COMP519 Web Programming

Lecture 10: JavaScript (Part 1)

Handouts

Ullrich Hustadt

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

Contents

JavaScript: Motivation

- HTML/CSS is sufficient for HTML documents aimed at disseminating information that does not change too frequently
- HTML documents that remain the same each time they are accessed are called static web pages
  $\rightsquigarrow$ HTML/CSS is sufficient for static web pages
- In order to develop interactive/reactive web pages we must integrate programming in some form or another
- One form is client-side programming/scripting:
  - Code is written in a separate programming/scripting language
  - Code is embedded in the HTML markup of a web page or linked to from the page and accessible by the web browser
  - Code is executed by the web browers (or external run-time environment) on the user's device

Scripting languages

Script

A user-readable and user-modifiable program that performs simple operations and controls the operation of other programs

Scripting language

A programming language for writing scripts

Scripting languages: Properties

- Program code is present at run time and starting point of execution
  - compilation by programmer/user is not needed
  - compilation to bytecode or other low-level representations may be performed 'behind the scenes' as an optimisation
- Presence of a suitable runtime environment is required for the execution of scripts
  - includes an interpreter, or just-in-time compiler, or bytecode compiler plus virtual machine
  - typically also includes a large collection of libraries
- Executation of scripts is typically slower then the execution of code that has been fully pre-compiled to machine code

Scripting languages: Properties

- Variables, functions, and methods
  typically do not require type declarations
  (automatic conversion between types, e.g. strings and numbers)
- Some built-in data structures (more than in C, fewer than in Java)
- Ability to generate, load, and interpret source code at run time
  through an eval function

```
var x   = 2;
var y   = 6;
var str = "if (x > 0) { y / x } else { -1 }";
console.log(eval(str));  // Output: 3
x       = 0;
console.log(eval(str));  // Output: -1
```

Scripting languages: Properties

- The evolution of a scripting language typically starts with a limited set of language constructs for a specific purpose

  Example: PHP started as set of simple 'functions' for tracking visits to a web page

- The language then accumulates more and more language constructs as it is used for a wider range of purposes
- These additional language constructs may or may not fit well together with the original core and/or may duplicate existing language constructs
- During this evolution of the language, backward compatibility may or may not be preserved

$\rightsquigarrow$ Language design of scripting languages is often sub-optimal

JavaScript

- JavaScript is a language for client-side scripting
  - script code is embedded in a web page and
    delivered to the client as part of the web page and
    executed by the user's web browser
    $\rightsquigarrow$ code is visible to the user/client
  - allows for better interactivity compared to server-side scripting as reaction time is improved and data exchange with the server can be minimised
  - a web browser may not support JavaScript or
    the user may have disallowed the execution of JavaScript code
  - different JavaScript engines may lead to different results, in particular, results not anticipated by the developer of JavaScript code
  - performance relies on the efficiency of the JavaScript engine and the client's computing power

JavaScript

- JavaScript is a language for client-side scripting
  - operations that refer to the location of the client are easy:

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

  - for security reason, scripts are limited in what they can do,
    e.g. scripts cannot access the client's filestore

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
- obtained its current name in December 1995 under a deal between
  Netscape and Sun Microsystems, the company behind Java,
  in December 1995
- does not have a particularly close relationship to Java,
  it mixes aspects of Java with aspects of other languages 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 Adobe's ActionScript

JavaScript: Use

Website	Client-Side	Server-Side	Database
Google	JavaScript, TypeScript	C, C++, Go, Java, Python, ~~PHP~~	BigTable, MariaDB
Facebook	JavaScript	Hack, PHP, Python, C++, Java, ...	MariaDB, MySQL, HBase, Cassandra
YouTube	~~Flash~~, JavaScript	C, C++, Python, Java, Go	BigTable, MariaDB
Yahoo	JavaScript	PHP	MySQL, PostgreSQL
Amazon	JavaScript	Java, C++, Perl	Oracle Database
Wikipedia	JavaScript	PHP, Hack	MySQL, MariaDB
Twitter	JavaScript	C++, Java, Scala	MySQL
Bing	JavaScript	C++, C#	MS SQL Server

Wikipedia Contributors: Programming languages used in most popular websites. Wikipedia, The Free Encyclopedia, 13 September 2019, at 01:04. `http://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites` [accessed 16 September 2019]

JavaScript: Hello World!

```
1  <!DOCTYPE html>
2  <html><head><title>Hello World</title></head>
3  <body>
4  <p>Our first JavaScript script</p>
5  <script>
6    document.writeln("<p><b>Hello World!</b></p>")
7  </script>
8  <noscript>
9  JavaScript not supported or disabled
10 </noscript>
11 </body></html>
```

- JavaScript code is enclosed in a `script`-element
- Alternative HTML markup that is to be used in case JavaScript is not enabled or supported by the web browser, can be specified in a `noscript`-element
- File must be stored in a directory accessible by the web server, for example `$HOME/public_html`, and be readable by the web server

JavaScript scripts

- JavaScript scripts are embedded into HTML documents and are enclosed in a `script`-element
- A JavaScript script consists of one or more statements and comments
  ⇝ there is no need for a main function (or classes)
  - Statements do not have to end in a semicolon but they can
    ⇝ stick to one convention in your code
  - Whitespace before and in-between statements is irrelevant
    (This does not mean it is irrelevant to someone reading your code)
  - One-line comments start with `//` and run to the end of the line

```
// This is a single-line comment.
```

  - Multi-line comments are enclosed in `/*` and `*/`

```
/* This is a multi-line comment. The syntax could also
   be used for a comment that is only one line long. */
```

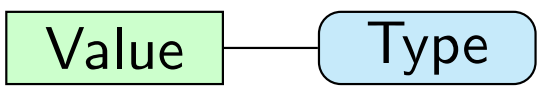
  - Comments should precede the code they are referring to

JavaScript: Type System

- JavaScript is a dynamically and loosely typed language
- All programming language provide operations that can be performed
- Most programming languages provide
  - values (as separate from operations)
  - variables to store values in
  - user-defined operations (functions)
  - parameters as special kind of variables used in the definition of an operation to refer to an argument of the operation
- Most programming languages categorise values and operations according to type

Data Type / Datatype / Type

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



Type Systems and Type Checking

Type System

A set of rules that determines the type of a value, variable, operation, expression possibly taking into account type declarations for variables and operations

Type Error

A failure of a type system to determine the type of a value, variable, operation, expression

Type Checking

The process of checking that a 'program' complies with the type system of its language

Type Systems and Type Checking

- JavaScript is a dynamically and loosely typed language

Statically typed programming language

Type checking is performed before program execution, e.g., by a compiler

Dynamically typed programming language

Type checking is performed during program execution when statements are executed / expressions are evaluated

Strongly typed programming language

Each value has exactly one type and operations can only be applied to argument values of the correct type

Loosely/Weakly typed programming language

A value of one type can be treated as being of another type
(or is automatically converted to the type required by an operation)

Type Systems and Type Checking

Statically typed programming language

Type checking is performed before program execution, e.g., by a compiler

Example: Java is a statically typed programming language

```
import java.io.*;

public class TEE {
  public static void f() {
    System.out,println("Called f()");
    System.out.println( "0" / 1 );
  }
  public static void main(String[] args) {  }
}
```

```
javac TEE.java
TEE.java:6: error: bad operand types for binary operator '/'
    System.out.println( "0" / 1 );
                            ^
    first type:  String, second type: int
```

The compiler indicates a type error before program execution

Type Systems and Type Checking

Dynamically typed programming language

Type checking is performed during program execution when statements are executed / expressions are evaluated

Example: Python is a dynamically typed programming language

```
def f():
  print("Called f()");
  print( "0" / 1 );
```

```
python TEE.py
```

```
No output
No error messages
```

Expression "0" / 1, that contains a type error, is never executed, so no error is indicated

Type Systems and Type Checking

Dynamically typed programming language

Type checking is performed during program execution when statements are executed / expressions are evaluated

Example: Python is a dynamically typed programming language

```
def f():
  print("Called f()");
  print( "0" / 1 );

f();
```

```
python TEE.py
```

```
Called f()
Traceback (most recent call last):
  File "TEE.py", line 5, in <module>
    f();
  File "TEE.py", line 3, in f
    print("0" / 1);
TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

Now that Python attempts to evaluate `"0" / 1`, a type error is indicated

Type Systems and Type Checking

Strongly typed programming language

Each value has exactly one type and operations can only be applied to argument values of the correct type

Example: Java is a (fairly) strongly typed programming language

```
public class STE {
  public static void main(String[] args) {
    double x = 2.1 + (5 * Integer.parseInt("2"));
    double y = 2.1 + (5 * "2");
  }
}
```

```
javac STE.java
```

```
STE.java:4: error: bad operand types for binary operator '*'
    double y = 2.1 + (5 * "2");
                          ^
  first type:  int
  second type: String
```

Applying arithmetic multiplication to a string is a type error

Type Systems and Type Checking

Loosely/Weakly typed programming language

A value of one type can be treated as being of another type
(or is automatically converted to the type required by an operation)

Example: JavaScript is a loosely typed programming language

```
y = 2.1 + (5 * "2")
console.log(y)
```

```
node STE.js
```

```
12.1
```

Applying arithmetic multiplication to a string works

JavaScript: Type System

- There are types and each value is of a particular type (or none)

  `519` and `1.9e3` are of type `number` (and only of that type)
  `'519'` and `"1.9e3"` are of type `string` (and only of that type)

- But the type of a variable does not need to be declared

```
var x;      // declares the variable x
```

- The type of a variable depends on the value it currently stores qnd the type can change if it is assigned a value of a different type

```
x = 519;    // x is of type number
x = '519'; // x is of type string
```

- Function declarations do not specify the type of their parameters
- In function applications the types of arguments will be adjusted automatically (if possible)

JavaScript: Type System

- There are types and each value is of a particular type (or none)
- But the type of a variable does not need to be declared
- The type of a variable depends on the value it currently stores and the type can change if it is assigned a value of a different type
- Function declarations do not specify the type of their parameters

```
function add(x,y) { return x + y; }
```

- In function applications the types of arguments will be adjusted automatically (if possible)

```
add(519,    1.9e3)   // returns the number 2419 = 519+1900
add('519',"1.9e3")   // returns the string '5191.9e3'
add(519,   '1.9e3')  // returns the string '5191.9e3'
add(true,   1.9e3)   // returns the number 1901
```

⇝ makes programming easier

⇝ potentially leads to more bugs

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 1: What is JavaScript?
  - Chapter 2: JavaScript in HTML
  - Chapter 3: Language Basics: Syntax

  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`