Internetprogrammering
DD1386

Föreläsning 7

Innehåll

- Egen modul
- HTTP Server exempel
- Express
- sqlite
- modulen path, __dirname

Egen modul

- Egna moduler skapar man genom att skriva koden som ska användas som modul i en fil med ändelsen js. Och se till att exportera funktioner och attribut som ska kunna användas av de program.

```
type = 'Truck';
exports.brand = 'Volvo';
exports.drive = ()=> console.log('I am driving a '+exports.brand+'
brand '+type)
exports.velo = function(velocity) {
    if (velocity>140)
   console.log('I am driving too fast.');
    else
   console.log('I dont drive fast.');
}
```

Använd modulen

```
let myCar = require('./car.js');
console.log('Brand: ' + myCar.brand);
myCar.drive();
```

```
let {velo} = require('./car.js');
velo(189);
velo(120);
```

Nodejs http-server exempel

```
//file: serverhttp.js
const {createServer} = require('http');
let server = createServer(
   (request, response) => {
      response.writeHead(200,
                {'Content-Type':'text/html'});
      response.write(`<h1>Du kör Nodejs!!! </h1>
            <p>urln:n i förfrågan:
            <code>${request.url}</code></p>`);
     response.end();
});
server.listen(8000);
console.log("Listening! (port 8000)");
```

Förklaring av koden

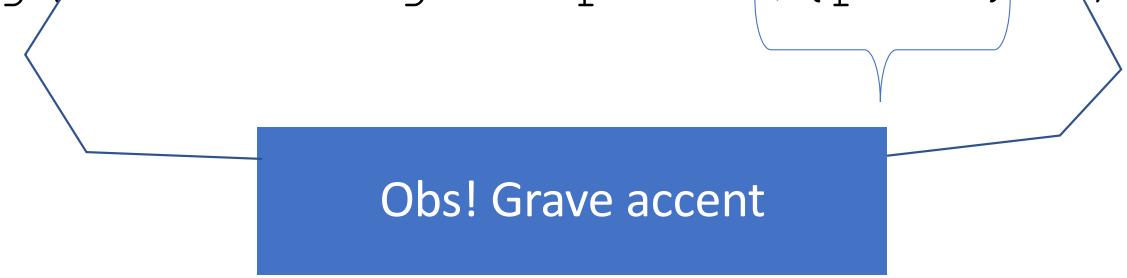
- `require`: importerar moduler, t.ex `require('http');`
- http modulen består av en del funktioner och variabler
  `const {createServer} = require('http');`
  är kompakt version av:
  `let http_modulen= require('http');`
  `let createServer = http_modulen.createServer;`
- `(request, response) => {…}`:är en anonym funktion som anropas för varje inkommande http-request
- `server.listen(8000)` börjar lyssna på port
- `console.log('Listening…')` skriver ut ett textmeddelande

Expressjs

- Express är en framework baserad på Nodejs, modulerna `'http'` och `'connect'`.
- Express höjer Ejs Ejs Ejs Ejs Ejs abstraktionen och underlättar följande:
  - Parsning av HTTP-request
  - Parsning av kakor
  - Hantera sessioner
  - Organisering av rutter med hjälp av if-satser baserat på URL och HTTP metoder i HTTP-requesten
  - Fastställande av lämpliga response baserat på datatyper

Express exempel

```
//file: serverexpress.js
const express = require('express');
const app = express();
const port = 8000;
app.get('/', (req, res)=>{res.send('hej');});
app.listen(port, () => {
    console.log(`Listening on port ${port}!`);
});
```



Sqlite

- `npm install sqlite3` //Installerar sqlite3, kör i terminalen

- `dbm = require('sqlite3')` //få länk till databasmanagern
- `db = new dbm.Database(path_to_database)` //starta databasen
- `db.serialize(callback_function)` //exekverar databasfrågorna sekventiell
- `db.parallelize(callback_fuction)` //exekverar databasfrågorna parallell
- `db.run(sql-statement)` //UNDVIK denna typ av sql-frågor, använd `prepare` istället
- `statement = db.prepare(sql-statement)` //för säkerhet och effektivitet
- `statement.finalize()` //frigör allokerade resurs

Node Sqlite Exempel

```
//file: database.js
const path = require('path');
const sqlite3 = require('sqlite3');
const databasePath=path.join(__dirname, 'db.sqlite');
const db = new sqlite3.Database(databasePath);
db.serialize(() => {
db.run('CREATE TABLE userinfo(
                    username TEXT, password TEXT) ');
```

…

Node Sqlite Exempel

```
…
const statement = db.prepare("INSERT INTO userinfo
(username, password)VALUES ('vahid', 'blahonga')");
statement.run();
statement.finalize();
});   //Slutparentes för db.serialize

module.exports = db;//om man vill ha det som en egen module
```