**HTML/CSS/JavaScript Notes**

Key:

* Element Tags
* Attributes
* CSS Style Property
* Values
* Other

Note: Code examples do not conform to these colors.

Attributes provide additional information for HTML elements.

* Ex: “style”, “href”, “id”

The “style” attribute provides visual styling for elements. Style attributes are defined in “property: value” pairs (similar to key: value pairs in python).

* “color: green;”, “margin: 10px;”

**Element Types**

There are two types of elements:

Block-level: Starts on new line and takes up full width available. Examples: <div>, <h1>, <li>

Inline: Does not start new line and only takes up necessary width. Examples: <span>, <a>, <strong>

**Commenting Code**

Begin a comment with “<!--” and end with “-->” to leave code comments.

**HTML Colors**

HTML colors can be defined with predefined color names, HEX, RGB, RGBA, HSL, and HSLA.

Examples:

- Using HEX codes:

color: #4286f4;

- Using RGB:

color: rgb(66, 134, 244);

- Using RGBA (last number is the alpha/opacity level from 0-1):

color: rgb(66, 134, 244, 0.5);

**HTML Entities**

HTML entities are a way to display characters reserved for HTML such as "<". To display entity numbers you would use the following formula: &entity\_name; OR &#entity\_number;

Example of using the HTML entity for "<":

&lt; or  &#60;

A useful HTML entity is the non-breaking space or &nbsp;

It is used where you do not want a space in between characters to result in a line break that separates the characters.

Examples: 10 PM or 10 MPH

**HTML Links**

There are different states of a <a> element:

a:hover - Active when the users mouse cursor hovers over link.

a:visited - Active when link has been visited.

a:active - Active when link is being pressed.

**Attributes of <a>:**

"target" - Specifies where to open link.

“target” values:

* “\_blank” - Opens link in new window/tab.
* “\_self” - Opens link in current window/tab. (Default)
* “\_parent” - Opens the link in the parent frame.
* “\_top” - Opens the link in the full body of the window.
* “framename” - Opens the link in a named frame.

**Bookmarking a Specific Page Position**

You are able to add a bookmark to a certain element by using the “id” attribute. Then using the <a> element to link to it.

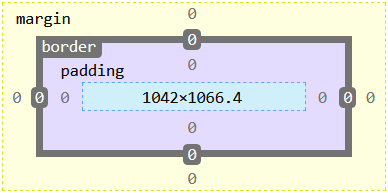
<h1 id=”my\_bookmark”>Bookmarked</h1>

<a href=”#my\_bookmark”> (if calling from a separate HTML file: <a href=”my\_page.html#my\_bookmark”>)

**Other Notes**

You can wrap images in links by placing <a> tags around the <img> tag.

**Composition of an HTML Element**



**Tags/Elements**

|  |  |
| --- | --- |
| **Tag** | **Definition** |
| <hr> | A divider in the page (horizontal ruler) |
| <pre> | Keeps spaces and indents in text |
| <small> | Creates smaller text |
| <mark> | Creates highlighted text |
| <del> | Creates text with line through middle |
| <ins> | Creates underlined text |
| <a> | Creates a Hyperlink |
| <q> | Creates a quotation mark |
| <bdo> | Used with “dir” attribute to change direction of text. (dir=”rtl” meaning right-to-left) |
| <abbr> | Used with “title” attribute to give the full title to an abbreviation |

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

**Tables**

- Tables are defined with <table>

- Each table row uses <tr> (including the table headings, but can use <thead>)

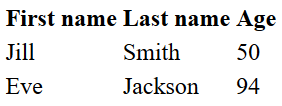
- Each table heading uses <th>

- Each table data cell uses <td>

Example of Table:

|  |
| --- |
| <table>   <tr>     <th>First name</th>     <th>Last name</th>      <th>Age</th>   </tr>   <tr>     <td>Jill</td>     <td>Smith</td>      <td>50</td>   </tr>   <tr>     <td>Eve</td>     <td>Jackson</td>      <td>94</td>   </tr> </table> |

Result:



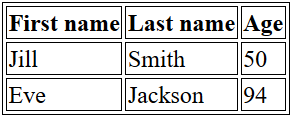
**Table Borders**

To create a border around a <table> element use the “border” style property.

Example:

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

Result:

****

Add the “border-collapse” style property to have only one line border.

Example:

|  |
| --- |
| table, th, td {     border: 1px solid black;     border-collapse: collapse; } |

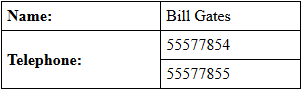
**Tables Spanning Multiple Rows/Columns**

Tables can span multiple rows/columns by using either the “colspan” or “rowspan” attribute in either a <th> element or an <td> element.

Example (also using a vertical heading):

|  |
| --- |
| <table style="width:100%">   <tr>     <th>Name:</th>     <td>Bill Gates</td>   </tr>   <tr>     <th rowspan="2">Telephone:</th>     <td>55577854</td>   </tr>   <tr>     <td>55577855</td>   </tr> </table> |

Result:



**Adding a Caption to a Table**

Use a <caption> tag below the <table> tag to add a caption above the table.

**Lists**

There are two main types of lists:

1. <ol> - Ordered list (uses numbers by default)
2. <ul> - Unordered list (uses bullet points by default)

Items in a list are marked with the <li> tag.

The numbering scheme can be changed on <ol> by using the “type” attribute.

* type=”1” - Default.
* type=”A” - Marked with uppercase alphabet letters.
* type=”a” - Marked with lowercase alphabet letters.
* type=”I” - Marked with uppercase roman numerals.
* type=”I” - Marked with lowercase roman numerals.

You can set the number to start counting from using the “start” attribute with <ol> elements.

The bullet type on <ul> can be changed using the style property “list-style-type”.

* list-style-type: disc; - Default.
* list-style-type: circle; - Changes bullet to an empty circle.
* list-style-type: square; - Changes bullet to a square.
* list-style-type: none; - Gets rid of bullet.

**Horizontal lists (with CSS)**

Horizontal lists are a popular way to create a navigation menu using <ul>, <a>, and CSS styles.

Example:

|  |
| --- |
| <head> <style> ul {     list-style-type: none;     margin: 0;     padding: 0;     overflow: hidden;     background-color: #333333; }  li {     float: left; }  li a {     display: block;     color: white;     text-align: center;     padding: 16px;     text-decoration: none; }  li a:hover {     background-color: #111111; } </style> </head> <body>  <ul>   <li><a href="#home">Home</a></li>   <li><a href="#news">News</a></li>   <li><a href="#contact">Contact</a></li>   <li><a href="#about">About</a></li> </ul>  </body> |

Result:

list_navbar

Note: There are also descriptive lists but they are not commonly used.

**Classes and IDs**

The “class” attribute is used to define equal styles for elements with the same class name.

The “id” attribute is used to define a style for a unique element. Should be unique to only one element.

**Referencing Classes/IDs with CSS Styles**

A “class” attribute is referenced with a “.” preceding the name of the class.

A “id” attribute is referenced with a “#” preceding the name of the class.

Example of a “class” with the name value of “cities”:

|  |
| --- |
| <style> .cities {     background-color: black;     color: white;     margin: 20px;     padding: 20px; }  </style>  <div class="cities">   <h2>London</h2>   <p>London is the capital of England.</p> </div> |

The above example can be assigned to any eligible elements by using the “class” attribute with the element.

Example of a “id” with the name value of “cities”:

|  |
| --- |
| <style> #cities {     background-color: black;     color: white;     margin: 20px;     padding: 20px; }  </style>  <div id="cities">   <h2>London</h2>   <p>London is the capital of England.</p> </div> |

There are two ways to specify what elements are affected by a class.

* element.class - This will select all elements with class.
* .class element - This will select all elements that are descendants of elements that have that class.

NOTE: .class element method does not affect the actual element, only the descendants. Used with <div> elements.

Example using the element.class method with a “class” with the name value of “note” being used on a <span> element:

|  |
| --- |
| <style> h1.note {     font-size: 120%;     color: red; } </style>  <h1 class="note">My Important Heading</h1> |

Example using the .class element method:

|  |
| --- |
| <style> .note h1 {     font-size: 120%;     color: red; } </style>  <div class="note">  <h1>My Important Heading</h1>  </div> |

Another example using the .class element method. It saves you from having to put classes on each <li> items:

|  |
| --- |
| <style> ul.note li {     float: left; } </style> |

NOTE: The same examples above are true with “id” attributes.

**Forms**

- The <form> element defines a form that is used to collect user input.

- The elements inside a form are called form elements.

- The <input> element is the most important form element and can be displayed in several different ways depending on the “type” attribute.

**Different “type” values:**

* “text” - Defines a one-line text input field.
* “radio” - Defines a radio button (for selecting one of many choices).
* “submit” - Defines a submit button (for submitting the form).
* “password” - Defines a text input field for passwords (will hide the input).
* “email” - Defines a text input field that must be an email.

**Using the Text Input**

Attributes of a text form:

* “name” - A variable to store the user input (also known as the “value” attribute). A “name” attribute must be given or otherwise the field will not be submitted.
* “value” - The submitted user input. If a “value” attribute is given a value it will be the default text in the text input.

Example:

|  |
| --- |
| <form>   First name:<br>   <input type="text" name="firstname"><br>   Last name:<br>   <input type="text" name="lastname"> </form> |

**Using Radio Buttons**

A form that uses radio buttons should have the attributes “name” and “value” defined.

- The “name” attribute links the other radio buttons together. All radio buttons must share the same “name” attribute to be used with each other.

- The “value” attribute is the value that is returned for the user selection when the form is submitted. This is often displayed in the URL.

Example:

|  |
| --- |
| <form>   <input type="radio" name="gender" value="male" checked> Male<br>   <input type="radio" name="gender" value="female"> Female<br>   <input type="radio" name="gender" value="other"> Other </form> |

The “checked” attribute specifies if a radio button should be selected by default.

**Using a Drop-down Menu**

A drop-down menu is created using <select> tags and <option> tags.

- The <select> tag establishes the drop-down menu. A “name” attribute is commonly given with the <select> tag.

- The <option> tag creates an option in the drop-down menu. A “value” attribute will display the users option in the URL. The displayed text will be between the <option> tags.

Example:

|  |
| --- |
| <form>   <select name="stars">   <option value="Great">3</option>   <option value="Okay">2</option>  <option value="Bad">1</option>  </select>  </form> |

**Using the Submit Button**

The submit button defines a button for submitting the form data to a form-handler. The form-handler is specified in the <form> elements “action” attribute.

The “action” attribute defines the action to be performed when the form is submitted. A website URL can be specified as the value to go to when the form has been submitted.

The “target” attribute specifies if the results will open in a new browser tab, a frame, or in the current window.

The “method” attribute specifies the HTTP method to be used when submitting the form data:

* “get” - Form data will be visible in the page address field. Non-secure. Can be bookmarked.
* “post” - Can be used to send large amounts of data. Cannot be bookmarked.

The “value” attribute is the text that will display on the submit button.

Example:

|  |
| --- |
| <form action="/action\_page.php" target**=**"\_blank" method**=**"get">   First name:<br>   <input type="text" name="firstname" value="Mickey"><br>   Last name:<br>   <input type="text" name="lastname" value="Mouse"><br><br>   <input type="submit" value="Submit"> </form> |

**Reading the URL**

After submitting a form the “name” attribute will appear in the URL with the “value” of the input.

So let’s use a form with a text input with the “name” attribute equal to “username”. If the user submits the form with the “value” of “testname” the URL would read the following:

http://www.website.com/?username=testname

If there are multiple inputs they will be separated with the & character. Example:

<http://www.website.com/?username=testname&otherinfo=othername>

**Using Labels**

To group text on the same line as your <input> element you can use a <label> tag.

- The “for” attribute in a <label> is used to link the <input> element with the label.

- Link the <label> and <input> by using the same value of the “for” attribute in the <label> as the “id” attribute in the <input>

Example:

|  |
| --- |
| <form>  <label for="userinput">Enter Input:</label>  <input id="userinput" type="text" name="" value="">  </form> |

**Useful Attributes**

* “placeholder” - A default value that will be displayed in a input that will be erased when the field is used.
* “required” - Specifies that an input has to be given a eligible value/not left blank. (To use: required=””)

**CSS Styles**

HTML elements are modified with CSS by using property:value pairs.

End each property:value pair with “;”.

To comment code inside CSS files use “/\*” at the beginning of code and “\*/” at the end. Example: /\* This code is commented \*/

There are three ways CSS styles can be added to elements:

* Inline - By using the “style” attribute directly in the element tag.
* Internal - By using the <style> element inside the <head> element.
* External - By using an separate CSS file (called in a <link> element).

**Using Inline CSS Styles**

Directly modify the element inside the elements tag.

Example:

|  |
| --- |
| <h1 style="color:blue; margin-left:30px;">This is a heading</h1> |

**Using Internal CSS Styles**

Can modify any eligible element within the <style> tag by calling on the elements selector.

Example:

|  |
| --- |
| <head> <style> body {     background-color: linen; }  h1 {     color: maroon;     margin-left: 40px; }  </style> </head> |

**Using External CSS Styles**

Uses a “.css” file to style the web-page. The file should not contain any HTML code. The file would be the equivalent of what is in the <style> tags.

Example:

|  |
| --- |
| <head> <link rel="stylesheet" type="text/css" href="mystyle.css"> </head> |

**Referencing Classes/IDs**

To select classes that you want to modify place a “.” before the class name.

To select IDs that you want to modify place a “#” before the ID name.

Example:

|  |
| --- |
| <head> <style> .my\_class {     background-color: linen; }  #my\_id {     color: maroon;     margin-left: 40px; }  </style> </head> |

**Using Selectors**

Selectors are patterns used to select the element(s) you want to style.

**Commonly Used Selectors**

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Example description** |
| \* | \* | Selects all elements |
| .class | .my\_class | Selects all elements with class=”my\_class” |
| #id | #my\_id | Selects the element with id=”my\_id” |
| element | p | Selects all <p> elements |
| element, element | div, p | Selects all <div> elements and all <p> elements |
| element element | div p | Selects all <p> elements inside <div> elements |
| :hover | a:hover | Selects links on mouse over |
| :focus | input:focus | Selects the input element which has focus |

More selectors at: [https://www.w3schools.com/cssref/css\_selectors.asp]( https:/www.w3schools.com/cssref/css_selectors.asp)

**Selecting Input Elements**

You can also modify different <input> types such as the text input fields and submit buttons.

Example:

|  |
| --- |
| input[type="text"] {     width: 150px;     display: block;     margin-bottom: 10px;     background-color: yellow; }  input[type="button"] {     width: 120px;     margin-left: 35px;     display: block; } |

**Common CSS Properties**

* “border” - Creates a border following the formula: {border-width} {border-style} {border-color} **Ex: “border: 2px solid black;”**
  + “border-style” - “dotted”, “dashed”, “solid”, “double”, “groove”, “ridge”, “inset”, “outset”
* “border-radius” - Rounds the corners of a border. **Ex: “border-radius: 2px;”**
  + Defining 2 corners: 1st value = top-left and bottom-right corners, 2nd value = top-right and bottom-left corners.
  + Defining 4 corners: top-left, top-right, bottom-right, and bottom-left.
* “color” - Changes the color of text. **Ex: “color: blue;”**
* “font-family” - Chooses what font to use. **Ex: “font-family: “Arial;”**
* “font-size” - Specifies the size of the font. **Ex “font-size: 20px;”**
  + Font sizes: “vw” (1vw = 1% of viewport), “em” (1em = current set font size. Default is 16px. So 2em = 32px), “px”
* “font-weight” - Specifies weight of a font. **Ex: “font-weight: bold;”**
* “font-variant” - Variations of fonts. **Ex: “font-variant: small-caps;”**
* “text-align” - Aligns text. **Ex: “text-align: center;”**
* “opacity” - Can be used on images to control opacity. **Ex: “opacity: 0.5;”**
* “text-shadow” - Creates a shadow around text following the formula: {horizontal shadow} {vertical shadow} {blur effect} {color}. **Ex: “text-shadow: 2px 2px 4px red;”**
* “box-shadow” - Creates a shadow around elements following the formula: {horizontal shadow} {vertical shadow} {blur effect} {color}. **Ex: “box-shadow: 10px 10px 5px grey;”**
* “transition” - Changes a property value smoothly over a given duration. Placed on the original element selector (Ex. Use on element and not “:hover” selector). **Ex. “transition: 0.2s;”**
* “linear-gradient” - Creates a gradient following the formula: {direction} {color-stop1} {color-stop2}. **Ex: “background: linear-gradient(to right, red, yellow);”**

**Positioning Elements**

The “position” property specifies the type of positioning method used for an element.

There are five different position values:

* “static” - The default positioning.
* “relative” - Positioned relative to its normal position using the “top”, “right”, “bottom”, and “left” properties.
* “fixed” - Positioned relative to the viewport, which means it always stay in the same place even if the page is scrolled. Also uses the “top”, “right”, “bottom”, and “left” properties.
* “absolute” - Positioned relative to the nearest positioned ancestor. Also uses the “top”, “right”, “bottom”, and “left” properties.
* “sticky” - Positioned based on the user’s scroll position.

**Using Relative Position**

Using the “top”, “right”, “bottom”, and “left” properties of a relatively-positioned element will cause it to be adjusted away from its normal position.

Example:

|  |
| --- |
| div.relative {     position: relative;     left: 30px;     border: 3px solid #73AD21; } |

Result:

relative_pos

**Using Fixed Position**

Using the “top”, “right”, “bottom”, and “left” properties of a fixed-positioned element will move the element relative to the viewport.

For example, if an element with “top” and “left” properties with a value of “0”, the element will stay in the top, left corner of the page even if the user scrolls through the page.

If using a fixed position with a nav-bar element that is fixed to the top, left of the page you will need to have a “margin-left” property with the value of the width of the nav-bar.

**Using Absolute Position**

**Using Sticky Position**

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it “sticks” in place (like position:fixed).

Example:

|  |
| --- |
| div.sticky {     position: -webkit-sticky; /\* Safari \*/     position: sticky;     top: 0;     background-color: green;     border: 2px solid #4CAF50; } |

**Bootstrap**

Bootstrap is a framework for developing responsive websites.

**Containers**

Bootstrap uses containers to to layout elements of web-pages and is required when using the bootstrap default grid system.

There are two types of containers:

“container” - Centers content to the middle of the screen.

“container-fluid” - Creates a container spanning the entire width of the viewport.

**Breakpoints**

Breakpoints use media queries to change the layout of the content depending on the users screen size.

Breakpoints defined at what point will the content begin stacking.

There are 5 different breakpoints:

“xs” - < 576px (portrait phones)

“sm” - < 576px (landscape phones)

“md” - < 768px (tablets)

“lg” - < 992px (desktops)

“xl” - < 1200px (large desktops)

Example:

In the case of three columns with “col-sm-4” the layout will stay the same from size sm and up. Once the screen hits the “sm” breakpoint the content will stack vertically.

**Multiple Breakpoints**

You can define multiple breakpoints for a row.

Example:

<div class=”col-lg-3 col-sm-6”>content</div>

<div class=”col-lg-3 col-sm-6”>content</div>

<div class=”col-lg-3 col-sm-6”>content</div>

<div class=”col-lg-3 col-sm-6”>content</div>

In the above example you have four equal sized columns when the screen is “lg”. When the “lg” breakpoint is reached the next defined layout is read and the layout goes to 2 stacked rows of 2 columns.

Once the screen is below the “sm” breakpoint the columns are stacked individually.

**Grid System**

Bootstrap uses a series of containers, rows, and columns to layout and align content. It uses the CSS flexbox.

The grid system usually consists of a “container” class <div>, followed by a “row” class <div>, and finally a “col” class <div>.

The grid system consists of 12 columns.

The formula for creating columns size and breakpoints are defined as: “col-{breakpoint}-{column-amount}”.

Basic structure of grid system:

|  |
| --- |
| <div class="container">  <div class="row">  <div class="col-sm">  One of three columns  </div>  <div class="col-sm">  One of three columns  </div>  <div class="col-sm">  One of three columns  </div>  </div>  </div> |

**Results of Different Grid Layouts**

|  |  |
| --- | --- |
| Code | Result |
| <div class=”col”> | In the case where the breakpoint and size are not defined, the **column will** **take up the full width of the container.** |
| <div class=”col”>  <div class=”col”> | In the case of two undefined columns, both columns will be equally sized filling up the width of the screen. The **columns will** **not stack when the screen size is changed.** |
| <div class=”col-sm”> | In the case of having a single column with a defined breakpoint but a undefined size, the **column will take up the full width of the container.** |
| <div class=”col-sm”>  <div class=”col-sm”> | In the case of multiple columns with a defined breakpoint and a undefined size, the **columns will be equally sized and will then stack once the breakpoint is hit.** |
| <div class=”col-sm-4”> | In the case of only one defined column, the **column will take up the first 1/3 of the container aligned to the left.** |
| <div class=”col-sm-4”>  <div class=”col-sm-4”> | In the case of two defined columns, each **column will fill 1/3 of the container aligned to the left.** |
| <div class=”col-sm-4”>  <div class=”col-sm-4”>  <div class=”col-sm-4”> | In the case of columns with a defined amount equal to 12, **each column will be equally sized.** |
| <div class=”col-lg-3 col-sm-6”>  <div class=”col-lg-3 col-sm-6”>  <div class=”col-lg-3 col-sm-6”>  <div class=”col-lg-3 col-sm-6”> | In the case of columns with multiple defined breakpoints/sizes, **the columns will abide by the largest defined breakpoint and continue to the next once the breakpoint is encountered.** (In this example, you begin with 4 equal sized columns and once the viewport is below the “lg” breakpoint the columns are stacked in 2 rows of 2 columns. Once the viewport is below the “sm” breakpoint the columns are individually stacked. |

**Spacing**

Spacing is used to modify distances of elements.

The spacing formatting is: {property}{sides}-{size}. Ex: “mb-4” = margin bottom size 4.

Spacing with no breakpoint defined applies to all classes.

Properties:

“m” - for classes that set margin.

“p” - for classes that set padding.

Sides:

“t” - top

“b” - bottom

“l” - left

“r” - right

“x” - both left and right

“y” - both top and bottom

Blank - If left blank will apply margin/padding to all 4 sides

Size:

0 - eliminates margin/padding.

1 - $spacer \* .25

2 - $spacer \* .5

3 - $spacer

4 - $spacer \* 1.5

5 - $spacer \* 3

“auto” - for classes that set the margin to auto.

NOTE: $spacer is a spacing variable defined in bootstrap.

NOTES TO ADD:

- Selectors (how to modify multiple selectors, how to use selectors with classes and ids, how to use selectors with event attributes)

(why do some classes use element\_name.class\_name and others .class\_name element\_name?)

(ex: p#special for a paragraph with an ID called “special”)

(ex:

ul#menu li {

float:left;

}

Is used to change all <li> elements inside <ul> with id of “menu” so you don’t have to put ids for each <li> item.)

(how to select <input> elements)

- Forms (go over what the “name” and “value”, etc attributes of input elements mean, other attributes such as placeholder, required)

(<select>, <textarea>, <button>, <datalist>, <output>)

- Style properties and values for elements (ex: <p>, <div>, <h1>, <ol>, <li>, <tr>, etc.)

- Making a website look good on monitor, tablet, or phone

- Flexboxes

- Floats and clears

- Importing fonts from fonts.google.com

Bootstrap:

- nav-bars

- aligning to right or left (mx-auto/my-auto)

- card