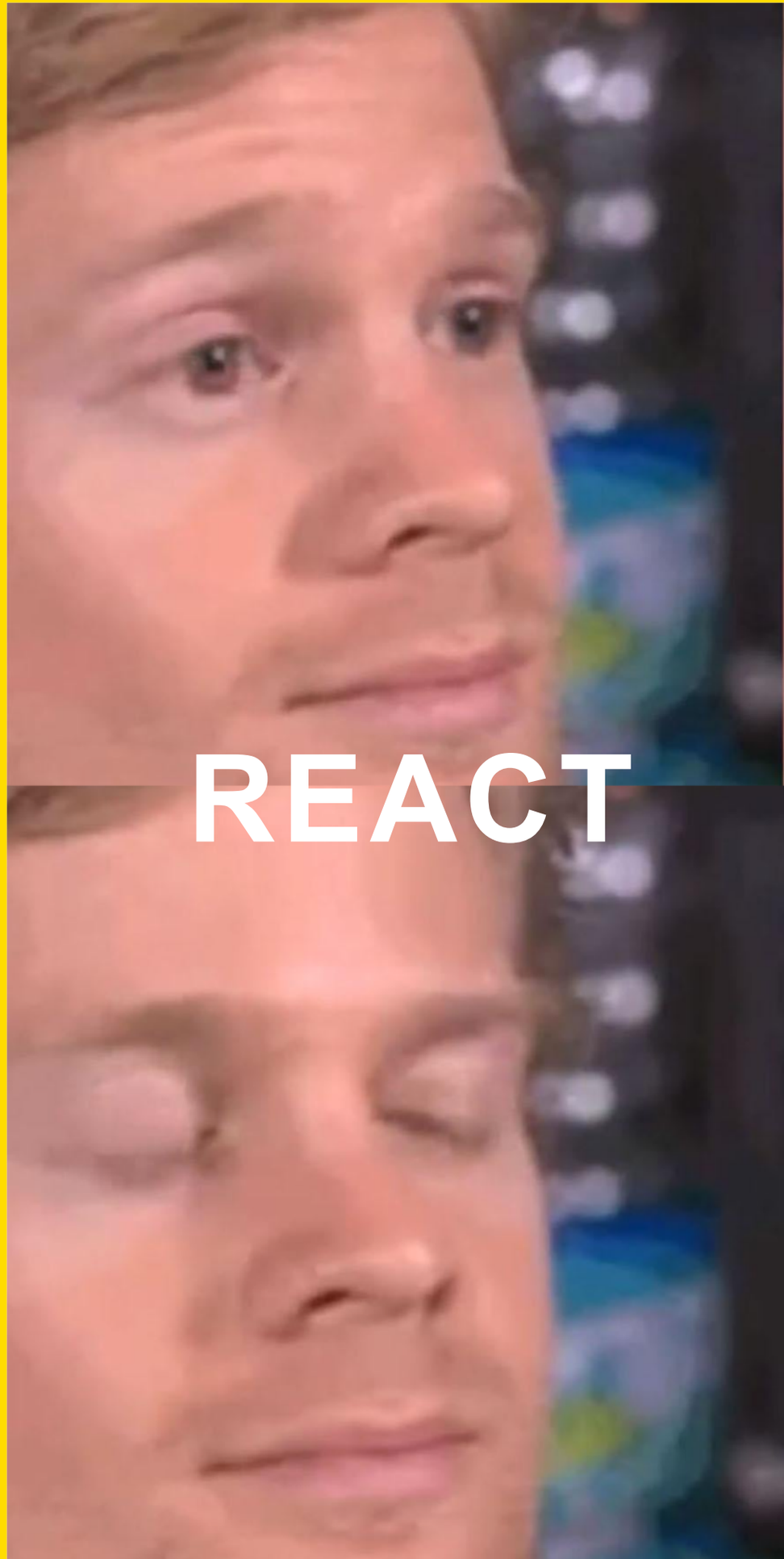
DATA 1700
Webprogrammering

React

Institutt for informasjonsteknologi

OSLO METROPOLITAN UNIVERSITY
STORBYUNIVERSITETET



VIKTIG INFORMASJON

1. Denne forelesningen blir tatt opp

2. React.js er **ikke** en del av pensum

3. Dette er en veldig minimal demonstrasjon av React. React er mye kraftigere enn det som dere vil se her.

Hva trenger du å installere?

Node.js + NPM

Windows / Mac

1. https://nodejs.org/en/download/
2. Last ned Windows / Mac versjon
3. Kjør installasjonsfil

Mac (with Brew)

$ brew install node

Debian / Ubuntu / Mint

$ sudo apt install nodejs npm

CentOS / Fedora / RedHat

$ sudo dnf install npm

Arch / Manjaro

$ sudo pacman -S npm

OpenSUSE

$ sudo zypper install npm

Hva er React?

- Bibliotek for webapplikasjoner → KLIENT!
- React.js vs React Native
  - React.js er et JavaScript-basert bibliotek for webapplikasjoner
  - React Native er et rammeverk for applikasjoner (eks: iOS og Android)
- Single-Page Applications (SPA)
- Komponenter og state

Hva med Angular? Vue? Svelte?

- Flere biblioteker og rammeverk → ditt valg!
- Jobbmarked:

Job Board	Angular	React	Vue
Linkedin Jobs	136,600	154,264	95,784
Indeed	16,370	61,799	3,780
Glassdoor	28,780	54,715	91
Monster.com	14,335	11,316	1,502
Stackoverflow jobs	363	875	169
ZipRecruiter	31,315	55,376	8,210
SimplyHired	12,134	31,512	3,092
Naukri	14,904	8,981	954
Hacker News jobs	1,027	5,195	603
Sum	**255,828**	**384,033**	**114,185**

Kilde

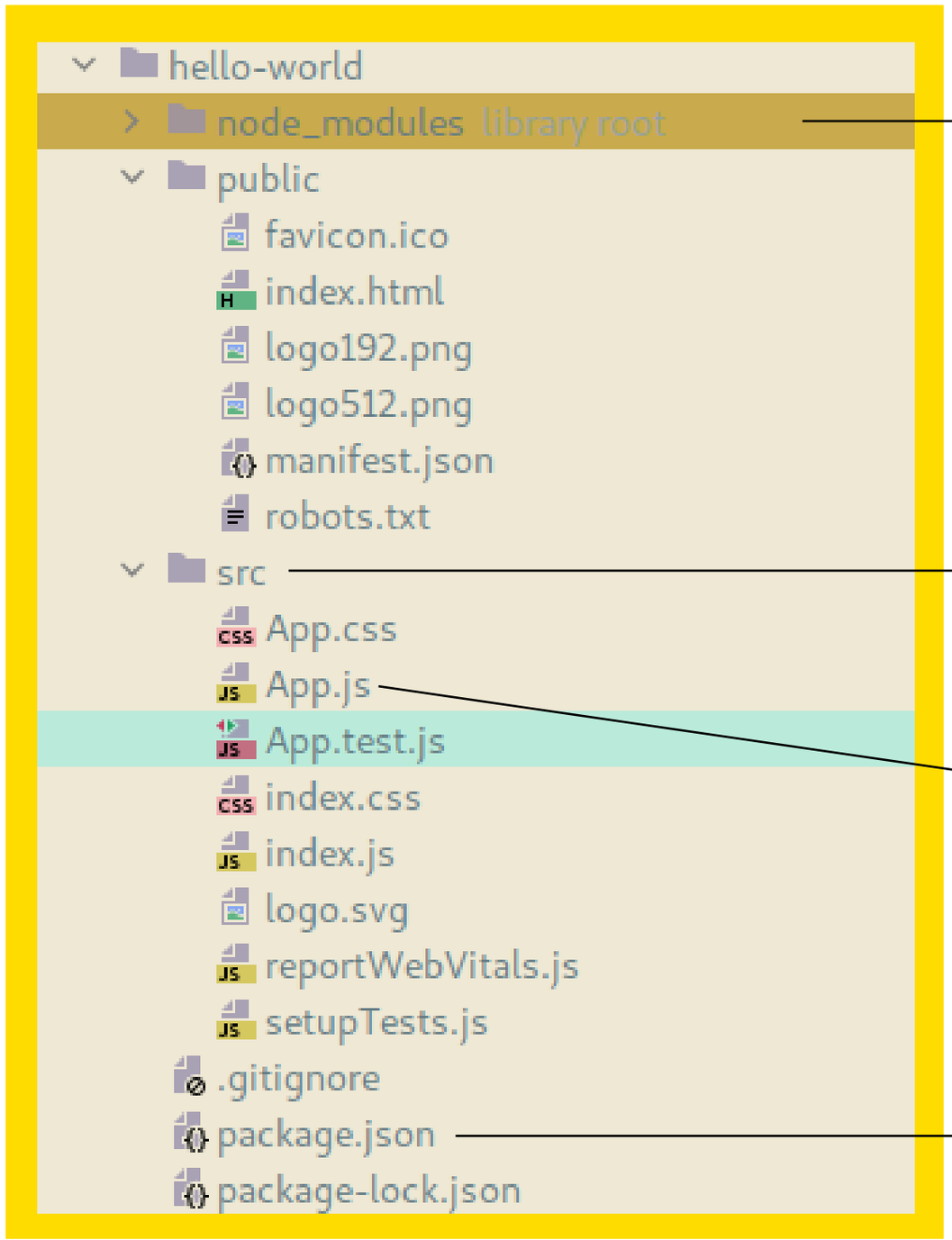
Oppsett av et React prosjekt

1. Velg en mappe for ditt React prosjekt
2. Åpne terminalen
3. Naviger til mappen (`$ cd mappe1/mappe2/valgtmappe`)
4. `$ npx create-react-app app-navn`
5. Wait. A lot...

Interlude – JSX og TSX

- Vi bruker JSX → HTML i JavaScript-filer!
- Men vi kan få konflikter pga. «reserverte» ord i JavaScript. Vi må erstatte HTML-nøkkelord:
  - class → className
  - for → htmlFor
- React kan brukes med TypeScript (eller en mix av JS og TS).
  - JSX : JS = TSX : TS

Mappestruktur



Moduler, aka “ekstra greier”.
F.eks: Bootstrap kommer hit

Source, hvor dev-filene (de som man endrer) er.

App.js, ekte app. Alt blir bygget herfra.

Package.json, viktig metadata om applikasjon. Inkl. dependencies.

Nå er det React!

6. Inn i `src` lager vi en ny mappe `components`

7. I `components` lager vi en ny fil (eks: `Component.js`)

```jsx
import React from "react";

export const Component = () => {
  return (
    <div>
      <button>Si hei!</button>
      <div id="output"></div>
    </div>
  );
};
```

Må importere React i **alle** komponentfiler

Eksporter her for å kunne importere i andre komponenter.

Hver komponent må ha en "wrapper".
Kun 1 element returneres.

En `const` som returnerer noe?

Ja! Dette er faktisk en «arrow function».
Som en lambda funksjon i Java.
Dette er nåværende konvensjon for React.

Arrow functions?

```
function myfunction(par1) {
    return (“This is par1”, par1);
}
```

```
const myfunction = (par1) => {
    return (“This is par1”, par1);
}
```

```
function myfunction2() {
    return “No parameters!”;
}
```

```
const myfunction2 = () => {
    return “No parameters!”;
}
```

Component i App.js

8. Importer komponenten(e) i `App.js` og bruk dem som vanlige HTML tags

```
import { Component } from 'react';

function App() {
  return (
    <div className="App">
      <Component />
    </div>
  );
}

export default App;
```

Og nå kan vi se hvordan appen ser ut med `$ npm start` → `localhost:3000`

Hold up! Hvor er event?

9. Lag en funksjon hvor vi skal gjøre en GET request

```
const siHei = () => {
  //Hvordan lager vi en GET request?
};

export const Component = () => {
  return (
    <div>
      <button onClick={siHei}>Si hei!</button>
      <div id="output"></div>
    </div>
  );
};
```

Hva er den { } greia?

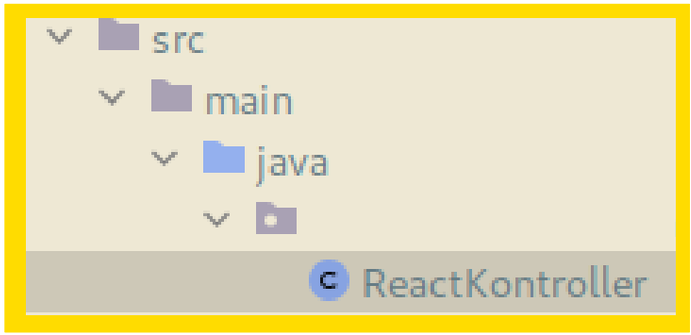
Koden som er inni `return` er i HTML.
For å bruke JS i `return`, må vi «escape» den.
{ } indikerer at inneholdet er JavaScript kode, ikke HTML.

`siHei` er navnet til variabelen med JavaScript funksjonen!

Men hvordan kan vi lage en GET request?

Først, serveren

10. Lag kontrolleren → @RequestMapping gir vår API sin egen URL




```
@RestController
@RequestMapping("/api")
public class ReactKontroller {
    @GetMapping("/hello")
    public String siHei(){
        return "Hello World";
    }
}
```

N.B.: I denne forelesningen bruker vi ikke en database. Hvis du vil ha en, er det bare å lage Repository som vanlig.

Vi trenger Axios!

11. Installer Axios med `$ npm install axios`

- Axios er en HTTP request handler

- Flere forskjellige mulige dependencies → valgte Axios fordi den er lett og enkel å bruke

GET-request time!

12. Lag GET request

13. Legg informasjon inn i `output div`

```
import axios from "axios";
const API_URL_HELLO = "http://localhost:8080/api/hello";

const siHei = () => {
  axios
    .get(API_URL_HELLO)
    .then((response) => {
      document.getElementById("output").innerHTML = response.data;
    })
    .catch((err) => {
      console.error(err);
    });
};
```

Importer axios

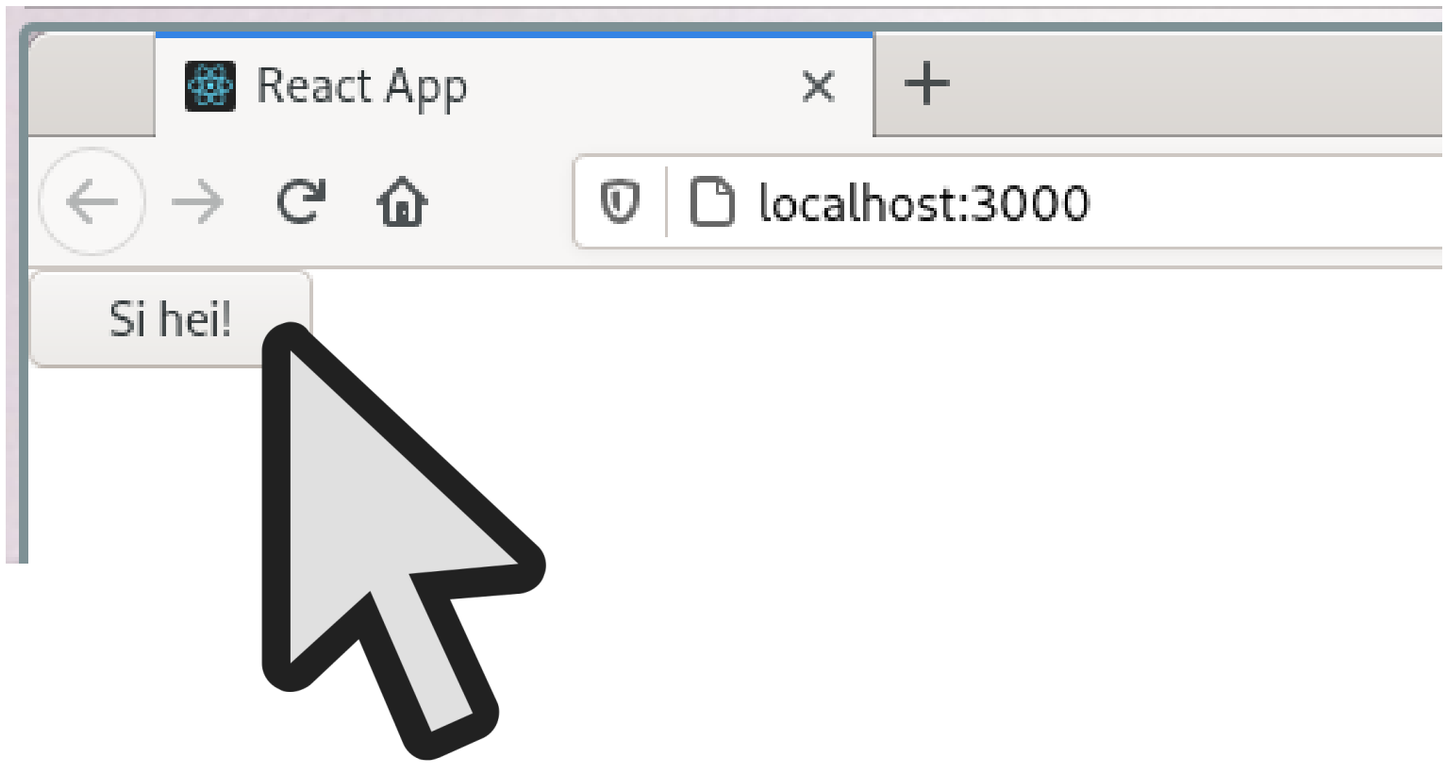
Lag GET request

Når request er fullført, sett response inn i output div

Handle eventuelle feil

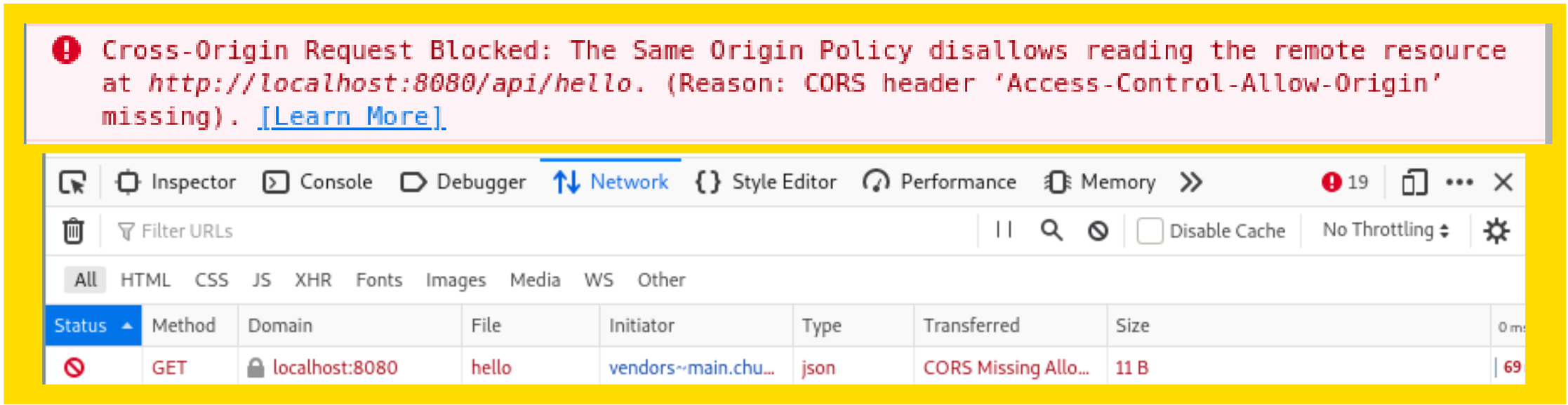
Nå kjører vi serveren og klienten

14. Start Spring Boot serveren
15. Start klienten med `npm start`



CORS

16. Cry.



- CORS = Cross-Origin Resource Sharing
- localhost:3000 og localhost:8080 er forskjellige adresser
- Sikkerhet → blokkerer requests som ikke kommer fra serveren sin adresse

Si OK til CORS

17. Endre Java Controller slik at vi tillater Cross-Origin Resource Sharing

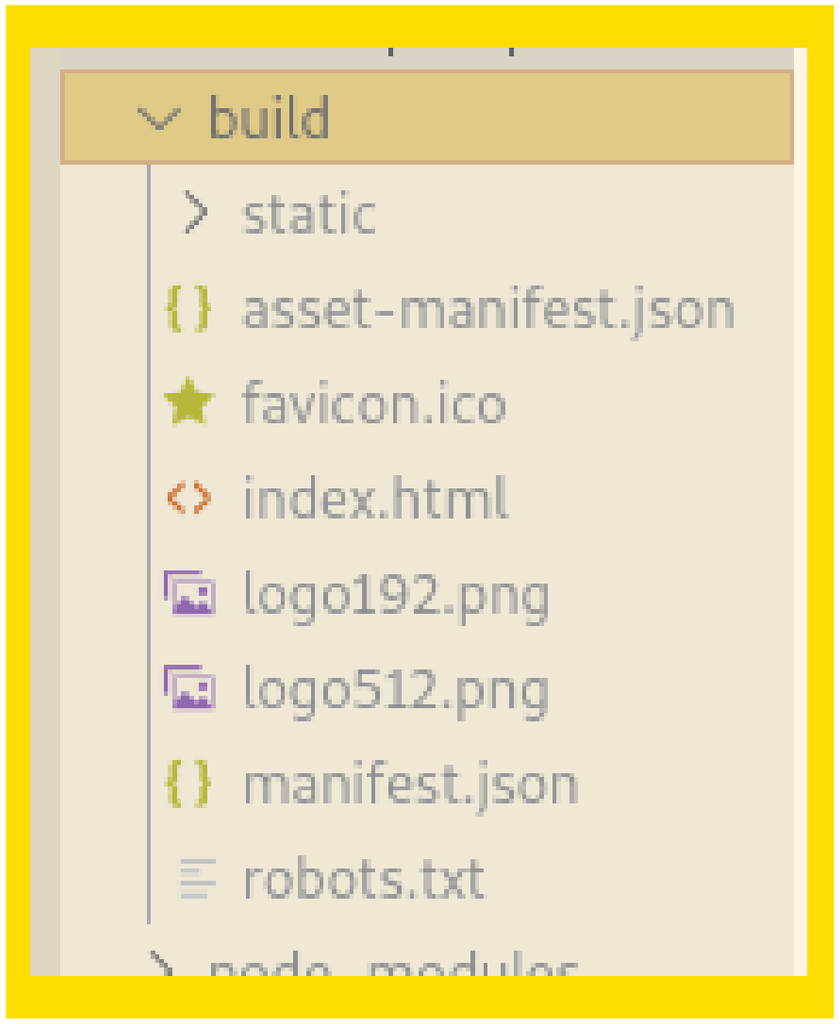
```java
@RestController
@RequestMapping("/api")
public class ReactKontroller {


    @CrossOrigin
    @RequestMapping(method = RequestMethod.GET, path = "/hello")
    public String siHei(){
        return "Hello World";
    }
}
```

Advarsel: standard policyer for CrossOrigin tillater forespørsel fra ALLE kilder. Ikke veldig sikker. Sjekk alternativer her: https://www.baeldung.com/spring-cors

Ferdig!

Når du er ferdig, kan du bygge en versjon for produksjon med **`npm run build`**



Noe ekstra om React

- Kan bruker med jQuery, Bootstrap, osv.
- Høyere abstrasjonsnivå
  - Kortere (og ofte mer lesbar) kode, mindre frihet
- Er fantastisk for dynamisk bygging av nettsider
  - Vil du oppdatere kun hovedinnhold av nettsiden, men ikke nav/footer? React!
  - Vil du legge flere produkter dynamisk til en nettside? React!
  - Vil du ha informasjon som endres dynamisk og vises på nettsiden? React!

Ekstra: State

- State = objekt som inneholder redigerbare data → Live updates!
  - Data kan kun sendes fra parent til child
- Hvordan bruker man det?

I. Import useState
II. Lag en variabelen med verdien og en «setter» (setter endrer verdien)
III. Lag en funksjon som definerer «setter» sin oppførsel
IV. Kall funksjonen inni eventen som utfører endringer
V. Bruk verdien i HTML

```
import React,{useState} from “react”;

export const Component = () => {
   const [counter, setCounter] = useState(0);
   const incrementer = () => {
      setCounter((prev) => prev + 1);
   };
   return (
      <div>
         <p>Counter: {counter};</p>
         <button onClick={incrementer}>Click</button>
      </div>
   );
};
```

Ekstra: Props

- Props (*i.e.* «properties») fungerer som vanlige HTML attributter, men blir laget av utvikleren selv
- Kan brukes for å sende data mellom komponenter
  - Data kan kun sendes fra parent til child!
- Hvordan bruker man det?

I. Lag en variabel i parent component

II. Legg til en prop i component-tag

III. Gi variabelen som verdi til prop'en

IV. I child component, ta inn props som parameter

V. Bruk prop inni HTML (husk { }!)

App.js

```
function App(){
    const myVariable = “name”;
    return (
        <Component myProp={myVariable}>
    );
};
```

Component.js

```
const Component = (props) => {
    return (
        <p>{props.myProp}</p>
    );
};
```

Ekstra: Mapping

- Så du har et array. Og du vil lage en Component med hver element i arrayet. Da kan du bruke mapping!

- Hvordan bruker man det?

I. Lag et array (eller få en fra serveren / parent component)
II. Bruk array.map funksjonen
III. For hvert element i arrayet, legg inn ønsket informasjon i HTML

```
const Messages = () => {
    const messages = [
        {name: "developer", msg: "hello world"},
        {name: "Simon", msg: "hello darkness"},
        {name: "Garfunkel", msg: "my old friend"}
    ];

    return (
        <div>
            {messages.map(message => (
                <p>{message.name} said: “{message.msg}”</p>
            ))}
        </div>
    );
};
```