* *HTML* stands for Hyper Text Markup Language. It is used to give websites structure with text, links, images, and other fundamental elements.
* *CSS* stands for Cascading Style Sheets. It is used to change the appearance of HTML elements.

A markup language is a computer language that defines the structure and presentation of raw text. Markup languages work by surrounding raw text with information the computer can interpret, "marking it up" for processing.

HyperText is text displayed on a computer or device that provides access to other text through links, also known as “hyperlinks”.

There are a few important things to keep in mind when you save it the file:

1. Use the .html HTML file extension, i.e. about\_me.html
2. Don't use any spaces or special characters in the file name. Use underscores (\_) or dashes (-) instead.
3. Decide where in your computer you will save the file, and make sure to remember the location!

Now you're ready to view your new page in your browser!

<h1>You're Building a Website!</h1>

1. All HTML elements begin with an *opening tag*. In this case, the opening tag is <h1>.
2. Most elements require a *closing tag*, denoted by a /. In this case, the closing tag is </h1>.
3. The website user only sees the content between the opening and closing tags.

There are six heading elements: h1, h2, h3, h4,h5, and h6.

h1 is the largest heading and h6 is the smallest.

The webpage now has a heading and a tagline. Next, we will add a description of the company.

p: used for non-heading text, such as the bodies of articles or company descriptions.

<p>Paragraph text here</p>

In the web browser, notice how text enclosed by the p tags is smaller than heading text and is not bold.

Paragraphs are great for expanding the amount of content (text) on your web page. if multiple paragraphs on your web page each contain large amounts of text, your web page could become difficult to consume.

<a href="http://google.com" > Click here for Google!</a>

Anchor elements use an attribute to link users to websites.

Attributes customize the behavior or appearance of HTML elements.

“Click here for Google!” will be underlined, and point to the destined website.

Attributes are made up of the following two parts:

1. The *name* of the attribute.
2. The *value* of the attribute.

For anchor elements, the name of the attribute is href and its value must be set to the URL of the page you'd like the user to visit.

Any valid URL can be used for the value of the href attribute.

The URL must be enclosed with quotation marks.

Text between the <a> and </a> tags can be as few or as many words as you would like.

For a link to open in a new window, the target attribute requires a value of \_blank. The target attribute can be added directly to the opening tag of the anchor element, just like the href attribute.  
<a href="https://en.wikipedia.org/wiki/Brown\_bear" target="\_blank">The Brown Bear</a>

<img src="https://s3.amazonaws.com/codecademy-content/projects/make-a-website/lesson-1/bikes1.jpg"/>

Just like websites have URLs, images on the web also have URLs. Image URLs typically end with the .jpg or .png file extension. The src attribute sets the source for an image element.

Image elements are self-closing, which means they do not need a closing tag.

<alt>, or alternative, text is provided as an attribute to the <img> tag to describe the image to the screen reader.

<img src="#" alt="A field of yellow sunflowers" />

The alt attributes also serves the following purposes:

1. If an image fails to load on a web page, a user can mouse over the area originally intended for the image and read a brief description of the image. This is made possible by the description you provide in the alt attribute.
2. Visually impaired users often browse the web with the aid of of screen reading software. When you include the alt attribute, the screen reading software can read the image's description outloud to the visually impaired user.

<video width="320" height="240" controls> <source src="video-url.mp4" type="video/mp4"> </video>

1. width and height: Set the size of the screen that displays the video.
2. controls: Adds play, pause and volume control.
3. source src: Sets the URL of the video to play.
4. type: Specifies different video formats.

HTML allows you to turn nearly any element into a link by wrapping that element with an anchor element. With this technique, it's possible to turn images into links by simply wrapping the <img> element with an <a> element.

<a href="https://en.wikipedia.org/wiki/Opuntia" target="\_blank"><img src="#" alt="A red prickly pear fruit"/></a>

In the example above, an image of a prickly pear has been turned into a link by wrapping the outside of the <img> element with an <a> element.

Items in an unordered list are referred to as *list items*. Each item is bulleted, not numbered. For example:

* A list item
* A second list item
* A third list item

The HTML code for the list above:

<ul>

<li>A list item</li>

<li>A second list item</li>

<li>A third list item</li>

</ul>

About unordered lists:

1. ul tags create the unordered list.
2. li tags contain each list item.

## ORDERED LISTS

Ordered lists' items are denoted with numbers.

**Example**

My numbered list

<ol>

<li>First item!</li>

<li>Second item!</li>

<li>Last item!</li>

</ol>

ARIA, also known as WAI-ARIA, stands for Accessible Rich Internet Applications. ARIA defines a variety of markup extensions, usually HTML5 attributes, that can be added to elements to give screen readers more information about the element

we can add the role attribute with value navigation:

<nav role="navigation">

<ul>

<li>Put navigation here</li>

</ul>

</nav>

Now, the screen reader will know that this is a menu and present the options to the user accordingly.

Div elements divide your page by enclosing other elements. These enclosed groups of elements can then be organized, moved and styled independently from one another.

Div elements are often used with the class attribute. Here's an example:

<div class="main"> ... </div>

metadata tags: provide metadata about a webpage. they communicate vital information to the web browser, but are not visible to a webpage visitor.

1. <!DOCTYPE html>: Tells the web browser to expect an HTML document.

[web browser](https://en.wikipedia.org/wiki/Web_browser) must know what language a document is written in before they can process the contents of the document.

This declaration is an instruction. It tells the browser what type of document to expect, along with what version of HTML is being used in the document.

1. <html>...</html>: The root of the HTML document and parent of all other HTML elements on the webpage.

Anything between <html> and </html> will be considered HTML code.

1. <head>...</head>: Enclose other metadata about the site, such as its title.

The <head> element will contain information about the page that isn't displayed directly on the actual web page

1. <title>...</title>: Contains the site's title, which is one way users can find your site through a search engine, like Google.

a web page's title would appear on a browser.

1. <meta charset="utf-8"/>: Tells the web browser which character set to use. In this case, the character set is "utf-8".
2. Code for visible HTML content will be placed inside of the <body> element.

<!DOCTYPE html>

<html>

<head>

<title>Ollie Bike Sharing</title>

<meta charset="utf-8"/>

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

</head>

<body>

… …

</body>

</html>

1. Angle brackets - In HTML, the characters < and >are known as angle brackets.
2. HTML element (or simply, element) - HTML code that lives inside of angle brackets.
3. Opening tag - the first, or opening, HTML tag used to start an HTML element.
4. Closing tag - the second, or closing, HTML tag used to end an HTML element. Closing tags have a forward slash (/) inside of them.

It's important to understand that the formatting of the code in **index.html** will not affect the positioning of the elements within the browser.

This is because the browser ignores *whitespace* present in HTML files.

**Attributes**

Code is a lot more readable when it is organized using IDs, classes, and divs.

**CLASS**

HTML elements can have one or more classes, separated by spaces.

You can style elements using CSS by selecting them with their classes.

Because unique IDs should not be used across multiple HTML elements, they are insufficient for quickly styling elements that should all share a specific style.

**Example**

<div class="big-box yellow-box">This is a big yellow box. </div>

To style elements of the same class, you can use a *class selector* in the stylesheet.

.science {

font-family: Georgia, Times, serif;

color: #A3B4C5;

text-transform: uppercase;

}

Class selectors begin with a period (.) and are immediately followed by the name of the class. In the example above, all elements with a class of science will have their typeface, color, and letter case modified.

The class selector targets all elements of a particular class. It's possible, however, for multiple elements on a web page to share a specific styling, but for one of those elements to differ slightly.

.breaking {

font-family: Georgia, Times, serif;

}

p.breaking {

line-height: 1.3em;

}

The .breaking selector targets *all* elements with a class of breaking. The p.breaking selector targets *only* <p> elements with a class of breaking. This type of selector allows you to be even more specific about a particular element's styling, even when that element must share some styling with other elements.

Unless otherwise specified, the rest of this course will use the element.class selector syntax.

When those same elements must also be differentiated, however, labeling with an additional class is helpful.

<h1 class="book domestic">The Way of the Deep</h1>

<h1 class="book foreign">A Night in the Sky</h1>

.book { font-family: Georgia, serif; }

.domestic { font-color: #0902CC; }

.foreign { font-color: #B097DD; }

To label an HTML element with an additional class, simply include the class within the double quotes, immediately after the first class. HTML elements can be labeled with many classes, but whenever possible, it's best to limit an element to four classes at most.

**ID**

An HTML element can have an id attribute to identify it. id elements should always be unique to that single element, and each element should never have more than one id.

**Example**

<div id="my-box">This is my box! Put your text in some other box.</div>

To style a specific element labeled with an ID, you can use an *ID selector* in the stylesheet.

#botswana {

background-color: #56ABFF;

}

In the example above, the HTML element with an ID of botswana is targeted with the ID selector #botswana. All ID selectors begin with the character: #.

In a previous exercise, you learned how to use a multiple element selector to style multiple elements at once.

h1, p { font-family: Garamond, serif; }

The same syntax can be used to style multiple classes at once.

.first, .last { font-size: 20px; }

**ID**

HTML offers an element that is the backbone of code organization: the *div*, represented by <div> in HTML.

You can think of the div as a box, or container, that groups elements that belong together.

Even when divs are labeled with classes, there will be many other times when an individual element will need to be labeled with a class.

<div class="container">

<h1 class="title">Alice In Wonderland</h1>

<p> ... </p>

</div>

div.container { background-color: rgb(252, 255, 205); font-family: Roboto, Helvetica, sans-serif; }

h1.title { color: #0D1A2F; }

When a div is styled, all elements inside of the div will inherit the styling applied to the div.

**Basic Formatting**

You can easily format text to be bold, italic, or underlined using simple formatting tags.

**Example**

This text is <b>bold</b>, <i>italicized</i>, and <u>underlined</u>.

The <b> and <i> elements simply denote how the text should *look*: text within these tags should appear as bold or italicized, respectively.

<strong> and <em>, however, denote how text should be *understood* and, though they result in the same visual appearance, they affect how the screen reader interprets them: text within these tags are read out with a different voice to indicate the emphasis for each. These tags are known as *semantic tags*.

**Body**

The body <body>…</body> is the container for a page's all content. Comes after the <head> tag, within the overall <html> tag.

**Comments**

HTML comments are sometimes used in code to explain parts of the markup. They are similar to comments in other languages. Users do not see comments in their browser.

**Syntax**

<!-- This is an HTML comment! -->

**Horizontal rules**

This tag creates a black line one pixel thick that runs the all the way across its container. It can be styled to look differently with CSS.

**Example**

This text is divided

<hr>

...from this text!

**Line breaks**

This tag is used in a block of text to force a line break.

**Example**

<p> Some text <br/> that spans two lines </p>

# Links

Link elements are used to connect your document to a related resource (very different from hyperlinks, which take you to another webpage when you click on them). Links appear only in the head section of a document so they do not alter the content, but only the presentation. Links are most commonly used to connect to a stylesheet, script, favicon, or alternate format of the page such as an RSS feed or PDF.

**Example**

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

# Semantic formatting

These tags are similar to the previously mentioned formatting tags which have fallen out of favor. The difference is that these tags have semantic value (meaning).<em> is used for something that you wish to emphasize and <strong> is used for something that is important. With both of these elements, you can convey the level of emphasis or importance with nesting. The more times that you nest the element within itself, the higher the magnitude of the text it contains.

**Example**

<p><strong><strong>Warning:</strong>Acid can cause severe burns</strong> </p>

# Tables

An element for displaying information in rows and columns. Supports headers and footers for labeling columns. Divides information into rows (denoted by the <tr> tag) which contain cells (denoted by the <td> tag).

**Example**

<table>

<thead>

<tr>

<th>Item</th> //table heading element, titles

<th>Price</th>

</tr>

</thead>

<tbody>

<tr> //table rows element

<td>Banana</td> //table data element

<td>$56.75</td>

</tr>

<tr>

<td>Yogurt</td>

<td>$12.99</td>

</tr>

</tbody>

<tfoot>

<tr>

<td>Total</td>

<td>$69.74</td>

</tr>

</tfoot>

</table>

|  |  |
| --- | --- |
| Item | Price |
| Banana | $56.75 |
| Yogurt | $12.99 |
| Total | $69.74 |

Long tables can be sectioned off using the *table body* element: <tbody>.

The <tbody> element should contain the all of the table's data, excluding the table headings

When a table's body is sectioned off, however, it also makes sense to section off the table's headings using the <thead> element.

The bottom part of a long table can also be sectioned off using the <tfoot> element.

The blank heading creates the extra table cell necessary to align the table headings correctly over the data they correspond to.

<table>

<tr>

<th></th>

<th scope="col">Saturday</th>

<th scope="col">Sunday</th>

</tr>

<tr>

<th scope="row">Temperature</th>

<td>73</td>

<td>81</td>

</tr>

</table>

Note, also, the use of the scope attribute, which can take one of two values:

1. row - this value makes it clear that the heading is for a row.
2. col - this value makes it clear that the heading is for a column.

table, td { border: 1px solid black; }

The code in the example above uses CSS instead of HTML to show table borders.

Data can span columns using the colspan attribute. The attributes accepts an integer (greater than or equal to 1) to denote the number of columns it spans across.

<table>

<tr>

<th>Monday</th>

<th>Tuesday</th>

<th>Wednesday</th>

</tr>

<tr>

<td colspan="2">Out of Town</td>

<td>Back in Town</td>

</tr>

</table>

In the example above, the data Out of Town spans the Monday and Tuesday table headings using the value 2 (two columns). The data Back in Town appear only under the Wednesday heading.

The rowspan attribute is used for data that spans multiple rows (perhaps an event goes on for multiple hours on a certain day). It accepts an integer (greater than or equal to 1) to denote the number of rows it spans across.

<table>

<tr> <!-- Row 1 -->

<th></th>

<th>Saturday</th>

<th>Sunday</th>

</tr>

<tr> <!-- Row 2 -->

<th>Morning</th>

<td rowspan="2">Work</td>

<td rowspan="3">Relax</td>

</tr>

<tr> <!-- Row 3 -->

<th>Afternoon</th>

</tr>

<tr> <!-- Row 4 -->

<th>Evening</th>

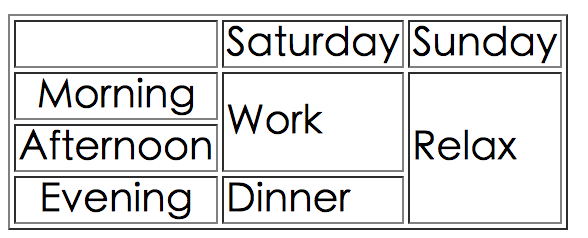
<td>Dinner</td>

</tr>

</table>

In the example above, there are four rows:

1. The first row contains an empty cell and the two column headings.
2. The second row contains the Morning row heading, along with Work, which spans two rows under the Saturday column. The "Relax" entry spans three rows under the Sunday column.
3. The third row only contains the Afternoon row heading.
4. The fourth row only contains the Dinner entry, since "Relax" spans into the cell next to it.



You can use CSS to style tables just like you have done in the past. Specifically, you can change style the various aspects mentioned above.

table, th, td {

border: 1px solid black; font-family: Arial, sans-serif; text-align: center;

}

thead th { //only change head cells in thead element

background: #88CCF1;

color: #FFF;

font-family: 'Lato', sans-serif;

font-size: 16px;

font-weight: 100;

letter-spacing: 2px;

text-transform: uppercase;

}

## FIGURE

Figure elements can be used to display visual content, such as photos, illustrations, diagrams or code snippets.

<figure class="gallery-item">

<img src="image-1.png">

</figure>

<figure class="gallery-item">

<img src="image-2.png">

</figure>

In the example code above, figure elements have the class "gallery-item", and each contains an img element.

## SECTION

Section elements, like divs, can be used to organize webpage content into thematic groups.

<section class="contact-form">

<h2>Contact Us</h2>

<form>

...

</form>

</section>

Above, a section element is used to organize h2 and form elements for a website's "Contact Us" feature.

## NAV

The nav element is used for the part of a website that links to other pages on the site. The links can be organized in a number of ways. Below, the links are displayed within paragraph elements. An unordered list could also be used.

<nav>

<p><a href="login.html">Log In</a></p>

<p><a href="signup.html">Sign Up</a></p>

<p><a href="contact.html">Contact Us</a></p>

</nav>

## FOOTER

The footer element is typically found at the bottom or foot of a webpage. It can contain copyright information, links to social media and additional site navigation items.

<footer>

<p>&copy; Acme Granola Corporation 2016<p>

<div class="social">

<a href="#"><img src="instagram-icon.png"></a>

<a href="#"><img src="facebook-icon.png"></a>

<a href="#"><img src="twitter-icon.png"></a>

</div>

</footer>

Above, between <footer> and </footer>, copyright information is contained in the p element, and social media links are contained within the div with class "social".

Usually, CSS is written in a separate CSS file (with file extension .css) or in a <style> tag inside of the <head> tag, but there is a third place which is also valid. The third place you can write CSS is inside of an HTML tag, using the style attribute. When CSS is written using the style attribute, it's called an "inline style". In general, this is not considered a best practice. However, there are times when inline styles are the right (or only) choice.

<p style="color:blue;font-size:46px;">

I'm a big, blue, <strong>strong</strong> paragraph

</p>

Unlike normal CSS syntax, inline styling does *not* use selectors or curly braces. Don't forget to include the semicolon; after each pair!

The inline style will affect only the <p> it's written in.

HTML is meant for conveying structured information. CSS is built to style that structured information. When inline styles are used, this clear separation between structured information and styling is blurred.

CSS, or Cascading Style Sheets, is a language that web developers use to *style* the HTML content on a web page. If you're interested in modifying colors, font types, font sizes, shadows, images, element positioning, and more, CSS is the tool for the job!

HTML files are meant to contain only HTML code. Similarly, CSS files are meant to contain only CSS code. You can create a CSS file by using the **.css** file name extension, like so: **style.css**

With a CSS file, you can write all the CSS code needed to style a page without having to sacrifice the readability and maintainability of your HTML file.

The HTML *link* element links a CSS file to an HTML file so that CSS styling can be applied. Here's an example of the link element:

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

About *link*:

1. href - like the anchor element, the value of this attribute must be the address, or path, to the CSS file.

If the CSS file is stored in the same [directory](https://en.wikipedia.org/wiki/Directory_(computing)) as your HTML file, then you can specify a [relative path](https://en.wikipedia.org/wiki/Path_(computing)#Absolute_and_relative_paths) instead of a URL

href=”/style.css”

1. type - this attribute describes the type of document that you are linking to (in this case, a CSS file). The value of this attribute should be set to text/css.
2. rel - this attribute describes the relationship between the HTML file and the CSS file. Because you are linking to a stylesheet, the value should be set to stylesheet.

The <style> element allows you to write CSS code between its opening and closing tags. To use the <style> element, it must be placed inside of the head.

<head>

<style>

</style>

</head>

Take a look at **index.html**. Cut the CSS code in between the opening and closing <style> tags and paste it directly in the new file called **style.css**.

Make sure to delete the remaining <style>element (now empty) from **index.html**.



To style an HTML element using CSS, you must first *select* that element in the CSS file. The diagram to the right shows the anatomy of a CSS rule:

1. rule: a list of CSS instructions for how to style a specific HTML element or group of HTML elements.

A CSS rule consists of a CSS selector and the declarations inside of the selector.

1. selector: specifies exactly which HTML elements to style. Here h1 is the selector.
2. properties and values: located inside the { } brackets, properties and values specify what aspect of the selector to style. In the diagram's example, the color property is set to red, which will display all h1 elements in red.

For example, to style a <p> element, the syntax to select it using CSS is:

p { }

In the example above, all paragraph elements are selected using a CSS *selector*. The selector (in this case) is p. Note that the CSS selector essentially matches the HTML tag for that element, but without the angle brackets.

Fortunately, you can select multiple elements at once so that you can save time styling a shared property.

h1, h2, p {

color: Green;

font-size: 18px;

}

\* {

font-family: Arial;

}

In the example above, the universal selector, \*, is used to select every element on the page and set the font to Arial.

One line of spacing should exist between CSS rules. In the example above, there is one line of spacing between the CSS rule for the heading and the CSS rule for the paragraph.

CSS comments begin with /\* and end with \*/, like so:

/\* This is a comment in CSS! \*/

Color can affect the following design aspects:

1. The foreground color
2. The background color

Foreground color is the color that an element appears in. For example, when a heading is styled to appear green, the *foreground color* of the heading has been styled.

Conversely, when a heading is styled so that its background appears yellow, the *background color* of the heading has been styled.

h1 {

color: Red;

background-color: Blue;

}

Notice that the background-color property did not change the background color of the entire page. Instead, only the background area behind heading changed.

CSS comes equipped with 147 named colors, such as red, used above. For many situations, these named colors will suffice. However, web developers who want to get even more exact with their color choices can use hexadecimal and RGB color values.

1. Hexadecimal color (#RRGGBB): Hexadecimal values that represent mixtures of red, green and blue. For example, red can be expressed with the hexadecimal value of #FF0000: the value ff represents red, 00 represents green, and 00 represents blue.

h2 {

color: #AA33BB;

color: #A3B; /\* This is the same color as above \*/

}

1. RGB (Red, Green, Blue) colors: Color created by three numbers representing red, green, and blue. When mixed together, the three values create a specific color. For example: purple can be represented as rgb(128,0,128). Can be transformed from Hex expression.

RGB (Red, Green, Blue) colors offer the option of 16,777,216 possible colors.

h1 {

color: rgb(123, 20, 233);

background-color: E4BB97;

}

RGB values and hex color codes are different ways to represent the same thing: color. It's possible to convert back and forth between RGB values and hex color codes (color pickers often help with this conversion).

There are many resources on the Internet known as "[color pickers](https://color.adobe.com/create/color-wheel/)" that allow you to view the result of different RGB values before you decide to use a certain color.

<https://color.adobe.com/zh/create/color-wheel/>

1. The current revision of CSS, CSS3 (at the time of this writing), introduces a new way to specify colors using *HSL colors*.

HSL stands for **H**ue, **S**aturation, and **L**ightness. Specifically, this is what each means:

* Hue - the technical term that describes what we understand as "color." In HSL, hue is represented on a color wheel. It can take on values between 0 and 360.
* Saturation - the amount of gray in a given color. In HSL, saturation is specified using a percentage between 0% and 100%. The percentage 0% represents a shade of gray, whereas 100% represents full saturation.
* Lightness - the amount of white in a given color. Similar to saturation, lightness is specified using a percentage between 0% and 100%. The percentage 0% represents black, whereas 100% represents white. 50% is normal.

You can use HSL colors in your CSS like this:

h1 { color: hsl(182, 20%, 50%); }

Notice that using HSL is very similar to using RGB.

1. Opacity is a measure of how transparent a color is. To modify opacity in RGB colors, CSS offers the rgba()value. Note the slight difference in rgb() and rgba().

The extra a character in the rgba() value is known as the *alpha value*. It represents the opacity of a color. The alpha value can be a number between 0 or 1, inclusive.

h1 {

color: rgba(123, 88, 9, 0.5);

}

In the example above, the alpha value has been set to 0.5. This indicates that the color of the heading will be set to 50% of its normal opacity.

**Note:** The alpha value can also be used for HSL colors, using hsla():

h1 {

color: hsla(239, 45%, 22%, 0.4);

}

Specifically, we can add multiple CSS color declarations, just in case a user's browser can't support a certain declaration.

h1 {

color: rgb(22, 34, 88);

color: rgba(22, 34, 88, 0.4);

}

In CSS, the latter of multiple declarations takes priority.

1. The vertical spacing between lines of text can be modified with the line-spacing property.
2. The horizontal spacing between words can be modified with the word-spacing property.
3. The spacing between letters, the kernel, can be modified with the letter-spacing property.
4. Text can appear bold with the font-weight property.
5. Text can appear in italics with the font-style property.
6. Text can appear in all uppercase or all lowercase with the text-transform property.
7. Text can be aligned with the text-align property.

In CSS, font is managed using the *font-family* property:

Fallback fonts are included in case a visitor's web browser does not support the first font. Sometimes, more than one fallback font is included.

To use fallback fonts, the following syntax is required:

h1 {

font-family: Garamond, Times, serif;

}

The CSS rule above says: "Use the Garamond font for all <h1> elements on the web page. If that font is not available, use the Times font. If both of those fonts are not available, use any serif font pre-installed on the user's computer." The fonts specified after Garamond are the fallback fonts.

Most computers have a small set of typefaces pre-installed. This small set includes serif fonts and sans-serif fonts, like Times New Roman and Arial, respectively.

1. Serif - the letters in these fonts have extra details on the ends of each letter. Examples include fonts like Times New Roman or Georgia, among others.
2. Sans-Serif - the letters in these fonts do not have extra details on the ends of each letter. Instead, letters have straight, flat edges. Some examples include Arial or Helvetica.

When setting typefaces on a web page, keep the following points in mind:

1. The font specified in a stylesheet must be installed on a user's computer so that font to display when a user visit the web page. We'll learn how to work around this issue in a later exercise.
2. You've probably noticed that we haven't been specifying a typeface in previous exercises of this course. How exactly does the browser know what typeface to use when displaying the web page? The default typeface for all HTML elements is Times New Roman. You may be familiar with this typeface if you have ever used a formatted word processor.
3. It's a good practice to limit the number of typefaces used on a web page to 2 or 3.
4. When the name of a typeface consists of more than one word, it must be enclosed in double quotes (otherwise it will not be recognized), like so:

h1 {

font-family: "Courier New";

}

<!—-in html file :

href = “https://fonts.googleapis.com/css?family=Courier+New” -->

For example, Google offers [Google Fonts](https://fonts.google.com/), a directory of thousands of open-source fonts that are free to use by anyone.

To use a Google Font, you can use a <link> element in HTML file, just like you did for a CSS stylesheet:

<head>

<link href="https://fonts.googleapis.com/css?family=Raleway" type="text/css" rel="stylesheet" > </head>

You can use the new font just as you would use any other font in CSS file:

h1 {

font-family: Raleway, Georgia, serif;

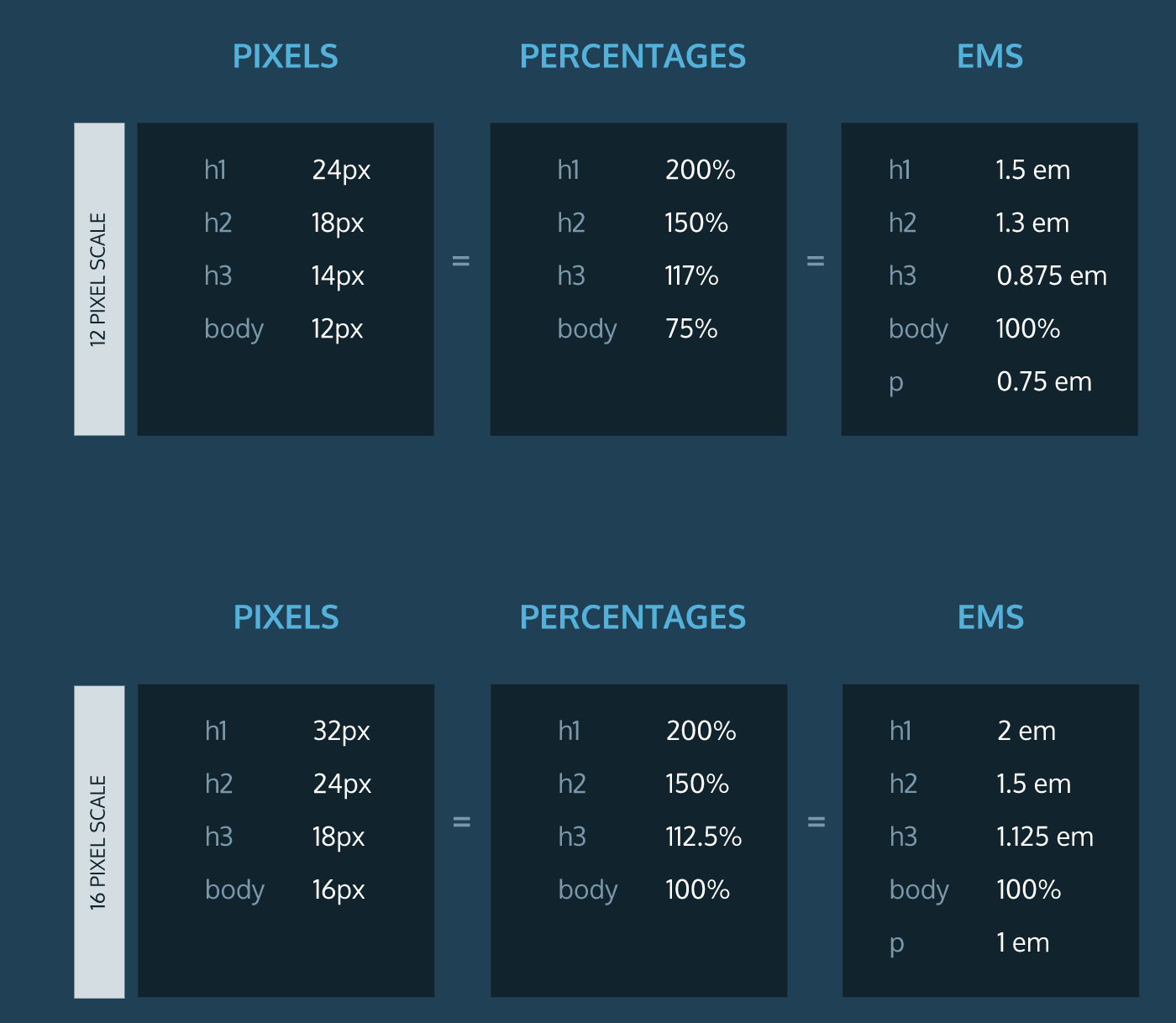
}

To change the size of text on your web page, you can use the font-size property.

p {

font-size: 18px;

}



In CSS, size can be assigned in pixels (px), rems, or ems.

1. pixel (px): Standard unit of measurement for sizing fonts and other HTML elements.

Pixels are used to set the exact size of an element

1. rem: Represents the default font size for the web browser. Rems can be used to ensure that HTML elements scale in proportion to each other on various web browsers and screen sizes. On most web browsers, 1rem is equivalent to 16px, 2rem is equivalent to32px (a doubling), 3rem is equivalent to 48px (a tripling) and so on.
2. em: A relative value that changes in proportion to the size of the parent element. For example, if a parent element has font-size: 20px;, child elements with font-size: 1em; would be equivalent to 20px. Child elements with font-size: 0.5em; would be equivalent to10px (a halving) and so on.

An em is equal to the width of the letter "m".

1. % - Percentages are also a relative unit of measurement. The default size of text in web browsers is 16 pixels, or 16px. When percentages are used, they set the size of text relative to this default size. For example, setting the font size to 200% would be equivalent to setting it to 32px.

you can modify the spacing between lines of text with the line-height property.

p {

line-height: 1.5em;

}

When the line height of an element is modified, you are manipulating the *leading* (pronounced "ledding") of the font. When the line height is increased, the spacing between lines of text increases, which can make text easier to read.

The line height can be modified using pixels or ems, but the unit of ems is preferred, since ems offer a spacing relative to the size of the text on the page.



You can also increase the spacing between words in a body of text, technically known as *word spacing*.

To do so, you can use the word-spacing property:

h1 {

word-spacing: 0.3em;

}

The default amount of space between words is usually 0.25em

The technical term for adjusting the spacing between letters is called "kerning". Kerning can be adjusted with the letter-spacing property in CSS.

h1 {

letter-spacing: 0.3em;

}

Like word spacing, it's not common to increase the kerning in text, but sometimes it enhances the readability of uppercase text.

In CSS, the font-weight property turns bold on or off.

p {

font-weight: bold;

}

In the example above, all paragraphs on the web page would appear bolded.

The font-weight property has a second value: normal. Why does it exist?

If we wanted *all* text on a web page to appear bolded, we could select all text elements and change their font weight to bold. If a certain section of text was required to appear normal, however, we could set the font weight of that particular element to normal, essentially "shutting off" bold for that element.

Earlier, you added the following line of code to **index.html**:

<link href="https://fonts.googleapis.com/css?family=Roboto:100" rel="stylesheet">

Notice the 100 in the URL. This specifies that you'd like to use a font weight of 100 when linking to the Roboto font. It's possible to specify a different font weight, or even multiple within the same URL.

Font weights can be set to 100 (thin), 200, 300, 400 (normal weight), 500, 600, 700, 800, or 900 (bold).

You can also *italicize* words with the font-style property.

h3 { font-style: italic; }

Text can also be styled to appear in either all uppercase or lowercase with the text-transform property.

h1 { text-transform: uppercase; }

Alternatively, the lowercase value could be used to format text in all lowercase.

No matter how much styling is applied to text (typeface, size, weight, etc.), text always appears on the left side of the browser.

To move, or align, text, we can use the text-align property.

h1 { text-align: right; }

The text-align property can be set to one of the following three values:

1. left - aligns text to the left hand side of the browser.
2. center - centers text.
3. right - aligns text to the right hand side of the browser.

Consider the HTML below:

<div class="header">

<h2>Heading</h2>

<p>Paragraph</p>

</div>

Here, the div is the parent element and the h2and p are children. CSS styles applied to the header *class selector* will automatically apply to the h2 and the p.

In CSS, class selectors can be identified by a dot .followed by the class name, as seen below:

.header { color: #ffffff; }

As a result of this code, child elements of divs with the header class will have a font color of #ffffff(white).

.hero { background-image: url("https://s3.amazonaws.com/codecademy-content/projects/make-a-website/lesson-2/bg.jpg");

background-size: cover; }

Here, we have specified that we want the image to completely cover elements with the .hero class.

For example, to style one anchor element differently than all the others on a webpage, you could use the HTML id attribute:

<a id="learn-code" href="https://www.codecademy.com">Click here to learn to code!</a>

Then in the CSS file, you would create a rule for the id selector, using a # symbol:

#learn-code { color: #2e69a3; }

Ids have greater specificity than classes. If an HTML element is using both id and class attributes, the CSS rule for the id will take precedence over that of the class.

Id is defined inside the opening tag.

Properties are defined within selectors by defining a property and a value. They are separated with a colon and delineated with a semi-colon.

**Syntax**

selector {

property: value;

}

**Example**

h1 {

color: blue;

}

## PADDING

The padding is the spacing between the content and the border (edge of the element.). We can adjust this value with CSS to move the border closer to or farther from the content. Here, the div with id 'box' will get 10px of padding all around it.

**Example**

#box {

padding: 10px;

}

## MARGIN

The margin is the space around the element. The larger the margin, the more space between our element and the elements around it. We can adjust the margin to move our HTML elements closer to or farther from each other. Here, the div with id 'box' will get 10px of margin above and below it, and 5px of margin to the left and right.

**Example**

#box {

margin: 10px 5px 10px 5px; //U L D R

}

## ELEMENT SELECTORS

You are able to select HTML elements first by simply using the name of the element.

**Example**

h1 {

color: blue;

}

## ATTRIBUTE SELECTORS

HTML elements are also able to be selected by their attributes.

**Example**

a[href="http://codecademy.com"] {

color: purple;

}

/\* HTML Selected: <a href="http://codecademy.com"> \*/

## CHILD SELECTORS

You can also use multiple selectors to get the exact elements you want, by using parental nesting. By using the "greater-than" symbol (>), you can select only the direct children of an element, going down only one level.

**Example**

ul > li {

display: inline-block

}

/\* Selects only the first-level list items in all unordered lists in the HTML \*/

**Example**

ul a {

text-underline: none;

}

/\* Selects all anchors which have an unordered list their ancestry \*/

**Example**

ul + span {

display: inline;

}

/\* Selects only spans that directly follow an unordered list \*/

**Example**

a ~ h1 {

color: blue;

}

/\* Selects all h1 elements that are in the general vicinity of an anchor \*/

## UNIVERSAL SELECTOR

The universal selector (\*) may be used to select all the elements in a particular range. Be aware that the universal selector is the most performance taxing selector, and should be used sparingly.

**Example**

\* {

background-color: blue;

}

/\* Selects ALL HTML elements in the page \*/

**Example**

body \* {

color: red;

}

/\* Selects ALL children of the body \*/

**Example**

div > \* {

color: red;

}

/\* Selects ALL first-level children of all divs on the page \*/

**Example**

a:hover {

text-decoration: underline;

}

/\* Will underline all links when the user puts their mouse over them \*/

a:active {

font-weight: bold;

}

/\* Will make all links bold while the user is clicking on them. \*/

:active is a psuedo-class selector, which we use to style an element only while it's being clicked.

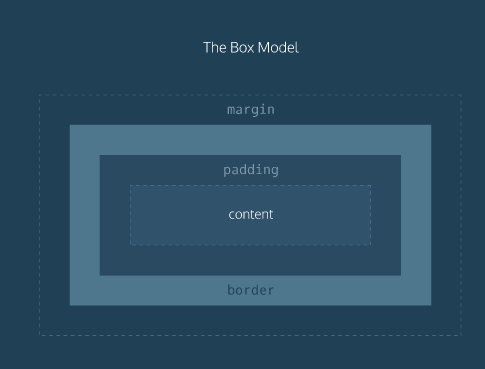
body {

background-image: url("http://image-gallery.io/mountain-scene.png");

background-size: cover;

}

Here, the image covers the entire HTML body element.



An element's box has two dimensions: a *height* and a *width*. In HTML, all boxes have default dimensions.

1. Pixels - You learned about pixels when you learned about fonts. This unit lets you set the exact size of an element's box.
2. Ems - This unit sets the dimensions of the box relative to the size of the text within the box.
3. Percentages - This unit sets the dimensions of the box relative to the size of the box that it is encased in. For example, consider an element (a box) of class blue set to a height of 200 pixels and a width of 200 pixels. Inside of blue, consider another box of class red, set to a height of 37% and a width of 45%. The resulting dimensions of the red box would be a height of 74 pixels and a width of 90 pixels.

Because a web page can be viewed through displays of differing screen size, the content on the web page can suffer from those changes in size. To avoid this problem, CSS offers two properties that can limit how narrow or how wide an element's box can be sized to.

1. min-width - this property ensures a minimum width for an element's box.
2. max-width - this property ensures a maximum width for an element's box.
3. min-height - this property ensures a minimum height for an element's box.
4. max-height - this property ensures a maximum height for an element's box.

The overflow property controls what happens to content when it spills, or *overflows*, outside of its box. It can be set to one of the following values:

1. hidden - when set to this value, any content that overflows be hidden from view.
2. scroll - when set to this value, a scrollbar will be added to the element's box so that the rest of the content can be viewed by scrolling.

margin: The space between the HTML page element and the next nearest element(s).

In CSS, the *border* property's value requires three parts: frame

*A.thickness*: You can control the thickness, or width, of borders with the border-width property, using pixels, ems, or rems.

It's also possible to also set the border-width property to one of the following named thicknesses:

1. thin
2. medium
3. thick

Another version of the border-widthproperty allows you to specify the width for each side of the border.

p { border-style: solid; border-width: 3px 1px 2px 1px; }

The values in the example above refer to the border width in *clockwise* order (top: 3 pixels, right: 1 pixel, bottom: 2 pixels, left: 1 pixel).

If you'd like to be even more specific about the width of different sides of the border, you can use the following properties:

1. border-top-width
2. border-right-width
3. border-bottom-width
4. border-left-width

Each property affects the width of only one side of the border, giving you more flexibility in customization.

p { border-style: solid; border-left-width: 4px; }

*B.type*: Sets the border type.

A border's style can be set with the border-style property. This property can take on one of the following values:

1. solid - border is a solid line.
2. dashed - border is a series of lines or dashes.
3. dotted - border is a series of square dots.
4. double - border is two solid black lines.
5. groove - border is a groove (or carving).
6. inset - border appears to cut into the screen.
7. outset - border appears to pop out of the screen.
8. ridge - border appears as a picture frame.
9. hidden or none - no border.

*C.color*: sets the border's color, using named colors, HEX, or RGB values.

CSS allows you to style all three properties at once with a shorthand property.

a solid black border that is 2 pixels thick:

p { border: 2px solid black; }

It's also possible to set separate margin spacing on each side of an element.

D. The corners of an element's border box can be modified with the border-radius property. 圆角矩形

div.container { border: 3px solid rgb(22, 77, 100); border-radius: 5px; }

The code in the example above will set *all four corners* of the border to a radius of 5 pixels (i.e. the same curvature that a circle with radius 5 pixels would have).

You can create a border that is a perfect circle by setting the radius equal to the height of the box, or to 100%.

div.container { height: 60px; width: 60px; border: 3px solid rgb(22, 77, 100); border-radius: 100%; }

The margin refers to the space directly outside of the box. You can adjust this spacing with the margin property.

In that case, another version of the margin property lets you specify exactly how much margin there should be on each side of the box.

p { margin: 6px 12px 6px 12px; }

In the example above, the four values 6px 12px 6px 12px refer to the amount of margin around the box in a *clockwise* rotation. In order, it specifies the amount of margin on the top (6 pixels), right (12 pixels), bottom (6 pixels), and left (12 pixels) sides of the box.

Additional margin properties:

1. margin-top: Sets the top margin.
2. margin-bottom: Sets the bottom margin.
3. margin-left: Sets the left margin.
4. margin-right: Sets the right margin.

When the margin property is set to auto, the element being styled will center in the page. In order to center an element, a width must be set for that element. Otherwise, the width of the div will be automatically set to the full width of its containing element, like the <body>, for example. It's not possible, therefore, to center an element that takes up the full width of the page.

div.headline { width: 400px; margin: auto; }

In the example above, the width of the div is set to 400 pixels, which is less than the width of the page's body. This will cause the div to center properly on the page.

**Note**: When margin: auto is used, an element will center *relative* to its container. For example, the div in the example above was centered relative to the width of the body. If the div was contained in larger div, the smaller div would center relative to the width of the larger div.

The space between the contents of a box and the borders of a box is known as *padding*. In CSS, you can modify this space with the padding property.

p { border: 3px solid #A2D3F4; padding: 10px; }

The code in the example will put 10 pixels of space between the content of the paragraph (the text) and the box borders, on all four sides.

In that case, another version of the padding property lets you specify exactly how much padding there should be on each side of the content.

p { border: 3px solid #XXXXXX; padding: 5px 10px 5px 10px; }

In the example above, the four values 5px 10px 5px 10px refer to the amount of padding in a *clockwise* rotation. In order, it specifies the amount of padding on the top (5 pixels), right (10 pixels), bottom (5 pixels), and left (10 pixels) sides of the content.

When you're certain that the top and bottom values for padding will equal each other, and that the left and right values for padding will also equal each other, you can use the following shortcut:

p { padding: 5px 10px; }

The first value, 5px, sets a padding value for the top and bottom sides of the content. The second value, 10px sets a padding value the left and right sides of the content.

If you want to be even more specific about the amount of padding on each side of a box's content, you can use the following properties:

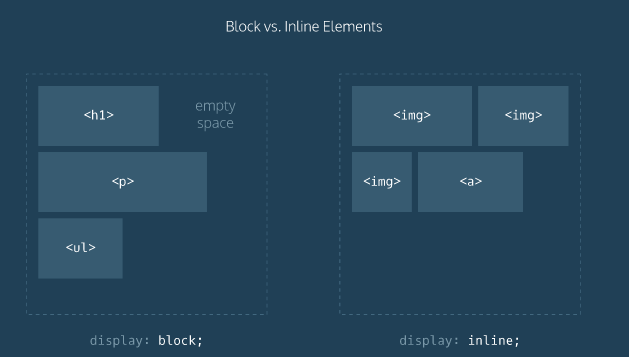
1. padding-top
2. padding-right
3. padding-bottom
4. padding-left

All major web browsers have a default stylesheet they use in the absence of an external stylesheet. These default stylesheets are known as user agent stylesheets. In this case, the term "[user agent](https://en.wikipedia.org/wiki/User_agent)" is a technical term for the browser.

\* { margin: 0; padding: 0; }

The code in the example above resets the default margin and padding values of all HTML elements. It is often the first CSS rule in an external stylesheet.

Note that both properties are both set to 0. When these properties are set to 0, they do not require a unit of measurement.



In the diagram, notice:

1. The two dotted rectangles represent webpages.
2. HTML heading, paragraph, and unordered list elements are block level: each appears on its own line on the webpage.
3. HTML image and anchor elements are displayed inline: they appear on the same line as their neighboring elements on the webpage.

Display types can be overwritten in CSS by using the *display* property.

Modifying the display property of an element can help achieve a desired layout for a web page. The display property can take on one of four values:

1. inline - causes block-level elements (like a div) to behave like an inline element (like a link).
2. block - causes inline elements (like a link) to behave like a block element (like a div).
3. inline-block - causes block-level elements to behave like an inline element, but retain the features of a block-level element.
4. none - removes an element from view. The rest of the web page will act as if the element does not exist.

The visibility property can be set to one of the following values:

1. hidden - hides an element.
2. visible - displays an element.

Keep in mind, however, that users can still view the contents of the list item by viewing the source code in their browser. Furthermore, the web page will *only* hide the contents of the element. It will still leave an empty space where the element is intended to display.

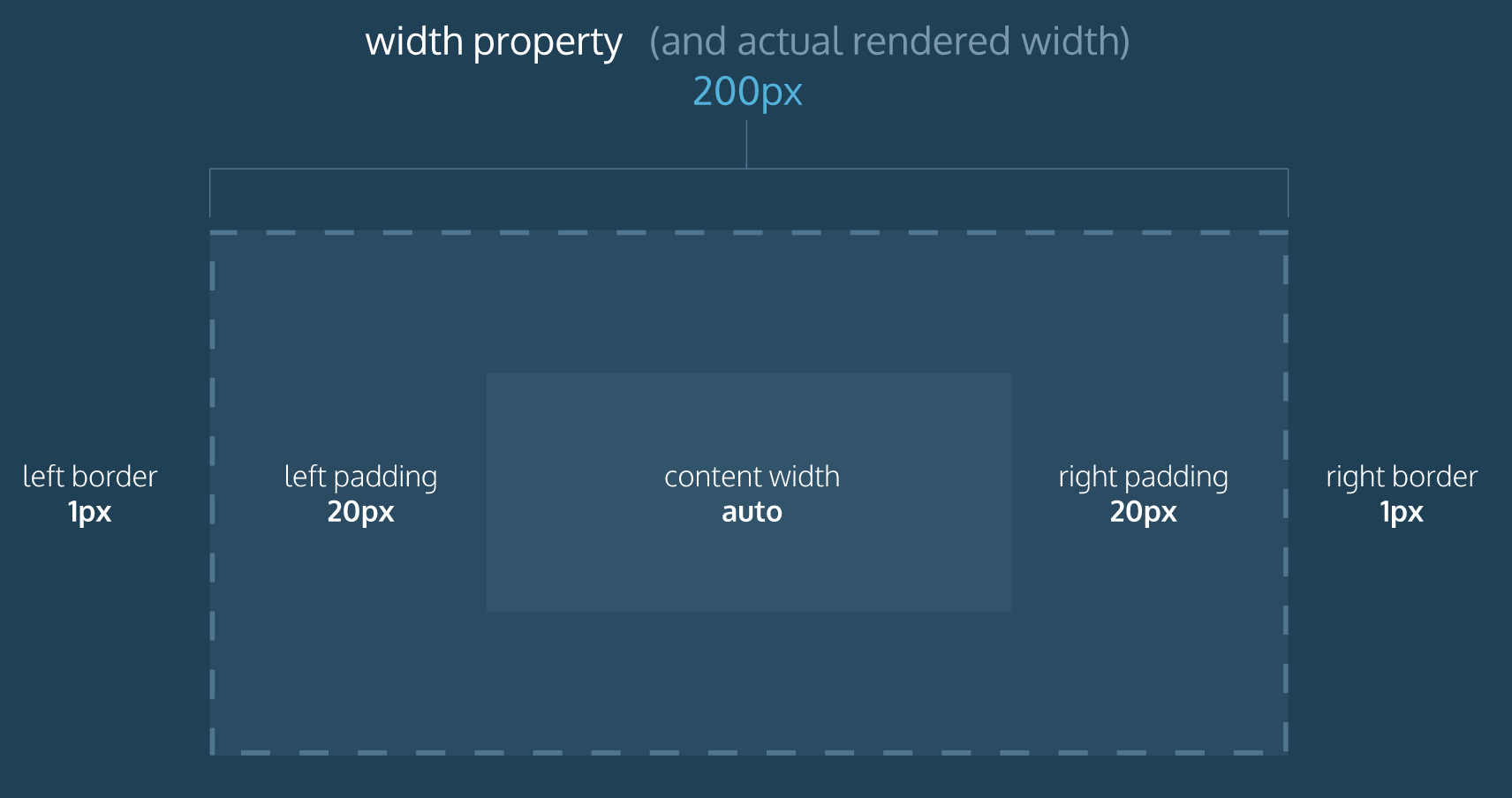
What's the difference between display: none and visibility: hidden? An element with display: none will be completely removed from the web page. An element with visibility: hidden, however, will not be visible on the web page, but the space reserved for it will.

Unfortunately, under the current box model, the border thickness and the padding will affect the dimensions of the box.

1. The box-sizing property controls the box model used by the browser.
2. The default value of the box-sizing property is content-box.
3. The value for the new box model is border-box.

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

In this box model, the height and width of the box will remain fixed. The border thickness and padding will be included inside of the box, which means the overall dimensions of the box do not change.



The default position of an element can be changed by setting its position property. The position property can take one of four values:

1. static - the default value (it does not need to be specified)
2. relative

This value allows you to position an element relative to its default static position on the web page.

.box-bottom { background-color: DeepSkyBlue; position: relative; top: 20px; left: 50px; }

In the example above, the div has been positioned using two of the four *offset properties*. The valid offset properties are:

1. top - moves the element down.
2. bottom - moves the element up.
3. left - moves the element right.
4. right - moves the element left.

In the example above, the div will be moved down 20 pixels and to the right 50 pixels from its default static position.

Units for offset properties can be specified in pixels, ems, or percentages. Note that offset properties will not work if the position of the element is not set to relative.

1. Absolute

When an element's position is set to absolute all other elements on the page will *ignore* the element and act like it is not present on the page. When an element's position is set to absolute, the element will scroll out of view when a user scrolls.

1. fixed

When set to fixed, an element's position can be pinned to any part of the web page. The element will remain in view no matter what, the div will remain fixed to its position no matter where the user scrolls on the page,

The z-index property accepts integer values. Depending on their values, the integers instruct the browser on the order in which elements should be displayed on the web page. Priority

.box-top { background-color: Aquamarine; position: relative; z-index: 2; }

.box-bottom { background-color: DeepSkyBlue; position: absolute; top: 20px; left: 50px; z-index: 1; }

In the example above, we set the .box-top position to relative and the z-index to 2. We changed position to relative, because the z-index property does *not* work on static elements. The z-index of 2 moves the .box-top element forward, because it is greater than the .box-bottom z-index, 1.

Navs are used to organize site navigation menus on a webpage.

By using float, we have the option of floating elements left or right. This works for static and relative positioned elements

The float property can be set to one of two values:

1. left - this value will move, or float, elements as far left as possible.
2. right - this value will move elements as far right as possible.

Consider the example code below. The class selector, .logo, floats left, and the id selector #search-bar floats right:

.logo { float: left; }

#search-bar { float: right; }

The clear property specifies how elements should behave when they bump into each other on the page. It can take on one of the following values:

1. left — the left side of the element will not touch any other element within the same containing element.
2. right — the right side of the element will not touch any other element within the same containing element.
3. both — neither side of the element will touch any other element within the same containing element.
4. none — the element can touch either side.

As with any other element, the dimensions of an image can be set with the height and width properties.

**Note:** Images should be saved at the dimensions they will be displayed in on the web page. Using dimensions for an image that exceed the original dimensions will distort the image.

<img src="#" alt="A red leaf" class="leaf" />  
img.leaf { width: 300px; height: 200px; display: block; margin: 0px auto; }

In the example above, the image is aligned using the margin property. The top and bottom margins of the image's box are set to 0 margin. The left and right margins are set to auto, which automatically sets the exact amount of margin needed on the left and right sides in order to center the image.

**Note:** To align images to the left or right side of a page, you can use the float property you learned about earlier.

Images can also be added to the backgrounds of HTML elements with the background-image property.

body { background-image: url("https://www.example.com/leaf.jpg"); }

You can control how a background image *repeats* with the background-repeat property. This property can take one of four values:

1. repeat - the default value — the image will repeat horizontally and vertically.
2. repeat-x - the background image will be repeated only along the x-axis (horizontally).
3. repeat-y - the background image will be repeated only along the y-axis (vertically).
4. no-repeat - the background image will not be repeated at all and will appear only once.

When a background image is not repeated, its position can be modified with the background-positionproperty.

p { background-image: url("#"); background-repeat: no-repeat; background-position: right center; }

A background image is positioned using a 3 by 3 grid (three rows, three columns), meaning there are 9 total possible positions for the image:

1. left top - top left corner of the element's box.
2. center top - top center of the element's box.
3. right top - top right corner of the element's box.
4. left center - left column, center row.
5. center center - the center of the element's box.
6. right center - right column, center row.
7. left bottom - bottom left corner of the element's box.
8. center bottom - bottom center of the element's box.
9. right bottom - bottom right corner of the element's box.

CSS allows you to set all three properties at once using a shorthand property: background.

p { background: url("#") no-repeat right center; }

Note that the background property includes all of the properties that we previously styled individually: background image, repetition, and position (in that order). It's considered best practice to follow this order of values when setting the background property.

This exercise will focus on two of the most common values of the background-size property:

1. cover - expands the image as large as possible to cover the full width or height of a container. If the dimensions of the container (say, a div) are larger than the dimensions of the image, the image will become distorted. This value is best for images that don't communicate important content to the user, like background images.
2. contain - expands the image as large as possible, but the image will be [letterboxed](https://www.codecademy.com/courses/learn-html-css/lessons/adding-images/exercises/background-size?action=lesson_resume), which means it won't cover the full width or height of a container. (1.33 : 1 || 4 : 3)

The background-attachment property can take one of two values:

1. scroll - this value allows the image to move up and down as a user scrolls on the web page (this is the default value).
2. fixed - this value pins the image's position on the page.

p { background: url("#") no-repeat right center; background-attachment: fixed; }

The background-image property can be set to the following value:

1. -webkit-linear-gradient() - this value accepts two arguments: the two colors the linear gradient will transition to and from. The colors are usually specified as hex color codes. 垂直渐变色

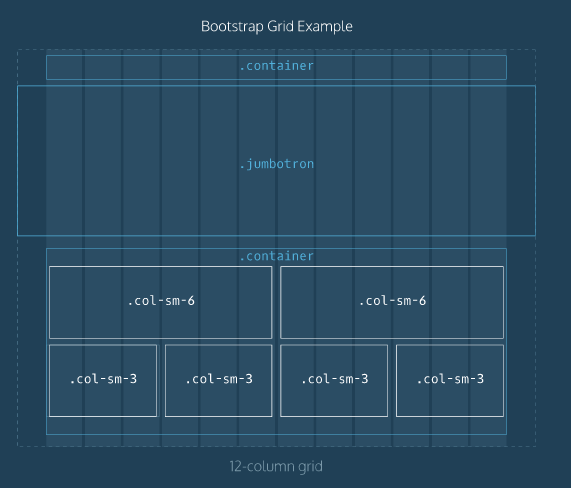
div.header { height: 400px; width: 400px; background-image: -webkit-linear-gradient(#666CCC, #BC1324); }

When an image communicates important information, you can use the <img> element and style the image using CSS, if needed.

When an image is intended to style a web page, you can use the background property and further style it with CSS.

Bootstrap is a popular CSS framework with prewritten CSS rules designed to help you build webpages faster.

In earlier lessons, we linked only to **main.css**. Now, in addition to **main.css**, we will link to a URL that hosts Bootstrap.



The Bootstrap grid contains 12 equal-sized columns, as seen in the diagram on the right. HTML elements are arranged to span different numbers of columns in order to create custom page layouts.

1. The words "container", "jumbotron", "col-sm-6" and "col-sm-3" refer to Bootstrap classes.
2. The element with class "jumbotron" spans the entire width of the webpage, beyond the borders of the grid.

Elements labeled "col-sm-3" take up three grid columns; elements labeled "col-sm-6" take up six grid columns.

<header class="container">

<div class="row">

<h1 class="col-sm-4">Heading</h1>

<nav class="col-sm-8 text-right">

<p>nav item 1</p>

<p>nav item 2</p>

<p>nav item 3</p>

</nav>

</div>

</header>

the row is cut into two parts:

The first part consists of the h1 with Bootstrap's class col-sm-4. It will take up the first four columns on the grid.

The second part consists of the nav element with class col-sm-8. It will take up the remaining eight grid columns. text-right indicates that text will be arranged on the right side of the nav.

In addition to a header/navigation, many websites have a large showcase area featuring important content. Bootstrap refers to this as a jumbotron.

A div with the Bootstrap class row text-center can center subsequent child elements which will contain text:

<section class="jumbotron">

<div class="container">

<div class="row text-center">

...

</div>

</div>

</section>

The anchor element will have Bootstrap's btn btn-primary class, which will transform it into a button.

div elements with the row class are added. Finally, the rows are divided by using divs with Bootstrap's col-sm-... class.

<section class="container">

<div class="row">

<div class="col-sm-6"> ... </div>

<div class="col-sm-6"> ... </div>

</div>

<div class="row">

<div class="col-sm-6"> ... </div>

<div class="col-sm-6"> ... </div>

</div>

</section>

Above, two rows are divided into two equal parts. Each part takes up 6 of bootstrap's 12 columns. Using the col-sm-6 class ensures that this layout will appear when the user's screen is the width of a tablet device(768 pixels). On narrower screens, such as an iPhone, only one image per row will appear.

Footer:

<footer class="container">

<div class="row">

<p class="col-sm-4">...</p>

<ul class="col-sm-8">

<li class="col-sm-1">...</li>

<li class="col-sm-1">...</li>

<li class="col-sm-1">...</li>

</ul>

</div>

</footer>

Above, the row is broken into three parts: a p element that takes up four columns, a ul which takes up 8 columns, and li items which take up 1 column each. The lis could hold navigation menu items or social media icons.

Between the <p class="col-sm-4"> and closing </p> add the website copyright:

&copy; 2016 Skillfair

&copy is a character code, which web browsers interpret as the copyright symbol: ©.