Node.js og Express

Node.js

- Node.js
  - er designet til skalerbare webapplikationer med mange I/O operationer
  - er en asynkron, eventstyret JavaScript runtime
  - events udføres en ad gangen i en enkelt tråd, så en langvarig event handler blokerer for, at andre events håndteres
  - understøtter asynkron (og synkron) I/O
  - har ikke et DOM API

https://nodejs.org/en/

Modul system

- JavaScript har et modul system kaldet ES modules, som Node.js nu understøtter fra version 13.9.0
- Men Node.js har fra starten haft sit eget modul system baseret på Common JS, hvor et modul kan eksportere på to måder – og importeres med require(…)

```
// modul1.js
module.exports = { x: 1 };
```

```
// modul2.js
exports.p = { y: 2 };
exports.q = { z: 3 };
```

```
// modulApp.js
const o1 = require(__dirname + '/modul1');
console.log(o1.x); // => 1

const {x} = require(__dirname + '/modul1');
console.log(x); // => 1

const o2 = require(__dirname + '/modul2');
console.log(o2.p.y); // => 2
console.log(o2.q.z); // => 3

const {p, q} = require(__dirname + '/modul2');
console.log(p.y); // => 2
console.log(q.z); // => 3
```

Indbyggede moduler i Node.js

- http – HTTP server og klient
- https – HTTPS server og klient
- fs/promises – fil system
- …

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

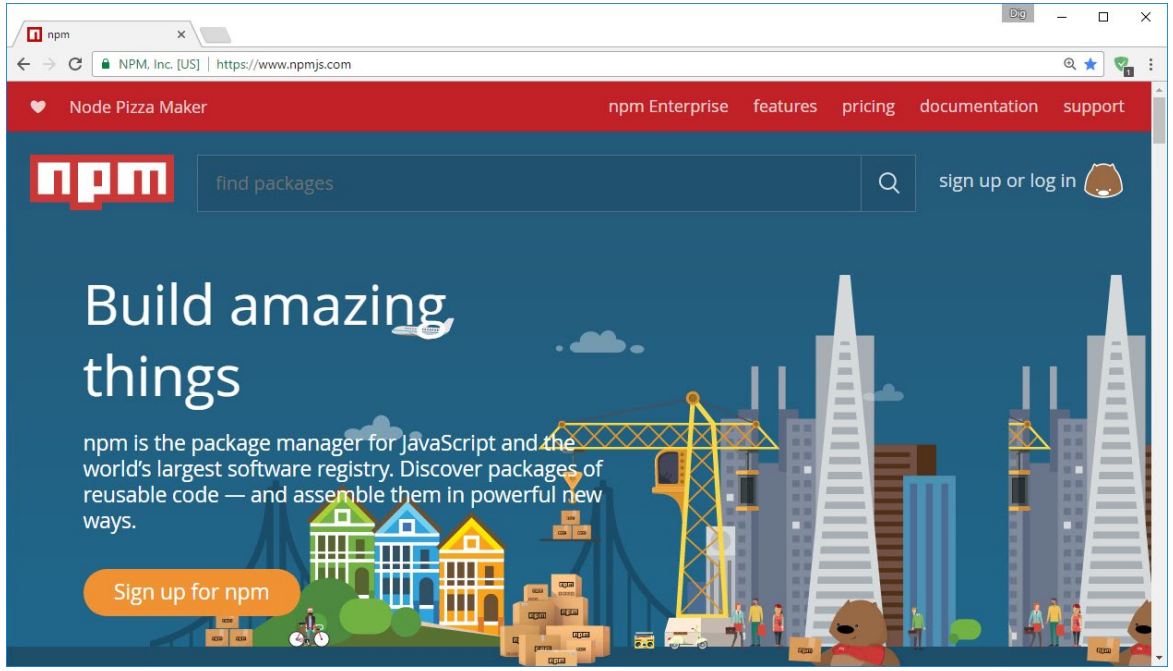
```
// createServer.js
const http = require('http');

http.createServer((request, response) => {
    let array = [request.method, request.url];
    response.writeHead(200, {"Content-Type": "application/json"}); // OK
    response.write(JSON.stringify(array));
    response.end();
}).listen(8080);

console.log('Lytter på port 8080 ...');
```

```
// filServer.js
const http = require('http');
const fs = require('fs').promises;

function genererLinks(filnavne) {
    let html = '';
    for (let filnavn of filnavne) {
        html += '<a href="' + filnavn + '">' + filnavn + '</a><br>\n';
    }
    return html;
}
```

```
http.createServer(async (request, response) => {
    if (request.url === '/') {
        let filnavne = await fs.readdir(__dirname + '/filer');
        let html = genererLinks(filnavne);
        response.writeHead(200, {"Content-Type": "text/html"}); // OK
        response.write(html);
    } else {
        try {
            let sti = __dirname + '/filer' + request.url;
            let filData = await fs.readFile(sti);
            response.writeHead(200); // OK
            response.write(filData);
        } catch (e) {
            response.writeHead(404); // Not Found
            response.write(e.name + ": " + e.message);
        }
    }
    response.end();
}).listen(8080);

console.log('Lytter på port 8080 ...');
```

npm

- npm er en package manager for Node.js
- Bliver installeret sammen med Node.js
- Bruges til download af eksterne moduler
- Hvis et modul anvender andre moduler, installers de automatisk



Installation af moduler

- Et eksternt modul installeres med kommandoen npm i <modul-navn>
- npm download'er og installerer nyeste version af modulet
- Installerede moduler er i mappen node_modules i projektet
- Hvis der er en package.json manifest fil i projektet med { }, tilføjes modulet som en afhængighed i denne fil
- npm anvender semantic versioning, hvor fx
  - ^1.2.3 betyder 1.2.3 <= … < 2.0.0, dvs. ingen inkompatible API ændringer
- Med kommandoen npm i installeres alle moduler nævnt i package.json

Eksterne moduler

- node-fetch
- express
- mongoose
- moment
- nodemailer
- …

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

package.json

- Kommandoen npm i node-fetch installerer modulet i mappen node_modules
- Desuden opdateres filen package.json med modulets navn og version

```
{
  "dependencies": {
    "node-fetch": "^2.6.1"
  }
}
```

```
// deleteTest.js
const fetch = require('node-fetch');

async function deLete(url) {
    let respons = await fetch(url, {
        method: "DELETE"
    });
    if (respons.status !== 200) // OK
        throw new Error(respons.status);
    return await respons.json();
}

async function main(url) {
    try {
        let objekt = await deLete(url);
        console.log(objekt);
    } catch (fejl) {
        console.log(fejl);
    }
}
main('http://localhost:8080/test');
```

```
// userServer.js
const http = require('http');
const fetch = require('node-fetch');
const userUrl = 'https://jsonplaceholder.typicode.com/users';

async function get(url) {
    const respons = await fetch(url);
    if (respons.status !== 200) // OK
        throw new Error(respons.status);
    return await respons.json();
}

function genererTabel(users) {
    let html = '<table>';
    for (user of users) {
        html += '<tr><td>' + user.id + '</td><td>' + user.name +
                '</td><td>' + user.company.name + '</td></tr>\n';
    }
    html += '</table>';
    return html;
}
```

```
http.createServer(async (request, response) => {
    if (request.method === 'GET') {
        try {
            let users = await get(userUrl);
            let html = genererTabel(users);
            response.writeHead(200, { "Content-Type": "text/html" }); // OK
            response.write(html);
        } catch (fejl) {
            if (typeof fejl.message === 'number')
                response.writeHead(fejl.message);
            else
                response.writeHead(400); // Bad Request
            response.write(fejl.name + ": " + fejl.message);
        }
    }
    response.end();
}).listen(8080);

console.log('Lytter på port 8080 ...');
```

express - npm

npmjs.com/package/express

express

Fast, unopinionated, minimalist web framework for node.

npm v4.17.1 | downloads 57M/month | linux passing | windows passing | coverage 100%

```
const express = require('express')
const app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

Install

```
> npm i express
```

Weekly Downloads

13.882.650

Version	License
4.17.1	MIT

Unpacked Size	Total Files
208 kB	16

Issues	Pull Requests
97	52

Homepage

expressjs.com/

Express.js

- Express.js er et web applikations framework bygget oven på Node.js
- Installeres med npm i express
- Dens routing gør det let at koble http requests til request handlers:
  - get(sti, (request, response) => {…})
  - post(sti, (request, response) => {…})
  - put(sti, (request, response) => {…})
  - delete(sti, (request, response) => {…})
  - all(sti, (request, response) => {…})
  - …
- En sti kan indeholde parametre

```
// expressServer.js
const express = require('express');
const app = express();

app.all('/', (request, response) => {
    let array = [request.method, request.url];
    response.send(array);
});

app.get('/fil/:navn', (request, response) => {
    let array = [request.method, request.url, request.params.navn];
    response.send(array);
});

app.listen(8080);

console.log('Lytter på port 8080 ...');
```

Repsonse metoder

- send(tekst|HTML|objekt|array)
  - sender et HTTP response som tekst, HTML eller JSON
- sendStatus(kode)
  - sender den givne HTTP status kode sammen med den tilsvarende status tekst
- sendFile(sti)
  - sender filen med den specificerede sti og sætter content-typen ud fra filens extension
- status(kode)
  - sætter den givne HTTP status kode på response
- redirect(sti)
  - Omdirigerer til en URL, der genereres ud fra den specificerede sti
- render(sti, data)
  - Genererer HTML ud fra en template og data
- …

Middleware

- Express.js er baseret på middleware funktioner
  - behandler og/eller omformer et request, før request'et håndteres
    – fx autentifikation, validering, logning, parsing, …
  - har signaturen function(request, response, next)
  - skal enten kalde next() eller generere et response
  - next() videregiver request'et til den næste middleware funktion eller request handler
- En request handler er en middleware funktion, der afslutter et request ved at sende et response til klienten
- Middleware funktioner monteres på en sti med
  - use([sti], middleware-funktion)
  - hvis ingen sti angives, udeføres middleware-function for alle request

```
// middleware.js
const express = require('express');
const app = express();

function log(request, response, next) {
    console.log(request.method + ': ' + request.url);
    next();
}

app.use(log);

app.get('/', (request, response) => {
    response.sendStatus(200); // OK
});

app.listen(8080);

console.log('Lytter på port 8080 ...');
```

Middleware

- express.static – indbygget
  - returnerer static filer
- express.json – indbygget
  - parser request.body
- cors → anden middleware installers med npm
  - aktiverer CORS
- express-session
  - håndterer session
- …

```
// static.js
const express = require('express');
const app = express();

app.use(express.static(__dirname + '/filer'));

app.get('/*', function (request, response) {
    response.sendStatus(404); // Not Found
});

app.listen(8080);

console.log('Lytter på port 8080 ...');
```

```
// body.js
const express = require('express');
const app = express();

app.use(express.json());

app.post('/', (request, response) => {
    response.status(201).send(request.body); // Created
    console.log(request.body.navn);
});

app.listen(8080);

console.log('Lytter på port 8080 ...');
```

```
// postNavn.js
const fetch = require('node-fetch');

async function post(url, objekt) {
    const respons = await fetch(url, {
        method: "POST",
        body: JSON.stringify(objekt),
        headers: { 'Content-Type': 'application/json' }
    });
    if (respons.status !== 201) // Created
        throw new Error(respons.status);
    return await respons.json();
}

async function main(url, objekt) {
    try {
        let respons = await post(url, objekt);
        console.log(respons);
    } catch (fejl) {
        console.log(fejl);
    }
}
main('http://localhost:8080', {navn: 'Viggo'});
```

SOP og CORS

- Same-Origin Policy (SOP) er en generel sikkerhedsmekanisme, der forhindrer, at en webside via JavaScript kan få adgang til ressourcer på en fremmed server
- Cross-Origin Resource Sharing (CORS) er en mekanisme til at give selektiv adgang til ressourcer på en server

https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

cors

CORS is a node.js package for providing a **Connect**/**Express** middleware that can be used to enable **CORS** with various options.

Follow me (@troygoode) on Twitter!

Install

```
> npm i cors
```

Weekly Downloads

3.890.569

Version	License
2.8.5	MIT

Unpacked Size	Total Files
20 kB	6

Issues	Pull Requests
6	0

Homepage

github.com/expressjs/cors#readme

Repository

github.com/expressjs/cors

Simple Usage (Enable *All* CORS Requests)

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

app.use(cors())

app.get('/products/:id', function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for all origins!'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```