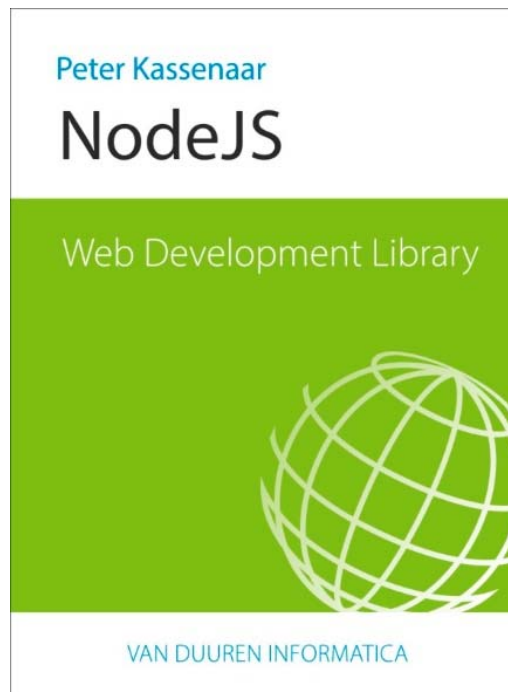


Node.js



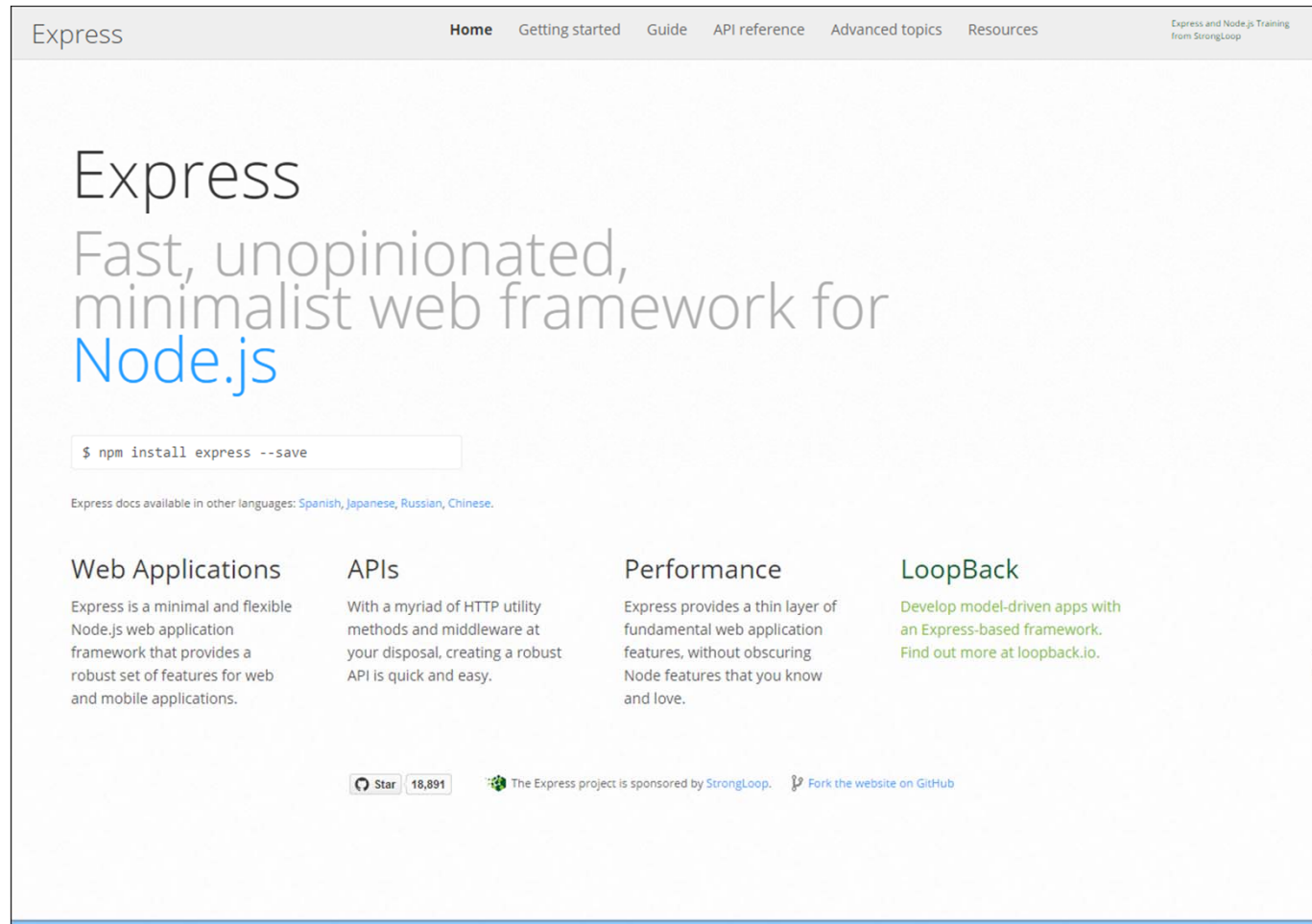
Peter Kassenaar

Module 3 – Express.js framework



Hoofdstuk 5, p.99 en verder

Express.js framework



<http://expressjs.com/>

Express: twee manieren

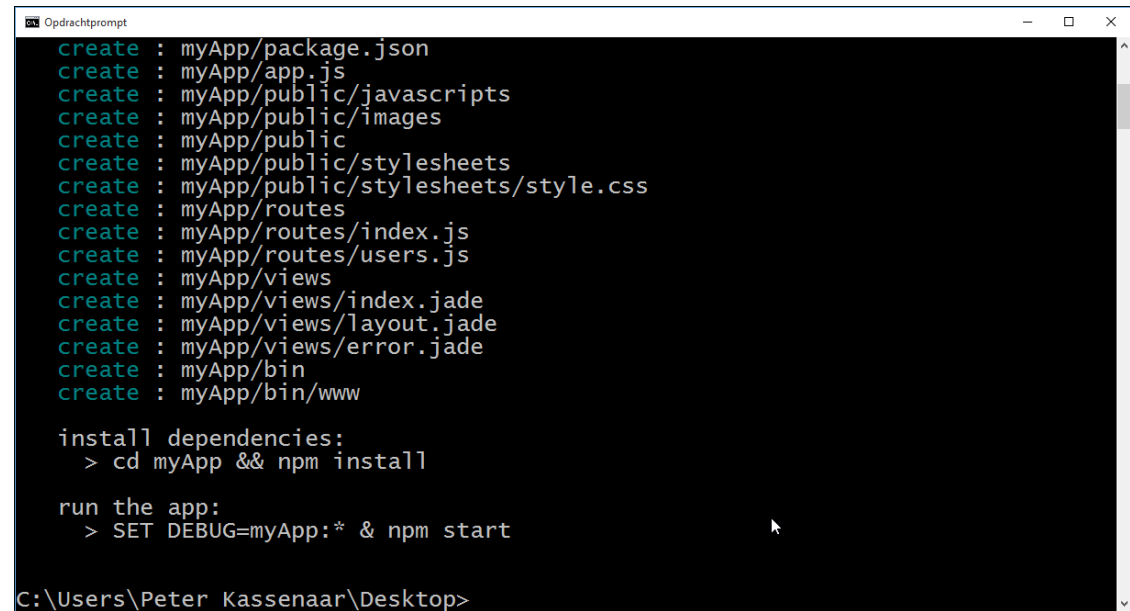
1. Globaal: `npm install -g express-generator`

- Generator
- Snelle scaffolding van kale basiswebsite
- Voordeel: snel op weg
- Nadeel: veel code die je wellicht niet wilt gebruiken, of volgens andere conventies

2. Lokaal: `npm install express`, per project

Express-generator

1. Ga naar gewenste directory
2. `express myApp`
3. `cd myApp`
4. `npm install`
5. `npm start`

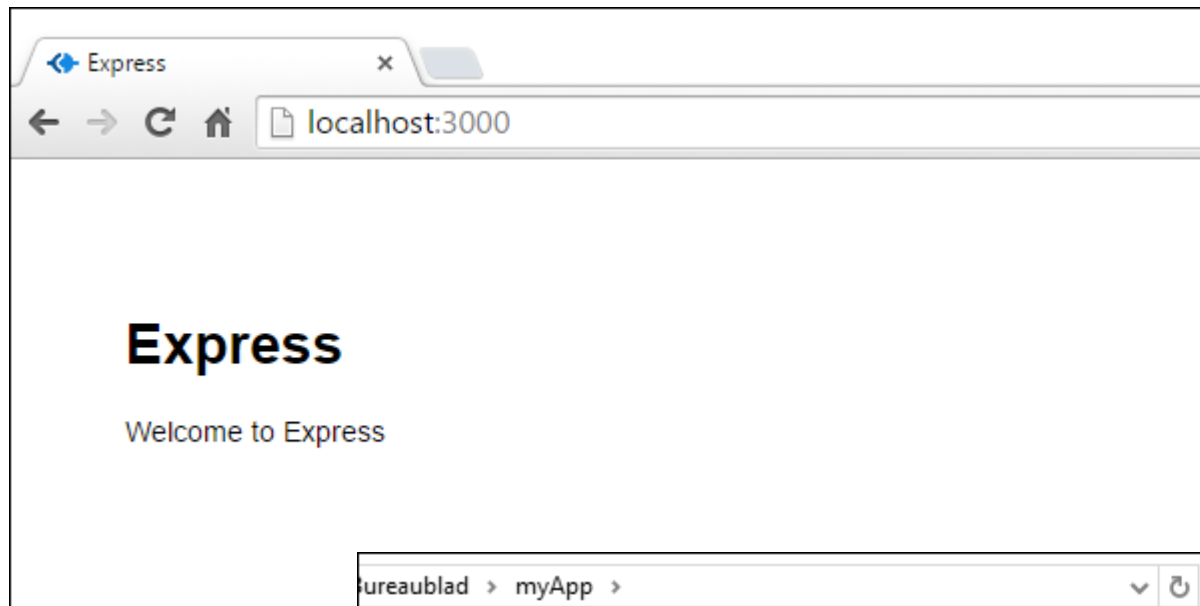


```
Opdrachtprompt
create : myApp/package.json
create : myApp/app.js
create : myApp/public/javascripts
create : myApp/public/images
create : myApp/public
create : myApp/public/stylesheets
create : myApp/public/stylesheets/style.css
create : myApp/routes
create : myApp/routes/index.js
create : myApp/routes/users.js
create : myApp/views
create : myApp/views/index.jade
create : myApp/views/layout.jade
create : myApp/views/error.jade
create : myApp/bin
create : myApp/bin/www

install dependencies:
> cd myApp && npm install

run the app:
> SET DEBUG=myApp:* & npm start

C:\Users\Peter Kassenaar\Desktop>
```



bureaublad > myApp >

Zoeken in myApp 🔍

Naam	Gewijzigd op	Type	Grootte
bin	24-9-2015 08:30	Bestandsmap	
node_modules	24-9-2015 08:31	Bestandsmap	
public	24-9-2015 08:30	Bestandsmap	
routes	24-9-2015 08:30	Bestandsmap	
views	24-9-2015 08:30	Bestandsmap	
app.js	24-9-2015 08:30	JavaScript File	2 kB
package.json	24-9-2015 08:30	JSON File	1 kB

```
.
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.jade
    ├── index.jade
    └── layout.jade
```

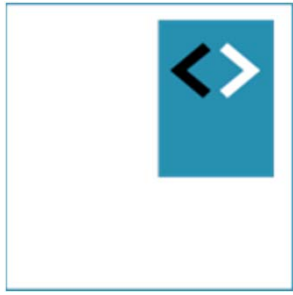
7 directories, 9 files

The app structure generated by the generator is just one of the multiple ways of structuring Express apps. Feel free to not use it or to modify it to best suit your needs.

Persoonlijke mening: *ik* gebruik het niet veel (=nooit).

Moet te veel refactoren voordat de indeling naar mijn wens is.

Ik gebruik geen View Engines zoals Jade, EJS of andere.



Lokale installatie Express 4.x

Unopiniated framework –

“Configuration over Convention”

Lokaal toevoegen en configureren

```
> npm install express --save
```

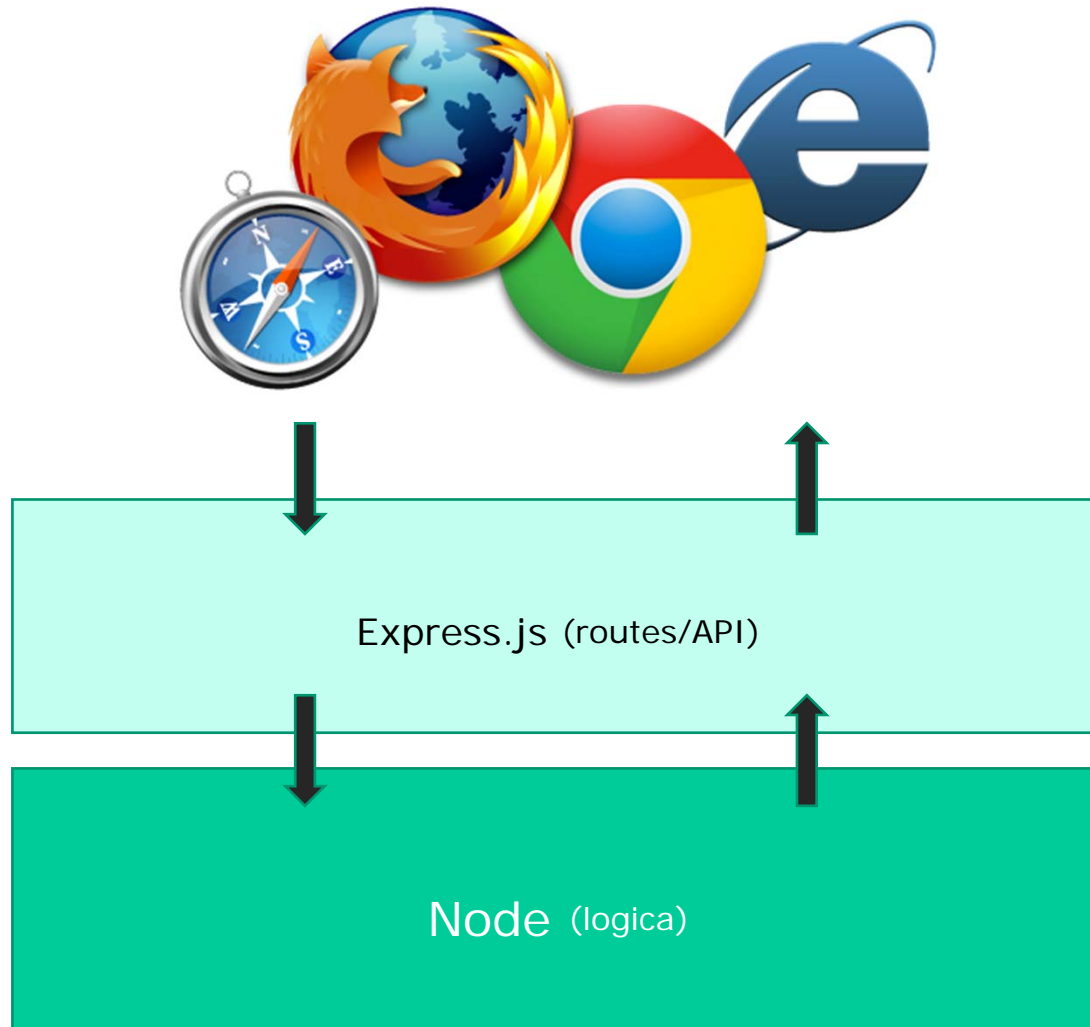
```
> npm install body-parser --save
```

Wordt geïnstalleerd in `\node_modules`

body-parser: automatisch omzetten van JSON in POSTS-requests

Dan: `app.js` (of `server.js`) maken.

Middleware



Eerste express-app

```
var express = require('express');
var app = express();

// 1. Routes voor deze app - ten eerste de Homepage
app.get('/', function (req, res) {
  res.send('Hello World - dit is Express!');
});

// 2. JSON retour zenden
var persoon = {
  voornaam : 'Peter',
  achternaam: 'Kassenaar'
};
app.get('/json', function (req, res) {
  res.send(persoon);
});

app.listen(3000);
console.log('Express-server gestart op http://localhost:3000');
```



p.104

Checkpoint

- Express is een framework om Node.js-webapplicaties te maken
- Zowel globale als lokale installatie mogelijk
- Configuration over convention
- Snel op weg: `npm install --save express`

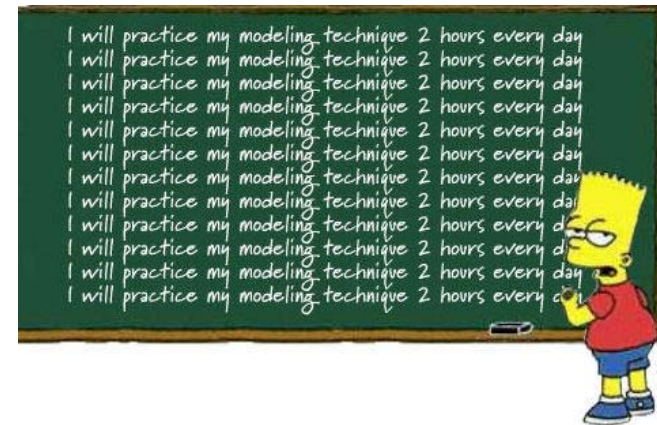
Oefening – generator & lokaal

1. Installeer Express-generator.

- Maak een globale app en draai deze.
- Bekijk de structuur van de site, pas aan en beoordeel deze.

2. Installeer Express lokaal

- Maak eerste Express-app en serveer data.



Express API maken

Routes schrijven binnen je app

```
app.get('/api/auteurs', function (req, res) {  
  // retourneer gegevens, hier een JSON-bestand met auteurs  
  res.json(auteurs);  
});
```

Elke functionaliteit krijgt een eigen RESTful route



p.109

Routeparameters – dubbele punt

```
app.get('/api/boeken/:id', function (req, res) {  
  var id = req.params.id;  
  var gezochtBoek;  
  boeken.forEach(function (boek) {  
    if (boek.id === parseInt(id)) {  
      gezochtBoek = boek;  
    }  
  });  
  // Boek niet gevonden  
  if (!gezochtBoek) {  
    gezochtBoek = {  
      error: 'Boek niet gevonden'  
    }  
  }  
  res.json(gezochtBoek);  
});
```



Serving static files

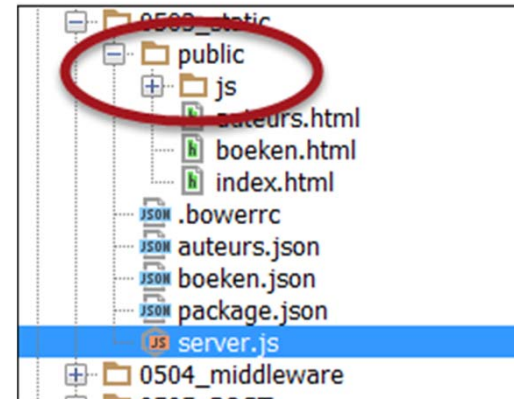
- *middleware* gebruiken
- `app.use()`

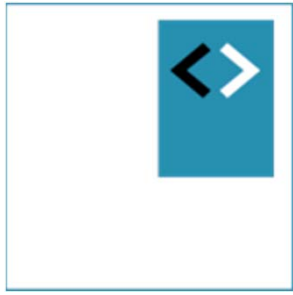
```
var express = require('express');  
var app = express();  
// ...  
// Stel middleware in voor serveren van  
// statische bestanden (HTML, CSS, images)  
app.use(express.static(__dirname + '/public'));  
// ...
```



Publieke website

- Front-end van de website/-app komt in \public
 - HTML, CSS, JavaScript, images, pdf's, enzovoort.
 - AngularJS vaak ingezet als client-sided library





Middleware gebruiken

Extra bewerkingen uitvoeren op inkomende en uitgaande requests

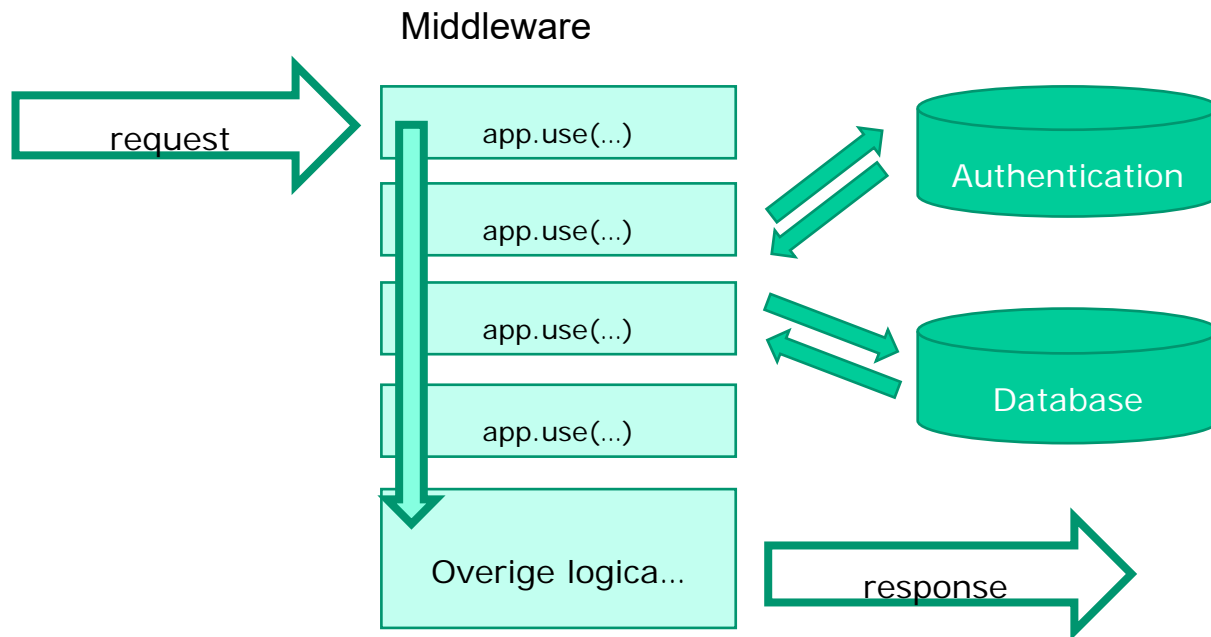
Over middleware

“Middleware is een functie die wordt uitgevoerd op het binnenkomende request. Het resultaat wordt doorgegeven aan een volgende functie in de keten. Net zo lang tot er geen middleware-functies meer zijn.”

Voorbeeld middleware

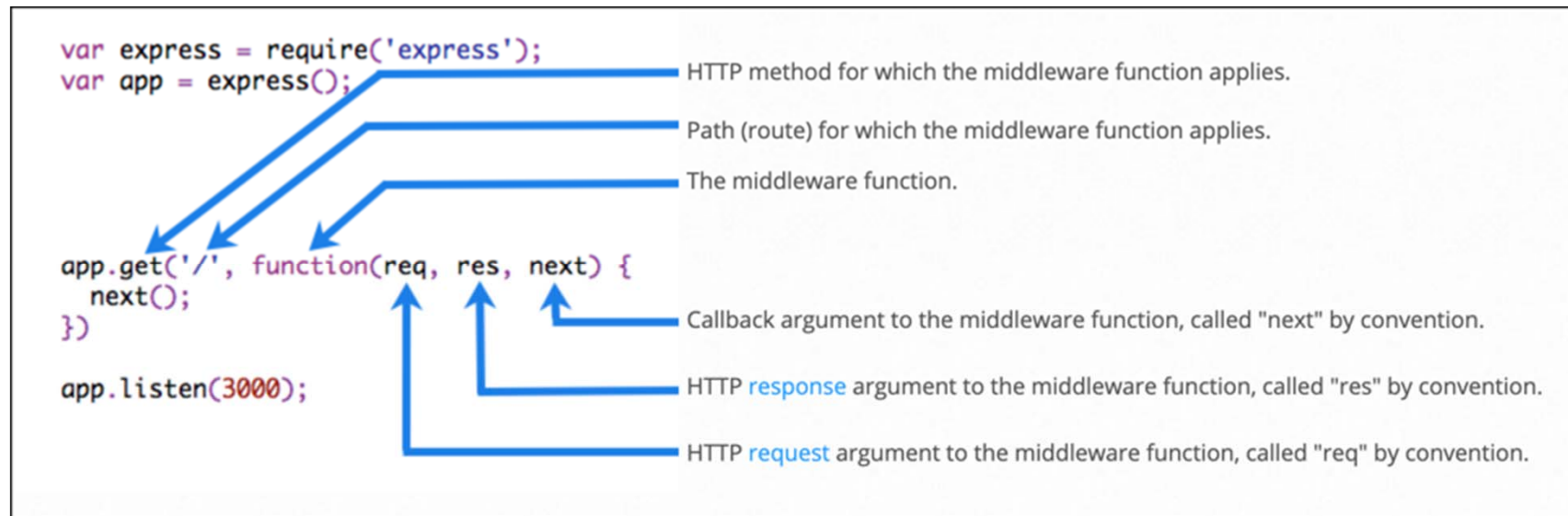
```
//*****  
  
// 3. Eigen middleware-functie  
  
//*****  
  
app.use(function(req, res, next){  
    console.log('Requested file: ', req.url);  
    next();  
});  
  
//*****  
  
// 3a. Middleware-functie voor één route  
  
//*****  
  
app.use('/index.html', function(req, res, next){  
    console.log('Homepage opgevraagd: ', new Date());  
    next();  
});  
  
// 4. Stel middleware in voor serveren van statische bestanden (HTML, CSS, images)  
app.use(express.static(__dirname + '/public'));
```

Meer middleware



“Order *does* matter!”

Middleware



<http://expressjs.com/en/guide/writing-middleware.html>

Type middleware

Application-level middleware

Bind application-level middleware to an instance of the `app` object with `app.use()` and `app.METHOD()`, where `METHOD` is the HTTP method of the request that it handles, such as GET, PUT, POST, and so on, in lowercase. For example:

```
var app = express();

// a middleware with no mount path; gets executed for every request to the app
app.use(function (req, res, next) {
  console.log('Time:', Date.now());
  next();
});

// a middleware mounted on /user/:id; will be executed for any type of HTTP request to /user/:id
app.use('/user/:id', function (req, res, next) {
  console.log('Request Type:', req.method);
  next();
});

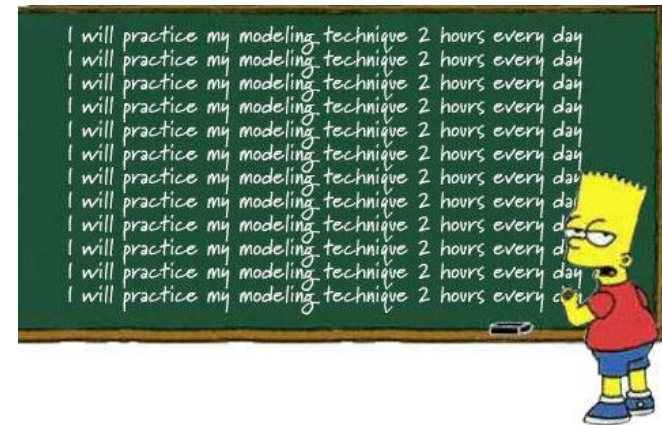
// a route and its handler function (middleware system) which handles GET requests to /user/:id
app.get('/user/:id', function (req, res, next) {
  res.send('USER');
});
```

<http://expressjs.com/guide/using-middleware.html>

Checkpoint

- Middleware zijn functies die worden uitgevoerd op binnenkomende requests
- Transformatie, authenticatie, filtering, logging, ...
- De volgorde van middleware is belangrijk

Oefening...



Werken met POST

- Extra middleware: `body-parser` om JSON in request te kunnen parsen
- Let op verschil Express 3.x en -4.x bij Blogs, Google en Stack Overflow!

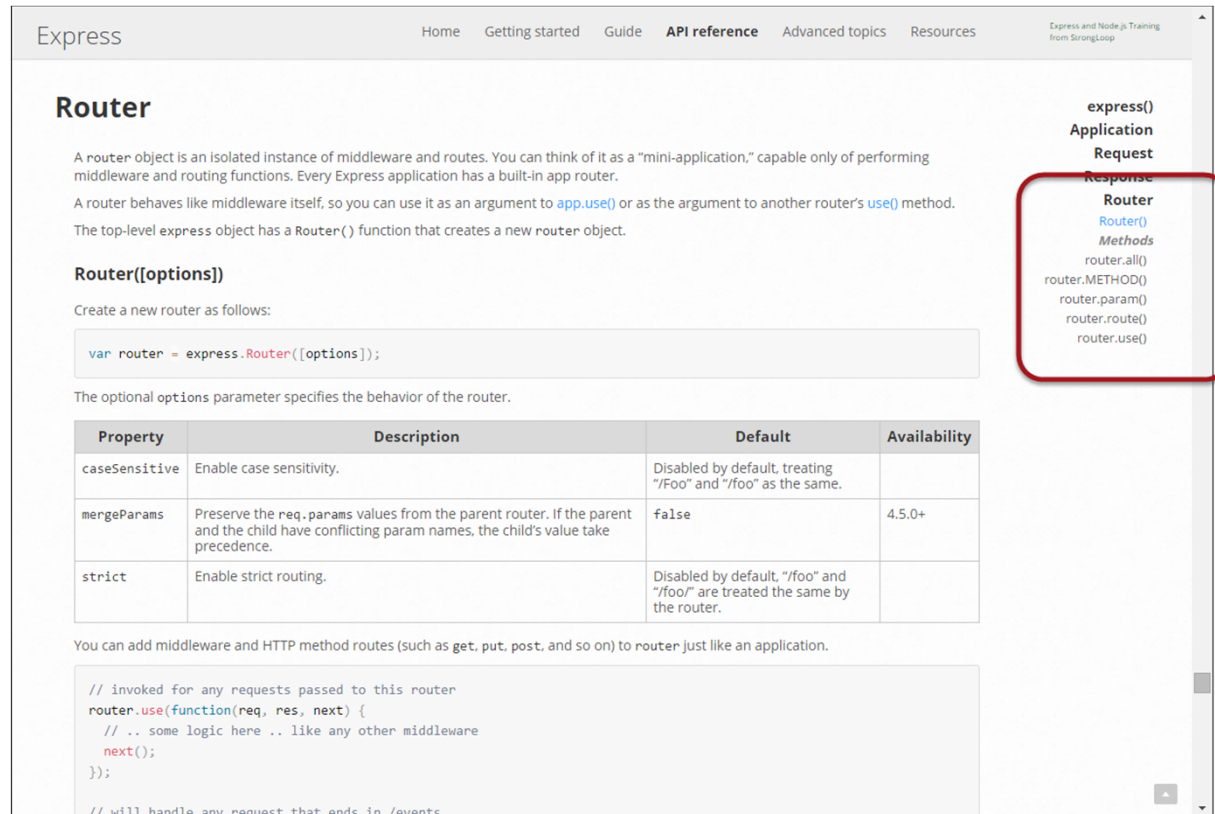
```
// 3. Middleware laden voor het parsen van JSON in het request  
app.use(bodyParser.urlencoded({  
  extended: true  
}));
```

- Gebruik `app.post()` voor schrijven POST-route
- Gebruik Postman (Chrome Extension) of HTML-formulier voor testing.

```
app.post('/user', function (req, res) {  
  // verwerk binnenkomende request. We gaan er van uit  
  // dat de parameter 'username' en 'email' aanwezig zijn.  
  // TODO: error checking!  
  console.log(req.body);  
  user.username = req.body.username;  
  user.email = req.body.email;  
  
  // Echo het user-object naar de client  
  res.json(user);  
});
```

express.Router()

- Voor abstractie van Routes in de app
- Gebruiken in combinatie met `app.use()`
- Wildcards en regex



The screenshot shows the Express.js API reference page for the `Router` module. The page is titled "Router" and provides a detailed description of its purpose and usage. It includes a code example for creating a new router, a table of optional options, and a code example for adding middleware and HTTP method routes. A sidebar on the right lists the available methods, with the `Router` section highlighted by a red box.

Router

A router object is an isolated instance of middleware and routes. You can think of it as a "mini-application," capable only of performing middleware and routing functions. Every Express application has a built-in app router.

A router behaves like middleware itself, so you can use it as an argument to `app.use()` or as the argument to another router's `use()` method. The top-level express object has a `Router()` function that creates a new router object.

Router([options])

Create a new router as follows:

```
var router = express.Router([options]);
```

The optional `options` parameter specifies the behavior of the router.

Property	Description	Default	Availability
<code>caseSensitive</code>	Enable case sensitivity.	Disabled by default, treating <code>"/foo"</code> and <code>"/foo/</code> as the same.	
<code>mergeParams</code>	Preserve the <code>req.params</code> values from the parent router. If the parent and the child have conflicting param names, the child's value take precedence.	<code>false</code>	4.5.0+
<code>strict</code>	Enable strict routing.	Disabled by default, <code>"/foo"</code> and <code>"/foo/</code> are treated the same by the router.	

You can add middleware and HTTP method routes (such as `get`, `put`, `post`, and so on) to router just like an application.

```
// invoked for any requests passed to this router
router.use(function(req, res, next) {
  // .. some logic here .. like any other middleware
  next();
});

// will handle any request that ends in /events
```

Router Methods

- `router.all()`
- `router.METHOD()`
- `router.param()`
- `router.route()`
- `router.use()`

Voorbeeld Routing

```
// routes.js - middleware via express.Router()
```

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

```
var router = express.Router();
```

```
// Dit mag ook in één regel:
```

```
// var router = require('express').Router();
```

```
// middleware specifiek voor deze router - merk op dat geen 'app' nodig is.
```

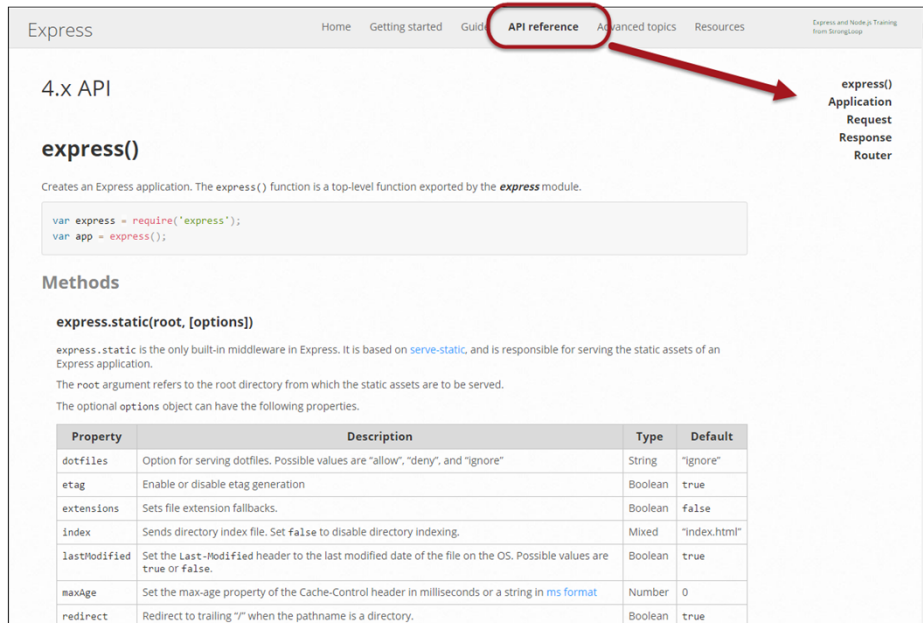
```
router.use(function(req, res, next){  
    console.log('Router aangeroepen: ', new Date());  
    next();  
});
```

```
// homepage voor deze router.
```

```
router.get('/', function(req, res){  
    res.send('Homepage van de router');  
});
```

Meer over Express

- Lees online documentatie
- Vijf hoofdcategorieën
- expressjs.com/en/4x/api.html



Express

Home Getting started Guide **API reference** Advanced topics Resources Express and Node.js Training from StrongLoop

4.x API

express()

Creates an Express application. The `express()` function is a top-level function exported by the **express** module.

```
var express = require('express');
var app = express();
```

Methods

express.static(root, [options])

`express.static` is the only built-in middleware in Express. It is based on `serve-static`, and is responsible for serving the static assets of an Express application.

The `root` argument refers to the root directory from which the static assets are to be served.

The optional `options` object can have the following properties.

Property	Description	Type	Default
<code>dotfiles</code>	Option for serving dotfiles. Possible values are "allow", "deny", and "ignore"	String	"ignore"
<code>etag</code>	Enable or disable etag generation	Boolean	true
<code>extensions</code>	Sets file extension fallbacks.	Boolean	false
<code>index</code>	Sends directory index file. Set <code>false</code> to disable directory indexing.	Mixed	"index.html"
<code>lastModified</code>	Set the Last-Modified header to the last modified date of the file on the OS. Possible values are <code>true</code> or <code>false</code> .	Boolean	true
<code>maxAge</code>	Set the max-age property of the Cache-Control header in milliseconds or a string in ms format	Number	0
<code>redirect</code>	Redirect to trailing "/" when the pathname is a directory.	Boolean	true

Checkpoint

- Routing abstraheert de routes in je app ten opzichte van de functionaliteit
- Houdt `app.js` / `server.js` zo compact mogelijk
- Lees verdere online documentatie

Oefening...

