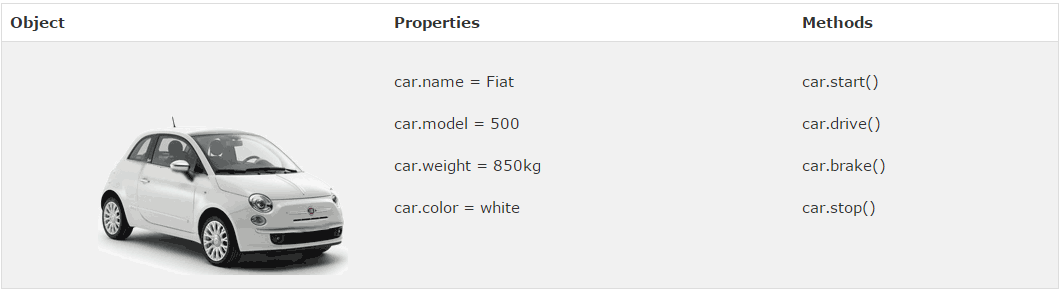
**INT106 – Lab – JavaScript#2**

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

**JavaScript Function is used the same way as Java’s method.**

**JavaScript Object is also used the same way as Java’s object, too.**



Objects are variables too. But objects can contain many values.

This code assigns many values (Fiat, 500, white) to a variable named car:

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

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p>Creating a JavaScript Object.</p>  <p id="demo"></p>  <script>  **var car = {type:"Fiat", model:500, color:"white"};**  document.getElementById("demo").innerHTML = car.type;  </script>  </body>  </html> |

Type, model and color are called the property.

“Fiat”, 500 and “white” are called the property value.

**Object Methods**

Methods are actions that can be performed on objects.

Methods are stored in properties as function definitions.

**Property Property Value**

firstName John

lastName Doe

age 50

eyeColor blue

fullName function() {return this.firstName + " " + this.lastName;}

**Object Definition**

You define (and create) a JavaScript object with an object literal:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_objects_create_1>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p>Creating a JavaScript Object.</p>  <p id="demo"></p>  <script>  **var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};**  document.getElementById("demo").innerHTML =  person.firstName + " is " + person.age + " years old.";  </script>  </body>  </html> |

Use the object’s value by:   
objectName.propertyName

Example: person.firstName

**Object with method (function)**

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

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p>An object method is a function definition, stored as a property value.</p>  <p>If you access it without (), it will return the function definition:</p>  <p id="demo"></p>  <script>  var person = {  firstName: "John",  lastName : "Doe",  id : 5566,  **fullName : function() {**  **return this.firstName + " " + this.lastName;**  }  };  document.getElementById("demo").innerHTML = person.fullName;  </script>  </body>  </html> |

Use the object’s method by: objectName.methodName  
For Example: person.fullName;

**Note: Do Not Declare Strings, Numbers, and Booleans as Objects!**

var x = new String(); // Declares x as a String object

var y = new Number(); // Declares y as a Number object

var z = new Boolean(); // Declares z as a Boolean object

Avoid String, Number, and Boolean objects. They complicate your code and slow down execution speed.

**\*JavaScript has Global and Local variables too**

**Automatically Global**

If you assign a value to a variable **that has not been declared**, it will automatically become a **GLOBAL** variable.

**The Lifetime of JavaScript Variables**

The lifetime of a JavaScript variable starts when it is declared.

**Local** variables are deleted **when the function is completed**.

**Global** variables are deleted **when you close the page**.

**JavaScript Events**

See all the events that can be used here:

<http://www.w3schools.com/jsref/dom_obj_event.asp>

\*It is a VERY LONG list.

Example:

**Mouse Events**

Event Description

onclick The event occurs when the user clicks on an element

oncontextmenu The event occurs when the user right-clicks on an element to open a context menu

ondblclick The event occurs when the user double-clicks on an element

onmousedown The event occurs when the user presses a mouse button over an element

onmouseenter The event occurs when the pointer is moved onto an element

onmouseleave The event occurs when the pointer is moved out of an element

onmousemove The event occurs when the pointer is moving while it is over an element

onmouseover The event occurs when the pointer is moved onto an element, or onto one of its children

onmouseout The event occurs when a user moves the mouse pointer out of an element, or out of one of its children

onmouseup The event occurs when a user releases a mouse button over an element 2

**Keyboard Events**

Event Description

onkeydown The event occurs when the user is pressing a key

onkeypress The event occurs when the user presses a key

onkeyup The event occurs when the user releases a key

**Form Events**

Event Description

onblur The event occurs when an element loses focus

onchange The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <keygen>, <select>, and <textarea>)

onfocus The event occurs when an element gets focus

onfocusin The event occurs when an element is about to get focus

onfocusout The event occurs when an element is about to lose focus

oninput The event occurs when an element gets user input

oninvalid The event occurs when an element is invalid

onreset The event occurs when a form is reset

onsearch The event occurs when the user writes something in a search field (for <input="search">)

onselect The event occurs after the user selects some text (for <input> and <textarea>)

onsubmit The event occurs when a form is submitted

And a lot more waiting for you to find out.

**String Length**

The length of a string is found in the built in property length:

Example

var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

var sln = **txt.length;**

**Special Characters**

Because strings must be written within quotes, JavaScript will misunderstand this string:

var y **= "We are the so-called "Vikings" from the north."**

The string will be chopped to "We are the so-called ".

The solution to avoid this problem, is to use the \ escape character.

The backslash escape character turns special characters into string characters:

var y = "We are the so-called **\"Vikings\"** from the north."

**JavaScript Arrays**

Displaying Arrays

In this tutorial we will use a script to display arrays inside a <p> element with id="demo":

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

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p id="demo"></p>  <script>  **var cars = ["Saab", "Volvo", "BMW"];**  document.getElementById("demo").innerHTML = **cars[0]**;  </script>  </body>  </html> |

**Array Properties and Methods**

The real strength of JavaScript arrays are the built-in array properties and methods:

Examples

var x = cars.length; // The length property returns the number of elements in cars

var y = cars.sort(); // The sort() method sort cars in alphabetical order

**Adding Array Elements**

The easiest way to add a new element to an array is to use the length property:

Example

var fruits = ["Banana", "Orange", "Apple", "Mango"];

**fruits[fruits.length] = "Lemon";** // adds a new element (Lemon) to fruits

**Looping Array Elements**

The best way to loop through an array, is using a "for" loop:

Example

|  |
| --- |
| var index;  var fruits = ["Banana", "Orange", "Apple", "Mango"];  **for (index = 0; index < fruits.length; index++)** {  text += fruits[index];  } |

**JavaScript Array Methods**

**Converting Arrays to Strings**

In JavaScript, all objects have the valueOf() and toString() methods.

The valueOf() method is the default behavior for an array. **It returns an array as a string**:

For JavaScript arrays, **valueOf() and toString() are equal**.

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

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p>The valueOf() method returns an array as a comma separated string.</p>  <p id="demo"></p>  <script>  var fruits = ["Banana", "Orange", "Apple", "Mango"];  **document.getElementById("demo").innerHTML = fruits.valueOf();**  **//document.getElementById("demo").innerHTML = fruits.toString();**  </script>  </body>  </html> |

Try comment and uncomment the 2 lines above, the result will be the same.

**The join()** method also joins all array elements into a string.

It behaves just like toString(), but you can specify the separator:

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

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p>the join() method joins array elements into a string.</p>  <p id="demo"></p>  <script>  var fruits = ["Banana", "Orange", "Apple", "Mango"];  document.getElementById("demo").innerHTML = **fruits.join(" \* ");**  </script>  </body>  </html> |

Result:

|  |
| --- |
| the join() method joins array elements into a string.  Banana \* Orange \* Apple \* Mango |

**Popping and Pushing**

When you work with arrays, it is easy to remove elements and add new elements.

This is what popping and pushing is: Popping items out of an array, or pushing items into an array.

The **pop()** method removes the last element from an array:

|  |
| --- |
| var fruits = ["Banana", "Orange", "Apple", "Mango"];  fruits.pop(); // Removes the last element ("Mango") from fruits |

The **push()** method adds a new element to an array (at the end):

|  |
| --- |
| var fruits = ["Banana", "Orange", "Apple", "Mango"];  fruits.push("Kiwi"); // Adds a new element ("Kiwi") to fruits |

\* The pop() method returns the string that was "popped out".

The push() method returns the new array length.

**JavaScript Best Practices**

**Avoid Global Variables**

Minimize the use of global variables.

This includes all data types, objects, and functions.

Global variables and functions can be overwritten by other scripts.

Use local variables instead, and learn how to use closures.

**Always Declare Local Variables**

All variables used in a function should be declared as local variables.

Local variables must be declared with the var keyword, otherwise they will become global variables.

**Declarations on Top**

It is a good coding practice to put all declarations at the top of each script or function.

This will:

Give cleaner code

Provide a single place to look for local variables

Make it easier to avoid unwanted (implied) global variables

Reduce the possibility of unwanted re-declarations

\* By default, JavaScript moves all declarations to the top (JavaScript hoisting).

**Initialize Variables**

It is a good coding practice to initialize variables when you declare them.

This will:

Give cleaner code

Provide a single place to initialize variables

Avoid undefined values

**Never Declare Number, String, or Boolean Objects**

Always treat numbers, strings, or booleans as primitive values. Not as objects.

Declaring these types as objects, slows down execution speed, and produces nasty side effects:

**Don't Use new Object()**

Use {} instead of new Object()

Use "" instead of new String()

Use 0 instead of new Number()

Use false instead of new Boolean()

Use [] instead of new Array()

Use /()/ instead of new RegExp()

Use function (){} instead of new function()

**Beware of Automatic Type Conversions**

Beware that numbers can accidentally be converted to strings or NaN (Not a Number).

JavaScript is loosely typed. A variable can contain different data types, and a variable can change its data type:

**\*Use === Comparison**

The == comparison operator always converts (to matching types) before comparison.

The === operator forces comparison of values and type:

“0”==0

**End Your Switches with Defaults**

Always end your switch statements with a default. Even if you think there is no need for it.

|  |
| --- |
| switch (new Date().getDay()) {  case 0:  day = "Sunday";  break;  case 1:  day = "Monday";  break;  case 2:  day = "Tuesday";  break;  case 3:  day = "Wednesday";  break;  case 4:  day = "Thursday";  break;  case 5:  day = "Friday";  break;  case 6:  day = "Saturday";  break;  default:  day = "Unknown";  } |

**JavaScript Common Mistakes**

**Misunderstanding Floats**

All numbers in JavaScript are stored as 64-bits Floating point numbers (Floats).

All programming languages, including JavaScript, have difficulties with precise floating point values:

|  |
| --- |
| var x = 0.1;  var y = 0.2;  var z = x + y // the result in z will not be 0.3  if (z == 0.3) // this if test will fail |

Result:

|  |
| --- |
| 0.30000000000000004 |

**JavaScript Performance**

**Reduce Activity in Loops**

Loops are often used in programming.

Each statement in a loop, including the for statement, is executed for each iteration of the loop.

Search for statements or assignments that can be placed outside the loop.

**Bad Code:**

|  |
| --- |
| for (i = 0; i < arr.length; i++) { |

**Better Code:**

|  |
| --- |
| l = arr.length;  for (i = 0; i < l; i++) { |

**Reduce DOM Access**

**Accessing the HTML DOM is very slow**, compared to other JavaScript statements.

If you expect to access a DOM element several times, **access it once, and use it as a local variable**:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p id="demo"></p>  <script>  var obj;  **obj = document.getElementById("demo");**  **obj.innerHTML = " Hello";**  </script>  </body>  </html> |

**Reduce DOM Size**

Keep the number of elements in the HTML DOM small.

This will always improve page loading, and speed up rendering (page display), especially on smaller devices.

Every attempt to search the DOM (like getElementsByTagName) will benefit from a smaller DOM.

**Avoid Unnecessary Variables**

Don't create new variables if you don't plan to save values.

Often you can replace code like this:

**Bad Code:**

|  |
| --- |
| var fullName = firstName + " " + lastName;  document.getElementById("demo").innerHTML = fullName; |

**Better Code:**

|  |
| --- |
| document.getElementById("demo").innerHTML = firstName + " " + lastName |

**Delay JavaScript Loading**

**Putting your scripts at the bottom of the page body**, lets the browser load the page first.

While a script is downloading, the browser will not start any other downloads. In addition all parsing and rendering activity might be blocked.

|  |
| --- |
| **Practice** Write a HTML page for an online shopping web site with the follow requirements:   1. Shopping cart must be an object. 2. The shopping cart can hold only 1 type of item at a time. 3. Adding item into the cart need 3 information: itemID, itemPrice and itemAmount. Make the input boxes and the button for the task. The user cannot add item if there is already an item in the cart. 4. Make a remove button for removing the item from the cart. 5. Make text boxes to always show the itemID, itemPrice, itemAmount and total price of the item in the cart. If the cart is empty, show zero. 6. All of the above, should be wrote according to the Best practice and performance mentioned in this document. 7. For further challenge, enable the cart to hold more than 1 type of item. Enable the user to choose the item to be removed from the cart. Expand the report area from NO.5 to cover many items. |

**Storing object inside an array might be needed. (Credit: stackoverflow.com)**

Put anything into an array using **Array.push().**

var a=[], b={‘id’:’111’, ‘name’:’Extra’};

a.push(b);

// a[0] === b;

Call for the property of the object inside and array using

itemName = a[0].name;