JavaScript Basic

Week 4 Session2

Contents of This session

- JavaScript Basic
  - Objects
  - Event models and Event handling

Object-oriented JavaScript

- JavaScript is actually object-based, supporting a collection of properties and methods for an object

- There are four types of objects
  - Built-in objects
  - Browser objects
  - Document objects (DOM)
  - User-defined objects

Why should we care about objects?

- Document and browser objects are “objects”, and need to be manipulated
- JavaScript objects are a convenient way to handle structured data sent from the server in the JSON format
- Objects can help us to structure programs more easily
- In JavaScript, objects have properties and methods
- They are accessed via the same “dot notation” as in Java, e.g., browser.alert(“Hello world”);

Built-in Objects

- Instantiating a new object with a keyword: *new*

  var myArray = new Array();

- String -- stores a series of characters

  var carname = "Volvo XC60";

  str.length, str.toUpperCase(), str.substring()

- Date -- used to work with dates and times

  var d = new Date();

  d.getDate(), d.setYear(2023), d.toString()

- Math – allows you to perform mathematical tasks

  Math.abs(x), Math.floor(y), Math.random()

Built-in Objects

- Array -- used to store multiple values in a single variable
  - Defining an array

```
var myUnits=new Array();
var myUnits3=new Array("WAD", "Ajax", "XML");
```

  - Adding values

```
myUnits[0] = "WAA";   myUnits[1] = "WS";
```

- Properties and methods

```
myUnits.length; myUnits.reverse(); myUnits.sort(); myUnits.push();
myUnits.pop();
```

Browser and Document Objects

- When we run JavaScript, Browser object and document object are automatically available, and can be used in scripts

- Using JavaScript commands, we can manipulate these objects, and consequently update the interface that is presented to the user

Browser Objects

- A collection of objects, that interact with the browser window.
  - **window:** top level object in the BOM hierarchy
  - **history:** keep track of every page the user visits
  - **location:** contains the URL of the page
  - **navigator:** contains information about the browser name and version
  - **screen:** provides information about display characteristics
  - **document:** belongs to both BOM and DOM

Window Object

- Example of window object properties and methods

Property	Description
closed	Returns whether or not a window has been closed
document	Returns the Document object for the window
history	Returns the History object for the window
Method	**Description**
alert()	Displays an alert box with a message and an OK button
open()	Opens a new browser window
print()	Prints the contents of the current window

More on: http://www.w3schools.com/jsref/obj_window.asp

History and Location Objects

- History

Property/method	Description
length	Return the number of elements in the history list
back()	loads the previous URL in the history list
forward()	loads the next URL in the history list

- Location

Property/method	Description
href	sets or returns the entire URL
replace(url)	replaces the current document with a new one
reload()	reloads the current document

More on: http://www.w3schools.com/jsref/obj_history.asp

http://www.w3schools.com/js/js_window_location.asp

User-Defined Objects

- There are several ways to define objects in JavaScript
  - Using a “Object()” Constructor
  - Using an Object Literal
  - Using a “Constructor” function

User-Defined Objects --Object() Constructor

- We can also create a user-defined object with the Object() constructor

```
var member = new Object();
member.name = “Julia Ma”;
member.email = “jma1@unitec.ac.nz”;
member.isRegistered = true;
```

- we can define methods, first just defining a separate function

```
function showMe() {alert(“I’m here!”);} // at this stage, not yet a method
```

- and then assigning the function as a property of an object

```
member.present = showMe; // now assigned as a method of member
member.present = function() {alert(“I’m here!”);} // alternative approach
```

- All done, we now can call `member.present();`

User-Defined Objects -- Object Literal

- We can also create a new object by using an object literal, which can even include the code for a method

```
var member =
{ name: “Julia Ma”,
     email: “jma1@unitec.ac.nz”,
     isRegistered: true,
     present: function () {alert(“I’m here!”);}
};
```

User-Defined Objects -- “Constructor” function

- The standard way to create an "object type" is to use an object constructor function

```
function person(first, last, age, eye) {
        this.firstName = first;
        this.lastName = last;
        this.age = age;
        this.eyeColor = eye;
        this.sayPersonName = function(){alert("My name is "+
                                this.firstName );}
}
var myFather = new person("Mike", "Tutui", 50, "blue");
var myMother = new person("Sara", "Tutui", 48, "green");
```

JavaScript and Event Models

- Events are necessary to create interaction between JavaScript and HTML
- Events occur when either a user (eg clicks a button) or the browser does something (eg loads a page)
- We can bind an event to an event handler
- The handler is the function that is called automatically when the event occurs.

JavaScript Events

- onblur An element loses focus
- onchange The user changes the content of a field
- onclick Mouse clicks an object
- onload A page or an image is finished loading
- onmouseover The mouse is moved over an element
- onsubmit The submit button is clicked
- More on http://www.w3schools.com/jsref/dom_obj_event.asp

Event Registration

- **Inline event registration** by using HTML attributes

```
<a href="goThere.html" onclick="startNow()" >
```

- **Traditional event registration**

```
var myElement = document.getElementById('1stpara');
myElement.onclick = startNow;
```

to remove myElement.onclick = null;

Event Registration

- **IE event registration**

  myElement.attachEvent('onclick', startNow);

  to remove myElement.detachEvent('onclick', startNow);

- **W3C DOM event registration**

  var myElement = document.getElementById('1stpara');

  myElement.addEventListener('click', startNow, false);

  to remove myElement.removeEventListener('click', startNow, false);

Exercise

- JavaScript exercise
- JavaScript & CSS Menu Exercise

End of The Session 2

Week 4