PB138 - Modern Markup Languages

T. Pitner, L. Bártek, A. Rambousek, L. Grolig

Presentation Outline

- The standard **ECMAScript**
  - ECMAScript language syntax
  - ECMAScript implementations
- ECMAScript vs **JavaScript**
- Application Building and Deployment
  - NPM
- **TypeScript**
- Conventions, Development support.
- https://slides.com/lukasgrolig/pb138-lab-introduction-to-javascript

Standard ECMA-262

European Computer Manufacturers Association (ECMA) International

- In 1959 need of standardization of computer operation techniques (programming) and input and output codes .
- Founded in 1960 by Compagnie des Machines Bull (FR), IBM World Trade Europe Corporation (European subsidiary of the US Company) and International Computers and Tabulators Limited (UK)
- Task - coordination of the national standards.

ECMAScript (ECMA-262)

- Scripting language specification.
  - Standard 1st edition released on June 1997.
- Standardized by ECMA International.
- Created to standardized JavaScript to allow multiple independent implementations.
  - Most common implementation is JavaScript.
  - Other implementations:
    - JScript
    - ActionScript
    - ...

ECMAScript (ECMA-262)

- Commonly used for:
  - client-side scripting
  - Server-side using Node.js

ECMAScript - Features

- Imperative and structured C-like programming language.
- Weakly typed - type of some variables is assigned according performed operations.
  - This makes several problems in a variable type conversions criticized by developers.
- Transpiling
  - Source-to-source compilation
  - Newer versions of ECMAScript are transpiled into the version according user client.
- Interpreters uses JIT (many interpreters)
  - Just-in-Time compilation (interpret compiles into binary before interpretation starts)
  - Optimization for particular platform.

ECMAScript Runtimes

- Interpreters with JIT:
  - Chakra
    - MS IE, Microsoft Edge
  - Chakaran
    - Opera from 10.50 until version 15
  - SpiderMonkey
    - Mozilla Gecko applications (Firefox, Seamonkey, …), OptimTalk (VoiceXML platform), ...
  - JavaScriptCore
    - WebKit projects (Safari, ..)

ECMAScript Runtimes (cont.)

- Interpreters with JIT:
  - Tamarin
    - ActionScript interpreter (Adobe Flash)
  - V8
    - Google Chrome, Node.js, V8.net, …
  - Nashhorn
    - JDK since
  - ...

ECMAScript Runtimes

- Without JIT
  - Continuum - ECMAScript 2015 interpreter written in ECMAScript 3, can run in older browsers (IE6, …)
  - Opera Engines:
    - Linear B - Opera 7.0 - 9.50
    - Futhark - Opera 9.50 - 10.10
  - ...

JavaScript

JavaScript

- Designed by Netscape on 1995 by Netscape Corporation.
- 1996 submitted to ECMA International.
- 1996 Reverse engineered and adopted by Microsoft in Internet Explorer as JScript.
- 1997 published as ECMA Standard ECMAScript.
- During the IE dominance JScript was de-facto scripting language standard.

JavaScript (cont.)

- 2005 - J. J. Garrett published whitepaper on AJAX
  - Uses JavaScript
  - Allows webpage to download data on background without
- Ajax - started JavaScript renaissance.
- Current JavaScript ecosystem:
  - Many libraries and frameworks.
  - Increased usage out of the web browser (i.e. Node.js, ...)
  - ...

JavaScript to ECMAScript relation

- Both languages are closely related.
- ECMAScript standardized JavaScript syntax.
  - No additional libraries used in web browsers for example.
- JavaScript - one of the ECMAScript implementations.
  - Libraries
  - Frameworks
  - ...

JavaScript - Website client usage

- Used in 95% of websites.
- Scripts embedded in HTML documents.
  - May interact with HTML DOM.
  - AJAX communication.
  - ...
- All major web browsers contains JavaScript engines.

JavaScript - Pros

- Speed
  - Interpreted language - compilation is not needed
  - Client-side program run - no client to server communication needed.
- Simplicity
  - Easy to learn.
  - SImple structure for users and developers simplifies dynamic web development.
  - Saves money for dynamic web development.

JavaScript - Pros

- Popularity
  - Modern web browsers support.
  - Supported and used by famous companies
    - Google
    - Amazon
    - PayPal
    - ...

JavaScript - Pros

- Interoperability
  - Can be included in web page as well as inside the script of another programming language.
- Server-load
  - Many actions are performed on client-side (data input validation).
  - Browser does not need to reload entire web page but only the changed part.
- RIA (Rich Internet Application)
  - Drag & Drop components
  - Sliders
  - …
  - See HTML5 for more for example.

JavaScript - Pros

- Extended functionality
  - Adding code snippets using 3rd party add-ons (Mozilla GreaseMonkey, …) to developer code.
- Versatility
  - Capability of front-end (AngularJS, ReactJS) as well as back-end (Node.JS) development.
- Less overhead
  - Improves performance of web sites and web applications by reducing the code length.

JavaScript - Cons

- Client-side Security
  - Browser may download malicious javascript code or even malicious binary and run it.
- Browser support
  - Despite JavaScript standardisation different browsers may interpret JavaScript differently.
  - Older browser do not support new versions of JavaScript, etc.
- Lack of Debugging Facility
  - JavaScript debugging support in browsers is not efficient as debugging support in other programming languages.

JavaScript - Cons

- Single Inheritance
  - JavaScript does not support multiple inheritance - some application may need it.
- Sluggish Bitwise Function
  - Numbers stored as 64bit floating-point numbers vs 32bit integer operators
  - Need of multiple conversions of numbers during operation
    - Operands to 32bit integer
    - Result to 64bit floating-point.
- Rendering stop
  - A single error code can stop rendering of entire JavaScript code. This looks to user like there is no javascript at all.
  - Modern browsers are tolerant to these erorrs.

JavaScript - syntax

- JavaScript syntax is based on ECMAScript.
- The syntax is very close to syntax of languages like C or Java.
- Code example:

```
function validateforms() {
                    if(document.forms["first"]["text"].value==""){
                  document.forms["first"]["result"].value="Chybi vstup";
                  return false;
                } else {
                  document.forms["first"]["result"].value="ok";
                }
}
```

Node.js

Node.js

- Node.js is asynchronous event-driven JavaScript server-side runtime environment.
  - Built on the top of Google V8 runtime under supervision of Openjs foundation.
- Designed to build scalable network applications.
- Similar in design to Ruby's Event Machine and Python Twisted.
  - Moves the event model further
    - Event loop is runtime instead of library.
    - The event loop runtime is non-blocking.
- Designed without threads
  - Offers multiple cores that allows you run multiple processes simultaneously..

Node.js - Use-cases

- Unique I/O model allows building of I/O heavy and data-heavy application fast and easy:
  - streaming web applications,
  - real-time collaboration tools,
  - complex single page applications,
  - real-time chat applications,
  - microservices architectures,

Node.js - cons

- Node.js is NOT suitable:
  - You build CPU intensive application due to its single thread nature,
  - You build relational database application,
  - every time using a callback end up with tons of nested callbacks,
  - without diving in depth of JavaScript if someone starts Node, he may face conceptual problem,

Node.js Simple Server Example
(entity)

```
class Person{
  constructor(name, surname){ this._name=name;  this._surname=surname;   }
  set name(name){ this._name=name;  }

  set surname(sname){ this._surname=sname;  }

    get name(){ return this._name; }

    get surname(){ return this._surname; }
}
```

Node.js Simple Server Example
(params parsing)

```
var paramsToPerson = function(req){
    var q = url.parse(req.url).query;
    var name = qs.parse(q).name;
    var sname = qs.parse(q).surname;
    return new Person(name, sname);
 };
```

Node.js Simple Server Example (control)

```
http=require(HTTP);
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'application/json'});
  var person = paramsToPerson(req);
  res.end(JSON.stringify(person));
}).listen(8080);
```

Node.js Application Building and Deployment

Node.js building and deployment

- Every application must be:
  - Created
  - Developed
  - Tested
  - Deployed
- Needed automation - automation differs in different languages
  - C/C++ - cmake, make, imake, ...
  - Java - ant, maven, gradle, ...
  - Node.js - npm CLI, ...

NPM - Node Package Manager

- Node.js software registry.
- Consists from:
  - Website - allows to discover packages, setup profiles, etc.
  - Command line tools - allows npm interaction.
  - Node.js package registry - large public database of JavaScript software and modules.
- npm.js - company hosting and maintaining NPM.

NPM - CLI tool

- npm - commonly used commands:
  - init - initialises new package (creates basic package.json file),
  - build - tries to build Node.js package,
  - Install - installs package and all its dependencies, prepares it to run,
  - start - starts the package application,
  - rebuild - rebuilds the package,
  - publish - publishes package to npm registry,
  - For more commands see documentation at npmjs.com.

NPM Module Creation and Configuration

- Using command npm init
  - Recommended to fill questionnaire about the project - initialises some values in package.json file.
- Rest of configuration using the package.json file
  - Described on next slide

NPM package.json file content description

- package.json file - NPM module descriptor
- Format - JSON
- File structure:
  - Enclosed in curly braces ‘{‘ and ‘}’;
  - Every line contains pair attribute value terminated by comma (‘,’).
    - The attribute and value are separated by ‘:’.
  - Exact JSON description in some of the next lectures.

package.json Example

```
{
  "name": "test",
            module name
  "version": "1.0.0",
            module version
  "description": "Some project description",      module description
  "main": "index.js",
            module main source file
  "scripts": {

  },
```