RC Coding Guidebook

Version 1.1

Introduction

The purpose of this guidebook is to help bridge that gap in Vanderbilt University's REDCap application with the fundamentals of HTML, CSS, and JavaScript. This guide will attempt to give a background to some of the HTML/CSS fundamentals to make sense of some of the things I've found with REDCap with examples that you can tinker with and images to try to demonstrate certain concepts.

Current Versions
REDCap : 14.8.2
PHP 8.1.5 (Windows OS)
MySQL : 8.0.34
Bootstrap: 5.3.3

REDCap Modules
Shazam – v1.3.14
REDCap CSS Injector – v2.0.1
REDCap JavaScript Injector – v2.2.1

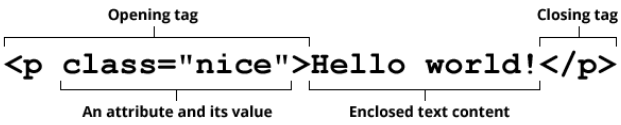
Hyper Text Markup Language (HTML)

For all purposes of this guide I'll keep it simple as to what the different tools are and what they do.

HTML stands for Hyper Text Markup Language, and it's the fundamental building block for websites. All websites at their core are just long text documents like you'd type into a document program. The only difference is that instead of sentences like we're used to, HTML is made up of **elements**. All elements compose of generally an opening or closing **tags** with possibly attributes to give more information or functionality to the element.

There are many types of elements that you will see on a page, some semantic to what the element's purpose on the page is, like <p> for paragraph or <h1> for a level 1 header on the page. But there's a few worthy to note because they are common and serve different functionality depending on what you're going for. And I will teach them to you with another important concept which is inline vs block level elements.

Anatomy of an HTML element



Lesson Example: [HTML Attributes]

Inline vs. Block Level

The two main elements that people learn Inline and Block with are **Span** elements and **Div** elements. Span being the inline element and Div being the block level element.

Like I said in the previous section, webpages are essentially just like a word document, so think of in a text document indentions. That's the best way to differentiate the two. When you place an inline element, it doesn't indent on the page, it stays on the same line—but when you place a block level element it indents to a new line or "block".

I believe the easiest way to get a grip on how elements work in general is to think of all elements as buckets. Buckets hold all the things you want on a webpage from text, to pictures, to anything. The two main kinds of buckets are <spans> and <divs> and you can fit buckets into each other.

So from an organizational standpoint, a <p> element and a element functionally work the exact same, it's just easier to understand when you're reading your website that <p> is a paragraph on quick glance. And that's the same with inline as well, a button is just a element with some build in functionality to make it look like a button you're more used to seeing.

Inline Elements:

-
- <a>
- <b> /
- <button>
- <code>
- <img>
- <input>
- <label>
- <textarea>

Block Level Elements:

-
- <section>
- < header / footer>
- <h1-h6>
-
- <nav>
- <ul / ol >
- <p>

Lesson Example: Inline Vs. Block

Classes & IDs

Now when it comes to targeting HTML elements, an important concept is understanding ID's and Classes. The importance of both is that they give you a way to target individual items on a webpage or a grouping of items to behave or look a certain way.

First we'll begin with **ID**s. Just like an individual person would have their own ID to identify them and it can't be used by another individual. IDs target that single element on a webpage. IDs are added to an element by adding an attribute to an element called "id"; for example:

```
<article id="dailyHighlight"> </article>
```

Assigning an id attribute to an article element tag

What this will allow you to do is target this particular article on a webpage and be able to manipulate it; which we'll learn more about in the next section of this guide. But know for now, just know that this is how you assign an ID to an element.

```
<section class="frontPage mediumArticle"> </section>
<section class="frontPage largeArticle"> </section>
```

Assigning class attributes to section element tags

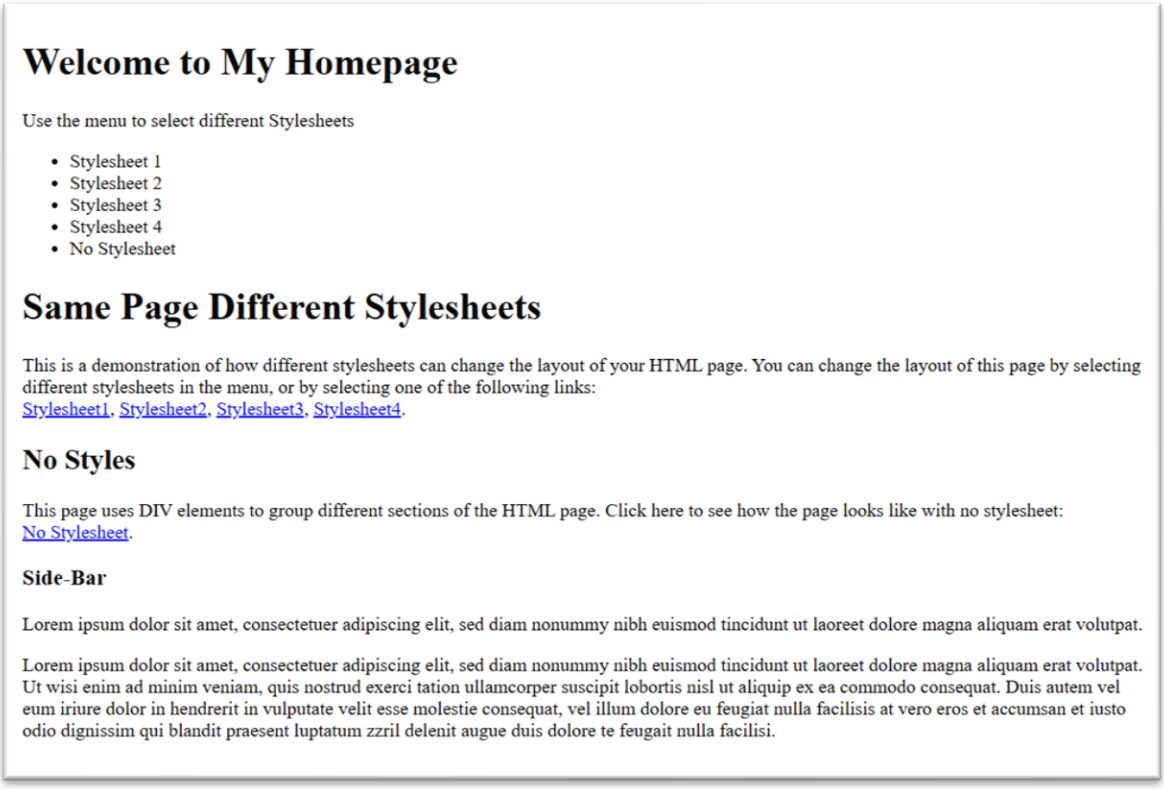
Similarly for classes you add an attribute named "class" to assign a class or classes to an element. That is a major distinction between classes is that classes can be assigned to multiple elements, but also elements can have multiple classes. In the example above, the first section both has a class of "frontPage" and "mediumArticle". The separation comes from the space. If there was an underscore or dash(hyphen); that would be one class name. For example, "front-page" can be a class. Classes are case sensitive and separated by spaces.

Cascading Style Sheet (CSS)

Now it's time for the "make it pretty" part of this guide. CSS, short for Cascading Style Sheet is the backbone of what tells a webpage how to look. What colors items should be, how spaced apart they should be and so forth. A good example of what the difference CSS can make is over on W3 Schools CSS guide, they're a Demo page where you can flip through the same HTML but with different CSS attached to it.

W3 Schools CSS Demo

CSS Lesson Example



Welcome to My Homepage

Use the menu to select different Stylesheets

- Stylesheet 1
- Stylesheet 2
- Stylesheet 3
- Stylesheet 4
- No Stylesheet

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links:
Stylesheet1, Stylesheet2, Stylesheet3, Stylesheet4.

No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet:
No Stylesheet.

Side-Bar

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

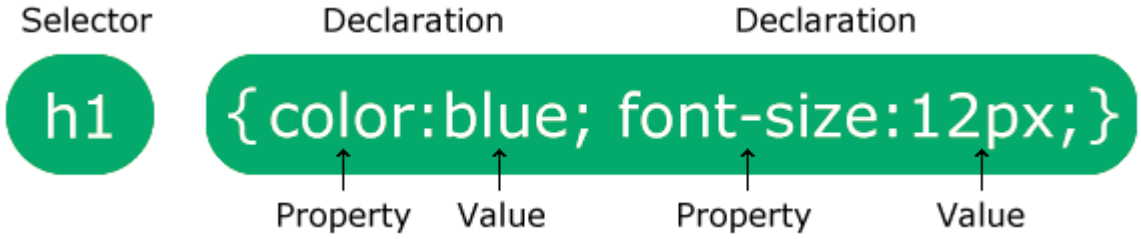
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

Webpage example from W3 Schools

CSS Syntax

Now the short hand here is that there is a way that CSS is written, or **syntax**. Each section comprises of a **selector** followed by curly brackets containing what you want that selector to do. These are called **declarations** and consist of **property** and **value** pairs separated by semicolons.

An important thing to get a grip of with writing CSS is how to select the things that you want to target. The main way to do this is by using IDs and Classes which we learned earlier but there are other unique combinations to be aware of.



The first part above is the selector. This can be an HTML *element*, an *ID*, a *Class* or it can be a calculation of other *attributes* or *selectors*.

CSS Selector Reference Cheat Sheet

*	Selects all elements
div	Element example, selects all div elements on a page
div, p	Selects all divs and paragraphs elements on a page
div p	Selects all paragraph elements inside of div elements
div > p	Selects all paragraph elements that are a direct child of a div element
.className	Selects all elements with a class of “className”
#idName	Selects element with the ID of “idName”
div.className	Selects all div elements with a class of “className”

Bootstrap

Now it's time for Bootstrap! Bootstrap is what's known as a **Framework** and allows people to build good looking website pages very quickly using predefined cascading style sheets. Bootstrap was originally created by employees Mark Otto and Jacob Thornton at Twitter to expediate creating a standard for how Twitter looked and now something like a quarter of the internet uses Bootstrap as far as we know.

Typically, to use Bootstrap it must be installed onto a website, and there's several ways to do that, but in our case, REDCap already uses Bootstrap so some of the components we can naturally use within development. In order to access the functionality, we just have to add corresponding bootstrap classes to elements and they take on their bootstrap properties.

For example, a standard button, when passed the classes from Bootstrap "btn" and "btn-success" for example, a Bootstrap button will show with styling that's added from bootstrap.

```
<button type="button" class="btn btn-success">Success</button>
```

There's a number of Bootstrap components available, REDCap has some caveats like anything like a button like this will pull you away from a survey, but somewhere like a dashboard page, you can have buttons that link you to different sites or surveys etc.

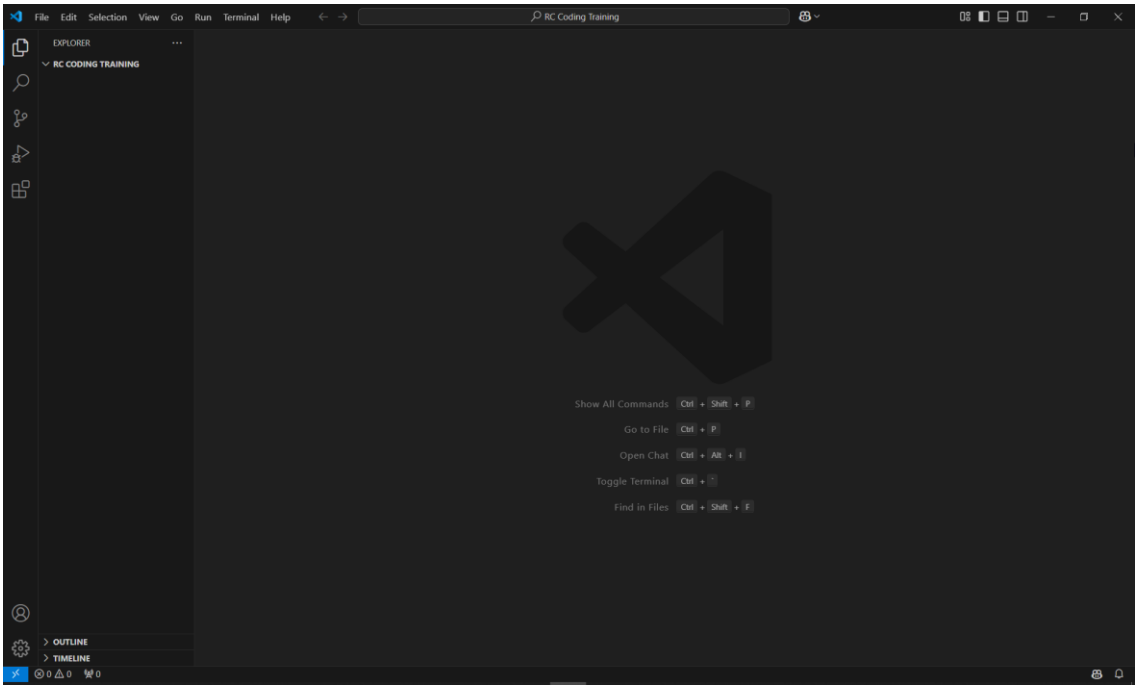
Another thing, when it comes to surveys, you can't use Bootstrap items like popover menus, any kind of tabs or most things you can click because clicks register as you're leaving the survey but experiment. I will link you to the Bootstrap Cheatsheet where you can see some of the components that Bootstrap has to offer with a link to the documentation on how to get started with that component.

Bootstrap Cheatsheet

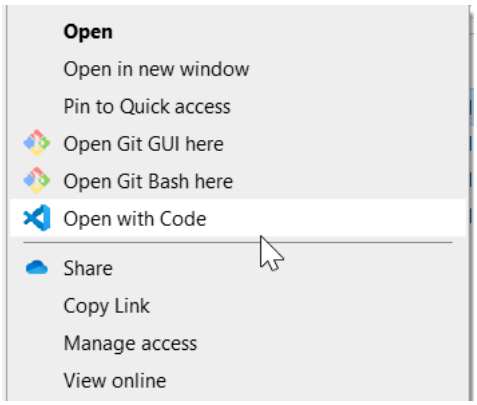
Bootstrap Lesson Example

Visual Studio Code

I thought I'd take time to take a break from some of these coding fundamentals to talk about some tools to help practice some of these fundamentals and some quick tips. One such tool that can be pivotal is a text-editor. A text-editor is just what it sounds like, it's a piece of software that allows you to edit text. My favorite text editor is Visal Studio Code, VS Code for short.



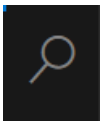
VS Code can be downloaded from the Microsoft website. VS Code can allow you to edit a folder of files or text files in a quick manner. There's a number of ways to begin a project or to begin editing files in VS Code. One easy way to begin a project is when you're downloading VS Code there's an option to add VS Code in your right click menu, select that.



This allows you to open up a folder on your right click menu in explorer(Windows) / finder (Mac).

One of the first things to get used to with the interface is the side panel. From top to bottom these are the sections and what they do in summary.

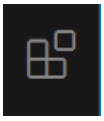
Explorer: This is the main page that you need to be aware of. This is where you can edit and navigate between files in a directory



Search: This is where you can search through a directory for a word, phrase, or file.

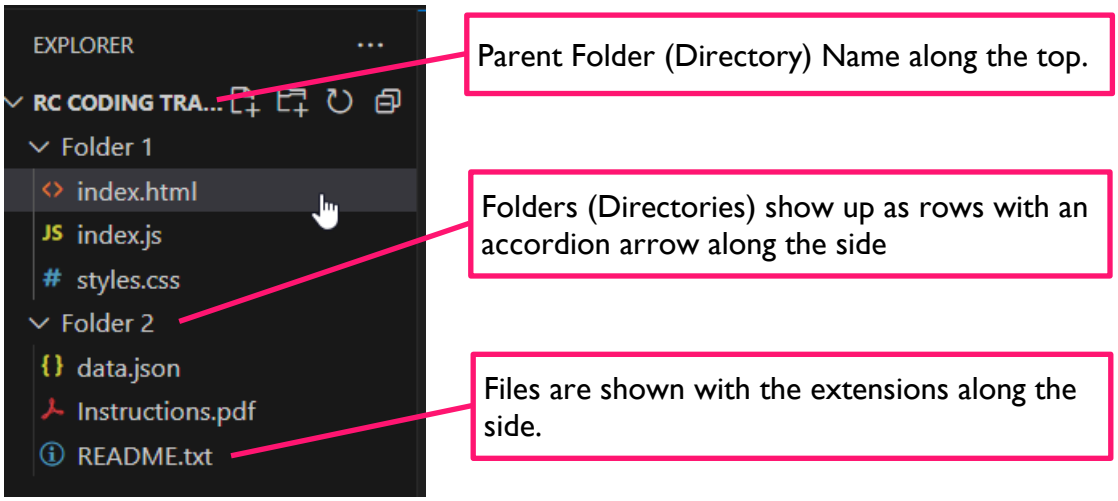
Source Control: This is usually tied to git where you can keep up with updates in a directory, this won't be important in this guide for the time being.

Run & Debug: In some languages you can run the code directly in the browser and debug it (look for errors) but this won't be important in this guide for the time being.



Extensions: Where you can download plugins & extensions to help with your editing workflow.

The main section you'll spend time in is the Explorer tab. Whenever you open a directory (folder) it'll open along the left-hand side in the Explorer panel.



Whenever you want to practice some of these things, VS Code can be a helpful tool. For another getting started guide visit the VS Code website.

[VS Code Getting Started Guide]

JavaScript

Atlas where programming really begins. When it comes to the actual functionality of a website with elements behaving dynamically, or whenever a set of actions occurs when you push a button—JavaScript is the engine behind that. JavaScript is the programming language that's used between browsers alongside HTML and CSS.

JavaScript can be written in a number of places to affect a webpage. Most often JavaScript is stored within its own file and referenced in a document using a **script** or **link** element, but it can also be written inline using a **script** element usually within the **head** element or the **body** element.

Variables

Fundamentally programming languages are composed of **variables**. Think of a variable as a bucket or container that holds something. More accurately think of it as a bucket that holds a value. Just like in algebra with an equation like y + 2 = 4, y has a value of 2, variables work the same way.

Variables are set using one of three JavaScript keywords: **var**, **let**, or **const**.

var | let	Variables that can change
const	Variables that cannot be changed

You set the variable by using an **assignment operator** statement, that just means you declare the variable to have a certain value before it can be used for example using out math equation from before:

```
var y = 2;
```

One thing to note is that JavaScript is read from top to bottom much like written language and that all executable lines end with an implied semicolon. Meaning that where you declare a variable, after it's declared is where it can be used otherwise, you'll get some errors.

Kind of like when you're telling a story, and you mention somebody's name, but they're never met that person before? You must explain who somebody is to make the story make sense.

In the above example we used a number to declare the variable y. But JavaScript has 8 different datatypes that can be used, but we're going to focus on three commonly used datatypes.

Variable Datatypes

Variable Type	Description	Example
Number	Numbers both positive & negative	4 , -50, 40.00056
String	Text, letters, sentence etc.	“Cow jumped over the fox.”
Boolean	True of False Value	true, false

First we’ll start with a **String** datatype. A string is just a string of text tied together with single or double quotation marks, just make sure that which one you start with you end with. For example, “Everything between these quotes is a string.”

var stringExample = “Everything between these quotes is a string.”;

var stringExample2 = ‘ This is also a string example. ‘;

Strings allow us to pass text phrases to be used throughout the webpage as well as interpret things in human readable language. The important thing to remember is that strings are always between quotation marks.

Next we have **Number**, which we’re previously visited. Numbers allow us to make mathematical calculation on webpage like in the examples of calculating finances or date fields.

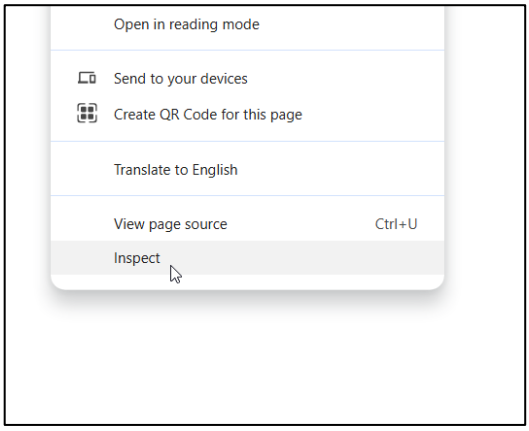
var numberExample = 2024

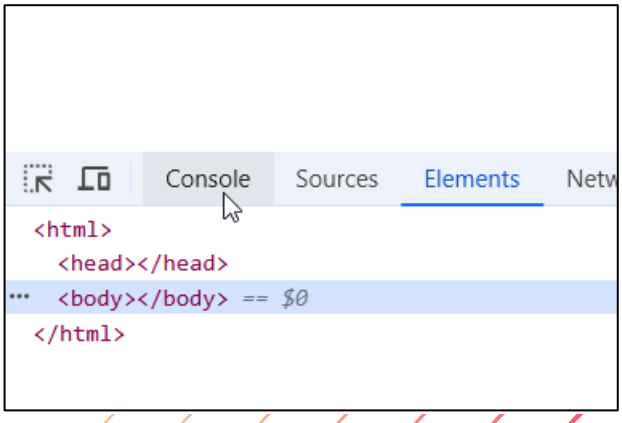
Lastly we have **Boolean** values which allow us to later use logic in our code to differentiate if a value is true or false. This will make more sense later. When we get into examples of when and why we would want to do that.

var = isBoolExample = true;

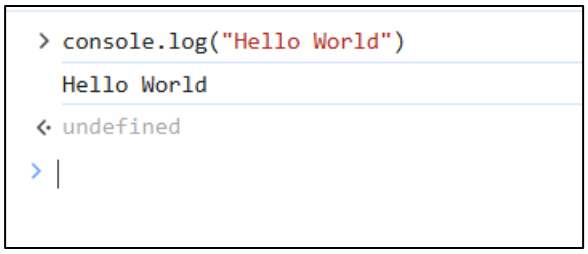
The Console

The **console** is a tool built into browsers as well as most computers that let you interact and run code. For the sake of this guide, I will focus on the console in the browser and how it relates to JavaScript. In order to use the console in a browser you can right-click on a page and hit inspect and select the “Console” tab.



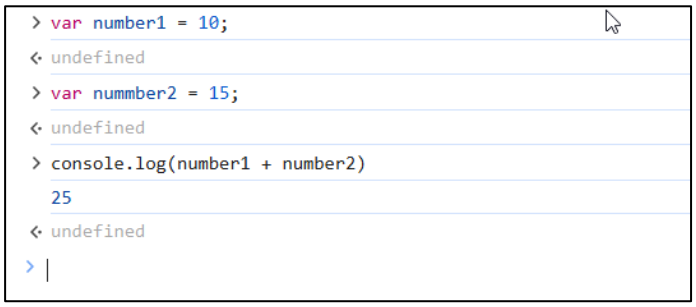


In the console you're able to declare variables in the same way that we've discussed. So if you declare a variable you can see the value of the variable. To see the value of a variable we're going to log it to the console using something called console.log().



Example of logging a string to the console

Get some practice in setting variables in the console and console logging them.



Example of declaring two variables and using them in simple math problem.

The same way you can use JavaScript in a console window is the same way it can be interpreted in a JavaScript file or inline on an HTML file. When creating JavaScript files the file extension is .js files. In the examples of code pen, they're automatically attached for you and will work accordingly.

Javacript Practice 1

Functions

Functions are reusable chunks of code in programming. Some people also call them methods. What they allow you to do is create a set of instructions that can be duplicated multiple times.

```
function functionName(parameter1, parameter2){
    // Code here
    return parameter1 + parameter2;
};
```

Similar to a variable, a function starts with declaring that you're creating a function using the "function" special word followed by what you want to name the function. Then you tell JavaScript the pieces that you need to execute the function, for example in a math problem it would be numbers or is can be strings or any of JavaScripts variable types.

These pieces are called **parameters** and are passed in parenthesis separated by commas. Lastly you tell the function what you want it to do and then you have a value that's returned which I'll return to later.

Arithmetic and Conditionals

As you may have noticed with the above JavaScript examples, JavaScript allows you to do mathematic operations and other neat things.

Operator	Description	Example	Result
+	Addition	5 + 3	8
-	Subtraction	10 – 4	6
*	Multiplication	2 * 3	6
/	Division	8 / 2	4
%	Remainder (Modulo)	7 % 3	1

Calling Functions

Once a function is declared, you have to call it in order for it to work. To call a function you type in a functions name followed by parenthesis with any needed parameters. Here are some examples.

sayHello()

Example of a function without parameters.

addNumbers(10,15)

Example of a function with parameters.

[Javascript Practce 2]

Conditionals

Conditionals are used in JavaScript to perform different actions based on certain conditions. It's like making decisions in your code. If something is true, do this vs. if not, do something else.

One conditional statement that is common is called an **if statement**. If statements are started by the JavaScript special word if followed by a condition in parenthesis. If the condition is met, then the following code is run.

```
if (condition) {
  // Code to run if the condition is true
};
```

You can chain on other conditions or add in code to be executed if the condition is not met as well.

```
let age = 18;

if (age >= 18) {
  console.log("You can vote!");
} else {
  console.log("Sorry, you're not able to vote.")
}
```

[Javascript Practice 3]