Node.js

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMIWEBA Webové aplikace

Node.js

- Doposud jsme spouštěli JavaScriptový (JS) kód výhradně ve webovém prohlížeči, který poskytuje pro JS běhové prostředí pro.
- Open-source technologie node.js poskytuje (alternativní) běhové prostředí (založené na V8) pro JS kód. Ten je díky tomu možné spouštět i mimo webový prohlížeč.
- Pro úplnost dodejme, že prohlížeče mají různá běhová prostředí pro JS. V8 je prostředí z Google Chrome.
- Instalátor node.js je k dispozici na oficiálních stránkách projektu.

Hello world aplikace

- V následující části si vytvoříme jednoduchou aplikaci. Uvažme kód uložený v souboru Apj.js.

```
console.log("Hello world");
```

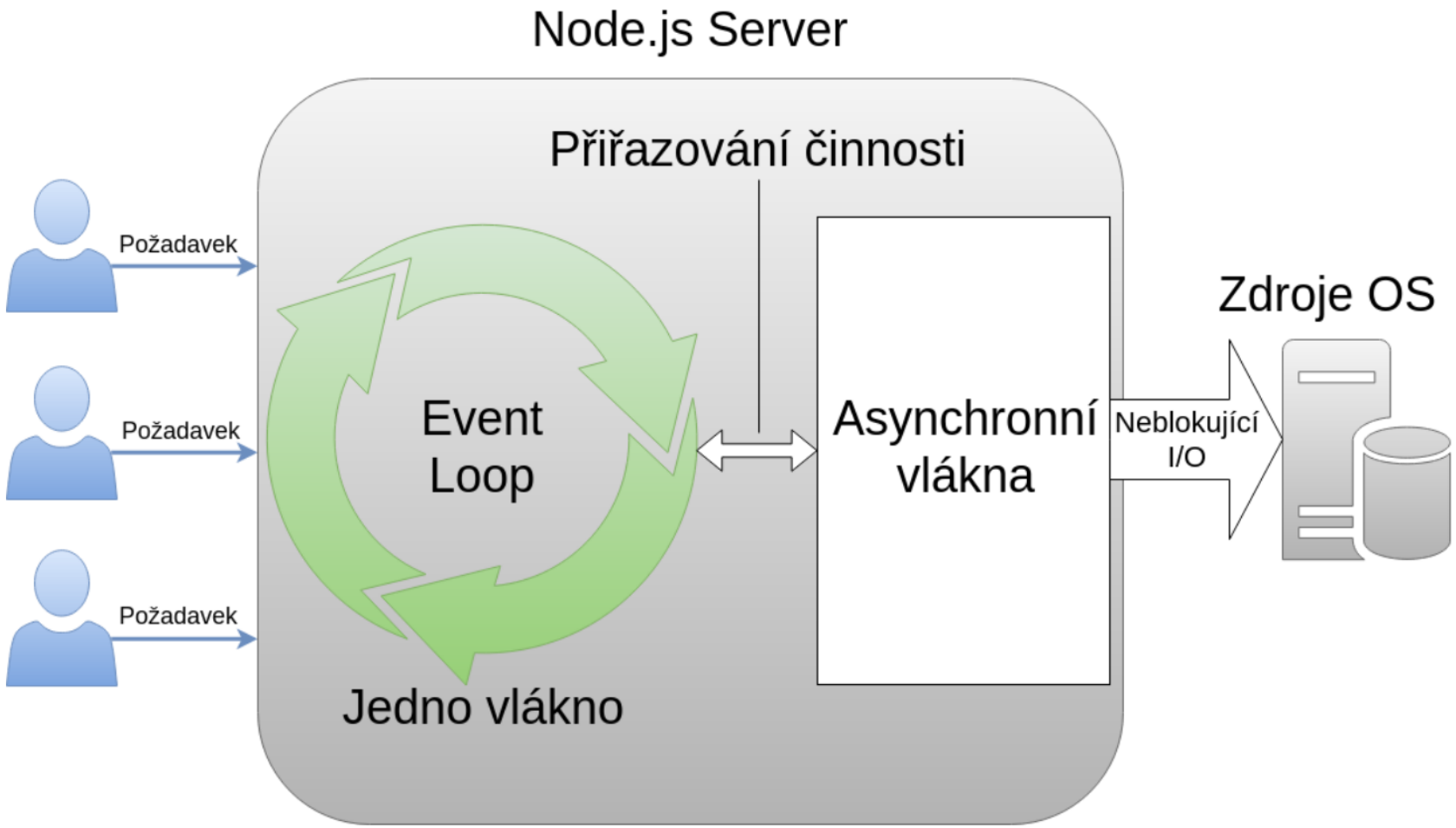
- Kód spustíme příkazem:

```
node node1.js
```

Základní vlastnosti

- Node.js umožňuje spouštět programy zapsané v JS kódů.
- Oproti klasickému vykonávání JS v prohlížeči, které jsme již několikrát viděli, je zde několik rozdílů:
  - Node.js JS kód neinterpretuje, ale provádí (JIT) kompilaci. Ne vše z ECMAScript je v node.js podporováno.
  - Node.js podporuje ES6 moduly a CommonJS moduly.
- Součástí node.js je rozsáhlé API, které značně usnadňuje vývoj aplikací. Kromě API je možné využívat i moduly z npm. Ty je třeba vždy doinstalovat pomocí nástroje npm.
- Dodejme, že ve výchozím stavu je node.js vždy spouštěn v development módu. Pro finální produkci je třeba tento mód vypnout. Podívejme se nyní na několik klíčových vlastností node.js.

Node.js server



Jedno vlákno

- Node.js běží v jednom neblokujícím vlákně a používá asynchronní událostní řízení.
- Tím je vynuceno adaptování funkcionálního paradigmatu.

```
// triviální webový server s počitadlem
const http = require('http');

var pocet = 0;

// na rozdíl od běžných webových serverů, zde budou jednotlivé spojení
inkrementovat proměnou pocet
http.createServer(function (req, res) {
  res.end(`${++pocet}`);
}).listen(3000);
```

Událostní řízení

- Typicky se kód dělí na funkce, které jsou mezi sebou provázány pomocí callback.
- Běžně se používají tři základní vzory:
  - sekvenční,
  - paralelní,
  - omezený paralelní (kombinace sekvenčního a paralelního).
- Zde je nutné si uvědomit, že funkce se běžně vykonávají asynchronně.
- Příklad níže ukazuje událostní řízení pomocí modulu events.

```
const EventEmitter = require('events');

const eventEmitter = new EventEmitter();

eventEmitter.on('start', number => {
  console.log(`started ${number}`);
});

eventEmitter.emit('start', 42);

// poznámka: off odstraní listener
```

Neblokující I/O

- Veškeré vstupně výstupní operace jsou neblokující.

```
const fs = require('fs');

fs.readFile('soubor.txt', function(err, data){
  console.log('A');
});

console.log('B');

// vypíše B a A
```

- Dodejme, že je možné provádět operace i synchronně.

Webový server

- Hlavní výhoda technologie node.js spočívá ve spojení webového a aplikačního serveru.
- Tím je možné dosáhnout velkého výkonu.
- Typicky se technologie node.js využívá pro tvorbu (škálovatelného) webového backendu, přičemž je možné snadno vytvořit monolitickou i mikro architekturu.
- Následující kód vytváří webový server a získává veškeré (relevantní) informace z HTTP dotazu.

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

// request a response jsou stream
const server = http.createServer((request, response) => {
  const { headers, method, url } = request;

  let body = [];

  request.on('error', (err) => {
    console.error(err);
  });

  request.on('data', (chunk) => {
    body.push(chunk);
  });
```

Webový server

```
  request.on('end', () => {
    body = Buffer.concat(body).toString();

    response.on('error', (err) => {
      console.error(err);
    });

    const responseBody = { headers, method, url, body };

    // HTTP status, hlavicka
    response.writeHead(200, {'Content-Type': 'application/json'})
    response.write(JSON.stringify(responseBody));
    response.end();
  });
});

server.listen(8080);
```

Webový server

- Tvorbu webového serveru usnadňuje modul express, který umožňuje snadnější zpracování HTTP požadavků a routování.
- Pro instalaci Expressu opět použijeme balíčkovací systém npm:

```
npm install express@~4.18.1
```

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const port = 3000;

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

// GET
app.get('/', (req, res) => {
  res.send(JSON.stringify(req.query));
});

// POST
app.post('/', (req, res) => {
  res.send(JSON.stringify(req.body));
});

app.listen(port, () => {
  console.log(`Listening on port ${port}`)
});
```

Webový server

- Komplexnější příklad využívající expres následuje.

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const port = 3000;


app.use(express.json());
app.use(express.urlencoded({ extended: true }));


// log
const log = function(req, res, next) {
  console.log(`${req.method} ${req.url}`);
  next(); // předání řízení
}
```

Webový server

```
// obsluha všech
const handler = function(req, res, next) {
  const { controler, method, params } = req.params;

  // vypsání POST
  console.log(JSON.stringify(req.body));

  res.send(`controler: ${controler}, method: ${method}, method: ${params}`);
  next();
}

app.all('/:controler/:method/:params(*)', [log, handler]);

app.listen(port, () => {
  console.log(`Listening on port ${port}`)
});
```

Napojení na databázi

- Příklad napojení na MySQL databázi následuje.
- Instalace:

```
npm install --save mysql2
```

```
var mysql = require('mysql2');

var connection = mysql.createConnection({
  host: 'localhost',
  port: '8889',
  user:'root',
  password:'root',
  database:'test'});

connection.connect();

var queryString = 'SELECT * FROM users';

connection.query(queryString, (err, rows, fields) => {
  if (err) throw err;
  for (var i in rows) {
    console.log(`ID: ${rows[i].id}, ${rows[i].name} ${rows[i].surname}`);
  }
});

connection.end();
// poznámka: modul mysql nepodporuje nový typ autentifikace MySQL databáze
```

Kam dál?

- Zejména v poslední době nabírá na popularitě projekt deno (přeházený název node).
- Autorem je Ryan Dahl, který vytvořil node.js.
- Hlavní myšlenkou deno je odstranit komplikace v architektuře node.