Web 3

Node.js

HO
GENT

Node.js

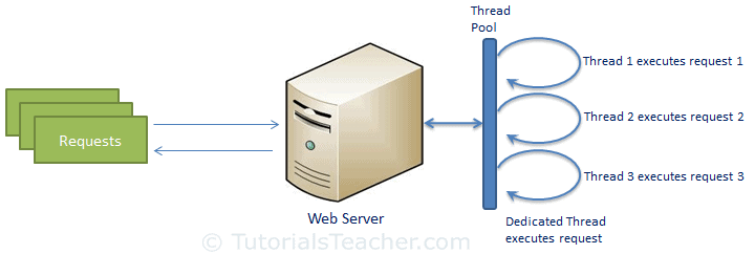
- Open source server runtime environment ontwikkeld met de Chrome’s V8 JavaScript engine.
- Server applicaties ontwikkeld met JavaScript
- Van nature asynchroon en cross platform
- Dient voor verschillende applicaties
  - Web applicatie
  - Real-time chat applicatie
  - **REST API server**
  - …

HO GENT

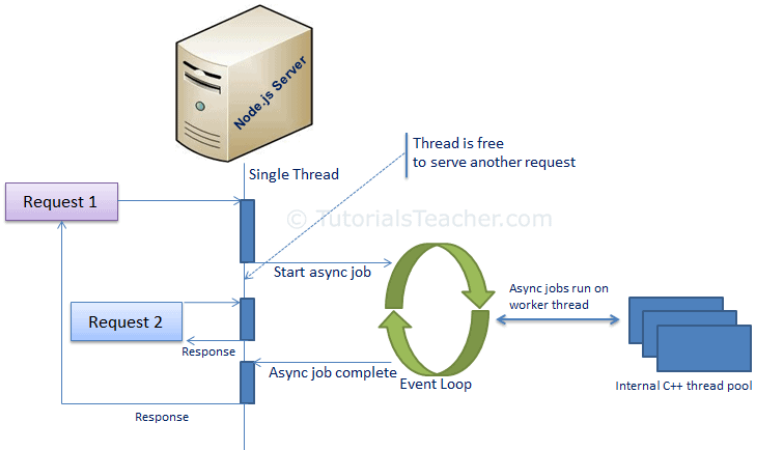
Node.js

Proces model

Traditioneel



Node



HO
GENT

Node.js

REPL

- **R**ead **E**val **P**rint **L**oop - Node.js console (REPL)
- Snelle en makkelijke manier om Node.js/JavaScript code te testen.
- Open de opdrachtprompt (Windows) of de terminal (Mac of UNIX/Linux).
- Geeft het volgende commando in

```
$ node
```

Node.js

REPL (2)

- Je kan nu bijna elke Node.js/JavaScript expressie testen in REPL
  - Optelling van 10 en 20:

```
$ 10 + 20                    //30
```

  - Samenvoegen van twee strings:

```
$ 'Hello' + 'World'          //Hello World
```

  - Definieren van variabelen en een operatie op uit voeren:

```
$ let x = 10, y = 20;
$ x + y                      //30
```

HO GENT

Node.js

REPL (3)

- Je kan dus elk statement van Node.js/JavaScript gaan uitvoeren.
- De REPL console sluit je terug af door twee maal op **`CTRL + C`** te drukken of door **`.exit`** te typen.

HO GENT

Node.js

REPL (4)

- Je kan dus elk statement gaan uitvoeren in de REPL console. Je kan ook nog een aantal commando's uitvoeren in de console:

REPL Commando	Beschrijving
`.help`	Toont de hulp van alle commando's
tab toets	Toont een lijst met alle beschikbare commando's
`.save filename`	Bewaart de huidige REPL sessie in een bestand
`.load filename`	Laadt het specifieke bestand in de huidige REPL sessie
ctrl + c	Beëindigt het huidige commando
ctrl + c (twee maal)	Beëindigt de REPL console
ctrl + d	Beëindigt de REPL console

HO GENT

Node.js

Basis

- Node.js werkt met JavaScript. Dus de syntax die we gebruikten in de browser gebruiken we ook met Node.
- Primitieve types in Node
  - String
  - Number
  - Boolean
  - Undefined
  - Null
  - RegExp
- Om een variabele, een object of een functie te definiëren is het dezelfde syntax

HO GENT

Node.js

Basis (2)

- Standaard lokale variabelen
  - In de JavaScript van de browser wordt elke variabele aangemaakt zonder het `var`, `let` of `const` keyword een globale variabele (hoisting principe). In Node is dit automatisch een lokale variabele.
- Globale scope
  - In de browser, is de globale scope het `window` object. In Node representeer het `global` object de globale scope.
  - Om iets toe te voegen aan de globale scope, moeten we een variabele, object, functie gaan exporteren. Door gebruik te maken van `exports` / `module.export`

HO GENT

Node.js

Modules

- Een module is een simpele of complexe functionaliteit die georganiseerd is in één of meerdere JavaScript bestanden en dat opnieuw gebruikt kan worden in de volledige Node applicatie.
- Elke module heeft zijn eigen context, dus het kan geen effect hebben op andere modules of de globale scope vervuilen. Iedere module kan in een aparte `.js` file gezet worden en in een aparte map.

HO GENT

Node.js

Modules (2)

- Node.js heeft drie types van modules
  - **Core modules**
    - Minimum functionaliteit dat aanwezig is in Node.js
  - **Local modules**
    - Lokaal gecreëerde modules in Node.js
  - **Third Party modules**
    - Modules die je kan gebruiken van andere partijen

HO GENT

Node.js

Modules (3)
Core modules

Core module	Beschrijving
http	http module bestaat uit klasses, methodes en events om een Node.js http server aan te maken
url	url module bestaat uit methode om een URL te bouwen en te verkrijgen
querystring	Module die bestaat uit methodes om, om te gaan met een query string
path	path module bestaat uit methodes om met paden te kunnen werken
fs	fs module bestaat uit klasses, methodes en events om in Node te werken met I/O
util	util module bestaat uit functies handig voor developers

Node.js

Modules (4)
Core modules

- Om Node.js core modules te kunnen gebruiken moeten we deze eerst importeren door gebruik te maken van `require()`

  `const module = require('module_name');`

- Vb.
  - ```
    const http = require('http');
    const server = http.createServer(function(req, res){
        // code
    });
    server.listen(5000);
    ```

HO GENT
    ```

Node.js

Modules (5)
Local modules

- Local modules zijn modules gecreëerd in uw Node.js applicatie door de developer zelf. Deze stellen verschillende functionaliteiten in onze app voor door middel van verschillende bestanden en mappen.
- In Node.js zetten we elke module in een apart JavaScript bestand.
- Voordeel van deze manier van werken is de herbruikbaarheid van een module.
- Bvb. Om te connecteren met een databank en het ophalen van data kan een module gebruikt worden waardoor je deze kan hergebruiken op verschillende plaatsen in de applicatie

HO
GENT

Node.js

Modules (6)
Local modules

- Voorbeeld van het aanmaken van een simpele logging module
  - ```
    const log = {
                        info: function(info) {
                                console.log('Info :' + info);
                        },
                        warning: function(warning) {
                                console.log('Warning: ' + warning);
                        },
                        error: function(error) {
                                console.log('Error: ' + error);
                        }
    };
    module.exports = log;
    ```

HO GENT
    ```

Node.js

Modules (7)
Local modules

- Voorbeeld van het laden van een simpele logging module
  - ```
    const myLogModule = require('./Log.js');
    myLogModule.info('Node.js started');
    ```

HO GENT
    ```

Node.js

Modules (8)
Export modules

- Verschillende types mogelijk om als een module te exporteren
  - Literals
  - Objects
  - Functions
- Het `module.exports` object is een speciaal object dat je in elke JavaScript file moet hebben in een Node.js applicatie. De `module` variabele stelt het de huidige module voor, en `exports` is een object dat meegegeven wordt aan deze module.

HO GENT

Node.js

Modules (9)
Export modules

- **Export Literals**
- Het exporteren van een **string literal** als een module.
  - `module.exports = 'Hello World';`

```
const msg = require('./Messages.js');
console.log(msg);
```

HO
GENT

Node.js

Modules (10)
Export modules

- **Export Object**
- Het exporteren van een **object** als een module.
  - ```
    exports.SimpleMessage = 'Hello World';
    // module.exports.SimpleMessage = 'Hello World';

    const msg = require('./Messages.js');
    console.log(msg.SimpleMessage);
    ```
  - ```
    module.exports = {
        firstName: 'James',
        lastName: 'Bond'
    };

    const person = require('./data.js');
    console.log(person.firstName + ' ' + person.lastName);
    ```

HO GENT

Node.js

Modules (11)
Export modules

- **Export Functions**
- Het exporteren van een **function** als een module.
  - ```
    module.exports = function(msg) {
        console.log(msg);
    };

    const msg = require('./Log.js');
    msg('Hello world');
    ```
  - ```
    module.exports = () => {
        console.log(msg);
    };
    ```

HO GENT

Node.js

Modules (12)
Load modules

- **Load module**
- Het laden van een lokale module gebeurt steeds met respect voor het pad waar de module instaat.
- De . aanduiding in het begin van de parameter van de require functie geeft de root aan van het project.
  - `const log = require('./utility/log.js');`

- Als je de bestandsnaam niet meegeeft zoals onderstaand voorbeeld zoekt Node.js ofwel naar een package.json bestand of naar een index.js bestand.
  - `const log = require('./utility');`

HO GENT

Node Package Manager (NPM)

- Command line tool om Node.js paketten te installeren, te updaten of te verwijderen. Online repository voor open-source packages die je kan gebruiken.
- Installeren kan op twee manieren:
  - Lokaal in het project
    - Een package wordt geïnstalleerd in de lokale projectfolder en is dus niet bruikbaar in andere projecten.
      `$ npm install <package name>` OF `$ yarn add <package name>`
  - Globaal
    - Een package wordt geinstalleerd in het systeem zodat alle Node.js applicaties op de computer gebruik kunnen maken van deze packages.
      `$ npm install -g <package name>` OF `$ yarn global add <package name>`

HO GENT

NPM

package.json

- Het package.json bestand houdt alle dependencies van het project bij. Dus als je het volgende commando uitvoert wordt er een dependency toegevoegd aan het project.
  `$ npm install <package name>` OF
  `$ yarn add <package name>`
- Een package.json bestand ziet er uit zoals dit.
- ```
  {
     "name": "NodejsConsoleApp",
     "version": "0.0.0",
     "description": "NodejsConsoleApp",
     "main": "app.js",
     "author": { "name": "Dev", "email": "" },
     "dependencies": { "express": "^4.13.3" }
  }
  ```
  ```

NPM

Updaten van een package

- Om een package te updaten naar de nieuwste versie maak je gebruik van het volgende commando

```
$ npm update <package name>      OF
$ yarn upgrade <package name>
```

HO GENT

NPM

Verwijderen van een package

- Om een package te verwijderen maak je gebruik van het volgende commando

```
$ npm uninstall <package name>      OF
$ yarn remove <package name>
```

https://docs.npmjs.com/

HO GENT

NPM

Initialiseren

- Node.js en NPM maken dus gebruik van het package.json bestand om dependencies voor het project bij te houden.
- Je hoeft dit bestand zelf niet aan te maken. NPM heeft hier ook een functie voor om dit te doen.
- Het volgende commando zal in de folder waar je momenteel zit een nieuw package.json bestand aanmaken

```
$ npm init
$ yarn init
```

HO GENT

Node.js

Web Server

- Node.js heeft een ingebouwde web server die het ons mogelijk maakt.
- Node.js maakt het ons dus mogelijk om alle http requests te verwerken van de web applicatie.
- Node.js doet dit op een asynchrone manier.
- Gebruik makend van de `http` core module van Node.js

HO GENT

Node.js

Web Server (2)

- Het aanmaken van een simpele Node.js webserver.
- Eerst het aanmaken van een server.js file

```
const http = require('http');    //1. Importeren van de http core module
const server = http.createServer((req, res) => {
   //2. Inkomende requests hier opvangen
});
server.listen(5000)              //3. Luisteren voor inkomende requests
console.log('Node.js web server at port 5000 is running..');
```

HO GENT

Node.js

Web Server (3)
HTML

```
const http = require('http');
const server = http.createServer((req, res) => {
   if(req.url == '/') {
        res.writeHead(200, { 'Content-Type': 'text/html'});
        res.write('<html><body><p>Home page.</p></body></html>');
        res.end();
   }
   else if (req.url == "/student") {
        res.writeHead(200, { 'Content-Type': 'text/html'});
        res.write('<html><body><p>Student page.</p></body></html>');
        res.end();
   }
   else
        res.end('Invalid request');
});
server.listen(5000)
console.log('Node.js web server at port 5000 is running..');
```

HO GENT

Node.js

Web Server (4)
JSON

```
const http = require('http');
const server = http.createServer((req, res) => {
   if (req.url == "/data") {
        res.writeHead(200, { 'Content-Type': 'application/json'});
        res.write(JSON.stringify({ message: "Hello World"}));
        res.end();
   }
});
server.listen(5000)
console.log('Node.js web server at port 5000 is running..');
```

HO GENT

Node.js

File System

- De fs module die ook standaard in Node.js ingebouwd zit maakt het mogelijk om bestanden op het systeem op te vragen. Verantwoordelijk voor alle synchrone en asynchrone bestands I/O operaties.
- Verschillende I/O operaties
  - Lezen van een bestand
  - Schrijven van een bestand
  - Openen van een bestand *
  - Verwijderen van een bestand

HO GENT

Node.js

File System
Lezen

- Het lezen van een bestand op een asynchrone manier.
- Gebruik makend van de volgende methode:
  - `fs.readFile(filename, [,options], callback)`
  - Parameters:
    - `filename:` Volledige pad en naam van het bestand
    - `options:` Optionele opties met de encoding en een flag
    - `callback:` Callback functie met twee paramters `err` en `fd`

  - ```
    fs.readFile('TestFile.txt', (err, data) => {
                    if(err) throw err;
       console.log(data);
    });
    ```

HO GENT
    ```

Node.js

File System
Schrijven

- Het schrijven van een bestand. Als het bestand al bestaat wordt het overschreven.
- Gebruik makend van de volgende methode:
  - `fs.writeFile(filename, data, [,options], callback)`
  - Parameters:
    - `filename`: Volledige pad en naam van het bestand
    - `data`: De data die in het bestand moeten geschreven worden
    - `options`: Optionele opties met de encoding en een flag
    - `callback`: Callback functie met twee paramters `err` en `fd`

  - ```
    fs.writeFile('test.txt', 'Hello World', (err) => {
                        if(err)
                             console.log(err);
                        else
                             console.log('Write operation complete.');
    });
    ```

HO GENT
    ```

Node.js

File System Verwijderen

- Het verwijderen van een bestand.
- Gebruik makend van de volgende methode:
  - `fs.unlink(path, callback)`
  - Parameters:
    - `path:` Volledige pad en naam van het bestand
    - `callback:` Callback functie met twee paramters `err` en `fd`

  - ```
    fs.unlink('test.txt', () => {
       console.log('delete operation completed.');
    });
    ```

https://nodejs.org/api/fs.html

HO GENT
    ```

Opdracht

- **Web server**
  - Maak een simpele webserver met Node.js, met een index.js bestand.
  - Maak het mogelijk om **drie** links op te vangen
    - Home page – Geeft een p element terug met hierin de tekst “Home Page”
    - “/profile” – Geeft een p element terug met hierin de tekst ”Profile Page”
    - “/data” – Geeft JSON terug van een student object dat je zelf maakt in JavaScript (voornaam, achternaam, leeftijd, woonplaats)
- **File System**
  - Maak een nieuw bestand file.js aan in hetzelfde project
  - De bedoeling is dat je een nieuw tekstbestand (.txt) aanmaakt met de writeFile methode. Waarin je jouw naam zet als data.
  - Lees hierna het bestand dat je aangemaakt hebt ui met de readFile methode.
  - Steek deze twee methodes in een functie die je dan gaat exporteren en importeren in het index.js bestand, om hier deze functie aan te roepen.

Opdracht (2)

- **NPM / Yarn**
  - Installeer lokaal met npm of yarn het ‘express’ package. Controleer of in het package.json bestand ‘express’ als een dependency staat.

HO GENT