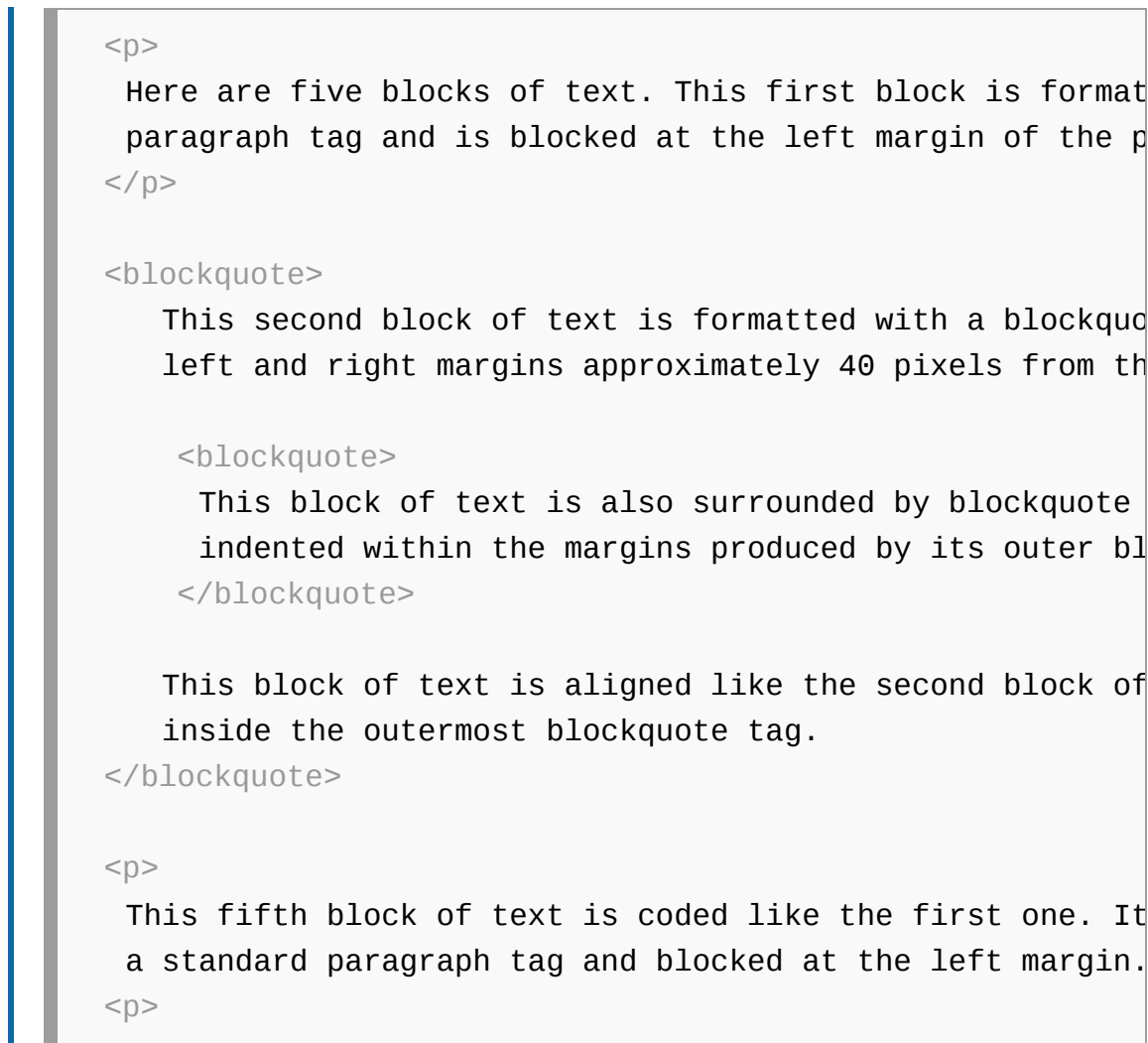


Figure 3.3.2: Text formatted with nested `<blockquote>` tags. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials. 4](#), GALILEO Open Learning Materials is licensed under [CC BY-SA 4.0](#))

In this example, the two `<blockquote>` tags are nested; that is, a `<blockquote>` tag appears inside a `<blockquote>` tag. The outer `<blockquote>` tag surrounds and indents its three contained text blocks approximately 40 pixels from the left and right margins of the page. The inner `<blockquote>` tag indents its enclosed text an additional 40

pixels from the margins established by the outer tag. Any additional nested tags further indent their enclosed text from the previous margins.

When nesting `<blockquote>` tags inside one another, make sure that opening and closing tags are properly paired, and that inner tags are fully closed inside their next outer-most enclosing tags. As in the case with `<p>` tags, back-to-back `<blockquote>` tags have their preceding and following blank lines collapsed to a single blank line.

Remember that the `<blockquote>` tag should not be used to just indent text. Instead it should only be used when displaying large blocks of quoted text within a web document. In a later tutorial, you will learn style sheet methods that can be used for indenting text.

Deprecated (OLD) Tags and Attributes

HTML 5 is a replacement for the older XHTML and HTML4 languages. In addition to adding many new elements and features, HTML 5 retains many of the tags found in previous HTML versions. Still, many older styling attributes are no longer supported officially by HTML 5. Many of these attributes may still be recognized by browsers which support compatibility with older HTML versions. For example, the `<p>` tag historically included the `align` attribute to control text alignment. For instance, by setting this attribute to `<p align="center">` the text on each line of a paragraph is centered rather than aligned at the left margin.

In HTML 5, the `align` attribute is not supported. It is recommended that you refrain from using it and other styling attributes which have been phased out. Instead, newer style sheet declarations should be used to produce the same effect. Still, deprecated attributes are described in these tutorials since you may encounter them on existing Web pages even though most attributes are not valid under HTML 5 standards.

Certain HTML tags are also deprecated under HTML 5. Tags that are being phased out are presented since you will likely encounter them in existing pages. These tags may display properly in modern browsers, but newer alternatives should be used in their place. Furthermore, HTML 5 changes the contextual meaning of some older tags. These tags should now be used in the context where they are appropriate, instead of simply being used for styling purposes.

Deprecated (OLD) align Attribute

By default, paragraph text displayed in a browser is word wrapped and blocked at the left margin of the page. It is possible, however, to block text at the right margin, to center it, or to justify all lines by expanding them to both margins. These alignment options are specified by coding the `align` attribute inside a block-level tag. The general format for coding this attribute along with its possible values are:

```
align="left|center|right|justify"
```

The keyword `align` is assigned one of the quoted alignment values. For example, the following code centers the heading of the previous example page and justifies all paragraph lines by expanding them to the right margin. Browser output is shown in Figure 3.3.3.

```
<h2 align="center"> Web Page Layout </h2>
```

```
<p align="justify">
```

A Web page begins with preliminary sections of code that HTML5 document. Also included is a title which appears in Title Bar.

```
</p>

<p align="justify">
  All content that appears in the browser window is coded
  of the page. Text information, graphic images, links, and
  are added to the page and are surrounded by HTML tags to
  them for presentation.
</p>

<p align="justify">
  It is important to note that content is arranged on the
  HTML tags coded in the document. Otherwise, all content
  unformatted string of text.
</p>
```

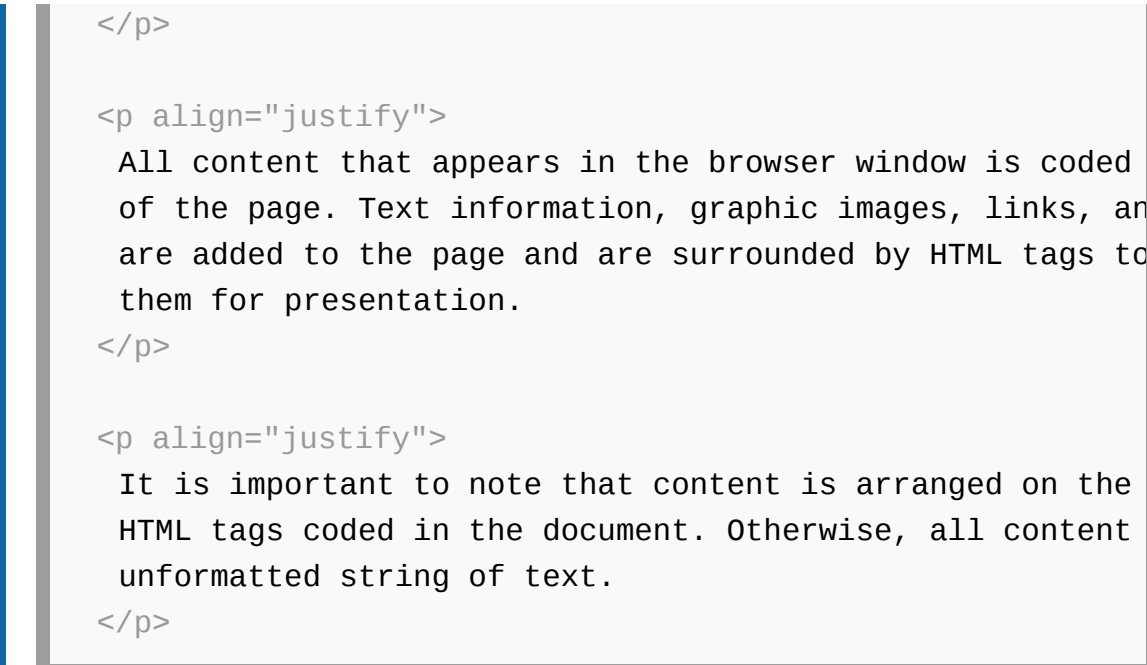
 Browser w/ aligned paragraphs

Figure 3.3.3: Web page formatted with aligned paragraphs in browser.

Be aware that the align attribute is deprecated under current HTML5 standards. It is presented here as a quick and easy way to align paragraphs and will serve as a convenient, but temporary, styling method until you learn how to code and integrate style sheets on your Web pages.

3.3: Paragraphs is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

3.4: Line Breaks

The `<br />` Tag

Coding paragraphs with `<p>` tags is the most common method of structuring text on a Web page. Most Web pages are text pages, and paragraphs are convenient and readable methods of text presentation. Other varieties of text structuring are nonetheless available.

The `<br />` (line break) tag causes the browser to end a line of text and to continue display on the next line in the browser window. It does not, as in the case of paragraphs, leave blank lines before or after the ended text line. The general format for the break tag is shown here:

```
<br />
```

General format for `<br />` tag. Note that in HTML5 the `/` is optional. `<br>` is fine but note that it does not have a closing `</br>` .

The `<br />` tag is **not a container tag**. It does not enclose text and it does not have an accompanying closing tag. HTML5 elements that do not enclose text and have no accompanying closing tag are referred to void elements. This empty tag is simply placed within the text wherever a line break should occur. The tag is handy for displaying lists of items, lines of verse, or other groupings of individual, single-spaced text lines. For example, the following code uses line breaks to end individual lines of verse packaged inside a `<blockquote>` tag.

```
<p>
  Here is a tale about Mary and a pesky little lamb that f
  anywhere and everywhere she went.
</p>

<blockquote>
  Mary had a little lamb, <br />
  Its fleece was white as snow; <br />
  And everywhere that Mary went, <br />
  The lamb was sure to go.
</blockquote>

<p>
  Mary had an awkward social life. It's awfully difficult
  trailing around after you all the time.
</p>
```

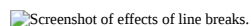
 Screenshot of effects of line breaks.

Figure 3.4.1: Using line break for single-spaced output. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials. 4](#), GALILEO Open Learning Materials is licensed under [CC BY-SA 4.0](#))

The four lines of verse are enclosed inside a blockquoted paragraph to offset and indent them from surrounding paragraphs. Each line of verse appears on a separate text line on a single space below the preceding line.

Multiple Line Breaks

With `<p>` and `<blockquote>` tags, a single blank line appears before and after the enclosed indented text block. When `<br />` tags are inserted to end lines of text, no spacing appears between the lines. This is the correct way to use the `<br />` tag. Historically, multiple `<br />` tags were often used to insert extra blank lines in order to increase vertical spacing between lines of text or between other content appearing on the page. This practice is no longer preferred under HTML 5, and should be replaced with style sheets where necessary.

Below is a recoding of the previous page to leave an additional blank line before and after the blockquoted verse. These blank lines are produced by coding `<br />` tags to force additional line breaks. Browser output is shown in Figure 3.4.2.

```
<p>
    Here is a tale about Mary and a pesky little lamb that
    anywhere and everywhere she went. <br /> <br /> <br />
</p>

<blockquote>
    Mary had a little lamb, <br />
    Its fleece was white as snow; <br />
    And everywhere that Mary went, <br />
    The lamb was sure to go.
</blockquote>

<p>
    <br />
    Mary had an awkward social life. It's awfully difficult
    trailing around after you all the time.
</p>
```

 Multiple Line Breaks

3.4.1 Using line break for single-spaced output. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials](#). 4, [GALILEO Open Learning Materials](#) is licensed under [CC BY-SA 4.0](#))

Each `<br />` tag inserts an additional line break on the page. Therefore, you can code multiple `<br />` tags back-to-back to produce multiple blank lines down the page.

Notice that it requires three `<br />` tags at the end of the first paragraph to match the number of blank lines given by the single `<br />` tag at the beginning of the last paragraph. This is a peculiarity of the way the browser renders the tag at the beginning and end of paragraphs. You may have to experiment to determine the number of tags to code to produced the desired number of blank lines. Still, it is preferable to control vertical spacing with other block level elements or style sheets.

3.4: Line Breaks is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

3.5: Headings

The Heading Element

Text appearing on a Web page is displayed in a default font style and size unless specified otherwise. Most browsers display text in the Times New Roman font face, or font family at approximately 12-point size. In order to add visual variety to your pages, you will likely want to choose other font faces and sizes for different sections of your document. This is a topic that is taken up in more detail later; however, there are simple ways to change text sizes to permit you to add various levels of headings to your pages.

The `<h1>` (heading) tag is a block-level and container tag that encloses a string of text for display in one of six heading styles. Its general format is shown in Figure 3.5.1.

```
<h1> Heading Text </h1>
```

The number *n* in the tag ranges from 1 (largest) to 6 (smallest). These six heading levels are coded in the following example and displayed in the browser as shown in Figure 3.5.1. A standard paragraph is displayed for comparison.

```
<h1> This is Heading Level 1 </h1>
<h2> This is Heading Level 2 </h2>
<h3> This is Heading Level 3 </h3>
<h4> This is Heading Level 4 </h4>
<h5> This is Heading Level 5 </h5>
<h6> This is Heading Level 6 </h6>
<p> This is a normal text paragraph. </p>
```

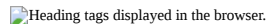
Heading tags displayed in the browser.

Figure 3.5.1: Browser display of `<h1>` tags. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials. 4](#), [GALILEO Open Learning Materials](#) is licensed under [CC BY-SA 4.0](#))

Headings are displayed in bold characters in the default font face. They appear on a line by themselves preceded and followed by a single blank line. Thus, the `<h1>` tag cannot be used to change text sizes. In the middle the tag cannot be used to change text sizes in the middle of a line or middle of a paragraph since it forces a line break before and after its enclosed text. It is used for block-level, stand-alone text, and is most often used as content headings to identify and set apart sections of a Web page.

HTML 5 Heading Semantics

While heading tags do provide a convenient way to style text, they are designed to mark the heading for a section of content on the page. Headings with lower numbers are considered higher in rank, and content on a page should be structured with the appropriate header ranking. This listing illustrates this use of headings to section content.

```
<h1> North America </h1>
<p> North America is a large continent which contains many countries </p>
<h2> The United States of America </h2>
<p> The United States of America is a country located in North America </p>
<h3> The State of Georgia </h3>
<p> Georgia is one of fifty states found in The United States of America </p>
```

headings and paragraphs displayed in browser.

Figure 3.5.2: Heading and paragraphs displayed in browser. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials. 4](#), [GALILEO Open Learning Materials](#) is licensed under [CC BY-SA 4.0](#))

The main section, "North America", is marked with an `<h1>` tag to indicate that it is the start of a major section on the page. Next, "The United States of America" is marked with an `<h2>` tag to indicate that it begins a subsection of "North America". Finally, "The State of Georgia" is marked with an `<h3>` tag to indicate that it begins a subsection of "The United States of America". In this way, tags can be used to section and subsection content on a page.

Deprecated align Attribute

Headings can be aligned to the left (default) or right, or they can be centered on the line by applying the

```
| align="left|center|right"
```

attribute to the `<hn>` tag. The following code, for example, centers a heading horizontally between page margins.

```
| <h1 align="center">Centered Heading></h1>
```

Since align is a deprecated attribute under HTML 5 standards, it should be used only as a temporary method to align headings. Later tutorials cover style sheet standards for aligning text horizontally on a page.

3.5: Headings is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

3.6: Horizontal Rules

The Horizontal Rule Element

The primary way to separate and identify sections of a Web page is by use of major and minor headings coded within tags. Paragraph level content within these sections can also be visually separated by displaying horizontal rules between them. In addition to its visual properties, HTML 5 prescribes the horizontal rule for use when a thematic break among such content is required. The general format for this tag is shown in Figure 2-15.

```
<hr />
```

Note that in HTML5 the / is optional. <hr> is fine but note that it does not have a closing </hr>

The `<hr />` tag causes a line break with the rule appearing on a line by itself. The default rule is 2 pixels in height and spans the width of the browser window. The rule shown in Figure [Runaway argument for 'eatSpaces?'](#) is produced by the `<hr />` tag coded on a line by itself is:

```
<h2> Horizontal Rules </h2>
```

```
<p> A horizontal rule is used to separate content sections of a page. The following  
is a default rule. </p>
```

```
<hr />
```

```
<p> The rule is 2 pixels in height and spans the width of the browser window. It is  
preceded and followed by blank lines. </p>
```

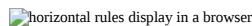
 horizontal rules display in a browser

Figure [Runaway argument for 'eatSpaces?'](#): Default horizontal rule. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials. 4](#), GALILEO Open Learning Materials is licensed under [CC BY-SA 4.0](#))

In a later tutorial you learn how to style a horizontal rule by adjusting its height, width, color, alignment, and display characteristics. For present purposes, the default rule can be effective in visually breaking up text without starting a new section.

Deprecated `<hr />` Attributes

Historically, the `<hr />` tag included the following attributes to style the rule. These attributes are deprecated, but many browsers still recognize them.

```
align="left|center|right"
size="n"
width="n|n%"
noshade
color="color"
```

The align attribute positions the rule to the left (default), centered, or to the right across the line. The size attribute sets the height of the horizontal rule in pixels. The width attribute sets the rule's width in pixels or as a percentage of the window width. Rules can be

displayed as a solid bar by coding noshade (without an associated value) and assigned a color with the color attribute. The color value is given as a color name or hexadecimal value. We will discuss such assignments in a later tutorial.

The following rule is given by the code:

```
<hr align="center" size="5" width="50%" noshade color="red">
```

Since style attributes within the tag have been deprecated under current standards, they should not be used in HTML5 documents. Instead, they should be avoided in favor of the newer styling methods that will be discussed later.

[3.6: Horizontal Rules](#) is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

3.7: List Structures

Listing Elements

HTML 5 provides text structuring in the form of lists. You can produce lists of bulleted items, lists of numbered items, and lists of terms and definitions. The first two lists resemble single-spaced lines of text with the addition of prefixed bullets or numbers. The latter list is similar in display to a series of blockquoted paragraphs.

Unordered Lists

An **unordered list** is a series of items preceded by bullet characters and set off from surrounding text by blank lines. The list is single spaced and indented from the left margin. Unordered lists should be used when the order of each item in the list is not important. An example unordered list:

- List Item #1
- List Item #2
- List Item #3

An unordered list is created with the block-level `<ul>` container tag enclosing listed items identified with block-level `<li>` `<ul>` container tag enclosing listed items identified with block-level `<li>` (list item) container tags. The general format for an unordered list is:

```
<ul>
  <li> list item </li>
  <li> list item </li>
  ...
</ul>
```

For example, the bulleted list shown above is given by the following code.

```
<ul>
  <li> List Item #1 </li>
  <li> List Item #2 </li>
  <li> List Item #3 </li>
</ul>
```

Items in the list are single spaced and preceded by a bullet character. If text for a list item is wider than the width of the page it is word wrapped and indented inside the bullet character. Items can be enclosed inside `<p>` tags or `<br />` tags can be coded between items to increase line spacing between them. The following list, for example, surrounds list items with `<p>` tags to leave blank lines between entries.

```
<ul>
  <li><p> This is the first item in the list. Text follow
character is word wrapped inside the bullet. Paragraph t
leave blank lines between items in the list. </p></li>

  <li><p> This is the second item in the list. Text follo
```

```
character is word wrapped inside the bullet. Paragraph t  
leave blank lines between items in the list. </p></li>  
</ul>
```

Browser output for the above html is:

- *This is the first item in the list. Text following the bullet character is word wrapped inside the bullet. Paragraph tags are used to leave blank lines between items in the list.*
- *This is the second item in the list. Text following the bullet character is word wrapped inside the bullet. Paragraph tags are used to leave blank lines between items in the list.*

Nested Unordered Lists

Unordered lists can be nested inside each other. For example, a bulleted list appearing inside a second bulleted list appearing inside a third bulleted list is produced by the following code and is displayed in the browser as shown after the html code below. Note that a nested unordered list is part of the list items of its container list. That is, the `<ul> ... </ul>` tags for the nested list appear inside a pair of `<li> ... </li>` tags of `<ul> ... </ul>` tags for the nested list appear inside a pair of `<li> ... </li>` tags of the enclosing list.

```
<ul>  
  <li> List Item 1 </li>  
  <li> List Item 2  
    <ul>  
      <li> List Item 2a </li>  
      <li> List Item 2b  
        <ul>  
          <li> ListItem 2b1 </li>  
          <li> List Item 2b2 </li>  
        </ul>  
      </li>  
    </ul>  
  </li>  
  <li> List Item 3 </li>  
</ul>
```

- *List Item 1*
- *List Item 2*
 - *List Item 2a*

- *List Item 2b*
 - *ListItem 2b1*
 - *List Item 2b2*
- *List Item 3*

Each nested list is further indented inside its container list. Also, different bullet characters are used for nested lists. By default, a disc character marks the outer-most list, a circle marks the next inner-most list, and a square marks the inner list. Notice that when lists are contained inside other lists that no blank lines surround the interior lists as they do when a single list appears within the normal flow of text on the page.

Deprecated type Attribute

The `type="disc|circle|square"` attribute can be coded inside the opening `<ul>` tag in order to specify the style of bullet character to use if different from the default disc character. HTML5 standards do not allow the use of the `type` attribute and suggest using CSS for specifying bullet characters.

Ordered Lists

An ordered list is a series of items preceded by sequence numbers and set off from surrounding text by single blank lines. By default, the list is numbered with decimal numbers beginning with 1 and numbered consecutively through the last item in the list. The list is single spaced and indented from the left margin in the same way as an unordered list. Ordered lists should be used when the order of each list item is important. An example ordered list is shown below.

1. *List Item #1*
2. *List Item #2*
3. *List Item #3*

The html to create that ordered list is here:

```
<ol>
  <li> List Item #1 </li>
  <li> List Item #2 </li>
  <li> List Item #3 </li>
</ol>
```

Items in the list are single spaced and word-wrapped inside the numbering character. `<li>` can contain other tags like `<p>` , `<blockquote>` , `<h1>` , and other tags. It is important to remember that `<ol><li>` and `<ul><li>` is a structure

Nested Ordered Lists

Ordered lists can be nested inside each other, with subordinate lists indented from the next outer-most list. All nested ordered lists use the same decimal numbering system beginning with decimal 1. For example, the following code produces the numbered lists.

```
<ol>
  <li> List Item 1 </li>
  <li> List Item 2
```

```
<ol>
  <li> List Item 2.1 </li>
  <li> List Item 2.1 </li>
</ol>
</li>
<li> List Item 3 </li>
</ol>
```

1. *List Item 1*
2. *List Item 2*
 1. *List Item 2.1*
 2. *List Item 2.2*
3. *List Item 3*

Note that when ordered lists are contained inside other ordered lists that no blank lines surround the interior lists as they do when a list appears within the normal flow of page text.

Ordered List Attributes

A **type** attribute can be coded inside the opening `<ol>` tag in order to specify one of five different numbering characters. The attribute value can be `<ol type='1'>` for decimal numerals (the default), `<ol type='A'>` for upper-case letters, `<ol type='a'>` for lower-case letters, `<ol type='I'>` for upper-case Roman numerals, and `<ol type='i'>` for lower-case Roman numerals. The tag `<ol type='A'>`, for example, produces the following list of alphabetically ordered items.

```
<ol type='A'>
  <li> List Item 1 </li>
  <li> List Item 2 </li>
  <li> List Item 3 </li>
</ol>
```

- A. *List Item 1*
- B. *List Item 2*
- C. *List Item 3*

You can choose the beginning sequence value of an ordered list by coding the optional `<ol start='n'>` attribute for the `<ol>` tag. Here, *n* must be a valid integer. For instance, a starting sequence value is needed, when an ordered list is interrupted by other elements on the page.

As shown in the output below, two consecutively sequenced lists are separated by a paragraph. The first list begins with "A" and is sequenced through "E" since there are five items in the list. The second list needs to begin its sequence with the sixth letter "F". If a starting value is not specified, sequencing begins anew with "A".

This is the beginning of the list:

- A. List Item A*
- B. List Item B*
- C. List Item C*
- D. List Item D*
- E. List Item E*

This is a continuation of the list:

- F. List Item F*
- G. List Item G*
- H. List Item H*
- I. List Item I*
- J. List Item J*

Code for the above lists is shown below. The first ordered list uses upper-case letters (type="A") beginning with "A" and ending with "E". The second list overrides this default sequencing by specifying start="6" in its opening `<ol>` tag. Thus, the second list is (type="A") beginning with "A" and ending with "E". The second list overrides this default sequencing by specifying start="6" in its opening `<ol>` tag. Thus, the second list is ordered consecutively from "F" through "J".

```
<p> This is the beginning of the list. </p>
```

```
<ol type='A'>
```

```
<li> List Item A </li>
```

```
<li> List Item B </li>
```

```
<li> List Item C </li>
```

```
<li> List Item D </li>
```

```
<li> List Item E </li>
```

```
</ol> >
```

```
<p> This is a continuation of the list. </p>
```

```
<ol type='A' start='5'>
```

```
<li> List Item F </li>
```

```
<li> List Item G </li>
```

```
<li> List Item H </li>
```

```
<li> List Item I </li>
```

```
<li> List Item J </li>
```

```
</ol> >
```

You can reverse the order of the sequence of list markers through the use of the reversed attribute. This is a boolean attribute that can be applied to an `<ol>` tag in multiple ways. For instance, you could reverse the order of markers through adding the attribute itself without a value, such as in `<ol reversed>`. Optionally, you could also add a value to the attribute, and set it

equal to either "reversed" or a blank string "". For example, `<ol reversed='reversed'>` and `<ol reversed=''>` also accomplishes the same effect. This is true of other boolean attributes you may come across in future elements. All of these forms are valid under HTML5. Below is an example of ordered list with the attribute reversed.

3. List Item 1
2. List Item 2
1. List Item 3

Definition Lists

A **definition list** is a series of terms and definitions offset from surrounding text by blank lines. The terms in the list are blocked at the left margin, and definitions are indented and word wrapped on the following lines. An example definition list is shown below with the html to create the list.

Term 1

This is the Term 1 definition. The definition term appears on a line by itself and is followed by a definition text block. The definition is indented and word wrapped.

Term 2

This is the Term 2 definition. The definition term appears on a line by itself and is followed by a definition text block. The definition is indented and word wrapped.

A definition list is enclosed inside `<dl>` tags and contains one or more `<dt>` (definition term) tags listing the items to be defined. Each definition term has an associated `<dd>` (definition description) tag enclosing the definition for the term.

The definition list shown above is produced with the following code.

```
<dl>
  <dt> Term 1 </dt>
  <dd> This is the Term 1 definition. The definitio
    a line by itself and is followed by a definition t
    The definition is indented and word wrapped. </dd>
  <dt> Term 2 </dt>
  <dd> This is the Term 2 definition. The definitio
    a line by itself and is followed by a definition t
    The definition is indented and word wrapped. </dd>
</dl>
```

When displayed in the browser, items in the list are single spaced with no blank lines appearing between the terms. If you wish to include additional line spacing, you can code `<p>` tags surrounding the definitions or `<br />` tags between them.

A definition list, of course, can be used for purposes other than defining terms. Any situation where you have a name and value relationship among content you can employ a definition list. Another example where a definition list might be appropriate is for a frequently asked questions, or FAQ, section of a web site.

3.7: List Structures is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

3.8: Linking Pages

Hyperlinks

The original purpose of the World Wide Web was to provide a mechanism for non-linear access to information. In order to activate this type of retrieval, information is packaged as a set of Web pages, and then **hypertext links**, or **hyperlinks** are created to navigate from one page to another in whatever sequence the reader prefers. In a later tutorial, the varieties of page links are discussed more fully. Here you are introduced to the basic concept of linking and shown how to set up text links for navigation between pages.

Most often a link from one page to another is made by creating a text "hot spot" on one page that, when clicked with the mouse, opens a second page which replaces the first page in the browser window. The string of text that is made sensitive to a mouse click can be a single word or several words, usually suggestive of the contents of the destination page. Normally, a second link appears on the linked page for return to the original page. It is considered poor Web citizenship to leave visitors at the end of a series of links with no way to get back to the starting page.

In the illustration in Figure 2-26 two text links cause navigation to different pages. On the linked pages a text link provides for return to the original page.

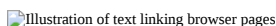
 Illustration of text linking browser pages

Figure 3.8.1: Text links between pages. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials](#), 4, [GALILEO Open Learning Materials](#) is licensed under [CC BY-SA 4.0](#))

You can test these links by clicking the "Linking Web Pages" text link below. The original page is opened in a separate browser window. From there, you can link to and return from two different pages.

[Linking Web Pages](#)

The <a> Tag

The most common type of link is a clickable word or phrase that transfers directly to the target page. A text link of this sort is created by surrounding the text string with an `<a>` tag (also called an **anchor** tag), and specifying the location of the page to which to link. The `<a>` tag is classified as a container and inline-level tag. The basic format for coding this tag is:

```
<a href="url">linking text</a>
```

The **href** (hypertext reference) attribute specifies the URL of the linked page. Links can be made to other pages at the local Web site or to pages at remote sites anywhere on the World Wide Web.

If the linked page is in the same location – in the same directory – as the linking page, then the URL is simply the page name. This local link is called a *relative* link since the path to the linked page is relative to the location of the linking page.

When linking to a remote site, the URL must be a full Web reference including the `"http://"` protocol followed by the domain name of the site. This external link is called an **absolute** link since the path to the linked page is a full Web address.

The `<a>` tag surrounds a string of text that is to be made click-sensitive and that activates the link. Usually the text is descriptive of the destination link. It can be a single letter, a word, a phrase, or any amount of text that can reasonably serve as a link. When this text is clicked, the linking page is immediately replaced by the linked page in the browser window.

The `<a>` tag is an in-line tag. It does not add any new lines or spaces before or after the content which it wraps.

Local Links

When creating your own Web site, most of the links will be between your own pages. The easiest way to store these pages for linking is to save them in the same location. That is, in the same disk directory or on the same diskette or storage unit. In this case the URL specified in the `<a>` tag is simply the name of the page.

In the example in Figure 2-26 there are three Web pages named `Home.htm` , `Organize.htm` , and `Supplement.htm` . The `Home.htm` page contains text links to the other two pages and is coded as follows.

```
<h2>Linking Web Pages</h2>

<p>
  Navigation between pages of a Web site is made possible
  links. One or more words on a "linking" page are made s
  clicks that take the reader to a different "linked" pag
  reasons for breaking up a topic into separately linked
</p>

<p>
  <a href="Organize.htm">Topical Organization</a><br>
  <a href="Supplement.htm">Supplemental Information</a>
</p>

<p>
  When creating links between Web pages it is important t
  links to the linking pages. The page author should not
  at the end of a series of links with no direct way to r
</p>
```

Notice the line containing the code

```
<a href="Organize.htm">Topical Organization</a>
```

An `<a>` tag surrounds the string of text "Topical Organization" to convert it into a hyperlink to open the **Organize.htm** page in the browser window. Only the name of this page is required as the **href** URL since the linked page appears in the same directory as the **Home.htm** page from which the link is made. Similar coding links to the **Supplement.htm** page. When either of the two links is clicked, the associated page is retrieved and loaded into the browser, replacing the **Home.htm** page.

When either `Organize.htm` or `Supplement.htm` are showing in the browser window, it is easy to return to the `Home.htm` page by clicking the Back button on the browser's Tool Bar. However, it is a better navigation solution to code your own internal links between pages. In other words, both the `Organize.htm` and `Supplement.htm` pages should include, as shown here, their own links back to `Home.htm` .

Coding for the `Organize.htm` page is shown below. Here the string of text "Return to Home" is made into a link by surrounding it with an `<a>` anchor tag. A click on this link reopens the `Home.htm` page in the browser window. Similar coding appears on the `Supplement.htm` page to return to the home page.

```
<h2>Topic Organization</h2>
```

```
<p>
```

One of the reasons for breaking up a Web topic into separate pages is that there simply may be too much information for including on a single page. For the same reason that books are divided into chapters and chapters are divided into sections and sections are divided into paragraphs, a Web topic may need to be divided into logical units that are presented separately. Rather than being presented as a single continuous stream of text, the topic is organized into related chunks of information that are presented in an overall structure and that provide for isolated reading of individual subtopics.

```
</p>
```

```
<p><a href="Home.htm">Return to Home</a></p>
```

The manner in which you set up links between your pages is at your discretion. To a large extent, it depends on how you subdivide your site into separate pages and the way in which the separate topics relate to one another. Make sure there are no "dead end" pages where visitors arrive with no way to get back to where they came from.

Remote Links

If you know the URL of a Web site anywhere on the World Wide Web you can make a link to that site. This URL is normally in the form

`http://domain name`

where the `http` protocol prefixes the site location (domain name). This type of absolute reference takes you to the home page of that site. The following link code, for example, is used to open the home page at *www.google.com*.

```
<a href="http://www.google.com">Go to Google</a>
```

[Go to Google](http://www.google.com)

To find out the URL of a Web site simply go there and read the URL that appears in the Address box of your browser. Copy this address into the `href` attribute of your `<a>` tag and add the text to make a clickable link.

If you are linking to a particular page at a remote site then the domain name is followed by a path through the directory structure to the specified page. For instance, the following URL accesses a page at *sports.yahoo.com* containing links to various business sites and services.

```
<a href="http://sports.yahoo.com/nfl/teams/">NFL Teams</a>
```

[NFL Teams](http://sports.yahoo.com/nfl/teams/)

This page is located in the `teams` subdirectory of the `nfl` directory of the site. The URL is taken from the browser's Address box for the displayed page.

Link Colors

Depending on your browser's settings, a text link shows up in different colors as a visual clue to the status of the link. Before a linked page is visited for the first time its text link displays in blue. After the page is visited, the link changes to purple. You can tell at a glance whether or not you have previously visited a page.

[Unvisited Link](#)

[Visited Link](#)

In a later tutorial you will learn how you can change these default colors for links that you code.

The target Attribute

By default, linked pages are opened in the same browser window as the linking page subsequently replacing it. When linking to a remote site, a risk is that your visitors will navigate the remote site and eventually lose contact with your site. They can repeatedly click the browser's Back button to get back to your site, but if they have wandered around the remote site to any great extent then backing up to your site may be impractical. You end up losing your visitors to the remote site.

One way around this problem is to **always** link to remote sites in a separate browser window. It is quite easy to open a page in a different window. Simply add the attribute `target="_blank"` to your `<a>` tag as shown below for the above link to the Google site.

```
<a href="http://www.google.com" target="_blank">Go to Goo
```

With this technique, visitors can browse the remote site as much as they wish in a second browser window, and by closing that window return immediately to your linking page in the original window.

Creating text links as described here represent only a couple of the ways in which hyperlinks can tie together multiple Web pages. In a later tutorial, other varieties of links and linking methods are discussed.

3.8: Linking Pages is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

3.9: Displaying Pictures

In addition to presenting text content, Web pages can display graphic images. Later tutorials discuss this topic more fully. Here, basic methods of placing pictures on a page are introduced.

Graphic Formats

There are three popular graphic formats used for Web images. **GIF (Graphics Interchange Format)** is the most widely supported. Pictures saved in this format have the `.gif` file extension. GIF format can display images in black and white, grayscale, or color. It is most often used for line drawings, logos, icons, and other "spot color" images, and is usually not for "continuous color" images such as photographs. Incidentally, the acronym GIF is pronounced "jiff."

The **PNG (Portable Network Graphics)** format was designed as a replacement for the GIF format. PNG, pronounced "ping," has the advantage over GIF of improving image quality while offering better image compression. It does not, however, support animated images. PNG files use the `.png` extension.

The **JPEG (Joint Photographic Experts Group)** format, pronounced "j-peg," is designed for storing photographic images. It can represent millions of colors, and uses compression techniques to compact all of color information into small file sizes. Pictures saved in this format usually have the `.jpg` file extension.

You can display pictures taken with digital cameras or scanned from photographs, or you can copy images from public clip art collections on the Web. If you have available any of the popular graphic editing software packages, then it is usually a simple matter to edit, resize, and convert digital photographs or scanned images to Web formats. Be sure that images are saved in GIF, PNG, or JPEG format for display on your Web pages. Other formats may display properly in the browser, but their large file sizes may make it impractical for visitors to download these images during Web access.

Capturing Images

Some of you may have the talent to create your own graphic images; most of you probably do not. Therefore, you may need to locate existing illustrations that complement the text content on your pages. The Web is a rich source of graphics, and many pictures are available for capturing. You can use your favorite search engine to locate "clip art" sites with millions of images to choose from. The screen display in Figure 3.9.1 shows a Google search finding over 32 million sources of clip art.

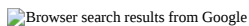
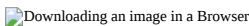
 Browser search results from Google

Figure 3.9.1: Search results for clip art sites. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials. 4](#), [GALILEO Open Learning Materials](#) is licensed under [CC BY-SA 4.0](#))

Once you have found the picture you want to download, it is easy to capture it for your use. As shown in Figure 3.9.2, right-click on the picture and choose "Save Picture As..." from the pop-up menu. In the Save dialog box choose the location on your local PC to download the image. The location can be in the same directory as the Web page on which it will be displayed, or you can create a separate subdirectory for storing media files. You can also rename the picture during its saving if you wish. Be alert to images that are copyrighted. You need permission to use these.

 Downloading an image in a Browser

3.9.2 Downloading an image from the Web

The Tag

Image formats GIF, PNG, and JPEG are placed on Web pages by using the in-line `<img>` tag. The general format for this tag is shown below.

```

```

An `<img>` tag is coded on the Web page *at the location* at which the image is to be displayed. The `src` attribute gives the source, or location, of the image file. If the file is in the same directory as the Web page on which it is displayed, then only the file name is needed as the `src` URL.

The `alt` attribute is required under HTML5. Its `"text"` value gives a brief pop-up description of the image when the mouse is moved on top of the picture. This text string is displayed when users have graphic displays turned off in their browsers, thus giving a text description of missing pictures. Also, the text can be vocalized by special reader software used by visually impaired visitors who otherwise might not be able to see or focus clearly on images.

Note that the `<img>` tag is an empty tag; it does not use a pair of opening and closing tags since it does not surround any content. It only marks the page location of an image. By default, the picture is aligned at the left margin of the page.

The JPEG image shown in Figure 3.9.3 is assumed to be located in the same directory as the Web page that displays it. The image is placed on the page at the location of the `<img>` tag with the accompanying code. (Move your mouse pointer on top of the image to view its pop-up `alt` box.)

```
<p></p>
```

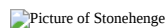


Figure 3.9.3 Code and browser output of image displayed with `<img>` tag.

Notice that you do not physically place the image on the Web page. Rather, you include an `<img>` tag containing a URL address for the image file. When the Web page is loaded into the browser this URL is used to locate the image file, which is then transmitted separately to the browser for display on the page. This is why image file sizes are important considerations. Displaying an image requires an additional data transmission from the Web server – one for each image appearing on the page. Therefore, try to keep the number of images and their sizes as small as possible in order to reduce the number and length of transmissions needed to fill a Web page.

More information about placing images on Web pages is presented in later tutorials. For now, you should be able to modestly decorate your pages with your own pictures or those copied from clip art collections.

Using Graphics as Links

Graphic images can be used as links in the same way as text strings. In this case, an `<img>` tag is surrounded by an `<a>` tag to make the image click sensitive and to link to the page or site coded in the `href` attribute.

The following image was copied from the www.google.com Web site and saved to a local Web directory. It is placed on the page and made into a clickable link to the site that opens in a separate browser window.

```
<a href="http://www.google.com">` tag. More current techniques for working with borders are described in a later tutorial.

### Creating a Media Subdirectory

As you produce more and more Web pages, you will get to the point where your Web directory starts to become overloaded and disorganized. Preferably, documents and picture files that relate to different Web sites or applications should be kept together inside their own separate subdirectories. This arrangement makes it easier to locate and work with related files.

One of the common ways to begin this organization is to create a separate subdirectory within the main Web directory just for your graphic files. In this way, all of your Web pages remain in the main Web directory and all supporting files reside in the subdirectory. Figure 3.9.4 shows a typical arrangement of Web directories, with a `Media` subfolder inside the main Web folder.

Figure 3.9.4: Web directory structure. ("MGA Web Development Tutorials" by Floyd, Kevin; Kwak, Myungjae; and Stines, Alan, [Computer Science and Information Technology Ancillary Materials](#), 4, [GALILEO Open Learning Materials](#) is licensed under [CC BY-SA 4.0](#))

If you create this `Media` subfolder, then the relative URL for the `src` attribute of `<img>` tags needs to reflect this directory structure. For example, if `Page1.htm` in the accompanying illustration displays `Pic1.gif` located in the `Media` subfolder, then its `<img>` tag needs to be coded as:

```

```

This coding indicates that, relative to the directory location of `Page1.htm`, the `Pic1.gif` image is located one level down in the directory hierarchy, inside the `Media` subdirectory. In a later tutorial, the organization of Web pages and supporting files is covered in more detail. At present, you can begin to bring order to your Web files by placing images in a separate subdirectory.

### Deprecated Image Alignment

Horizontal placement of an image can be controlled by enclosing the `<img>` tag inside a `<p>` tag which aligns the image to the left, to the right, or centered on the line. Paragraph alignment can use the deprecated `align="left|center|right"` attribute of the `<p>` tag. The picture below, for example, is centered on a line by itself and is preceded and followed by blank lines with the enclosing `<p>` tag.

```
<p align="center">` tag's `width` and `height` attributes:

```
width="n"
height="n"
```

These `n` sizes are given in pixels. You need only to set one *or* the other dimension, but not both, to maintain proportionality. The browser automatically adjusts the other dimension to keep the picture in correct proportion.

The original size of the downloaded image shown below is 480 pixels (width) by 641 pixels (height). It is resized during browser display with the tag:

```
` Tag

Previously you were introduced to the `<pre>` tag as one method for arranging data into rows and columns for tabular presentation. A block of text surrounded by the `<pre>` tag is displayed in a monospace font in precisely the way it is typed in the text editor. The tag is often used to create simple tables where alignment of columns is produced with embedded blank spaces. For example, the following editor code shows a table produced by using the keyboard Enter key to create separate rows, and the Space Bar key to align columns of data. This table layout appears on a Web page in exactly the format it is typed inside the `<pre>` tags.

```
<pre>
 Table 1
 Sales by Region

Region/Year 2000 2001 2002 2003

East 35.2 37.4 39.8 40.0
West 28.0 25.6 27.4 29.8
South 102.3 86.1 98.6 100.2
North 10.5 8.6 9.8 10.4

</pre>
```

Figure 9.1.1: Code to create a table using the `<pre>` tag.

This method provides limited capabilities for arranging and styling a table. It is useful only for producing fairly simple tables without the need for styling it in conformance with the overall page design. A better method is to construct tables using special HTML tags designed for that purpose.

## A Simple Table

Like any other tabular presentation, a table produced with HTML table tags contains rows and columns. The intersections of these rows and columns form the **cells** of the table. Information can then be placed into these cells. A simple 3 x 3 table is shown in Figure 8-1.

|          |          |          |
|----------|----------|----------|
| Cell 1.1 | Cell 1.2 | Cell 1.3 |
| Cell 2.1 | Cell 2.2 | Cell 2.3 |
| Cell 3.1 | Cell 3.2 | Cell 3.3 |

Figure 9.1.2: A 3 x 3 table.

The rows, columns, and cell intersections of a table are defined with three basic tags. A `<table>` tag surrounds the entire table description; `<tr>` (table row) tags defined the rows of the table; and `<td>` (table data) tags define the cells, or columns, that appear along each row. The `<td>` tag encloses the content that appears in a cell. These tags are shown in their general structural formats in Figure 8.2.

```
<table>
 <tr>
 <td>cell content</td>
 <td>cell content</td>
 ...
 </tr>
 ...
</table>
```

Figure 9.1.3: General formats for basic table tags.

There are as many `<tr>` tags as there are rows in the table, and there are as many `<td>` tags as there are cells (columns) in each row. When building tables, you need to be cautious about properly pairing opening and closing tags. One minor oversight, and the table arrangement can be totally disrupted. Also, take care to organize your coding so that the table structure is clearly evident. For example, the following code is used to create the table shown above in Figure 8.1.

```
<table>
<tr>
 <td>Cell 1.1</td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
</tr>
<tr>
 <td>Cell 2.1</td>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
</tr>
<tr>
 <td>Cell 3.1</td>
 <td>Cell 3.2</td>
 <td>Cell 3.3</td>
</tr>
</table>
```

Figure 9.1.4: Code to create a 3 x 3 table.

Note that `<table>` , `<tr>` , and `<td>` tags are container tags that all require both an opening and closing tag. The `<table>` tag encloses all coding for the table; `<tr>` tags enclose each row of the table; `<td>` tags enclose each cell of a table row.

The code in Listing 8-2 is arranged in a line-by-line hierarchy of tags suggestive of the structure of the table. This manner of coding is helpful in visualizing the table, although you can code table tags in free-format style, like any other HTML coding. For instance, the above table displays correctly coded in the following fashion.

```
<table>
<tr><td>Cell 1.1</td><td>Cell 1.2</td><td>Cell 1.3</td></tr>
<tr><td>Cell 2.1</td><td>Cell 2.2</td><td>Cell 2.3</td></tr>
<tr><td>Cell 3.1</td><td>Cell 3.2</td><td>Cell 3.3</td></tr>
</table>
```

Figure 9.1.5: Alternate code to create a 3 x 3 table.

However, since table cells ( `<td>` ... `</td>` ) can enclose information of varying length and complexity, it is best to code them on separate lines as in Listing 8-2 for more accurate visualization and typing of cell contents.

The table below presents a summary of the HTML tags used for creating tables by discussing what was covered in this tutorial.

Tag	Description
table	Defines a table. All other table elements are coded within this element.
tr	Defines a table row.
td	Defines a table data cell within a row.
th	Defines a table header cell within a row.
caption	Contains a description of the table. The element must be coded immediately after the opening table tag.
thead	Groups one or more rows or tr elements into a table header.
tfooter	Groups one or more rows or tr elements into a table footer.
tbody	Groups one or more rows or tr elements between the header and footer into a table body.

### Table and Cell Borders

By default, table tags produce no borders around the table or around its cells. The code in Listing 8-2 and 8-3 actually create the table layout shown below in Figure 8-3. Data are aligned within rows and columns; however, no borders are displayed. In certain cases, you may not wish to display table borders, but in most cases you will. Therefore, style sheets must be applied to the table and to its cells to display borders around both.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.1.6: A standard 3 x 3 table.

A style sheet is applied to the table *and* to its cells to display borders around both. A border surrounds the entire table by styling the `<table>` tag. Borders surround individual cells by styling the `<td>` tag. The code below produces the bordered table shown in Figure 8.1. The entire table is surrounded by a 1-pixel outset border and each of the cells is surrounded by a 1-pixel inset border. This styling is typical for a standard HTML table.

```
<style type="text/css">
 table {border:outset 1px;}
 td {border:inset 1px;}
</style>

<table>
<tr>
 <td>Cell 1.1</td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
</tr>
<tr>
 <td>Cell 2.1</td>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
</tr>
<tr>
 <td>Cell 3.1</td>
 <td>Cell 3.2</td>
 <td>Cell 3.3</td>
</tr>
</table>
```

Figure 9.1.7: Code to create a bordered 3 x 3 table.

An embedded style sheet is the most convenient method of styling table elements. In-line style sheets can be used; however, individual style sheets would have to appear in *each* of the many `<td>` tags. It is much easier declaring embedded styling for table and td selectors. Tables can have any style of outer borders, and cells can have there own border styles. Border styling is covered in a following tutorial.

### Table Size

By default, the size of a table depends on the size of the data appearing in its cells. A column is as wide as the widest entry in the column. All rows are as wide as the combined width of data within the widest cells. In effect, table cells collapse around the data they contain. Although it is not apparent in the above examples, data are aligned horizontally left and vertically centered within the cells. Style settings to manage table and cell sizes and data alignment are described later in this tutorial.

## Nested Tables

Tables can be nested; that is, a table can appear inside a cell of another table. This arrangement does not involve any particular coding complication, Table specifications are simply inserted as contents of a cell. The cell expands in size to permit full display of the nested table as shown in the following table and HTML code.

Cell 1	Cell 2				
Cell 3	<table><tr><td>Cell A</td><td>Cell B</td></tr><tr><td>Cell C</td><td>Cell D</td></tr></table>	Cell A	Cell B	Cell C	Cell D
Cell A	Cell B				
Cell C	Cell D				

Figure 9.1.8: Nested tables.

```
<style type="text/css">
 table {border:outset 1px;}
 td {border:inset 1px;}
</style>

<table>
<tr>
 <td>Cell 1</td>
 <td>Cell 2</td>
</tr>
<tr>
 <td>Cell 3</td>
 <td>

 <table>
 <tr>
 <td>Cell A</td>
 <td>Cell B</td>
 </tr>
 <tr>
 <td>Cell C</td>
 <td>Cell D</td>
 </tr>
 </table>

 </td>
</tr>
</table>
```

Figure 9.1.9: Code for nested tables.

In this example, you can also see the default alignment of data within cells. Notice in "Cell 2" that the entry is horizontally aligned at the left of the cell; in "Cell 3" the entry is vertically aligned in the middle of the cell.

### Styling Multiple Tables

Since tables provide one of the most effective methods for arranging content on a Web page, it is often the case that a single page contains two or more tables that are either nested tables or multiple tables containing different data content. It is also often the case that these tables need to be styled differently. Therefore, the embedded style sheet must differentiate among the tables so that each can have its own special styling.

Recall from earlier discussions that page elements can be assigned id values to uniquely identify them. These ids are then used in the embedded style sheet to indicate which styles pertain to which unique elements. This technique can be used to apply different styles to different tables on the same page.

For example, the two tables in Figure 8-5 have different border styles and sizes. The first table is styled with standard borders; the second table uses the outset style for both table and cell borders and uses different border widths.

Cell 1	Cell 2
Cell 3	Cell 4

Cell 1	Cell 2
Cell 3	Cell 4

Figure 9.1.10: Styling multiple tables on the same page.

The style sheet and table definitions for these two tables are shown in Listing 8-6.

```
<style type="text/css">
 table#Table1 {border:outset 1px;}
 table#Table1 td {border:inset 1px;}

 table#Table2 {border:outset 3px;}
 table#Table2 td {border:outset 2px;}
</style>

<table id="Table1" >
<tr>
 <td>Cell 1</td>
 <td>Cell 2</td>
</tr>
<tr>
 <td>Cell 3</td>
 <td>Cell 4</td>
</tr>
</table>

<table id="Table2" >
<tr>
 <td>Cell 1</td>
 <td>Cell 2</td>
</tr>
<tr>
 <td>Cell 3</td>
 <td>Cell 4</td>
</tr>
</table>
```

Figure 9.1.11: Code to style multiple tables on the same page.

ID selectors are used to separately style the `<table>` tags for the two tables. Contextual selectors are used to style their `<td>` tags. In this way, any number of tables can take on different styles without styling conflicts. (See tutorial "Applying Special Styles" for a review of ID selectors; see "Contextual Selectors" to review these style selectors.)

This introduction to tables has given you a general overview of tables and style sheet approaches to styling them. The following tutorials cover detailed aspects of table design and formatting.

---

9.1: Creating Tables is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 9.2: Table Borders

When a table is defined by a `<table>` tag with no styles specified, a borderless table as wide and as tall as its enclosed data is displayed. From this simple beginning, you can structure and style a table to take on virtually any display characteristics you can imagine.

### Table Border Styles

Any of the border styles that were introduced earlier can be applied to the outer border of a table. Figure 9.2.2 shows example border styles coded in the general fashion giving in Listing 8-7.

```
<style type="text/css">
 table {border:style 5px;}
 td {border:inset 1px;}
</style>
```

Figure 9.2.1: General style sheet code for table borders.

The border *style* is identified in Figure 9.2.2 for each of the tables. The style sheet uses the general format `{border:style size;}` for all border stylings.

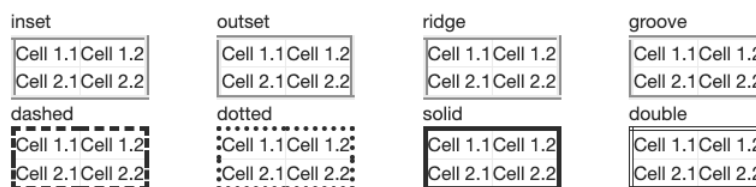


Figure 9.2.2: Varieties of table borders

All tables above have a outer border of 5 pixels to make them visually obvious. Different effects can be achieved with different border sizes.

### Individual Border Styles

In the above examples, borders appear around all four sides of the table; however, you can specify individual styling for the separate sides. Use the properties `border-top`, `border-bottom`, `border-left`, and `border-right` to selectively apply borders around the table. For example, the table in Figure 9.2.4 displays only top and bottom borders with the style sheet given in Listing 8-8.

```
<style type="text/css">
 table { border-top:outset 5px; border-bottom:outset 5px; }
 td {border:inset 1px;}
</style>
```

Figure 9.2.3: Style sheet for top and bottom table borders.

Cell 1.1	Cell 1.2
Cell 2.1	Cell 2.2

Figure 9.2.4: Table with top and bottom borders.

### Collapsed Borders

Borders surrounding the cells in a table normally are displayed independently from each other. That is, each cell is surrounded by its own border thereby giving a "double border" look to side-by-side cells. These double borders can be collapsed into single

borders with the border-collapse style property applied to the table.

Property	Values
border-collapse	separate collapse

Figure 8-8. Table border-collapse style property.

Figure 9.2.5: Table border-collapse style property.

The default border setting for a table is border-collapse:separate. By specifying border-collapse:collapse, cells share "single borders" between them. This effect is illustrated in the tables in Figure 9.2.7 styled with collapsed borders.

```
<style type="text/css">
 table {border:style 5; border-collapse:collapse; }
 td {border:solid 1px;}
</style>
```

Figure 9.2.6: Style sheet to collapse table borders.

inset	outset	ridge	groove
Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2
dashed	dotted	solid	double
Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2

Figure 9.2.7: Tables with collapsed cell borders.

Note that collapsed borders are specified for the **<table>** tag even though the setting is applied to the table's cells. Also, you cannot selectively apply collapsed border style to individual cells. When table borders are collapsed, all cell borders are styled as border:solid 1px; other styles and widths are not recognized.

## Cell Border Styles

If cell borders are not collapsed, the same set of styles that are applied to the table's outer border can be applied to its individual cells. In the examples in Figure 9.2.8, the outer borders of the tables are 1px in width and cell borders are 3px in width to emphasize their styling.

```
<style type="text/css">
 table {border:solid 1px;}
 td {border:style 3px;}
</style>
```

Figure 9.2.8: General style sheet code for cell borders.

inset	outset	ridge	groove
Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2
dashed	dotted	solid	double
Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2	Cell 1.1Cell 1.2 Cell 2.1Cell 2.2

Figure 9.2.9: Varieties of cell borders.

## Selective Border Styles

Borders surrounding cells can be selectively styled with `border-top`, `border-bottom`, `border-left`, and `border-right` styles applied to the `<td>` tag. These styles are normally used to display top-and-bottom or left-and-right borders to emphasize the rows or columns of a table. The two tables shown in Figure 9.2.11 are displayed with the style sheets given in Listing 8-11.

```
<style type="text/css">
 table {border:outset 5px; border-collapse:collapse;}
 td { border-top:solid 2px; border-bottom:solid 2px; }
</style>

<style type="text/css">
 table {border:outset 5px; border-collapse:collapse;}
 td { border-left:solid 2px; border-right:solid 2px; }
</style>
```

Figure 9.2.10: Style sheets to apply row and column borders.

top-and-bottom	left-and-right
Cell 1.1Cell 1.2Cell 1.3	Cell 1.1Cell 1.2Cell 1.3
Cell 2.1Cell 2.2Cell 2.3	Cell 2.1Cell 2.2Cell 2.3
Cell 3.1Cell 3.2Cell 3.3	Cell 3.1Cell 3.2Cell 3.3

Figure 9.2.11: Application of row and column borders.

In order to extend row and column borders across the entire table, borders must be collapsed. Otherwise, breaks in the lines appear between the separate cells. As mentioned above, collapsed borders also means that the various cell border styles do not display. Therefore, a solid border style is the only one that need be used for the cells.

## Individual Border Styles

Cell borders can be styled independently. If needed, each cell can have a different border style. Usually, only selected cells are given emphatic borders to highlight their contents. Of course, in order to view cell borders, the table border cannot be collapsed.

The following table displays different cell borders along its bottom row.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.2.12: Emphasizing a table row with cell borders.

The above effect can be achieved by applying in-line style sheets to the selected cells. However, this is the type of situation especially made for the use of ID selectors and contextual selectors. Complete table and style sheet coding is shown below.

```
<style type="text/css">
 table#TAB1 {border:outset 3px;}
 table#TAB1 td {border:inset 1px;}
 table#TAB1 tr#ROW3 td {border:outset 3px;}
</style>

<table id="TAB1" >
<tr>
 <td>Cell 1.1</td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
</tr>
<tr>
 <td>Cell 2.1</td>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
</tr>
<tr id="ROW3" >
 <td>Cell 3.1</td>
 <td>Cell 3.2</td>
 <td>Cell 3.3</td>
</tr>
</table>
```

Figure 9.2.13: Code to emphasizing a table row with cell borders.

The table itself is assigned an id in order to uniquely identify it among other tables on the page. Moreover, the final row of the table is given an id to uniquely identify it among the other rows of the table. This is the row in which special border styling is applied to its cells.

Within the style sheet, the table is given an outer border style through the ID selector `table#TAB1`, selecting the `<table>` tag with `id="TAB1"` for application of the border style. The contextual selector `table#TAB1 td` applies a 1-pixel inset border to all cells of this table -- i.e., where a `<td>` tag is contained within a `<table>` tag for a table with `id="TAB1"`.

A contextual selector is also used for styling the last row of the table. A 3-pixel outset border is applied with the selector `table#TAB1 tr#ROW3 td` -- i.e., where a `<td>` tag is contained within a `<tr>` tag with `id="ROW3"`, which is contained within a `<table>` tag with `id="TAB1"`.

If only a single cell has its borders styled, then that cell can be given an id, such as, `id="CELL9"`. Under such conditions, the selector for styling that one cell would be

```
table#TAB1 td#CELL9 {border:outset 3px;}
```

where a `<td>` tag with `id="CELL9"` is contained within a `<table>` tag with `id="TAB1"`. In this case, it would not be necessary to identify the table row.

### Deprecated Table Attributes

Borders are displayed around the table and between table cells with the `border="n"` attribute of the `<table>` tag, where `n` is the width of the outer border in pixels. The following table has 1-pixel borders:

<table border="1">

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Changing the width of the outer border does not effect the size of cell borders. The default style of the outer border is "outset"; the default style of inner borders is "inset":

<table border="5">

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

---

9.2: Table Borders is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 9.3: Table Captioning

### The `<caption>` Tag

You can caption a table by coding a `<caption>` tag immediately following the `<table>` tag. Enter a table title between the opening and closing `<caption>` tags and the title displays centered over the table. The caption is displayed in the page default font style and size, and formatting is applied with style sheets as is done in the following code example for the table appearing in Figure 8.13.

This is My Table

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.3.1: A table with a caption.

```
<style type="text/css">
 table {border:outset 3px;}
 table td {border:inset 1px;}
 table caption {font:bold 10pt arial;}
</style>

<table>
 <caption>This is My Table</caption>
 <tr>
 <td>Cell 1.1</td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
 </tr>
 <tr>
 <td>Cell 2.1</td>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
 </tr>
 <tr>
 <td>Cell 3.1</td>
 <td>Cell 3.2</td>
 <td>Cell 3.3</td>
 </tr>
</table>
```

Figure 9.3.2: Code for a table with a caption.

You can apply styling to the caption through a caption selector. In this example, font styling is applied with the shorthand font notation,

*font:style variant weight size family*

The caption is centered by default. The text-align property can be applied to the caption to align it with the left or right edge of the table.

Captions appear only at the top of the table. Style sheet standards propose the caption-side:top|bottom|left|right style declaration for positioning captions; however, this property has not been implemented in current browsers.

### The `<th>` Tag

Column headings can be supplied with `<th>` tags in place of `<td>` tags enclosing cell contents. The `<th>` tag automatically centers and bolds the enclosed text. Additional styles can be applied to the tags for additional formatting. Notice in the following example that the th selector is added to the style sheet for borders surrounding these cells.

This is My Table

Column 1	Column 2	Column 3
Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.3.3: Using the `<th>` tag for column headings.

```
<style type="text/css">
 table {border:outset 3px; border-collapse:collapse;}
 table td, th {border:inset 1px;}
 table caption {font:bold 10pt arial;}
</style>

<table>
 <caption>This is My Table</caption>
 <tr>
 <th> Column 1 </th>
 <th> Column 2 </th>
 <th> Column 3 </th>
 </tr>
 <tr>
 <td>Cell 1.1</td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
 </tr>
 <tr>
 <td>Cell 2.1</td>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
 </tr>
 <tr>
 <td>Cell 3.1</td>
 <td>Cell 3.2</td>
 <td>Cell 3.3</td>
 </tr>
</table>
```

Figure 9.3.4: Code to produce column headings for a table.

The `<th>` tag is unique only in that it provides default bold and centered styling for its contents. With style sheets, however, this tag becomes redundant since this same styling can be applied without using this special tag. The following table duplicates the previous table without `<th>` tags simply by identifying and styling the first row of the table.

This is My Table

Column 1	Column 2	Column 3
Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.3.5: Producing column headings by styling a row.

```
<style type="text/css">
 table {border:outset 3px; border-collapse:collapse;}
 table tr#HEAD {font:bold; text-align:center;}
 table td {border:inset 1px;}
 table caption {font:bold 10pt arial;}
</style>

<table>
 <caption>This is My Table</caption>
 <tr id="HEAD">
 <td>Column 1</td>
 <td>Column 2</td>
 <td>Column 2</td>
 </tr>
 <tr>
 <td>Cell 1.1</td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
 </tr>
 <tr>
 <td>Cell 2.1</td>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
 </tr>
 <tr>
 <td>Cell 3.1</td>
 <td>Cell 3.2</td>
 <td>Cell 3.3</td>
 </tr>
</table>
```

Figure 9.3.6: Code to produce column headings by styling a row.

The contextual selector `tr#HEAD td` indicates styling for `td` tags that appear inside a `tr` tag with `id="HEAD"`. It is your choice whether to use the `th` tag to take advantage of its inherent styling or to manage all styling through style sheets. As a general rule, the fewer tag to deal with the better.

### Deprecated Caption align Attribute

Captions can appear at the top or bottom of a table using the `align="top"` (default) or `align="bottom"` attribute of the `<caption>` tag.

Top Table Caption

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

e.g., `<caption align="top">Top Table Caption</caption>`

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Bottom Table Caption

e.g., `<caption align="bottom">Bottom Table Caption</caption>`

As noted above, style sheet standards propose the caption-side property to replace the align attribute; however, this property has not been implemented in current browsers.

---

9.3: Table Captioning is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 9.4: Table Colors and Backgrounds

Tables can have outer border colors as well as cell border colors. They also can have background colors, patterns, and graphic images as backdrops for the full table or for selected rows and cells.

### Table Border Colors

The appearance of border colors is different depending on the border style. Below are example border styles displayed in red.

```
<style type="text/css">
.inset { border:inset 5px red; }
.outset { border:outset 5px red;}
.ridge { border:ridge 5px red; }
.groove { border:groove 5px red;}
.dashed { border:dashed 5px red;}
.dotted { border:dotted 5px red;}
.solid { border:solid 5px red;}
.double {border:double 5px red;}

.inset td, .outset td, .ridge td, .groove td, .dashed td,
.dotted td, .solid td, .double td {border:inset 1px #000;}
</style>
```

Figure 9.4.1: General style sheet for table border colors.

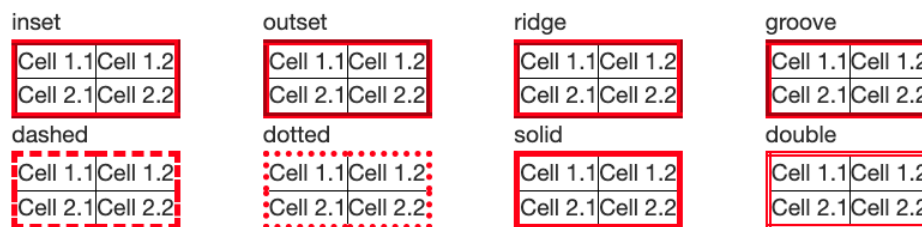


Figure 9.4.2: Varieties of table border styles and colors.

You can create your own 3-D shading effects by styling each of the borders separately with border-top, border-right, border-bottom, and border-left colors. The following tables creates their own 3-D effects with solid borders tinted to resemble inset and outset styles. The style sheets used for these two tables are shown in Figure 9.4.4.

Table 1			Table 2		
Cell 1.1	Cell 1.2	Cell 1.3	Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3	Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3	Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.4.3: Table border 3-D effects using border colors.

```
<style type="text/css">
/* INSET STYLE */
#TAB1 { border:solid 7px;
 border-color:#B22222 #FEB9B9 #FEB9B9 #B22222; }

/* OUTSET STYLE */
#TAB2 { border:solid 7px;
 border-top-color:#FEB9B9;
 border-right-color:#B22222;
 border-bottom-color:#B22222;
 border-left-color:#FEB9B9; }

#TAB1 td, #TAB2 td { border:inset 1px #000; }
</style>
```

Figure 9.4.4: Style sheet code for tables with individual border colors.

### Cell Border Colors

Borders around table cells are styled and colored independently from the outer border. In the following examples, the outer border is 1 pixel in width for deemphasis, and cell borders are red and 3 pixels in width.

```
<style type="text/css">
table { border:solid 1px #000; }
td { border:style 3px red; }
</style>
```

Figure 9.4.5: General style sheet for table cell border colors.

inset	outset	ridge	groove																
<table><tr><td>Cell 1.1</td><td>Cell 1.2</td></tr><tr><td>Cell 2.1</td><td>Cell 2.2</td></tr></table>	Cell 1.1	Cell 1.2	Cell 2.1	Cell 2.2	<table><tr><td>Cell 1.1</td><td>Cell 1.2</td></tr><tr><td>Cell 2.1</td><td>Cell 2.2</td></tr></table>	Cell 1.1	Cell 1.2	Cell 2.1	Cell 2.2	<table><tr><td>Cell 1.1</td><td>Cell 1.2</td></tr><tr><td>Cell 2.1</td><td>Cell 2.2</td></tr></table>	Cell 1.1	Cell 1.2	Cell 2.1	Cell 2.2	<table><tr><td>Cell 1.1</td><td>Cell 1.2</td></tr><tr><td>Cell 2.1</td><td>Cell 2.2</td></tr></table>	Cell 1.1	Cell 1.2	Cell 2.1	Cell 2.2
Cell 1.1	Cell 1.2																		
Cell 2.1	Cell 2.2																		
Cell 1.1	Cell 1.2																		
Cell 2.1	Cell 2.2																		
Cell 1.1	Cell 1.2																		
Cell 2.1	Cell 2.2																		
Cell 1.1	Cell 1.2																		
Cell 2.1	Cell 2.2																		
dashed	dotted	solid	double																
<table><tr><td>Cell 1.1</td><td>Cell 1.2</td></tr><tr><td>Cell 2.1</td><td>Cell 2.2</td></tr></table>	Cell 1.1	Cell 1.2	Cell 2.1	Cell 2.2	<table><tr><td>Cell 1.1</td><td>Cell 1.2</td></tr><tr><td>Cell 2.1</td><td>Cell 2.2</td></tr></table>	Cell 1.1	Cell 1.2	Cell 2.1	Cell 2.2	<table><tr><td>Cell 1.1</td><td>Cell 1.2</td></tr><tr><td>Cell 2.1</td><td>Cell 2.2</td></tr></table>	Cell 1.1	Cell 1.2	Cell 2.1	Cell 2.2	<table><tr><td>Cell 1.1</td><td>Cell 1.2</td></tr><tr><td>Cell 2.1</td><td>Cell 2.2</td></tr></table>	Cell 1.1	Cell 1.2	Cell 2.1	Cell 2.2
Cell 1.1	Cell 1.2																		
Cell 2.1	Cell 2.2																		
Cell 1.1	Cell 1.2																		
Cell 2.1	Cell 2.2																		
Cell 1.1	Cell 1.2																		
Cell 2.1	Cell 2.2																		
Cell 1.1	Cell 1.2																		
Cell 2.1	Cell 2.2																		

Figure 9.4.6: Varieties of table cell border styles and colors.

As in the case with table borders, cells borders can be styled and colored independently. In Figure 8.19, all cells in the two tables are styled with solid borders 5 pixels in width. Different shades of red are applied to their sides to create 3-D effects resembling inset and outset borders. The two applicable style sheets are shown in Listing 8-19.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.4.7: Tables with individual cell border colors.

```
<style type="text/css">
/* INSET STYLE */
table {border:solid 1px #000;}
table td {border:solid 5px red;
 border-top-color:#FEB9B9;
 border-right-color:#B22222;
 border-bottom-color:#B22222;
 border-left-color:#FEB9B9;}

</style>

<style type="text/css">
/* OUTSET STYLE */
table {border:solid 1px #000;}
table td {border:solid 5px red;
 border-top-color:#B22222;
 border-right-color:#FEB9B9;
 border-bottom-color:#FEB9B9;
 border-left-color:#B22222;}

</style>
```

Figure 9.4.8: Style sheet code for tables with individual cell border colors.

### Table Background Colors

You can specify a background color for the whole table or select specific rows or cells to apply background colors. In the following table, a light gray background color is applied to the entire table by setting the background-color property of the `<table>` tag.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.4.9: Table with a background color.

```
<style type="text/css">
table { border:ridge 5px red; background-color:#F6F6F6; color:#FFF; }
table td { border:inset 1px #000; }

</style>
```

Figure 9.4.10: Code for table with a background color.

### Cell Background and Text Colors

The following table contains different background colors for each of its rows by specifying a background-color property for each identified `<tr>` tag. The text color of selected rows is set with the color property. In addition, one particular cell is assigned a background color that overrides the color of the row it occupies, and it is given a different border style.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.4.11: Table with a row color.

```
<style type="text/css">
 table {border:ridge 5px red;}
 table td {border:inset 1px #000;}
 table tr#ROW1 {background-color:red; color:white;}
 table tr#ROW2 {background-color:white;}
 table tr#ROW3 {background-color:blue; color:white;}
 table td#CELL9 {border:inset 4px red; background-color:navy;}
</style>

<table>
 <tr id="ROW1" >
 <td>Cell 1.1</td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
 </tr>
 <tr id="ROW2" >
 <td>Cell 2.1</td>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
 </tr>
 <tr id="ROW3" >
 <td>Cell 3.1</td>
 <td>Cell 3.2</td>
 <td id="CELL9" >Cell 3.3</td>
 </tr>
</table>
```

Figure 9.4.12: Code for table with row colors.

### Table Backgrounds

In the same fashion used for background colors, you can specify a background image for a table or any of its rows or cells. In the following example, a textured background is applied to the entire table with the background-image property applied to the `<table>` tag. By default, the pattern is repeated across and down to fill the table area.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.4.13: Table with background image.

```
<style type="text/css">
 table {border:ridge 5px red; background-image:url(heart.png); }
 table td {border:inset 1px #000;}
</style>
```

Figure 9.4.14: Code for table with background image.

If needed, other background properties -- background-repeat and background-position -- can be applied.

## Deprecated Table Attributes

Border colors can be specified with the **bordercolor**="color" attribute of the `<table>` tag, where *color* is a color name or hexadecimal value.

```
<table border="5" bordercolor="red">
```

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

A colored shadow effect can be applied to a table's outer borders with the **bordercolordark**="color" and **bordercolorlight**="color" attributes.

```
<table border="5" bordercolordark="red" bordercolorlight="pink">
```

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

The background color of the table is given by the **bgcolor**="color" attribute. When applied to the `<table>` tag, the color fills the background.

```
<table border="5" bgcolor="lightblue">
```

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Cell background colors are set by applying the **bgcolor** attribute to a `<tr>` tag (to color the row) or to a `<td>` tag (to color the cell). Cell colors override row colors which, in turn, override table background colors. The following table specifies a row color with one of its cells taking on a different background color.

```
<table border="5" bgcolor="gainsboro">
```

```
<tr bgcolor="lightblue">
 <td bgcolor="pink">
```

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

A background image is given by the **background**="url" attribute. Background images can be applied to individual table cells by coding the attribute in a `<td>` tag.

```
<table border="5" background="bkgnd.jpg">
```

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

These table, row, and cell attributes are deprecated and should be substituted with equivalent style sheet properties.

9.4: Table Colors and Backgrounds is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 9.5: Table Size and Alignment

The sizes of cells in a table are governed by the size of the data within the cells. The table is as wide as the total of the widths of the widest cells. The table is as tall as the combined heights of the rows. In some cases, however, you may want to specify a width and/or height for a table irrespective of the data it contains.

### Table Widths and Heights

The overall table width is supplied by the width property, and its height is given by the height property. Sizes can be measured in pixels -- to set exact sizes; or they can be expressed as percentages of the width and height of the browser window -- to vary the table size to correspond with the window size.

The table in Figure 8-23 is shown at its default width and height. It is sized no larger than is needed to display the data contained in its cells.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.5.1: Table with default width and height.

This same table is redisplayed in Figure 8-24 with the specified width and height shown in the accompanying style sheet.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.5.2: Table with specified width and height.

```
<style type="text/css">
 table {border:outset 1; width:50%; height:100px; }
 table td {border:inset 1;}
</style>
```

Figure 9.5.3: Code for table with given width and height.

The table width is set to 50% of the page width. If you resize your browser window, you can see that the table width remains half as wide as the page. The height is specified as 100 pixels and remains at that height irrespective of the window size. Although it is common to specify table widths for visual design purposes, table heights are normally permitted to expand in size for as many rows as it contains.

### Variable and Fixed Table Widths

A table's width can be set using percentages or pixels. A percentage width such as 50%, creates a variable-width table that the browser stretches to take up 50% of the browser window regardless of the screen resolution or window size. A table width of 100% can be used when you want the table to fill the entire browser window. Pixels can be used to set the width for a fixed-width table. This provides a more consistent look across different browsers. A table with a width of 500px will *only* take up 500 pixels of the browser window regardless of the screen resolution or window size. In many cases, fixed-width tables are centered on the page.

## Column Widths

You can control column widths by specifying a size for any *one* cell in each column. This single cell width becomes the width to which all other cells in the column conform. The following table is sized to 50% of the width of the window. Subsequently, cells in the first row of the table (containing column headings in this case) are assigned width percentages. These percentages are of the *width of the table*, not of the page. Therefore, the sum of column widths should add up to 100%: the full width of the table.

Column 1	Column 2	Column 3
Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.5.4: Table with specified column widths.

```
<style type="text/css">
 table {border:outset 1; width:50%; }
 table td {border:inset 1;}
 table tr#HEAD {font-weight:bold;
 text-align:center;
 background-color:#F0F0F0;}

 table td#CELL1 { width:25%; }
 table td#CELL2 { width:50%; }
 table td#CELL3 { width:25%; }
</style>

<table>
 <tr id="HEAD">
 <td id="CELL1" >Column 1</td>
 <td id="CELL2" >Column 2</td>
 <td id="CELL3" >Column 3</td>
 </tr>
 <tr>
 <td>Cell 1.1</td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
 </tr>
 <tr>
 <td>Cell 2.1</td>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
 </tr>
 <tr>
 <td>Cell 3.1</td>
 <td>Cell 3.2</td>
 <td>Cell 3.3</td>
 </tr>
</table>
```

Figure 9.5.5: Code for table with specified column widths.

It is not necessary to provide width percentages for all cells across a row. Any remaining cells on a row are sized equally by the browser so that the widths of all cells along the row total 100% of the width of the table. If cell widths are not sized sufficiently to display their data, the browser increases the widths to the minimum needed.

Rather than using percentages, you can specify table widths in pixels. However, you should typically use percentages to give the browser sufficient latitude in sizing the table to fit within the available window width.

You can set rows heights in the same way you set column widths: specify the height for *one* cell along the row and the other cells expand to conform to that height. Unless you have a reason for expanding the height of a row, you should let the browser determine the necessary height to display the data in the table.

A Table

Column 1	Column 2	Column 3
Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.5.6: A right-floated table.

Tables can be aligned to the left or right of the page by applying the float property to the table selector. When the table is floated, word wrapping takes place around the table in a fashion similar to the way word wrapping occurs around floated images.

Code for a floated table appears immediately before the text to be wrapped around the table. Margins can be added to sides of the table to leave white space between the table and the wrapped text. The following partial style sheet applies to the floated table shown in Figure 8-26.

```
<style type="text/css">
 table {border:outset 1; float:right; margin-left:20px; margin-bottom:10px; }
 ...
</style>
```

Figure 9.5.7: Partial style sheet for a floated table.

A table can be aligned to the left, right, or centered on a line by itself by using margin-left and margin-right properties. The table in Figure 8-27 shows a table centered by using margin-left:auto; margin-right:auto;.

A Centered Table

Column 1	Column 2	Column 3
Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.5.8: A centered table.

```
<style type="text/css">
 table {border:outset 1; margin-left:auto; margin-right:auto; }
 table td {border:inset 1; text-align:center;}
 table caption {font-weight:bold;}
 table tr#HEAD {font-weight:bold;
 text-align:center;
 background-color:#F0F0F0;}
</style>

<table>
 <caption>A Centered Table</caption>
 ...
</table>
```

Figure 9.5.9: Style sheet for a centered table.

### Deprecated Table Attributes

The width of a table is given by the **width**="n" attribute of the `<table>` tag, where *n* is a pixel width value or a percentage of the width of the browser window.

```
<table border="1" width="35%">
```

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Cell widths can be set to a pixel value or percentage with the width attribute coded in a `<td>` or `<th>` tag. Cell heights are given with the **height**="n" attribute.

A table can be aligned at the left or right of the page with the **align**="left|right" attribute of the `<table>` tag. Text is wrapped around the positioned table. White space surrounding the table must be added with style sheet margin settings. A table with **align**="center" appears centered on a line by itself.

```
<table border="1" align="center">
```

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

<table border="1" align="right">

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

---

9.5: Table Size and Alignment is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 9.6: Aligning Text and Cells

Table cells are normally only as large as the data they contain. Their borders close in around the data. Often, this effect makes a table seem crowded and not particularly easy to read. Fortunately, you can introduce white space around data within table cells, and you can expand the amount of space between cells.

### Cell Padding

Cell padding refers to the amount of blank space surrounding the text within a cell. Its value is set with padding applied to the cell selector in a style sheet. In the following table, padding of 5 pixels is placed around the data in the table cells.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.6.1: A table with cell padding.

```
<style type="text/css">
 table {border:outset 1px #000;}
 table td {border:inset 1px #000; padding:5px }
</style>
```

Figure 9.6.2: Style sheet to apply padding to table cells.

### Text Alignment

By default, text appears in a table aligned horizontally to the *left* of the cell and vertically within the *middle* of the cell. This alignment becomes noticeable when cell sizes are larger than needed to display their data.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.6.3: A table sized to show text alignment within cells.

You can override these default horizontal and vertical alignments to position content anywhere within the cell.

### Vertical Alignment

By using the vertical-align property, content can be aligned at the top, middle, or bottom of a cell. The table in Figure 8-31 applies these alignments separately to its three rows. Notice that the table is given width and height settings to make cells larger than needed so that vertical alignment of cell data is visually apparent.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.6.4: A table sized to show vertical alignment within cells.

```
<style type="text/css">
 table {border:outset 1px #000; width:250px; height:150px}
 table td {border:inset 1px #000}
 table tr#ROW1 { vertical-align:top }
 table tr#ROW2 { vertical-align:middle }
 table tr#ROW3 { vertical-align:bottom }
</style>

<table>
<tr id="ROW1" >
 <td>Cell 1.1</td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
</tr>
<tr id="ROW2" >
 <td>Cell 2.1</td>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
</tr>
<tr id="ROW3" >
 <td>Cell 3.1</td>
 <td>Cell 3.2</td>
 <td>Cell 3.3</td>
</tr>
</table>
```

Figure 9.6.5: Style sheet to apply vertical alignment of text within cells.

### Horizontal Alignment

By using the text-align property, content can be aligned at the left, center, or right of a cell. The following table applies these alignments separately to its three rows that have vertical alignments like the previous table.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Figure 9.6.6: A table sized to show horizontal alignment within cells.

```
<style type="text/css">
 table {border:outset 1px #000; width:250px; height:150px}
 table td {border:inset 1px #000}
 table tr#ROW1 {vertical-align:top; text-align:left }
 table tr#ROW2 {vertical-align:middle; text-align:center }
 table tr#ROW3 {vertical-align:bottom; text-align:right }
</style>
```

Figure 9.6.7: Style sheet to apply horizontal alignment of text within cells.

### Empty Cells

Borders surround cells only when there is data inside the cells. If this does not occur, no borders are displayed. This effect can be seen in the first of the two tables in Figure 8-33 where data are missing from three of the cells. This may be an acceptable display on occasion, but usually borders should display even around empty cells. This situation can be handled by coding a non-break space character (&nbsp;) in the cells thereby producing the second of the table displays where all borders are visible.

	Cell 1.2	Cell 1.3
Cell 2.1		Cell 2.3
Cell 3.1	Cell 3.2	

	Cell 1.2	Cell 1.3
Cell 2.1		Cell 2.3
Cell 3.1	Cell 3.2	

Figure 9.6.8: Tables with missing cell data.

```
<table>
<tr>
 <td> </td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
</tr>
<tr>
 <td>Cell 2.1</td>
 <td> </td>
 <td>Cell 2.3</td>
</tr>
<tr>
 <td>Cell 3.1</td>
 <td>Cell 3.2</td>
 <td> </td>
</tr>
</table>
```

Figure 9.6.9: Using non-break space characters to display borders around empty cells.

### Non-wrapped Table Cells

By default, text within cells is word wrapped to fit within the overall size of the table. Normally, this is acceptable, to let the browser arrange cell content for best fit with the cells. For certain cells, you may not wish the browser to wrap its content. Such an example might be a row of column headings or a column of row labels. The following table shows default sizing and alignment of cell contents with non-wrapped display of column and row headings.

	Table Column 1	Table Column 2	Table Column 3
Table Row 1	Cell 1.1	Cell 1.2	Cell 1.3
Table Row 2	Cell 2.1	Cell 2.2	Cell 2.3

Figure 9.6.10: Table with properly spaced row and column headings.

The problem arises when the table is resized. Such a problem may occur when the browser window is sized smaller than the width of the table needed for non-wrapped presentation of data. When this happens, the headings may wrap inside their cells which leads to the table display shown below.

	Table Column 1	Table Column 2	Table Column 3
Table Row 1	Cell 1.1	Cell 1.2	Cell 1.3
Table Row 2	Cell 2.1	Cell 2.2	Cell 2.3

Figure 9.6.11: Table with improperly spaced row and column headings.

There is nothing wrong with the content of the table, but it may look quite different from what you intended. Text wrapping can be prevented in table cells by coding the **white-space** style property for any cell in which you do not wish text to be wrapped.

Property:	Value
white-space	normal nowrap pre

Figure 9.6.12: The white-space style property.

This style property determines how the browser should handle white spaces within a text string. The normal setting collapses all contiguous spaces to a single space, and breaks the line of text at a blank space where the remaining text on the line does not fit within the specified width of its container. In contrast, the nowrap setting treats spaces as non-break space characters (&nbsp;) and does not break the line. The pre setting works like a `<pre>` tag, and retains as many blank spaces as are coded by again treating them as non-break spaces.

If a table cell is not wide enough to display its text on a single line, then the text normally is wrapped at a blank space within the text string. This occurs with the column and row headings in the table in Figure 8-35. By setting the `white-space:nowrap` property for these table cells, wrapping of headings can be avoided. These specifications are made in recoding the table as follows.

	Table Column 1	Table Column 2	Table Column 3
Table Row 1	Cell 1.1	Cell 1.2	Cell 1.3
Table Row 2	Cell 2.1	Cell 2.2	Cell 2.3

Figure 9.6.13: Table with non-wrapped row and column headings.

The following listing shows the complete code for the above table with non-wrapped cells. In this case, style class `.NOWRAP` is defined with the property setting `white-space:nowrap`. All cells that should not permitting text wrapping are assigned to this class.

```
<style type="text/css">
 table {border:outset 1px #000; width:230px}
 table td {border:inset 1px #000; padding:3px}
 table tr#C-HEAD {font-weight:bold; background-color:#F0F0F0; text-align:center}
 table td#R-HEAD1 {font-weight:bold; background-color:#F0F0F0; text-align:left}
 table td#R-HEAD2 {font-weight:bold; background-color:#F0F0F0; text-align:left}
 table td#BLANK {background-color:#FFFFFF}
 .NOWRAP {white-space:nowrap}
</style>

<table>
<tr id="C-HEAD">
 <td id="BLANK"> class="NOWRAP" </td>
 <td class="NOWRAP" >Table Column 1</td>
 <td class="NOWRAP" >Table Column 2</td>
 <td class="NOWRAP" >Table Column 3</td>
</tr>
<tr>
 <td id="R-HEAD1" class="NOWRAP" >Table Row 1</td>
 <td>Cell 1.1</td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
</tr>
<table>
<table>
 <td id="R-HEAD2" class="NOWRAP" >Table Row 2</td>
 <td>Cell 2.1</td>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
</tr>
</table>
```

Figure 9.6.14: Using non-wrapping of table cells.

### Deprecated Table Attributes

Text is horizontally aligned within cells with the **align**="left|center|right" attribute applied to the **<tr>** , **<td>** , or **<th>** tags. Text is vertically aligned within cells by applying the **valign**="top|middle|bottom" attribute to the cell.

Cell padding is specified with the **cellpadding**="n" (pixels) attribute coded in the **<table>** tag. Spacing between cells is given by the **cellspacing**="n" (pixels) attribute. The cellspacing attribute must be used until browsers adopt the proposed border-spacing style property. More information on this deprecated attribute is below.

```
<table border="1" cellpadding="5">
```

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

```
<table border="1" cellpadding="5" cellspacing="5">
```

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

Wrapping of text within cells can be controlled by coding **nowrap** or **nowrap="true"** for any table cell: `<td nowrap>` or `<td nowrap="true">`.

### Cell Spacing

Cell spacing refers to the amount of space (in pixels) between the cells of the table, effectively, the width of the inner borders between cells. Although CSS standards propose the border-spacing property to adjust this spacing, current browsers do not recognize it. Therefore, cell spacing must be applied with the deprecated **cellspacing="n"** attribute of the `<table>` tag, specifying the number of pixels between cell borders. Although this is a deprecated attribute, it remains valid under HTML5 Strict standards. In the following example, cellspacing is set to 10px.

Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3
Cell 3.1	Cell 3.2	Cell 3.3

```
<style type="text/css">
 table {border:outset 1px #000}
 table td {border:inset 3px #000; padding:5px}
</style>

<table cellspacing="10" >
 ...
</table>
```

9.6: Aligning Text and Cells is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 9.7: Column and Row Spanning

Table cells can be combined to make one larger cell from two or more contiguous cells. A cell can span two or more columns or two or more rows. One use of spanning is to display a heading across several columns as shown by the heading cells in the following table.

Heading 1		Heading 2	
Cell 2.1	Cell 2.2	Cell 2.3	Cell 2.4
Cell 3.1	Cell 3.2	Cell 3.3	Cell 3.4

Figure 9.7.1: A table with cell padding.

### The colspan Attribute

In the above example, the first row of the table is used for column headings. The cell specifications use the **colspan**="n" attribute to extend each of the `<td>` heading cells across 2 cells.

```
<style type="text/css">
 table {border:outset 1px}
 table td {border:inset 1px; padding:3px}
 table tr#HEAD {font-weight:bold;
 text-align:center;
 background-color:#F0F0F0}
</style>

<table>
<tr id="HEAD">
 <td colspan="2">Heading 1</td>
 <td colspan="2">Heading 2</td>
</tr>
<tr>
 <td>Cell 2.1</td>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
 <td>Cell 2.4</td>
</tr>
<tr>
 <td>Cell 3.1</td>
 <td>Cell 3.2</td>
 <td>Cell 3.3</td>
 <td>Cell 3.4</td>
</tr>
</table>
```

Figure 9.7.2: Code to span table cells.

Although there are four `<td>` cell tags for each data row, note that there is a need for only two `<td>` heading tags in the first row. Since each `<td>` heading tag spans two cells (`colspan="2"`), a total of four cells, the full width of the table, is accounted for.

### The rowspan Attribute

In the same way that you span columns, you can span rows across two or more contiguous cells by using the `rowspan="n"` attribute as shown in Figure 8-39.

	Cell 1.2	Cell 1.3	Cell 1.4
Heading	Cell 2.2	Cell 2.3	Cell 2.4

Figure 9.7.3: Table with spanned rows.

```
<style type="text/css">
 table {border:outset 1px}
 table td {border:inset 1px; padding:3px}
 text-align:center;
 background-color:#F0F0F0}
</style>

<table>
<tr>
 <td id="HEAD" rowspan="2" >Heading</td>
 <td>Cell 1.2</td>
 <td>Cell 1.3</td>
 <td>Cell 1.4</td>
</tr>
<tr>
 <td>Cell 2.2</td>
 <td>Cell 2.3</td>
 <td>Cell 2.4</td>
</tr>
</table>
```

Figure 9.7.4: Code to span table rows.

Note that the first cell in the first row spans two rows. As a result, there is one less cell to specify in the second row. The first row has *four* `<td>` tags. The second row needs only three `<td>` tags since the first cell in that row is encompassed by the previously spanned row.

### Spanning Rows and Columns - TOP

You can span cells across rows and columns to produce interesting and useful table structures. The coding is not that complex if you follow closely across each row, and determine whether you need to code a `<td>` tag for each cell position or whether you

can leave out the `<td>` tag because the cell has already been spanned. In the following code, the `<td>` tags that do not have to be coded are displayed in brackets [ ]. These cells are accounted for by other cells that have spanned across their table positions.

Cell 1.1		Cell 1.3	Cell 1.4
Cell 2.1	Cell 2.2	Cell 2.3	
Cell 3.1		Cell 3.3	

Figure 9.7.5: Table with spanned rows and columns.

```
<style type="text/css">
 table {border:outset 1px}
 table td {border:inset 1px; padding:3px}
</style>

<table>
<tr>
 <td colspan="2">Cell 1.1</td>
 [<td>Cell 1.2</td>]
 <td>Cell 1.3</td>
 <td rowspan="2">Cell 1.4</td>
</tr>
<tr>
 <td>Cell 2.1</td>
 <td rowspan="2">Cell 2.2</td>
 <td>Cell 2.3</td>
 [<td>Cell 2.4</td>]
</tr>
<tr>
 <td>Cell 3.1</td>
 [<td>Cell 3.2</td>]
 <td colspan="2">Cell 3.3</td>
 [<td>Cell 3.4</td>]
</tr>
</table>
```

Figure 9.7.6: Code to span table rows and columns.

#### Listing 8-35.

#### A Spanning Example

The output below is an example of using row and column spanning to produce a complex table structure.

Quantity and Value of Installed Software

Software	Department A		Department B	
	Copies	\$ Value	Copies	\$ Value
Word Processors	10	700.00	15	1,050.00
Spreadsheets	12	780.00	18	1,170.00
Databases	7	595.00	9	765.00
Total Value	\$ 2,075.00		\$ 2,985.00	

Figure 9.7.8: Complex table with spanned rows and columns.

## Determining Cell Spanning

Before you begin coding, it is a good idea to visualize the row and column spanning that needs to take place. Figure 9.7.9 shows the table with vertical arrows indicating row spanning, and horizontal arrows indicating column spanning. Once this structure is visualized, it becomes relatively easy to produce the code to describe the table.

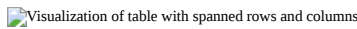


Figure 9.7.9: Visualization of table with spanned rows and columns.

## Coding the Table Structure

The following code begins the table description with a caption followed by the coding for the first row of the table. At this point, setting up the style sheet is not a consideration. Focus needs to be on the structure of the table and required column and row spanning.

```
<table>
<caption>Quantity and Value of Installed Software</caption>
<tr>
 <td rowspan="2">Software</td>
 <td colspan="2">Department A</td>
 <td colspan="2">Department B</td>
</tr>
```

Figure 9.7.10: Code for column header spanning.

Beginning at the top-left of the table in Figure (\PageIndex{9}\), notice that the first cell spans two rows. Thus, this top-left cell is coded with `rowspan="2"` to combine it with the cell beneath to produce a single, combined cell. Furthermore, note that this cell contains no content, only the pair of `<td></td>` tags. By leaving the cell empty, cell borders are not displayed.

Moving from left to right across this first row, notice that the next cell, containing Department A, is not a single cell. It spans two horizontal cells. Therefore, the `<td>` tag contains `colspan="2"` to combine this cell with the one in the next column.

Moving to the right, you come to the cell containing the heading Department B. Again, this is not a single cell, but a spanned cell that crosses two columns. This `<td>` tag, like the previous one, contains the `colspan="2"` attribute.

You have now reached the end of the first row of the table. It is important to realize that, although you have coded only three cell descriptions, you have accounted for a total of five cells: the total number of columns in the full table. Always keep in mind this fact that you must describe as many rows and as many columns as there are in the full dimensions of the table.

Continuing with the coding for the second row of the table, this row contains the Copies and \$ Value subheadings for their respective columns.

```
<tr>
 <td>Copies</td>
 <td>$ Value</td>
 <td>Copies</td>
 <td>$ Value</td>
</tr>
```

Figure 9.7.11: Code for second row of table.

Again, begin at the left of the table and move to the right across the row. The first thing to notice is that the first cell in this row does not need to be coded. It has already been accounted for by the row spanning that took place for the cell above. Now move to the right to the next cell along the row. The next four cells are, in fact, regular non-spanned cells that are each described by a `<td>` tag. You come to the end of the second row having coded four cells, but still accounting for a total of five columns.

Coding for the third through fifth row of the table is added below.

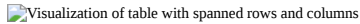
```
<tr>
 <td>Word Processors</td>
 <td>10</td>
 <td>700.00</td>
 <td>15</td>
 <td>1,050.00</td>
</tr>
<tr>
 <td>Word Processors</td>
 <td>12</td>
 <td>780.00</td>
 <td>18</td>
 <td>1,170.00</td>
</tr>
<tr>
 <td>Word Processors</td>
 <td>7</td>
 <td>595.00</td>
 <td>9</td>
 <td>765.00</td>
</tr>
```

Figure 9.7.12: Code for third through fifth row of table.

Here, there are five separate cells containing a row heading and the main information content of the table. There are no spanned cells which means that there are five `<td>` tags across the row. This same coding format can be followed for the next two rows

of the table.

The final row of the table is a summary line presenting totals of the dollar amounts in the body of the table. This row is coded below.

 Visualization of table with spanned rows and columns

```
<tr>
 <td>Total Value</td>
 <td colspan="2">$ 2,075.00</td>
 <td colspan="2">$ 2,985.00</td>
</tr>
</table>
```

Figure 9.7.13: Code for final row of table.

**Listing 8-39.** Code for final row of table.

Again, trace across the row beginning with the first cell. This is a standard cell containing the text Total Value. Next, you come to a cell that is combined with the following cell and requiring the `colspan="2"` attribute. Finally, you encounter another cell that is combined with its following cell. You have coded three cell specification, but again have accounted for a total of five cells across the row.

### Viewing the Table Structure

When the above coding is completed, the overall structure of the table and the cell contents are finished. This unstyled table is shown in Figure 8-43. Borders have been added to help visualize the cells. The issue at this point is whether you have correctly produced the table structure, ignoring for the moment any styling of the cells.

Quantity and Value of Installed Software				
Software	Department A		Department B	
	Copies	\$ Value	Copies	\$ Value
Word Processors	10	700.00	15	1,050.00
Spreadsheets	12	780.00	18	1,170.00
Databases	7	595.00	9	765.00
Total Value	\$ 2,075.00		\$ 2,985.00	

Figure 9.7.14: Basic coding of table with spanned rows and columns.

### Styling the Table

Now, it is a matter of adding a style sheet to decorate the table. In this example, shown in Listing 8-40, style classes are used to assign header, row, and footer styling. Class `.HEAD` is applied to the column headings, setting bold and centered text with a background color. Class `.ROW` formats each data row with right-aligned text, and class `.LABEL` overrides this row styling for the first cell in the row by aligning text labels to the left. Class `.TOTAL` formats the footer row with bold and right-aligned text with a background color.

```
<style type="text/css">
 table {border:outset 1px}
 table td {border:inset 1px; padding:5px}
 table caption {font:bold 12pt}
 .HEAD {font-weight:bold; text-align:center; vertical-align:middle;
 background-color:#F0F0F0}
 .ROW {text-align:right}
 .LABEL {text-align:left}
 .TOTAL {text-align:right; font-weight:bold;
 background-color:#F0F0F0}
</style>

<table>
<caption>Quantity and Value of Installed Software</caption>
<tr class="HEAD" >
 <td rowspan="2">Software</td>
 <td colspan="2">Department A</td>
 <td colspan="2">Department B</td>
</tr>
<tr class="HEAD" >
 <td>Copies</td>
 <td>$ Value</td>
 <td>Copies</td>
 <td>$ Value</td>
</tr>
<tr class="ROW" >
 <td class="LABEL" >Word Processors</td>
 <td>10</td>
 <td>700.00</td>
 <td>15</td>
 <td>1,050.00</td>
</tr>
<tr class="ROW" >
 <td class="LABEL" >Databases</td>
 <td>7</td>
 <td>595.00</td>
 <td>9</td>
 <td>765.00</td>
</tr>
<tr class="TOTAL" >
 <td>Total Value</td>
 <td colspan="2">$ 2,075.00</td>
 <td colspan="2">$ 2,985.00</td>
</tr>
```

Figure 9.7.15: Final code for table with style sheet attached.

What appeared to be a complex coding exercise turns out to be fairly easily managed by separating structure from styling. First, work systematically across the rows of the table, code all stand-alone cells and code cells that require spanning. Ignore cells that are encompassed by a previous span. Take a look at the row table, and make sure that its structure is correct. Finally, apply a style sheet to format those rows or cells that require styling.

---

9.7: Column and Row Spanning is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 9.8: Table Groupings

When working with large tables with varying cell sizes, alignments, and colors, you can bring a more orderly structure to the table by *grouping* rows. In this fashion, you can set structural and styling specifications for the groups, and have those features propagate automatically across the individual cells contained within the groups.

The table in Figure 9.8.1 makes use of the special table tags `<thead>` , `<tbody>` , and `<tfoot>` to classify and organize the rows of the table.

Sales Order

Your Name Your Address City, ST 55555				
No.	Description	Quantity	Price	Amount
11111	Microsoft Windows XP Professional	2	169.99	339.98
22222	Microsoft Office XP Professional	2	449.99	999.98
33333	Adobe Photoshop 7.0	1	579.95	579.95
44444	HP PhotoSmart 7550 Printer	1	299.99	299.99
55555	USB Device Cable	1	5.49	5.49
Shipping				50.00
Sales Tax				155.77
Total				\$ 2,431.16

Figure 9.8.1: Table with row groupings.

It is instructive to view the general layout of this table and its component parts in outline form. This view is produced below, and identifies the various grouped parts of the table as well as the major tags used to structure it.

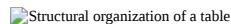


Figure 9.8.2: Structural organization of a table.

There are three main structural parts to a table: the "table head," "table body," and "table foot," which are set apart within associated `<thead>` , `<tbody>` , and `<tfoot>` tags. The outline XHTML coding for these parts is shown below.

```
<table>
<caption>Sales Order</caption>

<thead>
 XHTML for table head
</thead>

<tfoot>
 XHTML for table foot
</tfoot>

<tbody>
 XHTML for table body
</tbody>

</table>
```

Figure 9.8.3: Code outline of head, body, and foot of a table.

When arranging the code for these sections, both the `<thead>` and `<tfoot>` tags must appear *before* the `<tbody>` tag.

### The `<thead>` Tag

The `<thead>` section contains information that appears one time at the beginning of the table. It is coded very much like a normal set of headings. In this example, there are two rows comprising the heading. The first row contains name and address information that is placed in a single cell that spans the width of the table. The second row is a set of column headings.

```
<thead>
<tr>
 <td colspan="5">
 Your Name

 Your Address

 City, ST 55555

 </td>
</tr>
<tr>
 <td>No.</td>
 <td>Description</td>
 <td>Quantity</td>
 <td>Price</td>
 <td>Amount</td>
</tr>
</thead>
```

Figure 9.8.4: Code for `<thead>` section of table.

## The `<tfoot>` Tag

The table footer appears one time at the bottom of a table. It is identified with a `<tfoot>` tag. The present table contains three rows in the footer. All rows have a spanned label area and a single cell for data.

```
<tfoot>
 <tr>
 <td colspan="4">Shipping</td>
 <td>50.00</td>
 </tr>
 <tr>
 <td colspan="4">Sales Tax</td>
 <td>155.77</td>
 </tr>
 <tr>
 <td colspan="4">Total</td>
 <td>$ 2,431.16</td>
 </tr>
</tfoot>
```

Figure 9.8.5: Code for `<tfoot>` section of table.

## The `<tbody>` Tag

The table body displays the main content of the table. It appears in the `<tbody>` section, and contains as many rows as there are data items to report. In the present table, data appear in five separate cells across the row.

```
<tbody>
<tr>
 <td>11111</td>
 <td>Microsoft Windows XP Professional</td>
 <td>2</td>
 <td>169.99</td>
 <td>339.98</td>
</tr>
<tr>
 <td>22222</td>
 <td>Microsoft Office XP Professional</td>
 <td>2</td>
 <td>449.99</td>
 <td>999.98</td>
</tr>
<tr>
 <td>33333</td>
 <td>Adobe Photoshop 7.0</td>
 <td>1</td>
 <td>579.95</td>
 <td>579.95</td>
</tr>
<tr>
 <td>44444</td>
 <td>HP PhotoSmart 7550 Printer</td>
 <td>1</td>
 <td>299.99</td>
 <td>299.99</td>
</tr>
<tr>
 <td>55555</td>
 <td>USB Device Cable</td>
 <td>1</td>
 <td>5.49</td>
 <td>5.49</td>
</tr>
</tbody>
```

Figure 9.8.6: Code for `<tbody>` section of table.

After coding the three sections of the table, you might wish to view the output to make sure the table is properly structured before applying styles to it. Figure 9.8.6 shows the current structure of the example table with borders added to make the cells visible.

Sales Order

Your Name				
Your Address				
City, ST 55555				
No.	Description	Quantity	Price	Amount
11111	Microsoft Windows XP Professional	2	169.99	339.98
22222	Microsoft Office XP Professional	2	449.99	999.98
33333	Adobe Photoshop 7.0	1	579.95	579.95
44444	HP PhotoSmart 7550 Printer	1	299.99	299.99
55555	USB Device Cable	1	5.49	5.49
Shipping				50.00
Sales Tax				155.77
<b>Total</b>				<b>\$ 2,431.16</b>

Figure 9.8.7: Organization of unstyled table.

## Applying Header Styles

Once the column groups and individual columns are styled, additional special styling may be needed for the `<thead>` section of a table. This is the case in the current table where the address lines (the cell in the row with `id="ADDR"`) should be aligned left, and the column headings (the cells in the row with `id="HEAD"`) should be centered, bold, and with background and text colors. This additional styling is added to the style sheet in Listing 8-48 and applied in Figure 9.8.9.

```
<style type="text/css">
 table {border:outset 1px; border-collapse:collapse}
 table caption {font:bold 14pt; color:#707070}
 table td {border:inset 1px; padding:3px}

 table tr#ADDR td {text-align:left; background-color:#FFFFFF}
 table tr#HEAD td {font-weight:bold; text-align:center;
 background-color:#A0A0A0; color:#FFFFFF}
</style>
```

Figure 9.8.8: Additional style sheet entries for `<thead>` styling of example table.

Sales Order

Your Name				
Your Address				
City, ST 55555				
No.	Description	Quantity	Price	Amount
11111	Microsoft Windows XP Professional	2	169.99	339.98
22222	Microsoft Office XP Professional	2	449.99	999.98
33333	Adobe Photoshop 7.0	1	579.95	579.95
44444	HP PhotoSmart 7550 Printer	1	299.99	299.99
55555	USB Device Cable	1	5.49	5.49
Shipping				50.00
Sales Tax				155.77
<b>Total</b>				<b>\$ 2,431.16</b>

Figure 9.8.9: Table with `<thead>` styling applied.

## Applying Footer Styles

The table footer is styled in the same general fashion as the header. In this case, the "Shipping," "Sales Tax," and "Total" labels should be right aligned, and the total line should display in bold with background and text colors. These additions are made in Listing 9.8.10 and shown in Figure 9.8.11.

```
<style type="text/css">
 table {border:outset 1px; border-collapse:collapse}
 table caption {font:bold 14pt; color:#707070}
 table td {border:inset 1px; padding:3px}

 table tr#ADDR td {text-align:left; background-color:#FFFFFF}
 table tr#HEAD td {font-weight:bold; text-align:center;
 background-color:#A0A0A0; color:#FFFFFF}

 table tfoot td {text-align:right}
 table tfoot tr#TOTAL {font-weight:bold;
 background-color:#A0A0A0; color:#FFFFFF}
</style>
```

Figure 9.8.10: Additional style sheet entries for `<tfoot>` styling of example table.

All cells in the footer section (table tfoot td) should be right aligned. The footer row identified by id="TOTAL" (table tfoot tr#TOTAL) should be bold with background and text colors. Again, if all rows and cells in the `<tfoot>` section are styled the same, then that styling can be applied through the single tfoot selector.

#### Sales Order

Your Name				
Your Address				
City, ST 55555				
No.	Description	Quantity	Price	Amount
11111	Microsoft Windows XP Professional	2	169.99	339.98
22222	Microsoft Office XP Professional	2	449.99	999.98
33333	Adobe Photoshop 7.0	1	579.95	579.95
44444	HP PhotoSmart 7550 Printer	1	299.99	299.99
55555	USB Device Cable	1	5.49	5.49
			Shipping	50.00
			Sales Tax	155.77
			<b>Total</b>	<b>\$ 2,431.16</b>

Figure 9.8.11: Table with `<tfoot>` styling applied.

Note that when you use table row groups, the `<thead>` and `<tfoot>` sections must be coded before the `<tbody>` section to pass W3C XHTML 1.1 validation.

9.8: Table Groupings is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## CHAPTER OVERVIEW

### 10: Playing Multimedia

10.1: Audio and Video Formats

10.2: Playing Audio

10.3: Playing Video

10.4: Third Party Content

---

10: Playing Multimedia is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 10.1: Audio and Video Formats

---

10.1: Audio and Video Formats is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 10.2: Playing Audio

---

10.2: Playing Audio is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 10.3: Playing Video

---

10.3: Playing Video is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 10.4: Third Party Content

---

10.4: Third Party Content is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## CHAPTER OVERVIEW

### 11: Creating Forms

- [11.1: Form Controls](#)
- [11.2: Creating Forms](#)
- [11.3: Textbox Controls](#)
- [11.4: Textarea Control](#)
- [11.5: Radio Buttons](#)
- [11.6: Checkboxes](#)
- [11.7: Drop-Down Lists](#)
- [11.8: Submit and Reset Buttons](#)
- [11.9: Other Buttons](#)
- [11.10: Grouping and Labels](#)
- [11.11: Submitting Forms](#)
- [11.12: Forms and CSS](#)

---

[11: Creating Forms](#) is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.1: Form Controls

---

11.1: Form Controls is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.2: Creating Forms

---

11.2: Creating Forms is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.3: Textbox Controls

---

11.3: Textbox Controls is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.4: Textarea Control

---

11.4: Textarea Control is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.5: Radio Buttons

---

11.5: Radio Buttons is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.6: Checkboxes

---

11.6: Checkboxes is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.7: Drop-Down Lists

---

11.7: Drop-Down Lists is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.8: Submit and Reset Buttons

---

11.8: Submit and Reset Buttons is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.9: Other Buttons

---

11.9: Other Buttons is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.10: Grouping and Labels

---

11.10: Grouping and Labels is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.11: Submitting Forms

---

11.11: Submitting Forms is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 11.12: Forms and CSS

---

11.12: Forms and CSS is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## CHAPTER OVERVIEW

### 12: HTML/CSS Appendix

12.1: New HTML5 Tags

12.2: HTML Reference

12.3: CSS Reference

12.4: HTML Special Characters

12.5: HTML Colors

---

12: HTML/CSS Appendix is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 12.1: New HTML5 Tags

---

12.1: New HTML5 Tags is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 12.2: HTML Reference

---

12.2: HTML Reference is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 12.3: CSS Reference

---

12.3: CSS Reference is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 12.4: HTML Special Characters

---

12.4: HTML Special Characters is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

## 12.5: HTML Colors

---

12.5: HTML Colors is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.



# Glossary

**Sample Word 1** | Sample Definition 1

---

## Detailed Licensing

---

### Overview

**Title:** [Web Site Design](#)

**Webpages:** 97

**All licenses found:**

- [Undeclared](#): 100% (97 pages)

### By Page

- [Web Site Design - Undeclared](#)
  - [Front Matter - Undeclared](#)
    - [TitlePage - Undeclared](#)
    - [InfoPage - Undeclared](#)
    - [Table of Contents - Undeclared](#)
    - [Licensing - Undeclared](#)
  - [1: Creating Web Pages - Undeclared](#)
    - [1.1: Internet History and Usage - Undeclared](#)
    - [1.2: Serving Web Pages - Undeclared](#)
    - [1.3: HTML Tags and CSS Styles - Undeclared](#)
    - [1.4: HTML Tags and CSS Styles - Undeclared](#)
    - [1.5: Creating Your First Web Page - Undeclared](#)
  - [2: Web Site Design - Undeclared](#)
    - [2.1: Goal identification - Undeclared](#)
    - [2.2: Scope definition - Undeclared](#)
    - [2.3: Sitemap and wireframe - Undeclared](#)
    - [2.4: Content Creation - Undeclared](#)
    - [2.5: Visual elements - Undeclared](#)
    - [2.6: Testing - Undeclared](#)
    - [2.7: Web Site Launch - Undeclared](#)
  - [3: Basic Document Layout - Undeclared](#)
    - [3.1: Structuring Page Content - Undeclared](#)
    - [3.2: Document Structure Tags - Undeclared](#)
    - [3.3: Paragraphs - Undeclared](#)
    - [3.4: Line Breaks - Undeclared](#)
    - [3.5: Headings - Undeclared](#)
    - [3.6: Horizontal Rules - Undeclared](#)
    - [3.7: List Structures - Undeclared](#)
    - [3.8: Linking Pages - Undeclared](#)
    - [3.9: Displaying Pictures - Undeclared](#)
  - [4: Basic Document Styling - Undeclared](#)
    - [4.1: Applying Styles - Undeclared](#)
    - [4.2: Applying Styles - Undeclared](#)
    - [4.3: Alignment Styles - Undeclared](#)
    - [4.4: Width and Height Styles - Undeclared](#)
    - [4.5: Basic Color Styles - Undeclared](#)
    - [4.6: Basic Color Styles - Undeclared](#)
    - [4.7: ID and Class Selectors - Undeclared](#)
  - [5: Text Formatting - Undeclared](#)
    - [5.1: Font Settings - Undeclared](#)
    - [5.2: Text Properties - Undeclared](#)
    - [5.3: Font Style Tags - Undeclared](#)
    - [5.4: Text and Background Colors - Undeclared](#)
    - [5.5: Special Characters - Undeclared](#)
  - [6: Graphic Images - Undeclared](#)
    - [6.1: Graphic Formats - Undeclared](#)
    - [6.2: Displaying Images - Undeclared](#)
    - [6.3: Background Images - Undeclared](#)
    - [6.4: Image Transparency - Undeclared](#)
  - [7: Applying Special Styles - Undeclared](#)
    - [7.1: Styling Lists - Undeclared](#)
    - [7.2: Borders and Padding - Undeclared](#)
    - [7.3: Sizing Elements - Undeclared](#)
    - [7.4: Positioning Elements - Undeclared](#)
  - [8: Linking Pages - Undeclared](#)
    - [8.1: Text Links - Undeclared](#)
    - [8.2: Graphic Links - Undeclared](#)
    - [8.3: Image Maps - Undeclared](#)
    - [8.4: Redirecting Pages - Undeclared](#)
    - [8.5: Traversing Folders - Undeclared](#)
  - [9: Using Tables - Undeclared](#)
    - [9.1: Creating Tables - Undeclared](#)
    - [9.2: Table Borders - Undeclared](#)
    - [9.3: Table Captioning - Undeclared](#)
    - [9.4: Table Colors and Backgrounds - Undeclared](#)
    - [9.5: Table Size and Alignment - Undeclared](#)
    - [9.6: Aligning Text and Cells - Undeclared](#)
    - [9.7: Column and Row Spanning - Undeclared](#)
    - [9.8: Table Groupings - Undeclared](#)
  - [10: Playing Multimedia - Undeclared](#)
    - [10.1: Audio and Video Formats - Undeclared](#)
    - [10.2: Playing Audio - Undeclared](#)
    - [10.3: Playing Video - Undeclared](#)

- 10.4: Third Party Content - *Undeclared*
- 11: Creating Forms - *Undeclared*
  - 11.1: Form Controls - *Undeclared*
  - 11.2: Creating Forms - *Undeclared*
  - 11.3: Textbox Controls - *Undeclared*
  - 11.4: Textarea Control - *Undeclared*
  - 11.5: Radio Buttons - *Undeclared*
  - 11.6: Checkboxes - *Undeclared*
  - 11.7: Drop-Down Lists - *Undeclared*
  - 11.8: Submit and Reset Buttons - *Undeclared*
  - 11.9: Other Buttons - *Undeclared*
  - 11.10: Grouping and Labels - *Undeclared*
  - 11.11: Submitting Forms - *Undeclared*
  - 11.12: Forms and CSS - *Undeclared*
- 12: HTML/CSS Appendix - *Undeclared*
  - 12.1: New HTML5 Tags - *Undeclared*
  - 12.2: HTML Reference - *Undeclared*
  - 12.3: CSS Reference - *Undeclared*
  - 12.4: HTML Special Characters - *Undeclared*
  - 12.5: HTML Colors - *Undeclared*
- Back Matter - *Undeclared*
  - Index - *Undeclared*
  - Glossary - *Undeclared*
  - Detailed Licensing - *Undeclared*