**Git Commands**

* **git init**- initialize the repository.
* **git add .** - add all the files that were changed since the last back up to the staging area.
* **git status** - shows you all the files that were changed since the last backup and which ones are already added to the staging area.
* **git commit -m "..."**- commits the changes to the repository.
* **git checkout \_\_\_\_**- switches to the branch name provided in your git repository. This will create a new branch if the name provided doesn't exist.
* **git branch** - shows all of your git branches and marks the one you are currently on.
* **git log** - shows all the backups created in the repository.
* **git blame \_\_\_\_**- shows who wrote which line of code or in other words who is to be blamed for that particular line of code.
* **git remote add origin \_\_\_\_** - tells git to add a remote place called *'origin*' to a remote URL \_\_\_.
* **git push** - pushes the changes in your local repository to the remote repository.
* **git pull** - pulls the changes in a remote repository to your own local repository.
* **git clone \_\_\_** - clones a remote repository in \_\_\_ to your own local folder.

**HTML**

**Overview**

HyperText Markup Language (HTML) is the main markup language for all web pages. HTML elements are the basic building-blocks of the internet.

Objectives

* Understand the theoretical components of the web and how they interact
* Understand the client-server model
* Understand the most important parts of what is displayed on the browser
* Understand the roles of HTML vs. CSS vs. Javascript
* Understand how to incrementally break down building a web page into its fundamental components

**What is HTML**

HTML is the language of the Web. It stands for Hypertext Markup Language.

Its purpose is to let us communicate with the browser the meaning of the content that we wish to place on a web page. To achieve this, HTML defines a number of **tags**that we can wrap the contents with.

* Headings and paragraphs of text
* Images
* Links
* Lists
* Tables
* Forms

HTML has pre-defined tags for each of these, which we will cover in the following tabs.

HTML tags **describe the content they contain.**For example, there are paragraph tags for paragraphs, anchor tags for links, etc. **The browser does not display the HTML tags** but uses the tags to interpret the content of the page.

**Why HTML and CSS**

The HTML and CSS chapters are designed to take you through the theoretical components of the web as well as the first step in creating a web application (building out the client-side). When building a web application we believe it is best to always start with a clickable prototype involving only HTML, CSS, and Javascript so it is only natural to teach these technologies first.

**Create a Simple HTML Document**

* Create a new file in the Text Editor of your choice
* Type "Hello World!" into the file
* Save the file as **hello\_world.html**
* Drag the file into your browser to open it

That's it! You should see the text "Hello World!" displayed in your browser.

Now, you may have noticed that we didn't add any tags to our text. To see how the browser handles this situation, right click in your browser and click **Inspect Element**. A new module should open at the bottom of your browser with various tabs. We're currently interested in the **Elements** tab. When you click on it, you should see:



Copy

In the case where your HTML document isn't "valid", the browser will try to add appropriate tags to fix it.

One of each of **<html>**, **<head>**, and **<body>** above are required for each valid HTML document.

Some terminology here:

* **Opening and closing tags**

Since tags are supposed to wrap things, **most tags come in pairs**: one opening and one closing tag, that denote the beginning and end of content. Tags without a leading forward slash are called opening tags while tags with leading forward slashes are called closing tags.

In the above example, <body> is an opening tag - it's equivalent to you telling the browser "Hey, I'm going to start putting in body content now". </body> is a closing tag that means "OK, I'm done with body content."

* **Nesting**

Notice that between the opening and closing HTML tags, we also have head and body tags.

**Tags can encapsulate other tags**. This is called nesting. Nested items are **indented with a tab** to make the document easier to read.

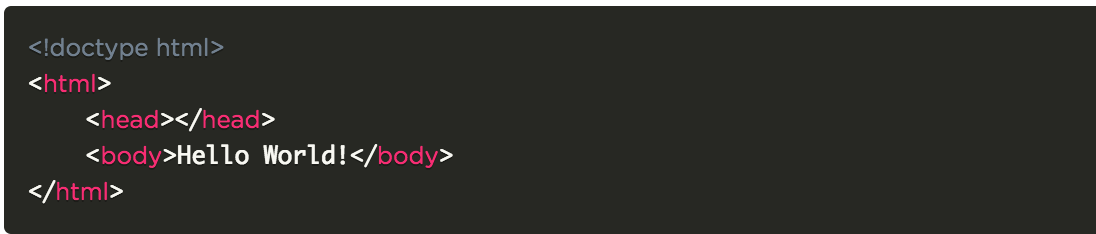
The hierarchy that rises from **nesting** is called **DOM** - Document Object Model.

* **Commenting**

The format for commenting in an HTML file is as follows:



Let's look at our example from the previous page and go through it line by line:

Copy

* You may notice that this time we added the line **<!doctype html>** .The purpose of this line is to tell the browser that we want our document to be read in *standard mode*. What this basically means is that we are using the latest accepted HTML conventions. This line **should be the first line in every HTML document**you create.
* **<html>** indicates that everything between it and its closing tag </html> will be **marked-up using HTML conventions**.
* The **<head>** will eventually contain **meta-data** about the document, the **external Stylesheets and JavaScript libraries** that the document will use, and the document's **title**.
* The **<body>** will contain the **user content** - everything in the document that will be **visible in the browser** window.
* For additional sectioning needs, we will use the **<div>** tag which stands for division.

# Head

Here's an example head section:

<Copy

Let's go through this line by line again:

### **<head>**

This is the **opening head tag**which indicates that we are about to begin talking about the properties of the document.

### Meta Tags

Meta tags make your web page more meaningful for search engines like Google.

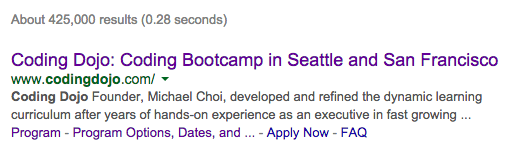
The attribute content of the description meta tag describes the basic purpose of your web page (a summary of what the web page contains). For each web page, you should place a concise and relevant summary in this section.

For example, this description:

<title>Coding Dojo: Coding Bootcamp in Seattle and San Francisco</title>

<meta name="description" content="Coding Dojo Founder, Michael Choi, developed and refined the dynamic learning curriculum after years of hands-on experience as an executive in fast growing..."><br>

This is what shows up in Google's search engine results page:



The **"Coding Dojo: Coding Bootcamp in Seattle and San Francisco"**comes from the <title> tag.

* **<meta charset="utf-8">**

Properly encoded Web pages declare the encoding to a browser through a meta tag in the header. **Without this tag,** a browser may not know to switch to the proper encoding and characters may be displayed as gibberish.

* **<title>My Awesome Webpage</title>**

This is the title of your web page, which means that when you open this page in the browser, the tab that opens it will read "My Awesome Webpage". This will be the name that it gets bookmarked with, as well as the name that is used when it is displayed as a search result in a search engine.

* **<meta name="description" content="description content">**

The description meta tag is used by search engines when displaying results.

* **<link rel="stylesheet" href="my\_css\_file.css">**

This line links a **stylesheet** to our page, which will determine how our HTML elements are **visually displayed** on the page. We will learn more about what goes into my\_css\_file.css in the CSS section.

* **<script src="my\_javascript\_library.js"></script>**

This line links a **JavaScript** or **jQuery** file to our document. JavaScript makes our pages **interactive**. We will learn more about these files in the jQuery section.

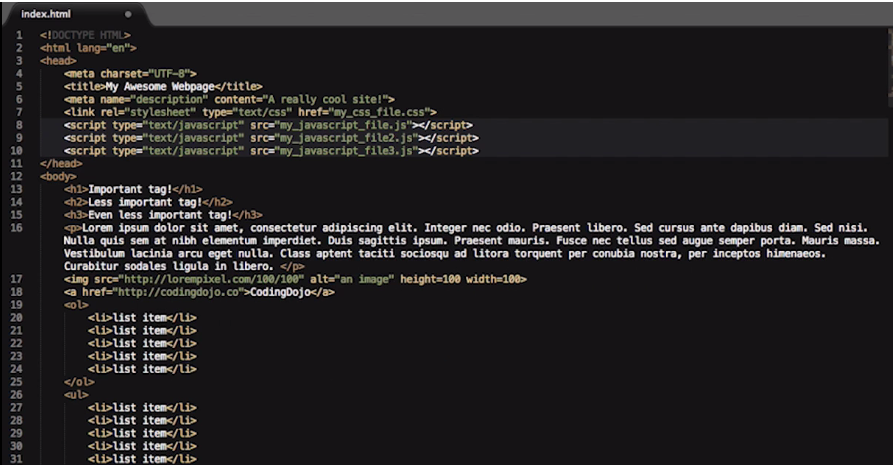
**NOTE: You can link to as many stylesheets or JavaScript files as you want within the head tags.**

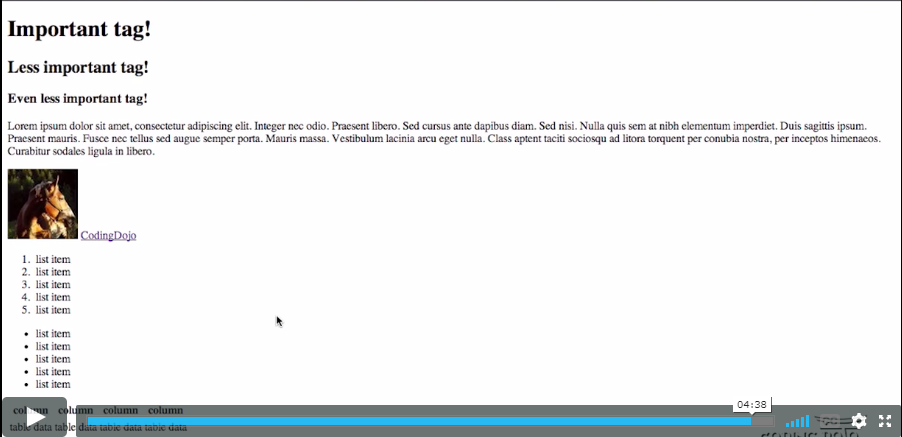
### **</head>**

This is the **closing head tag**. It indicates that we are done talking about the properties of our page and we can move on to the body!

## ****Headings****

**A heading is a section title**, which means that often (but not always!) each section (**<div>**) will have a heading. There are 6 levels of headings that you can use (named **<h1> through <h6>**) each indicating the importance of its section.

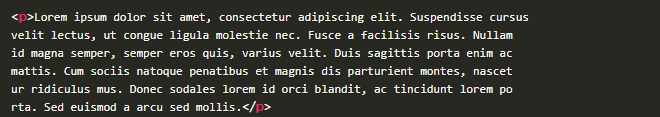




## ****Paragraphs****

Any chunk of text is a paragraph and, therefore, needs to be encapsulated in paragraph **<p> tags**.

Here's an example of how to use <p> tags:



**Images**

There are two ways that we use images on a web page: as **page elements** (such as album art in Pandora, or the photos in your Facebook feed), or as **background images** (this is covered in CSS section).

It has two required attributes: **src** and**alt.**  The **src** attribute stands for **source**. This is the link to where the image is residing. The **alt**attribute stands for **alternate**. This is a few words of text to describe the image, in case it fails to load. This is also used by screen readers, for the vision-impaired. If (and only if) we also specify its **height and width attributes**, this text will show up where the image should be in case the image fails to load.

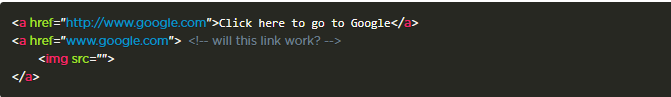
## ****Links****

Links are things that we click on that redirect us to another page. Usually, links are in text format, but you can also use an image as a link.

The tag used for links is the **<a>** tag, which stands for the **anchor tag**. Similar to images, links also need to have an attribute that tells the browser where the link is pointing. For links, this is called the **href** attribute.

**Possible values for the href attribute are:**

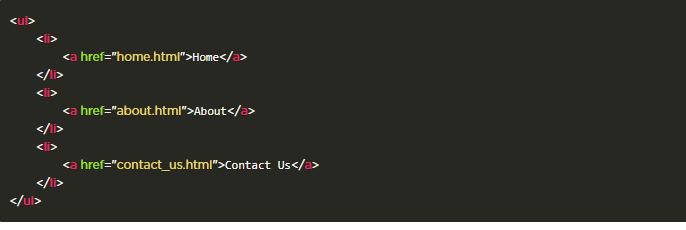
* An absolute URL - points to another website (like href="http://www.example.com/default.html")
* A relative URL - points to a file within a website (like href="default.html")
* An anchor URL - points to an anchor inside a page (like href="#top")



## ****Lists****

It is any collection of elements that are of the same type. The most common use for lists in HTML is for navigation links.

There are two types of HTML lists: **ordered lists** (lists that are numbered) and **unordered lists**. Ordered lists use the **<ol> tag**, and unordered lists use the **<ul> tag**. Both lists use the **<li> tag** to describe each **list item**.



## ****Tables****

We will often find ourselves using tables to display data. Don't be intimidated by the word "data", it just means information.

Tables have many tags associated with them because they are made up of many different parts. They have:

* A table **head** (<thead>), which contains **rows**(<tr>) and column **names**(<th>).
* A table **body** (<tbody>), which contains **rows**(<tr>) filled with table **data**(<td>).

So the tags we need are:

**<table>, <thead>, <th>, <tbody>, <tr> and <td>**



**Forms**

Forms are **one of the most important** HTML tags you will learn. They are responsible for **all the data exchange** between the user (front-end) and the server (back-end). So it's very important that you have a good understanding of how to build them.

A form's job is to**take in user input**and **send it to the back-end to be processed**. A form is declared by using the **<form> tag**, which will have **attributes action and method** that decide **where** the form information gets sent, and **how** it gets sent respectively. (Don't worry about the values we're using for these attributes yet, this is covered during your first stack.)

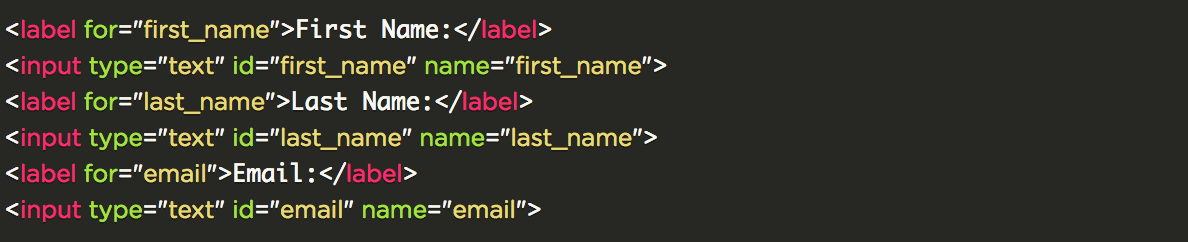
Taking in the input is done by using**input fields**, usually designated by the **<input>** tag. Depending on the type of information required, the way in which we take in the information may be different. This is sometimes designated by a**type attribute**, and other times by a different tag. Each input will typically also have a label (**<label>**) that is the name of the field. To make sure that a specific label is linked/associated to a specific input element, we must add corresponding***for attribute*** on the label with input's ***id attribute***. Including a label tag around the input field is a convention that allows us to click on the label to focus on the input field.

A **name attribute** also will typically go hand in hand with your input tags. They are used mainly for sending form data to the back end so don't worry about the name attribute for now. It will be explained further in your first stack.

Let's look at what input types would be used in the following circumstances:

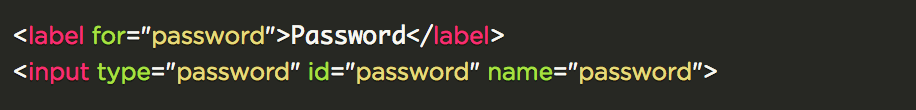
* **When the user needs to enter a short amount of text, such as an email address or name.**

The appropriate input type here is **text**.

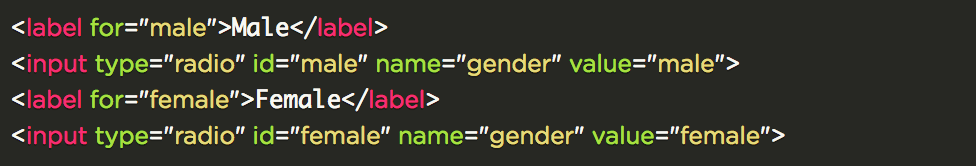
">

* **A password field.**

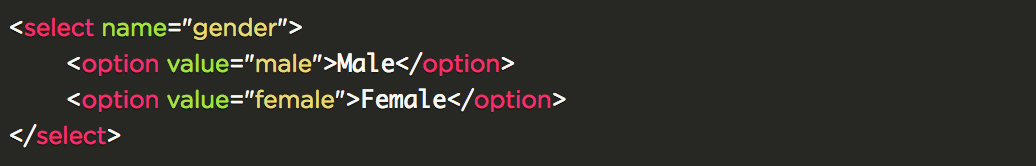
The appropriate input type here is **password**.

**When the user can choose only 1 option from a variety of options. A good example is a gender selector.**

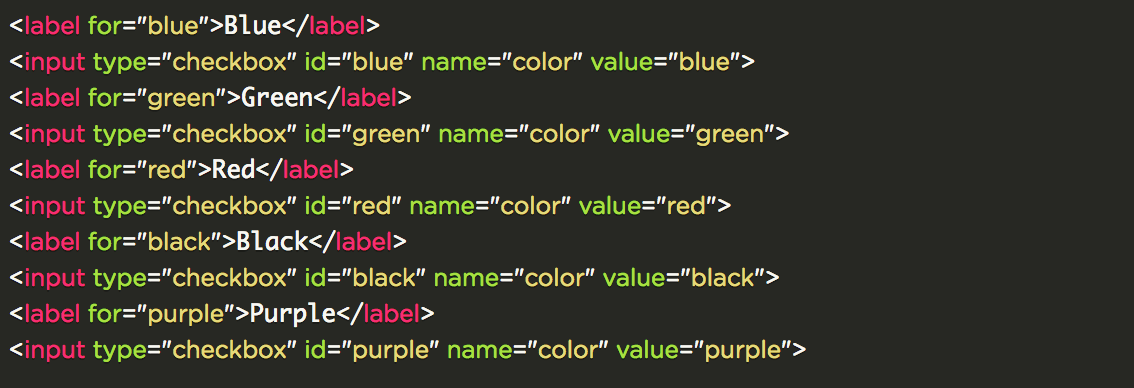
One appropriate input type here is **radio** buttons.

<">

Another option is a dropdown menu, which uses **<select>** and **<option>** tags.

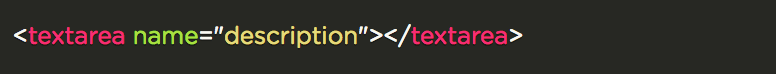
**When the user can choose multiple things from a variety of options, such as choosing their favorite 3 colors from 5 options.**

The appropriate input type here is **checkboxes**.

<">

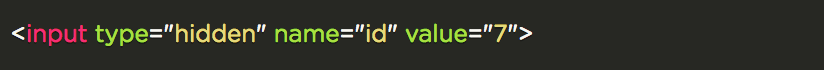
* **When the user might want to enter longer text. This can be used in forums for comments, or for user profile descriptions.**

In this case we use the **<textarea>** tag.

<>

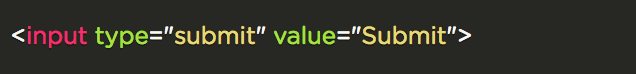
* **When a form needs to submit more than just user input.**

Input type **hidden** is similar to text fields, except they do not show on the page and users can not enter anything into them. This is useful for the back-end authentication and passing data.

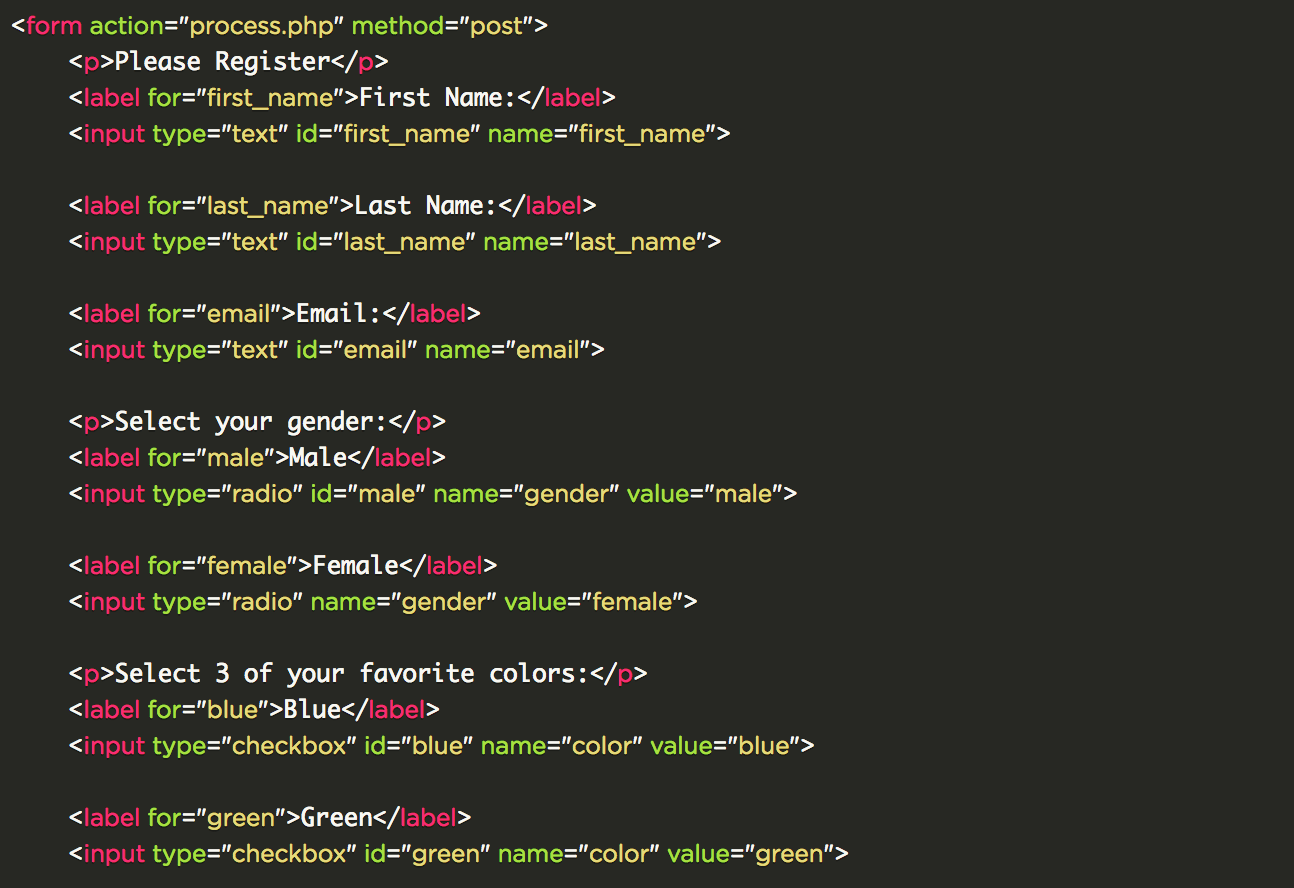
<">

* **Finally, to create a submit button:**

The appropriate input type is **submit**.

">

Let's look at a sample full registration form:

<

**Other *label-input* Declaration**

Most CSS frameworks (especially on Twitter Bootstrap), use the label-input pairing shown above, but you may encounter a different format on how the label-input set is being declared. Below is an example:

<Copy

Notice that the ***input*** element is now nested inside the ***label***element and we no longer need to link the two using the label's for attribute and the input's id attribute.