

Odisee  
DE CO-HOGESCHOOL

# Referenties



## **The Complete JavaScript Course 2022: From Zero to Expert!**

Jonas Schmedtmann, Web Developer, Designer, and Teacher



## **Node.js, Express, MongoDB & More: The Complete Bootcam...**

Jonas Schmedtmann, Web Developer, Designer, and Teacher



## **Advanced CSS and Sass: Flexbox, Grid, Animations and...**

Jonas Schmedtmann, Web Developer, Designer, and Teacher



## **Build Responsive Real-World Websites with HTML and CSS**

Jonas Schmedtmann, Web Developer, Designer, and Teacher

- ▣ <https://www.udemy.com/>
- ▣ <https://codingheroes.io/>

# Introductie Node.js





## Introductie Node.js

- Node.js is een JavaScript runtime ontwikkeld op Google's open-source V8 JavaScript engine
- Laat toe om JavaScript code uit te voeren op een server zonder de restricties van de browser (toegang tot het bestandssysteem e.d.)
- De V8 engine parses de JavaScript code en voert ze uit (bv. starten HTTP-server)



# Introductie Node.js

- ▣ Wat kunnen we met JavaScript op de server?
  - Laat toe het bestandssysteem te benaderen
  - Heeft betere netwerkfunctionaliteiten
  - Laat toe om aan server-side web development te doen
  - Laat toe om snelle en schaalbare web applicaties te ontwikkelen

# Introductie Node.js

## ■ Waarom en wanneer Node.js gebruiken?

- ▬ Single threaded, event driven, non-blocking I/O model
- ▬ Perfect voor snelle en schaalbare data-intensieve web applicaties
- ▬ Perfect geschikt voor:
  - API met een database (liefst NoSQL)
  - Data streaming (denk aan YouTube)
  - Real-time chat applicaties
  - Server-side web applicaties waarbij de volledige content op de server wordt gegenereerd
- ▬ Zelfde taal (JavaScript) voor front-end en back-end

# Installatie Node.js

- ▣ <https://nodejs.org/en/>
- ▣ Kies voor de LTS versie
- ▣ Installeer ook de bijhorende tools
- ▣ Controleer de installatie via de terminal:
  - node -v
  - npm -v



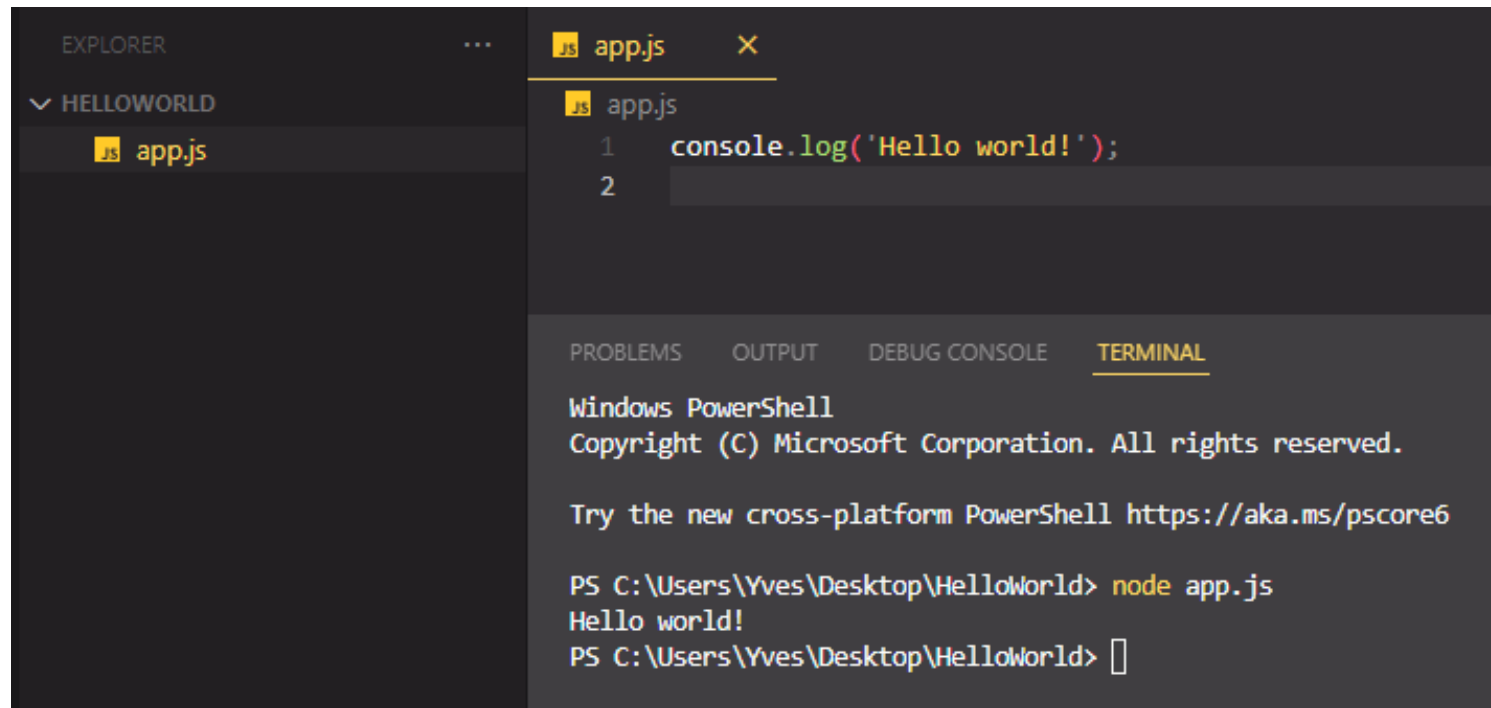
## De REPL

- ▣ Om Node.js code te schrijven, gewoon node in terminal typen
- ▣ Dit opent de REPL (Read-Eval-Print-Loop)
- ▣ In de REPL kan JavaScript worden geschreven
- ▣ De REPL verlaten doe je via .exit of CTRL+d
- ▣ 2 x tab geeft alle globale variabelen en node modules weer
- ▣ De underscore bevat het vorige resultaat van een berekening



## Uitvoeren .js-bestand

▣ node someJavaScriptFile.js



The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows a folder named 'HELLOWORLD' containing a file 'app.js'. The main editor area shows the content of 'app.js', which is a single line of JavaScript code: `console.log('Hello world!');`. Below the editor, the TERMINAL pane is active, displaying the Windows PowerShell prompt. The command `node app.js` has been entered and executed, resulting in the output `Hello world!`.

# File System Module





## File System Module

- ▣ Met node hebben we ook toegang tot het bestandssysteem
- ▣ Dit wordt mogelijk gemaakt via een node module, de fs module
- ▣ Node.js is gebouwd rond het concept van modules voor het toevoegen van functionaliteiten
- ▣ Modules kunnen gebruikt worden via het require keyword en het resultaat toe te kennen aan een variabele

# File System Module

- ▣ "require" is het keyword voor het CommonJS module systeem om modules te importeren

```
Js index.js  X  output.txt
Js index.js > ...
1  const fs = require('fs');
2
3  const textIn = fs.readFileSync('./txt/input.txt', 'utf-8');
4  console.log(textIn);
5
6  const textOut = `This is what we know about the avocado: ${textIn}.\nCreated on ${Date.now()}`;
7  fs.writeFileSync('./txt/output.txt', textOut);
8  console.log('File written');
```

# File System Module

## ▣ RECAP:

- ▣ Synchron: elke lijn code wordt pas uitgevoerd nadat de vorige lijn klaar is
  - ▣ Synchrone code is blokkerende code
  - ▣ Asynchrone code is non-blocking
- ▣ In het Node.js proces waar onze applicatie wordt uitgevoerd is er slechts één enkele thread
  - ▣ Een thread is een set van instructies die wordt uitgevoerd in de CPU



## File System Module

- ▣ De thread is dus de plaats waar onze applicatie wordt uitgevoerd in de CPU
- ▣ Voor elke applicatie op Node.js is er dus slechts één thread
- ▣ Alle gebruikers die gebruik maken van een applicatie gebruiken dus allemaal dezelfde thread

## File System Module

- ▣ Veronderstel nu dat de applicatie voor een specifieke gebruiker een stukje blokkerende synchrone code is voorzien, dan zal het uitvoeren van de applicatie bij alle andere gebruikers op het moment dat die gebruiker die synchrone blokkerende code uitvoert, ook geblokkeerd worden. Wat indien er duizenden gebruikers op dat moment de applicatie gebruiken???

# Eenvoudige webserver

http module





# Eenvoudige web server

```
JS index.js X
JS index.js > ...
1  const fs = require('fs');
2  const http = require('http');
3
4  > /* //Blocking, synchronous way...
11
12 > /* //non-blocking, asynchronous way...
26
27  // Web Server
28  const server = http.createServer((req, res) => {
29    |   res.end('Hello from the server!');
30    | });
31
32  server.listen(8000, '127.0.0.1', () => {
33    |   console.log('Listening to requests on port 8000');
34    | });
```

## Eenvoudige web server

- ▣ Importeer de http module
- ▣ de `createServer()` methode krijgt een callback function als parameter die elke keer zal worden uitgevoerd wanneer de server een request ontvangt
- ▣ De callback function krijgt toegang tot de request en response argumenten (dit zijn objecten!)
- ▣ Via de `end()` methode van het response object kan iets worden teruggestuurd



## Eenvoudige web server

- ▣ Om te luisteren naar requests moet het resultaat van de `createServer()` methode aan een variabele worden toegekend
- ▣ Vervolgens wordt de `listen()` methode op dit resultaat gebruikt waarbij de poort moet worden meegegeven, de host en een callback die wordt uitgevoerd van zodra de server is opgestart

# Routing



# Routing

- ▣ Configureren van verschillende acties voor verschillende URL's
- ▣ Voor geavanceerde applicaties worden frameworks zoals Express gebruikt dat out-of-the-box voorzien is van routing functionaliteiten
- ▣ Om het manueel te configureren in Node.js wordt de module "url" gebruikt
- ▣ Om verschillende routes te configureren volstaat een if-statement
- ▣ Gebruik de writeHead() methode om ook de statuscode in te stellen

# Routing

```
JS index.js X
JS index.js > ...
1  const fs = require('fs');
2  const http = require('http');
3  const url = require('url');
4
5  > /* //Blocking, synchronous way...
12
13 > /* //non-blocking, asynchronous way...
27
28 // Web Server
29 const server = http.createServer((req, res) => {
30   const pathName = req.url;
31
32   if (pathName === '/' || pathName === '/overview') {
33     res.end('This is the OVERVIEW');
34   } else if (pathName === '/product') {
35     res.end('This is the PRODUCT');
36   } else {
37     res.writeHead(404, {
38       'Content-type': 'text/html',
39     });
40     res.end('<h1>This page could not be found</h1>');
41   }
42 });
43
44 server.listen(8000, '127.0.0.1', () => {
45   console.log('Listening to requests on port 8000');
46 });
```

## Routing

- ▣ In de writeHead() methode kan ook een object worden meegegeven om de headers van de response in te stellen
- ▣ Headers is informatie over de response
- ▣ LET OP: de headers moeten altijd worden ingesteld voordat de response wordt verstuurd

# Eenvoudige API





## Eenvoudige API

- ▣ Een web API is een service waar we data kunnen van opvragen of naartoe sturen
- ▣ Als dataformaat wordt meestal JSON gebruikt (JavaScript Object Notation)
- ▣ JSON lijkt heel goed op JavaScript objecten
- ▣ De keys in JSON zijn altijd een string tussen quotes
- ▣ JSON is eigenlijk gewoon een string
- ▣ `JSON.parse()` transformeert een JSON string naar JavaScript objecten

```

index.js x {} data.json
index.js > ...
1  const fs = require('fs');
2  const http = require('http');
3  const url = require('url');
4
5  > /* //Blocking, synchronous way...
12
13 > /* //non-blocking, asynchronous way...
27
28 // Web Server
29 const data = fs.readFileSync(`${__dirname}/dev-data/
data.json`, 'utf-8');
30
31 const server = http.createServer((req, res) => {
32   const pathName = req.url;
33
34   if (pathName === '/' || pathName === '/overview')
   {
35     res.end('This is the OVERVIEW');
36   } else if (pathName === '/product') {
37     res.end('This is the PRODUCT');
38   } else if (pathName === '/api') {
39     res.writeHead(200, {
40       'Content-Type': 'application/json',
41     });
42     res.end(data);
43   } else {
44     res.writeHead(404, {
45       'Content-type': 'text/html',
46     });
47     res.end('<h1>This page could not be found</h1>');
48   }
49 });
50
51 server.listen(8000, '127.0.0.1', () => {
52   console.log('Listening to requests on port 8000');
53 });
54

```

```

data.json x
dev-data > {} data.json > ...
1  [
2
3    {
4      "id": 0,
5      "productName": "Fresh Avocados",
6      "image": "🥑",
7      "from": "Spain",
8      "nutrients": "Vitamin B, Vitamin K",
9      "quantity": "4 🥑",
10     "price": "6.50",
11     "organic": true,
12     "description": "A ripe avocado yields to gentle
pressure when held in the palm of the hand and
squeezed. The fruit is not sweet, but
distinctly and subtly flavored, with smooth
texture. The avocado is popular in vegetarian
cuisine as a substitute for meats in sandwiches
and salads because of its high fat content.
Generally, avocado is served raw, though some
cultivars, including the common 'Hass', can be
cooked for a short time without becoming
bitter. It is used as the base for the Mexican
dip known as guacamole, as well as a spread on
corn tortillas or toast, served with spices."
13   },
14   {
15     "id": 1,
16     "productName": "Goat and Sheep Cheese",
17     "image": "🧀",
18     "from": "Portugal",
19     "nutrients": "Vitamin A, Calcium",
20     "quantity": "250g",
21     "price": "5.00",
22     "organic": false,
23     "description": "Creamy and distinct in flavor,
goat cheese is a dairy product enjoyed around
the world. Goat cheese comes in a wide variety
of flavors and textures, from soft and
spreadable fresh cheese to salty, crumbly aged
cheese. Although it's made using the same
coagulation and separation process as cheese
made from cow's milk, goat cheese differs in
nutrient content."
24   }
25 ]

```

```

127.0.0.1:8000/api
Google - My Activity Archive Dashboards Online Drives
[
  {
    "id": 0,
    "productName": "Fresh Avocados",
    "image": "🥑",
    "from": "Spain",
    "nutrients": "Vitamin B, Vitamin K",
    "quantity": "4 🥑",
    "price": "6.50",
    "organic": true,
    "description": "A ripe avocado yields to
but distinctly and subtly flavored, with
in sandwiches and salads because of its h
common 'Hass', can be cooked for a short
guacamole, as well as a spread on corn to
  },
  {
    "id": 1,
    "productName": "Goat and Sheep Cheese",
    "image": "🧀",
    "from": "Portugal",
    "nutrients": "Vitamin A, Calcium",
    "quantity": "250g",
    "price": "5.00",
    "organic": false,
    "description": "Creamy and distinct in fl
wide variety of flavors and textures, from
using the same coagulation and separation
  }
]

```

# Modules



# Modules

- In Node.js wordt elk bestand als een module behandeld
- In elke module hebben we toegang tot de module variabele waarvan we de exports property kunnen instellen: `module.exports = ...`
- Volgorde van importeren:
  - ▬ Altijd eerst de core Node.js modules
  - ▬ Vervolgens de externe libraries
  - ▬ Tot slot de eigen modules

# Modules

- ▣ Gebruik `__dirname` variabele in paden
- ▣ `__dirname` verwijst naar het absolute pad van het bestand dat wordt uitgevoerd
- ▣ De `./` in een pad verwijst naar de locatie van waaruit gewerkt wordt (met uitzondering van `require` waar het zich zoals `__dirname` gedraagt!)

# Modules

```
JS app.js X
JS app.js > ...
1  const path = require('path');
2
3  console.log('__dirname: ', __dirname);
4  console.log('./ : ', path.resolve('./'));
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Yves\Desktop\HelloWorld> node app.js
__dirname:      C:\Users\Yves\Desktop\HelloWorld
./ : C:\Users\Yves\Desktop\HelloWorld
PS C:\Users\Yves\Desktop\HelloWorld> 
```

```
JS app.js X
JS app.js > ...
1  const path = require('path');
2
3  console.log('__dirname: ', __dirname);
4  console.log('./ : ', path.resolve('./'));
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Yves\Desktop> node HelloWorld/app.js
__dirname:      C:\Users\Yves\Desktop\HelloWorld
./ : C:\Users\Yves\Desktop
PS C:\Users\Yves\Desktop> 
```



index.js × replaceTemplate.js template-product.html template-overview.html template-card.html

index.js > [0] replaceTemplate

```
1 const fs = require('fs');
2 const http = require('http');
3 const url = require('url');
4 const replaceTemplate = require('./modules/replaceTemplate');
5
6 > /* //Blocking, synchronous way...
13
14 > /* //non-blocking, asynchronous way...
28
29 // Web Server
30
31 const tempOverview = fs.readFileSync(
32   `${__dirname}/templates/template-overview.html`,
33   'utf-8'
34 );
35 const tempCard = fs.readFileSync(
36   `${__dirname}/templates/template-card.html`,
37   'utf-8'
38 );
39 const tempProduct = fs.readFileSync(
40   `${__dirname}/templates/template-product.html`,
41   'utf-8'
42 );
43 const data = fs.readFileSync(`${__dirname}/dev-data/data.json`, 'utf-8');
44 const dataObj = JSON.parse(data);
45
```

replaceTemplate.js ×

modules > replaceTemplate.js > ...

```
1 module.exports = (temp, product) => {
2   ...
3   let output = temp.replace(/{%PRODUCTNAME%}/g, product.productName);
4   output = output.replace(/{%IMAGE%}/g, product.image);
5   output = output.replace(/{%PRICE%}/g, product.price);
6   output = output.replace(/{%FROM%}/g, product.from);
7   output = output.replace(/{%NUTRIENTS%}/g, product.nutrients);
8   output = output.replace(/{%QUANTITY%}/g, product.quantity);
9   output = output.replace(/{%DESCRIPTION%}/g, product.description);
10  output = output.replace(/{%ID%}/g, product.id);
11
12  if (!product.organic)
13    output = output.replace(/{%NOT_ORGANIC%}/g, 'not-organic');
14  return output;
15 };
16
```

**Odisee**  
DE CO-HOGESCHOOL

◀

# npm packages





# npm packages

## ▣ RECAP:

- npm init (-y)
- npm i somePackage
- npm i -D somePackage of npm i somePackage --sav-dev
- dependencies: modules die in eigen code worden gebruikt (afhankelijkheden)
- devDependencies: tools voor development (testing, debugging, building etc.)
- npm i -g someDevDependency



# npm packages

## ■ nodemon:

- ▬ Installeren onder devDependencies of globaal via -g
- ▬ Tool die server automatisch herstart na opslaan van wijzigingen
- ▬ Opstarten:
  - globaal: `nodemon someModule.js`
  - lokaal:
    - ▼ `npx nodemon someModule.js`
    - ▼ `npm start` (indien met npm scripts wordt gewerkt)



## npm packages

- De npm symantic version notation bestaat uit 3 getallen:
  - ▬ Het eerste getal is de major version
  - ▬ Het tweede getal is de minor version
  - ▬ Het derde getal is de patch version
- Enkel de patch version wordt gebruikt om bugs op te lossen
- De minor version introduceert nieuwe features maar geen breaking changes zodat alle wijzigingen steeds backwards compatibel zijn



## npm packages

- De major version is een grote nieuwe release die de huidige werking van de code kan breken omdat bijvoorbeeld functions een andere naam hebben gekregen of omdat de parameters gewijzigd zijn etc.
- Het eerste symbool geeft aan welke updates automatisch aanvaard worden:
  - ^ betekent minor en patch release updates
  - ~ betekent enkel patch releases
  - \* betekent alles, zowel major, minor als patch (niet aanbevolen!)

## npm packages

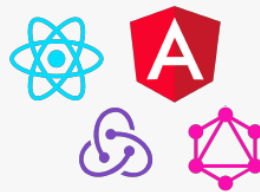
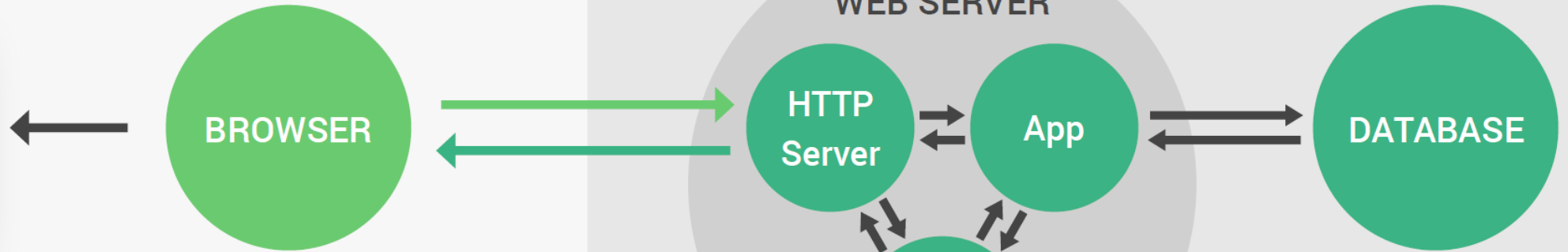
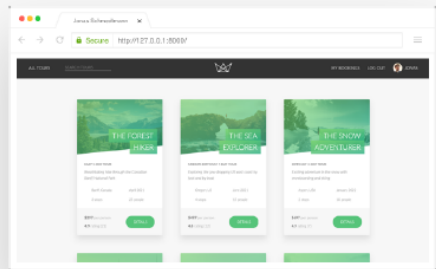
- ▣ Om te controleren of er packages verouderd zijn:
  - ▬ npm outdated
- ▣ Om een bepaalde versie van een package te installeren:
  - ▬ npm i package-name@1.0.0
- ▣ Om een package te updaten:
  - ▬ npm update package-name
- ▣ Om dependencies opnieuw te installeren:
  - ▬ npm install

# Front-end vs. back-end web development



FRONT-END

BACK-END



FRONT-END STACK



BACK-END STACK



# Express.js





## Express.js

- ▣ <https://expressjs.com/>
- ▣ Een minimaal Node.js framework hetgeen betekent dat het bovenop Node.js is gebouwd
- ▣ Het populairste Node.js framework
- ▣ Bevat een robuuste set van features zoals complex routing, eenvoudigere afhandeling van requests en responses, server-side rendering etc.
- ▣ Hierdoor laat het toe om snel Node.js applicaties te ontwikkelen
- ▣ Laat toe om applicaties te organiseren volgens de MVC architectuur

# Express.js

- ▣ npm i express
- ▣ De conventie is dat alle Express configuratie in het bestand app.js wordt geplaatst
- ▣ De require van Express geeft een function terug dat bij het oproepen ervan een hoop methodes teruggeeft aan de variabele waar de function werd aan toegekend
- ▣ Via de listen() methode wordt de web server opgestart

- Aan de `listen()` methode wordt een poort en een optionele callback meegegeven als argumenten
- Om routes toe te voegen:
  - `app.SOME_HTTP_METHOD('some_url', (request, response) => { ... } );`
- De `json()` methode zet automatisch het Content-Type op `application/json`

package.json app.js

app.js > app.post('/') callback

```
1 const express = require('express');
2
3 const app = express();
4
5 app.get('/', (req, res) => {
6   //res.status(200).send('Hello from the server side!');
7   res
8     .status(200)
9     .json({ message: 'Hello from the server side!', app: 'Natours' });
10 });
11
12 app.post('/', (req, res) => {
13   res.send('You can post to this endpoint...');
14 });
15
16 const port = 3000;
17 app.listen(port, () => {
18   console.log(`App running on port ${port}...`);
19 });
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Yves\OneDrive - Full Stack Solutions\NodeExpressMongoDB\complete-node-bootcamp-master\4-natours\starter> nodemon app.js

[nodemon] 2.0.16  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): \*.\*  
[nodemon] watching extensions: js,mjs,json  
[nodemon] starting `node app.js`

File Edit View Help

Home Workspaces API Network Reports Explore

GET localhost:3000

localhost:3000

GET localhost:3000

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Hello from the server side!",
3   "app": "Natours"
4 }
```

```
package.json app.js x
app.js > app.post('/') callback
1 const express = require('express');
2
3 const app = express();
4
5 app.get('/', (req, res) => {
6   //res.status(200).send('Hello from the server side!');
7   res
8     .status(200)
9     .json({ message: 'Hello from the server side!', app: 'Natours' });
10 });
11
12 app.post('/', (req, res) => {
13   res.send('You can post to this endpoint...');
14 });
15
16 const port = 3000;
17 app.listen(port, () => {
18   console.log(`App running on port ${port}...`);
19 });
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Yves\OneDrive - Full Stack Solutions\NodeExpressMongoDB\complete-node-bootcamp-master\4-natours\starter> nodemon app.js

```
[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
App running on port 3000...
```

File Edit View Help

Home Workspaces API Network Reports Explore

POST localhost:3000 + ...

localhost:3000

POST localhost:3000

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize HTML

```
1 You can post to this endpoint...
```

## ▣ GET requests

- Voorzie altijd een versienummer in de API zodat bij wijzigingen de oude gewoon kan blijven werken
- Data inlezen nooit in de route handler callback doen maar ervoor (synchroon)!
- Geeft status 200 terug

```

1  const fs = require('fs');
2  const express = require('express');
3
4  const app = express();
5
6  // app.get('/', (req, res) => {
7  //    //res.status(200).send('Hello from the server side!');
8  //    res
9  //        .status(200)
10 //        .json({ message: 'Hello from the server side!', app: 'Natours' });
11 // });
12
13 // app.post('/', (req, res) => {
14 //    res.send('You can post to this endpoint...');
15 // });
16
17 const tours = JSON.parse(
18     fs.readFileSync(`${__dirname}/dev-data/data/tours-simple.json`)
19 );
20
21 app.get('/api/v1/tours', (req, res) => {
22     res.status(200).json({
23         // Use the JSend formatting convention
24         status: 'success',
25         results: tours.length,
26         data: {
27             // When property and variable have the same name
28             // it's not necessary to write them both, just tours
29             // will also work
30             tours: tours,
31         },
32     });
33 });
34
35 const port = 3000;
36 app.listen(port, () => {
37     console.log(`App running on port ${port}...`);
38 });

```

## ■ POST requests

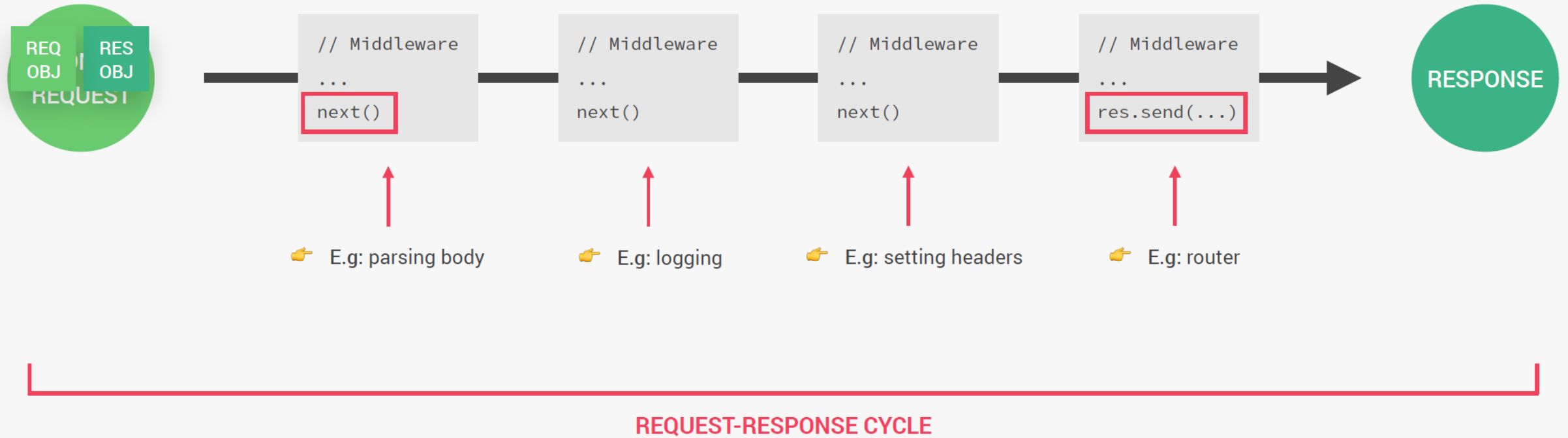
- Out-of-the-box voorziet Express.js niet de body data van een post request in de request
- Om aan de data in de body van een POST request te kunnen moet de `express.json()` middleware worden gebruikt
- Om middlewares te gebruiken wordt de `use()` methode gebruikt:
  - `app.use(express.json());`
- Een middleware is een function die de binnenkomende request kan wijzigen



👉 “Everything is middleware” (even routers)

👉 “Pipeline”

👉 Order as defined in the code!



## ■ POST requests

- ▬ De `express.json()` middleware gaat de data in de body van de POST request toevoegen aan het request object
- ▬ Geeft status 201 terug



## ■ POST requests

- Gebruik altijd dubbele quotes in JSON
- Met `Object.assign` kan een nieuw object worden gecreëerd door twee objecten te mergen met elkaar
- In het voorbeeld wordt `fs.writeFile()` (ASYNCHROON) gebruikt omdat dit reeds wordt uitgevoerd in een asynchrone callback en het niet mogelijk is om de event loop te blokkeren
- TIP: om een nummer als string te converteren naar een nummer kan de string gewoon vermenigvuldigd worden met 1

## ■ URL parameters

- ▬ Om variabelen in een route te definiëren wordt ":" gebruikt
  - `app.get('/api/v1/tours/:id' ...`
- ▬ Via `request.params` zijn de variabelen toegankelijk in code
- ▬ De ingegeven route moet exact gelijk zijn aan de variabelen
- ▬ Wanneer een variabele niet wordt meegegeven dan geeft dit een fout
- ▬ De oplossing zijn optionele parameters:
  - `app.get('/api/v1/tours/:id/:x/:y?' ...`

```
1  const fs = require('fs');
2  const express = require('express');
3
4  const app = express();
5
6  // Middleware to add body data to the request object
7  app.use(express.json());
8
9  > // app.get('/', (req, res) => {...
19
20 > const tours = JSON.parse(...
22   );
23
24 > app.get('/api/v1/tours', (req, res) => {...
36   });
37
38   app.get('/api/v1/tours/:id', (req, res) => {
39     console.log(req.params);
40
41     const id = req.params.id * 1;
42
43     const tour = tours.find((el) => el.id === id);
44
45     if (!tour) {
46       return res.status(404).json({
47         status: 'fail',
48         message: 'Invalid ID',
49       });
50     }
51
52     res.status(200).json({
53       // Use the JSend formatting convention
54       status: 'success',
55       data: {
56         tour,
57       },
58     });
59   });
60
61 > app.post('/api/v1/tours', (req, res) => {...
78   });
79
80   const port = 3000;
81 > app.listen(port, () => {...
83   });
```

# Express.js

## ▣ PUT requests

- ▬ Vereist het volledige object, inclusief de niet gewijzigde properties
- ▬ Geeft status 200 terug

## ▣ PATCH requests

- ▬ Vereist enkel de te wijzigen properties
- ▬ Geeft status 200 terug

## ▣ DELETE requests

- ▬ Geeft status 204 terug en als data null

# MySQL





## MySQL

- ▣ <https://expressjs.com/en/guide/database-integration.html>
- ▣ <https://expressjs.com/en/guide/database-integration.html#mysql>
- ▣ npm i mysql
- ▣ Voorzie een afzonderlijke gebruiker met een standaard wachtwoord en geef die alle rechten op de database

```
const mysql = require('mysql')
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'dbuser',
  password: 's3kre337',
  database: 'my_db'
})

connection.connect()

connection.query('SELECT 1 + 1 AS solution', (err, rows, fields) => {
  if (err) throw err

  console.log('The solution is: ', rows[0].solution)
})

connection.end()
```