pawel.rajba@gmail.com, http://itcourses.eu/

Node.js & Express.js

Agenda

- Wprowadzenie do Node.js
- Jak zacząć?
- Zdarzenia
- Podejście asynchroniczne
- Moduły
- Klient i serwer HTTP
- Express.js na przykładzie

Wprowadzenie

- Po raz pierwszy przedstawiony w 2009 przez Ryan Dahl (jsconf.eu conference)
  - Server side JS
  - Oparty o Google V8, zdarzenia
- Jeden z najpopularniejszych projektów na GitHub
- Główne składowe:
  - Libuv (międzyplatformowa biblioteka I/O)
  - Google's JS Engine V8
  - Kod w JavaScript i C++

Jak zacząć?

- Pobrać ze strony Node.js
  - Adres: https://nodejs.org/en/download/
  - Dostępne instalatory, binaria, kod źródłowy na …
  - … Linux, Mac, Windows, itd.
  - Wchodzimy i patrzymy co tam jest
- Użyć Node Version Manager (nvm)
  - https://github.com/creationix/nvm

Jak zacząć?

- Odpalamy node.exe lub node
  - Tryb interaktywny
  - Wpisujemy console.log("Hello World!");
- Tworzymy plik
  - example.js
  - node.exe example.js

Jak zacząć?

- Tworzymy pierwszy serwer WWW
  - https://nodejs.org/en/about/
  - Plik server.js

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');

console.log('Server running at http://127.0.0.1:1337/');
```

  - Odpalamy node server.js i przeglądarkę

Jak zacząć?

- https://c9.io/

Zdarzenia

- W pełni asynchroniczna obsługa zdarzeń
  - Przychodzące żądania
  - Obsługa bazy danych
  - setTimeout
  - … i wiele innych
- W klasycznych rozwiązaniach do obsługi wielu żądań zwykle uruchamianych jest wiele wątków

Podejście asynchroniczne

Synchronous

```
var fs = require('fs'),
    filenames,
    i,
    processId;

filenames = fs.readdirSync(".");
for (i = 0; i < filenames.length; i++) {
    console.log(filenames[i]);
}
console.log("Ready.");

processId = process.getuid();
```

Asynchronous

```
var fs = require('fs'),
    processId;

fs.readdir(".", function (err, filenames) {
    var i;
    for (i = 0; i < filenames.length; i++) {
        console.log(filenames[i]);
    }
    console.log("Ready.");
});

processId = process.getuid();
```

http://blog.shinetech.com/2011/08/26/asynchronous-code-design-with-node-js/

Podejście asynchroniczne

- Konwencja w node.js
  - getSth(inputParam, handleResults)
  - var handleResults = function(error, results) {
    // check errors
    // handle results
    }
- Oczywiście można używać funkcji anonimowych

Demo

- 01-intro:
  - example.js
  - server.js

Moduły

- Biblioteki z dodatkowymi funkcjonalnościami
- Można dołączać
  - var sm = require('someModule');
    var c = sm.count+1;
    var s = sm.size()-2;
  - var Car = require('Car');
    var car = new Car();
  - var sort = require('algorithms').quickSort;
    var tabSorted = sort(tab);
- Builtin modules: fs, http, crypto, os

Moduły

- Można udostępniać funkcje i zmienne w ramach projektu
- Aby udostępnić (somemodule.js):
  - module.exports.funkcja = jakasfunkcja;
  - module.exports.zmienna = jakaszmienna;
- Aby użyć w innym pliku:
  - var m = require('./somemodule');
  - m.funkcja();

Moduły

- 3rd party modules można instalować poprzez Node Package Manager (npm)
  - Instaluje się wraz node.js
  - Polecenie

    npm install *nazwa_modulu*
  - Dostęp do modułu jak poprzednio – przez require
- Przykład
  - npm install express
- Przegląd: http://npmjs.org/

Demo

- 02-modules

Zdarzenia

- Callbacks
  - Request/Reply
  - Wszystko albo błąd

```
getItems(pageNo, function(err, items)
{
  // sprawdzenie błędów
  // przetwarzania całej tablicy
});
```

- Events
  - Publish/Subscribe
  - Przetwarzanie wyników częściowych

```
var res = getItems(pageNo);

res.on('item', function(i) {
  // przetwarzenie elementu;
});

res.on('error', function(err) {
  // obsługa błędu
});
```

Zdarzenia

- EventEmitter
  - Klasa będąca bazą do publikacji zdarzeń
  - Po stronie publishera
    - emitter.emit(event [, args])
  - Po stronie subscribera
    - emitter.on(event, listener)
  - Kilka uwag:
    - „Event” to może być dowolny napis
    - Zdarzenie może mieć jeden lub wiele argumentów

Demo

- 03-events
  - events1
  - events2, repository

Klient HTTP

- Podstawowa konstrukcja

```
var http = require('http');


var req = http.request(options, function(response) {
  // handling response
});
```

- Można też użyć

  http.get() for GET requests

Serwer HTTP

- Podstawowa konstrukcja

```
var http = require('http');
var server = http.createServer(function(req, res) {
    // handle request
}).listen(port, host)
```

  - req – instancja server.ServerRequest
  - res – instancja server.ServerResponse

Demo

- 04-web

Express.js

- Sam Node.js oferuje dosyć podstawowe funkcjonalności
- Dlatego będziemy korzystać z ExpressJS
  - Framework do web MVC, web API
  - Ułatwia wiele interakcji
  - Dalej przyjrzymy się pod kątem budowania RESTful API
- Jest też kilka innych frameworków, np. MeteorJS, HapiJS

Express.js

- Jak zacząć?
  - npm init
    - Podajemy kilka szczegółów i tworzony jest plik package.json
  - npm install express --save
  - npm install body-parser --save
    - Instaluje moduł do konwersji treści żądania (POST) na format JSON

Demo

- 05-express-rest