## Languages

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.

## HTML Elements

### h1-h6: indicate text headings on a webpage. h1 is the largest heading; h6 is the smallest.

<h1>Heading</h1>

All HTML elements begin with an opening tag. In this case, the opening tag is <h1>.

Most elements require a closing tag, denoted by a /. In this case, the closing tag is </h1>.

The website user only sees the content between the opening and closing tags.

Headings are a frequently used HTML element. You can think of them like headlines in a newspaper. Your eyes may notice headings first because the words are large and bold compared to other text on the webpage.

There are six heading elements: h1, h2, h3, h4, h5, and h6. h1 is the largest heading and h6 is the smallest.

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

<p>Description of company here.</p>

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

The HTML paragraph element, p, is used to hold one or more sentences, just like paragraphs in an essay or a book.

The webpage is starting to come together.

### a: short for anchor and used to add links to other webpages. Anchor elements typically have an href attribute:

<a href="http://codecademy.com">Click here</a> to learn how to make a website!

Attributes

Attributes change the behavior or appearance of HTML elements. Src sets an image or video source, href sets a URL for anchor and link elements, and class assigns a class attribute to HTML elements.

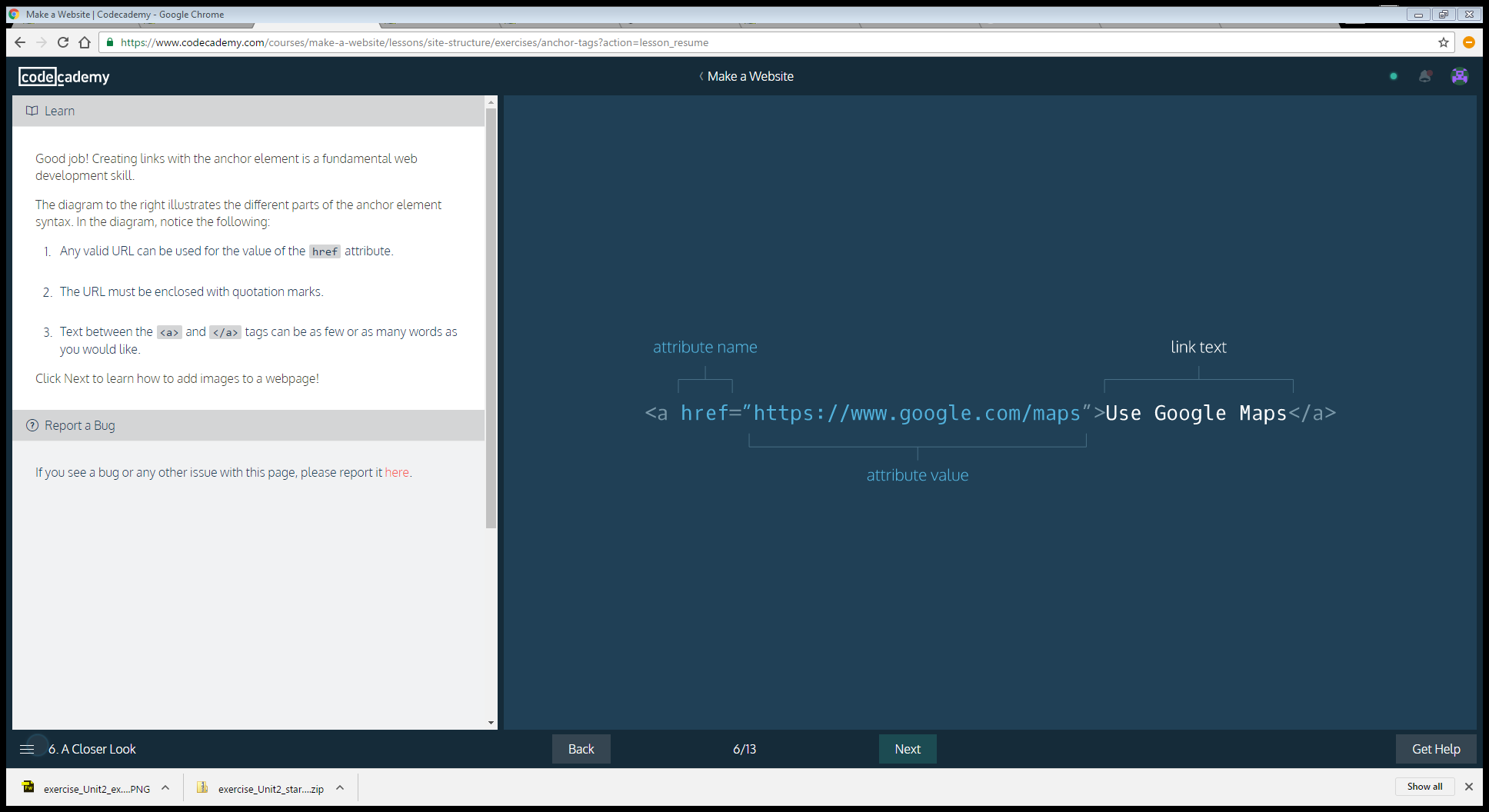
What if you wanted to link users to a different webpage? The HTML anchor element makes it possible to do this with a single click.

<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. Anchor elements use the href attribute, which specifies the webpage to link to. In the example above, the text "Click here for Google!" links to http://google.com.

IMPORTANT: Web addresses, such as http://google.com, have a technical name: URL.

Creating links with the anchor element is a fundamental web development skill.

The diagram to the right illustrates the different parts of the anchor element syntax. In the diagram, notice the following: 

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.

### *img*: used to add an image to a webpage. Image elements are *self-closing*and do not require a closing tag:

<img src="https://images.com/favorite.png">

To add images to a webpage, use the HTML image element:

<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.

### *video*: used to add videos to a webpage, and uses multiple attributes and a nested source element:

<video width="320" height="240" controls>

<source src="https://movies.io/great-clip.mp4" type="video/mp4">

</video>

The HTML video element can add video to a webpage.

<video width="320" height="240" controls>

<source src="video-url.mp4" type="video/mp4">

</video>

The video element uses a number of attributes. Let's take a look at them:

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.

### *unordered list*: used to create lists on a webpage and requires li elements inside a ul:

<ul>

<li>list item</li>

<li>another item</li>

<li>yet another</li>

</ul>

Another essential HTML element is the unordered list. Items in an unordered list are referred to as list items. Each item is bulleted, not numbered. For example:

About unordered lists:

1. ul tags create the unordered list.

2. li tags contain each list item.

Unordered list elements can be used to organize content on a webpage in a number of ways. Below we will use one to organize our website's navigation menu, sometimes called a navbar.

With the video and unordered list elements, you may have noticed something interesting: these HTML elements had other HTML elements nested inside of them.

For example, in unordered lists, li elements are nested inside the ul.

<ul>

<li>First item</li>

<li>Second item</li>

</ul>

Web developers refer to the enclosing element as the parent element and the enclosed elements as children.

Referring to HTML elements as parents and children may sound funny, but it's a core web development concept. The web browser also knows about these parent/child relationships, which will be important as we explore CSS in the next lesson.

Now that we know about HTML element nesting and parent/child relationships, let's see another way these concepts are applied on a real-life webpage.

### div: used to organize HTML elements into different groups, which can be given a class attribute:

<div class="main">

<h2>Subheading!</h2>

</div>

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>

Note: The ... indicates where other HTML elements would normally be enclosed by the div.

Below, we will use divs to divide our page into three separate parts.

### metadata tags: provide metadata about a webpage

The last HTML elements we will explore are involved in metadata processes. You can think of these elements as the "brains" of a webpage because 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.

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

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

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

5. <meta charset="utf-8"/>: Tells the web browser which character set to use. In this case, the character set is "utf-8".

# CSS

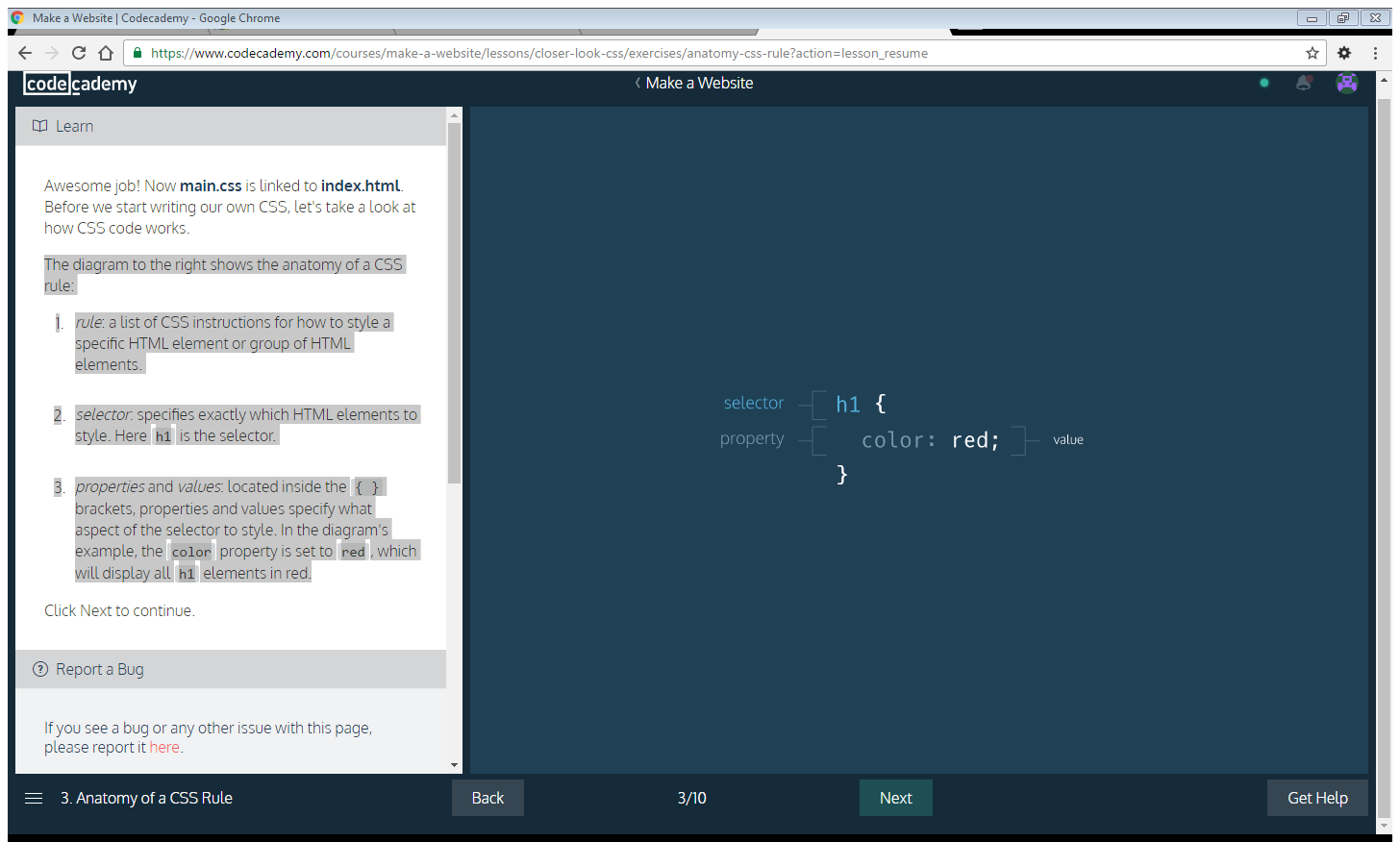
CSS is a language used to style websites. Colors, fonts, and page layouts for a site can all be managed using CSS. The more comfortable you are with CSS, the better equipped you will be to create stylish and contemporary-looking websites.

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. The link element uses three attributes:  
     
   -rel: Specifies the *relationship* between the current file and the file being linked to: in this case, the rel attribute is "stylesheet".  
     
   -type: Specifies the type of file linked to.  
     
   -href: Provides the URL of the file being linked to.
2. Like the HTML image element, link is self-closing. It does not need a closing tag.
3. In the example above, **main.css** is an external style sheet. Using external stylesheets is one of the most popular ways to write CSS. Inline CSS is another method.



The diagram 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.
  2. selector: specifies exactly which HTML elements to style. Here h1 is the selector.
  3. 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.

One of the most effective ways to enhance the look and feel of a website is by changing the font. In CSS, font is managed using the *font-family* property:

h1 { font-family: Georgia, serif; }

Above, the font-family property of the h1 selector is set to the value of Georgia, with serif as a fallback font. 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.

In CSS, the color property sets the color of a CSS selector's font:

h1 { color: red; }

CSS comes equipped with 140 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.
2. 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).

we can useclass selectors to target classes of HTML elements.

Consider the HTML below:

<div class="header"> <h2>Heading</h2> <p>Paragraph</p> </div>

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

[Here's a refresher](https://www.codecademy.com/en/courses/make-a-website/lessons/site-structure/exercises/parent-child) on parent and child elements.

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).

Below, we will use a CSS class selector to style more than one HTML element at once.

The *font-size* property sets the size of an HTML element's text.

h1 { font-size: 60px; }

In the CSS above, font-size is set to 60px. 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.
2. *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 to 32px (a doubling), 3rem is equivalent to 48px (a tripling) and so on.
3. *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.

In CSS, the background-image property sets a background image of your choice for a given selector, as seen below.

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

The CSS rule above assigns the image hosted at

https://s3.amazonaws.com/codecademy-content/projects/make-a-website/lesson-2/bg.jpg

to elements that have a class attribute set to hero.

To control the size of the chosen background image, use the property background-size as seen below:

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

#### **WEB CONCEPTS**

* CSS: Language used to style websites. Colors, fonts, and page layouts for a site are managed using CSS.
* CSS Selectors: specifies exactly which HTML elements to style
  + class selectors: used to target classes of HTML elements
  + id selectors: used to style an HTML element which has an id attribute.
* External Stylesheet: CSS file that styles an HTML file externally via the HTML link element.

#### **CSS PROPERTIES**

* font-family: sets font for a CSS selector.
* color: sets font color for the CSS selector.
* font-size: sets font size for text.
* background-image: sets a background image of your choosing for a given selector.

Observe the CSS box model diagram to the right:

1. *content*: Includes text, images, or other media contained within an HTML element.
2. *padding*: The space between the content and the border. You can think of this like inner space.
3. *border*: The outline of an HTML page element. You can think of it like a picture frame that contains the element.
4. *margin*: The space between the HTML page element and the next nearest element(s).

After you have familiarized yourself with the Box Model, click Next to continue.

The *border* property can be used to visually define a page element's outer edge.

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

1. *thickness*: Sets the thickness of the border, using pixels, ems, or rems.
2. *type*: Sets the border type. Common options aresolid, dotted, and dashed. There are many others.
3. *color*: sets the border's color, using named colors, HEX, or RGB values.

The CSS below gives a paragraph element a solid black border that is 2 pixels thick:

p { border: 2px solid black; }

The CSS padding property controls the empty space between the page element's content and its border. Increasing a page element's padding makes the space around the content more spacious, while decreasing it makes the space more compact.

p { padding: 20px; }

The CSS above would give paragraph elements a padding of 20px.

The CSS margin property controls the space between different HTML elements on a webpage. Use margin to bring page elements closer together or to move them further apart.

The CSS below ensures 2rems of space between elements with the class answer and surrounding page elements.

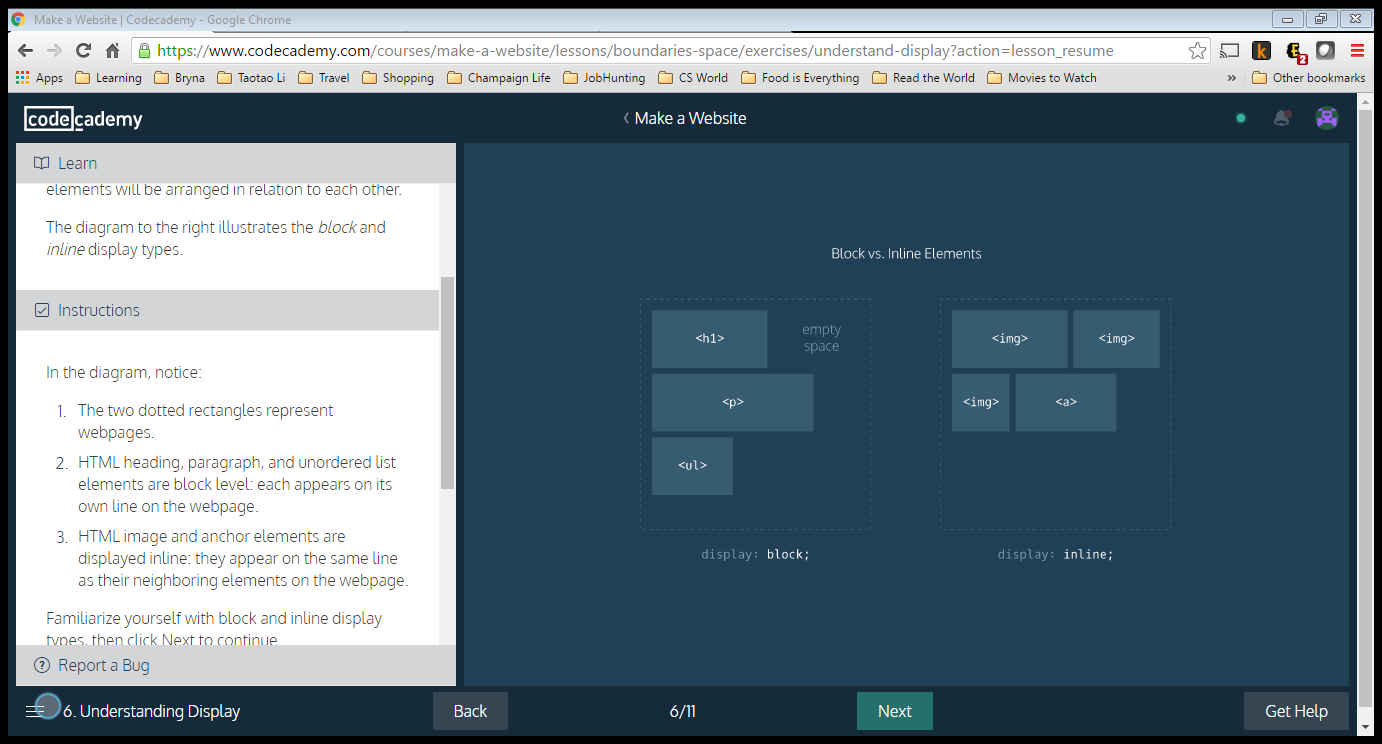
.answer { margin: 2rem; }

The margin property creates space on all sides of a page element. It's also possible to set separate margin spacings on each side of an element.

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.

**Note**: Below we will change margin properties for a div that encloses HTML *figure* elements. Figures are used to organize visuals, such as photos and diagrams.



#### **DISPLAY**

Not all HTML elements are displayed on a page in the same way. Display types determine how HTML elements will be arranged in relation to each other.

The diagram to the right illustrates the block andinline display types.

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.

Familiarize yourself with block and inline display types, then click Next to continue.

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

For example, we can make list items appear on the same line by setting display to inline:

li { display: inline; }

**Note**: Below, we will encounter an HTML *nav* element. Navs are used to organize site navigation menus on a webpage.

The navbar is starting to come together nicely. It would be even better if we could get the Contact button to fill in the empty corner on the bottom right.

To achieve this, we can use the CSS *float* property. By using float, we have the option of floating elements left or right.

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; }

Let's arrange the Contact button using float.

The CSS display value that arranged the images,*flex*, has been removed. In addition to other capabilities, flex can be used to easily align multiple page elements horizontally.

In the example code below, there is a div with class parent:

<div class="parent">

...

</div>

To make children of the div align horizontally on the webpage, in CSS we can use:

.parent {

display: flex;

}

Children elements of the div with class parent will now align horizontally. We can make sure no child element moves off the page by using flex-wrap: wrap;:

.parent {

display: flex;

flex-wrap: wrap;

}

Finally, to center rows of children elements, we can use justify-content: center;:

.parent {

display: flex;

flex-wrap: wrap;

justify-content: center;

}

The CSS *position* property enables you to position HTML elements in exact locations on a webpage. One useful value for this property is *relative*. This value positions page elements on a webpage relative to where they would normally appear.

By first setting position: relative;, you can then use the CSS properties top, left, bottom, and right to shift an element away from where it would have normally appeared on the page.

The code snippet below moves a div with the class container 10px away from the up and 20px away from the left side of the page.

.container {

position: relative;

top: 10px;

left: 20px;

}

Ever click a button on a webpage that seemed to move down and then back up like a real-life button? We can implement this trick using the position property.

# Bootstrap

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

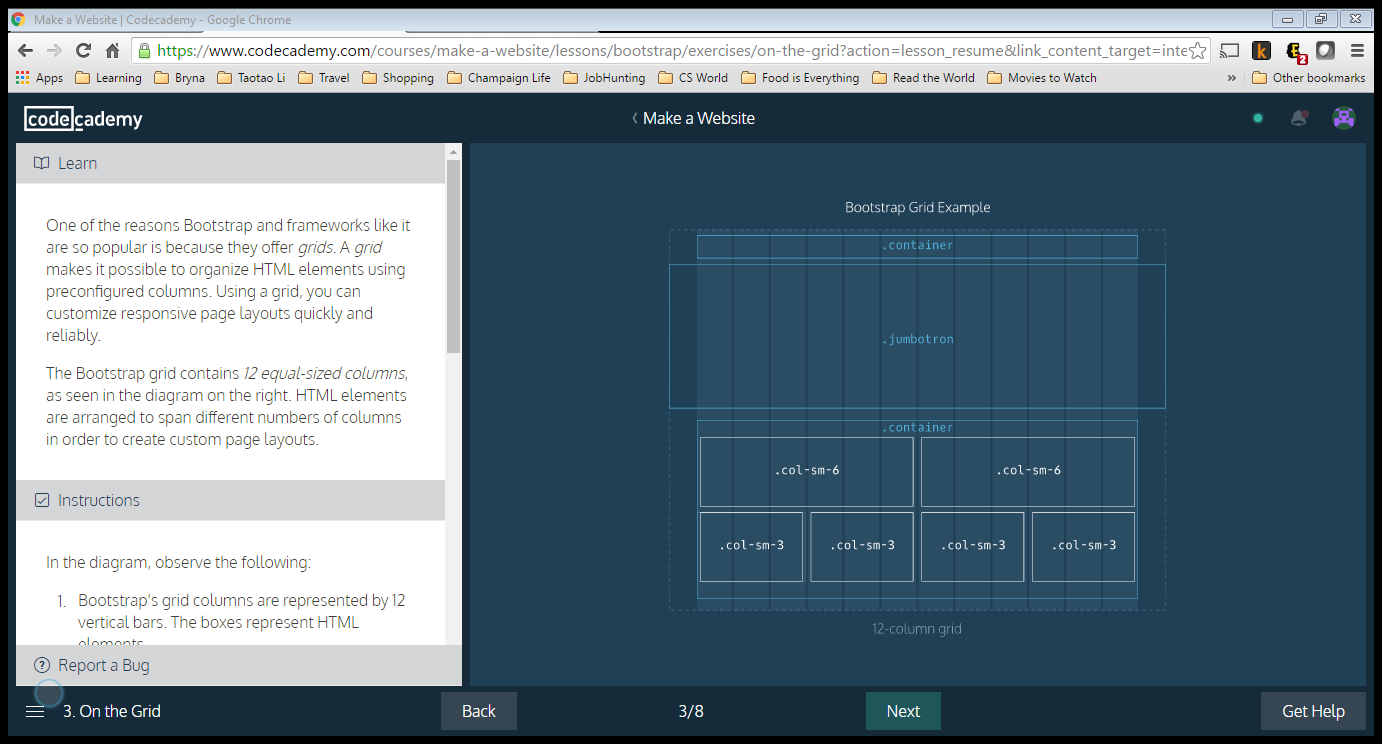
Also in this lesson, we will encounter three new HTML elements:

1. *header*: Used to organize headers on a webpage.
2. *footer*: Used to organize footers on a webpage.
3. *section*: Defines sections on a webpage.

Before Bootstrap can work for us, we need to link to it.

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 HTML link element makes Bootstrap available to us. Remember that the link element is typically contained within HTML head tags.



One of the reasons Bootstrap and frameworks like it are so popular is because they offer *grids*. A *grid* makes it possible to organize HTML elements using preconfigured columns. Using a grid, you can customize responsive page layouts quickly and reliably.

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.

In the diagram, observe the following:

1. Bootstrap's grid columns are represented by 12 vertical bars. The boxes represent HTML elements.
2. The words "container", "jumbotron", "col-sm-6" and "col-sm-3" refer to Bootstrap classes.
3. The element with class "jumbotron" spans the entire width of the webpage, beyond the borders of the grid.
4. Elements inside the second "container", such as "col-sm-6" and "col-sm-3" are contained within the grid columns.
5. Elements labeled "col-sm-3" take up three grid columns; elements labeled "col-sm-6" take up six grid columns.

In addition to a header/navigation, many websites have a large showcase area featuring important content. Bootstrap refers to this as a *jumbotron*. Below is an example implementation of a jumbotron.

First, a new *section* element is created and assigned the jumbotron class:

<section class="jumbotron"> </section>

Next a div with the Bootstrap class container is used:

<section class="jumbotron"> <div class="container"> ... </div> </section>

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

<section class="jumbotron"> <div class="container"> <div class="row text-center"> ... </div> </div> </section>

The ... is a placeholder for HTML elements containing text, such as h2, p or anchor elements.

Many websites have a supporting content area. Supporting content can be arranged using Bootstrap's grid. Below is an example implementation of a supporting content area.

First, an HTML section element with the containerclass is used:

<section class="container"> </section>

Next, div elements with the row class are added:

<section class="container"> <div class="row"> </div> <div class="row"> </div> </section>

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.

Footers display copyright information, social media links and sometimes site navigation.

Below is one possible implementation for a footer section using Bootstrap:

First, a footer element with Bootstrap's containerclass is used:

<footer class="container"> </footer>

Inside the footer, a div with Bootstrap's row class is added to hold footer content:

<footer class="container"> <div class="row"> ... </div> </footer>

Finally, the row is divided into parts using Bootstrap's grid. Here is one suggestion:

<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 pelement 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.

Again, because the col-sm-... class is used, this layout will appear on tablet-width screens and wider. Screen sizes smaller than tablet-width (768 pixels), will not maintain this layout.

# **HTML Glossary**

Programming reference for HTML elements

# **Attributes**

## class

HTML elements can have one or more classes, separated by spaces. You can style elements using CSS by selecting them with their classes.

**Example**

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

## 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>

## href

Links tell the browser where to go using an href attribute, which stores a URL.

**Example**

<a href="http://google.com">Google it!</a>

# **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>.

# **Body**

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

**Example**

<html>

<head>

<title>An example of the body tag</title>

</head>

<body>

This is inside the body!

</body>

</html>

**Read more**

* <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/body>

## Usage

Almost all content belongs inside the body tag. The main exceptions are script and style tags, as well as the page title tag. As you can see in this example, there is a heading, an image, and a link all inside the body tag. The head tag contains only external files and the page title.

**Example**

<html>

<head>

<title>My homepage</title>

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

<script type="text/javascript" src="homepage.js"></script>

</head>

<body>

<h1>Hello, this is a picture of my cat!</h1>

<img src="cat.jpg" />

<a href="mailto:cat@codecademy.com">Email my cat</a>

</body>

</html>

# **Children**

An element that is an immediate descendent of another element or nested within another element is called a child. These become useful when using CSS child selectors and psuedo-elements.

**Example**

<ul id="parent">

<li id="child">I'm a child of parent!</li>

</ul>

# **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! -->

# **Div**

A block level container (or 'division' of the web page) for content with no semantic meaning.

**Syntax**

<div>This is a div element.</div>

# **Head**

Tag that surrounds important content that is invisible to the user, but is important to the browser. Elements within this tag contain metadata about the page and links to stylesheets, scripts, etc.

<html>

<head>

</head>

<body>

</body>

</html>

# **Headings**

Heading elements like <h1>, <h2>, <h3>, ... allow you to use six levels of document headings, ranging from largest to smallest, breaking up the document into logical sections. For example, the word 'Headings' above is wrapped in a <h2>tag.

**Syntax**

<h1> This is a header! </h1>

# **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!

**Read more**

* <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/hr>

# **HTML**

## What is HTML?

HTML stands for Hyper Text Markup Language. It is the language used to create all websites.

**Read more**

* <http://www.w3.org/wiki/HTML/Training/What_is_HTML>

## <html> tag

All HTML files live within an over-arching html tag. This is the basic tag that defines an html document.

**Syntax**

<html>

The rest of your web page goes in here!

</html>

**Read more**

* <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/html>

# **Hyperlinks**

Hyperlinks (or just links) take the user to another webpage when they click on it. The most common attribute used with links is href, which tells the browser where the link goes.

**Syntax**

<a href="url this link goes to">Link text</a>

**Example**

The following text is <a href="http://google.com">goes to Google</a>.

# **Images**

The img tag embeds an image into your HTML. Always found with the 'src' attribute, which tells the browser where to find the image. Note that the <img/> tag is self-closing.

**Syntax**

<img src='mylocalimage.jpg'/>

# **Line breaks**

This tag is used in a block of text to force a line break. This is to be used for things which are a single paragraph, but where this formatting is necessary such as poems or addresses. To separate paragraphs, separate each paragraph into a separate element instead. The resulting element on a web page will look like:

**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.

**Exampl**

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

# **Lists**

HTML supports two kinds of lists: ordered lists and unordered lists. Within lists each individual list item has its own tag.

## Unordered Lists

Unordered lists are just lists whose items are denoted with bullet points.

**Example**

Shopping list

<ul>

<li>Dish soap</li>

<li>Kitty litter</li>

<li>Tomato sauce</li>

</ul>

**Read more**

* <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ol>

## 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>

**Read more**

* <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/li>

# **Paragraphs**

## <p>

One of the most common tags in HTML - it denotes a paragraph of text. It often has other elements nested inside of it, such as <img/>, <a>, <strong> and <em>.

**Syntax**

<p>This is paragraph text!</p>

# **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>

<th>Price</th>

</tr>

</thead>

<tbody>

<tr>

<td>Banana</td>

<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>

# **Tags & Elements**

Tags are basic labels that define and separate parts of your markup into elements. They are comprised of a keyword surrounded by angle brackets <>. Content goes between two tags and the closing one is prefixed with a slash (Note: there are some self-closing HTML tags, like image tags). Tags also have attributes, which are

**Syntax**

<tag attribute='value'>content</tag keyword>

# **Title**

This tag tells the browser what to display as the page title at the top and tells search engines what the title of your site is. It goes inside <head> tags. Try and make your page titles descriptive, but not overly verbose.

**Example**

<title> HTML Glossary </title>

# **HTML5 Features**

Web languages need regular upgrades in order to stay current and solve new problems faced by web developers. HTML5 is the latest version of HTML. Below are some HTML5 features you will encounter as you learn with Codecademy.

## video

The video element allows you to easily stream video from a website.

<video width="450px" height="350px" controls>

<source src="video-url.mp4" type="video/mp4">

</video>

In the HTML above, width and height set the dimensions for the video element. The controls attribute creates playback buttons such as "Play" and "Pause". The source src tag provides the URL where the video is hosted and type specifies the video's type, in this case, "video/mp4".

## 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 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>

## header

The header element can be used to group together introductory elements on a website, such as a company logo, navigation items, and sometimes, a search form.

<header>

<img src="company-logo.png">

<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>

</header>

Above, the header element encloses the img and 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".

# **CSS Glossary**

Programming reference for CSS covering Comments, Properties, and Selectors

# **Comments**

Comments in CSS are signified by a forward-slash and asterisk.

**Example**

/\* This is a single line comment \*/

**Example**

/\* This

is a multi-line

comment \*/

# **Properties**

## Definition

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;

}

**Read more**

* <http://www.htmldog.com/reference/cssproperties/>

## Defining many properties

Each CSS rule can have as many properties as you like. Each of them applies to the elements that the selector applies to.

**Example**

h1 {

font-size: 24px;

font-weight: bold;

border: 1px solid black;

color: pink;

}

/\* This will make all <h1> headers big, bold, pink, and inside of a thin black rectangle! \*/

## 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;

}

## font-family

The font-family property sets the font of an HTML element's text.

**Syntax**

p {

font-family: Arial, Helvetica, sans-serif;

}

# **Selectors**

What are selectors?

Selectors are used in CSS to select the parts of the HTML that are being styled. You can use several different methods for selecting an element.

**Syntax**

selector {

rules;

rules;

rules;

}

**Read more**

* <https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started/>

## Class name selectors

You can also select HTML elements by their Class name. Unlike ID selectors, Class selectors select all elements with a matching class.

**Example**

a.link {

font-size: 12px;

}

/\* HTML Selected: <a href="http://google.com" class="link">,

<a href="http://codecademy.com" class="link jumbo"> \*/

**Example**

.jumbo {

text-size: 1000px;

}

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

<span class="jumbo"> \*/

## Element selectors

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

**Example**

body {

background-color: #333;

}

**Example**

h1 {

color: blue;

}

**Example**

a {

text-underline: none;

}

## ID selectors

ID selectors are used to select only a single item on a page. Like the term ("identification") indicates, ID selectors will ONLY select the first element with a matching ID.

**Example**

#thatThingINeededToStyle {

color: blue;

font-size: 24px;

}

/\* HTML Selected: <span id="thatThingINeededToStyle"> \*/

**Example**

a#codecademy {

color: purple;

}

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

## 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"> \*/

**Example**

input[type="text"] {

width: 100px;

}

/\* HTML Selected: <input type="text"> \*/

**Example**

input[required] {

border: 1px red solid;

}

/\* HTML Selected: <input type="text" required> \*/

**Read more**

* <http://dev.opera.com/articles/view/27-css-basics/#attribute>

## 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 \*/

**Read more**

* <https://developer.mozilla.org/en-US/docs/Web/CSS/Descendant_selectors>

## 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 \*/

**Read more**

* <https://developer.mozilla.org/en-US/docs/Web/CSS/Universal_selectors>
* <http://www.stevesouders.com/blog/2009/06/18/simplifying-css-selectors/>
* <http://dev.opera.com/articles/view/27-css-basics/#universal>

## Pseudo class selectors

Pseudo Selectors can be used to narrow down a selection with certain rules.

**Example**

li:first-child {

color: red;

}

/\*

This selects only <li> elements that have no elements before them

<ul>

<li>Selected; will be red</li>

<li>Not selected</li>

<li>Not selected</li>

</ul>

\*/

li:last-child {

color: red;

}

/\* This does the opposite; only the last <li> will be red. \*/

**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. \*/

**Read more**

* <https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes>
* <http://dev.opera.com/articles/view/27-css-basics/#pseudoclasses>

# **CSS3 Features**

Overview of CSS3 features used in the Make a Website course.

# **CSS3 Features**

CSS3 is the latest version of CSS and contains a number of exciting, new features that make it easier for web developers to create great styling for websites.

## rem values

You will encounter rem values as you learn about the CSS font-size property, and other CSS properties that specify element size. In the case of the font-size property, a rem value displays a font-size relative to the font-size of the root element, called simply html.

For example, consider the following CSS:

html {

font-size: 20px;

}

p {

font-size: 0.75rem;

}

In the code, font-size is set to 20px for the html selector. The p selector has a font-size of 0.75rem, which displays a size 2/3 that of html, or 15px.

## Flexbox

The CSS3 flexbox feature makes it much easier for web developers to arrange HTML elements vertically or horizontally. Website layouts designed with flexbox can respond to users with various screen widths, including mobile devices. To access the feature in CSS, the display property must be set to flex, as seen below:

.main {

display: flex;

}

HTML elements that are children of an element with the "main" class are now flex items and can be arranged using flexbox properties. Here two such such properties:

* flex-wrap: arranges flex items into a single line or wraps them across multiple lines of a webpage layout.
* justify-content: can orient flex items in a number of different positions on a webpage, including the center.

Interested in learning more about flexbox? [Here is a great supplemental resource](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Using_CSS_flexible_boxes).

## background-image

The CSS3 background-image property is used to set a background image for an HTML element.

For example, consider the following CSS:

body {

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

}

## background-size

The CSS background-size property controls the size of an HTML element's background image. Used along with the background-image property, background-size values control how a background image is proportioned and scaled. One such value is cover as seen in the following CSS:

body {

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

background-size: cover;

}

Here, the image covers the entire HTML body element.