**Front-end Development**

Front-end development can be split up into three major coding languages.

**HTML** - HyperText Markup Language defines the structure of a website semantically and the content that will be rendered by the web browser.

**CSS** - Cascading Stylesheets deal with the presentation of content including aspects such as layout, formatting and colors.

**Javascript** - One of the most popular coding languages in the world, brings movement and user interaction to websites.

This Curriculum focuses on the first two cornerstones of modern front-end development: HTML & CSS.

**HTML**

HyperText Markup Language is made up of HTML Elements denoted using opening and closing tags:

<p>This is an example of a paragraph element in HTML</p>

These elements form the building blocks of an HTML document and may either directly introduce content or wrap content to provide information about document text.

For example, the above <p></p> tags wrap the containing document text into an HTML Element. The <img/> tag below is self-closing in that it does not wrap anything but instead directly inputs the provided image as an HTML Element.

<img src="appacademy.io/images/app-academy-logo.img" />

Notice that this image tag received additional information by declaring what is called an **attribute**. Attributes receive **values** in double quotes. The src attribute provides a url to the image tag for the browser to get and render the proper image inside the HTML Element. Mastering the various tags and attributes is the first key to becoming proficient in HTML.

**Tags**

The following are examples of popular tags in HTML and how they are used.

**Paragraphs and Headings**

<p>A paragraph tag is used to wrap a multi-line body of text</p>

<h1>A heading1 tag is used to denote the largest title on the page</h1>

**N.B.** The <h1> tag is used in site ranking algorithms by search engines such as Google. It is important to only have one <h1> element per page and to have it contain a keyword summarizing the page content.

<h2>Used for titles with a smaller font size than h1</h2>

These heading tags should be used in descending order with regards to their containing font sizes. The smallest size is a:

<h6>Heading six tag</h6>

**Lists**

Lists are used to contain a series of list elements. These may be ordered or unordered but must contain at least one element. There are three different types of HTML list tags. The following is an example of the first, most common type, the unordered list.

<ul>

<li>Unordered List</li>

<li>Ordered List</li>

<li>Definition List</li>

</ul>

The ordered list is usually used to display numbered list elements.

<ol>

<li>First</li>

<li>Second</li>

<li>Third</li>

</ol>

The definition list it not as common but is used to hold definition data and term elements.

<dl>

<dt>Unordered List</dt>

<dd>An HTML element made up of list elements in no particular order</dd>

<dt>Ordered List</dt>

<dd>An HTML element made up of list elements in order</dd>

<dt>Definiton List</dt>

<dd>An HTML element with definition term and definition data elements</dd>

</dl>

**Links**

What is commonly referred to around the internet as a "link" is a clickable element that directs the browser to another page or area on the page. This is not to be confused with a link tag, which will be discussed in future sections.

To create a link using an html element we use what is called an anchor tag:

<a href="http://appacademy.io">App Academy</a>

This clickable anchor tag element will link the user the URL provided through the href attribute. The text within the anchor tag element is the what appears as clickable on the page.

In the following code we create two anchor tags. One is self-closing with a name attribute and the other contains text that has an href attribute with the same name value preceded by a #. This is how we create in-page links. Clicking the 'Back to top' link wherever it is placed on the page will now take the user to the location of the named anchor tag.

<a name="top-of-page" />

<!-- Imagine all of your other content in between. By the way, this is how we denote comments in HTML. -->

<a href="#top-of-page">Back to top</a>

The use of the # in the beginning of the url tells the browser to find an element on the page with the given name attribute instead of making a request to a URL.

**Images**

We create image elements in HTML using a self-closing image tag with the src attribute referring to the path to the image data.

`<img src="http://appacademy.io/images/app-academy-logo.img" alt="app-academy-logo" />`

The alt attribute provides alternate text for the image which is used in image search rankings, speech-accessibility, and displayed when a user hovers over the image.

## Thinking in boxes

As we continue structuring web pages we should try to maintain a perspective of seeing all of these HTML element building blocks as boxes.

By thinking in boxes and boxes surrounding other boxes we will be able to create more advanced layouts and have an easier time styling our layouts in the future with Cascading Style Sheets.

So how do we step into this mindset of seeing the web in boxes?

Practice.

Front-end development is more of an art than a science. There are many ways to structure and style a page to achieve the same design, but by maintaining this box perspective we will minimize our efforts.

## Containers

When we are referring to boxes we are talking about both HTML Elements for the sake of holding content and HTML Elements for the sake of holding other HTML Elements.

The HTML Elements that are purely used to hold other HTML elements are commonly called containers.

Containers are vital to our next step in front-end development which will be to use CSS to layout our elements on the page.

## Semantic HTML

First, let us go over some of the tags used to define container elements:

<header>

Container for elements in the top header section of a page

</header>

<footer>

Container for elements in the bottom footer section of a page

</footer>

<nav>

Usually used to hold navigation elements like lists of links

</nav>

<article>

Used to hold content that makes sense on its own like Posts or Comments

</article>

<aside>

Used to hold sidebar sections that are tangentially related to the content

</aside>

<figure>

Holds images, graphic content, code samples etc.

</figure>

<figcaption>

Holds caption content for a corresponding figure element

</figcaption>

<section>

A section of the page or chapter of an <article> with a heading

</section>

<div>

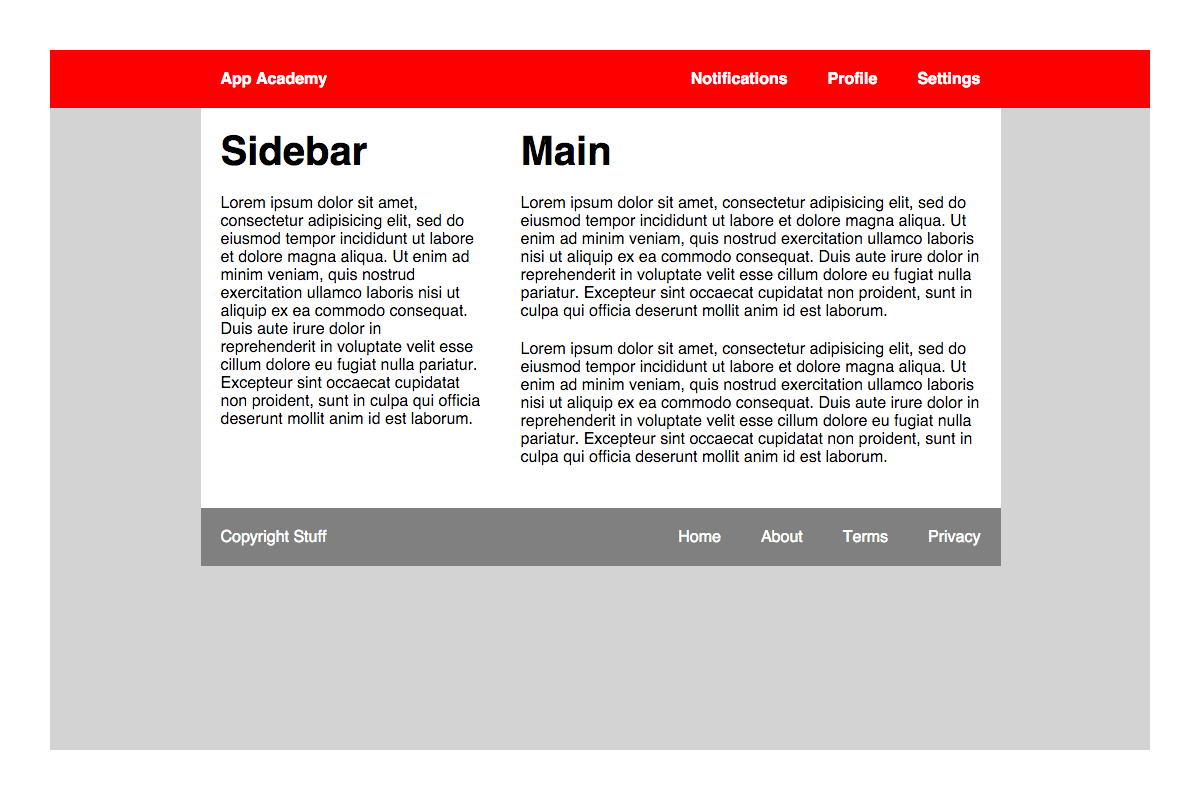
A common container element used when other semantic elements do not seem appropriate

</div>

Other than the div element, all of these tags were introduced in HTML5. This means you will likely find a lot of developers still containing everything in <div> elements.

Here is a great figure from HTML5 Doctor describing the decision of which semantic element we should use: [Semantic HTML Flow Chart](http://html5doctor.com/downloads/h5d-sectioning-flowchart.png)

Watch the following GIF several times to see how this developer thought through containing each of their HTML Elements:



When looking for the right HTML Element we can always refer back to the great [MDN HTML Elements Documentation](https://developer.mozilla.org/en-US/docs/Web/HTML/Element).

**We should try to be as semantic as possible with our HTML structure to not only make our code more readable, but to make styling our code easier as well.**

## HTML Form Element

Forms in HTML allow for users/clients to interact with your application/website by inputting data. We create form elements using <form></form> tags and then fill the form with a selection of different types of control elements.

The following list references the HTML Forms Guide, excellent documentation written by the Mozilla Developer Network. Teaching and remembering every aspect of HTML would be both difficult and fairly unneccessary. Instead we should get used to being able to quickly search correct HTML syntax and practice reading online documentation.

The table contains links to references about the most popular HTML form elements, their most popular attributes and some of those attribute's most popular values. Read through the docs as well as the notes below and then use both to complete tonight's exercise.

| **Popular Elements** | **Popular Attributes** | **Popular Values** |
| --- | --- | --- |
| [Form](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form) | [action](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form#attr-action), [method](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form#attr-method) | Action takes a URI like "/example.com" and Method takes a HTTP request method of either GET or POST. |
| [Input](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input) | [type](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#attr-type) | button, checkbox, color, date, email, hidden, number, password, radio, submit, text, url |
| [Input](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input) | [checked](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#attr-checked), [name](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#attr-name), [placeholder](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#placeholder), [value](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#value) | Checked is either true or false. Name, placeholder, and value will be equal to a string. |
| [Label](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/label) | [for](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/label#for) | The for attribute takes a string corresponding to the id attribute of an element being labeled. |
| [Select](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/select) | name, disabled | Disabled is either true or false. Name will be equal to a string. |
| [Option](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/option) | value, selected, disabled | Disabled/Selected is either true or false. Value is equal to a string. |
| [Textarea](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea) | maxlength, minlength, name, rows, cols, wrap, spellcheck | These values depend on the attribute but are usually just a string of text or numbers. |

## Notes

Some Notes on these popular HTML Elements:

* Input Elements are by far the most common and special element of a form because they can take on completely different functionalities depending on their attribute type.
* Labels can be written two ways, by wrapping the input element or by using the for attribute and applying a corresponding id attribute to the input.
* Submitting forms makes a default post request with parameters made up by the provided values for each name attribute.
* By giving each radio input element the same name attribute value, the radio buttons will only allow the user to select one per radio of that name per form.
* Option elements are defined inside a select element and the value attributes for each option display what will be inside the dropdown.

## Form Example

Below is an example of a form that would be used on a typical signup page for a website:

<form action="http://example.com/signup" method="POST">

<label for="username">Username</label>

<input type="text" id="username" name="username">

<br />

<label>Password

<input type="password" name="password">

</label>

<br />

<input type="submit" value="Sign Up!">

</form>

The action of the form is a URL where it will be submitted. At this point, we won't go over everything involved in a HTTP request, but in short, the information provided by the inputs will be sent in a request to the URL provided.

We can see two styles of using a label, in the first example the for attribute is equal to the id of the corresponding input, and in the second example the label is wrapped around the input. Both are valid ways of labelling an input. The purpose of the label is just to associate text to the input field. Also, if a user clicks on the label, the corresponding input will be selected.

The above example shows three types of inputs: text, password, and submit. Text inputs will show up as a box which a user can type in. A password input is the same as a text input except the text is hidden by black dots. A submit input looks like a button, which the user can not type into, and the value will be the text on the button. There are many other types of inputs, as listed in the table, and they all have their own differences. I recommend making a form and testing them out for yourself!

## HTML Head Section

The <head></head> section of an HTML document contains important information about the page such as title, description, external style sheets, scripts and more.

This information does not render as content and therefore does not belong in the <body></body>.

### Title

The <title></title> element displays a title for the webpage. The title of a webpage is the text that shows up in the tab.

### MetaData

The <meta> tag is extremely important for all modern web pages because it is used to describe our websites to search engines.

This topic of optimizing web pages for the best search results makes up an entire field of study called Search Engine Optimization (SEO).

Because Google represents the #1 source for information, it is no wonder that having a Search Engine Optimized site is one of the best ways to gain traffic.

The first two steps are to optimize your page titles and descriptions. Example:

<head>

<title>Shakshuka Recipe</title>

<meta name="description" content="The best, simple recipe for delicious Middle Eastern Shakshuka, a cumin spiced tomato sauce with sautéd onions and chili peppers topped with poached eggs. ">

</head>

When writing our title tags and meta descriptions it is important to think as if we were a user typing into the Google search field.

Keywords that the user types in will be bolded within our description and the more closely our title/description matches the searched keywords, the better our page will rank on Google.

Here are a couple articles regarding titles and descriptions from the SEO industry experts MOZ: [SEO Title Tags](https://moz.com/learn/seo/title-tag), [Meta Description](https://moz.com/learn/seo/meta-description)

Another metatag is the characterset information mentioned in the beginning of the HTML curriculum. Example:

<meta charset="utf-8">

[W3schools](http://www.w3schools.com/tags/ref_charactersets.asp) has a decent history lesson ending in the modern HTML default: Unicode UTF-8.

Additional metatags include page authorship, content-language, social media specific descriptions, and information for the browser about how the page should be served.

Additional Resources: [MDN Meta Tag Documentation](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meta)

### Scripts

Another popular element found in the head section is a <script> tag. The language for these scripts defaults to Javascript but may also be specified with a type attribute.

We may either write Javascript within the script element or provide a src attribute that refers to an external Javascript file.

Here is an example of a script we should all have on our websites which loads Google's popular free analytics platform: Google Analytics.

<script>

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){

(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1\*new Date();a=s.createElement(o),

m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)

})(window,document,'script','https://www.google-analytics.com/analytics.js','ga');

ga('create', 'UA-12345678-9', 'auto');

ga('send', 'pageview');

</script>

Don't worry about reading this script and trying to understand what it does. This script is provided by Google when creating a Google Analytics account and we are instructed to place it at the bottom of our head section.

Here is an example using an external Javascript file with a declared type attribute:

<script type="text/javascript" src="review-submit-example.js"></script>

### Link Tag

The final piece we will cover with regards to the head section is the link element. This is most commonly used to load an external stylesheet, the specifics of which is covered in the CSS part of the curriculum.

Example:

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

The rel attribute defines the relationship and must refer to stylesheet when loading an external stylesheet file.

The href attribute specifies the URL of the defined resource, in this case the path to the CSS file being requested.

Similar to its use in the <script> tag, type declares the content-language for the file being loaded.

Additional Documentation: [MDN Link Tag Documentation](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link)