Lecture 9
The Basics of JavaScript

Boriana Koleva
Room: C54
Email: bnk@cs.nott.ac.uk

Overview

- Overview of JavaScript
- Object Orientation
- Syntactic Characteristics
- Primitives, Operations and Expressions
- Math, Number, String and Date objects
- Screen Output
- Control Statements, Arrays and Functions

Origins of JavaScript

- Originally developed by Netscape, as LiveScript
- Became a joint venture of Netscape and Sun in 1995, renamed JavaScript
- Now standardized by the European Computer Manufacturers Association as ECMA-262
- An HTML-embedded scripting language
- We'll call collections of JavaScript code *scripts*, not programs

JavaScript and Java

- JavaScript and Java are only related through syntax
- JavaScript is dynamically typed
- JavaScript's support for objects is very different
- JavaScript is interpreted
  - Source code is embedded inside HTML doc, there is no compilation

Uses of JavaScript

- Transfer of some load from server to client
- User interactions through forms
  - Events easily detected with JavaScript
  - E.g. validate user input
- The Document Object Model makes it possible to create dynamic HTML documents with JavaScript

JavaScript Execution

- JavaScript scripts are executed entirely by the browser
- Once downloaded there is no exchange of information with the server
  - NB JavaScript programs can issue HTTP requests and load other pages
- JavaScript scripts do not require the Java VM to be loaded
- Thus JavaScript scripts tend to be fast

Object Orientation

- JavaScript is NOT an object-oriented programming language
  - Rather object-based
- Does not support class-based inheritance
  - Cannot support polymorphism
- JavaScript objects are collections of properties, which are like the members of classes in Java
  - Data and method properties
- JavaScript has primitives for simple types
- The root object in JavaScript is Object – all objects are derived from Object

Embedding in HTML docs

- Either directly, as in

```
<script type = "text/javascript">
    -- JavaScript script –
</script>
```

- Or indirectly, as a file specified in the `src` attribute of `<script>`, as in

```
<script type = "text/javascript"
    src = "myScript.js">
</script>
```

Syntactic Characteristics

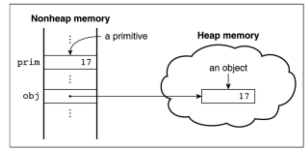
- Identifier form: begin with a letter or underscore, followed by any number of letters, underscores, and digits
  - Case sensitive
- 25 reserved words, plus future reserved words
- Comments: both // and /* … */
- Scripts are usually hidden from browsers that do not include JavaScript interpreters by putting them in special comments

```
<!--
-- JavaScript script –
//-->
```

- Semicolons can be a problem
  - They are "somewhat" optional
  - Problem: When the end of the line can be the end of a statement – JavaScript puts a semicolon there

Primitives

- All primitive values have one of the five types: Number, String, Boolean, Undefined, or Null
- Number, String, and Boolean have wrapper objects
- In the cases of Number and String, primitive values and objects are coerced back and forth so that primitive values can be treated essentially as if they were objects



Primitives

- All numeric values are stored in double-precision floating point
- String literals are delimited by either ' or "
- Boolean values are true and false
- The only Null value is `null`
- The only Undefined value is `undefined`

Declaring Variables

- JavaScript is dynamically typed – any variable can be used for anything (primitive value or reference to any object)
- The interpreter determines the type of a particular occurrence of a variable
- Variables can be either implicitly or explicitly declared

```
var sum = 0,
    today = "Monday",
    flag = false;
```

Numeric Operators

Operator	Associativity
++, --, unary –	Right (though it is irrelevant)
*, /, %	Left
Binary +, binary –	Left

The first operators listed have the highest precedence.

`Math` and `Number` Objects

- The `Math` Object provides `floor`, `round`, `max`, `min`, trig functions, etc.
  - e.g., `Math.cos(x)`
- The `Number` Object has some useful properties

Property	Meaning
`MAX_VALUE`	Largest representable number
`MIN_VALUE`	Smallest representable number
`NaN`	Not a number
`POSITIVE_INFINITY`	Special value to represent infinity
`NEGATIVE_INFINITY`	Special value to represent negative infinity
`PI`	The value of π

`String` Object

- The number of characters in a string is stored in the `length` property

```
var str = "George";
var len = str.length;
```

- Common methods:

Method	Parameters	Result
`charAt`	A number	The character in the `String` object that is at the specified position
`indexOf`	One-character string	The position in the `String` object of the parameter
`substring`	Two numbers	The substring of the `String` object from the first parameter position to the second
`toLowerCase`	None	Converts any uppercase letters in the string to lowercase
`toUpperCase`	None	Converts any lowercase letters in the string to uppercase

`Date` Object

- Create one with the Date constructor (no params)

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

- Local time methods of Date:
  - toLocaleString – returns a string of the date
  - getDate – returns the day of the month
  - getMonth – returns the month of the year (0 – 11)
  - getDay – returns the day of the week (0 – 6)
  - getFullYear – returns the year
  - getTime – returns the number of milliseconds since January 1, 1970
  - getHours – returns the hour (0 – 23)
  - getMinutes – returns the minutes (0 – 59)
  - getMilliseconds – returns the millisecond (0 – 999)

Screen Output

- JavaScript models the HTML document with the `Document` object
- The model for the browser display window is the `Window` object
  - The `Window` object has two properties, `document` and `window`, which refer to the `Document` and `Window` objects, respectively
- The `Document` object has a method, `write`, which dynamically creates content
  - The parameter is a string, often concatenated from parts, some of which are variables

```
document.write("Answer: ", result, "<br>");
```

  - The parameter is sent to the browser, so it can be anything that can appear in an HTML document (any HTML tags)

Screen Output

- The `Window` object has three methods for creating dialog boxes

1. **Alert**

```
alert("The sum is:" + sum + "\n");
```

   - Parameter is plain text, not HTML
   - Opens a dialog box which displays the parameter string and an OK button

Screen Output

2. Confirm

```
var question = confirm("Do you want
to continue this download?");
```

- Opens a dialog box and displays the parameter and two buttons, OK and Cancel
- Returns a Boolean value, depending on which button was pressed (it waits for one)

Screen Output

3. Prompt

```
prompt("What is your name?", " ");
```

- Opens a dialog box and displays its string parameter, along with a text box and two buttons, OK and Cancel
- The second parameter is for a default response if the user presses OK without typing a response in the text box (waits for OK)

http://www.cs.nott.ac.uk/~bnk/WPS/roots.html

Conditionals

- Selection statements – "`if`" and "`if…else`"

```
if (a > b)
  document.write("a is greater than b <br>");
else {
  a = b;
  document.write("a was not greater than b, now
          they are equal <br>");
}
```

- The `switch` statement

http://www.cs.nott.ac.uk/~bnk/WPS/borders.html

Loops

- `while` *(control_expression) statement or compound stmt*
- `for` *(init; control; increment) statement or cmpnd stmt*
  - init can have declarations, but the scope of such variables is the whole script
  - http://www.cs.nott.ac.uk/~bnk/WPS/date.html
- `do` *statement or compound* `while` *(control_expression)*

Arrays

- Array elements can be primitive values or references to other objects
- Array objects can be created in two ways, with `new`, or by assigning an array literal

```
var myList = new Array(24, "bread", true);
var myList2 = new Array(24);
var myList3 = [24, "bread", true];
```

- Length is **dynamic** - the `length` property stores the length
  - length property is writeable

```
myList.length = 150;
```

http://www.cs.nott.ac.uk/~bnk/WPS/insert_names.html

Functions

`function` function_name([formal_parameters]) {
 -- body –
}

- Return value is the parameter of `return`
  - If there is no return or if return has no parameter, undefined is returned
- We place all function definitions in the head of the HTML document
  - Calls to functions appear in the document body
- Variables explicitly declared in a function are local

Functions – parameters

- Parameters are passed by value, but when a reference variable is passed, the semantics are pass-by-reference
- There is no type checking of parameters, nor is the number of parameters checked
  - excess actual parameters are ignored, excess formal parameters are set to undefined
- All parameters are sent through a property array, `arguments`, which has the `length` property

http://www.cs.nott.ac.uk/~bnk/WPS/parameters.html

Summary

- Overview of JavaScript
- Object Orientation
- Syntactic Characteristics
- Primitives, Operations and Expressions
- Math, Number, String and Date objects
- Screen Output
- Control Statements, Arrays and Functions