Node.js

HTTP

christoffer.wallenberg@zocom.se

ZoCom

Repetition

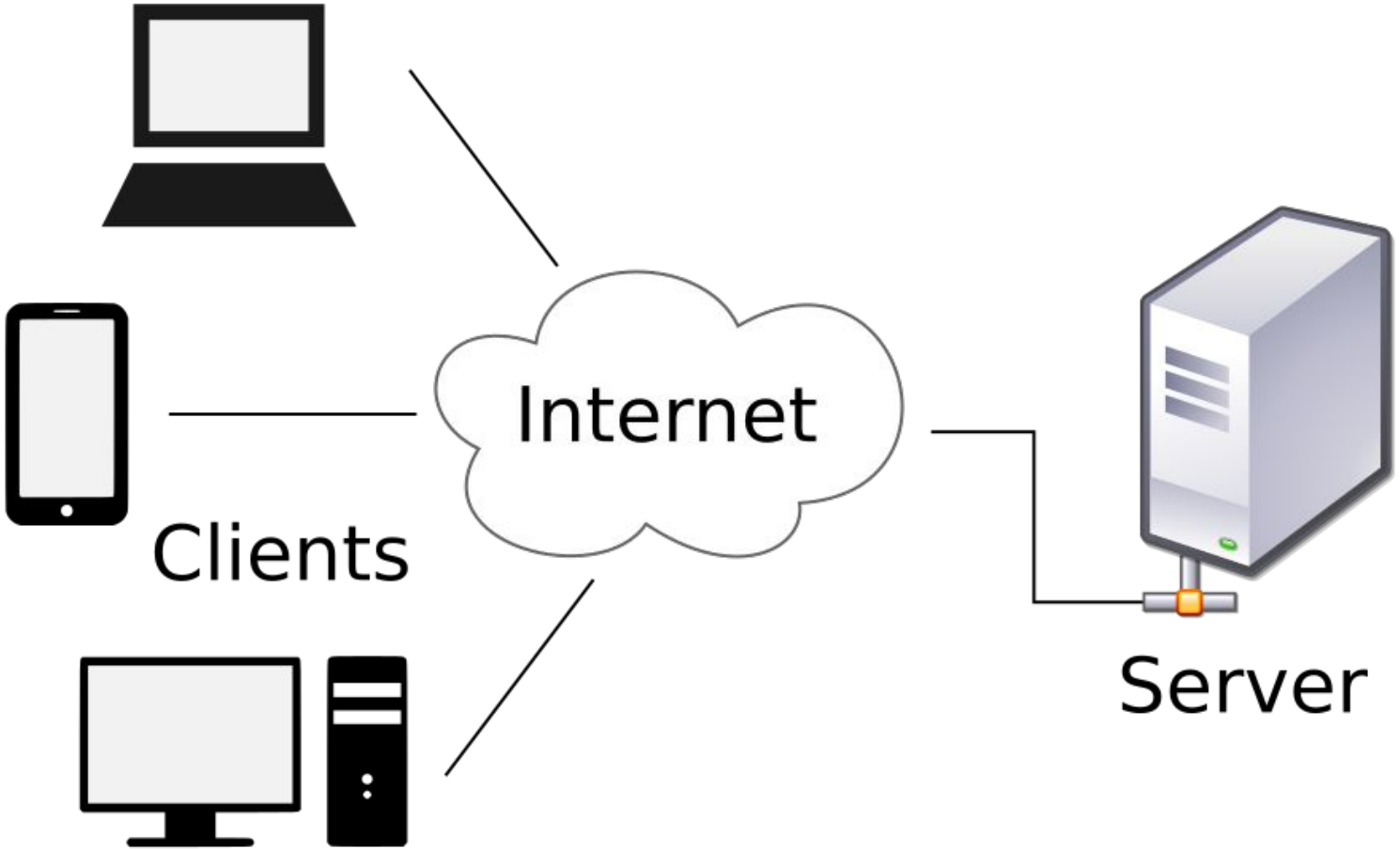
- Moduler används mycket i node
- Små bibliotek som avser att förenkla
- Finns både externa moduler via npm och inbyggda moduler
- Package.json innehåller alla moduler som används för detta projekt

Repetition

- Modulen **fs** används för att läsa/skriva
- **createReadStream** för att läsa från en fil
- **createWriteStream** för att skriva till en fil
- **Stdin** och **Stdout** hanterar indata och utdata
- **Console.log** använder exempelvis **stdout**

Klient-server modellen

Oldie but a goldie


Internet
Clients
Server

HTTP

- HTTP= HyperText Transfer Protocol används när webbläsaren skickar och tar emot information
- En förfrågan kallas för en **request** och webbserverns svar för **response**
- HTTP har olika metoder som **GET, POST, PUT, DELETE**
- **GET** används för att be om data från webbservern
- **POST** används för att skicka data till webbservern

HTTP-statuskoder

- Fungerar som en notis från webbservern till klienten och sätts av webbservern
- Berättar för klienten hur begäran hos webbservern gick
- Svarar med en kod ex. 200, 404
- Finns väldigt många statuskoder

Några vanliga statuskoder

- **200** - Allt gick bra!
- **403** - Tillgång förbjuden! Om klienten försöker nå resurser som kräver inloggning
- **404** - Resursen går inte att hitta
- **502** - Bad gateway. Om webbservern fått ett ogiltigt svar eller inget svar alls från exempelvis databasen.

HTTP modulen

- En inbyggd modul
- Används för att skapa en enkel webbserver
- Lyssnar på requests från en klient
- Skickar tillbaka ett response med data

En enkel webbserver

```
const http = require("http");
const server = http.createServer();

server.on('request', (request, response) => {
  response.end('Hello World!');
});

server.listen(8000);
```

response.end() säger att nu är svaret redo att skickas, det går inte att ändra efter det.

Sätta en statuskod

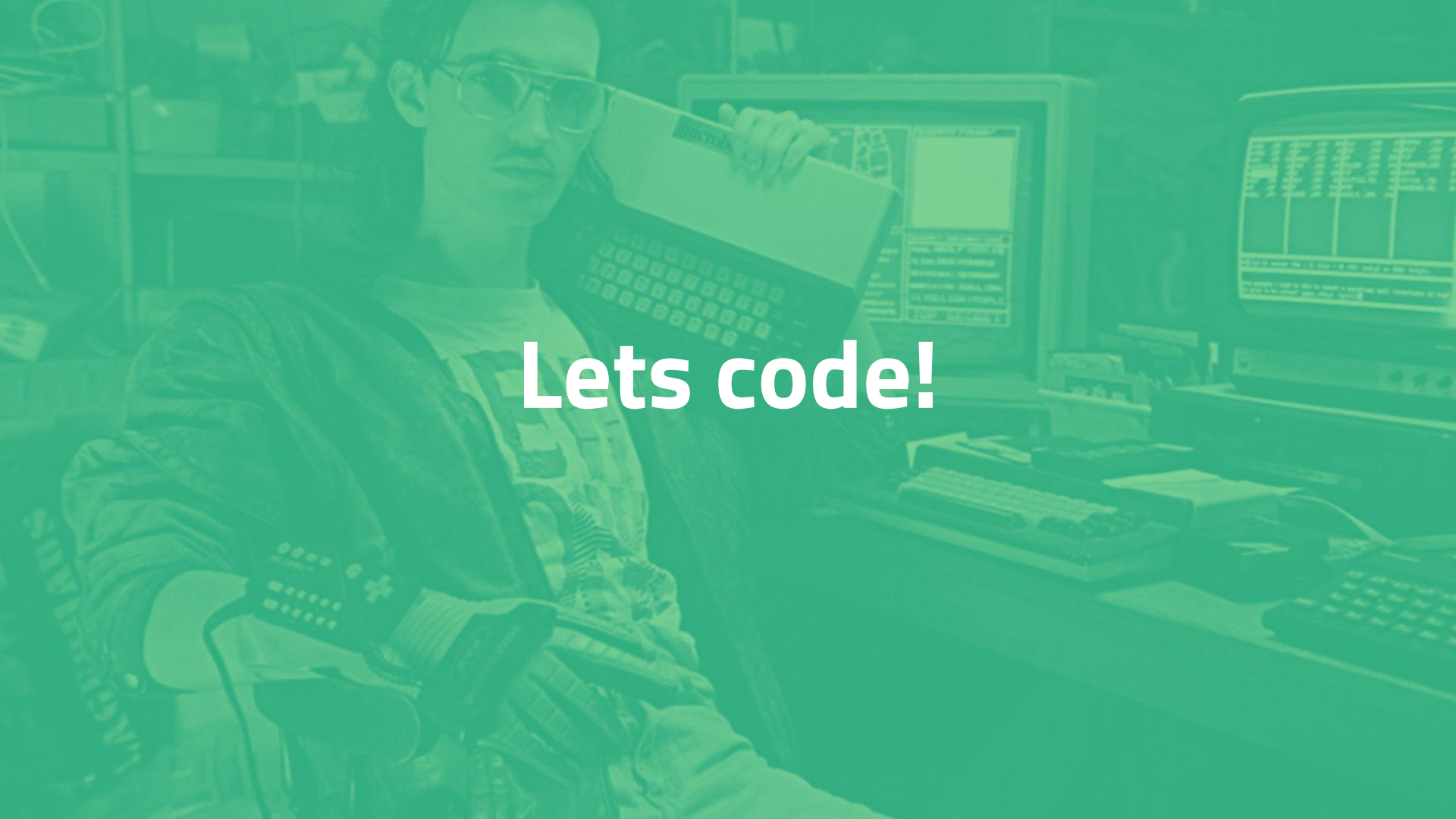
```
const http = require("http");
const server = http.createServer();

server.on('request', (request, response) => {
  response.writeHead(200).end('Hello World!');
});

  server.listen(8000);
```

St

atuskod 200 är dock default och behöver inte explicit sättas


Lets code!

Skicka tillbaka data som en ström

```
const fs = require("fs");
const http = require("http");
const server = http.createServer();

server.on('request', (request, response) => {
 const src = fs.createReadStream('OcscarWilde.txt');
 src.pipe(response);
});

server.listen(8000);
```

Kan jag inte använda readFile istället?

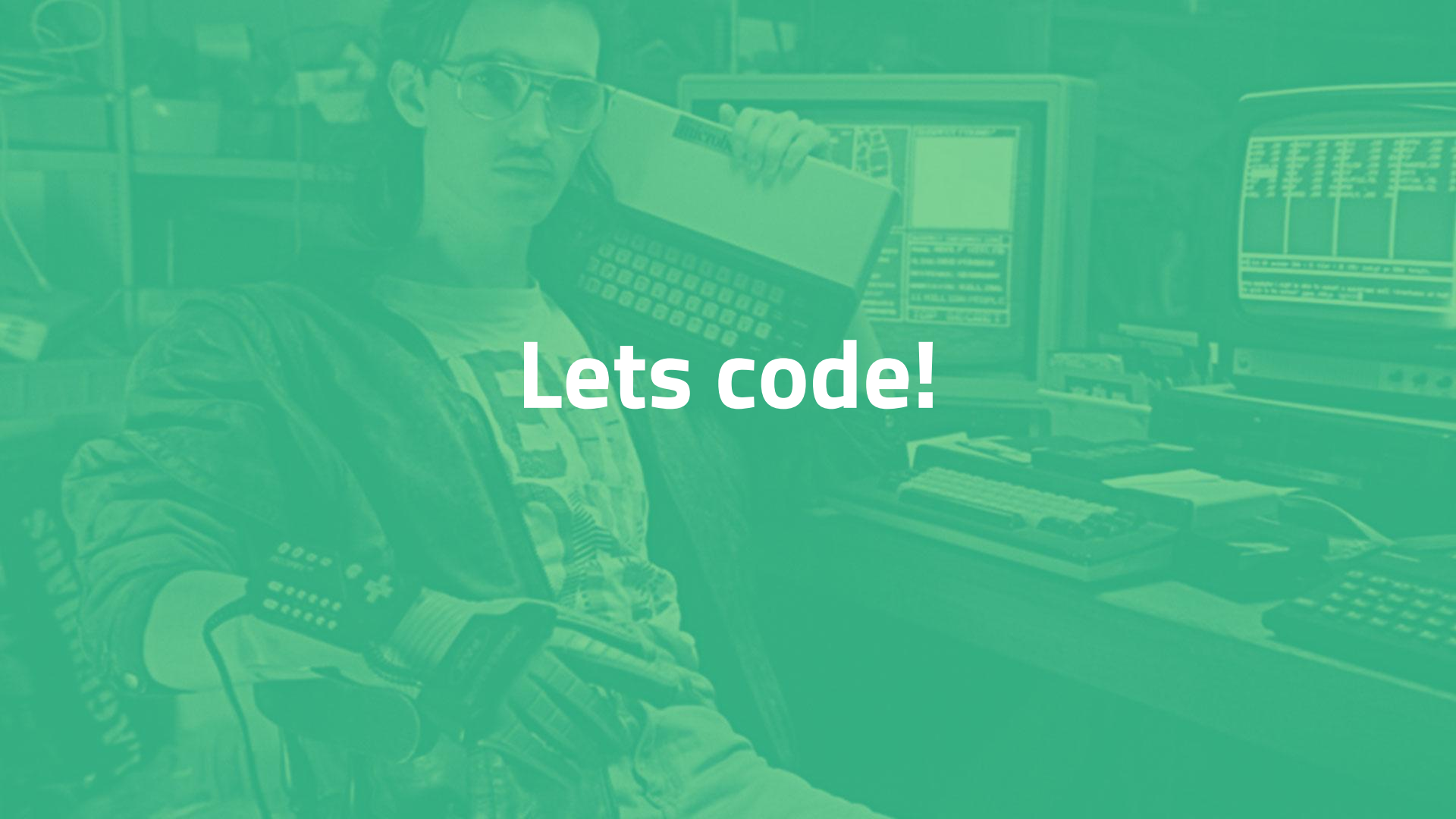
Låt oss kolla vad som händer!

Skicka tillbaka en HTML-fil

```
const fs = require("fs");
const http = require("http");
const server = http.createServer();

server.on('request', (request, response) => {
 const src = fs.createReadStream('index.html');
 src.pipe(response);
});

server.listen(8000);
```


Lets code!

Vad händer om jag lägger till css-fil?

Eller försöker gå in på en annan url?

Hantera olika url:er

Request

- Request-händelsen som vi har jobbat med har två parametrar **request** och **response**.
- **Request**-objektet innehåller om information om anropet från klienten
- Har bl.a. två egenskaper som är användbara **url** och **method**

Request

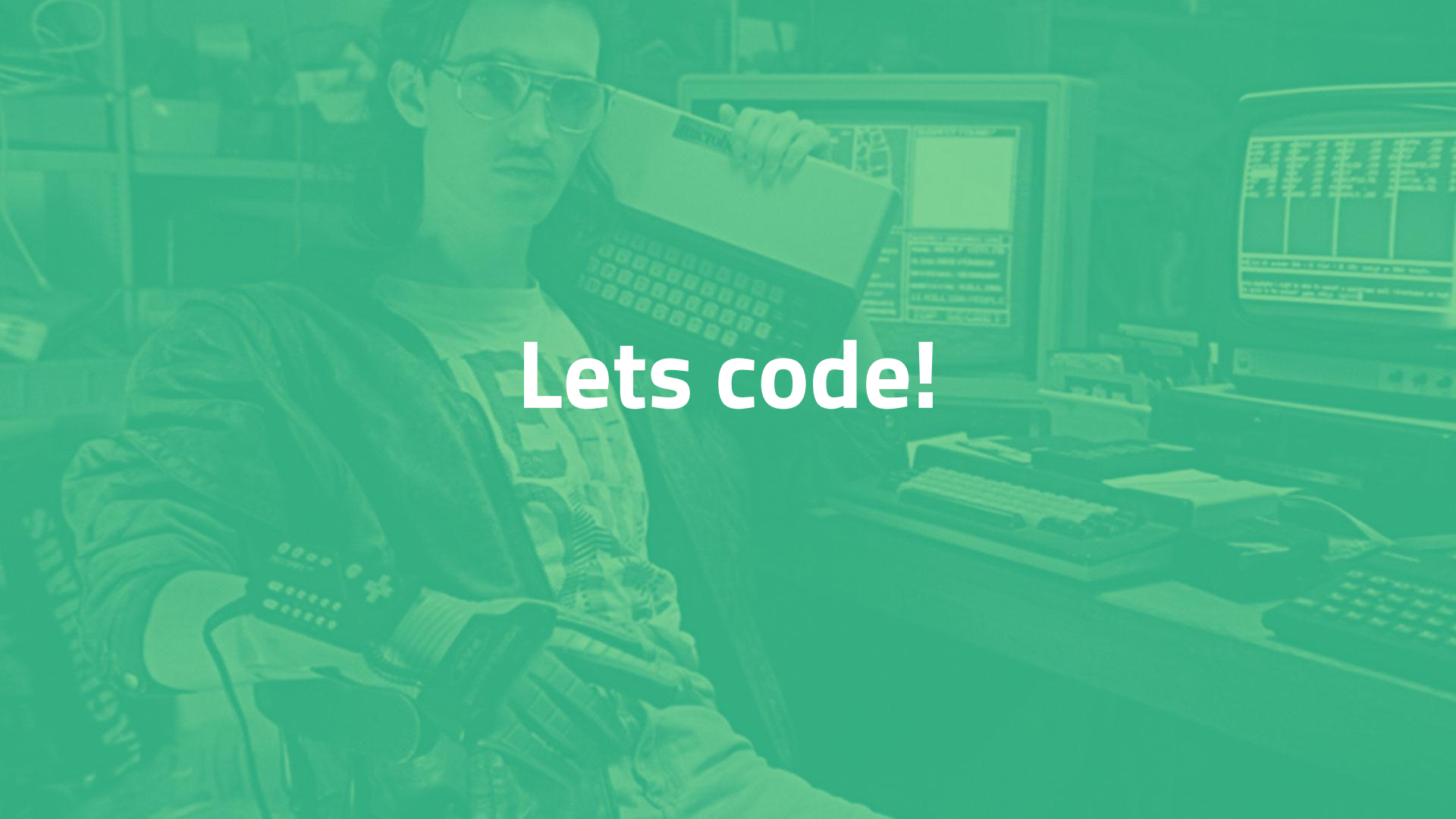
- **url** - innehåller den efterfrågade url:en från klienten
- **method** - om det är **GET, POST** osv. som ska användas

Ifall jag går in på url:en localhost:8000/test

```
const http = require("http");
const server = http.createServer();

server.on('request', (request, response) => {
 console.log(request.url); // skriver ut /test
 console.log(reguest.method); //Metoden blir GET
});

server.listen(8000);
```


Lets code!