JavaScript for App Development

By Mark Lassoff, Founder Framework Television

Section Two

I hope you enjoyed working your way through Section One of the course and now have a good grasp of creating JavaScript programs, editing them, debugging and executing them in the browser. Even if you feel shaky at this point, you're going to practice these skills over an over again, so, please, don't worry!

Section 2 Goals

In this section of the course your goals are:

- ☐ Declare a Variable
- ☐ Initialize a Variable by Assigning an Initial Value
- ☐ Use Integer and Floating Point Variables
- ☐ Use String Variables
- ☐ Make your first Calculation using a Variable Operator

Declaring a Variable

Declaring a variable is just a matter of registering it with the JavaScript processor. Every variable must be declared before it can be used. The ES5 version of JavaScript that we have been using so far uses the keyword var to declare a variable.

```
var age;
var name;
var playerScore;
```

Above are examples of three variable declarations. You should note that when you name your variables, you want to use names that are descriptive. The idea is for a variable to document the value the variable holds.

A couple of rules you should note for naming variables:

- Variable names cannot be JavaScript keywords.
- Variable names can contain alphabets and numbers.
- Variable names cannot contain spaces and special characters, except the underscore (_) and the dollar ($) sign.
- Variable names cannot begin with a number.

Initialize a Variable with an Initial Value

Once a variable is declared, it must be initialized before use. When you initialize a variable, you're storing an initial value in that variable. Here's an example of the three variables we declared earlier being initialized:

```
<script>
var age;
var name;
var playerScore;

age = 44;
name = "Mark Lassoff";
playerScore = 0;
</script>
```

You'll notice that variables were initialized with the '=' sign. While by most of us will call that the "equal sign", the correct terminology in coding is

the “assignment operator”. It assigns a value to a variable. The “proper” way to read `age = 44` is “variable age is assigned the value 44”. (In the real world, plenty of programmers say “equals”. Correct them at your own risk!

Combined Declaration and Initialization

Most of the time developers will combine the declaration process and initialization process into a single
step. It’s just more convenient.

```
<script>
var age = 44;
var name = “Mark Lassoff”;
var playerScore = 0;
</script>
```

Why write six lines of code when you can write three? Right?

Functionally, there is no difference between combined declaration and initialization and declaring and initializing separately.

I’ve purposely left out a discussion of variable scope at this point. We’ll cover variable scope when it’s a bit more cogent.

A Note on ES6

We’ll be covering the ES6 standard in greater detail later on in the course. However I wanted to preview the relevant ES6 method of declaring a variable. Most commonly in ES6 you’ll see the keywords `let` and const instead of var.

The advantage of the ES6 `let` is that it allows access to the variable to be restricted to the nearest block.

(This is a safety feature preventing an outside object from manipulating the value of the variable unintentionally.)

Before ES6 JavaScript really had no ability to deal with constants. Constants are *read-only* variables. Essentially, once a constant is assignment a value, that value does not change. I would use a constant, for example, to define the width and height of a playing field for a game. That's not going to change throughout the program execution, so a constant is appropriate.

To declare a constant in ES6 `const` is used.

Much hasn't changed as both ES6 keywords are used similarly to `var`. As you would expect, the script below will output "Audi" and "windows".

```
<div id="output"></div>
<script>
"use strict"

   let car = "Audi";
   const OS = "windows";

   document.getElementById("output").innerHTML = car +
"<br/>";
   document.getElementById("output").innerHTML += OS;
</script>
```

Numerical Variables: Integers and Floating Point Numbers

JavaScript variables can hold two types of numerical values. First, variables can hold integers, which are whole numbers. Variables can also hold floating point numbers which are numbers that contain a decimal point. Both types of numbers are assigned to variables the same way – with the assignment operator.

```
<script>
var age = 44; //Positive Integer
var status = -100; //Negative Integer
var gpa = 3.55; //Float
</script>
```

Numbers stored in JavaScript variables can be positive or negative.

Regardless of the type of value stored, the value stored in a variable is retrieved by naming the variable in your code. Continuing the above example:

```
<div id="output"></div>
<script>
var age = 44; //Positive Integer
var status = -100; //Negative Integer
var gpa = 3.55; //Float
var out = "Age: " + age;
   out += "<br/>Status: " + status;
   out += "<br/>Gpa: " + gpa;
document.getElementById("output").innerHTML = out;
</script>
```

As we create the string stored in the variable `out`, we retrieve each of the values stored in the variables declared and initialized in the first part of the script. We retrieve the variable by simply naming it.

Do This: Why Does This Happen?

I've made some small alterations to the code below. Run the code in your browser by copying in to your text editor and saving it as a .html file. Why do you think the output appear the way it does? What does this tell you about double quotes in JavaScript?

```
<div id="output"></div>
<script>
var age = 44; //Positive Integer
var status = -100; //Negative Integer
var gpa = 3.55; //Float
var out = "Age: " + "age";
   out += "<br/>Status: " + "status";
   out += "<br/>Gpa: " + "gpa";
document.getElementById("output").innerHTML = out;
</script>
```

A Comment on Comments

You may have noticed that I used the // symbol in the code. These are comments. They are ignored by the browser and not processed. They are there for the developer. It's a good habit to use comments to document your code as you go—especially for anything that's confusing. Keep in mind you likely won't be the only one to see this code. Write your comments in a way that they are helpful to everyone.

A Couple of New Operators

You may have noticed a couple of new operators have entered the scene. We are already familiar with the assignment operator, and now we've got two more to examine.

Operator	Explanation
+	Concatenation operator. Designed to allow us to create Strings out of smaller strings. For example `"Neil" + " " + "Diamond"` results in `"Neil Diamond"`
+=	Add then assign. This is a variant of the assignment operator. The value to the right of operator is added to what is already stored in the variable. (I like to think of this as an "Append" operation)

If you like operators, you're in luck. We'll be learning more operators in future sections of the course.

"String" Variables

You've already discovered string variables. I think the term string is most easily defined by how the JavaScript processor sees it: "A series of random characters." The characters may mean something in some written language, but, to the JavaScript processor it's nothing more than a meaningless series of characters.

You declare and retrieve string variables the same way you do with numerical variables.

```
<div id="output"></div>
<script>
   var message = "The best bands on the planet:<br/>";
   message += "Journey<br/>";
   message += "Led Zeppelin<br/>";
   message += "The Cure<br/>";
```

```
    message += "Cheap Trick<br/>";
    message += "U2";
    document.getElementById('output').innerHTML = message;
</script>
```

Nothing really new in this code block. Run the code and you'll see the obvious result.

Do This: Output Your Own List of Favorite Bands with Strings

Similar to how I created the list of bands before, create your own list of favorite bands or artists. One wrinkle – Create your list in the CSV (Comma separated value) format. My list would appear like this as a CSV:

```
Journey, Led Zeppelin, The Cure, Cheap Trick, U2
```

Make Your First Calculation using a Arithmetic Operator

This is a bit of a preview as we'll get in to the serious arithmetic soon. Before I described the plus sign as the concatenation operator. And it is.

Except when it's not.

The '+' is an overloaded operator in JavaScript, which means it has more than one job. In addition to gluing strings together as the concatenation operator, '+' does addition. For example:

```
<script>
   age = 7;
   age = age + 10;

   alert("In 10 years you will be " + age + " years old.");
</script>
```

If you look at the second line of the script examine what's to the right of the assignment operator. (The right side of the assignment operator is always evaluated before assignment occurs.) Variable `age`, which has an initial value of 7, has 10 added and becomes 17. The value 17 is then reassigned to `age`. This is an example of the '+' being used for arithmetic, not concateantion.

Do This: Debugging

This code has several errors. See if you can debug it and get it working as it should. When working correctly it should display a story, inserting a nouns, adjectives and verbs as indicated. As is true in the real world, these errors are not always apparent and easy to spot. Good luck.

```
<html>
<head>
`<title>Debug #2</title>
</head>
<body>
   <div id="output"></div>
   <script>
      var location = prompt("In which city or town do you live?");
      var age = prompt("How old are you?");
      var pets = prompt("Do you have any pets?");
      var food = prompt("What is your favorite food?");

      var out = "This is my friend " + name;
```

```
    var out += name + " lives in " + location;
    var out += name + " is " + age " years old";
    var out += name + " likes to eat +" food;

    document.getElementById('out').innerHTML(out);
  </script>
</body>
</html>
```

Submit This: Lab Exercise

Starting with the code below:

```
<!DOCTYPE html>
<html>
<head>
  <title>Lab #2</title>
</head>
<body>
  <div id="out"></div>
  <script>
  name1 = "Tom";
  name2 = "Bob";
  name3 = "Mary";
  name4 = "Caitlyn";
  name5 = "Red";
  name6 = "Douglas";
  name7 = "Judy";
  name8 = "Susan";



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

1. Prompt the user for a list separator. (For example comma, space, the pipe symbol | , ampersand, or any other character can be used as a separator).
2. Print out the list using the separator the user input between each name. For example: Tom | Bob | Mary | Caitlyn | Red | Douglas | Judy | Susan .
3. Create a brand new program in a new file that prompts the user for a number. Print out the number. Add 10 to the number and print again. Add 20 and print again. Add 50 and print a fourth time. Finally add 1568 and print a final time.

   (If you have trouble read this article and see if it's helpful: https://www.w3schools.com/jsref/jsref_parseint.asp)