# TDP013 - Webbprogrammering och interaktivitet

Föreläsning 2:
JavaScript, Node.js, Mocha och MongoDB

Robin Keskisärkkä
robin.keskisarkka@liu.se

Institutionen för Datavetenskap (IDA)

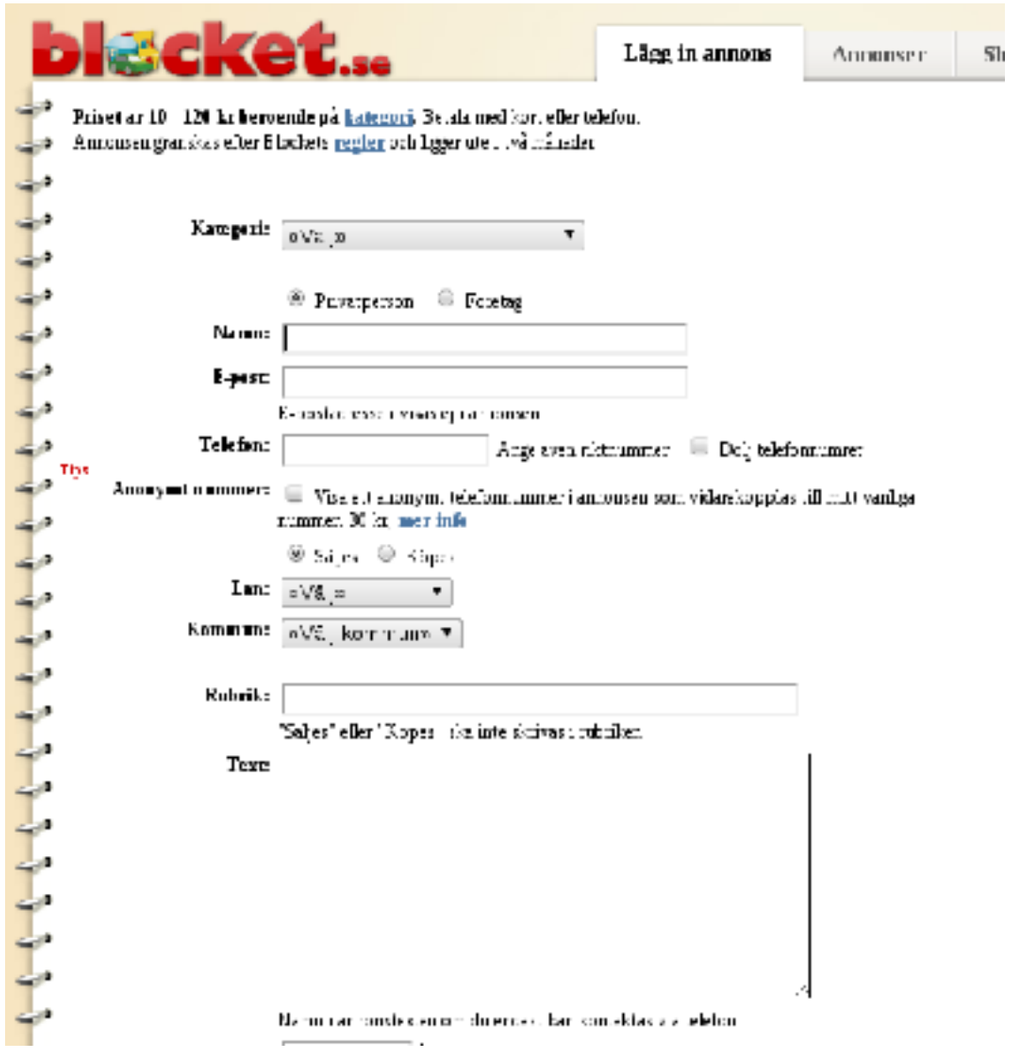
LINKÖPINGS UNIVERSITET

# Återblick från föreläsning 1

- HTML
    - Definierar struktur
    - Element representeras i DOM-trädet
- CSS
    - Definierar visuell stil och layout
    - Boxmodellen
    - Selectors för att välja ut specifika element
- ES6
    - Specifikationen för Javascript
    - Variabeldeklarationer (var (varning!), let och const)
    - Prototyper och objektorienterad programmering
    - Navigera i och modifiera DOM-trädet

# Den statiska webben

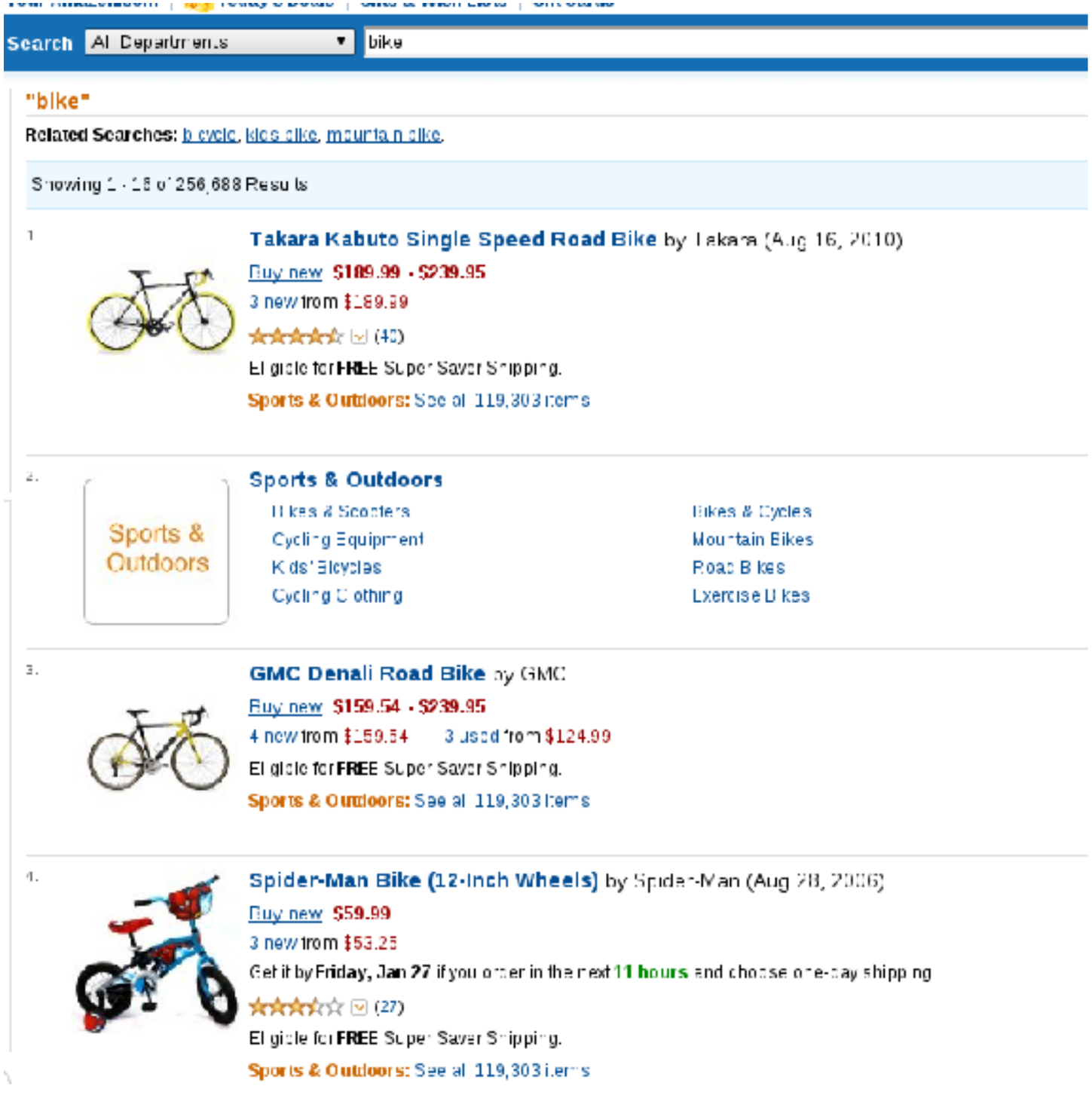
- HTML, CSS och Javascript sparas på filer
- Dokument skickas till och tolkas av webbläsaren
- Dokument förändras aldrig (annat än när de manipuleras manuellt)
- Varken praktiskt eller skalbart





- Användare fyller i formulär
- Annons dyker upp på hemsidan
- Vem skapar HTML, CSS och Javascript för hemsidan?

LINKÖPINGS UNIVERSITET

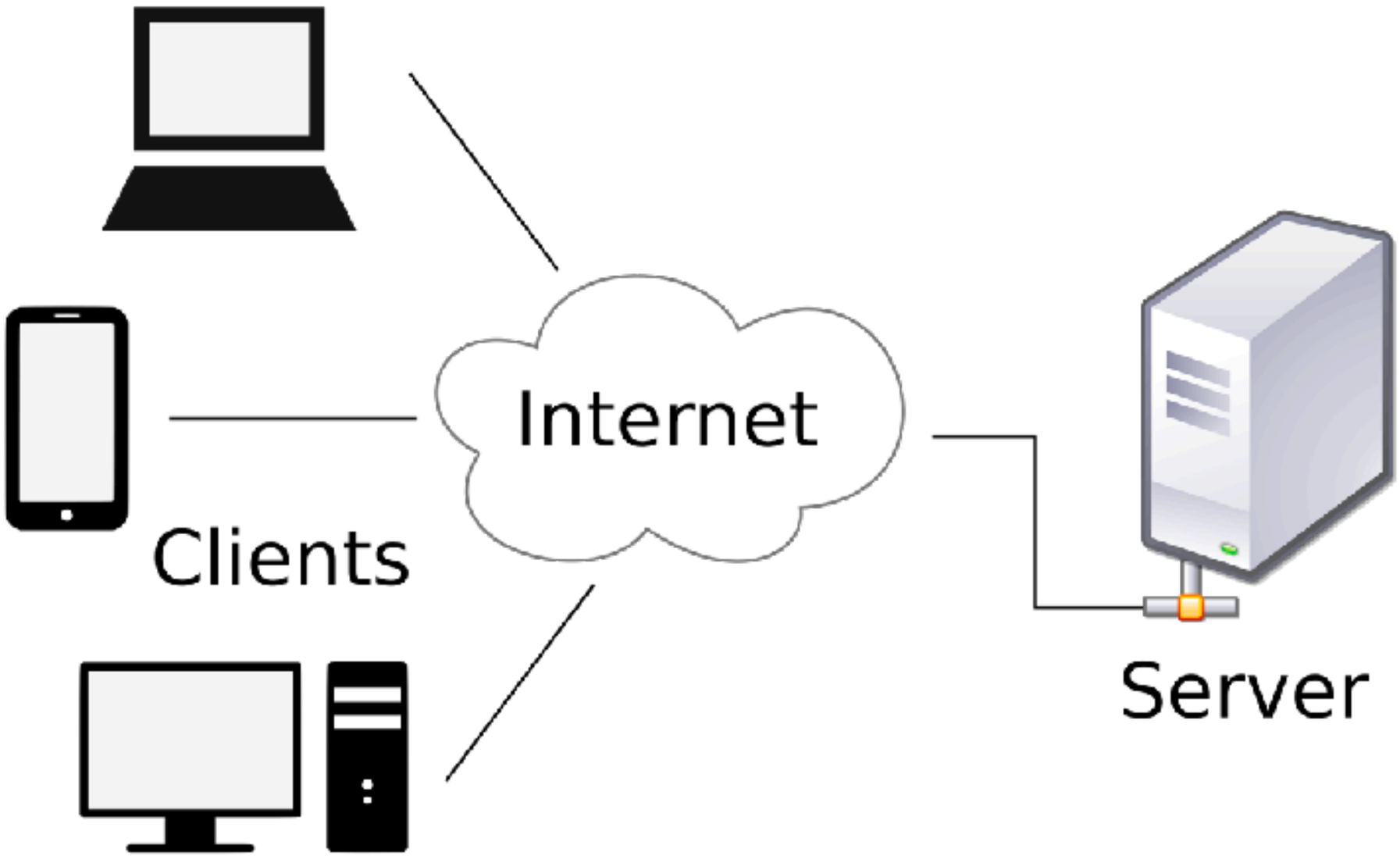



- Sökning på ordet ”bike” genererar en lista med både bilder och text.
- Hur skapades den här listan och vem skrev HTML, CSS och Javascript?
- Är det rimligt att ha färdiga HTML-sidor för alla tänkbara nyckelord?

LINKÖPINGS UNIVERSITET

# Det saknas något

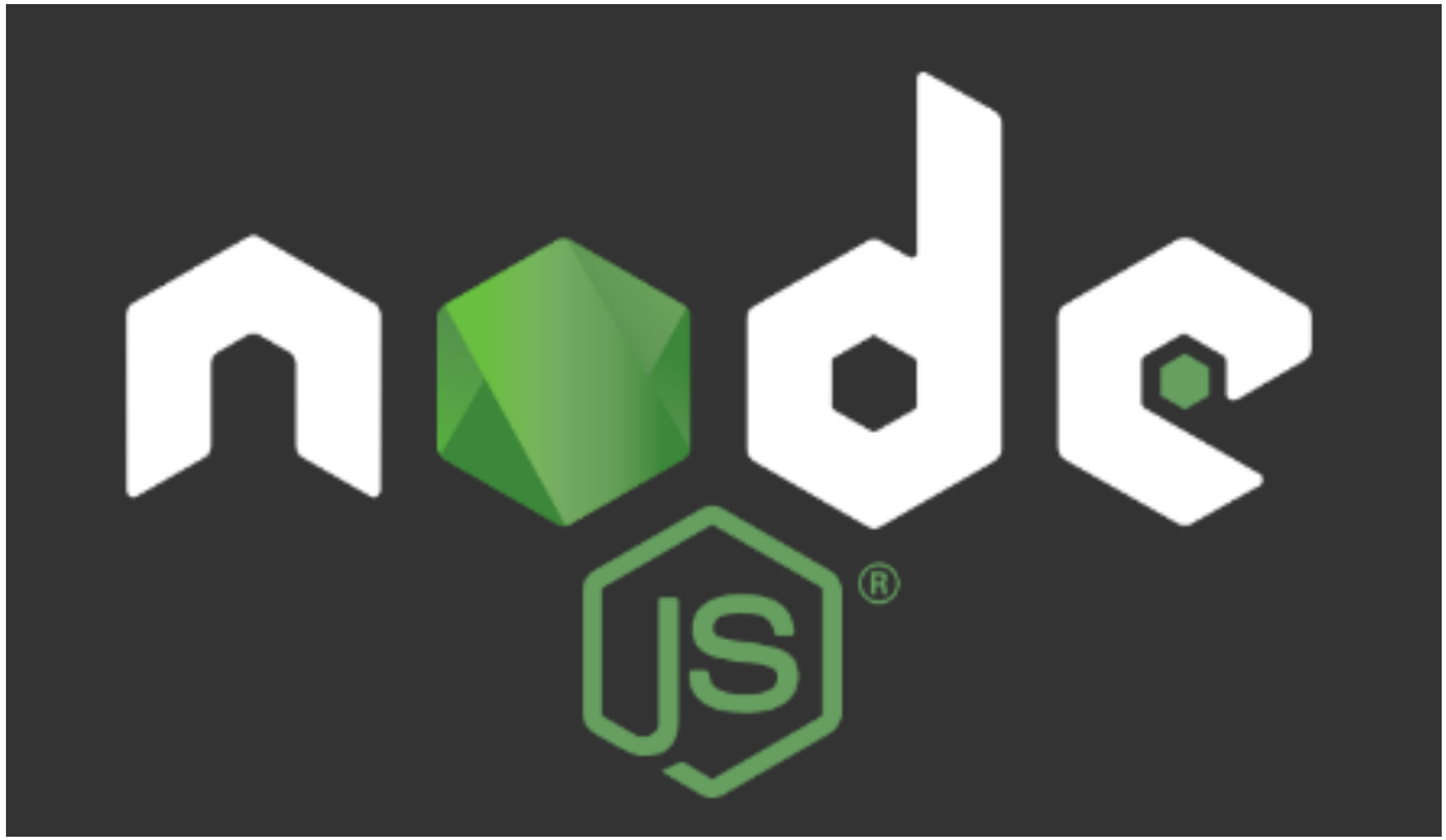
- JavaScript kan bara uppdatera DOM:en tillfälligt och ändringarna ses inte av någon annan
- HTML, CSS och Javascript räcker inte för att skapa dynamiskt innehåll på webbplatser
- ... men webbläsaren förstår inget annat än HTML, CSS och Javascript!
- Vi behöver något som kan skapa eller hämta innehåll som vi kan presentera för användaren


Internet
Clients
Server

# Server

- Statisk filserver
    - Filer uppdateras aldrig
- ... men flera alternativ finns:
    - PHP
    - Python
    - Java
    - Node.js
    - ...

# Node.js



LINKÖPINGS UNIVERSITET

# Node.js

- Kan användas för att sätta upp en HTTP-server
- Servern kan sedan ta emot data skickat med POST, GET, DELETE osv. och returnera exempelvis JSON
- Node.js kan även **kommunicera med en databas**
- Gör det möjligt att ha persistent data som vi både kan **skriva till och läsa ifrån**

# Node.js

- Miljö för att köra JavaScript på servern
- Bygger på **Chromes V8 JavaScript engine**
- JavaScripts event-struktur med **callbacks** som används flitigt i Node.js
    - Kan vara lite ovant i början för programmerare som inte arbetat så tidigare
- Node.js är open-source och kan hittas på GitHub
    - https://github.com/nodejs/node

# Vad är callbacks?

- Funktioner som **argument** till funktioner
- Lämnar över ansvaret för att fånga upp data och event till den kallade funktionen
- Stor del av både JavaScript och tredjeparts-bibliotek
- *”Om jag ger dig mitt leg, skulle du kunna hämta paketet jag beställt, lämna det utanför min dörr och sen ringa mig?”*

# Node.js med ES6

- Node.js stödjer nästan allt i ES6
- Fullt stöd kräver för ES6 kräver att vi använder Babel (en ‘code transpiler’)
  - Översätter ES6 till ES5 och vi kan använda alla nya ES6 features
- Vi kommer att använda ES6 till allt utom import och export av moduler och därmed kunna undvika Babel

# Node.js körs bara på serversidan!

- Ingen Node.js-kod körs på klientsidan
- Olika “ramverk” brukar användas för front- respektive backend för att underlätta och snabba på utveckling
- Håll det i åtanke när ni letar resurser!

LINKÖPINGS UNIVERSITET

# Skapa ett nytt Node.js-projekt

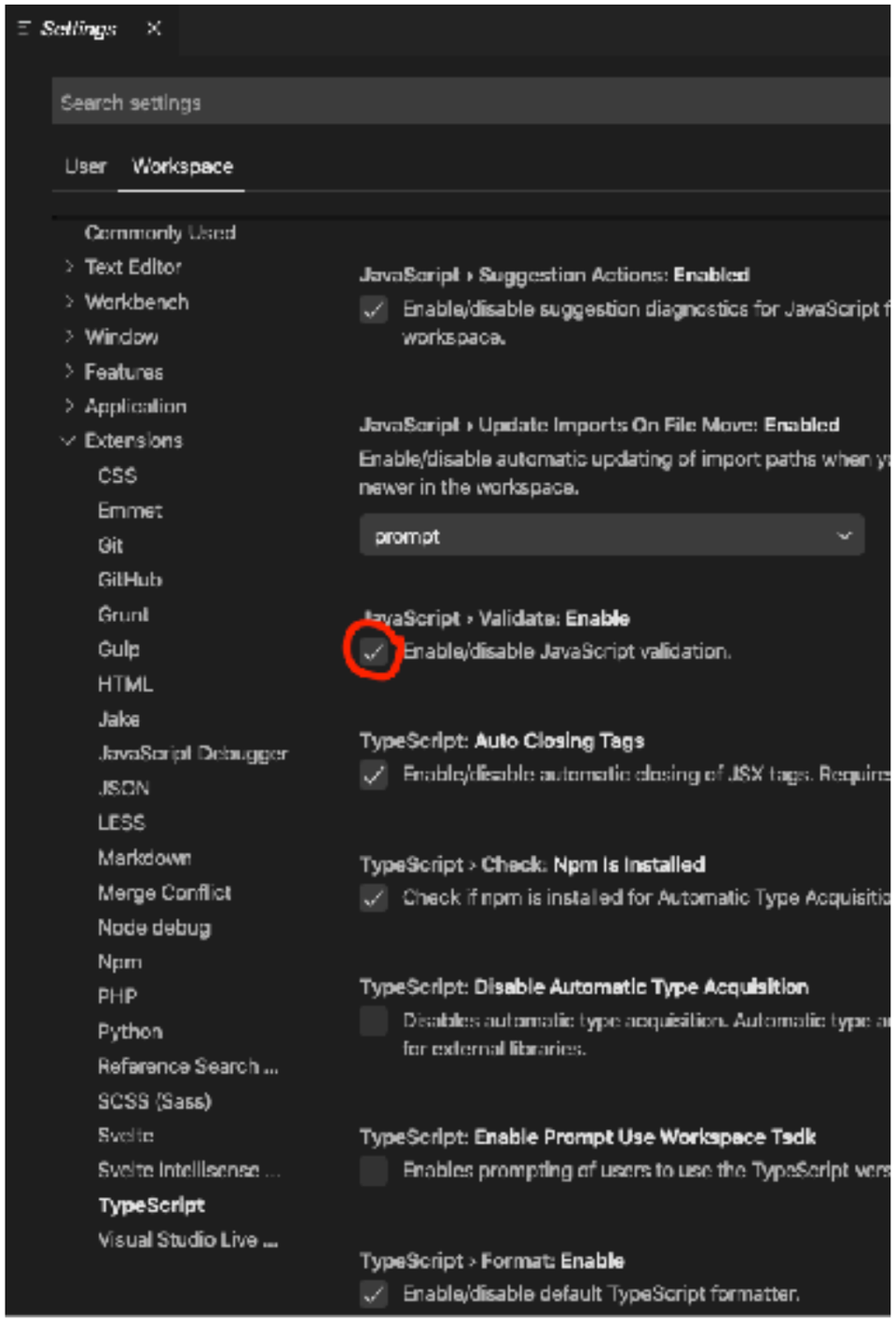
```
# Skapa mapp och index.js
mkdir my-app
cd my-app
touch index.js

# Initiera projekt
npm init -y

# Installera nodemon (så att vi slipper starta om node)
npm install --save-dev nodemon

# Lägg till start-instruktion till package.json under "script"
"start": "nodemon --exec node index.js”
```

# Skapa ett nytt Node.js-projekt med fullt stöd för ES6

```
# Skapa mapp och index.js
mkdir my-app
cd my-app
touch index.js

# Initiera projekt
npm init -y

# Installera babel (ES6-stöd)
npm install --save-dev @babel/core @babel/cli @babel/preset-env @babel/node

# Konfigurera babel
echo '{ "presets": [ “@babel/preset-env" ] }' > .babelsrc

# Installera nodemon (så att vi slipper starta om node)
npm install --save-dev nodemon

# Lägg till start-instruktion till package.json under "script"
"start": "nodemon --exec babel-node index.js”
```

# Mycket syntax-varningar i VS Code?

- Testa att avaktivera TypeScript-validering i inställningar

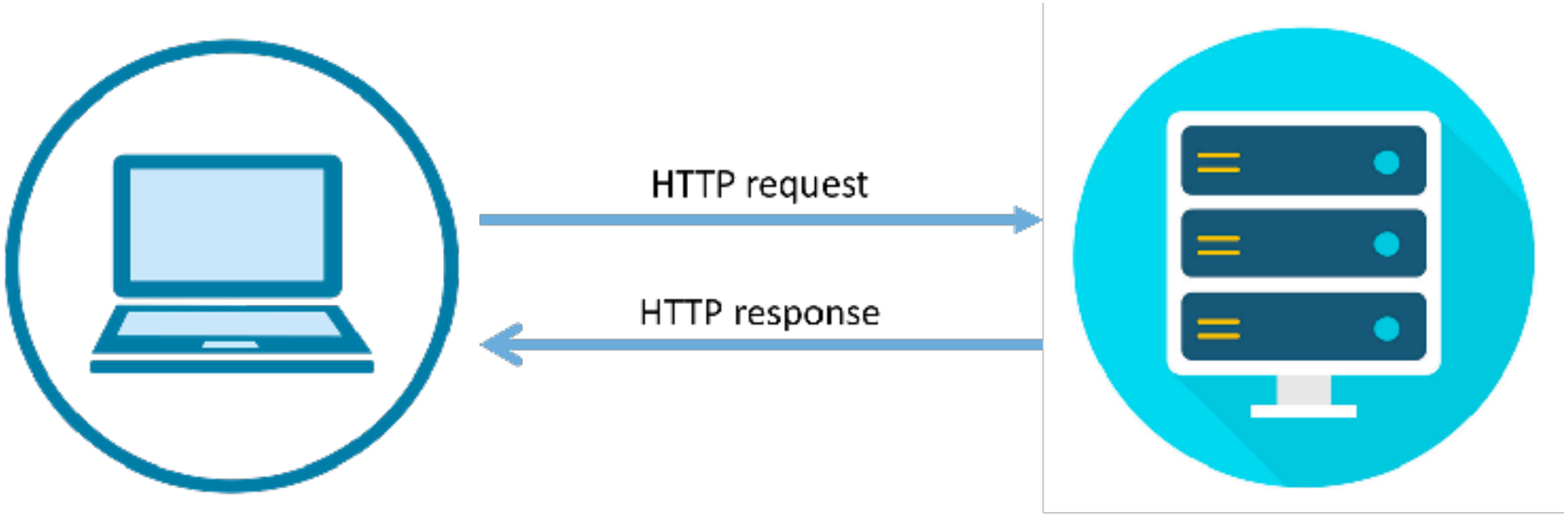



LINKÖPINGS UNIVERSITET

# Enkel http-server med Node.js


```
const http = require('http')
const url = require('url')
http.createServer((request, response) => {
    response.writeHead(200, {'Content-Type' : 'text/html'})
    let urlParts = url.parse(request.url, true)
    let name = urlParts.query['name'] || 'World'
    let phone = urlParts.query['phone']
    response.write(`<h1>Hello ${name}!</h1>`)
    if(phone){
        response.write(`<p>Is you're phone number ${phone}?</p>`)
    }
    response.end()
}).listen(8888)
```

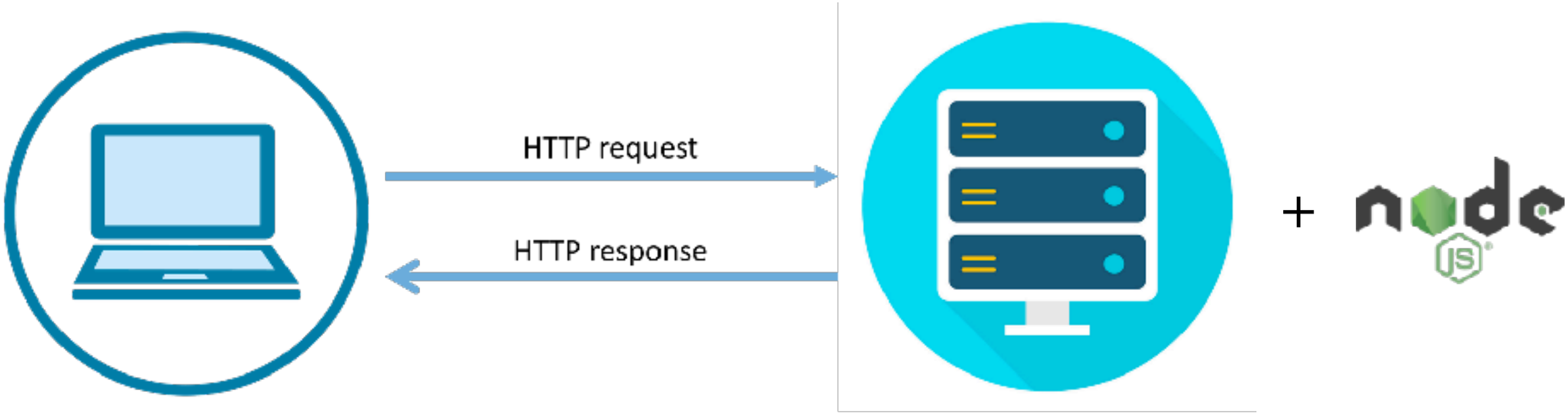
LINKÖPINGS UNIVERSITET




**Klassisk modell**

En ny tråd för varje inkommande anrop

Stor overhead per request




**Node.js**

Bara en tråd där alla requests hamnar i en event-loop

# Event-loop

- Node.js använder bara en tråd och alla requests körs i denna
- Om Node.js väntade på varje rad kod att exekvera innan den fortsatte skulle det alltså innebära att alla som gjorde anrop till servern fick vänta

```
let data = longRunningProcess()
```

Om vi är oförsiktiga och använder kod som ovan på fel ställe kan det resultera i att alla anrop får vänta väldigt länge...

# Asynkrona anrop

- Kör en funktion utan att pausa
- Kan utnyttja **callbacks** eller **Promises**
- Asynkrona funktioner markeras med **async**

```
async function doSomething(){
    // något tidskrävande
}
```

  - async returner implicit ett Promise

- För att vänta (inte att föredra!) på en **async**-function används **await**

```
await doSomething()
```

  - Kan användas för att få asynkrona anrop att bete sig seriellt
  - Måste i sin tur användas i en async-funktion

# Asynkrona anrop: Promise

- Objekt som representerar ett “löfte” (Promise)
- Ett Promise-objekt har ett state
    - **pending**
    - **fulfilled**
    - **rejected**
- Promise tar två callbacks som parametrar: **resolve** och **reject**
- När state ändras körs en av dessa funktioner
- resolve-parameterns värde sätts via **.then(...)**
- reject-parameterns värde sätts via **.catch(...)**
- Flera **.then(...)** kan definieras för samma Promise

# Asynkrona anrop: Promise

```
function loadData(){
    return [
        {'title': 'Gone in 60 seconds', 'year': 2000},
        {'title': 'Pulp Fiction', 'year': 1994}
    ]
}

let p = new Promise((resolve, reject) => {
    let data = loadData()
    if(data !== null){
        resolve(data)
    } else {
        reject('Failed to load data')
    }
})

p.then((x) => {
    // 'then' kallas på om vi lyckas
    console.log('Data loaded successfully:')
    console.log(JSON.stringify(x, null, 2))
}).catch((msg) => {
    // catch kallas på om vi misslyckas
    console.log(`Something went wrong: ${msg}`)
})
```

LINKÖPINGS UNIVERSITET

Promises skapas sällan manuellt! Vi behöver bara komma ihåg att Promises ofta returneras när vi använder inbyggda funktioner och moduler i Node.js!

# Kodexempel: Promise

LINKÖPINGS UNIVERSITET

# Mer om Node.js

- Egna moduler
- Organisation av filer i projekt
- Testramverket ‘mocha’
  - Modulerna ‘should’ och ‘superagent’
- Code-coverage med Istanbul

LINKÖPINGS UNIVERSITET

# Egna moduler

**// index.js**

```
let { start, PI } = require('/my-module')

console.log(`PI=${PI}`);

start(8888);
```

**// my-module.js**

```
let http = require('http');

function start(port){
    console.log(`Listening on port ${port}`);
    http.createServer((request, response) =>
{
        response.writeHead(200, {'Content-
Type' : 'text/html'});
        response.write(`<h1>Hello world!</
h1>`);
        response.end();
    }).listen(port);
}

const PI = 3.14159265358979323846;

module.exports = {
    start: start,
    PI: PI
}
```

LINKÖPINGS UNIVERSITET

# Organisation av filer

- Bör finnas en “entry point” till projektet i roten (om du inte utvecklar en modul)

```
./app.js

/lib
    requestHandlers.js
    route.js
    server.js
    main.js

/npm_modules
    Installerade moduler (bör ignoreras i ert repo)

/test
    test.js
```

# Mocha

- Testramverk
- Installera Mocha med hjälp av npm

```
$ npm install --save-dev mocha
```

- Skriv testfall och lägg i test/
  - Alla js-filer som ligger i test/ kommer att köras
- Lägg till “test” i package.json

```
“scripts” : {
    ”test”: “mocha”
}
```

Om testerna inte hinner köra färdigt:

```
“scripts” : {
    ”test”: “mocha --timeout 30s”
}
```

- Kör testerna: **npm test**

# Mocha: Exempel

```
const assert = require('assert')

describe('Array', () => {
  describe('#indexOf()', () => {
    it('should return -1 when the value is not present', () => {
      assert.equal([1, 2, 3].indexOf(4), -1)
    })
  })
})
```

# Mocha: Exempel med done()

```
const assert = require('assert')

describe('Array', () => {
  describe('#indexOf()', () => {
    it('should return -1 when the value is not present', (done) => {
      assert.equal([1, 2, 3].indexOf(4), -1);
      done();
    })
  })
})
```

# Mocha: Lite mer komplex (1)

```
const assert = require('assert')

describe('Array', () => {
  before((done) => {
    runServer().then(() => done());
  });

    describe('#indexOf()', () => {
      it('should return -1 when the value is not present', (done) => {
        request(url, mutation)
          .then((data) => {
            assert.equal(data.indexOf(4), -1);
            done();
          })
          .catch(err => done(err));
      });
    });

    after((done) => {
      closeServer().then(() => done());
    });
  })
```

# Mocha: Lite mer komplex (2)

```
const assert = require('assert')

describe('Array', () => {
  before((done) => {
    runServer().then(() => done());
  });
    describe('#indexOf()', () => {
      it('should return -1 when the value is not present', (done) => {
        runServer().then(() => { // start/close server for each test
          request(url, mutation)
            .then((data) => {
              assert.equal(data.indexOf(4), -1);
              closeServer().then(() => done());
            })
            .catch(err => closeServer().then(() => done(err));
          });
      });
    });
})
```

LINKÖPINGS UNIVERSITET

# should

- ‘should’ underlättar skrivandet av test (“an assertion library”)

```
$ npm install --save-dev should
```

```
const should = require('should')

let user = {
    name: ‘John’,
    pets: [‘Jane’, ‘Pete’, ‘Mary’]
}

user.should.have.property(‘name’, ‘John’)
user.should.have.property(‘pets’).with.lengthOf(3)
```

# superagent

- ‘superagent’ underlättar skrivandet av HTTP-anrop

```
$ npm install superagent
```

- Kan vara användbart när man testar anrop till sitt API

```
const superagent = require('superagent')

superagent
  .post('/api/pet')
  .send({ name: 'Manny', species: 'cat' })
  .set('accept', 'json')
  .end((err, res) => {
    // kontrollera resultat
  })
```

# Code coverage

- Hur stor del av min kod täcks av mina test?
- Installera Istanbul med npm
    - npm install --save-dev nyc
- Lägg till “coverage” till package.json:

```
“scripts” : {
    "coverage": "nyc --reporter=html npm test"
}
```

- Kör test med **npm run coverage**
- Öppna filen coverage/index.html för att see coverage
    - Visar endast coverage för de filer som importerats till något av testerna
- **Tips:** Kontrollera att även felhanteringen testas!

# Express.js

LINKÖPINGS UNIVERSITET

# Express.js

- Ett ramverk för node
- Underlättar och snabbar upp utveckling av Node.js backends
- Enkelt att komma igång

```
mkdir app
cd app
npm init
npm install express
```

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

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

let server = app.listen(3000, () => {
  let host = server.address().address
  let port = server.address().port

  console.log(`Lyssnar på http://${host}:${port}`)
})
```

LINKÖPINGS UNIVERSITET

# Express.js: express-generator

- För att snabbt skapa ett skelett för sin application kan man använda express-generator
    - Navigera till en tom mapp och kör: npx express-generator
- Bra beskrivning utav strukturen finns här: https://www.sitepoint.com/create-new-express-js-apps-with-express-generator/
- Kan behöva rensas upp bland filerna för att passa ens syfte

# Ramverk byggda på Express.js

- **Feathers**: Build prototypes in minutes and production ready real-time apps in days.
- **ItemsAPI**: Search backend for web and mobile applications built on Express and Elasticsearch.
- **KeystoneJS**: Website and API Application Framework / CMS with an auto-generated React.js Admin UI.
- **Kraken**: Secure and scalable layer that extends Express by providing structure and convention.
- **LEAN-STACK**: The Pure JavaScript Stack.
- **LoopBack**: Highly-extensible, open-source Node.js framework for quickly creating dynamic end-to-end REST APIs.
- **MEAN**: Opinionated fullstack JavaScript framework that simplifies and accelerates web application development.
- **Sails**: MVC framework for Node.js for building practical, production-ready apps.
- **Bottr**: Framework that simplifies building chatbot applications.
- **Hydra-Express**: Hydra-Express is a light-weight library which facilitates building Node.js Microservices using ExpressJS.
- **Blueprint**: Highly-configurable MVC framework for composing production-ready services from reusable components
- **Locomotive**: Powerful MVC web framework for Node.js from the maker of Passport.js

# Databas

- Hämta och spara data via Node.js
- Finns många databasalternativ
- Vi kommer att använda MongoDB
- Databasen kommer att ligga lokalt
    - Om ni vill använda molntjänsten för MongoDB måste ni ange hur vi kan koppla upp mot den

SQL
(relational databases)

| ID | NAME | BALANCE |
|---|---|---|
| 1 | Anders | 100 |
| 2 | Erik | 20 |
| 3 | Jalal | 200 |

NoSQL
(graph databases, key-value stores, mixed models …)

Cassandra
CouchDB
MongoDB
DynamoDB
Datastore
….
Sparar information som grafer, key-value, json, xml, etc.

# MongoDB

- Information sparas som JSON-objekt
- Objekten sparas i “collections”
    - t.ex. Movie, Actor, Director etc.
    - En collection kan jämföras med en tabell i en SQL-databas
    - Inget schema
        - … om man inte exempelvis använder Mongoose-modulen
- Ingen garanti för att två Movie-objekt innehåller samma fält, t.ex title, length, etc.
- MongoDB är lättviktigt, snabbt och skalar bra!

# MongoDB: Komma igång

- Installera och starta MongoDB
- Logga in via terminal:

```
mongo
```

- Visa alla databaser:

```
show dbs
```

- Välj databas:

```
use tdp013
```

  - Om databasen inte finns så skapas den

- Spara ett nytt objekt i collection movie:

```
db.movie.insert({'a': 1})
```

- Lista alla objekt i collection movie:

```
db.movie.find()
```

- Läs på om olika funktioner i MongoDB-dokumentationen
- **Tips:** MongoDB har interna ID:n för alla objekt.
- **Tips:** MongoDB Compass ger ett GUI mot MongoDB

# Node.js: MongoDB driver

- Installera en driver för MongoDB
    - Exempel: `npm install mongodb`
- Exempel: Lista data

```
const { MongoClient } = require('mongodb');
let url = "mongodb://localhost:27017/";

MongoClient.connect(url, (err, db) => {
    if(err) { throw err; }
    let dbo = db.db('tdp013');
    dbo.collection("movie").find({}).toArray((err, result) => {
        if(err){ throw err; }
        console.log(result);
        // close connection when done with mongo
        db.close()
    });
});
```

# Node.js: MongoDB driver

- Lägga till ny data
- Callback är inte ett krav
- … om vi inte vill hålla kolla på om vi lyckats lägga in datan eller ej

```
MongoClient.connect(url, (err, db) => {
  if(err){ throw err; }
  let dbo = db.db('tdp013');
  let myobjects = [
    { _id: 7, name: 'Chocolate Heaven'},
    { _id: 8, name: 'Tasty Lemon'},
    { _id: 9, name: 'Vanilla Dream'}
  ];
  dbo.collection("products").insertMany(myobjects, (err, res) => {
    if (err){ throw err; }
    console.log(res);
    db.close();
  })
})
```

# MongoDB driver:
# Att tänka på

- Även om det inte kommer märkas i labb 2 så är etablering av en ny connection **mycket** tidskrävande i relativa termer
- Bra att återanvända samma uppkoppling så långt som möjligt
- Vill man använda någon annan driver för MongoDB stå går det bra
- **Kom ihåg!** Labbassarna måste kunna köra er kod. Inkludera därför information om hur man ska konfigurera databasen om ni inte använder default.

LINKÖPINGS UNIVERSITET