COMP 2015

Introduction to JavaScript and jQuery

Contents

Different DOMs 3

If Statements 4

Logical Operators 5
- Notes On && And || . . . 5

Some Useful Methods For Testing Values 6
- isNaN() . . . 6
- parseInt() . . . 6
- indexOf() . . . 6

Arrays 7
- Creating An Array . . . 7

Important Note About Arrays 8

For Loops 9

Dates 10

Date Methods 11

Next Week 12

Different DOMs

The Document Object Model (DOM) is a set of code JavaScript can use to interact with the browser and the Web Page.

Since the mid 1990s, the DOM has evolved. The history in a nutshell is:

- Browsers existed first without any JavaScript
- JavaScript was then implemented in Netscape
- JScript – a JavaScript clone – was then implemented in Internet Explorer

DOM 0 is now used to refer to these "olden days" in which each browser did things their own way. Most references to the web page's elements were done either very awkwardly or with respect to the position of the element on the page.

Some examples:

`document.images` is a collection of all the images; `document.images[0]` is the first image on the page, etc.…

`document.forms` is a collection of all the forms; `document.forms[4]` is the fifth form on the page, etc.…

`document.links` is a collection of all the hyperlinks; `document.links[99]` is the hundredth link on the page, etc.…

There is no DOM 0 shortcut for other collections, such as `document.paragraphs`, etc. For those, we will use newer DOMs such as DOM 1.

Eventually, the W3C came to the rescue and implemented the new DOM standards, in the best interests of web developers and web users alike. The new code looked like:

```
document.getElementById("d").src="dog.jpg";
```
or
```
document.getElementsByTagName("p");
```

We will use DOM 1 (and 2 and 3; also created by the W3C) until we later augment it by using jQuery.

If Statements

Very often, a programmer writes pieces of code which should only be executed if some certain conditions apply.

Real-life examples could include:

- if the distance to the store is two miles or less, I will walk
- if it is not snowing, I will walk to the store
- if the car is broken down and I have no bus fare, I will walk to the store
- if it is sunny or cloudy, I will walk to the store

examples:

```
if (distance <= 2.0){
    walk_to_store = true;
}
if (!snowing){
    walk_to_store = true;
}
if (snowing == false){
    walk_to_store = true;
}
if ((car_is_broken == true) && (have_bus-fare == false)){
    walk_to_store = true;
}
if ((sunny == true) || (cloudy == true)){
    walk_to_store = true;
}
```

Logical Operators

When using If Statements, we use Logical Operators to compare two or more values

```
!   not
||  or
&&  and
==  equal (same value)
=== identical (same value AND same data type)
!=  not equal
!== not identical
>   greater than
>=  greater than/equal
<   less than
<=  less than/equal
```

Notes On && And ||

A statement using And (&&) only evaluates to true when *both* arguments are true. This can be demonstrated simply, using plain English, using the following logical statement:

```
Chris teaches COMP 2015 && Chris is 7 feet tall
```

Clearly the above statement, as a whole, is not true because Chris is not 7 feet tall. Thus, if either argument is false (Chris does not teach COMP 2015, or Chris is not 7 feet tall) or both arguments are false, then the whole statement will be false.

Similarly, when using Or (||) the statement only evaluates to false when *both* arguments are false. Using the same arguments as above:

```
Chris teaches COMP 2015 || Chris is 7 feet tall
```

The statement as a whole is not entirely false because Chris does teach COMP 2015. Thus, if either argument is true or both arguments are true, the whole statement will be considered true.

Some Useful Methods For Testing Values

isNaN()

Often we will need to test variables to see if the value they contain is a number or not. For example, when a user fills out a form on a webpage, we may want to ensure that the age they enter is in fact a number (and not, for example, a word).

For this we can use the “is not a number” method, isNaN()

examples:

```
var a = 5;

isNaN(a);   // false, it IS a number

var b = 'bcit';

isNaN(b);   // true, it IS NOT a number
```

parseInt()

Next, we can use the parseInt() method to extract numbers from the beginning of a string. Useful when we want to grab only the numerical portion of a CSS value like “10px”

example:

```
var myCSSvalue = '50px';

parseInt(myCSSvalue);   // returns 50
```

indexOf()

Last, we can use indexOf() to search a string for a given substring. Useful, for example, if we want to find all the links on a page that contain the text ‘BCIT’.

example:

```
var link = 'http://bcit.ca';

if (link.indexOf('bcit') >= 0) {    // 0 is the first position in the string
    alert('found a link with bcit!');
}
```

Arrays

Arrays are special data types. They are a single variable which can contain multiple values.

Use arrays to store information about a collection of related data

e.g. a list of products
e.g. a series of images for an animation

Arrays use an integer index (starting with zero) to keep track of where particular values are stored, and values can be accessed using 'square bracket' notation.

example:

```
var arrayOfCities = ['Vancouver', 'Richmond', 'Burnaby'];

alert(arrayOfCities[0])  // alerts 'Vancouver'
alert(arrayOfCities[1])  // alerts 'Burnaby'
```

Creating An Array

Arrays are also Objects, and can be created using the 'new' operator.

example:

```
var listOfSchools = new Array();  // create the array

listOfSchools[0] = 'BCIT';      // store a value in the array

alert(listOfSchools[1]);        // access a different stored value
```

Because Arrays are Objects, we can also use their properties and methods.

example:

```
var listOfPeople = new Array();

listOfPeople.push('chris');   // add a new value to the end
                              // of the array

listOfPeople.push('jason');   // add another value

alert(listOfPeople.length);   // accessing the length property,
                              // which returns 2 because there are two entries
```

Note that when we use an Object's property, we do not need parentheses.

example:

```
alert(listOfPeople.length);   // correct

alert(listOfPeople.length()); // incorrect
```

Important Note About Arrays

Often we will need to set the properties of several 'members' of an array (e.g. change the text color of every paragraph to red).

In order to accomplish this, we have to **loop** through an array of paragraphs and set the properties of each paragraph individually. They cannot be set all at once!

example:

```
var allParagraphs = document.getElementsByTagName('p');

// allParagraphs is now an array of paragraph elements

allParagraphs.style.color = 'red';  // ERROR!
```

For Loops

'For' loops provide us with a way to quickly and easily run through all the elements in an array. For loops can also be used to repeat any arbitrary section of code a specified number of times.

The syntax for the For Loop is:

```
for (let counter = 0; counter < number_of_repetitions; counter++) {
    // all of the code between the braces is repeated
}
```

We can thus loop through our array of paragraphs and change their text color to red like this:

```
var allParagraphs = document.getElementsByTagName('p');

for (let i = 0; i < allParagraphs.length; i++) {
    allParagraphs[i].style.color = 'red';
}
```

Dates

JavaScript's Date object and its methods are very useful for working with dates and times.

This is the technique to get the current date stored into a Date variable:

```
var today = new Date();
```

JavaScript can determine the date and time on the client's (browser's) computer, but it can also create dates and times.

These can be used for display purposes, or to calculate spans of time.

e.g. "5 more days until January 1"

You can also create Date objects which are not dependent on the current date.

example:

```
var newYearsEve = new Date(2018, 11, 31);
```

Note: the month count for custom dates ranges from 0 to 11, not 1 to 12! Thus, January is month 0 and December is month 11.

Date Methods

The getTime() method returns an integer representing the number of milliseconds since midnight, January 1, 1970, which can be used to determine the number of hours, days, weeks etc. between two dates.

example:

```
var newYearsEve = new Date(2017,11,31);
var today = new Date();

var difference = newYearsEve.getTime() - today.getTime();

var minutes = 1000*60;  //1000 millisec per sec; 60 sec/hr
var hours = minutes*60; //60 minutes per hour
var days = hours*24;    //24 hours per day

alert(difference/days+" days until new years eve!");
```

Next Week

Lab 2 due before the beginning of Lesson 3.

Download it from our course section of COMP2015 on D2L http://learn.bcit.ca (COMP2015 > Content > Lesson 2)

Quiz 2 at the beginning of Lesson 3.

Quiz topics can also be found on D2L under COMP2015 > Content > Lesson 2