**INT106 – Lab – Form and Event#3**

\*Credit: [www.w3schools.com](http://www.w3schools.com) and Jennifer Niederst Robbins, Learning web design 4th edition

**Event**

**Talk about JavaScript**

Despite its name**, JavaScript has nothing to do with Java.** It was created by Brendan Eich at Netscape in 1995 and originally named “LiveScript.” But Java was all the rage around that time, so for the sake of marketing, **“LiveScript” became “JavaScript.”**

JavaScript is a lightweight but incredibly powerful scripting language. We most frequently encounter it through our browsers, but **JavaScript has snuck into everything from native applications to PDFs to ebooks.** Even web servers themselves can be powered by JavaScript.

**The <script> Tag**

In HTML, JavaScript code must be inserted between <script> and </script> tags.

|  |
| --- |
| <script>  document.getElementById("demo").innerHTML = "My First JavaScript";  </script> |

**JavaScript Functions and Events**

A JavaScript function is a block of JavaScript code, that can be executed when "asked" for.

For example, a function can be executed when an event occurs, like when the user clicks a button.

**JavaScript in <head> or <body>?**

You can place any number of scripts in an HTML document. **Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both. But** keeping all code in one place, is always a good habit.

**JavaScript in <head>**

In this example, a JavaScript function is placed in the <head> section of an HTML page. The function is invoked (called) when a button is clicked:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_head>

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  **<script>**  **function myFunction() {**  **document.getElementById("demo").innerHTML = "Paragraph changed.";**  **}**  **</script>**  </head>  <body>  <h1>JavaScript in Head</h1>  <p id="demo">A Paragraph.</p>  <button type="button" **onclick="myFunction()"**>Try it</button>  </body>  </html> |

**JavaScript in <body>**

In this example, a JavaScript function is placed in the <body> section of an HTML page. The function is invoked (called) when a button is clicked:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_body>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>JavaScript in Body</h1>  <p id="demo">A Paragraph.</p>  <button type="button" **onclick="myFunction()"**>Try it</button>  **<script>**  **function myFunction() {**  **document.getElementById("demo").innerHTML = "Paragraph changed.";**  **}**  **</script>**  </body>  </html> |

**External JavaScript**

Scripts can also be placed in external files. External scripts are practical when the same code is used in many different web pages. JavaScript files have the file extension .js.

To use an external script, put the name of the script file in the src (source) attribute of the <script> tag:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_external>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>External JavaScript</h1>  <p id="demo">A Paragraph.</p>  <button type="button" onclick="myFunction()">Try it</button>  <p><strong>Note:</strong> myFunction is stored in an external file called "myScript.js".</p>  **<script src="myScript.js"></script>**  </body>  </html> |

You can place an external script reference in <head> or <body> as you like. The script will behave as if it was located exactly where the <script> tag is located.

**Note** External scripts cannot contain <script> tags.

**JavaScript Can Change HTML Content**

One of many HTML methods is getElementById().

This example uses the method to "find" an HTML element (with id="demo"), and changes the element content (innerHTML) to "Hello JavaScript":

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_inner_html>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>What Can JavaScript Do?</h1>  <p id="demo">JavaScript can change HTML content.</p>  <button type="button"  **onclick="document.getElementById('demo').innerHTML = 'Hello JavaScript!'"**>  Click Me!</button>  </body>  </html> |

**JavaScript Can Change HTML Attributes**

This example changes an HTML image, by changing the src attribute of an <img> tag:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_lightbulb>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>JavaScript Can Change Images</h1>  <img id="myImage" **onclick="changeImage()"** src="pic\_bulboff.gif" width="100" height="180">  <p>Click the light bulb to turn on/off the light.</p>  **<script>**  **function changeImage() {**  **var image = document.getElementById('myImage');**  **if (image.src.match("bulbon")) {**  **image.src = "pic\_bulboff.gif";**  **} else {**  **image.src = "pic\_bulbon.gif";**  **}**  **}**  **</script>**  </body>  </html> |

**JavaScript Can Change HTML Styles (CSS)**

Changing the style of an HTML element, is a variant of changing an HTML attribute:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_style>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>What Can JavaScript Do?</h1>  <p id="demo">JavaScript can change the style of an HTML element.</p>  **<script>**  **function myFunction() {**  **var x = document.getElementById("demo");**  **x.style.fontSize = "25px";**  **x.style.color = "red";**  **}**  **</script>**  <button type="button" **onclick="myFunction()"**>Click Me!</button>  </body>  </html> |

**JavaScript Can Validate Data**

JavaScript is often used to validate input:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_validate>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>JavaScript Can Validate Input</h1>  <p>Please input a number between 1 and 10:</p>  <input id="numb" type="number">  <button type="button" **onclick="myFunction()"**>Submit</button>  <p id="demo"></p>  <script>  **function myFunction() {**  **var x, text;**  **// Get the value of the input field with id="numb"**  **x = document.getElementById("numb").value;**  **// If x is Not a Number or less than one or greater than 10**  **if (isNaN(x) || x < 1 || x > 10) {**  **text = "Input not valid";**  **} else {**  **text = "Input OK";**  **}**  **document.getElementById("demo").innerHTML = text;**  **}**  </script>  </body>  </html> |

**JavaScript Output**

**JavaScript Display Possibilities**

JavaScript can "display" data in different ways:

Writing into an alert box, using **window.alert()**.

Writing into the HTML output using **document.write()**.

Writing into an HTML element, using **innerHTML**.

Writing into the browser console, using **console.log()**.

**Using window.alert()**

You can use an alert box to display data:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_output_alert>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>My First Web Page</h1>  <p>My first paragraph.</p>  <script>  window.alert(5 + 6);  </script>  </body>  </html> |

**Using document.write()**

For testing purposes, it is convenient to use document.write():

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_output_write>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>My First Web Page</h1>  <p>My first paragraph.</p>  <script>  document.write(5 + 6);  </script>  </body>  </html> |

Using document.write() after an HTML document is fully loaded, will delete all existing HTML:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_output_write_over>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>My First Web Page</h1>  <p>My first paragraph.</p>  <button type="button" **onclick="document.write(5 + 6)"**>Try it</button>  </body>  </html> |

**\*\*\*The document.write() method should be used only for testing.**

**Using innerHTML**

To access an HTML element, JavaScript can use the document.getElementById(id) method.

The id attribute defines the HTML element. The innerHTML property defines the HTML content:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_output_dom>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>My First Web Page</h1>  <p>My First Paragraph.</p>  <p id="demo"></p>  <script>  **document.getElementById("demo").innerHTML = 5 + 6;**  </script>  </body>  </html> |

**\*\*\*We usually use the innerHTML method to write into an HTML element.**

**Using console.log()**

In your browser, you can use the console.log() method to display data.

Activate the browser console with F12, and select "Console" in the menu.

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_output_console>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>My First Web Page</h1>  <p>My first paragraph.</p>  <p>  Activate debugging in your browser (Chrome, IE, Firefox) with F12, and select "Console" in the debugger menu.  </p>  <script>  **console.log(5 + 6);**  </script>  </body>  </html> |

**JavaScript Syntax**

**JavaScript Programs**

A computer program is a list of "instructions" to be "executed" by the computer. In a programming language, these program instructions are called statements.

JavaScript is a programming language.

**JavaScript statements are separated by semicolon.**

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_syntax_statements>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1>JavaScript Statements</h1>  <p>Statements are separated by semicolons.</p>  <p>The variables x, y, and z are assigned the values 5, 6, and 11:</p>  <p id="demo"></p>  **<script>**  **var x = 5;**  **var y = 6;**  **var z = x + y;**  **document.getElementById("demo").innerHTML = z;**  **</script>**  </body>  </html> |

**JavaScript Statements**

JavaScript statements are composed of:

**Values, Operators, Expressions, Keywords, and Comments.**

**JavaScript Values**

The JavaScript syntax defines two types of values: **Fixed values and variable values**.

Fixed values are called **literals.**

Variable values are called **variables.**

**JavaScript Literals**

The most important rules for writing fixed values are:

**Numbers** are written with or without decimals:

|  |
| --- |
| 10.50  1001 |

**Strings** are text, written within double or single quotes:

|  |
| --- |
| "John Doe"  'John Doe' |

**Expressions** can also represent fixed values:

|  |
| --- |
| 5 + 6  5 \* 10 |

**JavaScript Variables**

JavaScript uses the var keyword to define variables. An equal sign is used to assign values to variables.

In this example, x is defined as a variable. Then, x is assigned (given) the value 6:

|  |
| --- |
| var x;  x = 6; |

**JavaScript Operators**

JavaScript uses an assignment operator ( = ) to assign values to variables:

|  |
| --- |
| var x = 5;  var y = 6; |

JavaScript uses arithmetic operators ( + - \* / ) to compute values:

|  |
| --- |
| (5 + 6) \* 10 |

**JavaScript Keywords**

JavaScript keywords are used to identify actions to be performed.

The var keyword tells the browser to create a new variable:

|  |
| --- |
| var x = 5 + 6;  var y = x \* 10; |

**JavaScript Comments**

Not all JavaScript statements are "executed".

Code after double slashes // or between /\* and \*/ is treated as a comment.

Comments are ignored, and will not be executed:

|  |
| --- |
| var x = 5; // I will be executed  // var x = 6; I will NOT be executed |

**JavaScript is Case Sensitive**

All JavaScript identifiers are case sensitive.

The variables lastName and lastname, are two different variables.

|  |
| --- |
| lastName = "Doe";  lastname = "Peterson"; |

**\*\*\*JavaScript does not interpret VAR or Var as the keyword var.**

**JavaScript and Camel Case**

Historically, programmers have used three ways of joining multiple words into one variable name:

**Hyphens:**

first-name, last-name, master-card, inter-city.

**Underscore:**

first\_name, last\_name, master\_card, inter\_city.

**Camel Case:**

FirstName, LastName, MasterCard, InterCity.

**\*\*\*In programming languages, especially in JavaScript, camel case often starts with a lowercase letter:**

**firstName, lastName, masterCard, interCity.**

**\*\*\*Hyphens are not allowed in JavaScript. It is reserved for subtractions.**

**JavaScript Statements**

This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_statement>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p>In HTML, JavaScript statements are "commands" to the browser.</p>  <p id="demo"></p>  <script>  **document.getElementById("demo").innerHTML = "Hello Dolly.";**  </script>  </body>  </html> |

**JavaScript Programs**

Most JavaScript programs contain many JavaScript statements. The statements are executed, one by one, in the same order as they are written.

In this example, x, y, and z is given values, and finally z is displayed:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_statements>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p>JavaScript code (or just JavaScript) is a list of JavaScript statements.</p>  <p id="demo"></p>  <script>  var x = 5;  var y = 6;  var z = x + y;  document.getElementById("demo").innerHTML = z;  </script>  </body>  </html> |

**Semicolons ;**

Semicolons separate JavaScript statements.

Add a semicolon at the end of each executable statement:

|  |
| --- |
| a = 5;  b = 6;  c = a + b; |

Equal to

|  |
| --- |
| a = 5; b = 6; c = a + b; |

**JavaScript White Space**

JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

The following lines are equivalent:

|  |
| --- |
| var person = "Hege";  var person="Hege"; |

But you should put spaces around operators ( = + - \* / ):

|  |
| --- |
| var x = y + z; |

**JavaScript Line Length and Line Breaks**

For best readability, programmers often like to avoid code lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it, is after an operator:

|  |
| --- |
| document.getElementById("demo").innerHTML =  "Hello Dolly."; |

**JavaScript Code Blocks**

JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.

The purpose of code blocks this is to define statements to be executed together.

One place you will find statements grouped together in blocks, are in JavaScript functions:

|  |
| --- |
| function myFunction() {  document.getElementById("demo").innerHTML = "Hello Dolly.";  document.getElementById("myDIV").innerHTML = "How are you?";  } |

**JavaScript Keywords**

JavaScript statements often start with a keyword to identify the JavaScript action to be performed.

Here is a list of some of the keywords you will learn about in this tutorial:

**Keyword Description**

break Terminates a switch or a loop

continue Jumps out of a loop and starts at the top

debugger Stops the execution of JavaScript, and calls (if available) the debugging function

do ... while Executes a block of statements, and repeats the block, while a condition is true

for Marks a block of statements to be executed, as long as a condition is true

function Declares a function

if ... else Marks a block of statements to be executed, depending on a condition

return Exits a function

switch Marks a block of statements to be executed, depending on different cases

try ... catch Implements error handling to a block of statements

var Declares a variable

**JavaScript Comments**

Just like Java comments

**JavaScript Variables**

Only var, no other data types.

|  |
| --- |
| var pi = 3.14;  var person = "John Doe";  var answer = 'Yes I am!'; |

**JavaScript Identifiers**

Just like Java.

**One Statement, Many Variables**

You can declare many variables in one statement.

Start the statement with var and separate the variables by comma:

|  |
| --- |
| var person = "John Doe", carName = "Volvo", price = 200; |

Equal to

|  |
| --- |
| var person = "John Doe",  carName = "Volvo",  price = 200; |

**Re-Declaring JavaScript Variables**

If you re-declare a JavaScript variable, it will not lose its value.

The variable carName will still have the value "Volvo" after the execution of these statements:

|  |
| --- |
| var carName = "Volvo";  var carName; |

JavaScript Arithmetic

As with algebra, you can do arithmetic with JavaScript variables, using operators like = and +:

|  |
| --- |
| var x = 5 + 2 + 3; |

You can also add strings, but strings will be concatenated (added end-to-end):

|  |
| --- |
| var x = "John" + " " + "Doe"; |

Also try this:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_variables_add_string_number>

|  |
| --- |
| var x = "5" + 2 + 3; |

**Practice#1**

|  |
| --- |
| Open the file “twenties.html”  **1. For each part** of the web page, create **a button with JavaScript** to change the content’s appearance into what appear in the file “twenties-final.html”  **First part start at:**  The Back of the New $20  **Second part start at:**  Too Many 20s  **The Last part start at:**  Connect-the-Dots  **2.** Make the image on the web page change into another image (any image you choose) when clicked.  **3.** When the mouse point at the image on the web page (mouse over), show a message “TOO MANY 20s !!!”    **4.** Somewhere on the page,  - Create 2 input boxes those accept only numbers.  - Create 4 buttons shown as + button, - button, x button and / button.  - Each button has JavaScript to calculate the 2 input values as the button says.  - Show the result. |