peatutor.com/express

Noden avulla pystyy tekemään backendiä Javascriptillä

Esimerkki Javascriptilllä

let a=5;

let b=11;

let c=a+b;

console.log(“summa on “´+c);

Terminaalissa node eka.js

VSCodessa Terminal. Voidaan ajaa debuggerilla.

Joudutaan Noden kanssa käyttämään asynkronisia funktioita. Onko säie tai thread tuttu?

Apachen veppipalvelussa jos olisi pankkiautomaatilla monta käyttäjää, se antaisi jokaiselle oman säikeen. Sen sijaan Node työskentelee yhdellä säikeellä.

Jos tehdään login synkronisella funktiolla, koko sovellus pysähtyy eikä etene ennen kuin saa vastauksen onko login onnistunut vai ei. Vaikka se kestää millisekunteja niin jos on tuhansia käyttäjiä niin se kestää. -> Ei käytetä

Käytetään asynkronista siirtoa. Kun automaatilta lähetetään http-pyyntö asynkronisella siirrolla. Vastaus kun tulee niin homma etenee -> Joudutaan käyttämään CAllbackeja.

Esimerkki osoitteessa peatutor.com/express/javascript.php

What is Callback?

Callback is an asynchronous equivalent for a function. A callback function is called at the completion of a given task. Node makes heavy use of callbacks. All the APIs of Node are written in such a way that they support callbacks.

For example, a function to read a file may start reading file and return the control to the execution environment immediately so that the next instruction can be executed. Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as a parameter. So there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process a high number of requests without waiting for any function to return results.

Blocking Code Example

Create a text file named **input.txt** with the following content −

Tutorials Point is giving self learning content

to teach the world in simple and easy way!!!!!

Create a js file named **main.js** with the following code −

var fs = require("fs");

var data = fs.readFileSync('input.txt');

console.log(data.toString());

console.log("Program Ended");

Now run the main.js to see the result −

$ node main.js

Verify the Output.

Tutorials Point is giving self learning content

to teach the world in simple and easy way!!!!!

Program Ended

Non-Blocking Code Example

Create a text file named input.txt with the following content.

Tutorials Point is giving self learning content

to teach the world in simple and easy way!!!!!

Update main.js to have the following code −

var fs = require("fs");

fs.readFile('input.txt', function (err, data) {

if (err) return console.error(err);

console.log(data.toString());

});

console.log("Program Ended");

Now run the main.js to see the result −

$ node main.js

Verify the Output.

Program Ended

Tutorials Point is giving self learning content

to teach the world in simple and easy way!!!!!

These two examples explain the concept of blocking and non-blocking calls.

* The first example shows that the program blocks until it reads the file and then only it proceeds to end the program.
* The second example shows that the program does not wait for file reading and proceeds to print "Program Ended" and at the same time, the program without blocking continues reading the file.

Thus, a blocking program executes very much in sequence. From the programming point of view, it is easier to implement the logic but non-blocking programs do not execute in sequence. In case a program needs to use any data to be processed, it should be kept within the same block to make it sequential execution.

**Callback on funktio, joka voidaan antaa toiselle funktiolle argumenttina. Qt:ssa tulee sama homma. Lähetetään asynkronisia http-pyyntöjä. Qt:ssa voisi käyttää callbackeja, mutta sinne on tehty erillinen SIgnal-SLot-systeemi.**

**On siirrytty PHP:stä Nodeen. PHP:tä ei ole opetettu muualla meillä enää. Node on tosi suosittu. PHP:tä on prosentuaalisesti hirveästi käytössä.**

**Express.js on Nodejs:ään tehty framework, joka helpottaa http-sovellusten käyttöä. Sanotaan Web-frameworkiksi. Esimerkissä on käytetty apuohjelmaa nimeltä Express-generator. Se generoio sovellukselle rungon, pääsee nopeammin eteenpäin.**

**SEURAAVAKSI GITHUB:**

**Jos haluaa niin voi ruveta tekemään samaa koodia.**

**git clone jokurepo**

**Kannattaa tehdä uusi branch, ei mainiin pusketa suoraan.**

**git checkout -b dev**

**Express-asennus**

**Jokonkin sopivaan hakemistoon Expressin asennus peatutor.com/express/Examples/express.php**

**Sieltä asennukset vanhaan malliin.**

**CORS = Cross ORigin resource sharing. REST-API:lla on oma IP-osoite. Clientti eli Qt-sovellus on mahdollisesti eri koneella, jolloin sillä on eri IP-osoite. Se ei suostu vastaamaan jos ei ole CORSIA.**

**Helmet parantaa http-juttujen turvallisuutta.**

**Express-generator kirjoittaa package.json tiedostoon. Nyt kun katsotaan em. tiedostoa.**

**npm install lukee tätä tiedostoa ja ne menee node\_modulesiin. Jos node\_modulesin hävittää, ei npm start toimi, mutta kun ajaa npm instal, se lukee package.jsonin ja asentaa tarvittavat systeemit.**

**.gitignore-tiedostoon node\_modules/**

**Seuraavaksi demo Pull Requestista. checkout -b dev, sinne joku tiedosto tee vaikka**

**ESIM: git add remote myRepo https://github.com/jjauhiai/testi.git**

git push -u myRepo dev

Pyritään pitämään main puhtaana, että sinne mitä vaan bugista koodia jonka muut kloonaa.

Kavereita kutsutaan näin: Mennään repoon, Settings, Collaborators

npm start kun katsotaan package.jsonin sisältö, se viittaa bin/www-hakemistoon. Sieltä www sisältää mm.

var port = normalizePort(process.env.PORT || '3000');

app.set('port', port);

Eli portti joko luetaan ympäristömuuttujasta TAI se on 3000…

Nyt ei siis ole ympäristömuuttujaa, joten käytetään 3000. Ympäristömuuttujat voidaan määritellä tietokannan yhteydessä tai ne saadaan pilvestä esim Heroku tai Azure.

app.js

Routes-kansiosta löytyy index.js ja users.js

app.js:ssä

var indexRouter = require('./routes/index');

vastaa

app.use('/', indexRouter);

ja

var usersRouter = require('./routes/users');

app.use('/users', usersRouter);

Pekka tekee tässä välissä uuden branchin examples

Ajetaan localhost:3000 ja locahost:3000/users

get on yksi http-metodeista. get put post delete. Lisätään users.js ennen module.exports=router;

router.get('/toka', function(req, res) {

  console.log("Olen toka :D !");

  res.send('Tokapa hyvinkin...')

});

Nextiä ei tässä tarvita. Kirjoittaa kaikki käsin.

<http://localhost:3000/users/toka>

router.get('/kolmas/:name', function(req, res) {

  console.log(req.params.name);

  res.send('Terve '+req.params.name);

});

router.get('/neljas/:fname/:lname', function(req, res) {

  //console.log(req.params.name);

  res.send('Terve '+req.params.fname +" "+req.params.lname);

});

**Parametrit:**

http-pyynto lähetetään (request tai req)

http-vastaus tulee paluupostissa (response tai res)

GET-metodissa data menee urlin perässä. http Header-osassa

POST-metodissa menee Body-osassa.

**Katsotaan esimerkkinä POST:**

router.post('/',function(request,response){

        response.send(request.body);

        console.log(request.body);

    }

);

Tätähän ei voi selaimen kautta kutsua. Tarvitaan joku työkalu.

Postman ! Komentoriviltä CURL

Collections -> New Collection -> vaikka nimeksi TVT21SPL -> Add Request -> POST eka

Body -> x-www-form-urlencoded

**MIDDLEWARE**

Funktio on sellainen, että sitä ei kutsuta requestilla, mutta ne suoritetaan aina kun kutsutaan requestia, joka on sen jälkeen.

**Users.js**

|  |
| --- |
| var express = require('express'); |
|  |

|  |
| --- |
| var router = express.Router(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\* GET users listing. \*/ |
|  |

|  |
| --- |
| router.get('/', function(req, res) { |
|  |

|  |
| --- |
| res.send('Hello'); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| }); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| router.get('/toka', function(req, res) { |
|  |

|  |
| --- |
| console.log("Olen toka :D !"); |
|  |

|  |
| --- |
| res.send('Tokapa hyvinkin...') |
|  |

|  |
| --- |
| }); |
|  |

|  |
| --- |
| router.get('/kolmas/:name', function(req, res) { |
|  |

|  |
| --- |
| console.log("Syötettiin nimi "+req.params.name); |
|  |

|  |
| --- |
| res.send('Terve '+req.params.name); |
|  |

|  |
| --- |
| }); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| **router.use(** |
|  |

|  |
| --- |
| **function(req,res,next){** |
|  |

|  |
| --- |
| **console.log('The common middleware 1 called');** |
|  |

|  |
| --- |
| **next();** |
|  |

|  |
| --- |
| **}** |
|  |

|  |
| --- |
| **)** |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| **router.use(** |
|  |

|  |
| --- |
| **function(req,res,next){** |
|  |

|  |
| --- |
| **console.log('The common middleware 1b called');** |
|  |

|  |
| --- |
| **next();** |
|  |

|  |
| --- |
| **}** |
|  |

|  |
| --- |
| **)** |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| router.get('/neljas/:fname/:lname', function(req, res) { |
|  |

|  |
| --- |
| //console.log(req.params.name); |
|  |

|  |
| --- |
| res.send('Terve '+req.params.fname +" "+req.params.lname); |
|  |

|  |
| --- |
| console.log('Terve '+req.params.fname +" "+req.params.lname); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| }); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| **router.use(** |
|  |

|  |
| --- |
| **function(req,res,next){** |
|  |

|  |
| --- |
| **console.log('The common middleware 2 called');** |
|  |

|  |
| --- |
| **next();** |
|  |

|  |
| --- |
| **}** |
|  |

|  |
| --- |
| **);** |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| router.post('/',function(request,response){ |
|  |

|  |
| --- |
| response.send(request.body.lname+" "+request.body.fname); |
|  |

|  |
| --- |
| console.log(request.body.lname+" "+request.body.fname); |
|  |

|  |
| --- |
| }); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| router.put ('/:id',function(request,response){ |
|  |

|  |
| --- |
| response.send("Henkilölle "+request.params.id+" muutetaan nimeksi "+request.body.fname); |
|  |

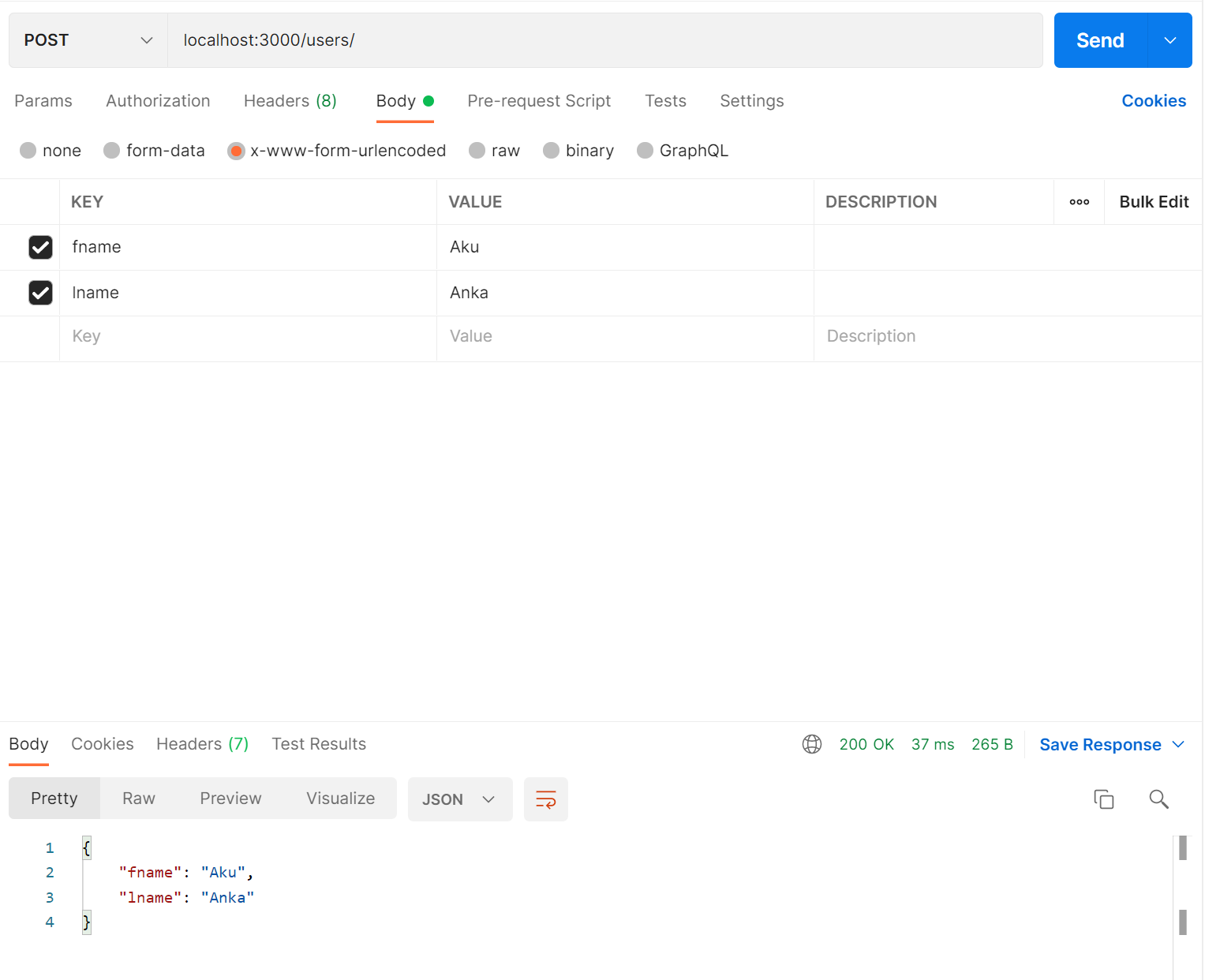
|  |
| --- |
| console.log("Henkilölle "+request.params.id+" muutetaan nimeksi "+request.body.fname); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| }); |
|  |

.GET sijaan .USE

Javascriptissä puolipiste ei ole pakollinen, Kokeile mitä tapahtuu jos ei laite nextiä loppuun (jää jumiin).



Paluupostissa saadaan JSON-objekti.

app.js sisältää:

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

Voidaan lähettää muodossa urlencoded (vrt. Postman).

router.post('/',function(request,response){

  response.send(request.body.lname+" "+request.body.fname);

  console.log(request.body.lname+" "+request.body.fname);

}

);

Nyt kenttien nimien on oltava lname ja fname !

**PUT**

Putissa on parametreja sekä headerissa että bodyssa.

router.put ('/:id',function(request,response){

  response.send("Henkilölle "+request.params.id+" muutetaan nimeksi "+request.body.fname);

  console.log("Henkilölle "+request.params.id+" muutetaan nimeksi "+request.body.fname);

});

