**LESSON 1: LEARN TO CODE**

**The World Wide Web**

HTML: *HyperText Markup Language* is the document that the majority of the web is made up of, browsers interpret the HTML language to make it viewable.

**Components of the Web**

**Client**>**Browser>Internet>HTTP>Servers**

**HTML Document Structure**

Generally an indent means a new idea. In HTML. The level increases at the beginning of the idea and decreases when finished.

**HTML Document example**

<!DOCTYPE html> This just tells the interpreter that this is an HTML 5 document

<html> this tag just opens our document

<head>

This is my Header that is not going to show on the page, but contains metadata or information that can be used by browsers and search engines

</head>

<title> typically a short one line tag that is closed on the same line</title>

<body> this is the content of the web page.

<h1>Headers can be from level 1 (smallest) to level 6</h1>

<p> Paragraphs and headers are both child elements of the body and can be indented at the same level</p>

</body>

**Basic HTML Elements**  
An opening and closing tag with some content in the middle

**Block-level Elements**block-level: starts on a new line and takes up the full width (stretches out to the left and right as far as it can).

Examples:

<div> is used as a container for other HTML elements. No required attributes for it, but usually used for style and class. In CSS, div can style content blocks.

<h1> - <h6>

<p>text</p> Defines a paragraph. The content of the <p> element is in a "box"

<form>

**Inline Elements**

inline element: doesn’t start on a new line and only takes up as much width as needed

Examples:

<b>text</b>  **bold**   
<em>text</em> italic

<br> Creates a line break   
<span> is a container used for text elements. No required attributes for it, but usually used for style and class. In CSS, span can style text blocks.

<a> Anchor

<strong> strong text

**HTML Attributes**

Some elements have **attributes** in their opening tags which give additional information to the browser  
<a href="url">website</a> Creates a hyperlink to the href attribute value url and names it **websites**  
<img src="url" alt="text"> Displays the image hosted at the src attribute value "url" or local file. If the image cannot be displayed, then the alt attribute value "text" is shown.

**LESSON 2 MAKE A STYLISH WEBPAGE:**

# Understanding CSS

### Search and Replace

You can also think about CSS as a search and replace tool: you identify a class or a tag of the element you want to find (or match, in CSS terminology), and then what you want to do with it, or what property values to replace with different ones.

### Order Matters

It also matters where you define the rules and in what order they are applied. Styles can be defined in different places and are applied in the following order, with definitions further down the list overwriting previous definitions:

* the default style of a browser (different browsers have slightly different styles)
* **stylesheet in a separate file (this is what you will be mostly using)**
* stylesheet inside HTML (this can be done for small projects but is not ideal)
* inline style in an element (this can also be done but should be avoided)

### Specifics Matter

"Cascading" means that rules are applied not only to the elements they directly match, but also to all of those elements' child elements. However, if a child element has multiple, overlapping rules defined for it, the more specific rule takes effect.

Follow us on Twitter!  
[**@cwpittman**](https://twitter.com/cwpittman)  
[**@JessicaUelmen**](https://twitter.com/JessicaUelmen)

CSS Summary and Reference

There was a lot of information in the previous video and you don't need to remember it all. This is a text summary. Quickly skim through it (just look at the headings so you know what's here) and come back when you need to.

### What is CSS?

CSS is code written to control the "style" of HTML elements.

### How does CSS "control" HTML?

If you want to write CSS that makes all h1 elements have a black background and white text, you would write:

h1 {

background-color : black;

color: white;

}

In this example, the h1 is a **selector**. It says to the browser "I want you to apply the rules I'm about to tell you to every h1 element."

After the h1 there is a left curly brace { and at the bottom there is a matching right curly brace }. Everything between these curly braces will be interpreted as a "rule" that should be applied to every h1 element.

The line of code that says background-color : black; is a **declaration**. background-color is a **property** and black is the **value** of that property.

### CSS Vocabulary

**CSS**: Cascading Style Sheets.

**Style**: This word can refer to many things and so it can be confusing. It can refer to:

* The HTML element. For example: <style>div {color:blue}</style>
* The HTML attribute. For example: <div style="color : blue">this text would be blue</div>
* The general look of a web page. For example: "I like that site's style."
* A verb. For example: "I'm making progress on my page. The structure is all done but now I have to style it."

**Rule**: a line of CSS code describing the value that a certain attribute should take.

**Property**: The property you want to change.

**Value**: The value that you want to assign to the attribute.

**Selector**: The name that you use to in order to target the elements that are assigned to a class or id attribute in the HTML.

**Class**: A class refers to a group of elements that can be styled together. Class names should not contain periods or any other punctuation marks such as class="1.1"

**ID**: ID's are unique identifiers that uniquely identifies an element in HTML.

### Selecting by class

In the example above, we set the background-color of every h1 to black. If we only want to add style to certain h1s, we can use **class selectors**. Try copying the following code into a new HTML file in your text editor to see how this works.

<div class="stop"></div>

<div class="slow"></div>

<div class="go"></div>

<style>

div {

height : 50px;

width : 50px;

border-radius: 25px;

}

.stop {

background-color: red;

}

.slow {

background-color: yellow;

}

.go {

background-color: green;

}

</style>

We should see a traffic light. Note how in the <style> element we refer to an HTML element's class name by writing .class-name

Also note how we can apply rules to **all** divs or just certain divs by using the appropriate selector.

### How do I include CSS Styling in my web page?

There are three ways you can do this.

**Method 1: Write CSS in the <head> of your HTML**

If you want to quickly style an HTML document, you can do the following:

1. At the top of your HTML document, add a <head> element.
2. Inside the <head> element, add a <style> element.
3. Write your CSS in the <style> element and then put the HTML for the rest of your page below.

For example:

<head>

<style>

div {

background-color : red;

}

</style>

</head>

<body>

<div>

This will have a red background.

</div>

</body>

**Method 2: Link your HTML to a separate CSS file**

This adds another step, but it lets you stay more organized when working on larger projects. To do this:

1. Write all of your structural HTML in one file (let's call it main.html).
2. In a separate file (let's call it main.css), put all of your CSS code.
3. Add a <head> element to the top of your HTML.
4. Add a <link> tag inside the <head> element. Since <link> is a "void tag" you don't need to add a closing </link>.
5. Add the following attributes to your <link>:
   * rel="stylesheet"
   * href="main.css"

For example, you would have a main.html file that looked like this:

<head>

<link rel="stylesheet" href="main.css">

</head>

<body>

<div>

This will have a red background.

</div>

</body>

and in the same folder you would have a file called main.css like this:

div {

background-color : red;

}

**Method 3: Write your style inline with your HTML**

This is generally a very bad idea because it leads to **lots** of repeated code. But you may see code that uses this method so it's good to be familiar with it. If you want to know more, check out the question and answers in [**this conversation**](http://stackoverflow.com/questions/2612483/whats-so-bad-about-in-line-css) on Stack Overflow (Stack Overflow is the most widely used programming Q&A community out there).

With this method, you modify the style attributes of every individual HTML element. For example:

<body>

<**div** style="background-color: red; color: white">

This **div** will have a red background **and** white **text**.

</**div**>

<**div** style="background-color: red; color: white">

So will this one.

</**div**>

<**div** style="background-color: red; color: white">

Now, what **if** I change **my** mind?

</**div**>

<**div** style="background-color: red; color: white">

I'd rather have a black background...

</**div**>

<**div** style="background-color: red; color: white">

Never do this!

</**div**>

</body>

**NOTE:** there are a lot of little mistakes you can and will at some point make that will cause any of these methods not to work. For example, when I was writing the code for method 2 I had main.html and main.css in different folders on my computer, and it didn't work at all.

## More on Semantic Elements

### Content Sectioning

[**https://developer.mozilla.org/en-US/docs/Web/HTML/Element#Content\_sectioning**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element#Content_sectioning)

### Sections and Outlines of an HTML5 Document - The HTML5 Outline Algorithm

[**https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Sections\_and\_Outlines\_of\_an\_HTML5\_document#The\_HTML5\_Outline\_Algorithm**](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Sections_and_Outlines_of_an_HTML5_document#The_HTML5_Outline_Algorithm)

### Default Style Rules

Browsers use default stylesheets to determine how to display HTML elements. You can view the default style rules for h1and other elements for the following browsers:

* [**WebKit (Chrome and Safari)**](http://trac.webkit.org/browser/trunk/Source/WebCore/css/html.css)
* [**Firefox**](https://developer.mozilla.org/en-US/docs/Web/CSS/Reference)
* [**Internet Explorer**](http://www.iecss.com/)

Because these rules differ sometimes between browsers, there are efforts to promote consistency in styles across browsers. One popular solution to this issue is using what is referred to as a CSS reset such as [**normalize.css**](http://necolas.github.io/normalize.css/).

# Quiz: The Box Revisited

### Web Pages Demonstrated in this Video

1. [**Box Model**](http://assignments.udacity-extras.appspot.com/courses/html-css/samples/box-model.html) page (box-model.html). In video: 00:00-03:21.
2. [**Style 1**](http://assignments.udacity-extras.appspot.com/courses/html-css/samples/style-1.html)page (style-1.html). In video: 04:33-06:01.
3. [**My Favorite App**](http://assignments.udacity-extras.appspot.com/courses/html-css/img/mock1-fav-app.pdf) (mock1-fav-app.pdf). In video 06:02 - 06:15.

Note that some browser plugins will interfere and try to access these links via the HTTPS protocol. If you get an error: "403. That’s an error. Your client does not have permission to get URL /courses/html-css/samples/box-model.html from this server. That’s all we know. " try disabling your browser extensions or plugins.

### CSS to Use for Quiz

In your style.css you should set the screenshot class to max-width of 460px, and the description to 705px. The other things to set in your style.css file are the box-sizing definitions and the outline that allows you to easily see all the boxes on your page. Put the following in your style.css file:

\* {

outline: 1px solid red !important;

}

\* {

-webkit-box-sizing: border-box;

-moz-box-sizing: border-box;

-ms-box-sizing: border-box;

box-sizing: border-box;

}

# Positioning Boxes

The mockup Jessica references can be found here: [**mock1-fav-app.pdf**](http://assignments.udacity-extras.appspot.com/courses/html-css/img/mock1-fav-app.pdf)

A great guide to flexbox can be found [**here**](http://css-tricks.com/snippets/css/a-guide-to-flexbox/)!

At 01:02 in the video, Jessica uses Dev Tools to add the CSS rule display: flex to element.style, which applies the CSS to the selected div element temporarily.

Students following along with the videos should create an .app class in their style.css as follows:

.app {

display: flex;

}

This rule as-is works with Chrome 29+, IE 11+, and Mozilla 28+. In order to support Safari, the -webkit- prefix must be added:

.app {

display: -webkit-flex;

display: flex;

}

# Box Sizing and Positioning Summary

### Box Sizing

There are four main points that Jessica addressed about box sizing.

1. HTML elements are boxes and each box has 4 components.
2. Because there are so many components to each box, it can often be hard to get the size of a box just right.
3. There are two techniques you can use to help deal with sizing issues:
   * Set sizes in terms of percentages rather than pixels.
   * Set the box-sizing attribute to border-box for every element
4. Different browsers work slightly differently. Sometimes this causes different browsers to display the same code differently.

### Box Positioning

1. Divs are **block** elements (as opposed to **inline**), so by default they take up the entire width of a page.
2. Adding the rule display: flex; to the appropriate CSS will override this behavior and let divs appear next to each other.

**Quiz: Adding Image**

You can download the sample image [**here**](https://storage.googleapis.com/supplemental_media/udacityu/2771378573/app.png) and the mockup PDF [**here**](https://www.udacity.com/api/nodes/2890318656/supplemental_media/mock1-fav-apppdf/download?_ga=1.231106904.672083044.1467344711%20target=%22_blank%22). (If the image opens in a new web page or tab you can right+click or option+click the image to save it.) When saving be sure to save the file in the images directory you created in your toplist project directory. You can also use your own image if you wish.

Adding an alt attribute is as simple as changing your code to:

<img src="images/app.png" alt="This is a screenshot">

In the previous video, Jessica added the CSS rule display: flex to element.style, which adds the CSS directly to the div element on the left-hand side of the development tools. In this and future videos, this CSS rule is in the main CSS file, style.css:

.app {

display: flex;

}

This rule as-is works with Chrome 29+, IE 11+, and Mozilla 28+. In order to support Safari, the -webkit- prefix must be added:

.app {

display: -webkit-flex;

display: flex;

}

You can compare your web page to the [**mockup PDF here**](http://assignments.udacity-extras.appspot.com/courses/html-css/img/mock1-fav-app.pdf) to see what is still different. (Note that the color boxes and code on the left of the mockup PDF are for reference only and are not intended to be part of your web page structure.)

# Quiz: Verifying HTML And CSS

### Verifying HTML and CSS

To verify HTML: [**http://validator.w3.org/#validate\_by\_input**](http://validator.w3.org/#validate_by_input)  
To verify CSS: [**http://jigsaw.w3.org/css-validator/#validate\_by\_input**](http://jigsaw.w3.org/css-validator/#validate_by_input)

### What Element Is Missing Something?

If there's an element (let's call it a but it could be body, div, head, or any other valid HTML element) missing an element that it should have inside of it (let's call it b), then enter a or <a> into the first box and b or <b> into the second box, where a and b correspond to the elements' names.

At 0:15 in the solution video It sounds like Jessica says "head elements". In fact there can be only one head element in an html page. Your CSS should be correct and validate without errors.

### Apply What You've Learned

This is the last work session for Stage 1. After this you'll be submitting your final project for Stage 1, your own personal web page. So let's get started!

* Download the stage1.zip file
* Unzip the file by double-clicking it
* Open the Project Folder in your text editor
* Confirm you have two project files: index.html and style.css

Your project is to create a webpage about anything you'd like, using all of the concepts you've learned thus far including:

* Using HTML tags like <div>, <p>, <h1> to structure your page
* Styling your page using CSS class selectors
* Adding images to your page by referencing local files on your computer or online files via a URL

Be Creative! Here are some ideas for your webpage:

1. Expand your notes page from Stage 0 - this time with style!
2. Create an "About Me" page that you can share with others
3. Make a list of your favorite baseball players with their stat lines
4. Compile a playlist by linking to songs on Spotify or YouTube

These are all things you can already do with the knowledge you have. Once you start working on being ready for project submission, be sure to check all project requirements discussed in on the Project Submission page.

If you need help completing this project, feel free to reach out on the [**Discussion Forums**](https://discussions.udacity.com/c/nd000-make-a-web-page) or schedule a [**1:1 coaching appointment**](https://calendly.com/ipnd-1-1).

