Web Development

Week 9: Introduction to JavaScript

Notes By: Jonathan McCarthy

Overview

- History of JavaScript
- Characteristics of JavaScript
- JavaScript basics
- JavaScript DOM and the browser

What is a scripting language?

- **Developed for a particular purpose**
  - Uses the facilities of an existing system
  - Exposes the system's functionality to programmatic control
- **Generally easy to learn and use**
  - Used by professional and non-professional programmers
- **Ideal for small applications**
- **Usually interpreted rather than compiled**
- **Most often dynamically typed**
  - It is not necessary to declare the type of a variable before you use it

Server-side scripting

- Access to server-side environment
- Business logic
- Database access
- Access to server file-system
- Authentication / Authorization
- Process requests and construct appropriate responses
- Examples
  - Ruby, PHP, Perl, Server-Side JavaScript

Client Side Scripting

- Access to client-side environment – typically a browser
- Manipulate client-side environment
- Respond to client-side events
- Perform client-side validation of data
- Security restrictions in place
-
- Examples
  - JavaScript, Jscript, VBScript, Tcl

History of JavaScript

- JavaScript was created in May 1995 by Brendan Eich who was working at Netscape. Initially the project was called Mocha and then LiveScript. After a trademark licence was received from Sun the name JavaScript was adopted.
- In 1996 JavaScript was taken to the ECMA for standardisation. ECMAScript is the name of the official standard.
- ECMAScript continued to be developed and enhanced over a number of years. The current version is ECMA-262 edition 6.
- Other vendors can implement based on the standard specification.
- Dialects
  - JavaScript (Mozilla.org)
  - Jscript (Microsoft)
  - ActionScript (Adobe)

What is JavaScript used for?

- JavaScript is a scripting language used to manipulate the web browser into performing checks, actions and tasks as desired.
- JavaScript is a standard and will work across browsers and mobile devices that supports JavaScript.
-
- **Question**
  - **List 5 tasks JavaScript can be used for in a web browser (ie. on a web page!!)**

Java and JavaScript

- JavaScript is not a subset of Java
- Java is a compiled OOP language
- JavaScript is an OOP scripting language

Java	JavaScript
Statically typed language	Dynamically typed language
Classed based	Prototype based
Compiled	Interpreted
No Closures	Supports Closures
OOP class and object	Supports OOP, no classes

JavaScript Characteristics

- JavaScript is dynamically typed
- Data-types are bound to values not variables
-
- JavaScript is case-sensitive
-
- Comments
  - Single line comments start with //
  - Multi line comments start with /* and end with */

Variables in JavaScript

- JavaScript variables are used to store data.
- Eg: var name = “Johnny”;
- The variable is declared using the var keyword.

- If the var keyword is not used, the variable will have a global scope. (What are the implications of this??)
- A variable declared inside a function using the var keyword has **local scope**.

Data Types in JavaScript

- JavaScript is loosely typed. You do not have to declare the type of data that will be stored in a variable.

- Eg:
  - var houseNumber = 21; // houseNumber is a number
  - var houseNumber = “21”; // houseNumber is a string
  - var houseNumber = true; // houseNumber is a boolean

Data Types in JavaScript

- The latest ECMAScript standard defines seven data types:
- Six data types that are primitives (any value with a type is primative):
  - Boolean
  - Null
  - Undefined
  - Number
  - String
  - Symbol (new in ECMAScript 6)
  - and Object
- Source: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures

Data Type Conversion

- **JavaScript is loosely typed**
- we do not have to specify a data-type when we declare variables
- We can initialise a variable with one type of data and then use the same variable to hold another type of data. JavaScript is said to be dynamically typed
- var myVariable = “Hello”;
- myVariable = 23;

Data Type Conversion

- **Automatic Conversion**
- The concatenation operator ‘+’ will automatically convert numbers to strings
- E.g.,
- “U” + 2 // results in string “U2”
- To convert a number to a string we can use the following technique
- var myNumber = 234;
- myNumber = myNumber + “”;

Testing variable types

- **NaN – Not a Number**
- NaN will be returned if we try to convert a string value to a number but the string value cannot be converted to a number.
- We can use the isNaN function to check values

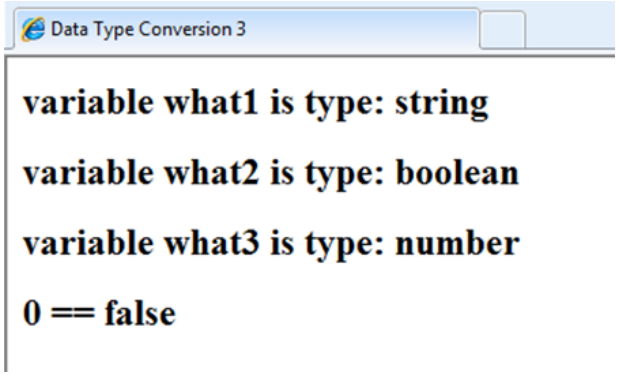
```
<html>
  <head>
    <title>Undefined Variable</title>
  </head>
  <body>
    <script language-"JavaScript" type="text/javascript">
      var num1 = parseFloat(prompt("Enter a number: "));
                    if (isNaN(num1)) {
        alert("Thats not a number");
      } else {
        alert("Thank you!!");
      }
    </script>
  </body>
</html>
```

Testing variable types

```
<html>
  <head>
    <title>Undefined Variable</title>
  </head>
  <body>
    <script language-"JavaScript" type="text/javascript">
      var what1 = "Hello!";
      var what2 = false;
      var what3 = 7756;
      document.write("<h2>variable what1 is type: " + (typeof what1) + "</h2>" );
      document.write("<h2>variable what2 is type: " + (typeof what2) + "</h2>" );
      document.write("<h2>variable what3 is type: " + (typeof what3 + "</h2>" );
      if ( 0 == false ) {
        document.write("<h2>0 == false</h2>");
      }

      if ( 0 === false ) {
        document.write("<h2>0 == false</h2>");
      }
    </script>
  </body>
</html>
```

- **typeof Operator**
- Used to determine a variable's data-type
- Syntax: typeof (operand) or (typeof operand)



Strings

- The JavaScript String object has many useful methods defined for it.
- These methods can be used with any JavaScript string type data

Example:

var message = “This is a message”;

message.toUpperCase();

This works by temporarily converting the string data to a String object. The toUpperCase() method is then called, the result stored back to message and the temporary String object is discarded.

Examples:

var strlen = “How long is this this sentence?”.length;

Arrays

- var students = ["John", "Billy", "Timmy"];
  - console.log(students[0]);

Using the new keyword:

- var animals = new Array("Cat", "Dog", "Parrot");
- console.log(animals[0]);

- Looping through an Array

```
for (i = 0; i < students.length; i++) {
   console.log(students[i]);
}
```

JavaScript Control Structures

Selection Structures

- if
- if .. else
- switch

Repetition Structures

- while
- do ….. while
- for
- for ….. in

Error Handling Structures

- try … catch

If Statement

```
if (grade >= 40)

{

    console.log("I Passed the exam!!");

}
```

If / Else Statement

```
if (grade >= 40)

{

    console.log("I Passed the exam!!");

} else {

    console.log("I Failed the exam!!");

}
```

Switch Statement

- The switch statement can be used to select one of many code blocks to be executed.

```
switch ( expression ) {
   case v:
     // do something here
      break;
   case v:
      // do something here
      break;
   default:
      // do something here if nothing matched
}
```

While Loop

- The while loop will stay in the loop if the evaluation condition is true.

```
while ( evaluation condition ) {
    // do something here
}
```

Example:

```
while ( i < 5 ) {
   console.log("The number is " + i);
   i++;
}
```

Do / While Loop

- The do/while loop is similar to the while loop. The main difference is that it will execute the code block once before evaluating the condition.

```
do {
   // do something here
} while (condition);
```

Example:

```
var i = 1;
do {
   console.log("The number is " + i);
   i++;
} while ( i < 5 );
```

For Loop

- For loops should be used when you know the exact amount of times a code block needs to be executed.

```
for (i = 0; i < 5; i++) {

    console.log(“Loop number: ” + i);

}
```

Try Catch Statement

- The try statement lets you test a block of code for errors.
- The catch statement lets you handle the error.
- Try – Catch – Throw - Finally

```
try {
   // do something
}
catch(err) {
  // do something if an error occurred in the code block above
}
```

Functions

- A JavaScript function is a block of code that performs a specific task. Once called the function will perform this task.

Example:

```
function areaSquare (side1, side2) {
    return side1 * side2;
}
```

Functions

- May pass optional arguments to a function
- We can access an arguments array that contains all the parameters passed to the function by the caller.

```
function product (arguments)
{
      var result = 0;
      for (var i = 0; i < arguments.length; ++i)
      {
          result += arguments[i];
      }
      return result;
}
```

In Class Example

HTML & JavaScript

- DOM – Document Object Model
  - Specifies all the parts of a web document that a browser can identify
  - We can access these components of the web document programmatically
  - Alter properties
  - Invoke methods
  - Add new elements
  - DOM is a hierarchical tree type structure
  - The web document becomes objects in the browser's memory when the document is loaded
  - The window object is the highest level object

DOM – Document Object Model

- The navigator object represents the browser
- The window is the top-level object which contains the document object
- The window also contains a location that specifies the location of the current document

The <script> Tag

- Is a HTML tag
- We need to specify the type
- There are other script languages such as VBScript and JScript
- If we want to reference an external script file we specify the src attribute
  - <script type="text/javascript" src="functions.js" />

- To use JavaScript in a HTML page we can
- Use a <script> tag in the head
- Use a <script> tag in the body
- Inline with HTML as an event handler
- In an external JavaScript file

The <script> Tag

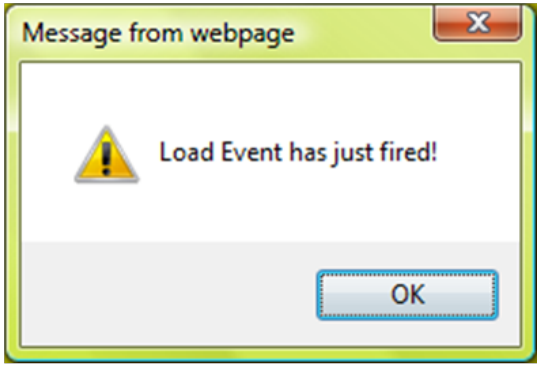
- **The <script> Tag in the head**
  - We can place any statements that need to be executed before the contents of the body load
  - We can also place user-defined functions in the head section
  - Global variables can also be placed in the head of the HTML document
- **The <script> Tag in the body**
  - We can place any statements that produce HTML content in the body

The <script> Tag

- The <script> Tag Inline with HTML as an Event Handle

We can write the event handler code inline with the relevant HTML tag.

Example:
<body
onLoad="alert('Load Event has just fired!')">



- The <script> Tag in an External JavaScript File
- Files have a .js extension
- Can put functions that will be used on many different pages in an external file
- Performance gains

References

- https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript
- http://www.ecma-international.org/publications/standards/Ecma-262.htm
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures

Additional Resources

- Books and Tutorials
  - http://tutorialzine.com/2015/05/15-awesome-and-free-javascript-books/
- Online learning
  - Codecademy: https://www.codecademy.com/en/tracks/javascript
  - W3Schools: http://www.w3schools.com/js/

Questions