COMP519 Web Programming

Lecture 11: JavaScript (Part 2)

Handouts

Ullrich Hustadt

Department of Computer Science
School of Electrical Engineering, Electronics, and Computer Science
University of Liverpool

Contents

Types

Data Type / Datatype / Type

A set of computer represented values together with a set of operations that can be performed on those values

- JavaScript distinguished four main types:
  - `boolean` – booleans
  - `number` – integers and floating-point numbers
  - `string` – strings
  - `object` – objects (including functions and arrays)
- Every value is of a particular type

Booleans

- JavaScript has a boolean datatype
  with constants `true` and `false` (case sensitive)
- JavaScript offers the following boolean operators

  `&&` (conjunction) `||` (disjunction) `!` (negation)
- The truth tables for the boolean operators are as follows:

A	B	(A && B)
true	true	B (true)
true	false	B (false)
false	true	A (false)
false	false	A (false)

A	B	(A || B)
true	true	A (true)
true	false	A (true)
false	true	B (true)
false	false	B (false)

A	(! A)
true	false
false	true

Boolean Operators

- As in almost every programming language the operators

  `&&` (conjunction) `||` (disjunction) `!` (negation)

  are so-called short-circuit boolean operators:

  A boolean expression is evaluated (using inorder traversal) only as far as is necessary to determine the overall truth value of the expression

```
(true || (false && ! (! (true && (false || true)))))
```

```
(true && (false && ! (! (true && (false || true)))))
```

- This means that `&&` and `||` are not commutative, that is,
  `(A && B)` is not the same as `(B && A)`

  ⇝ often taken advantage of in programs

```
(denom != 0) && (num / denom > 10)
```

Integers and Floating-point Numbers

- The JavaScript datatype `number` covers both
  - integer numbers `0` `2012` `-40` `1263978`
  - floating-point numbers `1.25` `256.0` `-12e19` `2.4e-10`

  ⇝ all numbers are stored as 64-bit floating-point numbers
- There are also some pre-defined number constants including

`Math.PI`	(case sensitive)	3.141592653589793846
`NaN`	(case sensitive)	'not a number'
`Infinity`	(case sensitive)	'infinity'

Operators on Integers and Floating-point Numbers

- Arithmetic operators supported by JavaScript include

`+`, `-`, `*`	Addition, Subtraction, Multiplication
`/`	Division
`**`	Exponentiation
`++`	Increment $(+1)$
`--`	Decrement (-1)

- The `Math` object provides a wide range of additional mathematical functions

`Math.abs(`*number*`)`	absolute value
`Math.ceil(`*number*`)`	round fractions up
`Math.floor(`*number*`)`	round fractions down
`Math.round(`*number*`)`	round fractions
`Math.log(`*number*`)`	natural logarithm
`Math.random()`	random number between 0 and 1
`Math.sqrt(`*number*`)`	square root

Beware of Rounding

- Rounding is an arithmetic operation commonly included in programming languages, but with different implementations:

JavaScript		Java		PHP	
`Math.round(2.4)`	2	`Math.round(2.4)`	2	`round(2.4)`	2
`Math.round(2.5)`	3	`Math.round(2.5)`	3	`round(2.5)`	3
`Math.round(2.6)`	3	`Math.round(2.6)`	3	`round(2.6)`	3
`Math.round(-2.4)`	-2	`Math.round(-2.4)`	-2	`round(-2.4)`	-2
`Math.round(-2.5)`	-2	`Math.round(-2.5)`	-2	`round(-2.5)`	-3
`Math.round(-2.6)`	-3	`Math.round(-2.6)`	-3	`round(-2.6)`	-3

- You should also check what values are returned or what errors are caused by `log(0)`, `sqrt(-1)`, `1/0`, `0/0`
(we'll do that in another lecture)

Strings

- In JavaScript a string literal is a sequence of characters surrounded by single-quotes or double-quotes

```
"This is a string"          "true"
'This is also a string'     '519'
```

- The escape character \ can be used to include single quotes in single-quoted strings and double quotes in double-quoted strings:

```
'This isn\'t a "number"'
"'We won't sing \"God Save the Queen.\"'"
```

- The escape character \ also must be used to include \ in a string

```
"This is a backslash\\"     'This is a backslash\\'
```

- Additional escape characters are available, but do not make much sense in the context of HTML

\b (backspace)	\f (form feed)	\n (newline)
\r (carriage return)	\t (tab)	

Strings

- JavaScript uses + for string concatenation

```
"519" + '123' // returns "519123"
'the' + 'end' // returns "theend"
```

- JavaScript supports multi-line strings

```
"\
 Your name is " + name + "\
 and you are studying " + degree + "\
 at " + university\
"
```

String Operators

- There are a range of additional string operators, for example:

string.substr(*start*, [*length*])

Returns (up to) *length* characters of *string* beginning at *start*

```
"university".substr(3,2)      // returns "ve" (count starts at 0)
```

string.indexOf(*str*, [*start*])

Returns the index number at which *str* starts in *string* after *start*

```
"university".indexOf("i",3)   // returns 7 (count starts at 0)
```

string.match(*regexp*)

Returns an array of matching substrings for the regular expression *regexp* in *string*

```
"0ab1".match(/[^0-9]/)        // returns ["a"]
```

string.replace(*regexp*, *str*)

Replaces occurrences of *regexp* in *string* by *str*

```
"0ab1".replace(/[^0-9]/g,"c") // returns "0cc1"
```

Variables

- JavaScript allows values to be stored in variables
- A JavaScript identifier may consist of
  letters, digits, the $ symbol, and underscore,
  but cannot start with a digit
- JavaScript variable names are JavaScript identifiers
- JavaScript variable names are case sensitive

```
weightInKilos $heightInMetres
```

good choice of variable names

```
_62M   _M62   _m62
```

valid variable names,
all distinct, but poor choice

```
_      $
```

valid variable names,
worst possible choice

Variable Declarations

- Variables can be declared (within an execution context) using one of the following statements:

```
var variable1, variable2, ...
var variable1 = value1, variable2 = value2, ...
```

  - The second statement also initialises the variables
  - A declaration does not specify the type of a variable
- A variable can be initialised without an explicit declaration by assigning a value to it:

```
variable = value
```

- It is good practice to always declare variables

Variable Declarations

- In JavaScript, the use of the value of a variable that is neither declared nor initialised will result in a reference error and execution of the program stops
- A declared but uninitialised variable has the default value `undefined` and has type `undefined`

```
var myVar1
var myVar2 = 5
console.log('myVar2 = ',myVar2)
console.log('myVar1 = ',myVar1)
console.log('myVar3 = ',myVar3)
```

```
myVar1 = undefined
myVar2 = 5
ex.js:5
console.log('myVar3 = ',myVar3);
                        ^

ReferenceError: myVar3 is not defined
```

Variable Declarations

- All variable declarations within an execution context are processed first before any other code
  ⇝ this does not include their initialisation

```
console.log('myVar4 =',myVar4)
console.log('myVar5 =',myVar5)
console.log('myVar6 =',myVar6)
var myVar4
var myVar5 = 2
```

```
myVar4 = undefined
myVar5 = undefined
ex.js:3
console.log('myVar6 = ',myVar6);
                        ^

ReferenceError: myVar6 is not defined
```

Variable Declarations

- All variable declarations within an execution context are processed first before any other code
  ⇝ this does not include their initialisation
- The same is not true for variables that are only ever initalised but not declared

```
console.log('myVar7 =',myVar7)
myVar7 = 7
```

```
console.log('myVar7 = ',myVar7);
                        ^
ReferenceError: myVar7 is not defined
```

Variable Declarations

- It is legal (though not sensible) to declare the same variable twice in the same context

```
var myVar8 = 8
console.log('myVar8 =',myVar8)
var myVar8 = 'eight'
console.log('myVar8 =',myVar8)
```

```
myVar8 = 8
myVar8 = 'eight'
```

- Re-declaring a variable does not affect its value

```
var myVar9 = 9
console.log('myVar9 =',myVar9)
var myVar9
console.log('myVar9 =',myVar9)
```

```
myVar9 = 9
myVar9 = 9
```

Assignments

- JavaScript uses the equality sign = for assignments

```
student_id = 200846369;
```

  As in Java, this is an assignment expression

- The value of an assignment expression is the value assigned

```
b = (a = 0) + 1;  // a has value 0, b has value 1
```

- JavaScript supports most of the standard binary assignment operators:

Binary assignment	Equivalent assignment
var += *expr*	*var* = *var* + *expr*
var -= *expr*	*var* = *var* - *expr*
var *= *expr*	*var* = *var* * *expr*
var /= *expr*	*var* = *var* / *expr*
var %= *expr*	*var* = *var* % *expr*
var **= *expr*	*var* = *var* ** *expr* (not in MS IE)

Constants

- Some JavaScript dialects allow the definition of constants using

```
const variable1 = value1, variable2 = value2, ...
```

  - defines one or more constants
  - constants follow the same scope rules as variables

- Attempts to change the value of a constant should result in a type error

```
const columns = 10;
columns++;              // type error
```

```
columns++;
        ^

TypeError: Assignment to constant variable.
```

- However, this construct is not supported by MS Internet Explorer 6–10 and does not have the desired effect in Safari before version 5.1.7 nor Opera before version 12

Determining the Type of a Value

- `string typeof value`
  returns a string representation of the type of *value*

Boolean	"boolean"	Number	"number"
String	"string"	Object	"object"
undefined	"undefined"	null	"object"
NaN	"number"	Infinity	"number"

Future versions of JavaScript may have an option to change `typeof null` to "null" (as in PHP)

```
console.log("Type of 23.0: " + typeof(23.0))
Type of 23.0: number
console.log("Type of \"23\": " + typeof("23"))
Type of "23": string
var a
console.log("Type of a:    " + typeof(a))
Type of a:    undefined
```

Revision and Further Reading

- Read
  - Chapter 14: Exploring JavaScript

  of R. Nixon: Learning PHP, MySQL & JavaScript: with jQuery, CSS & HTML5. O'Reilly, 2018.
- Read
  - Chapter 3: Language Basics: Variables, Data Types, Operators (except Relational and Equality)
  - Chapter 4: Variables, Scope and Memory

  of N. C. Zakas: Professional JavaScript for Web developers. Wrox Press, 2009.
  Harold Cohen Library 518.59.Z21 or
  E-book `http://library.liv.ac.uk/record=b2238913`