

Intro til React

Hva er React?

React = JavaScript-bibliotek

- Brukes til å lage og bygge brukergrensesnitt
- Brukergrensesnittet består av deler som kalles *komponenter*
- En komponent = JavaScript-funksjon

Fordeler med React

- Rask oppstart
- Relativt enkelt å lære
- Høy fleksibilitet
- Gjenbrukbare komponenter
- Mange brukere = stort utviklingsmiljø => Google har alltid svaret
- Fungerer bra med ChatGPT og Copilot

Koncepter i React

Komponenter

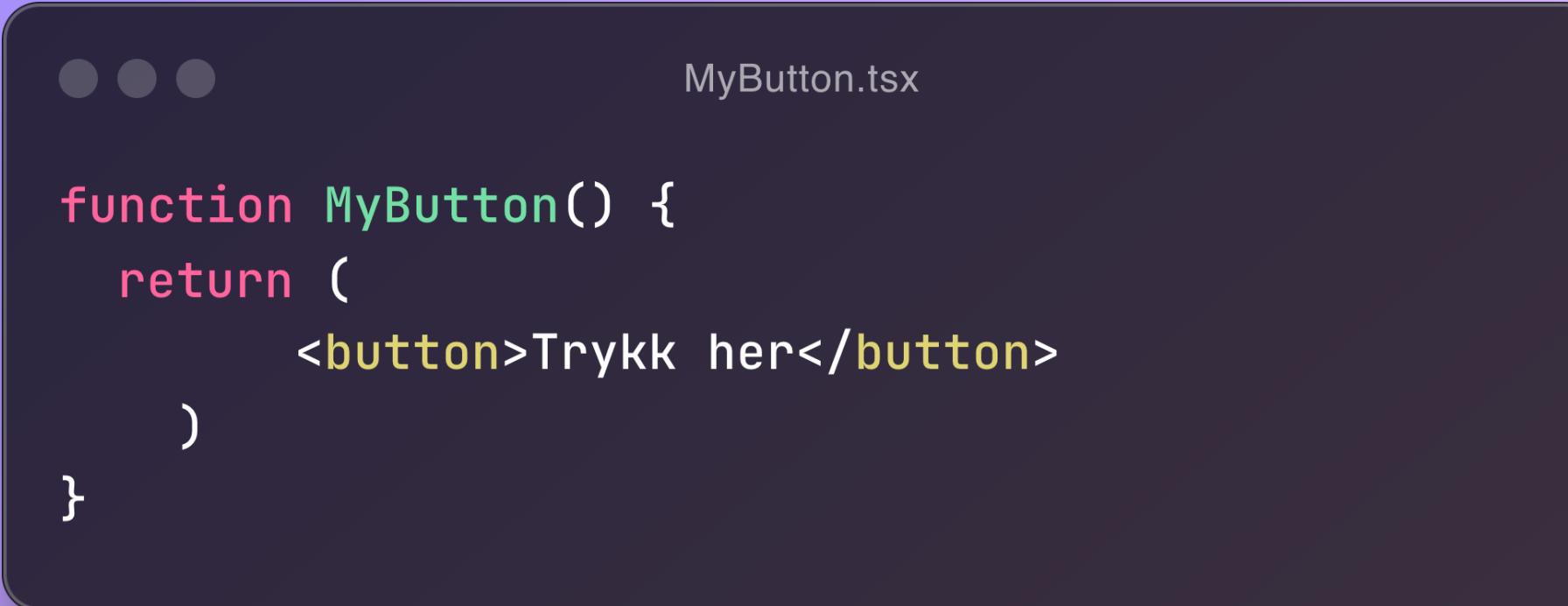


MyButton.tsx

```
function MyButton() {  
  return (  
    <button>Trykk her</button>  
  )  
}
```

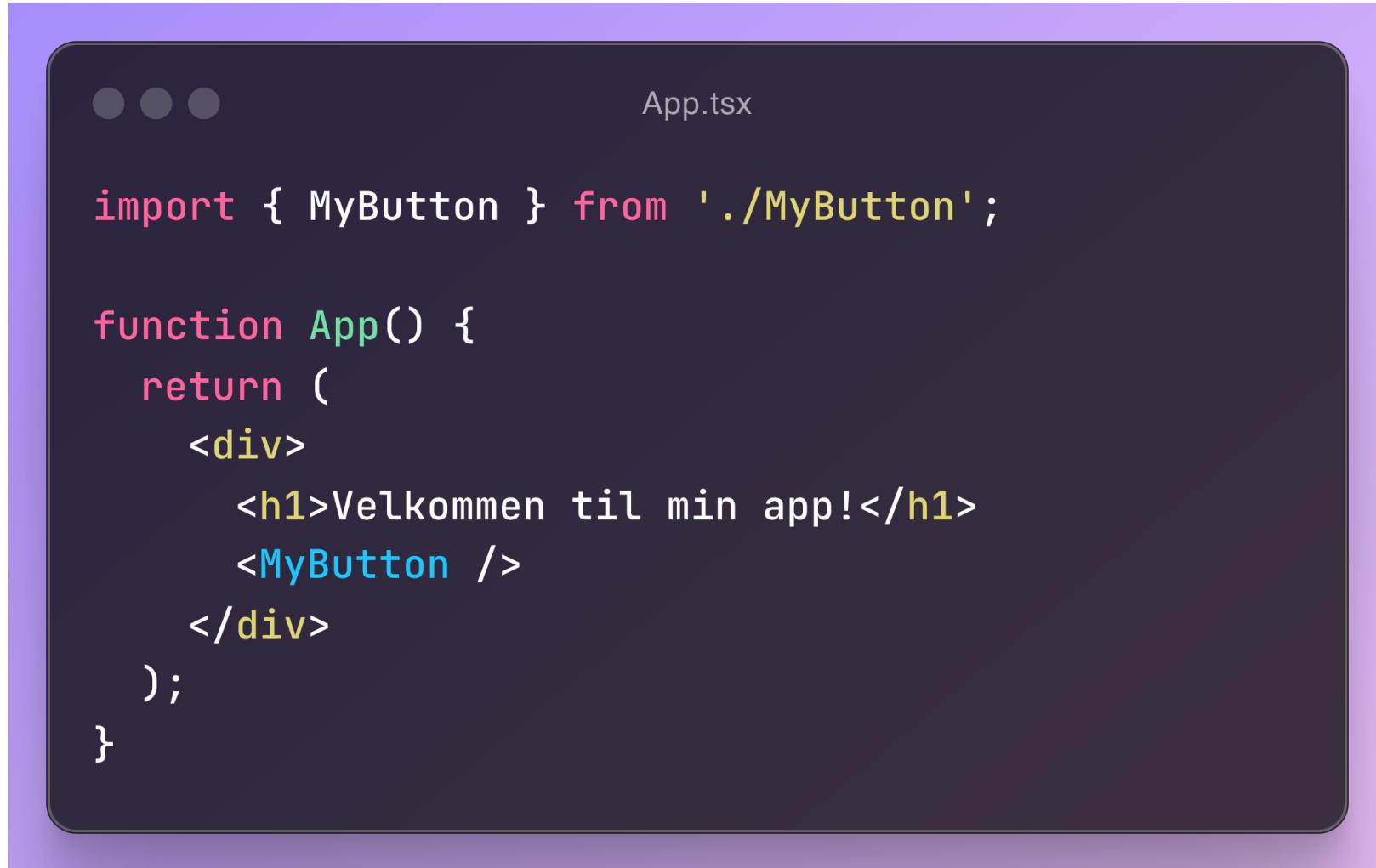
Returnerer JSX

Ligner på HTML



```
function MyButton() {
  return (
    <button>Trykk her</button>
  )
}
```

Ta i bruk en React komponent



The image shows a dark-themed code editor window with a rounded rectangle border. In the top right corner, the file name "App.tsx" is displayed. In the top left corner, there are three small circular icons. The main area contains the following code:

```
import { MyButton } from './MyButton';

function App() {
  return (
    <div>
      <h1>Velkommen til min app!</h1>
      <MyButton />
    </div>
  );
}


```

Props



PersonInfo.tsx

```
export function PersonInfo() {
  return (
    <div>
      <h1>Personinfo</h1>
      <p>Navn: Sara</p>
      <p>Alder: 25</p>
    </div>
  );
}
```

Props

```
PersonInfo.tsx
```

```
export function PersonInfo() {
  return (
    <div>
      <h1>Personinfo</h1>
      <p>Navn: Sara</p>
      <p>Alder: 25</p>
    </div>
  );
}
```

```
App.tsx
```

```
import { PersonInfo } from './PersonInfo';

function App() {
  return (
    <div>
      <h1>Velkommen til min app!</h1>
      <PersonInfo />
    </div>
  );
}
```

Props

```
...  
export function App() {  
  return (  
    <div>  
      <h1>Velkommen til min app!</h1>  
      <p>Personinfo</p>  
      <p>Navn: Sara</p>  
      <p>Alder: 25</p>  
    </div>  
  );  
}
```

Velkommen til min app!

Personinfo

Navn: Sara

Alder: 25

```
</div>  
);  
}
```

Props



PersonInfo.tsx

```
export function PersonInfo({ name, age }) {
  return (
    <div>
      <h1>Personinfo</h1>
      <p>Navn: {name}</p>
      <p>Alder: {age}</p>
    </div>
  );
}
```

Props



PersonInfo.tsx

```
export function PersonInfo({ name, age }) {  
  return (  
    <div>  
      <h1>Personinfo</h1>  
      <p>Navn: {name}</p>  
      <p>Alder: {age}</p>  
    </div>  
  );  
}
```



App.tsx

```
import { PersonInfo } from './PersonInfo';  
  
function App() {  
  return (  
    <div>  
      <h1>Velkommen til min app!</h1>  
      <PersonInfo name="Sara" age={25} />  
      <PersonInfo name="Ola Nordmann" age={23} />  
    </div>  
  );  
}
```

Props

```
...  
export function Personinfo({name, age}) {  
  return (  
    <div>  
      <h1>Personinfo</h1>  
      <p>Navn: ${name}</p>  
      <p>Alder: ${age}</p>  
    </div>  
  );  
}
```

Velkommen til min app!

Personinfo

Navn: Sara

Alder: 25

Personinfo

Navn: Ola Nordmann

Alder: 23

3} />

Pros er ganske likt som attributter i HTML

Det er noen forskjeller. F.eks. brukes *className* i stedet for *class*. Fordi det er et reservert og i JavaScript



Eksempel

```
<img className="avatar" />
```

Lister og gjenbruksbare komponenter

```
App.tsx

import { PersonInfo } from './PersonInfo';

const personalInfo = [
  { name: 'Sara', age: 25 },
  { name: 'Ola Nordmann', age: 23 },
  { name: 'Kari Nordmann', age: 21 },
];

function App() {
  return (
    <div>
      <h1>Velkommen til min app!</h1>
      {personalInfo.map((person, index) => (
        <PersonInfo key={index} name={person.name} age={person.age} />
      ))}
    </div>
  );
}
```

Lister og gjenbruksbare komponenter

```
App.tsx

import { PersonInfo } from './PersonInfo';

const personalInfo = [
  { name: 'Sara', age: 25 },
  { name: 'Ola Nordmann', age: 23 },
  { name: 'Kari Nordmann', age: 21 },
];

function App() {
  return (
    <div>
      <h1>Velkommen til min app!</h1>
      {personalInfo.map((person, index) => (
        <PersonInfo key={index} name={person.name} age={person.age} />
      ))}
    </div>
  );
}
```

Velkommen til min app!

Personinfo

Navn: Sara

Alder: 25

Personinfo

Navn: Ola Nordmann

Alder: 23

Personinfo

Navn: Kari Nordmann

Alder: 21

Children



Eksempel

```
function Card() {  
  return (  
    <div className="card">  
      <Avatar />  
    </div>  
  );  
}  
  
function App() {  
  return <Card />;  
}
```

Children



Eksempel

```
function Card({ children }) {
  return <div className="card">{children}</div>;
}

function App() {
  return (
    <Card>
      <Avatar />
    </Card>
  );
}
```

Children

Eksempel

```
function Card({ children }) {
  return <div className="card">{children}</div>;
}

function App() {
  return (
    <Card> ← Forelder
      <Avatar /> ← Barn
    </Card> ← Forelder
  );
}
```

Conditional rendering

Baser rendring på data

Eksempel

```
if (isLoggedIn) {  
    return <App />  
}  
  
return <LoginInForm />
```

Eksempel

```
return(  
    <div>{isLoggedIn ? <App /> : <LogInForm />}</div>  
)
```

Events



Eksempel

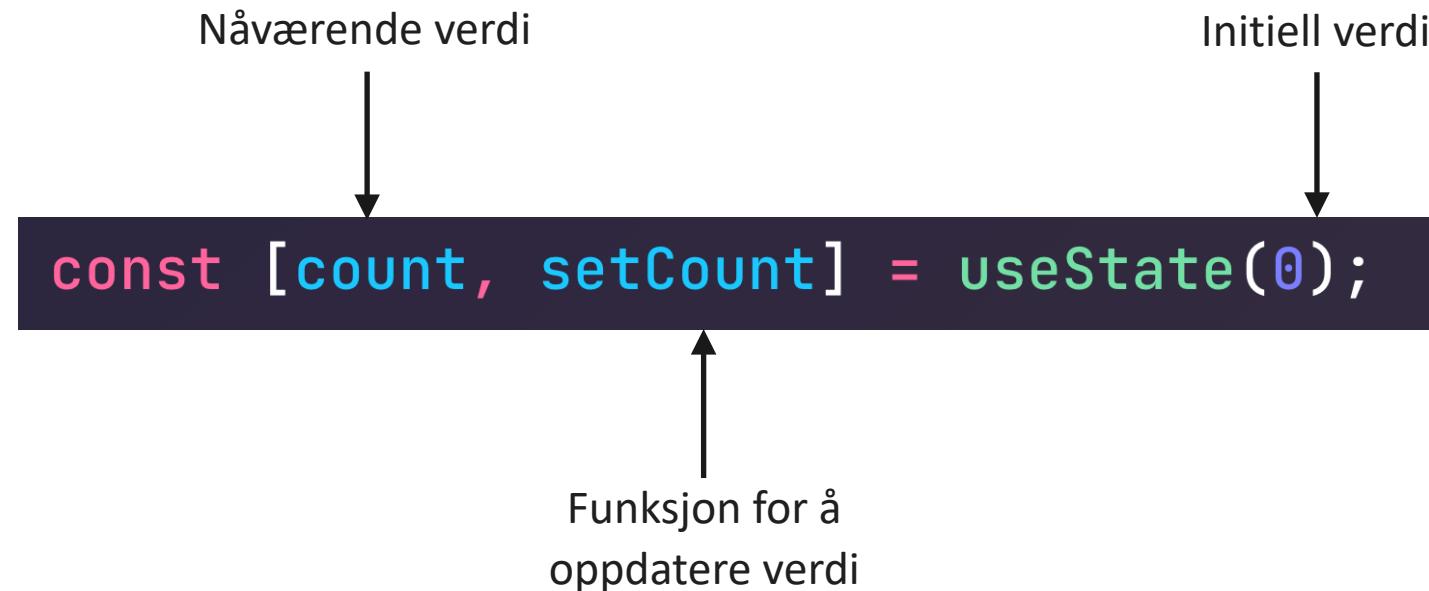
```
export default function Button() {  
  function handleClick() {  
    // Kode som skal skje når knapp trykkes på  
  }  
  
  return (  
    <button onClick={handleClick}>  
      Click me  
    </button>  
  );  
}
```

Hooks

- JavaScript funksjoner
- Starter alltid med ordet “use”
- Må kalles innenfor en komponent eller en annen hook
- Kan ikke kalles innenfor løkker eller if-setninger

useState

Når du trenger å ta vare på en state i komponenten



useState

Når du trenger å ta vare på en state i komponenten

```
useState

import React, { useState } from 'react';

function Button() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <button onClick={() => setCount(count + 1)}>Click me</button>
      <p>Number of clicks: {count}</p>
    </div>
  );
}
```

useEffect

Denne hooken lar deg utføre sideeffekter i funksjonskomponenter, som å hente data, abonnere på en tjeneste eller manuelt endre DOM.

```
useEffect(() => {
  console.log(`Count: ${count}`);
}, [count]); // Effekt kjører hver gang `count` eller `text` endrer seg
```

useEffect

Bruktes også til å trigge side-effekter når:

- Props endrer seg
- Komponenten blir tatt i bruk
- Komponenten tas vekk fra grensesnittet

```
useEffect(() => {
    // Kjører første render og når state eller props endres
}, [state, props]);

useEffect(() => {
    // Kjører første render
}, []);

useEffect(() => {
    // Kjører hver render
});
```

Datahenting

```
useEffect(() => {
  const fetchData = async () => {
    const result = await fetch(
      'https://hn.algolia.com/api/v1/search?query=redux'
    ).then((response) => response.json())

    setData(result);
  };

  fetchData();
}, []);
```