Lesson 1: Building blocks of Front End Development

Web page in a browser may be a combination of **structure, style and interactivity**.

These jobs are undertaken by these three different technologies:
HTML, CSS and JavaScript which the browser knows how to interpret.

The three main technologies used to create web pages (HTML, CSS and JavaScript) each do different special jobs.

- HTML should be used only for structuring content.
- Cascading Style Sheets should be used for applying all visual styles.
- JavaScript should be used for (almost) all interactive functionality, and should always be referenced in separate files, never written into HTML.

How HTML, CSS and JS work together

The web page you see in your browser may be a combination of structure, style and interactivity. These jobs are undertaken by 3 different technologies, HTML, Javascript, and CSS which your browser knows how to interpret.

HTML

marks the content up into different structural types, like paragraphs, blocks, lists, images, tables, forms, comments etc.

CSS

tells the browser how each type of element should be displayed, which may vary for different media (like screen, print or handheld device)

JavaScript

tells the browser how to change the web page in response to events that happen (like clicking on something, or changing the value in a form input)

The 2 most important things to learn about web page production are:

Use HTML properly

Different HTML tags describe the structure of content. Each one has its own meaning, and you should only ever use each one in accordance with its meaning, and in its proper place.

Keep them separate

This simply means: Don’t put JavaScript or CSS in your HTML. Put them in separate files, called in by your HTML pages.

There are lots of good reasons for this, the main ones being:

- It’s more logical and simpler. So simple is good.
- Easier management: Keeping all your styles and functional code in one place makes it quicker, easier and safer to change in future.
- Easier re-purposing: It’s easier to design your web site to look or work differently on different user agents (e.g. handheld browsers, audio browsers, or TV)

Lesson 2: Introduction to HTML

HTML is a core building block of the internet. Every single webpage on the internet is made up of HTML in it's most basic form. HTML stands for Hyper Text Markup Language. It is not a programming language, but a markup language meant to be interpreted by web browsers and the data presented in a graphical form. We will be learning about the basic HTML makeup of a webpage.

Basic HTML elements

HTML is a series of "elements" with information in between them. Each element is contained in <> brackets (we will refer to these as "tags") with information either presented inside of them or in between two bookend brackets. HTML tags MUST close by including a "/" at the beginning of the closing element (</elementName>), or before the close of the first element (<elementName />):

```
<span>Information to be displayed</span>
<img src="http://urlofimage" />
```

<html>

The building block of the entire page, all of the other elements will be contained within the html tags. This tells the browser to interpret the page using HTML. Consider this the outermost container, and within it live the head and body elements.

<head>

Data or elements within these tags will not render to the page but tells the browser what data it should load before rendering items to the page. Most notably it will contain the title of the page and any external javascript or CSS dependencies.

<title>

This is the title of the website, it will be displayed at the top of the browser screen or in the tab (not on the actual page itself).

<body>

This is where everything we want displayed to the screen will go, in between the body tags. Anything you see on the page will be in the body.

So far our page looks like this:

```
<html>
  <head>
    <title>10000 Coders - Day 2 Demo : Home work</title>
  </head>
```

```
     <body>
    </body>
    </html>
```

As you can see when we add tags inside of other tags, the best practice is to indent those tags. This will make it easier for you and other developers to read the code and find out exactly where elements are nested.

(From this point, all of these elements will go within the body tags)

<p>

The p ("paragraph") element. Will render text to the screen on a new line.

```
  <p>Here is a new Paragraph</p>
```

<span>

The span element is a generic text container. It does not create a new line like the p element does. This is primarily used for styling text.

<div>

The div element is a generic container. It is used primarily for styling, imagine it like a box that you get to determine the exact size and color and place other boxes in.

<a>

The a ("anchor") element, allows us to create links to other webpages (or even to other areas within our own webpage). You will always see the a element used with the href flag to tell the browser what address you want the link to point to.

```
   <a href="http://www.10000coders.com">Learn to become
 software engineer!</a>
```

<h1> ... <h6>

These are header tags, there are intended to be used as a way to present the subject matter of the page. Like a newspaper headline. the larger numbers will represent smaller text. Imagine 1 is most important and 6 is least.

```
<h1>Most Important Headline!</h1>
<h3>Sort of important headline.</h3>
<h6>Least important headline.</h6>
```

<img>

This element will display an image on the screen. It will always have an 'src' flag which points to the address of the image (kind of like the "href" flag in the "a" element). NOTE: img tags can be self closing, as in they do not need two tags. Simply put the / before the closing bracket in the first tag:

```
  <img src="http://www.10000coders.com/assets/10000coders-
logo.png" alt="10000 Coders Logo" />
```

<ul> and <ol>

This element represents an "unordered list" which is the parent element and will contain list items. There is also an "ordered list", but is hardly ever used in modern web development.

<li>

These are our list items. Within these tags we can place any other element, these just represent our list items.

```
<ul>
  <li>
    <span>List Item One</span>
  </li>
  <li>
    <span>List Item Two</span>
    </li>
    <li>
      <span>List Item Three</span>
    </li>
  </ul>
```

Lesson 3: Introduction to CSS

HTML is great, but HTML alone is plain and boring. This was the early days of the internet, very basic text only webpages. Then came along CSS (Cascading Style Sheets); a way to put some color and style into our webpages! It's like the difference between black & white, and color TV.

CSS selectors and the <style> element

Before we can begin applying style to our webpage, we need a way to tell the browser that we will be giving it some style rules. We can do this a few different ways. The two most popular are using the HTML <style> and <link> elements. Right now we will focus on the easier of the two and use the <style> element.

<style>

Remember when we said that data we wanted the browser to consume before the page loaded should go between the head tags? That is where the <style> tags will go. In between these tags, we will give the browser a set of style rules to abide by.

```
<html>
  <head>
    <style>
      <!-- style rules go here -->
    </style>
  </head>
  <body>
  </body>
</html>
```

selectors

In order for us to apply styling rules to html elements, we have to know which elements to apply the rules to, in come selectors. You can select all elements of a certain type: p, div, body, etc. or you can apply a class or id to each individual element.

We apply these selectors to the HTML tags themselves in the form of a flag:

```
<div id="divId"></div>
<div class="divClass"></div>
```

Ids: are titles that can only appear on a single element, think of it as you would your drivers license number ONLY you have that one number.
Classes: on the other hand can apply to multiple elements. Think of it like a class room, usually you aren't the only person in a class, although you might be, the class is big enough for lots of people.

We do not need to add anything to use every element of a certain type as a selector, CSS does that for us already.

Anatomy of styling rules

Now that we have our selectors in place we need to tell the browser what to do with those selectors. Inside of our style tags, we will insert the rules. Classes will always begin with ".", and Ids will always begin with "#", elements will begin with neither and just have the element name. After the name of the selector we will use braces ("{}") to hold our rules to that one selector.

```
<style>
  body {}

  .divClass {}

  #divId {}
</style>
```

Basic CSS styling

Now that we have some HTML elements selected we can begin to add styling. There are a LOT of different ways you can style an object, you can control how big or small it is, what color it is, where it is placed on the screen, or even if it is visible or not. We will go over some of the most common styling properties and how to use them.

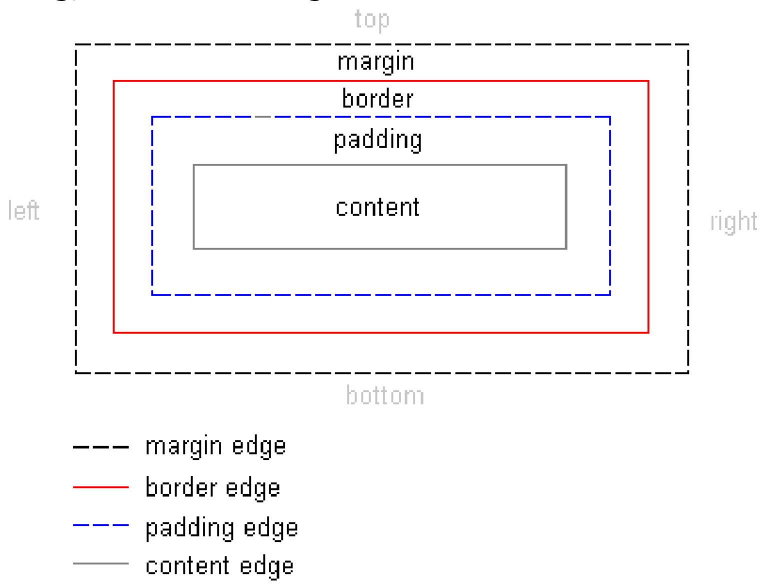
Styling rules

Styling rules will need to adhere to a certain syntax in our CSS so that the browser knows how to read them properly. Within the braces, we will then have the name of the property, a colon(":") and the value of the rule, this will be followed by a semicolon(";")

```
div {
  styling_property: value of rule;
}
```

Introduction to the Box Model

We can consider all html elements to be boxes, the make up of each box is the content, padding, border and margin. This is known as the Box Model.



height and width

We can tell the browser exactly how wide and how tall we want our element(content) to be, this is used in divs, imgs, and other height based elements(in order to determine the size of text, we will need to use a different styling property). Size values can be in lots of different measures, but the most common is the pixel "px".

```
div {
  height: 400px;
  width: 400px;
}
```

margin

The margin is the transparent area around the element that you want to leave open. It is the outermost layer in the Box Model.

border

Border will set a border around your element, you can determine the size color and style of the border. It will be set up in this order: width style color (a list of border styles can be found here: https://developer.mozilla.org/en-US/docs/Web/CSS/border). The border is outside the padding, but inside the margin.

```
div {
  border: 1px solid black;
}
```

padding

The padding is the transparent area between the border and the content. It is very similar to the margin.

Box Model Calculation

When we set the height and width of an element, we are only setting the content. In order to calculate the true height and width we have to factor in the padding, border and margin.

- Padding is a transparent area around the content.
- Border will wrap around the padding
- Margin is the outermost transparent area wrapping around the entire box.

Eg. If we set the height of the content to 20px and the width to 20px, the padding to 5px, border to 1px, and the margin to 10 px.

Actual height = 25px(content) + 2*5px(padding, each side) + 2 * 1(border each side) + 2 * 10(margin, each side) = 57px

Actual width = 25px(content) + 2*5px(padding, each side) + 2 * 1(border each side) + 2 * 10(margin, each side) = 57px

Knowing this will help us size and position our elements correctly.

A Couple of Other CSS Properties

background

Background can be set to a variety of rules, most common would be setting the background to a color or an image. Both are displayed below.

```
.divClass {
  background: red;
}
#divId {
  background: url('http://imageurl.com/image.jpg');
}
```

color

Color is used for text only. It will set the color of your text
font-size
We cant use width or height for text, but we can determine the size of the font used. You can use any size unit here that you would use with a font in a word processor (px, em, in, etc) most popular is px

External Stylesheets and the <link> element

We have gone over how to use the <style> html element. This is fine if you have a very small webpage and minimal styling, but most pages would start to feel cluttered very quickly if we included all of our CSS in the HTML. Thankfully we have a solution for that, external stylesheets and the <link> element.

An External style sheet is simply another file with the .css filetype on it. Conventionally this file is named something along the lines of "style.css". We can take all of the styling rules we wrote between the <style> tags and transfer them directly to the css file. We do not need to include anything else, just the styling rules.
Once we have an external stylesheet created we will need to make sure the browser reads that file and applies the rules to our page. We tell the browser to look for that file by using the <link> element. We can remove the <style> tags and in their place add the <link> element. Within the link element, we will need to supply the location and type of file we are linking. We will use two flags, the "rel" flag and the "href" flag.

The rel flag will just tell the browser what type of file it is and how to process it. In our case we will set that to "stylesheet"

The href flag will tell the browser where to find the file. If the file is in the same folder as our html file we can set it to: "./styles.css" (this path will be relative)

```html
<link rel="stylesheet" href="./styles.css" />
```

Now that we have our external stylesheet linked up to our HTML file, we should see the styling rules we set reflected on our page.