FRONTEND

- **HTML**
  - **Document** (individual elements combined to form an entire HTML page)
    - **Doctype**: <!DOCTYPE html>
      - Required as the first line of an HTML document (historical artifact).
    - **Root Element**: <html>
      - Follows the "doctype" and wraps around all content on the entire page.
    - **Head Element**: <head>
      - Container for things that do not appear as viewable content (e.g., keywords and descriptions that will appear in search results, CSS, character set declarations, etc.).
    - **Character Set**: <meta charset="UTF-8">
      - Allows document to use "utf-8" character set, which includes most characters from all known human languages (nests within head element).
    - **Title**: <title>
      - Sets the title that appears in browser tab (nests within head element).
      - Also appears as the search result in Google.
    - **Body**: <body>
      - Contains all of the content that will be shown to the viewer.
  - **Elements** (content + opening/closing tags)
    - **Block Elements** form a visible block on a page (e.g., paragraphs, lists, navigation menus, footers, etc.):
      - **Paragraph**: <p>
      - **Divider**: <hr>
      - **Headings**: <h1> through <h6>
        - **NOTE**: As a general rule, try to have only one <h1> tag in your HTML document, and it should be the biggest text element on the page.
      - Generic **Container**:
      - **Lists** (each item within a type of list needs to be identified by the "<li>" tag):
        - Ordered Lists (lists that are numbered):
        - Unordered Lists (lists composed of bullet points:
      - **Tables**: <table>
        - Table Row: <tr>
        - Table Header (consists of one cell within a row): <th>
          - Should be nested within <thead> under main table (semantics).
        - Table Data (consists of one cell within a row): <td>
          - Should be nested within <tbody> under main table (semantics).
        - Borders can be added by entering <table border="1">, although this is discouraged, as CSS should be used for styling.
      - **Forms** (interactive controls to submit information to a web server): <form>
        - Typically contain the "action" (the URL to send form data to) and "method" (the type of HTTP request, such as "GET" to receive information from the server and "POST" to add information to the serve) attributes, e.g.:
          <form action="/my-form-submitting-page" method="POST">
        - **Input** (used to accept data from the user): <input>

- The operation of <input> depends upon its type attribute. For a complete list of attributes, view **Form Input Types**. Examples:
  - **Text** (can be used for user names): type="text"
  - **Password**: type="password"
  - **Placeholder** (temporary text in input fields; used with "text" and "password" attributes): placeholder="insert-text-here"
  - **Button**: type="button" value="insert-text-here"
    - Simple **Submit** button: type="submit"
      - Alternatively, if placed at the end of a form, use the following to create an even simpler submit button:
        <button>insert-text-here</button>
  - **Checkbox** (square box for multiple choices): type="checkbox"
    - To have the checkbox already marked upon loading, add the attribute "checked" to the input.
  - **Radio Button** (circular icon for one choice): type="radio"
    - In order to make the user only able to select one choice, you must add the "name" attribute, which must be common among all choices.
    - The "value" attribute is necessary for the query string to understand the meaning behind each choice; otherwise, it will simply state "name=on".
    - Example:
      <label for="cats">Cats:</label>
      <input name="pet-choice" id="cats" type="radio" value="CATS">
      <label for="dogs">Dogs:</label>
      <input name="pet-choice" id="dogs" type="radio" value="DOGS">
- **Dropdown Menus**: <select>
  - For every possible option to select, use an <option> tag.
  - In order for the query string to understand that an option has been selected from the dropdown menu, the "name" attribute must be included in the <select> tag, e.g.:
    <select name="color">
      <option>White</option>
      <option>Black</option>
    </select>
  - If you want the query string to contain text other than "White" or "Black" in the example above, use the "value" attribute in the <option> tag, e.g.:
    <option value="happy">☺</option>
- **Text Areas** (multi-line plain-text editing control): <textarea>
  - You can specify how large the text area is by using the "rows" and "cols" attributes.

            - In order for the query string to process the data in the text area, you must use the "name" attribute.
            - Example:
              <textarea name="paragraph" rows="10" cols="50"></textarea>
        - **Labels** (add captions for individual items in a form): <label>
            - A label can be used by placing the control element inside the <label> element, or by using the "for" and "id" attributes:
                - For example,
                  <label>Username<input type="text"></label>
                  …is identical to:
                  <label for="username">Username</label>
                  <input id="username" type="text">
        - **Validations** ensure that users fill out forms in the correct format, e.g.:
            - The Boolean attribute "required" makes a field mandatory:
              <label>Username<input type="text" required></label>
                - Only works if the browser (like Chrome) allows it.
            - By changing type from "text" to "email", the browser will ensure that the field contains an @ symbol.
    - **Inline Elements** are contained within block level elements and do not cause new lines to appear:
        - Italics:
        - Bold:
        - Generic Container:
    - **BUT NOTE**: **Empty Elements** contain only a single tag:
        - Image: <img src="https://www.google.com/logo.png">
            - **NOTE**: Image width can be modified like so…
              <img width="50" src="https://www.google.com/logo.png">
              …but is discouraged, as styling should be done by CSS.
        - Input: <input type="text">
    - For a complete list of elements, view the **MDN Element Reference**.
    - **TIP**: In Sublime, press "Ctrl+Shift+D" to replicate an entire line of an element.
- **Attributes** (extra info; does not appear in content; target of style info)
    - Components:
        - Space between it and the element name (or prior attribute),
        - Attribute name followed by an equals sign, and
        - Attribute value surrounded by quotation marks.
            - Double or single quotes may be used, but must be done consistently. You can nest a single quote within a double quote (and vice versa), but if you want to nest the same type of quote, you must use **HTML Entities** (e.g., " or ').
    - Examples:
        - Class: content
        - Can be paired with the "anchor" element: <a>
            - Hyperlink with Title:
              content
    - **BUT NOTE**: **Boolean Attributes** can be written without a value, but technically always have only one value (generally the same as the attribute name:
        - Disabled: <input type="text" disabled="disabled">

        - Creates a text box in which typing is disabled.
        - May also be written as:
          <input type="text" disabled>
    - For a complete list of attributes, view the **MDN Attribute Reference**.
- **Entity References** (make special HTML syntax characters appear as normal text):
    - < = <
    - > = >
    - " = "
    - ' = '
    - & = &
- **HTML Comments** (write comments in the code that are invisible to the user by wrapping them in special markers):
    - <!-- and -->
    - **TIP**: In Sublime, you can (1) select text, and (2) hold "Ctrl+/" to turn text into comment.

- **CSS**
    - **The General Rule**

```
selector {
    property: value;
    anotherProperty: value;
}
```

        - For example, make all <h1> tags purple with 56-pixel font:

```
h1 {
    color: purple;
    font-size: 56px;
}
```

        - **Three Basic Selectors**
            - **Element** selectors select all instances of a given element. For example, "div" is a CSS element selector that will modify the properties of all HTML tags.
            - The **ID** selector selects a single element with an octothorp ("#") ID (only one per page). For example, the following HTML/CSS combination will result in the word "hello" appearing in yellow, while the word "goodbye" will remain as is:

```
<div>
    <p id="special">hello</p>
</div>
<div>
    <p>goodbye</p>
</div>
#special {
    color: yellow;
}
```

            - The **Class** selector selects all elements in a given class by functioning just like an ID selector; however, a class is instead prefaced with a period (".") . For example, the following items marked as "completed" on a "To Do List" will be crossed out with a line:

```
<div>
    <p class="completed">TASK #1</p>
</div>
```

```
<div>
    <p class="completed">TASK #2</p>
</div>
.completed {
    text-decoration: line-through;
}
```

- An element can be modified by multiple class or ID tags by simply adding a space between the two tag names, e.g.:
  `<p class="completed uncompleted">Text</p>`

- **Five More Advanced Selectors**
  - The **Star (*)** selector applies to every element on the page.
  - The **Descendant** selector applies to selectors that have been "nested" under another. For example, if you want to modify the color of only those <a> tags that are nested within the <li> tags of a <ul> list, use the following:

```
ul li a {
    color: red;
}
```

    - In addition to HTML tags, CSS selectors such as "ID" or "Class" may be used within a Descendant selector.
    - **HOWEVER**: If, for example, you have a second-tier <ul> nested within a first-tier <ul> that is nested within , and you only want to select the first-tier <ul> and not the second-tier, then you must use the ">" combinator to select only the **DIRECT** first-tier "child" (rather than the second-tier "grandchild") of :

```
#box > ul {
    color: red;
}
```

  - The **Adjacent (+)** selector will select only the element that comes **IMMEDIATELY** after another element (a "sibling" element, rather than a "nested" element). For example, to modify the font size of all <ul> tags that follow an <h4> tag (which are typed on the same "level" as the <ul> tags, and not nested under them), use the following:

```
h4 + ul {
    font-size: 24px;
}
```

    - If, in the above example, you want to select **ALL** tags after any <h4> tag, then use the more generalized sibling combinator of "~" instead of "+".
  - The **Attribute** selector will allow the selection of any element based off of any attribute. For example, to change the font family of all <a> tags that link to Google, use the following:

```
a[href="https://www.google.com/"] {
    font-family: cursive;
}
```

    - This selector an also be used to select all images of a particular source, or all inputs of a particular type, such as all checkboxes:

```
input[type="checkbox"] {
```

```
        border: 2px solid green;
    }
```

- **TIP**: See the complete list of **Attribute Selectors**.

- The **Nth-of-Type** selector takes a specific number and selects the "-nth" instance of an element. For example, to change the background color of every second <li> tag in every list (literally meaning the second tag, not every other tag), use the following:

```
li:nth-of-type(2) {
    background-color: rgba(100, 175, 225, 0.5);
}
```

- **NOTE**: To select every other tag, use the phrases (even) or (odd) instead of a specific number.

- For more advanced selectors, view **The 30 CSS Selectors You Must Memorize**.

- **CSS Location**
  - CSS should generally be saved to its own file, but can also be included in the HTML head by using the <style> tag:

```
<style type="text/css">
    li {
        color: red;
    }
</style>
```

  - The preferred method is to use a <link> tag in the HTML head to refer to the separate file containing CSS:

```
<link rel="stylesheet" type="text/css" href="directory/filename.css">
```

- **Specificity** is the means by which browsers decide which CSS property values are the most relevant to an element and, therefore, will be applied (e.g., if the body is styled to have red text, but a paragraph within the body is styled to have green text, then the text will be green because the green text style is more relevant to the specific paragraph than the general body).
  - The following list of selector types increases by specificity (in magnitudes of 10):
    1. Type selectors (e.g., li) and pseudo-element (e.g., :before)
    2. Class selectors (e.g., .hello), attributes selectors (e.g., [type="text"]) and pseudo-classes (e.g., :hover)
    3. ID selectors (e.g., #hello)
  - This **Specificity Calculator** may be used to test CSS specificity rules.
- **The Box Model**
  - In a document, each element is represented as a rectangular box. In CSS, each of these boxes is described using the standard "box model." Each box has four edges: (1) **Content Edge**, (2) **Padding Edge**, (3) **Border Edge**, and (4) **Margin Edge**. Padding is the space between the border and the element within the border, and the margin is the space between the border and everything outside of the border.
    - The content edge can be controlled by setting the "width" and "height" properties in "px" or "%" (with percentage in relation to the parent element), which in turn pushes out the border edge as well, as there is direct contact between the content and border (if no padding has yet been set).
      - **NOTE**: By using the "max-width" property in conjunction with "width", you can tell the browser to make an element's width a certain percentage, but then also cap that width to a maximum number of pixels, e.g.:

```
#container {
```

```
        width: 66.66%;
        max-width: 700px;
    }
```

- Space can be added between the content edge and border edge (and between the border edge and the next element's edge) by using the "padding" and "margin" properties respectively (in "px" or "%").
    - By default, padding and borders are set to go around all edges of an element, but can be limited by using more specific properties for top, right, bottom, and left—such as "padding-left" or "margin-top".
    - Alternatively, rather than typing a line of CSS for all four sides, the following shorthand can be used (with the first value setting the top property, and the remainder moving in a clockwise fashion):

```
p {
    margin: 20px 40px 60px 80px;
}
```

- **NOTE**: By setting the "margin" property to "auto" on the left and right, an element will automatically be horizontally centered:

```
p {
    margin: 0 auto 0 auto;
}
```

    - The above syntax can also be shorted as (with the first value representing the vertical axis, and the second value representing the horizontal axis):

```
p {
    margin: 0 auto;
}
```

- **Colors**
    - Colors can be created through the **Hexadecimal** system by combining the octothorp (#) with a string of 6 hexadecimal "numbers" from 0-F, e.g.:
      `color: #FF1493;`
        - This system follows an RGB scheme in which the first two numbers modify the amount of "red," the second two modify "green," and the last two modify "blue."
    - Alternatively, colors can be created through the **RGB** system: 3 channels consisting of red, green, and blue, with each ranging from 0-255, e.g.:
      `color: rgb(0, 255, 0);`
    - Colors can be made **Transparent** through the RGBA system. Just like RGB but with an alpha (transparency) channel ranging from 0.0-1.0, e.g.:
      `color: rgba(11, 99, 150, .6);`
- **Backgrounds**
    - Follows the same format as colors, e.g., `background: #FF6789;`
    - The background property can also set a background image, e.g.:

```
body {
    background: url(http://www.website.com/image.png);
}
```

        - To prevent the background from **Repeating** an image, add the following property:
          `background-repeat: no-repeat;`

- To allow the background image to **Stretch** out across the entire body, use:
  background-size: cover;

- **Borders**
  - Borders have three key properties: "width" (typically in pixels), "color" (as noted above), and "style" (generally solid, dotted, or dashed). All three properties must be present in order for a border to take effect, e.g.:

```
h1 {
    border-width: 5px;
    border-style: solid;
    border-color: purple;
}
```

  - The alternative shorthand syntax may also be used (in the specified order):

```
border: 5px solid purple;
```

- **Fonts**
  - **Font-Family** specifies the font for an element:

```
p {
    font-family: Arial;
}
```

    - While not always necessary, you may sometimes have to put quotation marks around the font name—particularly when the font name begins with a number.
    - **CSS Font Stack** shows what percentages of operating systems have a given system font (useful for choosing a safe bet on system compatibility).
      - However, rather than using those limited fonts, it is better to use **Google Fonts**, choose a font, and embed the font's stylesheet link in your HTML <head> prior to the CSS link, e.g.:
        <link href="https://fonts.googleapis.com/css?family=Esteban" rel="stylesheet">
  - **Font-Size** specifies how big the font appears (typically in pixels or "px"):

```
h1 {
    font-size: 25px;
}
```

    - Another size unit is "em", which dynamically sets font size in relation to a parent element. For example, if you want to make a section of a paragraph's text in tags be twice the size of the rest of the text in the <p> tags, you would say:

```
span {
    font-size: 2em;
}
```

      - **BUT NOTE**: What constitutes the "standard" 1em (i.e., the default font size on a page without CSS markup) varies from browser to browser, although the size is typically around **16 PIXELS**. To ensure uniformity among browsers, it is useful to set the body's font size at the outset.
      - **ALSO**: Similar to "em" is "rem", which—rather than setting font size in relation to the parent element—sets the font size in relation to the "root" element on the page (i.e., the default font size discussed above).
  - **Font-Weight** specifies how thick or thin the font appears.

        - Typically involves absolute values of "normal" or "bold", or relative (to parent) values of "lighter" and "bolder", but can also be assigned a numeric value in increments of 100 generally from "100" to "800" depending on the font itself.
    - **Line-Height** controls the height of a given line (similar to changing line spacing in Word to 1.5 or 2.0, which means that a larger font will result in larger spacing).
    - **Text-Align** controls where an element's text is aligned on the page (typically "left", "right", and "center").
    - **Text-Decoration** is used to give text effects such as "underline", "overline", or "line-through".
- **Float**
    - Normally, block level elements (such as) are stacked directly underneath the preceding element on the page. To change this, use the "float" property and specify a value of the direction in which the element should float ("left", "right", "none").
    - When an element is floated, it is taken out of the normal flow of the document (though still remaining part of it), and it is shifted to the left or right until it touches the edge of its containing box, or another floated element.
    - When <img> tags are laid out in a consecutive sequence, HTML automatically places some small amount of white space between the images. If you want to remove the white space, you can use "float" to remove that white space.