TDP013 - Webbprogrammering och interaktivitet

Föreläsning 2:
Cookies, Node.js, Express och Mocha

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

Cookies

LINKÖPINGS UNIVERSITET

Pilfunktioner

- Ett sätt att definiera anonyma funktioner
- Används mycket flitigt för att hantera ex. asynkrona anrop
- Exempel:

```
function hello1(){
  console.log('Hello world!')
}
do_time_consuming_thing.then(hello1)

let hello2 = () => console.log('Hello world!')
do_time_consuming_thing.then(hello2)

do_time_consuming_thing.then(
  () => console.log('Hello world!')
)
```

Callback-funktioner

Cookies

- Cookies är små mängder data som sparas i webbläsaren (max 4 kB)
- Cookies innehåller ofta data som laddats från en server eller genererats från webbläsaren
- Med JavaScript kan man läsa, uppdatera eller lägga nya cookies
- Cookies gör det möjligt att behålla data lokalt även om sidan skulle laddas om
- Viktigt
    - I vissa browsers fungerar cookies endast om koden körs från en webbserver, dvs. inte om man bara öppnar filen i webbläsaren.
    - En enkel webbserver kan startas i Python genom att köra:
      $ python3 -m http.server

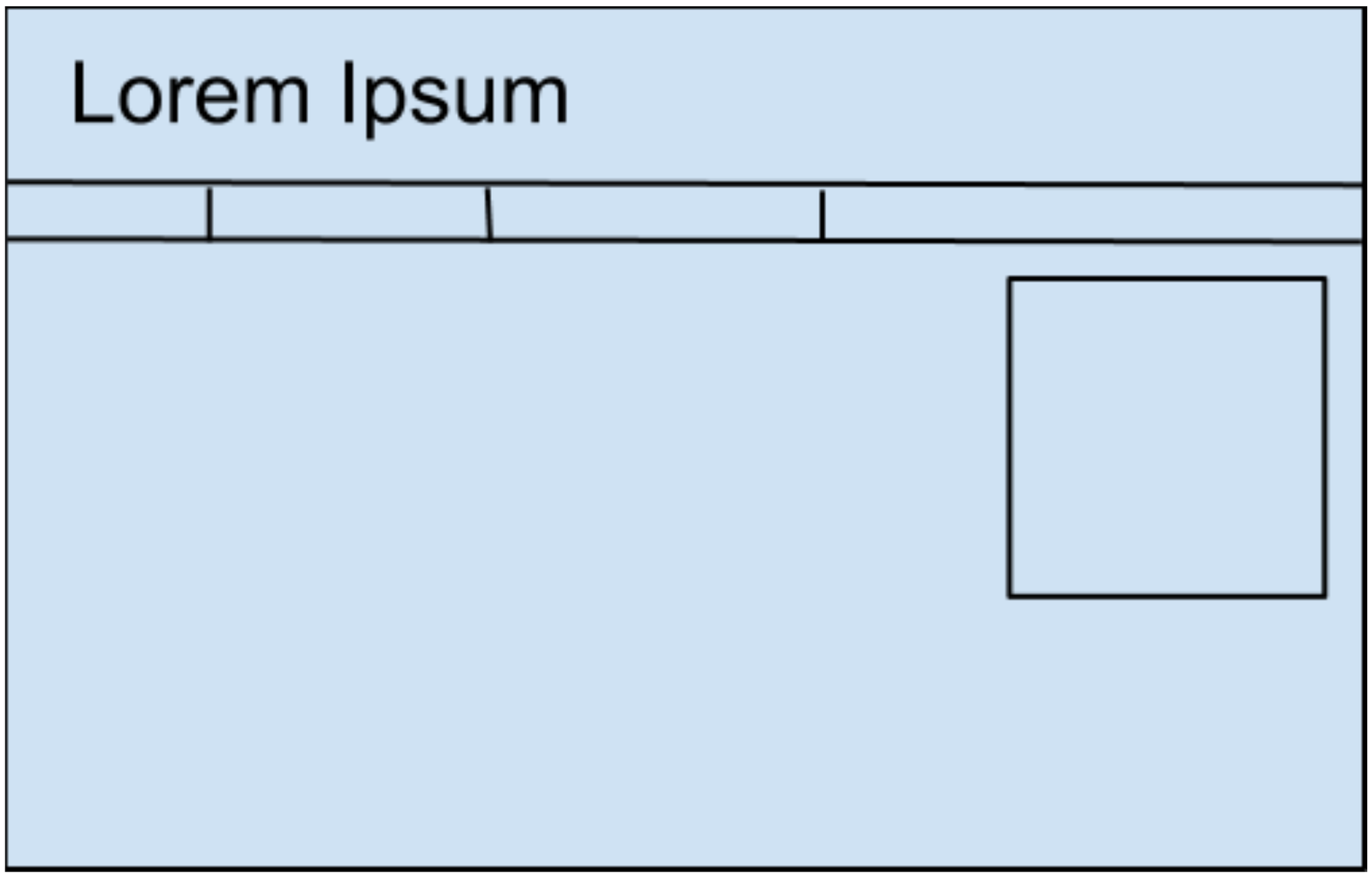
Cookies

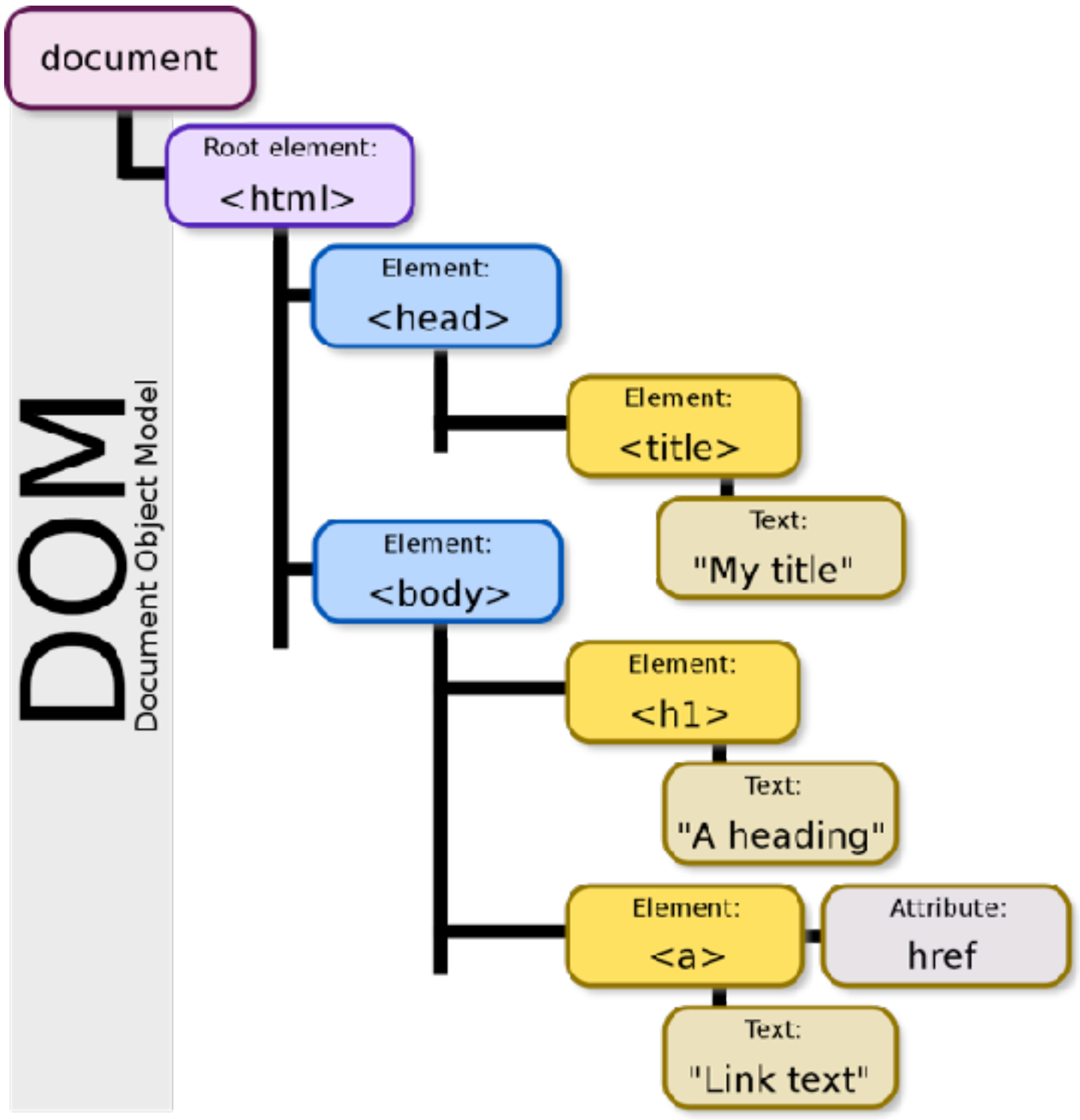
- Cookies från klienten skickas varje gång ett HTTP-anrop görs
- Används ofta för autentisering
- Cookies **bör** inte innehålla någon känslig information!
- Path-argumentet kan användas för att begränsa när din cookie ska skickas med

Document object model (DOM)

LINKÖPINGS UNIVERSITET

Hemsida





https://en.wikipedia.org/wiki/Document_Object_Model

Noder

- Varje element i ett HTML dokument är en nod i DOM-trädet (inklusive **<!-- kommentarer -->**)
- Det finns 12 olika typer av noder
  - Element, TextNode och AttributeNode är de tre typer som generellt sett är intressanta för webbdesign

Navigera i DOM

- För att göra ändringar i DOM-trädet med JavaScript måste man kunna få tag i specifika element, ex:
  - document.getElementById('param') returnerar elementet med angivet ID
  - document.getElementByTagName('param') returnerar en lista med element med en viss tag
  - document.querySelector(<css selector>) returnerar det fösta elementet baserat på en CSS selector.
  - document.querySelectorAll(<css selector>) hämtar en lista av element baserat på en CSS selector.

Operationer på noder

- element.childNodes returnerar en lista med alla noder direkt under element i DOM-trädet.
- element.parentNode returnerar den nod som finns direkt ovanför element i DOM-trädet.
- element.nextSibling returnerar den nod som finns direkt till höger och på samma nivå som element i DOM-trädet.
- element.previousSibling returnerar den nod som finns direkt till vänster och på samma nivå som element i DOM-trädet.

Operationer på noder

- document.createElement('param') skapar ett nytt element baserat på en tag uttryckt som en sträng.
- document.createTextNode('param') skapar en ny TextNode från en sträng.
- element.appendChild(child) placerar det angivna elementet child sist i listan av noder direkt under element.
- element.removeChild(child) tar bort ett element från listan av noder direkt under det specificerade elementet. Noden måste finnas i listan över elementets barn.

Livekodning

LINKÖPINGS UNIVERSITET

Node.js

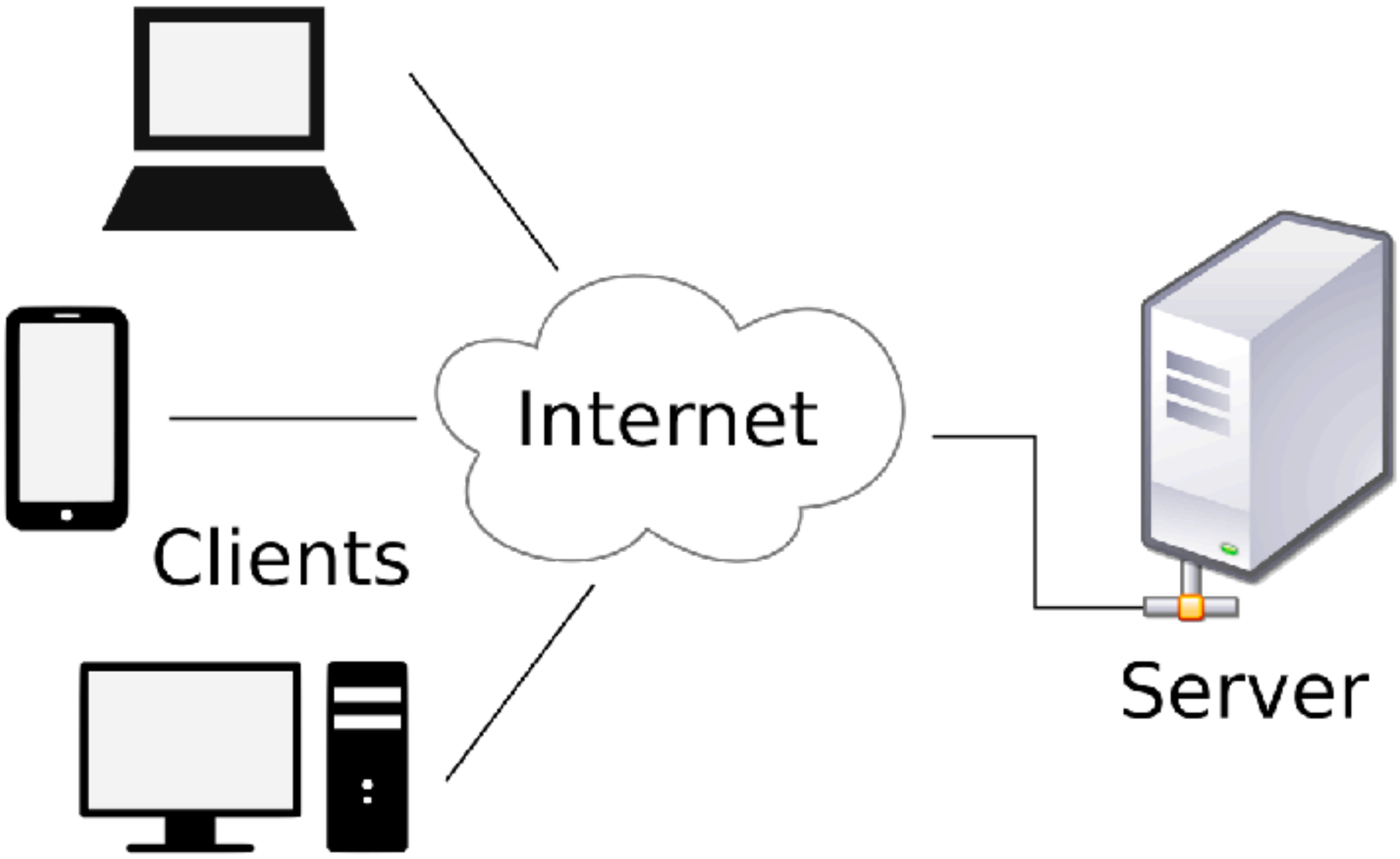
LINKÖPINGS UNIVERSITET

Den statiska webben

- HTML, CSS och JavaScript sparas som filer
- Dokument skickas till och tolkas av webbläsaren
- Dokument förändras aldrig (annat än när de manipuleras manuellt)
- Varken praktiskt eller skalbart
- Är det rimligt att bara ha färdiga HTML-sidor för exempelvis en webbutik eller ett webbforum?

Det saknas något

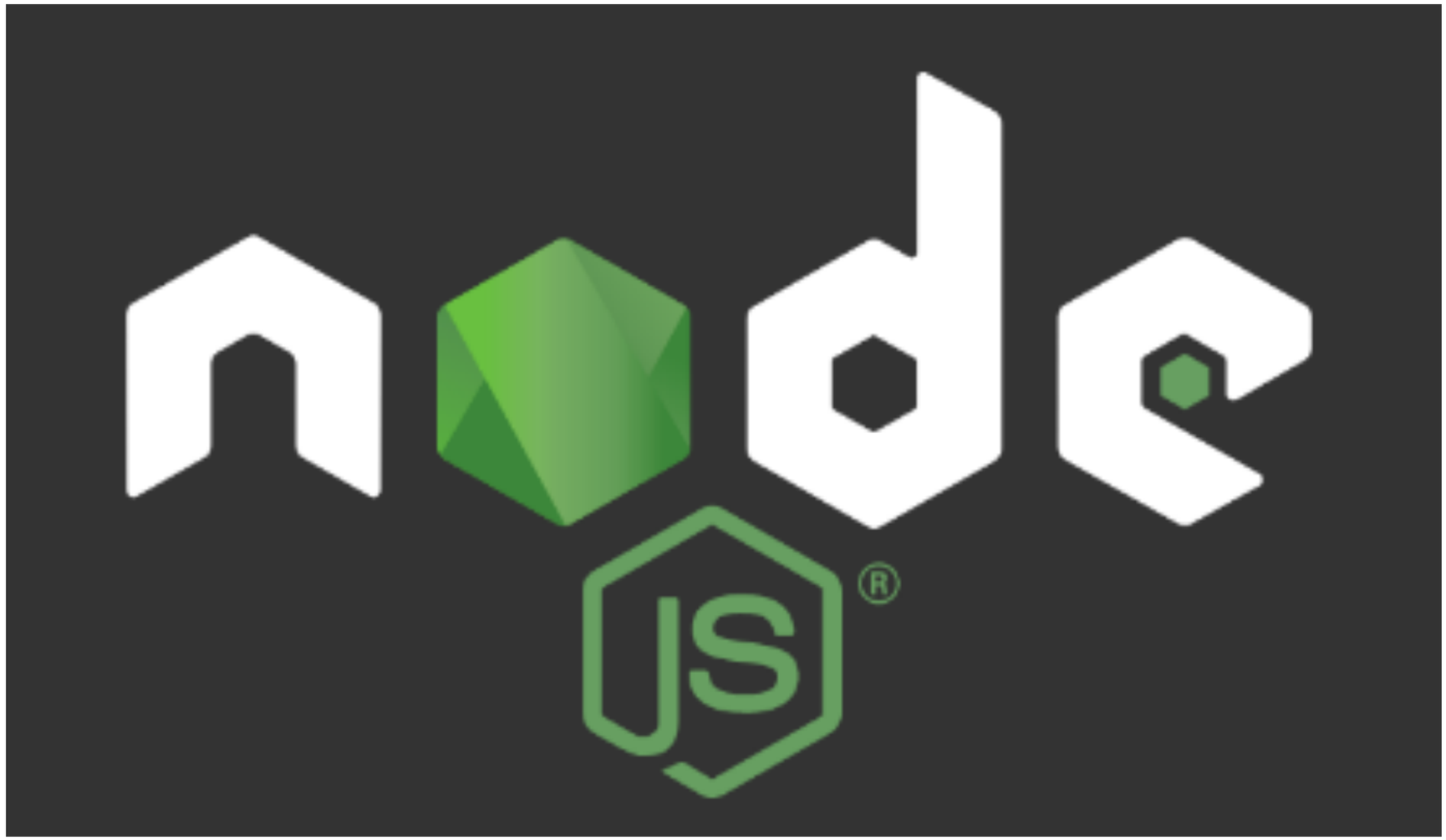
- JavaScript kan bara uppdatera DOM:en tillfälligt och ändringarna ses inte av någon annan
- HTML, CSS och JavaScript räcker inte för att skapa dynamiskt innehåll på webbplatser
- … men webbläsaren förstår inget annat än HTML, CSS och Javascript!
- Vi behöver något som kan skapa eller hämta innehåll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


Internet
Clients
Server

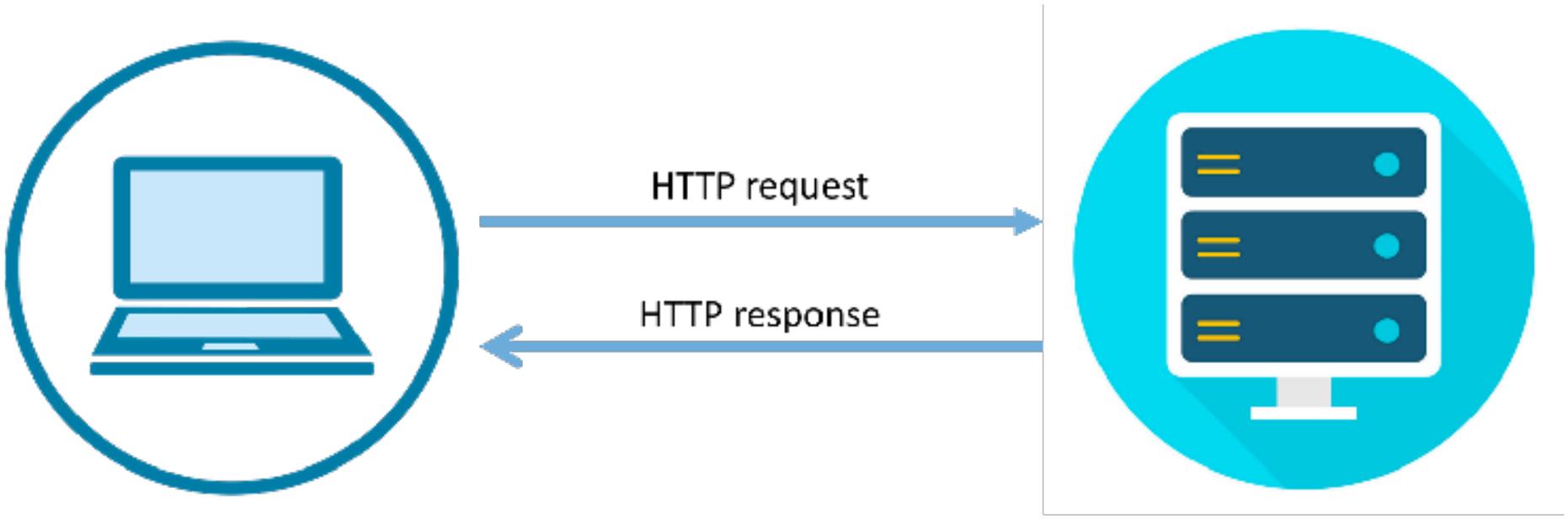
Server

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

Node.js



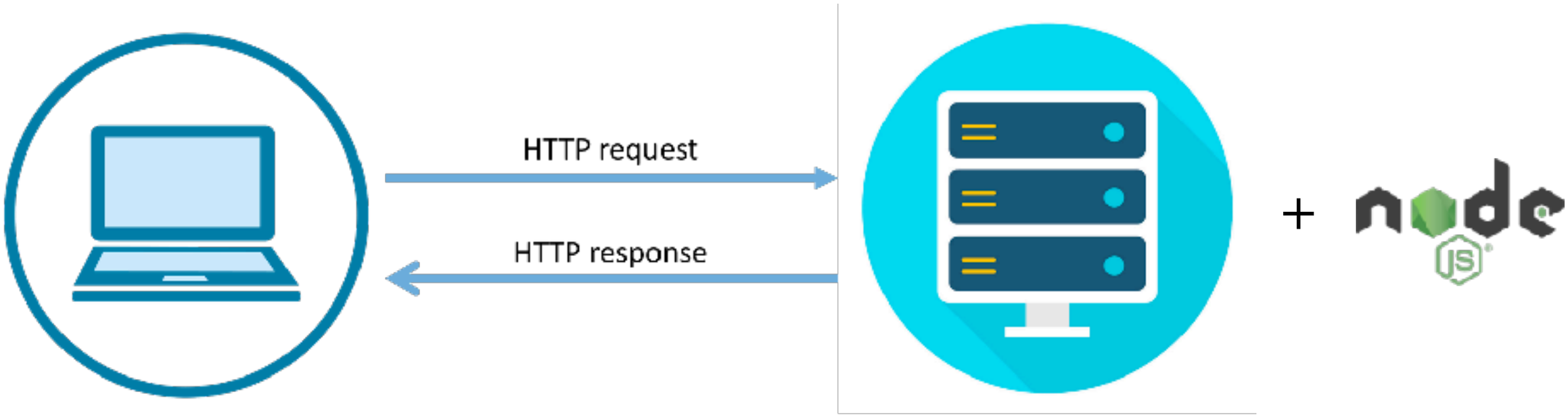
LINKÖPINGS UNIVERSITET



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

Node.js

- Miljö för att köra JavaScript på servern
- Bygger på **Chromes V8 JavaScript engine**
- JavaScripts event-struktur med **callbacks** 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

Node.js

- Kan användas för att sätta upp en HTTP-server
- Servern kan 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 med ES6

- Node.js stödjer nästan allt i ES6
- Import och export av moduler kräver att vi definierar type till module i package.json, använder en speciell filändelse (.mjs) eller sätter en flagga när vi kör koden (—input-type=module)

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!

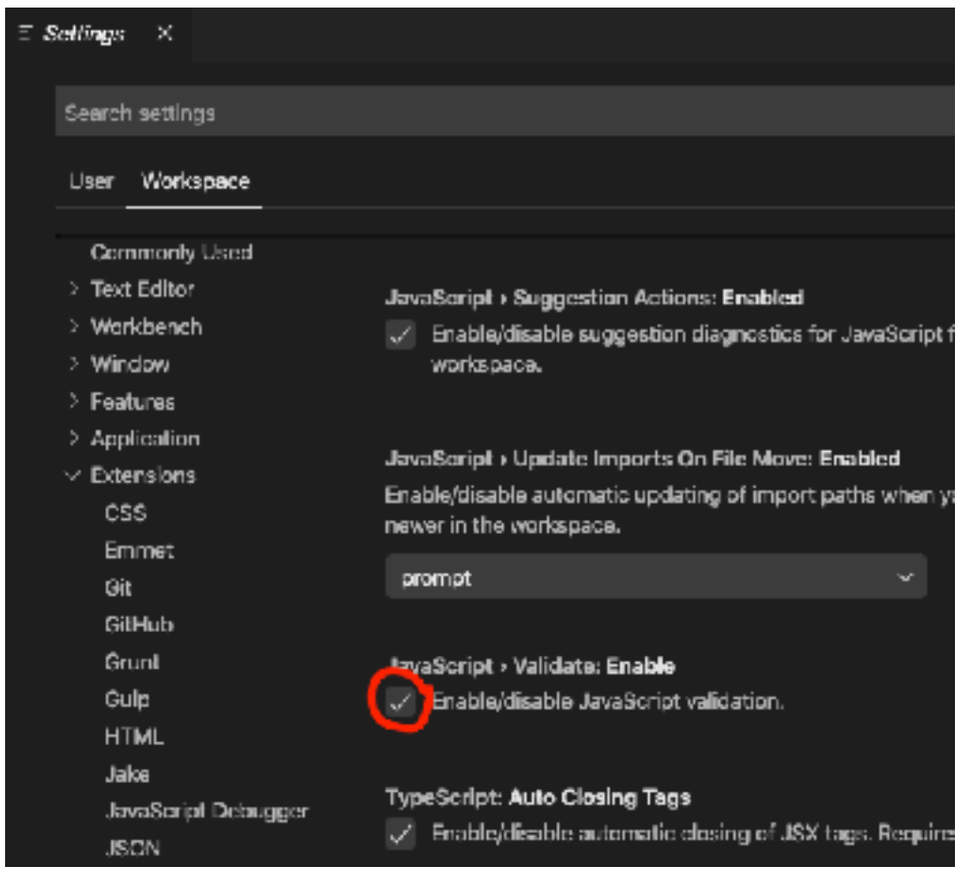
LINKÖPINGS UNIVERSITET

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 nodemon

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

Många syntax-varningar i VS Code?

- Testa att avaktivera TypeScript-validering i inställningar!



LINKÖPINGS UNIVERSITET

Livekodning

LINKÖPINGS UNIVERSITET

Callbacks och asynkrona anrop

LINKÖPINGS UNIVERSITET

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 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 responsen bli mycket långsam...

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}`)
})
```

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 pass, skulle du kunna hämta paketet jag beställt, lämna det utanför min dörr och sen ringa mig?”*

Livekodning

LINKÖPINGS UNIVERSITET

Mocha

LINKÖPINGS UNIVERSITET

Mocha

- Testramverk
- Installera Mocha med hjälp av npm

```
$ npm install 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 avancerad

```
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());
    });
  })
```

should

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

```
$ npm install should
```

```
import should from '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
```

- Användbart när man testar anrop till sitt API

```
import superagent from 'superagent'

superagent
    .get('https://pokeapi.co/api/v2/pokemon/pikachu')
    .set('accept', 'json')
    .end((err, res) => {
        console.log(res.body.name)
        console.log(res.body.weight)
        console.log(JSON.stringify(res.body.types))
    })
```

Code coverage

- Installera Istanbul med npm
    - npm install nyc
- Lägg till “coverage” till package.json:

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

- Kör med **npm run coverage**
- Öppna filen coverage/index.html för att see coverage

- **OBS:** Endast coverage för filer som importerats direkt eller indirekt av testerna inkluderas.
- **Tips:** Kontrollera att även felhanteringen testas!

Tomma rapporter?

- Istanbul har en känd bugg om man arbetar med moduler
- Om ni får tomma rapporter så testa att istället använda **c8**
- Installera **c8** med npm
  - npm install **c8**
- Lägg till “coverage” till package.json:

```
"scripts" : {
    "coverage": "c8 --reporter=html npm test"
}
```

Express

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
```

```
import express from '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}`)
})
```

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

Livekodning

LINKÖPINGS UNIVERSITET

LINKÖPINGS UNIVERSITET