Introduction to JavaScript

CORE JAVASCRIPT

POLITECNICO DI TORINO

Laura Farinetti - DAUIN

e-Lite

What and why JavaScript?

- JavaScript is a lightweight, interpreted programming language with object-oriented capabilities primarily used in web browsers for dynamic web pages and user interaction

  – JavaScript made its first appearance in Netscape 2.0 in 1995

  – Later standardized by ECMA (www.ecma.ch): ECMAScript

- JavaScript is one of the 3 languages all web developers must learn

  – HTML to define the content of web pages

  – CSS to specify the layout of web pages

  – JavaScript to program the behavior of web pages

What can JavaScript do for us?

- JavaScript can handle events (mouse click, page load, …)
- JavaScript can change

  esempi1-4.html

  - HTML content
  - HTML attributes
  - HTML styles (CSS)
- JavaScript can validate form data
- JavaScript can manage media and graphics
- JavaScript can work with HTML5 (HTML5 APIs)

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

Short history

- 1995
  - May: "Mocha" is invented in Netscape by Brendan Eich
  - September: renamed to LiveScript
  - December: renamed to Javascript (because Java was popular)
- 1996: JavaScript is taken to standardization in ECMA
  - From now on ECMAScript is the specification, Javascript is an implementation (ActionScript is another implementation)
- 1997: ECMA-262 (ECMAScript)
- 1998: ECMAScript 2
- 1999: ECMAScript 3

Short history

- 2005: Mozilla and Macromedia started work on ECMAScript 4 (feature rich and a very major leap from ECMAScript 3)
- Yahoo and Microsoft opposed the forming standard, and ECMAScript 3.1 was the compromise
- 2009: Opposing parties meet in Oslo and achieve an agreement, and ES3.1 is renamed to ES5
  - In the spirit of peace and agreement, the new Javascript long term agenda is named "Harmony"
- 2015: ES6 (part of the "Harmony" umbrella)
  - Starting with ES6 version names will be based on the year of release, so ES6 is ES2015 and ES7 should be ES2016

JavaScripts

- A JavaScript consists of JavaScript statements placed within the <script>... </script> HTML tags in a web page
- The <script> tag containing JavaScript code can be placed anywhere in a web page
  - In the head or the body section

prova.html

```
<html>
<body>
<script language="javascript" type="text/javascript">
<!--
    document.write("Hello World!")
//-->
</script>
</body>
</html>
```

Where to embed JavaScript code?

- In the head section
  - Scripts to be executed when they are called, or when an event is triggered, go in the head section
  - When you place a script in the head section, you will ensure that the script is loaded before anyone uses it

- In the body section
  - Scripts to be executed when the page loads go in the body section
  - When you place a script in the body section it generates the content of the page

What can JavaScript do?

- Generate dialog boxes
- Redirect a page
- Open new browser windows (pop-ups)
- Intercept mouse events
  - Clicks on links, buttons, ...
  - Mouse-overs, …
- Read user input in forms
- Modify HTML pages
  - Add/remove content
  - Change images
  - Modify form controls

What you need to know…

- JS variables and expressions
- JS language constructs (if, while, ...)
- JS objects
    - The most important built-in objects
- Interaction with the user
    - Mouse, keyboard
- Interaction with the browser
    - Windows, pages
- Interaction with the page: the Document Object Model (DOM)

Summary

- Core JavaScript
  - Lexical structure
  - Types, values, and variables
  - Expressions and operators
  - Statements
  - Objects
  - Arrays
  - Functions
  - Classes and modules
  - Pattern matching with regular expressions

Summary

- Client-Side JavaScript
  - JavaScript in web browsers
  - The window object
  - Scripting documents (DOM)
  - Scripting CSS
  - Handling events
  - Client-Side Storage
  - Scripted Media and Graphics
  - HTML5 APIs

JavaScript basics

- Syntax is similar to C language
- Case-sensitive language
- Uses the Unicode character set
- Ignores spaces and line breaks
- Semi-colons (at the end of a line) can be omitted
- Comments

```
// This is a single-line comment.
/* This is also a comment */ // and here is another comment.
/*
* This is yet another comment.
* It has multiple lines.
*/
```

Literals

- Data values that appear directly in a program
- Examples

```
12                // The number twelve
1.2               // The number one point two
"hello world"     // A string of text
'Hi'              // Another string
true              // A Boolean value
false             // The other Boolean value
/javascript/gi    // A "regular expression" literal
                     (for pattern matching)
null              // Absence of an object
```

Types, values and variables

- JavaScript types can be divided into two categories
  - Primitive types: numbers, strings, Booleans and the special JavaScript values "null" and "undefined"
  - Object types: any JavaScript value that is not a primitive type
- An object (i.e., a member of the type object) is a collection of properties where each property has a name and a value
- JavaScript defines two special kind of objects
  - an "array": an ordered collection of numbered values
  - a "function": an object that has executable code associated

Types, values and variables

- In JavaScript all variables must be declared before their use with the "var" keyword
- JavaScript variables are untyped
  - You can assign a value of any type to a variable, and you can later assign a value of a different type to the same variable
- JavaScript uses lexical scoping
  - Variables declared outside of a function are global variables and are visible everywhere in a JavaScript program
  - Variables declared inside a function have function scope and are visible only to code that appears inside that function

Numbers

- Unlike many languages, JavaScript does not make a distinction between integer values and floating-point values

- All numbers in JavaScript are represented as floating-point values
  - 64-bit floating-point format defined by the IEEE 754 standard

```
0
3
10000000
0xff                          // hexadecimal
0xCAFE911                     // hexadecimal
3.14
2345.789
.33333333333333333
6.02e23                       // 6.02 × 1023
1.4738223E-32                 // 1.4738223 × 10-32
```

Arithmetic in JavaScript

- Numeric operators: + - * / %
- Set of functions and constants defined as properties of the Math object

```
Math.pow(2,53)      // => 9007199254740992: 2 to the power 53
Math.round(.6)      // => 1.0: round to the nearest integer
Math.ceil(.6)       // => 1.0: round up to an integer
Math.floor(.6)      // => 0.0: round down to an integer
Math.abs(-5)        // => 5: absolute value
Math.max(x,y,z)     // Return the largest argument
Math.min(x,y,z)     // Return the smallest argument
Math.random()       // Pseudo-random number x where 0 <= x < 1.0
Math.PI             // π: circumference of a circle / diameter
Math.E              // e: The base of the natural logarithm
Math.sqrt(3)        // The square root of 3
Math.pow(3, 1/3)    // The cube root of 3
Math.sin(0)         // Trigonometry: also Math.cos, Math.atan, etc.
Math.log(10)        // Natural logarithm of 10
Math.log(100)/Math.LN10   // Base 10 logarithm of 100
Math.log(512)/Math.LN2    // Base 2 logarithm of 512
```

Text

- A string is an ordered sequence of 16-bit values, each of which typically represents a Unicode character
- The length of a string is the number of 16-bit values it contains
- JavaScript's strings (and arrays) use zero-based indexing: the first 16-bit value is at position 0
- The empty string is the string of length 0
- JavaScript does not have a special type that represents a single element of a string (character)
  - To represent a single 16-bit value, simply use a string that has a length of 1

String literals

- Examples

```
""                    // The empty string: it has zero characters
'testing'
"3.14"
'name="myform"'
"Wouldn't you prefer O'Reilly's book?"
"This string\nhas two lines"
"π is the ratio of a circle's circumference to its diameter"
'You\'re right, it can\'t be a quote' // escape sequence
```

- In client-side JavaScript programming, JavaScript code may contain strings of HTML code, and HTML code may contain strings of JavaScript code

```
<button onclick="alert('Thank you')">Click Me</button>
```

String operators, properties and methods

- Concatenation

```
msg = "Hello, " + "world";    // Produces the string "Hello,
                                 world"
greeting = "Welcome to my blog," + " " + name;
```

- The only property is
  - .length (the number of characters in the string)
- Many general methods
  - .charAt(), .concat(), .indexOf(), .localeCompare(), .match(), .replace(), .search(), .slice(), .split(), .substr(), .substring(), .toLowerCase(), .toUpperCase(), .toString(), .valueOf(), …
- Many methods specific for writing HTML

String methods for HTML formatting

- Methods that returns a copy of the string wrapped inside the appropriate HTML tag
  - Warning: not standard methods, may not work as expected in all browsers

- List of main methods
  - .big(), .small(), .italic(), .bold(), .fixed(), .sub(), .sup()
  - .fontcolor(c), .fontsize(s)
  - .anchor("name"), .link("url")

```
var str = "Hello World!";
document.write(str);
document.write("<br />");
str = str.fontcolor("red");
document.write(str + "<br/>");
str = str.fontsize(7);
document.write(str);
```

Main Javascript operators

- Numeric operators
  + - * / %

- Increment operators
  ++ --

- Assignment operators
  = += -= *= /= %=

- String operator
  + (concatenation)

- Comparison operators
  == (same value)   === (same value and same type)
  != > < >= <=

- Boolean and Logic operators
  && (logical "and") || (logical "or")   ! (logical "not")

Statements

- Conditionals (e.g. if, switch)
  - Make the JavaScript interpreter execute or skip other statements depending on the value of an expression
- Loops (e.g. while, for)
  - Execute other statements repetitively
- Jumps (e.g. break, return, throw)
  - Cause the interpreter to jump to another part of the program

If statement

```
if (condition)
{
    ...code...
}
```

```
if (condition)
{
    ...code if true...
}
else
{
    ...code if false...
}
```

```
if (condition1)
{
    ...code if 1 true...
}
else if (condition2)
{
    ...code if 2 true...
}
else
{
    ...if both false...
}
```

Choice statement

```
switch(n)
{
  case 1:
    code block 1
    break

  case 2:
    code block 2
    break

  default:
    code to be executed if n is
    different from case 1 and 2
}
```

Loop statements

```
for ( var=startvalue; var<=endvalue; var=var+increment )
{
    code to be executed

}
```

```
while ( condition_is_true )
{
    code to be executed

}
```

```
do {
    code to be executed
} while ( condition_is_true )
```

Jump statements

```
while ( ... ) // or for
{
    code
    break
    code
}
```

```
while ( ... ) // or for
{
    code
    continue
    code

}
```

Objects

- An object is a composite value
  - It aggregates multiple values (primitive values or other objects) and allows to store and retrieve those values by name
  - Unordered collection of properties, each of which has a name and a value
- Property names are strings: objects map strings to values
  - Similar to fundamental data structure called "hash", "hashtable", "dictionary" or "associative array"
- However an object is more than a simple string-to-value map: it also inherits the properties of another object, known as its "prototype"
  - The methods of an object are typically inherited properties, and this "prototypal inheritance" is a key feature of JavaScript

Object example

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

- All cars have the same properties, but the property values differ from car to car
- All cars have the same methods, but the methods are performed at different times

Objects

- JavaScript objects are dynamic: properties can usually be added and deleted

- Any value in JavaScript that is not a string, a number, true, false, null, or undefined is an object

- The most common operations to do with objects are create them and set, query, delete, test, and enumerate their properties
  - ES2015 added several advanced operations on objects

Arrays

- An array is an ordered collection of values
  - Each value is called an element, and each element has a numeric position in the array, known as its index
- JavaScript arrays are untyped: an array element may be of any type, and different elements of the same array may be of different types
- Creating arrays
  - With array literals
  - With the Array() constructor
- Reading and Writing Array Elements
  - Access to an element of an array: [] operator

Examples

```
var empty = [];                     // An array with no elements
var primes = [2, 3, 5, 7, 11];      // An array with 5 numeric elements
var misc = [ 1.1, true, "a" ];      // 3 elements of various types

var base = 1024;                      // The values in an array literal need
var table = [base, base+1, base+2, base+3]; // not be constants

var b = [[1,{x:1, y:2}], [2, {x:3, y:4}]]; // can contain object literals
                                           // or other array literals

var count = [1,,3]; // An array with 3 elements, the middle one undefined.

var a = new Array();    // An array with no elements
var a = new Array(10);
var a = new Array(5, 4, 3, 2, 1, "testing, testing");

var a = ["world"];    // Start with a one-element array
var value = a[0];     // Read element 0
a[1] = 3.14;          // Write element 1
i = 2;
a[i] = 3;             // Write element 2
a[i + 1] = "hello";   // Write element 3
a[a[i]] = a[0];       // Read elements 0 and 2, write element 3
a.length              // => 4
```

Array methods

- See references
  - join()
  - reverse()
  - sort()
  - concat()
  - slice()
  - splice()
  - push() and pop()
  - unshift() and shift()
  - toString()
- Several more in ES2015
  - E.g., foreach()

Functions

- A function is a block of JavaScript code that is defined once but may be executed, or invoked, any number of times
- JavaScript functions are parameterized
  - A function definition may include a list of identifiers, known as parameters, that work as local variables for the body of the function
  - Function invocations provide values, or arguments, for the function's parameters
- Functions often use their argument values to compute a return value that becomes the value of the function-invocation expression
- In addition to the arguments, each invocation has another value—the invocation context—that is the value of the this keyword
- If a function is assigned to the property of an object, it is known as a method of that object

Defining functions

```javascript
// Print the name and value of each property of o. Return undefined.
function printprops(o) {
  for(var p in o)
  console.log(p + ": " + o[p] + "\n");
}

// Compute the distance between Cartesian points (x1,y1) and (x2,y2).
function distance(x1, y1, x2, y2) {
  var dx = x2 - x1;
  var dy = y2 - y1;
  return Math.sqrt(dx*dx + dy*dy);
}

// A recursive function (one that calls itself) that computes factorials
// Recall that x! is the product of x and all positive integers less than it.
function factorial(x) {
  if (x <= 1) return 1;
  return x * factorial(x-1);
}

// This function expression defines a function that squares its argument.
// Note that we assign it to a variable
var square = function(x) { return x*x; }

// Function expressions can also be used as arguments to other functions:
data.sort(function(a,b) { return a-b; });
```

Invoking functions

- JavaScript functions can be invoked in four ways
  - As functions

```
printprops({x:1});
var total = distance(0,0,2,1) + distance(2,1,3,5);
var probability = factorial(5)/factorial(13);
```

  - As methods

```
var calculator = { // An object literal
  operand1: 1,
  operand2: 1,
  add: function() {
  // Note the use of the this keyword to refer to this object.
  this.result = this.operand1 + this.operand2;
  }
};
calculator.add(); // A method invocation to compute 1+1.
calculator.result // => 2
```

  - As constructors
  - Indirectly through their call() and apply() methods

References

- D. Flanagan, "Javascript: the Definitive Guide. Sixth Edition", O'Reilly
- N. C. Zakas, "Professional JavaScript for Web Developers. Third Edition", John Wiley & Sons
- ECMAScript® 2015 Language Specification
  - http://www.ecma-international.org/ecma-262/6.0/
- Alex MacCaw, "Javascript Web Applications"
- Stoyan Stefanow, "Javascript patterns"
- References
  - Mozilla Developer Network http://developer.mozilla.org
  - JSbooks https://jsbooks.revolunet.com
  - https://developers.google.com/chrome-developer-tools/
  - [Slides embedded references]

License

- This work is licensed under the Creative Commons "Attribution-NonCommercial-ShareAlike Unported (CC BY-NC-SA 3,0)" License.
- You are free:
  - to Share - to copy, distribute and transmit the work
  - to Remix - to adapt the work
- Under the following conditions:
  - Attribution - You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
  - Noncommercial - You may not use this work for commercial purposes.
  - Share Alike - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.
- To view a copy of this license, visit http://creativecommons.org/license/by-nc-sa/3.0/