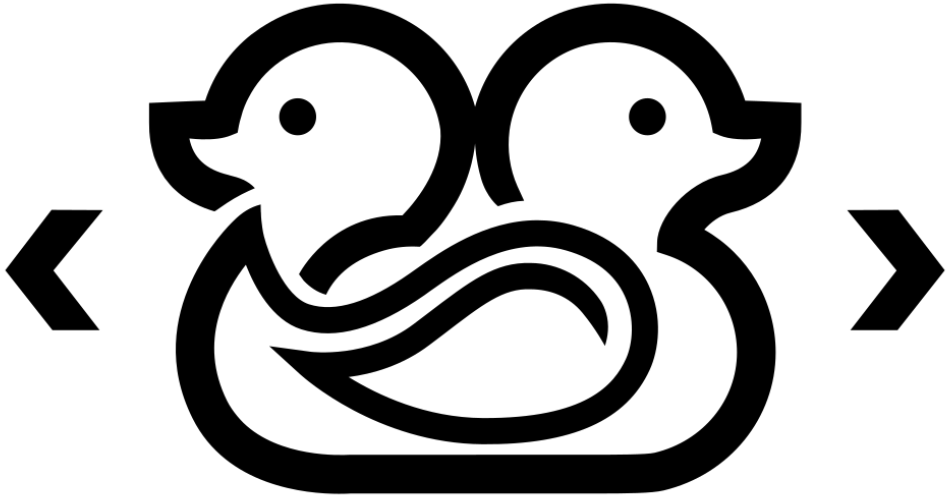


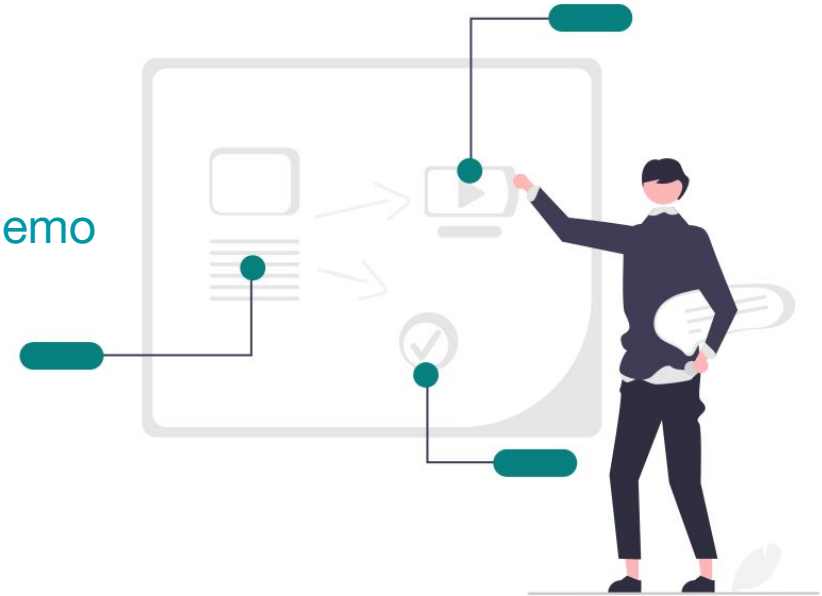
RUBBERDØK PRESENTERER:

Kræsjkurs i webutvikling

github.com/rubberdok/webdev-demo

Plan

- Struktur i webapplikasjoner
- Frontend
  - Interaktivt React-eksempel!
    github.com/rubberdok/webdev-demo
- Backend
  - Backend-eksempel
- Andre verktøy



- Pause ca. 15:00 — 15:15
- Ferdig 16:00

Typisk struktur i webapplikasjoner

Frontend ⟷ **API** ⟷ **Backend** ⟷ **Database**

På klienten — *På serveren*

Frontend

HTML

Struktur


```html
<html>
    <head>
        <title>My website</title>
    </head>
    <body>
        <h1>Welcome!</h1>
        <button class="my-btn">Click me!</button>
    </body>
</html>
```

Welcome!

Click me!

CSS

Utseende

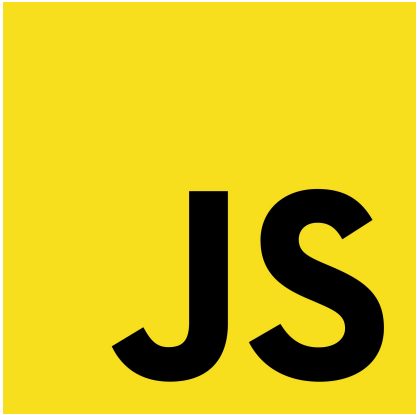

```css
h1 {
  font-family: "Arial";
  color: sienna;
}

.my-btn {
  background-color: sandybrown;
  border-radius: 10px;
}
```

Welcome!

Click me!

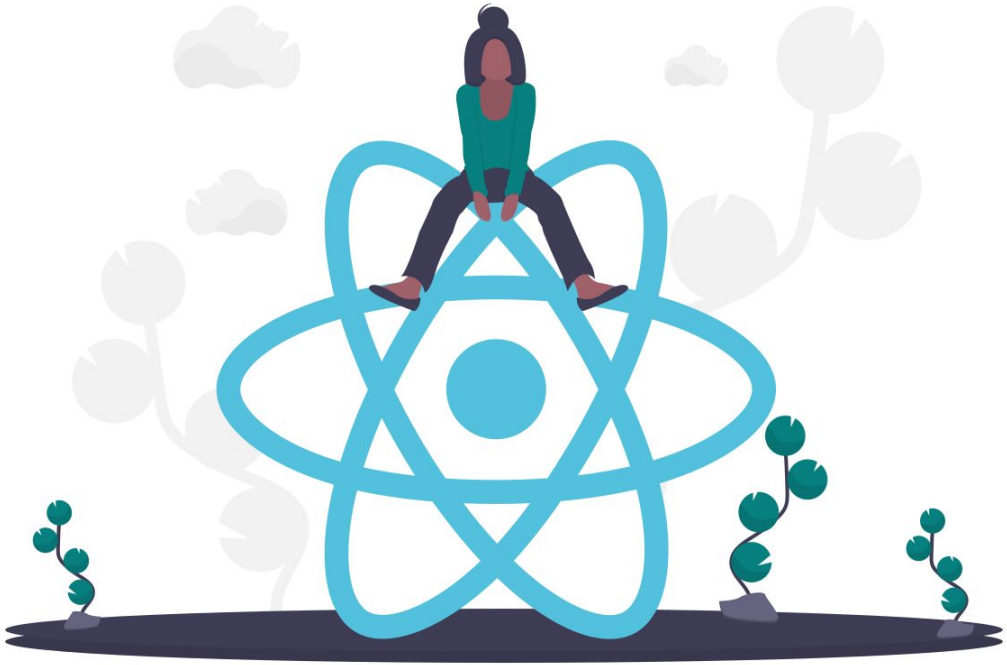
JavaScript

Interaksjon



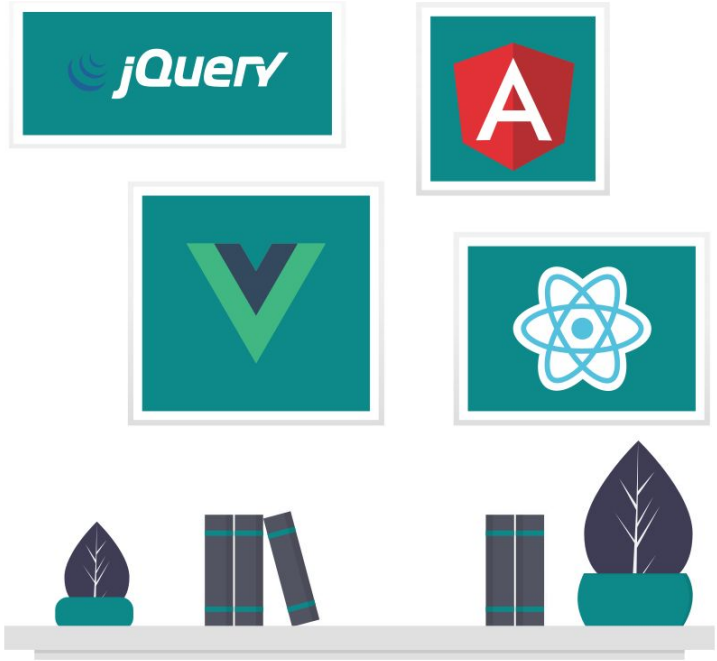

```
const header = document.querySelector("h1");
const button = document.querySelector(".my-btn");

function setGoodbye() {
  header.innerText = "Goodbye!";
}

button.addEventListener("click", setGoodbye);
```

Frontend-rammeverk



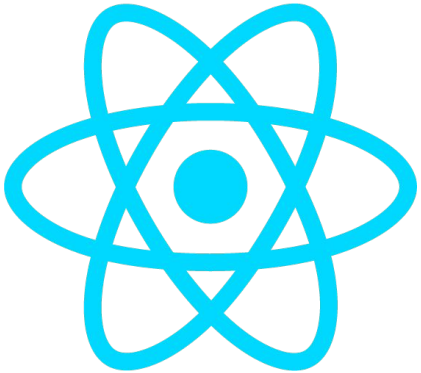
Frontend-rammeverk

- Hvorfor rammeverk?
  - **Struktur** og **interaksjon** er ofte tett koblet i applikasjoner
  - Synkronisere **state** og **rendering**
  - **Komponenter** gir bedre kodestruktur
- Mange alternativer
  - React
  - Vue
  - Angular
  - Svelte
  - ...



React

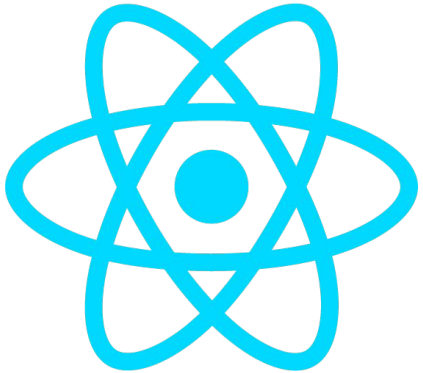
Komponenter



```
function App() {
  const [header, setHeader] = useState("Welcome!");

  function setGoodbye() {
    setHeader("Goodbye!");
  }

  return (
    <>
      <h1>{header}</h1>
      <button onClick={setGoodbye} className="my-btn">
        Click me!
      </button>
    </>
  );
}
```

React

Komponenter

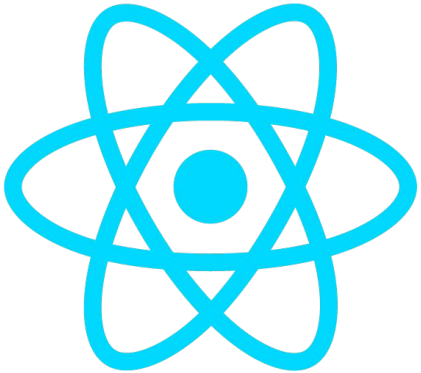


Komponent

```
function App() {
  const [header, setHeader] = useState("Welcome!");

  function setGoodbye() {
    setHeader("Goodbye!");
  }

  return (
    <>
      <h1>{header}</h1>
      <button onClick={setGoodbye} className="my-btn">
        Click me!
      </button>
    </>
  );
}
```

React

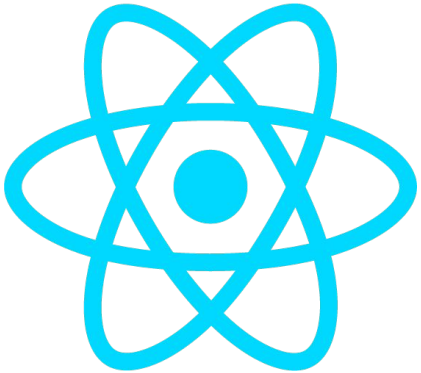
Komponenter



```
function App() {
  const [header, setHeader] = useState("Welcome!");

  function setGoodbye() {
    setHeader("Goodbye!");
  }

  return (                    JSX
    <>
      <h1>{header}</h1>
      <button onClick={setGoodbye} className="my-btn">
        Click me!
      </button>
    </>
  );
}
```

React

Komponenter



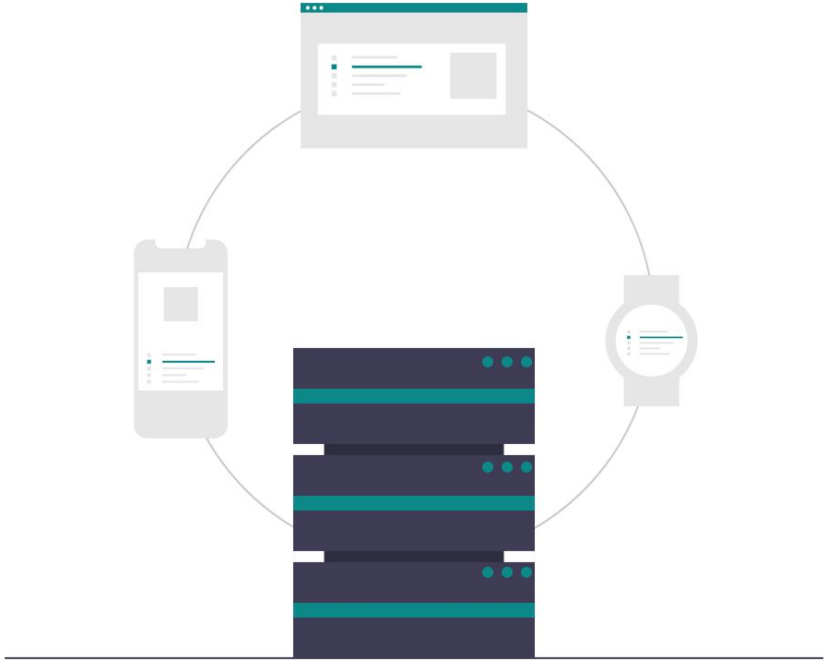
State

```
function App() {
  const [header, setHeader] = useState("Welcome!");

  function setGoodbye() {
    setHeader("Goodbye!");
  }

  return (
    <>
      <h1>{header}</h1>
      <button onClick={setGoodbye} className="my-btn">
        Click me!
      </button>
    </>
  );
}
```

Tid for å progge!

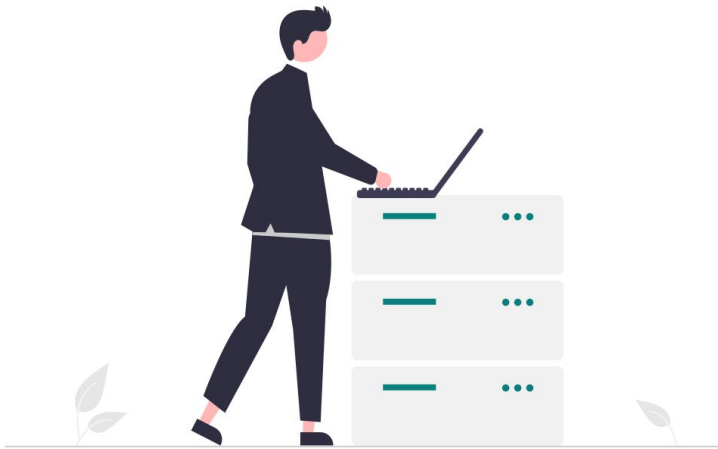
github.com/rubberdok/webdev-demo

Backend



Backend

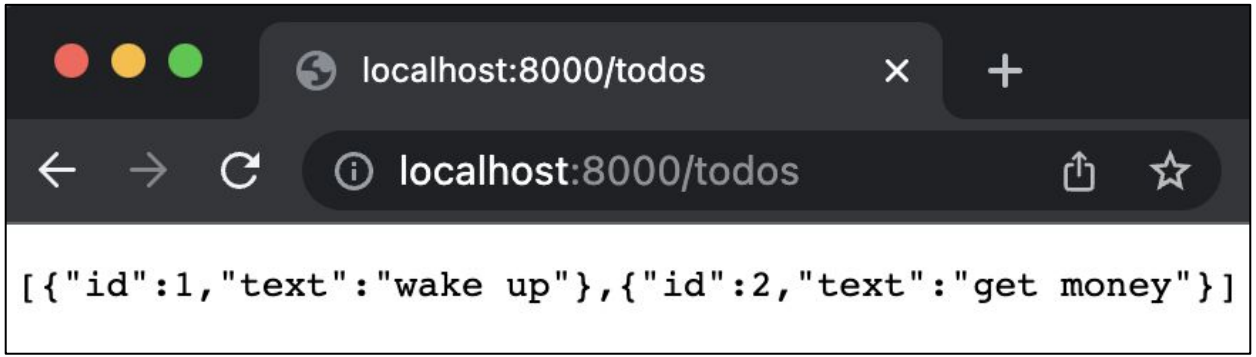
- Hvorfor ha en backend?
  - Lagre og behandle data
    - F. eks. brukersystem
    - Definere **modeller**
  - Vil ikke gi frontend direkte tilgang til databasen
    - Trenger en **“mellommann”**
  - Koble sammen tjenester
    - indokntnu.no → Vipps



API

- Et **grensesnitt** for å kommunisere med backend
- Typisk **REST-API**, men finnes andre typer også

- Består av **endepunkter**
  - F. eks. **/todos**, **/createTodo**, etc.
  - Som en nettside - bare for data!
- Kommuniserer over HTTP
  - **GET** requests, **POST** requests, etc.



Backend-språk og -rammeverk


+ django

Python + Django


+ Express

Node.js + Express


+ 

Java + Spring Boot

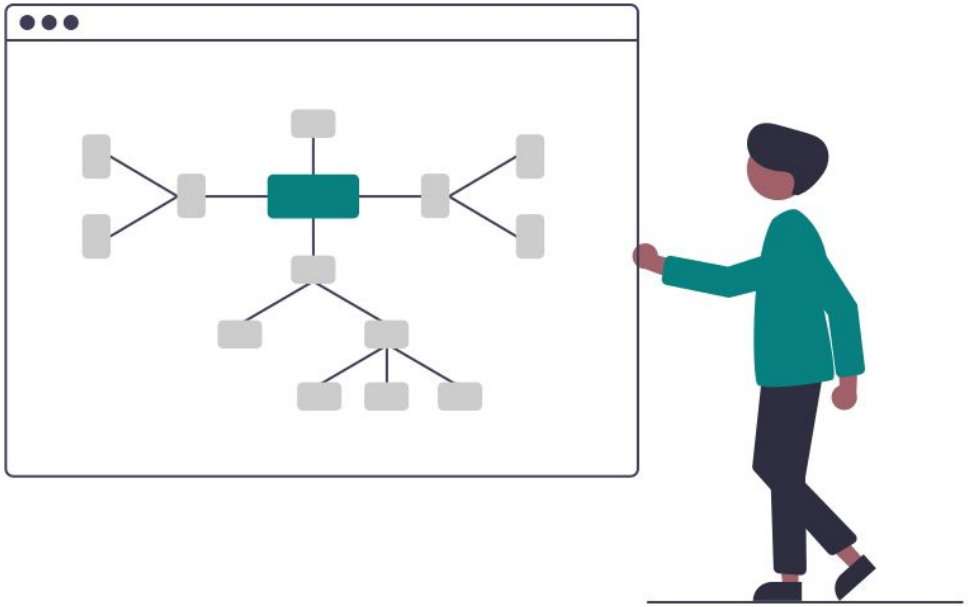
Go + GORM

…og mange fler!

Eksempel: Todo-backend i Go

github.com/rubberdok/webdev-demo

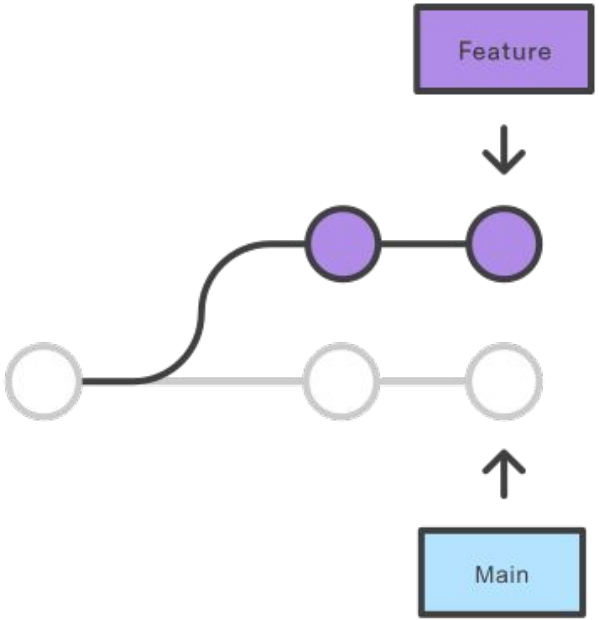
Andre verktøy



Git

Samkjøring av kode

- Lagrer en full historikk av endringer i koden
  - **Commits**
- Gjør det mulig å jobbe parallelt på **branches**



Git

Samkjøring av kode

Typisk Git-workflow:

- *Lag **repository** på GitHub/GitLab*
- `git clone github.com/my-name/my-repo.git`
- `git checkout -b myfeature`
- *Gjør endringer i koden*
- `git add .`
- `git commit -m “did some stuff”`
- `git push`
- *Lag **Pull Request/Merge Request** på GitHub/GitLab*

TypeScript

Type Checking

Vanlig JavaScript:

```
<TodoList todos={"not an array!"} />
```

```
Uncaught TypeError: todos.map is not a function
```

TypeScript

Type Checking

TypeScript:

```
type Props = {
    todos: Todo[];
};

export default function TodoList({ todos }: Props) {
```

```
<TodoList todos={"not an array!"} />
```

```
Type 'string' is not assignable to type 'Todo[]'. ts(2322)
```

Rubberdøk

For mer webutvikling!