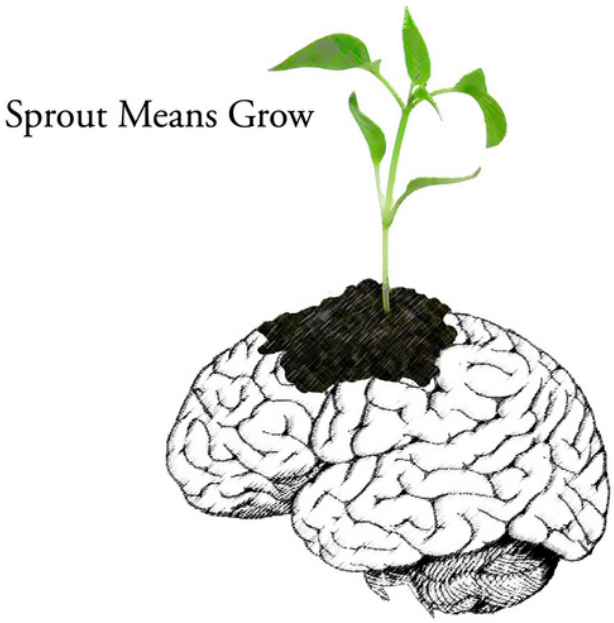
Introduction to JavaScript



Why should I care about JavaScript?

HTML tells the browser what to show--the content. CSS tells the content what to look like. Javascript tells the content and data how to act. Also, it's a great introduction to programming and a relatively easy gateway to more "advanced" programming languages. Getting to know Javascript means you'll be familiar with concepts you'll need to know in the future including:

- Objects (as in "Object-Oriented Programming")
- Properties
- Methods
- Variables
- Functions

Hello JavaScript!

In the <head> section of an HTML document, type the following and test it in a browser:

```
<script>
  alert ("Hello, World!");
</script>
```

Is Something Missing?

As of v5, HTML no longer requires the **type** attribute because JavaScript is the default scripting language in HTML5.

In telling our content "how to act" we tell it what to do and how to do it. We can use Javascript to make the page interact with the user by asking questions and providing answers. We can also have Javascript dynamically respond to actions or conditions without even asking questions.

Instead of a generic "Hello, World!" alert, let's write some Javascript asking the user's name and responding with a personal greeting.

```
<script>
  var name = null;
  name = prompt ("What's your name?", "");
  alert ("Hello, " + name);
</script>
```

The first line creates a **variable** called "name." A variable is an empty container for information we'll add later. It's like **x**, an unknown, in algebra. It might seem useless before we assign it a **value** but it's essential as a place-holder until we know what to put in its place. So, for now, we assign the variable "name" the value of "null" which just means its empty until we tell it otherwise.

The second line opens a little window (just like "alert" did), asking "What's your name?" and providing an empty field for the user to enter their name. The user assigns a value to the "name" variable.

The last line opens another little window with the text "Hello, " (notice the space after the comma) plus the value entered by the user.

Strings vs Numbers

Any series of numbers and letters within quotes like that is called a **string**. If what Javascript will display is, literally, the contents of the quotes—as in "Hello, World!"—that's a string. Variables don't have to be strings, they can also be numbers or conditional responses. Let me repeat that it is the opening and closing quotation marks that make the string. To really hammer this home, let's find out whether or not 2+2=4. Type and test the following:

```
<script>
  alert (2 + 2);
</script>
```

You should see the answer "4." You can also subtract, multiply, and divide. Now try this:

```
<script>
  alert ("2" + "2");
</script>
```

When using quotes, we're telling Javascript we want to literally see a "2" and another "2." Notice there was no space between "2" and "2" because we didn't put one in either set of quotation marks.

Super Fun and Exciting Practice Exercise

Type the following code to calculate numbers—without actually typing any numbers. Javascript asks the user for a number and returns the product of that number multiplied by itself.

```
<script>
  var theirNumber = null;
  theirNumber = prompt ("Enter a number to square", "");
  alert ("The square of " + theirNumber + " is " + theirNumber * theirNumber);
</script>
```

I don't know about you, but I think even these short & simple scripts are a lot of fun.

Consolidating Instructions Using Functions

Imagine you supervise a kitchen in a fast food restaurant and every time a customer orders a burger, you have to tell the cook:

1. Place a meat pattie on the grill.
2. When the pattie is fully cooked, place it on a bun-bottom.
3. Add onions.
4. Add mustard.
5. Add cheese.
6. Place the bun-top on top.

Every time a burger is ordered, you must repeat stating all of these steps. In code, it might look like this:

```
add ingredient ("bun bottom");
add ingredient ("meat pattie");
add ingredient ("mustard");
add ingredient ("onions");
add ingredient ("cheese");
add ingredient ("bun top");
```

Wouldn't it be great if, each time a customer ordered a burger, you could just say, "Make a burger"? A function is a gathering of multiple instructions into a single statement or command. First, we create the function by defining it (name it and list all the instructions it will follow). Then, we call it when we need it by using an **Event Handler**.

What's an Event Handler, you ask? First, let's learn about **Events**. An Event is something the user does like clicking a link or moving their pointer over an image or something a customer does like order a burger or pay for their order.

An Event Handler tells the web page or cook what event they're waiting for. The function, then, is the appropriate response to that event.

- "onMouseover" change what the button looks like
- "onOrder" make a burger

In our fast food example, defining a function would look like this:

```
function makeBurger (){
  add ingredient ("bun bottom");
  add ingredient ("meat pattie");
  add ingredient ("mustard");
  add ingredient ("onions");
  add ingredient ("cheese");
  add ingredient ("bun top");
}
```

To define a function, type **function**, the name of the function, open & close parentheses, and an open curly brace. Between that open curly brace and the closing curly brace, you'll list your instructions or *statements*.

From now on, you only need to call the makeBurger function. In fact, it's not like you even need to call it each time you want a burger--the Event Handler tells the cook "Every time somebody orders a burger, make a burger." That looks like this:

```
onOrder="makeBurger()"
```

Pretty cool, huh?

To be continued ...