12-Web Summary

<https://www.w3schools.com/>

<https://www.w3schools.com/html/default.asp>

for anything related to web design

place to go for review etc.

While many times W3 Schools will provide fast, easy to read answers, they aren't a

considered a reputable source and have been known to provide inaccurate information at times.

The [Mozilla Developer Network](<https://developer.mozilla.org/en-US/> ) is considered a

much more authoritative resource for web development technologies.

[Mozilla Docs](<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a#attr-target>

HyperText Markup Language is the authoring language used to create pages on the World Wide Web. **HTML** is a set of codes or **HTML** tags that provide a web browser with directions on how to structure a web page's information and features.

All HTML documents must start with a document type declaration: <!DOCTYPE html>.

* The <!DOCTYPE html> declaration defines this document to be HTML5
* The HTML document itself begins with <html> and ends with </html>.
* The <html> **element** is the root element of an HTML page
* The <head> element contains meta information about the document
* The <title> element specifies a **title** for the document
* The visible part of the HTML document is between <body> and </body>.
* The <body> element contains the visible page content
* The <h1> element defines a large **heading**
* The <p> element defines a **paragraph**
* <p>This is a paragraph.</p>
  + Note no “ “
* HTML **links** are defined with the <a> tag:
* <a href="https://www.w3schools.com">This is a link</a>
* HTML **images** are defined with the <img> tag.

The source file (src), alternative text (alt), width, and height are provided as **attributes**:

* <img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">
* All HTML elements can have **attributes**
* Attributes provide **additional information** about an element
* Attributes are always specified in **the start tag**
* Attributes usually come in name/value pairs like: **name="value"**
* <https://www.w3schools.com/tags/ref_attributes.asp>
* The HTML5 standard does not require lowercase **attribute** names.
* The title attribute can be written with uppercase or lowercase like **title** or **TITLE**.
* W3C **recommends** **lowercase** in HTML,
* Here, a title attribute is added to the <p> element. The value of the title attribute will be displayed as a **tooltip** when you mouse over the paragraph:
* <p title="I'm a tooltip">  
  This is a paragraph.</p>
* HTML **buttons** are defined with the <button> tag:
* <button>Click me</button>
* HTML **lists** are defined with the <ul> (unordered/bullet list) or
  + the <ol> (ordered/numbered list) tag, followed by <li> tags (list items):
* <ul>  
    <li>Coffee</li>  
    <li>Tea</li>  
    <li>Milk</li>  
  </ul>
* <br> is an empty element without a closing tag (the <br> tag defines a **line break**).
* HTML **tags** are not case sensitive: <P> means the same as <p>.
* The HTML5 standard does not require lowercase tags, but W3C **recommends** **lowercase** in HTML
* The <hr> tag defines a thematic break in an HTML page, and is most often displayed as a horizontal rule.
* The <hr> element is used to separate content (or define a change) in an HTML page:
* <pre> defines pre-formatted text

**Did exercises thru HTML paragraphs**

**HW notes:**

&nbsp = no breaking space…i.e. a space around text

target="\_blank" opens a new tab for a link or image

see other <a> tag target attribute values

<https://www.w3schools.com/tags/att_a_target.asp>

For data: <http://convertcsv.com/csv-to-html.htm>

Copied results

Added this: table-responsive

<https://getbootstrap.com/docs/4.1/content/tables/#responsive-tables>

**Atom**

HOW DO I GET THE AUTOCOMPLETE OF THE STARTING ELEMENTS IN ATOM?

AND EXPLANATIONS…like VSC

control option b = atom beautify to clean up code

command-shift-p = command palette

tr = table rows

td = table columns/data

colspan = lets info span multiple columns

<br> = space on page

within atom

skeleton code shortcuts (Emmet add-on in atom; or VScode auto)

tr\*3 = 3 rows

(tr>td\*3)\*7 = 3 columns within each row, and then 7 rows

9/22/18 – Sat. day 1 – AD

SEE POWERPOINT…

**H**yper**t**ext **M**arkup **L**anguage = **html**

Being a markdown language, HTML doesn’t use variables or loops or functions or have any kind of programming logic.

Instead we use HTML tags to declare what should be rendered to the page in the browser.

\*Most\* HTML elements follow the format of:

Opening tag <h1>

Content blah blah

Closing tag </h1>

* + - It’s a heading tag. They come in 6 numbered sizes (h1-h6), with h1 having the largest text and being the most important
    - Heading elements can help describe and section off our content.

vs. self-closing tags <img src= “…png”>

An img tag creates an image element, and the src attribute describes the path or URL of the image to display.

We don’t pass any content into an img tag, and it doesn’t need a closing tag.

SEE added files in folder 12.1.

TEXT EDITORS:

Dylan: “the \*best\* modular editor, use **atom** <https://atom.io/> “

**Visual studio code**

“also highly recommend https://code.visualstudio.com/ visual studio code since it is an actual IDE vs text editor”

Sublime <http://www.sublimetext.com/3>

Other options: TextWrangler,

TextEdit

IDE means "**Integrated Development Environment**"

Yukon: “for any IDE you use, try to add a plugin/extension called open in browser, this would make the process a lot easier”

Right click - inspect element to see details…

what we see in the elements inspector is what's known as the DOM,

or the **D**ocument **O**bject **M**odel.

This is essentially a tree of objects our web browser uses to model our HTML.

DOM = document object model

A bunch of html tags

AD: “Use this link to inspect the DOM object and by opening developer tools on chrome and view the html tags on the page

<file:///C:/07UT%20Data%20Bootcamp/Curriculum%20Repo/DataViz-Lesson-Plans/DataViz-Lesson-Plans/01-Lesson-Plans/12-Web/1/Activities/03-Ins_HTML_Tags/Solved/html-tags.html> “

**### Class Objectives**

\* gain a high-level understanding of HTML, CSS, and JavaScript and what their roles are when creating websites.

\* understand the basic parts of an HTML web page and how to create one from scratch.

\* learn to cover and utilize some of the most common HTML tags and selectors.

1.1

1.2

1.3 html-tags.html review later

<file:///Users/David/Desktop/1.Bootcamp/12-Web/1/Activities/03-Ins_HTML_Tags/Solved/html-tags.html>

encounter an unfamiliar HTML element,

a great resource is [HTML5 Doctor](http://html5doctor.com/), which has an [Element Index](http://html5doctor.com/element-index/) we can look up unfamiliar tags with.

1.5-Stu\_FixThe\_HTML

<https://validator.w3.org/>

for exercise 5 to check code

You can input your html files in there and it checks for errors.

[W3 Validator](<https://validator.w3.org/#validate_by_input> ) and show students how we can

validate our HTML code by pasting it into this tool.

1.6 Tables

\* A good example of these semantic table elements being used can be found here:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/table#More_Examples>

to choose the appropriate semantic tag. HTML5 Doctor has an informative

[flowchart] (<http://html5doctor.com/downloads/h5d-sectioning-flowchart.pdf>

for deciding which semantic tag to use.

the `div` element can also be used to divide content and behaves the same way the `article` and `section` elements do when rendered to the page. The `div` tag, however, doesn't have any semantic meaning and should be used only when no other semantic element is appropriate.

9/24/18 – Mon. day 2 – AD - 12.2 Lesson Plan - Intro to GitHub Pages and CSS

**### Overview**

Today's class will focus on introducing students to styling HTML pages using basic CSS while also teaching them how to deploy the websites they create to GitHub Pages.

**### Class Objectives**

\* have a firm understanding of how to deploy HTML webpages to the internet using GitHub Pages.

\* understand the basics of CSS styling.

\* have a basic grasp on how to position HTML elements on a webpage using CSS.

2.1 -Stu\_HTMLBio

2.2 - Stu\_GithubPages**Personal** & 2.3 **Project**

deploy HTML webpages to the internet using GitHub Pages.

Setup github access to website…html doc…so it is not limited to just my computer

<https://github.com/dmarcus9/marcus.github.io.git>

LOST IT IN MIDDLE OF AD EXPLANTION Personal repo setup; project repo NOT

\* A web host is the activity or business of providing storage space and access for websites. You cannot put a website online without it being hosted on a server somewhere.

turn GitHub repositories into live webpages without having to worry about pushing our code to another web host provider.

\* the steps of **creating a personal website using GitHub Pages**...

1. Create a new repository on GitHub called "\_username\_.github.io"

where \_username\_ is your account name on GitHub…for personal

Project = name this repository whatever you would like.

2. Next, open up Terminal on your computer. Navigate into the folder that you would like to store your project in and then clone the repository you just created.

3. Within this new folder, add an HTML file called "index.html" which contains the code for the website you would like to publish.

4. Add, commit, and push your changes to the repository and... That's it! Whenever anyone navigates to "\_username\_.github.io" they will now land on your webpage!

5. Go to the settings tab in the repo and scroll down to GitHub pages to confirm the page was published.

Project = in the section labeled "Source",

select that you would like to use the master branch as your source.

6. Finally click the link or navigate to `https://\_username\_.github.io` to visit the webpage.

Project = to "\_username\_.github.io/\_repositoryname\_"

**### Troubleshooting Guide from lesson plan**

Below is a list of the most common issues that students present when trying to do Github Pages deployments.

\* \*\*Forgetting to "git add", "git commit -m":\*\* Often students will completely skip the step where they save and commit their changes prior to pushing to GitHub. This will mean their web page is essentially blank. As a starting point, ensure their code is present in GitHub.

\* \*\*Didn't name the repo correctly:\*\* Students will likely not name the repository for their custom site correctly - ensure it follows the pattern `\_username\_.github.io`

\* \*\*Images and/or CSS not appearing:\*\* All filenames and paths are case sensitive. Ensure that all links in HTML are using case-sensitive paths that match the folder directories casing.

\* \*\*Not using relative paths:\*\* Many students are still using absolute paths to reference their CSS or image files. Help them to convert these to relative paths.

\* \*\*Not knowing where their site deployed:\*\* Show students that they need to login to the site and they will see the new app deployed on their menu. Give them guidance as to what the URL for their repo will be.

wondering how to get a **custom domain** ??? for projects as opposed to a site that is clearly linked to their GitHub account...

\* custom domains are more heavily coveted since they are more easily searchable online. This means that custom domains have to be purchased from companies known as "DNS Providers". These companies allow users to buy and register unique domain names and connect that name to an IP address. \*\*GitHub Pages does not sell domain names.\*\*

2.4-Ins\_BasicCSS

CSS = “**C**ascading **S**tyle **S**heets”.

**CSS** describes how and where elements should appear on the page

it is a computer language which is used to "format" HTML.

In simpler terms, CSS is a presentation language which tells web browsers how the content of a particular page should look… appearance

\* While **HTML** was developed to describe the **content** of a webpage,

**CSS** was developed to present what that content should look like.

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Great CSS Resource

<https://www.lifewire.com/types-of-css-styles-3466921>

**Inline, Embedded, and External Style**

**Inline** styles are styles that are written directly in the tag in the HTML document.

within a pair of `<style>` tags

applies only to the individual tags the `style=""` attribute is placed inside.

Rarely used.

**Embedded** styles are styles that are embedded in the head of the document.

Embedded styles affect only the tags on the page they are embedded in.

Most websites today use **external** style sheets. External styles are styles that are

written in a separate document and then attached to various web documents.

contains CSS rules that can then be applied to multiple tags/elements.

benefits you get from having your content and presentation separate from one

another, making the code that much harder to maintain.

External stylesheets can be changed out more easily than having to rewrite every CSS rule

reference external stylesheets in HTML using a `<link>` tag.

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

\* Make certain to point out the **syntax of CSS** once more

![CSS Syntax](./Images/CSS-Syntax.gif)

\* Selector points to the HTML element you would like to style

\* Declaration blocks are bounded by curly-brackets

\* Each declaration block is separated by semicolons

\* Each declaration includes a CSS property and a value that is separated by a colon

2.5-Stu\_DullCorp

External css style sheet…

2.6-Stu\_TargetedCSS

**div** elements are used to group elements into visually related segments while **section** elements define a specific part of a page and thus should be used as a container element regardless of styling.

Container elements like **div** and **section**, combined with classes and ids, allow users to group and style HTML elements in chunks. This is especially useful in positioning.

HTML classes and ids allow us to pick and choose which HTML elements we want to style in particular ways.

To create an HTML class, place a ` class="((className))" ` attribute within an HTML element. To **reference that class** within the CSS, simply put a **period** in front of \_className\_ in

your stylesheet.

Html = <h1 class="purple small-font">MAKE ME PURPLE</h1> css = **.**purple {color: purple;}

To create an HTML id, place a `id="((idName))"` attribute within an HTML element.

To **reference that id** within the CSS, put a **hashtag** in front of \_idName\_ in your stylesheet.

Html = <h1 id="red" class="back-black small-font">MAKE ME RED</h1> css = **#**red {color: red;}

**Bootstrap** is a css framework, that makes classes for html, so you don’t have to write css

When starting an html doc, use:

## Starter template from

<http://getbootstrap.com/docs/4.1/getting-started/introduction/>

GET AUTO COMPLETE STARTED HTML CONNECTED TO BOOTSTRAP in Atom

<http://getbootstrap.com/docs/4.1/layout/grid/>

<https://www.w3schools.com/tags/default.asp>

Explain that each HTML element is bounded by a "box" that is split up into 4 distinct parts:

Margin

border

padding

content

2.7-Ins\_CSSPositionedLayout

<link rel="stylesheet" href="**reset**.css">

Resets each time

<link rel="stylesheet" href="**static**.css">

Static placement = default

<link rel="stylesheet" href="**relative**.css">

nudge the boxes in relation to their “original” location.

<link rel="stylesheet" href="**absolute**.css">

An element that is positioned absolutely is taken out of the flow and thus takes up no space when placing other elements.

The absolutely positioned element is positioned relative to *nearest positioned ancestor (non-static)*.

If a positioned ancestor doesn't exist, the initial container (body) is used.

<link rel="stylesheet" href="**fixed**.css">

Position with exact coordinates to the browser window

<link rel="stylesheet" href="**zindex**.css">

Like z axis…allows you to layer elements on top of each other when they’re positioned (NOT static).

<link rel="stylesheet" href="**invisible**.css">

“display: none;” allows us to **hide** elements from view.

We can use this to show/hide elements based on user input later …

2.8-Stu\_AimedPositioning

Position things absolutely w %s and top left right bottom

h1 {position: absolute; top: 50%; left: 45%;}

Or with margins

h1 {margin-top: 20%; margin-left: 42%;}

\* Positioning can be used to better place elements without having to move them around in the HTML. They can also be placed on the same line far more easily and, if you use percentages, are more reactive to the viewport's size.

\* Using the box model alone makes the exact placement of separate elements quite difficult since they cannot easily be placed on the same line.

\* Using the box model alone to position elements is also heavily frowned upon and should be avoided when possible.

\* Floats and clears are useful but can be quite difficult to pick up initially. They are more situationally useful than positioning in cases where you would like text to wrap around an element.

2.9-Stu\_StudentBio

CREATE 1…

9/26/18 – Mon. day 3 – AD

color picker for VS code:

<https://marketplace.visualstudio.com/items?itemName=anseki.vscode-color>

chrome color picker extension:

<https://chrome.google.com/webstore/detail/colorpick-eyedropper/ohcpnigalekghcmgcdcenkpelffpdolg>

for firefox:

<https://addons.mozilla.org/en-US/firefox/addon/colorzilla/>

\* You can read about the grid system here:

<http://getbootstrap.com/css/#grid>

<https://getbootstrap.com/docs/3.3/components/>

### Overview

In today's class, we introduce students to the Bootstrap CSS framework.

### Class Objectives

\* become familiar with different Bootstrap components & understand the Bootstrap grid.

discover how to utilize it to position the elements on the page.

\* have a high-level understanding of / be able to discuss media queries:

the technology that is used to create the responsive Bootstrap grid.

\* have an understanding of how to use different column sizes to make websites

responsive for smaller screens and mobile devices.

\* Students will discover how to quickly and easily build web pages using pre-built Bootstrap components.

3.1-Stu\_ReviewActivity

3.2-Stu\_ChromeDevtools

we can change the text inside of the HTML here by changing the values inside the

element inspector

inside Chrome Dev Tools / Mozilla

Also, modify some of the CSS values in the right sidebar of the element inspector.

we can make changes locally to our HTML and CSS files using Chrome Devtools, and then copy/paste those changes into our actual HTML and CSS files when we're satisfied.

We can test changes quickly without having to refresh our browser or edit our actual document right away if we're unsure about making a change.

3.3-Ins\_MediaQueries

\* we can get an idea of how our web pages would look on smaller sized devices by dragging the browser window to shrink the page's width.

\* You'll notice that once we reach a certain point, our navbar doesn't look right and the padding we used to have on the sides of each article makes it harder to read.

\* Thankfully we can use **media queries** to fix this…there isn't much to them - they essentially tell the browser to apply additional styles if a given condition is met.

To that end, we can use them to alternate and override existing styles conditionally.

/\* Here we define some styles that will ONLY be applied at screen sizes below 480 pixels \*/

@media (max-width: 480px) {.box-1 {background: purple;}

@media is a keyword in CSS that means we're about to define some styles that are only going to be applied when our device is a specified size or type.

In this case, only when the viewport (device screen) is under 480px.

We can also declare whether we want this to happen on just screens or only when printing as well. We can add as many media queries as we want and at any screen size.

this works because we're defining our media queries after our base styles.

These new media query styles override the old styles because they come after them inside our CSS document.

3.4-Stu\_MediaQueries

\* A reference for media query syntax can be found at [MDN Docs](<https://developer.mozilla.org/enUS/docs/Web/CSS/Media_Queries/Using_media_queries> )

\* You can create a drop-down menu using the `select` and `option` elements. See [MDN Docs](<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/select> ) for `select` elements.

\* You can show and hide elements on a web page with the visibility property. See [MDN Docs](<https://developer.mozilla.org/en-US/docs/Web/CSS/visibility> )

it's important to understand how media queries work, at least on a high level, but we probably won't have to use them too frequently because of CSS frameworks.

3.5-Ins\_BootstrapDemo

\* **Bootstrap** helps us write front end HTML and CSS much more quickly because it provides us with a few features such as:

\* A responsive 12 column grid.

\* Rather than specify exact pixel locations we want specific elements to appear, we can instead define where we want to add our element inside of the grid.

Because the Bootstrap grid is responsive out of the box, we'll automatically have decent looking web pages on mobile without any extra work (although we'll want to tweak a few things here and there depending on how we want our content to be displayed)

\* Bootstrap offers us dozens of pre-built components we can use right away such as navbars, buttons, thumbnails, tables, and more.

We have these components available to us at [Bootstrap's Website](<http://getbootstrap.com/components/> ) to copy and paste into our apps.

\* We no longer need to worry about including a **reset.css file**, since Bootstrap normalizes CSS across various browsers, giving us a consistent looking web page on every device.

\* Bootstrap also includes various JavaScript components we can take advantage of such as sleek looking drop-down boxes and modals.

\* Explain to students that while Bootstrap does give us a lot, we want to further customize the provided components by adding additional CSS to our stylesheets.

\* link to the [Bootstrap Expo](<http://expo.getbootstrap.com/> )

spend a few minutes browsing a few of the featured websites built with Bootstrap.

how we include Bootstrap into a project, navigate to the [Bootstrap Getting Started Page](<https://getbootstrap.com/docs/3.3/getting-started/> ) and see where we can copy the Bootstrap CSS CDN.

\* Explain to students that CDN stands for \*\*C\*\*ontent \*\*D\*\*elivery \*\*N\*\*etwork. Essentially a CDN is a network of distributed servers designed to handle large amounts of traffic and deliver content to users based on their geographic location.

A CDN link will typically provide fast download speeds.

Additionally, this allows us to include Bootstrap without having to manually download the entire framework to our computers every time we wanted to use it.

\* Columns go inside rows. Rows sit inside containers.

\* A row is comprised of up to 12 columns.

\* Don't alter the Bootstrap grid. i.e. don't add new CSS rules directly to `container`, `row`, or `col-\*` (column) classes.

3.6-Stu\_LoremGrid

\* You can read about the grid system here: < <http://getbootstrap.com/css/#grid> >

3.7-Stu\_BootstrapComponents

[Bootstrap Components Docs](<http://getbootstrap.com/components/> )

3.8-Ins\_ResponsiveCols

\* Notice how when you shrink the browser window,

the columns resize twice at small and extra small screen sizes.

\* Notice how multiple column classes are applied to each column inside of `index.html`.

Each class's styles take over at the appropriate screen sizes.

3.9-Stu\_CloneAWebsite

**## Unit 12 - The Web - from StudentGuide.md**

This week we will cover the anatomy of a website. Students will build websites using HTML and add styles to them with CSS. Students will also learn to use the Bootstrap CSS framework and deploy their websites to Github Pages.

**### Objectives**

\* Gain a high-level understanding of HTML, CSS, and JavaScript and what their roles are when creating websites.

\* Understand the basic parts of an HTML web page and how to create one from scratch.

\* Learn to cover and utilize some of the most common HTML tags and selectors.

\* Understand how to deploy HTML webpages to the internet using GitHub Pages.

\* Understand the basics of CSS styling.

\* Position HTML elements on a webpage using CSS.

\* Be able to discuss media queries, the technology that is used to create the responsive Bootstrap grid.

\* Understand the Bootstrap Grid and discover how to utilize it to position the elements on the page.

\* Discover how to quickly and easily build web pages using pre-built Bootstrap components.

**### Helpful Links**

\* [Bootstrap 4 Tutorial](https://scrimba.com/g/gbootstrap4)

\* [Codecademy HTML & CSS](https://www.codecademy.com/learn/web)

\* [Mozilla HTML Docs](https://developer.mozilla.org/en-US/docs/Web/HTML)

\* [Github Pages](https://pages.github.com/)

\* [Bootstrap](<https://getbootstrap.com/>)

**## Unit 12 – Web** **### Overview from README.md…mixed in w above**