

Enhancing Web Pages with JavaScript

Introduction

- In this Tour we will cover:
 - The basic concepts of programming
 - How those concepts are translated into JavaScript
 - How JavaScript can be used to enhance web pages
- The basic concepts:
 - A **programming language** is an artificial language that is designed to communicate instructions to a computer.
 - A **program** is a series of steps written in a programming language that specify a task we want the computer to accomplish.
 - **Input** is information given from the user to the computer while a program is running.
 - **Output** is information provided from the computer to the user via a running program.
 - **Constants** are things that cannot change, *e.g.*, 2, 27, John.
 - **Variables** can be thought of as buckets that can hold values. This is like *x* and *y* in algebra.
 - **Functions** are small pieces of a program that can be used over and over to make programming easier.
 - **Conditional expressions** are expressions that ask a question, *e.g.*, is my checking account balance greater than 0?
- The programming language (JavaScript):
 - It is a scripting language which means it can talk to the computer from inside other programs that are running (for instance a web browser).
 - It is very popular for web programming.
 - It is not related to Java. Initially it was named Mocha, then it became LiveScript, but it was renamed JavaScript when Netscape agreed to bundle the Java runtime with its browser.
 - While JavaScript is primarily used to enhance websites, it is also used in other environments – for example in desktop widgets and the Unity game engine.
- Through this series of JavaScript programming steps you will learn how to enhance your web site that you developed in the previous module.

Guided Tour

There are a number of tutorials available for JavaScript if you wish to further explore its features or feel you need more information after this Tour. One example is: <http://www.w3schools.com/js/>

JavaScript Format

All JavaScript statements on a web page will either be between the **<head>** and the **</head>** or between the **<body>** and the **</body>**. You cannot start it in the head section and finish it in the body section. You can also put JavaScript code within some HTML tags to handle events. And sometimes you put JavaScript in a completely different file from the actual webpage file.

You insert JavaScript in either the head or the body using the **<script>** **</script>** tags. Since there are many kinds of scripting languages, you have to tell the computer that you're using JavaScript in the

script tag. So the tags really look like: `<script type = "text/javascript"> </script>`. Other things can be added to the script tag, but they are beyond the scope of this tutorial.

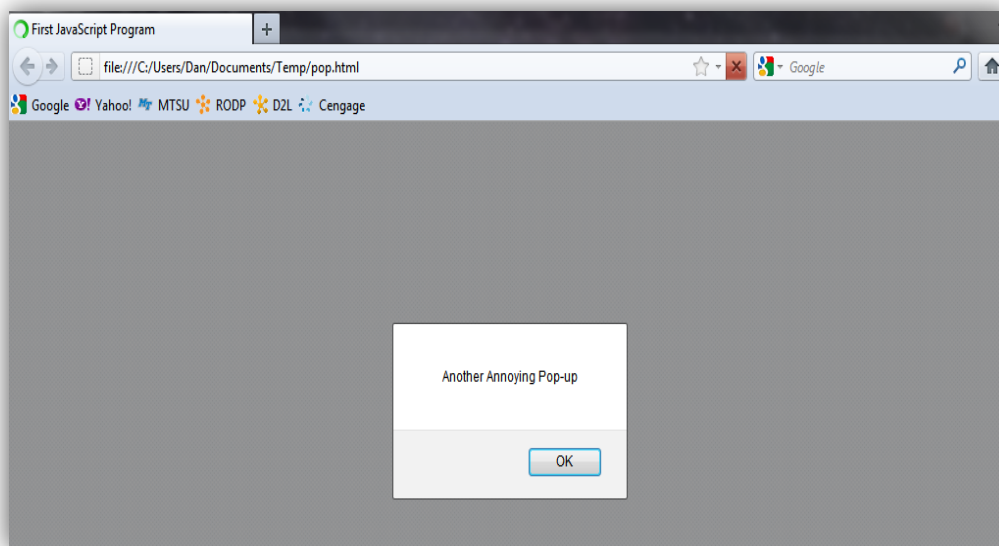
1. Open up the **Komodo editor** and type in the following JavaScript code to create your first JavaScript application

```
<!DOCTYPE html>
<html>
  <head>
    <title>First JavaScript Program</title>
  </head>
  <body>
    <script type="text/javascript">
      alert("Another Annoying Pop-up");
    </script>
  </body>
</html>
```

Type carefully. Remember that JavaScript requires perfect adherence to syntax.

2. Save the file you just created as **pop.html** (you can save it on your thumb drive or in the documents folder).
3. Open pop.html using Chrome or Firefox.

Your newly created web page should look like:



Creating Output Using JavaScript

Output statements are commands that tell the computer to print something somewhere. We might want something printed on the screen, or on a webpage, or we might want something printed in a file. For this

class, we will just print things to a webpage. In JavaScript, a way to print something to a webpage is to use the `document.write()` command. The syntax for that command is:

```
document.write("WhateverYouWantToPrint");
```

For example, if I want to print **Hello World** on the screen as an H1 heading, I would put the following command in a script:

```
document.write( "<h1>Hello World</h1>" );
```

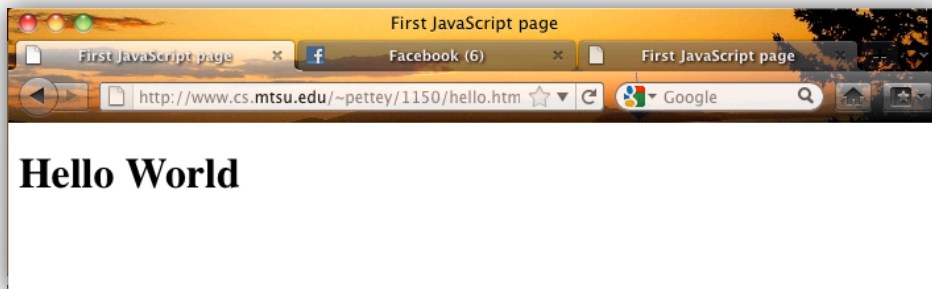
Notice that the line ends in a semicolon, and whatever HTML you want to write needs to be inside double quotes.

1. Create the following web page (**you can either create a new one or alter the preceding one**)

```
<!DOCTYPE html>
<html>
  <head>
    <title>First JavaScript page</title>
  </head>
  <body>

    <script type="text/javascript">
      document.write("<h1>Hello World</h1>");
    </script>
  </body>
</html>
```

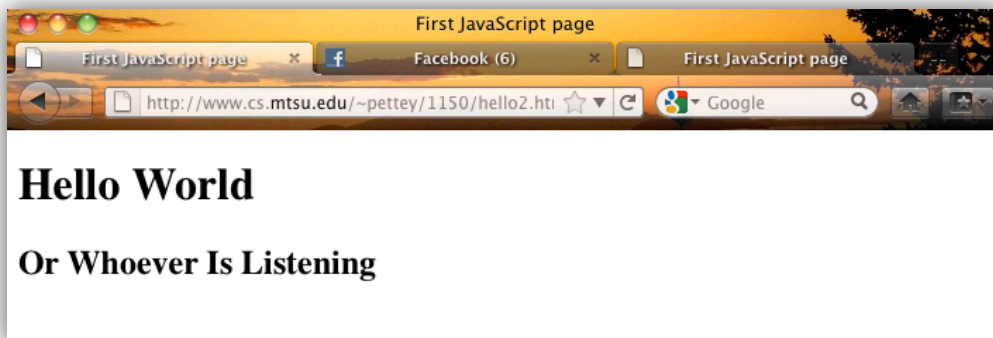
2. Open this file using Chrome or Firefox. Your new web page should look like:



3. You can have more than one output statement. Modify your page to print two things as follows:

```
<!DOCTYPE html>
<html>
  <head>
    <title>First JavaScript page</title>
  </head>
  <body>
    <script type="text/javascript">
      document.write("<h1>Hello World</h1>");
      document.write("<h2>Or Whoever Is Listening</h2>");
    </script>
  </body>
</html>
```

This will result in the following web page:



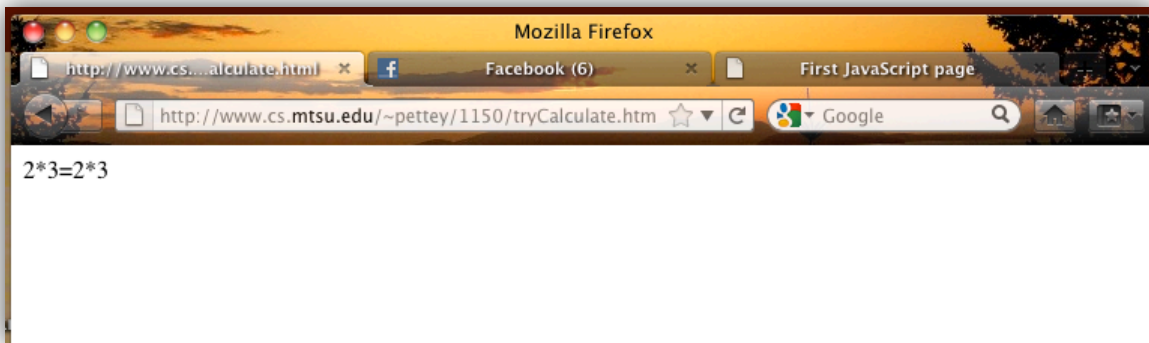
Your first thought might be, “I could have done that with one line of HTML! Why on earth would I want to put in three or four lines of JavaScript when one line of HTML will work?” Well, in the preceding example, you would be correct. But what if you wanted the page to actually perform some kind of calculation? For instance, what if you wanted to calculate the value of 2×3 ? Or the value of 2 times some user’s chosen value? The next section describes how JavaScript can be used to perform calculations.

Performing Calculations with JavaScript

Let’s try to calculate the value of 2×3 . Start by trying the following HTML code in Komodo editor and save it as **calculate.html**:

```
<!DOCTYPE html>
<html>
  <body>
    2*3 = 2*3
  </body>
</html>
```

Open this file using Chrome or Firefox. Unfortunately the results are:



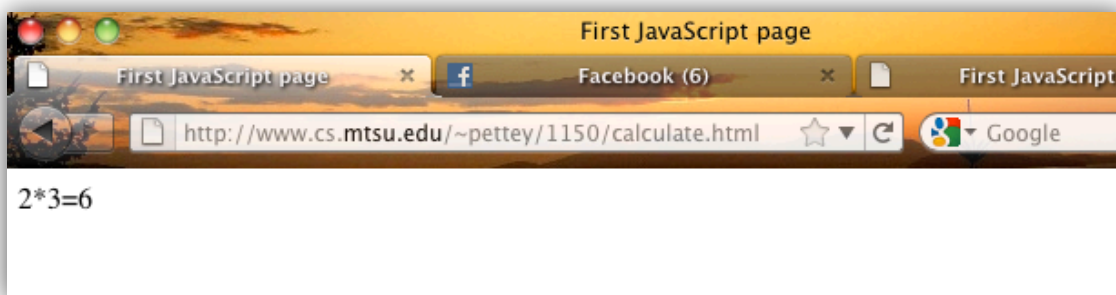
which is not at all what we wanted. I know, I know. You could have just put $2*3=6$, but the point is to get the computer to figure out the answer! To do this you need to make up a name to represent what information the computer will give us. These made up names are called **variables**. In this example a good made up name for what we want might be "answer". So we want to say something to the computer like: "Computer I need an answer to the problem $2*3$." And the JavaScript for that would be:

```
answer = 2*3;
```

What you want (your variable) comes first. What you want the computer to do to find your answer comes second. Note there's an "=" between what you want and what the computer has to do and a semicolon at the end of it all. Modify this web page to:

```
<!DOCTYPE html>
<html>
  <head>
    <title>First JavaScript page</title>
  </head>
  <body>
    <script type="text/javascript">
      answer = 2*3;
      document.write("2*3="+ answer);
    </script>
  </body>
</html>
```

The `document.write()` line looks a little funny. It turns out the only thing that you can write to a page is a string of characters, which you would normally put between double quotes in the `document.write()` line. But if you need to print something else, you can add it on to the list of characters being printed by using a "+". Your resulting web page should be:



Try the following example on your own to see what the resulting web page looks like:

```

<!DOCTYPE html>
<html>
  <head>
    <title>First JavaScript page</title>
  </head>
  <body>
    <script type="text/javascript">
      timesanswer = 2*3;
      plusanswer = 2+3;
      document.write("2*3="+ timesanswer + " and 2+3=" + plusanswer);
    </script>
  </body>
</html>

```

JavaScript Functions – Doing Something Over and Over the Easy Way

Let's look again at the web page that calculates the value of $2*3$. What if we wanted to calculate the value of $10*5$ and the value of $1024*4$ as well as the value of $2*3$. Well, we could just copy and paste lines of code and do a little changing of the lines and come up with something like the following:

```

<!DOCTYPE html>
<html>
  <head>
    <title>First JavaScript page</title>
  </head>
  <body>
    <script type="text/javascript">
      answer = 2*3;
      document.write("2*3="+ answer + "<br><br>");
      answer = 10*5;
      document.write("10*5="+ answer + "<br><br>");
      answer = 1024*4;
      document.write("1024*4="+ answer + "<br><br>");
    </script>
  </body>
</html>

```

This is not too bad for just three calculations. But what if I wanted to do 100 calculations, or what if I wanted to design a web-based calculator – which you probably wouldn't, but hang in there. Calculators are easy to understand because you've been using them for years, so they make good examples. 😊

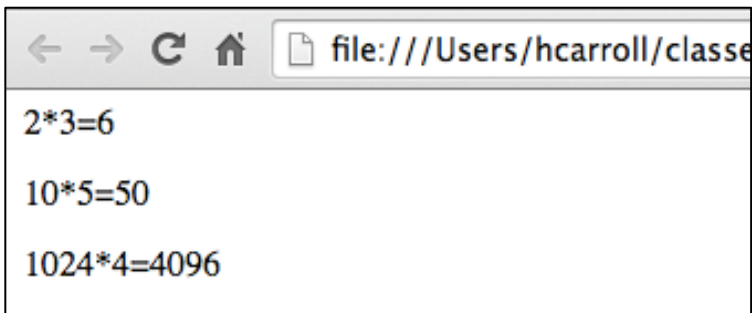
Anyway, back to the point. You don't want to have to copy and paste and then edit a bunch of lines. Instead, you can put a JavaScript function in the head and simply tell it to do the work with the appropriate numbers. Type in the following example that uses a function in **function.html** file:

```

<!DOCTYPE html>
<html>
  <head>
    <title>First JavaScript page</title>
    <script type="text/javascript">
      function calculateAnswers(number1, number2){
        timesanswer = number1*number2;
        document.write(number1 + "*" + number2 + "=" + timesanswer);
        document.write("<br><br>");
      }
    </script>
  </head>
  <body>
    <script type="text/javascript">
      calculateAnswers(2,3);
      calculateAnswers(10,5);
      calculateAnswers(1024,4);
    </script>
  </body>
</html>

```

Which should give you a page that looks like:



Notice the function appears in the **head** portion has a format like:

```

function NameOfFunction (list of variables the function needs to do its work){
  JavaScript statements to do the function
}

```

Don't forget the curly braces. Those are like the beginning and ending tags of a function. So in the above example, the name of the function is

calculateAnswers

The list of parameters (variables) the function needs to do its work is

number1, number2

The function is called in the **body** of the webpage by using its name and listing the constants you want the function to assign to its parameters. In our example the function is called three times.

Making Things More Flexible With User Input - **Modify function.html**

The preceding example would have been more interesting if the user could have determined which numbers to multiply. To do that we need an **HTML form**.

HTML forms are created using the **<form>** and **</form>** tags. As with other tags, we do not have time to cover all the attributes of form tags in this class. In order to get the user input, you need to place an

<input> tag inside the form tags. Input tags have various types (such as TEXT, BUTTON, and CHECKBOX to name a few). Each type has attributes associated with it. For instance

```
<input type="TEXT" size="10" name="num1">
```

says there will be a box on the page where the user can type in data and up to 10 characters will be visible in the box, and whatever they type in, we will call that num1.

Type in the following code to allow the user to input two numbers and pop up the answer in an alert:

```
<!DOCTYPE html>
<html>
  <head>
    <title>First JavaScript page</title>
    <script type="text/javascript">
      function calculateAnswers(number1,number2){
        timesanswer = number1*number2;
        alert("The answer is: " + timesanswer);
      }
    </script>
  </head>
  <body>
    Enter two numbers to be multiplied:
    <form>
      <p><b>Number1:</b> <input type="TEXT" size="10" name="num1" />
      <p><b>Number2:</b> <input type="TEXT" size="10" name="num2" />
      <p><input type="BUTTON" value="Calculate Answer" onClick =
"calculateAnswers(num1.value, num2.value);" /></p>
    </form>
  </body>
</html>
```

An alert is a simple popup window that has an ok button to close it. What goes in the parentheses of an alert is what you want the popup window to say.

The function is called in the third input tag. **onClick** is an event that is triggered when a user clicks on something (in this case a BUTTON). The

```
onClick = "calculateAnswers(num1.value, num2.value);"
```

tells the program to do the JavaScript function calculateAnswers using the values for num1 and num2 that the user typed in whenever they click the button. The value attribute in the third input tag tells what is displayed on the button.

Note: by using the form and function, the user can multiply however many pairs of numbers they want.

Making Decisions With JavaScript: **Modify function.html**

Change the preceding example so it will do division instead of multiplication:

```
<!DOCTYPE html>
<html>
  <head>
    <title>First JavaScript page</title>
    <script type="text/javascript">
      function calculateAnswers(number1,number2){
        divideanswer = number1/number2;
```



```

        alert("The answer is: " + divideanswer);
    }
</script>
</head>
<body>
    Enter two numbers to be divided:
    <form>
        <p><b>Number1:</b> <input type="TEXT" size="10" name="num1" />
        <p><b>Number2:</b> <input type="TEXT" size="10" name="num2" />
        <p><input type="BUTTON" value="Calculate Answer" onClick =
"calculateAnswers(num1.value, num2.value);" /></p>
    </form>
</body>
</html>

```

Now try out the web page and try several numbers. Be sure to try a divide by 0 to see what happens.

"The answer is infinity" is probably not what we want to happen. We'd rather have it say "Error! Divide by 0." That's where a conditional statement comes in handy. A conditional can be thought of as "If something happens, take some action, otherwise take a different action." More specifically, the JavaScript syntax is:

```

if( something happens){
    take some action
} else {
    take a different action
}

```

Notice the if and the else have curly braces to mark the beginning and ending of their section just like you do in a function. Of course "take some action" and "take a different action" are actual JavaScript statements. Frequently the "something happens" is just a comparison of two things (like "is this person's age over 18" or in our case "is num2 equal to 0"). There are six common comparisons:

Symbol	Meaning
==	Equal to, and yes that is two = signs right next to each other
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Given this information, the conditional statement in our example would look like

```
if( number2 == 0){
    alert("You cannot divide by 0");
} else {
    divideanswer = number1/number2;
    alert("The answer is: " + divideanswer);
}
```

Fix your web page so it correctly handles a divide by 0, and view it to see if it will work:

```
<!DOCTYPE html>
<html>
  <head>
    <title>First JavaScript page</title>
    <script type="text/javascript">
      function calculateAnswers(number1,number2){
        if(number2 == 0){
          alert("You cannot divide by 0");
        } else {
          divideanswer = number1/number2;
          alert("The answer is: " + divideanswer);
        }
      }
    </script>
  </head>
  <body>
    Enter two numbers to be divided:
    <form>
      <p><b>Number1:</b> <input type="TEXT" size="10" name="num1" />
      <p><b>Number2:</b> <input type="TEXT" size="10" name="num2" />
      <p><input type="BUTTON" value="Calculate Answer"
onClick="calculateAnswers(num1.value, num2.value);" /></p>
    </form>
  </body>
</html>
```

Upload your **function.html** to the CS webserver. Verify your webpage is available online at: <https://www.cs.mtsu.edu/~YourPipelineID/function.html>.

Now, make a new webpage called **simpleCalculator.html** that allows a user to input a number, a mathematical operator and another number. Have a button that uses those three inputs and calls a function. Display the mathematical equation and it's answer with a call to `document.write()`. **Your calculator should work for: +, -, *, and /**. For example, given the following inputs:

First Number:

Operator:

Second Number:

would result in 1+2=3. Additionally,

First Number:

Operator:

Second Number:

would result in: 3*7=21.

Note, if you have a line of JavaScript like:

```
answer = number1 + number2;
document.write("Answer: " + answer);
```

by default, JavaScript assumes that "+" means concatenation. If the variable `number1` is currently storing 3 and `number2` is currently storing 5 then, the following will be displayed:

```
Answer: 35
```

which is clearly not the answer. One way to make JavaScript add (instead of concatenate) is to first pass each variable through the `Number()` function:

```
answer = Number(number1) + Number(number2);
document.write("Answer: " + answer);
```

which, given the same values for the variables, will display:

```
Answer: 8
```

Rubric

Criteria	Points
Used exact filename, <code>simpleCalculator.html</code>	2
Interface (three input fields and a submit button)	4
Called function and passed it an operator character and numbers	2
Function definition, including 3 parameters	2
if-else statements testing the operator	8
Correctly displays results	8
Overall	5

More Examples

- * You'll find a collection of them including those detailed below at:
<http://www.cs.mtsu.edu/~djoaquin>
- * Calculate the average of three test scores <http://www.cs.mtsu.edu/~pettey/1150/form4.html>
- * Vote with alert after each button click <http://www.cs.mtsu.edu/~pettey/1150/vote.html>
- * Vote with alert only after announce the winner button is clicked
<http://www.cs.mtsu.edu/~pettey/1150/vote2.html>
- * Read info and send email if over 18 <http://www.cs.mtsu.edu/~pettey/1150/email.html>