Nodejs :

Intro till node js

<https://nodejs.org/en/about/>

|  |
| --- |
| JavaScript är en Hosted språk:  Man kan köra js på olika plattform. Det enda man behöver är en tolkare/översättare.  Nodejs kör i serversidan med hjälp av V8 engine(ägs av google) och har apier som kan manipulera server filer, databaser osv. |

Första program med node js:

|  |  |
| --- | --- |
| Steg 1: Skapa en file med namnet app.js  Steg 2: I filen skriv:   |  | | --- | | let namn="ditt namn"  console.log("Välkommen " + namn + ". Hej från server.")  8 |   Steg 3: Gå till terminalen och kör ( node sen filnamn):  **Node app.js** |

Skriva på en fil :

|  |
| --- |
| const fs = require("fs");  //skriver på en fil  fs.writeFile("text.txt", " Hej hej ", (err)=>{  console.log(err)  })  //writeFileSync run synchronously  samma kod som ovan med kommentar:  //skriver på en fil  //första arguement filnamn  //andra vad skulle du skriva i filen  //tredje en callback som fångar error.  fs.writeFile("text.txt", " 1000............. ", (err)=>{  console.log(err)  }) |

Läsa från fil :

|  |
| --- |
| fs.readFile("text.txt", (err, data)=>{  err? console.log(err):  console.log(data.toString());  })  Prova utan toString() metoden |

Skapa en server :

|  |
| --- |
| const http = require("http");  const server = http.createServer((req, res)=>{  res.write("hello from server");  res.end() //stänger kopplingen och skickar den  })  //lyssnar på port 8000  server.listen(8000) |

Kör filen och öppna i browser:

<http://localhost:8000/>

Diskussion:

\*\*Inspektera sidan

Vilken typ av content/innehåll är :

Hello from server.

Varför?

Övningar:

* Skapa en fil i din working directory och skriva “Välkommen ”
* Skapa en server och skicka ditt namn till ex Max med h1 element till port 8000
* Läs data från filen (den filen du skapade tidigare) och skicka/visa data i port 8000. Det ska visas i browser : “Välkommen ´ditt namn´ ”

Node js:

Definition, glosor osv.

|  |  |  |
| --- | --- | --- |
| “As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications.In a node js app many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.”  enligt official docs: <https://nodejs.org/en/about/>  I node jskod exekveras utan blockera nästa task(non-blocking).  Single thread som hanterar flera request.    Asynchronous   |  |  | | --- | --- | | |  | | --- | | synchronous kod //exekveras rad efter rad  synchronous kod  synchronous kod  **asynchronous kod(skickar req till server, väntar på respons)**  synchronous kod  synchronous kod  **respons från server**  synchronous kod  synchronous kod |   synchronous kod: exekveras direkt och rad efter rad  asynchronous kod: börjar exekvera och väntar tills den får nån respons (ofta från server / user event) sen exekveras helt.  Respons brukar hanteras av en funktion som kallas för callback funktion. |   Event-driven  JavaScript Runtime  Scalable  Network applications  Concurrently  Callback  Fördel med : “but if there is no work to be done, Node.js will sleep” ? |

----------------------- ----------------------- ----------------------------------------------------

asynchronous programming/callbacks:

<https://flaviocopes.com/javascript-callbacks/>

Node js i bakgrunden:

|  |
| --- |
| Kör node js → js engine registrerar variabler, funktioner och exekverar kod → Event loop har koll på alla callbacks som är redo med sina resultaten så att de kan exekveras. |

Läs mer : <https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/>

HTTP:

**HTTP(s)**: Hypertext Transfer Protocol : ett webb/applikation kommunikationsprotokoll där applikationer kommunicerar med varandra.

**Hur fungerar detta?**

|  |
| --- |
| - > Klient Öppnar koppling till server  -> Klient skickar request till server //HTTP request  -> Server skickar respons till Klient  (Ibland med **data/resurs** ibland med error meddelande) //HTTP response  -> Klient stänger koppling. |

**Struktur/Vad finns in i en HTTP request ?:**

**Metod URI String, Query string HTTP versionen**

**GET /index.html?namn:“rakib” http/2.00**

i) HTTP Method: GET, POST

HTTP:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

ii)

* URI String till ex. index.html,
* Query String : Namn och värde par som sepereras med &

ex. ?namn:“rakib”

iii) HTTP versionen till ex. HTTP 2.

\* Request header : <https://developer.mozilla.org/en-US/docs/Glossary/Request_header>

\* Request body

Läs mer : <https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages>

**HTTP Response:**

**http/2.0 200 ok**

i) HTTP versionen

ii) Statuskod

iii) status meddelandet

**Man brukar se de här statuskod jätteofta :**

**200 : betyder OK**

**404 : Not Found**

**401: Unauthorised**

**403 : Forbidden**

**500: server error**

**Olika Statuskod:**

[**https://developer.mozilla.org/sv-SE/docs/Web/HTTP/Status**](https://developer.mozilla.org/sv-SE/docs/Web/HTTP/Status)

Request , response :

Request objects : kommer från client

|  |
| --- |
| var http = require('http');  http.createServer(function (req, res) {  console.log(req);  }).listen(8000);  console.log('listening on localhost:8000');  Gå till localhost:8000  Sen kolla på node konsolen. |

var http = require('http');

http.createServer(function (req, res) {

console.log(req.url, req.method, req.headers );

}).listen(8000);

console.log('listening on localhost:8000');

Läs mer om HTTP headers: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

Respons:

|  |
| --- |
| const http = require("http");  const server = http.createServer((req, res)=>{  res.write("hello from server");  res.end() //stänger kopplingen och skickar den  })  //lyssnar på port 8000  server.listen(8000) |

Node modules:

console.log(global)

console.log(module)

Läs mer om modules:

<https://nodejs.org/api/modules.html>

Skapa en till fil med namnet auth.js:

const object ={obj: "just an object"}

function authenticateUser(user){

console.log(user + " har loggat in")

}

//exporting functions, var, obj etc so that other files can use it

module.exports = authenticateUser;

//alternative way

//module.exports.authenticateUser = authenticateUser;

//module.exports.object = object

Importera functions/var/object :

const auth = require("./auth")

auth("rakib")

Object destructuring:

const user = { name: "rakib", username:"något", email:"r@g.se"};

//destructuring av objekt

const {name, username, email} = user;

console.log(name);

Alternative sätt:

function authenticateUser(user){

console.log(user + " har loggat in")

}

const object = {name:"rakib"}

//exporting functions, var, obj etc so that other files can use it

//alternative way

module.exports.authenticateUser = authenticateUser;

module.exports.object = object

const {authenticateUser, object} = require("./authenticateUser");

Övning:

* Anropa authenticateUser med object.name som arguement.

Extra:

Bakgrunden:

## **“The module wrapper**

Before a module's code is executed, Node.js will wrap it with a function wrapper that looks like the following:

(function(exports, require, module, \_\_filename, \_\_dirname) {

// Module code actually lives in here

})();

By doing this, Node.js achieves a few things:

* It keeps top-level variables (defined with var, const or let) scoped to the module rather than the global object.
* It helps to provide some global-looking variables that are actually specific to the module, such as:  
  + The module and exports objects that the implementor can use to export values from the module.
  + The convenience variables \_\_filename and \_\_dirname, containing the module's absolute filename and directory path.”

Läs mer om module pattern js: <https://coryrylan.com/blog/javascript-module-pattern-basics>

Övning 2:

Modules:

* Skapa ett user objekt vilket kommer att ha namn, email egenskaper .
* Exportera objektet så att det går att använda till main js modulen.
* Visa/skriva ut alla värdet från objektet.

Övning 3:

Modules:

* Skapa en funktion som kan läsa från directory fil.
* Exportera funktionen och använd den i main js modulen.
* Anropa funktionen så att den kan läsa från directory filen (funktionen som du har skapat tidigare)

Refs:

<https://nodejs.org/api/modules.html>

os modules:

const os = require("os");

console.log("free memory: " +os.freemem());

console.log("total memory: " +os.totalmem());

console.log("os plattform " + os.platform());

console.log("type of os: " +os.type());

console.log("os uptime : " +os.uptime())

console.log( "user info : "+ os.userInfo().username);

console.log("temporary directory :"+ os.tmpdir());

console.log(" hostname till os :" + os.hostname());

Node modules:

<https://nodejs.org/api/modules.html>

-----------------------------------------------

Lösning förslag av övningarna :

File system lösningen:

var fs = require('fs'),

http = require('http');

var namn = "Max";

fs.writeFileSync("./index.html", `<h1> Välkommen ${namn} </h1>`)

http.createServer(function (req, res) {

var path = \_\_dirname + "/index.html";

console.log(path);

fs.readFile(path, function(err, data) {

// res.writeHead(200);

res.end(data);

});

}).listen(8000);

console.log('listening on localhost:8000');

Träna mer :

<https://www.freecodecamp.org/learn>

Restful api :

**Representational State Transfer api.**

**video:** [**https://www.youtube.com/watch?v=GZvSYJDk-us&t=65s**](https://www.youtube.com/watch?v=GZvSYJDk-us&t=65s)

**Extra : (extra extra)**

**--------------------------------------------- Extra ------------------------------------------------------**

Event emitter: <https://nodejs.org/api/events.html>

|  |
| --- |
| const EventEmitter = require("events")  const event = new EventEmitter();  //register an event  event.on("loggedIn", ()=>{  console.log("loggedIn event is called")  })  //Raise/emit an event  event.emit("loggedIn") |

En till ex.

const EventEmitter = require("events")

const event = new EventEmitter();

//registeras event : logIn

event.on("logIn", (e)=>{

console.log("loggedIn event is called on "+ e.username)

})

function triggeringEvent() {

//Raise/emit an event

event.emit("logIn", {username:"rakib", id:01})

}

// till ex. låter användare anropa triggeringEvent

triggeringEvent();

Event emitter med extra data:

const EventListner = require("events")

const event = new EventListner();

//register an event

//eventargs === objektet av loggedIn event

event.on("loggedIn", (e)=>{

console.log("loggedIn event is called on "+ e.username)

})

//Raise/emit an event

event.emit("loggedIn", {username:"rakib", id:01})

Till ex vi har en biljett registrering system där vi vill notifiera användare och mejla användare när det emitter en event:

const EventEmitter = require("events")

const TicketEvent = new EventEmitter();

//använda event för att mejla användare

TicketEvent.on('ticketInserted', function(ticket) {

emailUser(ticket);

});

//använda event för att notifiera användare i slack

TicketEvent.on('ticketInserted', function(ticket) {

notifyMobil(ticket);

});

addTicket("SJ", ()=> console.log("SJ ticket are registered"))

function emailUser(t) {

console.log("Mailing user " + t + " has added")

}

function notifyMobil(t){

console.log("notifying to slack" + t + " has added")

}

function addTicket(ticket, callback) {

TicketEvent.emit('ticketInserted', ticket);

callback();

}

Övning :

Skapa en en event för subscribe event dvs när subscribe event händer kommer att visa i konsolen : User X is subscribed .