COMP284 Scripting Languages
Lecture 14: JavaScript (Part 1)
Handouts

Department of Computer
School of Electrical Engineering, Electronics, an
University of Liverpo

Contents

1. JavaScript
   - Motivation
   - Ov
   - Exa
2. Types and Variables
   - Types
   - Variables
   - Typecasting
   - Comparisons

JavaScript: Motivation

- PHP and Perl both allow us to create dynamic web pages
- In web applications, PHP and Perl code is executed on the web server (server-side scripting)
  - allows to use a website template that is instantiated using data stored in a dat
  - 'bus
    the c
    the u
  - not ideal for interactive web applications:
    too slow to react and too much data needs to be transfer
  - operations that refer to the location of the user/client
    for example, displaying the local time

```
echo date('H:i l, j F Y');
```

displays the local time on the server not the local time for the user

JavaScript

- JavaScript is a language for client-side scripting
  - script code is embedded in a web page (as for PHP),
    but delivered to the client as part of the web page and
    executed by the user's web browser
    ⇝ code is visible to the user/client
  - allo
    dat
  - a web
    the user may have disallowed the execution of JavaScript code
  - different JavaScript engines may lead to different re
    results not anticipated by the developer of JavaScri
  - performance relies on the efficiency of the JavaScrip
    the client's computing power (not the server's)
  - operations that refer to the location of the client are easy:

```
document.write("Local time: " + (new Date).toString());
```

JavaScript: History

- originally developed by Brendan Eich at Netscape under the name Mocha
- first shipped together with Netscape browser in September 1995 under the name LiveScript
- obtai
  Netsc
  in Dece
- does not have a particularly close relationship to Jav
  it mixes aspects of Java with aspects of PHP and Perl and its own peculiarities
- is a dialect of ECMAScript, a scripting language standardised in the ECMA-262 specification and ISO/IEC 16262 standard since June 1997
- other dialects include Microsoft's JScript and TypeScript and Adobe's ActionScript

Websites and Programming Languages

Website	Client-Side	Server-Side	Database
Google	JavaScript	C, C++, Go, Java, Python, PHP	BigTable, MariaDB
Facebo			, MySQL, sandra
YouT	———		, MariaDB
Yahoo	JavaScript	PHP	reSQL
Amazon	JavaScript	Java, C++, Perl	e
Wikipedia	JavaScript	PHP, Hack	DB
Twitter	JavaScript	C++, Java, Scala	MySQL
Bing	JavaScript	ASP.NET	MS SQL Server

Wikipedia Contributors: Programming languages used in most popular websites. Wikipedia, The Free Encyclopedia, 20 October 2017, at 11:28. `http://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites` [accessed 23 October 2017]

JavaScript: Hello World!

```
1  <html><head><title>Hello World</title></head>
2  <body>
3  <p>Our first JavaScript script</p>
4  <script type="text/javascript">
5    document.writeln("<p><b>Hello World!</b></p>")
6  </script>
7  <noscri
8  JavaScr
9  </nosc
10 </body>
```


- JavaScript code is enclosed between <s
- Alternative HTML markup that is to be used in case Ja enabled or supported by the web browser, can be spe <noscript> and </noscript>
- File must be stored in a directory accessible by the web server, for example `$HOME/public_html`, and be readable by the web server
- No particular file name extension is required

JavaScript scripts

- JavaScript scripts are embedded into HTML documents and are enclosed between `<script` and `</script>` tags
  - A JavaScript script consists of one or more statements and comments
    ⇝ there is no need for a main function (or classes)
  - Stat
    ⇝ s
  - Whi
    (Th
  - One-line comments start with // and run to t
  - Multi-line comments are enclosed in /* a
  - Comments should precede the code they are referrin

Types

- JavaScript is a loosely typed language — like PHP and Perl
- JavaScript distinguished five main types:
  - `boolean` – booleans
  - `number` – integers and floating-point numbers
  - `str`
  - `fun`
  - `obj`
- Integers, floating-point numbers, and strings do n
  from the corresponding Perl scalars, including th
  single-quoted versus double-quoted strings
- JavaScript distinguishes between these five types including between the three primitive types boolean, number and string

Variables

- JavaScript variable names do not start with a particular character
- A JavaScript variable name may consist of letters, digits, the $ symbol, and underscore, but cannot start with a digit
  ⇝ you can still stick to the PHP and Perl 'convention' that
  (s
- JavaS

Variables

- Variables can be declared using one of the following statements:

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


  - The second statement also initialises the variables
  - Use
    (onl
  - Use

- A variable can be inialised without a declaration
  by assigning a value to it:

```
variable = value
```


  - Both inside and outside a function definition,
    initialising an undeclared variable creates a global variable

- Note: A declaration does not specify the type of a variable
  only assigning a value of a certain type gives a variable a type

Variables

- In JavaScript, the use of the value of a variable that is neither declared nor initialised will result in a reference error and script execution stops
- A declared but uninitialised variable has the default value undefined and has no specific type
- JavaS
  requir
- The val

Type	Default	Type	Defaul		ult
bool	false	string	’unde		

```
myVar1++                    // reference error
var myVar2
myVar2++                    // myVar2 has value NaN
var myVar3
myVar3 = myVar3 + '!' // myVar3 has value 'undefined!'
```

Assignments

- JavaScript uses the equality sign = for assignments

```
student_id = 200846369;
```


As in PHP and Perl, this is an assignment expression

- The value of an assignment expression is the value assigned

```
b = (a = 0) + 1;
```


- JavaS

Binary assignment	Equ
var += *expr*	*var*
var -= *expr*	*var*
var *= *expr*	*var*
var /= *expr*	*var* = *var* / *expr*
var %= *expr*	*var* = *var* % *expr*

Note: **= is not supported

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

- Howe
and do
nor Opera before version 12

Values, Variables and Types

- `string typeof value`

  returns a string representation of the type of *value*

Boolean	"boolean"	Number	"number"
String	"string"	Object	"object"
und			
NaN			

Futur

`typeof null` to "null" (as in PHP)


```
document.writeln("Type of 23.0: " + typeof(23.0) + "<br />")
document.writeln("Type of \"23\": " + typeof("23") + "<br />")
var a
document.writeln("Type of a:    " + typeof(a) + "<br />")
```

```
Type of 23.0: number<br />
Type of "23": string<br />
Type of a:    undefined<br />
```

Typecasting

JavaScript provides several ways to explicitly type cast a value

- Apply an identity function of the target type to the value


```
"12" * 1  ⇝ 12          !!"1"  ⇝ true
12 + ""           "12"          !!"0"          true
fals                            false
[12,                            true
                                NaN
                  [12] * 1      12
```

Typecasting

JavaScript provides several ways to explicitly type cast a value

- Wrap a value of a primitive type into an object
  - JavaScript has objects Number, String, and Boolean with unary constructors/wrappers for values of primitive types (J

```
Numb                                          rue
Stri                                          rue
String(false)  ⇝  "false"  |  Number(true)      1
```

- Use parser functions parseInt or pars

```
parseInt("12")      ⇝ 12   |  par                    2.5
parseInt("2.5")     ⇝ 2    |  parseFloat("2.5e1")   ⇝ 25
parseInt("E52")     ⇝ NaN  |  parseFloat("E5.2")    ⇝ NaN
parseInt("␣42")     ⇝ 42   |  parseFloat("␣4.2")    ⇝ 4.2
parseInt("2014Mar") ⇝ 2014 |  parseFloat("4.2end") ⇝ 4.2
```

Comparison operators

JavaScript distinguishes between (loose) equality == and strict equality === in the same way as PHP:

expr1 == *expr2*	Equal	TRUE iff *expr1* is equal to *expr2* after type coercion
expr1 != *expr2*	Not equal	TRUE iff *expr1* is not equal to *expr2*

- When c
numb
- When comparing with a boolean, the boolean is con `true` and to 0 if `false`

- If an object is compared with a number or string, Java `valueOf` and `toString` methods of the objects to produce a primitive value for the object
- If two objects are compared, then the equality test is true only if both refer to the same object

Comparison operators

JavaScript distinguishes between (loose) equality == and strict equality === in the same way as PHP:

expr1 === *expr2*	Strictly equal	TRUE iff *expr1* is equal to *expr2*, and they are of the same type
expr1 !== *expr2*	Strictly not	TRUE iff *expr1* is not equal to *expr2*,


```
"123" == 123                    ⇝ true
"123" != 123                    ⇝ false
"1.23e2" == 123                 ⇝ true
"1.23e2" == "12.3e1"            ⇝ false
5 == true                       ⇝ false
```

```
1.2                             ⇝ true
"1.                             ⇝ false
5 === t                         ⇝ false
```

Comparison operators

JavaScript's comparison operators also applies type coercion to their operands and do so following the same rules as equality ==:

expr1 < *expr2*	Less than	true iff *expr1* is strictly less than *expr2* after type coercion
expr1 > *expr2*	Greater than	true iff *expr1* is strictly greater than *expr2*
expr1		*expr2*
expr1	or equal to	*expr2*


```
'35.5' > 55          ⇝ true
'ABD' > 'ABC'        ⇝ true
'1.23e2' > '12.3e1'  ⇝ false
"F1" < "G0"          ⇝ true
true > false         ⇝ true
5 > true             ⇝ true
```

```
'35                  ⇝ true
'AB                  true
'1.23e2' >= '12.3e1' ⇝ false
"F1" <= "G0"         ⇝ true
true >= false        ⇝ true
5 >= true            ⇝ true
```

Equality

Why do we care whether `5 == true` is true or false?

⇝ it influences how our scripts behave

⇝ it influences whether more complex objects are equal or not

PHP:

```
if (5) print("  i
else   print("5 is not t
print(" and ")
if (5 == true) prin
          else print("5 is not equal to true");
```


Output: `5 is true and 5 is equal to true`

JavaScript:

```
if (5) document.writeln("5 is true");
  else document.writeln("5 is not true")
document.writeln(" and ")
if (5 == true) document.writeln("5 is equal to true")
          else document.writeln("5 is not equal to true")
```

Output: `5 is true and 5 is not equal to true`

Equality

Why do we care whether `5 == true` is true or false?

↝ it influences how our scripts behave

↝ it influences whether more complex objects are equal or not

PHP:

```
$array3 = arra
$array4 = arra
if (($array3
      pr
else print("The two arrays are not equal");
```


Output: The two arrays are equal

JavaScript:

```
$array3 = ["1.23e2",5]
$array4 = ["12.3e1",true]
if (($array3[1] == $array4[1]) && ($array3[2] == $array4[2]))
      document.writeln("The two arrays are equal")
else document.writeln("The two arrays are not equal")
```

Output: The two arrays are not equal

Equality

Note: The way in which more complex data structures are compared also differs between PHP and JavaScript

PHP:

```
$array3 = array("1.23e2",5);
$array4 = arra
if ($array3 == $a
    pr
else print("
```


Output: The two arrays are equal

JavaScript:

```
$array3 = ["1.23e2",5]
$array5 = ["1.23e2",5]
if ($array3 == $array5)
    document.writeln("The two arrays are equal")
else document.writeln("The two arrays are not equal")
```

Output: The two arrays are not equal

Revision

Read

- Chapter 14: Exploring JavaScript

of

R. Nixon:

Learnin

O'Reilly, 2009.