Node.js and npm

Authors: Julian Gimbel, Robert Pinsler

Node.js

- platform for building fast, scalable network applications
- published 2009 by Ryan Dahl
- built on Chrome's V8 JavaScript runtime
- applications are written in JavaScript
- **uses event-driven, non-blocking I/O model**

npm

- package manager of Node.js
- bundled with Node.js as of version 0.6.3
- important commands: npm install, npm publish

To publish a package with npm a package.json file is needed. “npm init” walks you through the most important settings of this file and http://browsenpm.org/package.json explains it even further.

***When* to use**	***When not* to use**
• lots of concurrent requests with few CPU cycles • JSON-based REST services • push-based data connections • streaming	• CPU-intensive tasks • few interaction with external resources • classical CRUD operations

Build a simple HTTP web server

1. Create helloworld.js:

```
1  var http = require('http');
2
3  http.createServer(function (req, res) {
4      res.writeHead(200, {'Content-Type': 'text/plain'});
5      res.end('Hello World\n');
6  }).listen(1337, '127.0.0.1');
7
8  console.log('Server running at http://127.0.0.1:1337/');
```

The web server at http://127.0.0.1:1337/ responds with *"Hello World"* for every request. The first line loads the *http* module into the variable *http*. Afterwards, the createServer function is called with an anonymous function as the callback parameter. The anonymous function is called at a later time when the callback event is fired.

2. Run helloworld.js:

```
> node helloworld.js
```

This line starts the node program and runs the helloworld.js file. Now curl http://127.0.0.1:1337 or use your browser to request the web site. That’s it, well done!

Build a simple TCP web server

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

var server = net.createServer(function (socket) {
    socket.write('Echo server\r\n');
    socket.pipe(socket);
});

server.listen(1337, '127.0.0.1');
```

The server at http://127.0.0.1:1337/ echoes whatever you send it. The basic principle is the same as shown above.

Asynchronously read a file

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

fs.readFile('index.html', function (err, data) {
    if (err) throw err;
    console.log(data);
});
```

The *fs* module allows reading files asynchronously. The callback function takes in two parameters: an error object and the data object with the file content. It is executed once the reading process is complete and either throws an error or writes the data object to the console. In the meantime, the node.js server can handle other events like new incoming requests thanks to non-blocking I/O .

Launch a preconfigured web server to run your own presentation

```
> npm install http-server -g
> cd to/presentation/folder
> http-server
```

First, install the *http-server* module using npm. Then *cd* to your presentation folder with your html file called *index.html*. Now simply start the http-server with *http-server* and enjoy your presentation at http://localhost:8080. Awesome, isn't it?

Congratulations, you know the basics to build even more awesome stuff! How about a REST web service, a streaming portal or a load balancer? Check out the following resources to learn more:

- http://nodejs.org/
- https://www.youtube.com/watch?v=ztspvPYybIY
- https://www.npmjs.org/
- http://www.nodebeginner.org/
- http://cre.fm/cre167-nodejs
- http://kunkle.org/nodejs-explained-pres/
- http://www.heise.de/developer/artikel/Zeitgemaesse-Webanwendungen-in-JavaScript-entwickeln-1731547.html
- http://t3n.de/magazin/serverseitige-javascript-entwicklung-nodejs-einsatz-231152/