

Web 2.0

Lecture 1: Introduction to JavaScript

doc. Ing. Tomáš Vitvar, Ph.D.

tomas@vitvar.com • @TomasVitvar • <http://vitvar.com>



Czech Technical University in Prague

Faculty of Information Technologies • Software and Web Engineering • <http://vitvar.com/courses/w20>



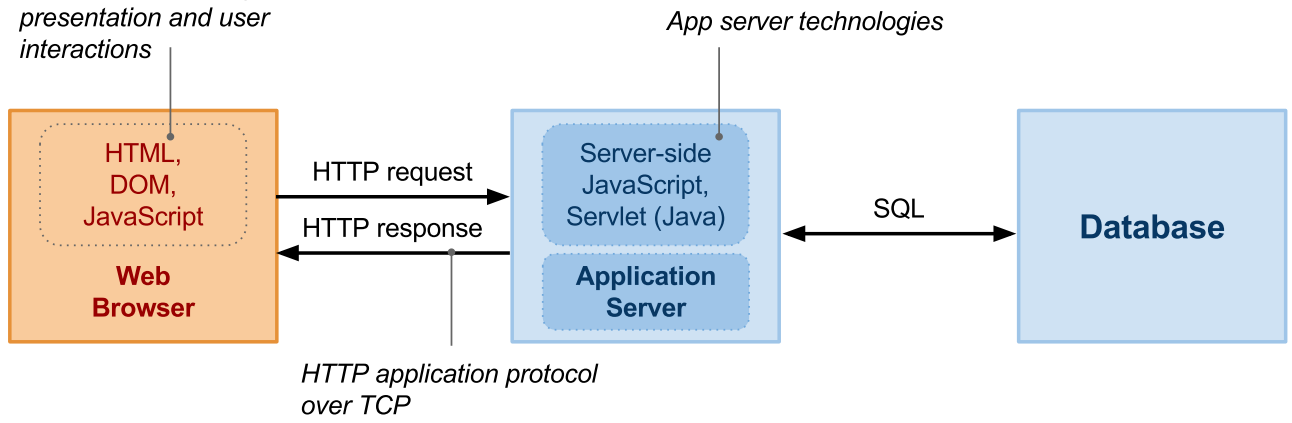
Evropský sociální fond
Praha & EU Investujeme do vaší budoucnosti

Modified: Thu Mar 16 2017
Humla v0.3

Web 2.0 Application Architecture

Web Application

*client-side technologies for
presentation and user
interactions*



Web 2.0 Application

JavaScript

- Lightweight, interpreted, object-oriented language
- Standard
 - *All major browsers support ECMAScript 6 and 7*
- Major characteristics
 - *First-class functions*
 - *functions as first-class citizens*
 - *language supports: passing functions as arguments to other functions, returning functions as values from other functions, assigning functions to variables or storing them in data structures.*
 - *Anonymous functions*
 - *declared without any named identifier to refer to it*
 - *Closures*

Lecture 1: Introduction to JavaScript, CTU Summer Semester 2016/2017, @TomasVitvar

Overview

- **JavaScript Basics**
- Server-side JavaScript

Objects and Arrays

- Objects and Arrays
- Functions

Lecture 1: Introduction to JavaScript, CTU Summer Semester 2016/2017, @TomasVitvar

Functions

- Function Callbacks
 - *You can use them to handle asynchronous events occurrences*
- Functions as values in object

Closures

- Closures
 - *A function value that references variables from outside its body*

Overview

- JavaScript Basics
- **Server-side JavaScript**

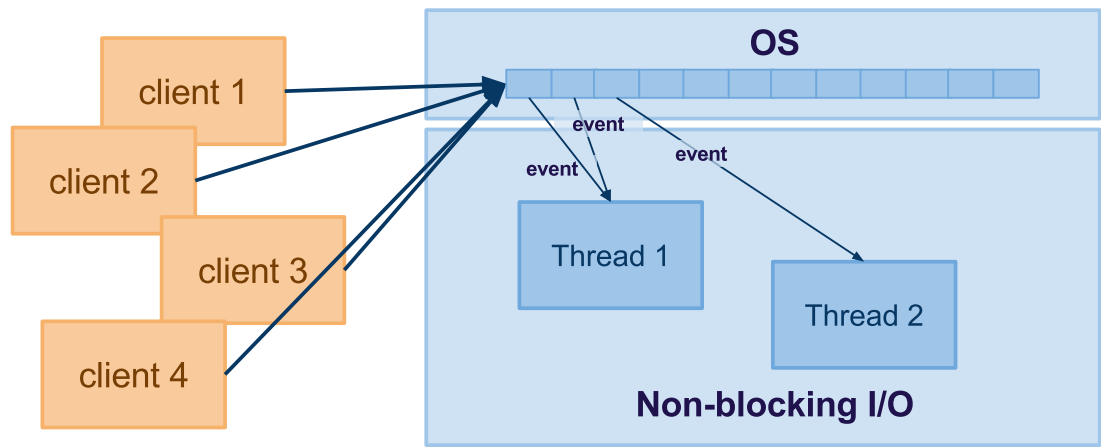
Recall: Application Server

- Environment that runs an application logic
 - *Client communicates with AS via an application protocol*
 - *Client – Browser, application protocol – HTTP*
- Terminology
 - *Application Server × Web Server × HTTP Server*
 - *AS is a modular environment; provides technology to realize enterprise systems*
 - *AS contains a Web server/HTTP server*
 - *We will deal with Web server only*
- Two major models to realize communication
 - *Blocking I/O (also called synchronous I/O)*
 - *Non-blocking I/O (also called asynchronous I/O)*
- A technology we will look at
 - *Node.js – runs server-side Javascript*

Lecture 1: Introduction to JavaScript, CTU Summer Semester 2016/2017, @TomasVitvar

Non-Blocking I/O Model

- Connections maintained by the OS, not the Web app
 - *The Web app registers events, OS triggers events when occur*



- Characteristics

Node.js

- Node.js [🔗](#)
 - *Web server technology, very efficient and fast!*
 - *Event-driven I/O framework, based on JavaScript V8 engine*
 - *Any I/O is non-blocking (it is asynchronous)*
 - *One worker thread to process requests*
 - *You do not need to deal with concurrency issues*
 - *More threads to realize I/O*
 - *Open sourced, @GitHub [🔗](#), many libraries [🔗](#)*
 - *Future platform for Web 2.0 apps*
- Every I/O as an event
 - *reading and writing from/to files*
 - *reading and writing from/to sockets*

HTTP Server in Node.js

- HTTP Server implementation
 - *server running at **138.232.189.127**, port **8080**.*
 - *Test it using Telnet*

Google Apps Script

- Google Apps Script
 - *JavaScript cloud scripting language*
 - *easy ways to automate tasks across Google products and third party services*
- You can
 - *Automate repetitive processes and workflows*
 - *Link Google products with third party services*
 - *Create custom spreadsheet functions*
 - *Build rich graphical user interfaces and menus*

Lecture 1: Introduction to JavaScript, CTU Summer Semester 2016/2017, @TomasVitvar

Rhino

- Rhino
 - *open-source implementation of JavaScript written entirely in Java*
 - *managed by the Mozilla Foundation*
 - *also provides another implementation of JavaScript engine written in C named SpiderMonkey*
 - *typically embedded into Java applications to provide scripting to end users*
 - *core language only and doesn't contain objects or methods for DOM, XMLHttpRequest, etc.*

manipulating HTML documents

- *enabling development of webapps with JavaScript in containers like JBoss Tomcat, and Google AppEngine*