**Step 2.**

**Variables**  
**Variables**are containers for storing data values. The value of a variable can change throughout the program.  
  
Declaring a variable is as simple as using the keyword **var**. Which would look like this:

**var** x = 10;

In this example we’ve assigned a value of **10**to the variable **x**.  
  
We’ve used the word assigned deliberately here, because in JavaScript, the equal sign (**=**) is actually called the "**assignment**" operator, rather than an "equal to" operator.  
  
Which means that in JavaScript, **x = y** will assign the value of **y** to **x** variable.

**Heads up!**  
JavaScript is sensitive, case sensitive that is. So variables like lastName and lastname are not the same.

**Using Variables**  
  
Ok, let’s put some of what we’ve learned together!  
  
How about we assign a value to a variable and output it to the browser. We’ve got this!

var x = 100;   
document.write(x);

But what’s the point of variables anyway? Well, imagine your program has 1000 lines of code that include the variable x. With variables you can change the value of the variables and use them multiple times in your code:  
  
**Like this:**

var x = 100;   
document.write(x);   
  
x = 42;   
document.write(x);

**Heads Up!**  
Every written "instruction" is called a statement. JavaScript **statements**are separated by **semicolons**.

**Naming Variables**

Let’s talk about names.  
It’s super important to remember that JavaScript variable names are case-sensitive.  
  
What do you think the output of the following code would be?

var **x** = 100;   
document.write(**X**);

That’s right! An error. That’s because x and X are different, and we didn't declare the X variable.  
  
Brace yourself for more rules!  
- The first character of a variable name **must be** a letter, underscore (\_), or a dollar sign ($) (Subsequent characters can be letters, digits, underscores, or dollar signs).  
- The first character of a variable name **can’t** be a number.  
- Variable names **can’t**include a **mathematical or logical operator** in their name. For instance, 2\*something or this+that;  
- Variable names **can’t contain spaces**.  
- You’re not allowed to use any special symbols, like my#num, num%, etc.

**Heads up!**  
JavaScript is a hyphen free zone. They’re reserved for subtractions.

**JavaScript Comments**

Great stuff! You’ll soon be a variable master.  
  
Ok, let’s talk about **comments** in JavaScript.  
So we know about statements, these are the instructions within our program that get "executed" when the program runs.  
But! Not all JavaScript statements are "executed".  
Any code after a double slash //, or between /\* and \*/, is treated as a **comment**, and will be ignored, and not executed.  
To write a **Single line** comment we use double slashes. Like this:

**// This is a single line comment**  
alert ("This is an alert box!");

What if we have more to say?  
If we want to create a **multi-line comment**, we write it between /\*and \*/

**Like this:**

/\* This code   
creates an   
alert box \*/   
alert("This is an alert box!");

**Data Types**

The term **data type** refers to the types of values a program can work with. The sky's the limit with JavaScript variables, which can hold a bunch of different data types–**numbers, strings, arrays,** you name it.  
  
Let's start simple though.  
Numbers can be written with or without decimals. Like this:

var num = 42; // A number without decimals

var price = 55.55; // A number with decimals   
document.write(price);

**Heads up!**  
Changing this variable is a breeze, just assign to it any other data type value, like num = 'some random string'.

**Strings**

Ok, let’s turn up the heat...  
In JavaScript we can use **strings**to store and manipulate text.  
A string can be any text wrapped in **quotes**. Single or double quotes, it doesn’t matter, so long as you’re consistent with them. Like this:

var name = 'John';   
var text = "My name is John Smith";

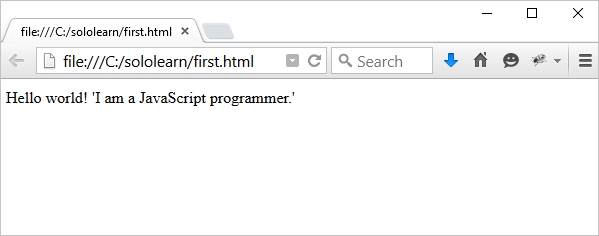
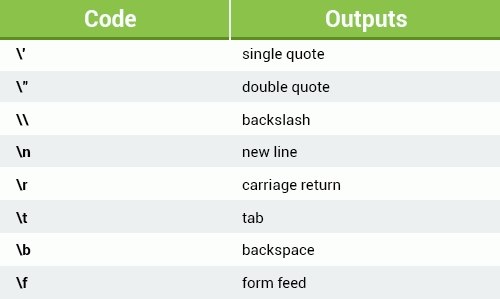
What if we want to use quotes inside a string though?? No problem! You can use quotes inside a string, as long as they don't match the quotes enclosing the string itself. Take a look:

var text = "My name is 'John' ";

**Heads up!**  
You can get double quotes inside of double quotes using the escape character like this: \" or \' inside of single quotes.

**Strings**  
Now is a good time to talk about the **backslash (\) escape character**. It comes to the rescue when you need to put quotes within strings (and a bunch of other situations) by transforming special characters into string characters.  
  
**Take a look:**

var sayHello = 'Hello world! **\'**I am a JavaScript programmer.**\'** ';   
document.write(sayHello);

**Result:**But the escape character (\) isn’t just for quotes, it works when you need to put other special characters inside strings too!

**Heads up!**  
If you start a string with a single quote, then you need to end it with a single quote too. This applies to double quotes. Otherwise, JavaScript will get confused. Poor JavaScript.

**Booleans**

Not just fun to say, Booleans in JavaScript serve a useful function by letting you have one of two values, either **true**or **false**.  
  
So when you need a data type that can only have one of two possible values, like Yes/No, on/off or true/false, look no further than Mr Boolean. Let’s look at an example:

var isActive = true;   
var isHoliday = false;

**Heads up!**  
The Boolean value of 0 (zero), null, undefined, empty string is **false**.  
Everything with a "real" value is **true**.