

DS3103 Webutvikling

Context

Rolando Gonzalez 2022

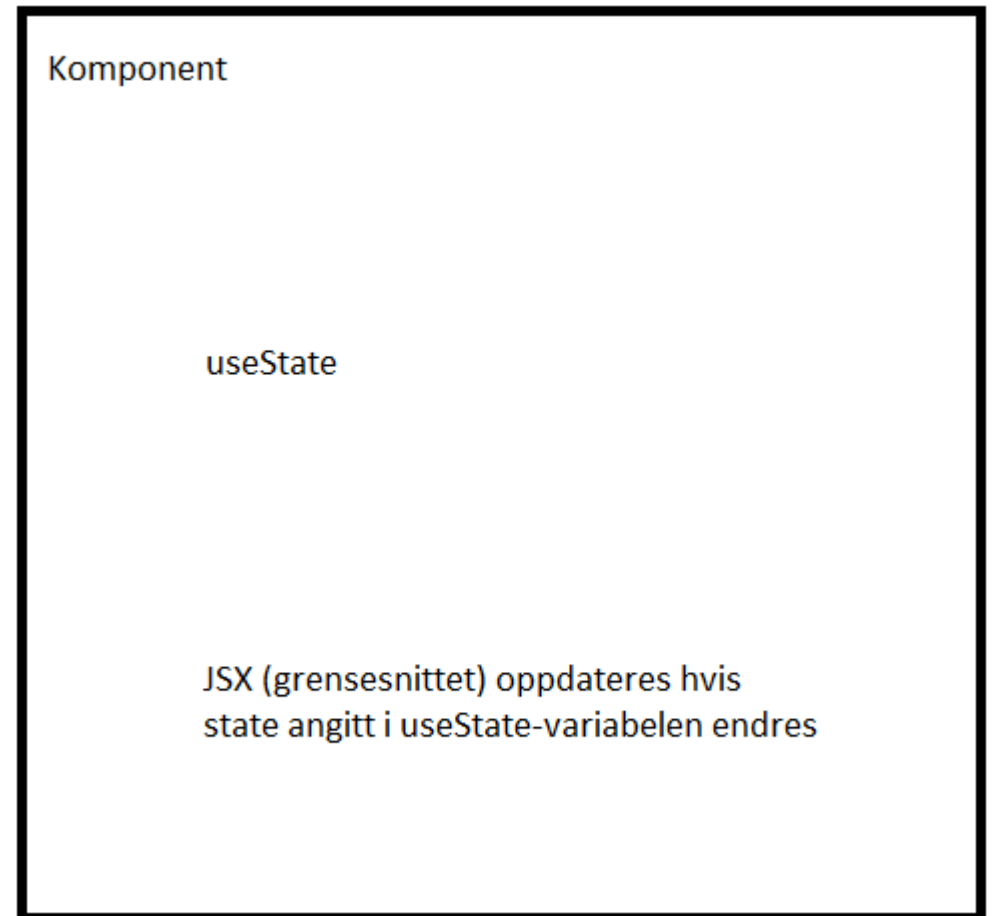
Innhold

- Bakgrunn
 - 1 komponent og dens state (useState)
 - List og Item hvor List har state (useState) og bruker denne for å generere Item
 - Komponenter med «tipp-oldebarn»: «props drilling» (ikke bra!)
- Hva er Context?
- Kodeeksempel Context med JavaScript
- Kodeeksempel Context med TypeScript

Bakgrunn

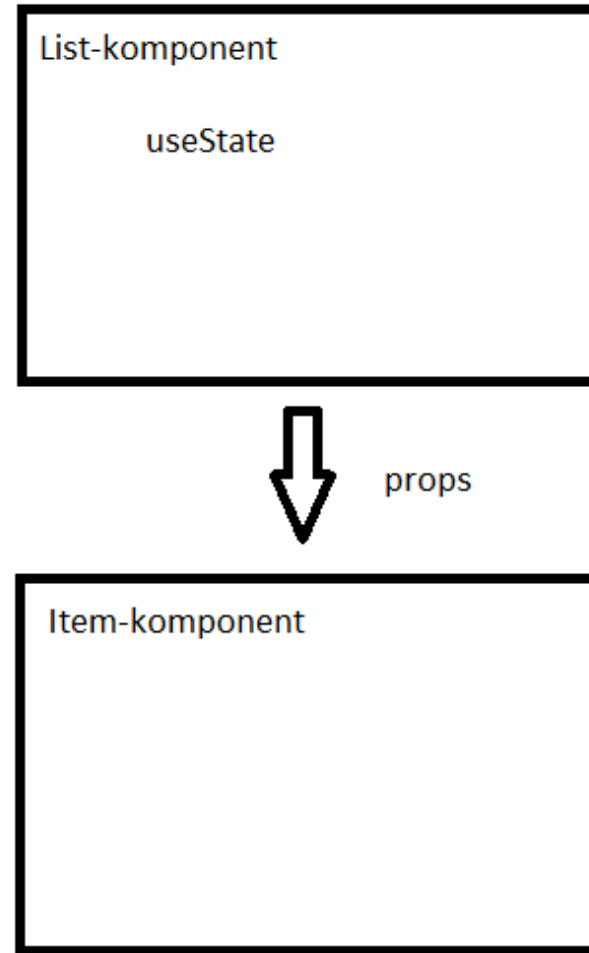
State, informasjon og automatisk rerendring

- En komponent som har informasjon hvis endringer skal lede til endringer i grensesnittet trenger en state.
- En komponent bruker for eksempel `useState` for å ha en tilstand.
- Hvis tilstanden i `useState` endres så oppdateres grensesnittet til komponenten for å vise oppdatert informasjon.



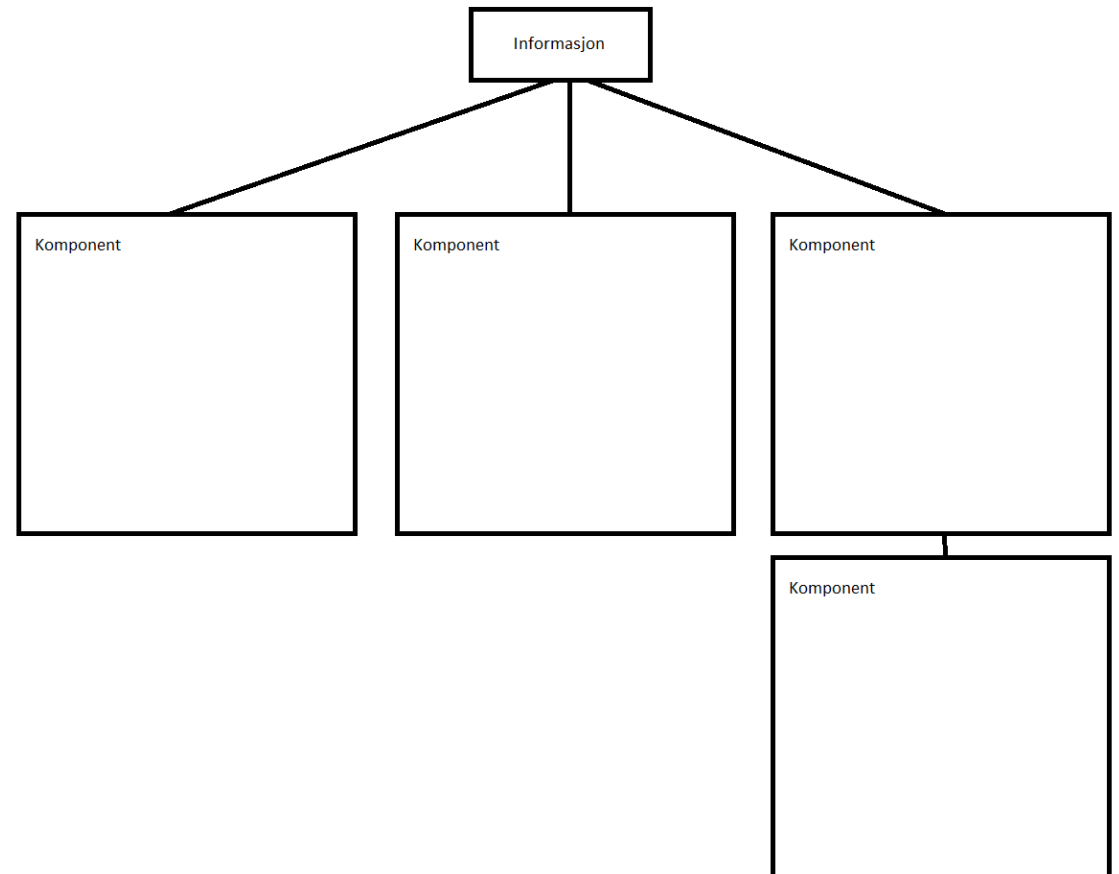
State, informasjon og automatisk rerendring

- Når en List-komponent har informasjon i en state og sender props til en barnekomponent vil barnekomponenten (Item) automatisk endres hvis List-komponentens state endres.



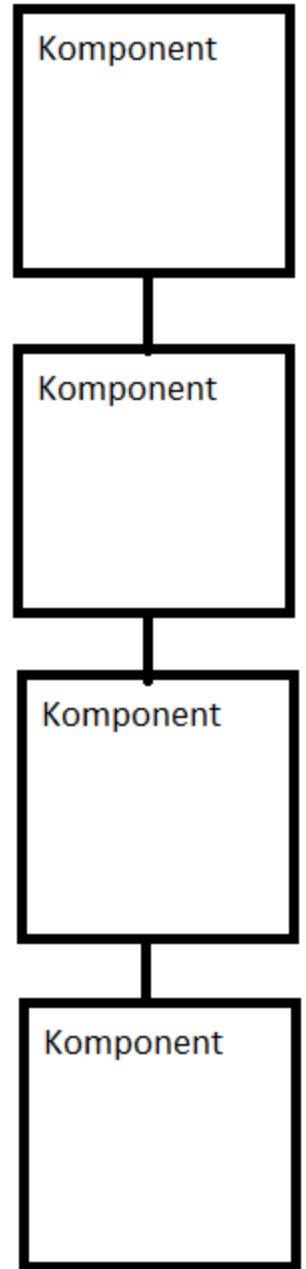
Hva med mange uavhengige komponenter?

- Men... hvordan skal man løse problematikken med at flere uavhengige komponenter (noen med barnekomponenter) trenger tilgang til samme informasjon og de skal oppdateres automatisk når informasjonen endres på?



Hva med komponenter som er barn av barn...osv.

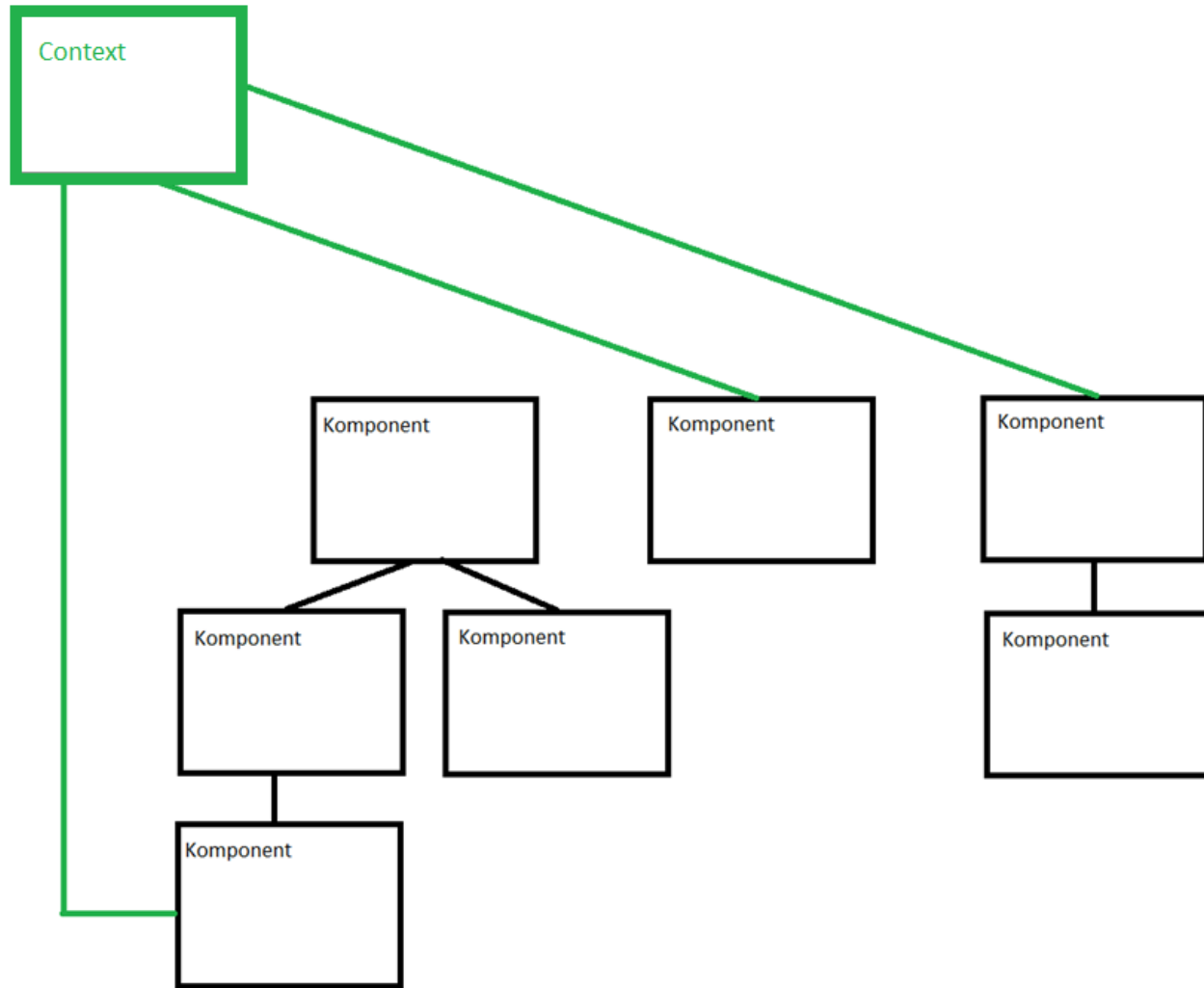
- En annen utfordring som danner bakgrunnen for problemstillingen er det som heter «props drilling».
- Hva om man har komponenter som har barn som har barn som har barn i nte. Forelderkomponenten har en del informasjon som kanskje alle barna trenger, men så er det kanskje noe informasjon som kun et spesifikt barneelement trenger.
- En løsning er å sende props nedover hele veien, men hvis det blir for mye av dette skapes det veldig sterke avhengigheter mellom flere komponenter og ledd av komponenter – «props drilling»



Context

Hva er Context?

- Context er en teknikk i React for å tilgjengeliggjøre data og tilstand globalt (innenfor et definert scope)
- Context blir direkte tilgjengelig for komponentene som trenger informasjon (og å oppdateres automatisk hvis tilstanden på info endres).
- Offisielle nettsider:
<https://reactjs.org/docs/context.html>



Hovedkoder

- **Context:** Context-objektet
- **createContext:** kode for å skape ny Context
- **Provider:** Komponent som man bruker for å tilgjengeliggjøre Context til komponentene - inneholder (tilgang til) informasjon og har state.
- **useContext:** kode som brukes av komponenter som gjør bruk av Context; slik kobles det opp til Context!

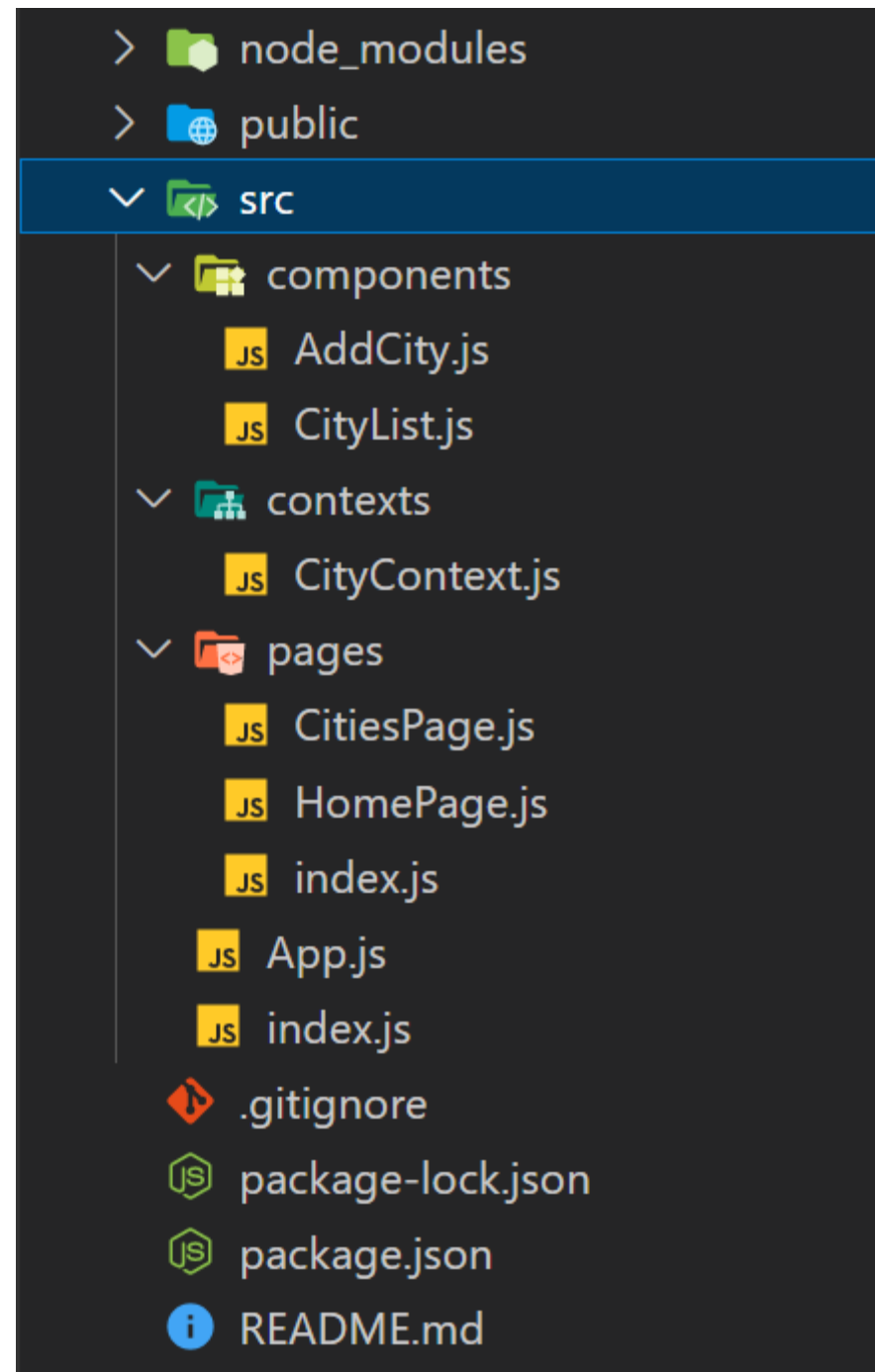
Kodeeksempel Context med JavaScript

Boilerplate og installering

- React Boilerplate:
 - `npx create-react-app context-test`
- React Router for navigering:
 - `npm install react-router-dom`

Oppsett

- I dette eksempelet er det brukt react-router-dom også for å vise hvordan informasjonen blir behandlet hvis man hopper mellom nettsider.
- Totalt:
 - 2 components
 - 1 context
 - 2 pages med index.js for å gjøre enklere å eksportere/importere
 - 1 App.js
 - 1 index.js



Context

- Importerer useState og createContext.
- Inneholder 1 Context-objekt og 1 Provider-komponent.
- useState fylles med informasjon som skal være tilgjengelig global
- Kan legge til funksjoner for å endre state.
- Returnerer Provider-komponent med value (det som skal være tilgjengelig fra komponenten)
- Inneholder {children} som er komponentene som skal få Provider

JS CityContext.js X

context-test > src > contexts > JS CityContext.js > ...

```
1  import { useState, createContext } from "react";
2
3  export const CityContext = createContext();
4
5  export const CityProvider = ({children}) => {
6
7      const [cities, setCities] = useState([
8          "Oslo",
9          "Stockholm",
10         "London",
11         "Paris"
12     ]);
13
14     const addCity = (newCity) => {
15         setCities( [ newCity, ...cities ] );
16     }
17
18     return (
19         <CityContext.Provider value={ { cities, addCity } }>
20             {children}
21         </CityContext.Provider>
22     )
23
24 }
```

HomePage

- HomePage er en vanlig komponent.

```
JS HomePage.js X
context-test > src > pages > JS HomePage.js >
1  const HomePage = () => {
2
3      return (
4          <section>
5              <h1>Hjem</h1>
6          </section>
7      )
8  }
9
10 export default HomePage;
```

CitiesPage

- CitiesPage importerer AddCity.js og CityList.js og gjør bruk av dem (de kommende slidesene viser koden for dem).

```
JS CitiesPage.js X JS AddCity.js JS CityList.js
context-test > src > pages > JS CitiesPage.js > ...
1  import AddCity from '../components/AddCity';
2  import CityList from '../components/CityList';
3
4  const CitiesPage = () => {
5    return (
6      <section>
7        <h1>Byer</h1>
8        <AddCity/>
9        <CityList/>
10     </section>
11   )
12 }
13
14 export default CitiesPage;
```


Index.js

- Denne filen skal gjøre det enklere å importere de x antall Page-komponentene man har.

```
JS index.js X
context-test > src > pages > JS index.js
1 import CitiesPage from "../CitiesPage";
2 import HomePage from "../HomePage";
3
4 export { CitiesPage, HomePage };
```

App.js

- App.js har her ansvaret for React Router og å wrappe de komponentene som skal ha tilgang til Context – her får også HomePage tilgang selv om den ikke trenger det.
- I prinsippet skal det ikke være av betydning om man wrapper hele App i 1 eller flere Providere.

```
JS App.js X
context-test > src > JS App.js > ...
1  import { BrowserRouter, Routes, Route, Link } from 'react-router-dom';
2  import { CitiesPage, HomePage } from './pages';
3  import { CityProvider } from './contexts/CityContext';
4
5  function App() {
6    return (
7      <div>
8
9        <BrowserRouter>
10
11          <nav> { /* TODO: Move nav to own component */}
12            <li><Link to="/">Hjem</Link></li>
13            <li><Link to="/cities">Byer</Link></li>
14          </nav>
15
16          <CityProvider>
17            <Routes>
18              <Route path="/" element={<HomePage/>}></Route>
19              <Route path="/cities" element={<CitiesPage/>}></Route>
20            </Routes>
21          </CityProvider>
22
23        </BrowserRouter>
24
25      </div>
26    );
27  }
28
29  export default App;
```

CityList

- CityList trenger tilgang til CityContext. Den må derfor både importere useContext og CityContext.
- Den henter inn det den trenger av variabler/funksjoner fra CityContext i useContext-linjen; her kun cities.
- For enkelhets skyld er det her ikke delt opp i List og Item-komponenter.

```
JS AddCity.js  JS CityList.js  X
context-test > src > components > JS CityList.js > ...
1  import { useContext } from "react";
2  import { CityContext } from "../contexts/CityContext";
3
4  const CityList = () => {
5
6      const {cities} = useContext( CityContext );
7
8      const showAllCities = () => {
9          return cities.map( ( city, i ) => (
10              <li key={`city-${i}`}>{city}</li>
11          ));
12      }
13
14      return (
15          <section>
16              <h3>Byliste</h3>
17              <ul>
18                  {showAllCities()}
19              </ul>
20          </section>
21      )
22  }
23
24  export default CityList;
```

AddCity.js

- AddCity må som CityList importere både useContext og CityContext for å få tilgang til CityContext.

```
JS AddCity.js X JS CityList.js
context-test > src > components > JS AddCity.js > ...
1  import { useRef, useContext } from 'react';
2  import { CityContext } from '../contexts/CityContext';
3
4  const AddCity = () => {
5
6      const { cities, addCity } = useContext(CityContext);
7
8      const nameElement = useRef(null);
9
10     const addNewCity = () => {
11         const name = nameElement.current.value;
12         addCity( name );
13         nameElement.current.value = "";
14     }
15
16     return (
17         <section>
18             <h3>Legg til ny by</h3>
19             <p>Antall byer: { cities.length }</p>
20             <label>Navn</label>
21             <input ref={nameElement} type="text"/>
22             <button onClick={addNewCity}>Legg til</button>
23         </section>
24     )
25 }
26
27 export default AddCity;
```

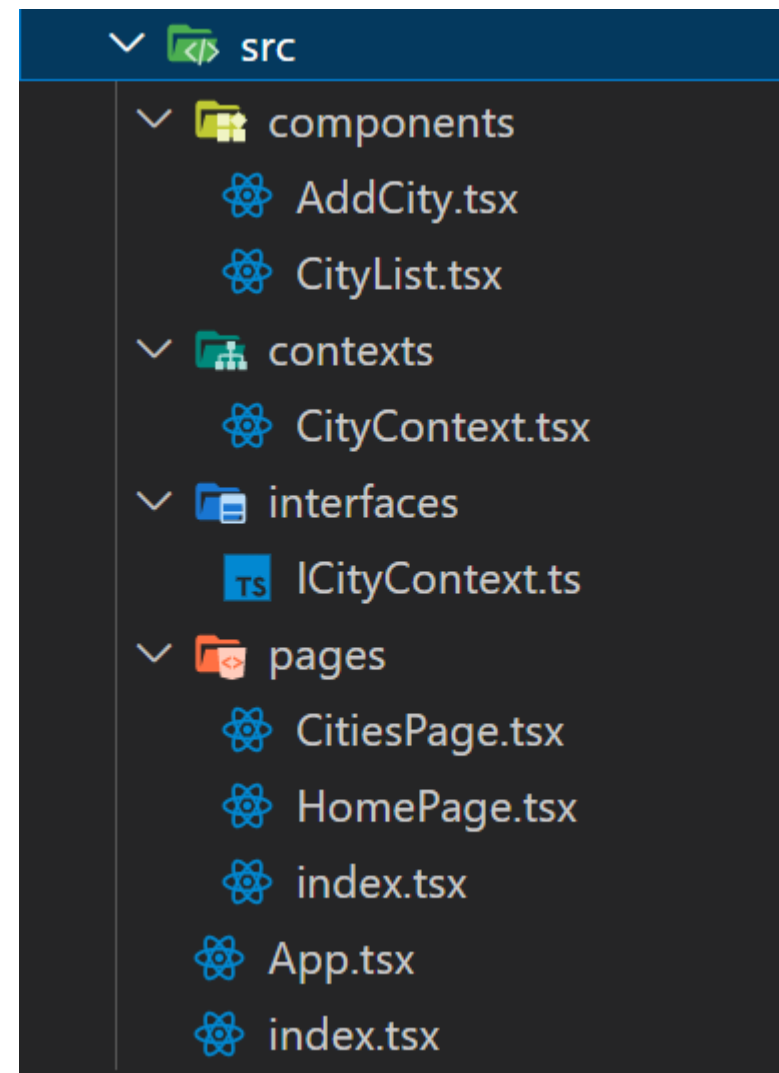
Kodeeksempel Context med TypeScript

Boilerplate og installering

- React Boilerplate:
 - `npx create-react-app context-test-typescript --template typescript`
- React Router for navigering:
 - `npm install react-router-dom`

Oppsett

- I dette eksempelet er det brukt react-router-dom også for å vise hvordan informasjonen blir behandlet hvis man hopper mellom nettsider.
- Totalt:
 - 1 interface
 - 2 components
 - 1 context
 - 2 pages med index.js for å gjøre enklere å eksportere/importere
 - 1 App.js
 - 1 index.js



Interfacet

- Interfacet angir hvilken data og funksjoner Context skal ha.
- Her er dataene et array med tekster og funksjonen har parameter for nytt bynavn og returnerer ikke noen ting

```
TS ICityContext.ts X
src > interfaces > TS ICityContext.ts > ...
1  interface ICityContext {
2      cities: string[];
3      addCity: (newCity: string) => void;
4  }
5
6  export default ICityContext;
```


Context

- createContext for å skape Context-objektet.
- ICityContext for å definere Context som av CityContext-slag
- Interface Props og FC for å kunne definere hva children er: komponenter
- useState for å ha et state som Context kan dele
- Value (linje 24) angir hva som skal være tilgjengelig fra Context

```
CityContext.tsx X
src > contexts > CityContext.tsx > ...
1  import { useState, createContext, FC, ReactNode } from "react";
2  import ICityContext from "../interfaces/ICityContext";
3
4  export const CityContext = createContext<ICityContext | null>(null);
5
6  interface Props {
7    children: ReactNode
8  }
9
10 export const CityProvider : FC<Props> = ({children}) => {
11
12     const [cities, setCities] = useState<string[]>([
13         "Oslo",
14         "Stockholm",
15         "London",
16         "Paris"
17     ]);
18
19     const addCity = (newCity: string) => {
20         setCities( [ newCity, ...cities ] );
21     }
22
23     return (
24         <CityContext.Provider value={ { cities, addCity } }>
25             {children}
26         </CityContext.Provider>
27     )
28
29 }
```

CityList

- useContext for å koble til Context
- CityContext og ICityContext for å definere hvilken Context som skal brukes
- Henter inn kun cities fra Context; trenger ikke addCity fra Context

```
CityList.tsx X
src > components > CityList.tsx > ...
1  import { useContext } from "react";
2  import { CityContext } from "../contexts/CityContext";
3  import ICityContext from "../interfaces/ICityContext";
4
5  const CityList = () => {
6
7      const {cities} = useContext( CityContext ) as ICityContext;
8
9      const showAllCities = () => {
10         return cities.map( ( city, i ) => (
11             <li key={`city-${i}`}>{city}</li>
12         ));
13     }
14
15     return (
16         <section>
17             <h3>Byliste</h3>
18             <ul>
19                 {showAllCities()}
20             </ul>
21         </section>
22     )
23 }
24
25 export default CityList;
```

AddCity

- useRef for å få tak i verdi fra tekstboks
- useContext for å sette Context-tilgang
- CityContext og ICityContext for å definere hvilken Context som skal brukes
- Linje 7 henter inn begge tingene som Context har angitt i sin value

AddCity.tsx X

src > components > AddCity.tsx > default

```
1  import { useRef, useContext } from 'react';
2  import { CityContext } from '../contexts/CityContext';
3  import ICityContext from '../interfaces/ICityContext';
4
5  const AddCity = () => {
6
7      const { cities, addCity } = useContext(CityContext) as ICityContext;
8
9      const nameElement = useRef<HTMLInputElement>(null);
10
11     const addNewCity = () => {
12         if(nameElement.current !== null){
13             const name = nameElement.current.value;
14             addCity( name );
15             nameElement.current.value = "";
16         }
17     }
18
19     return (
20         <section>
21             <h3>Legg til ny by</h3>
22             <p>Antall byer: { cities.length }</p>
23             <label>Navn</label>
24             <input ref={nameElement} type="text"/>
25             <button onClick={addNewCity}>Legg til</button>
26         </section>
27     )
28 }
29 export default AddCity;
```


HomePage

- HomePage er en vanlig komponent med .tsx-endelse

```
HomePage.tsx X
src > pages > HomePage.tsx > ...
1  const HomePage = () => {
2
3      return (
4          <section>
5              <h1>Hjem</h1>
6          </section>
7      )
8  }
9
10 export default HomePage;
```

CitiesPage

- CitiesPage er forelderkomponent til 2 komponenter hvor den ene er laget for å kunne legge til ny by, mens den andre er laget for å vise byer.

 CitiesPage.tsx ✕

src > pages >  CitiesPage.tsx > ...

```
1  import CityList from '../components/CityList';
2  import AddCity from '../components/AddCity';
3
4  const CitiesPage = () => {
5    return (
6      <section>
7        <h1>Byer</h1>
8        <AddCity/>
9        <CityList/>
10     </section>
11   )
12 }
13
14 export default CitiesPage;
```

index

- Index.tsx gjør det enklere for filen som skal importere alle Page-komponenter å importere dem

```
index.tsx ×  
src > pages > index.tsx  
1 import CitiesPage from "./CitiesPage";  
2 import HomePage from "./HomePage";  
3  
4 export { CitiesPage, HomePage };
```

App

- Importerer React Router-komponentene
- Henter inn page-komponentene
- Viktig moment: wrapper Routes i CityProvider som angir hvilke komponenter som skal ha tilgang til CityContext. Man kan ved behov ha Provider rundt hele App.

```
App.tsx X
src > App.tsx > ...
1  import { BrowserRouter, Routes, Route, Link } from 'react-router-dom';
2  import { CitiesPage, HomePage } from './pages';
3  import { CityProvider } from './contexts/CityContext';
4
5  function App() {
6
7      return (
8          <div>
9              <BrowserRouter>
10                 <nav>
11                     <ul>
12                         <li><Link to="/">Hjem</Link></li>
13                         <li><Link to="/cities">Byer</Link></li>
14                     </ul>
15                 </nav>
16                 <CityProvider>
17                     <Routes>
18                         <Route path="/" element={<HomePage/>}></Route>
19                         <Route path="/cities" element={<CitiesPage/>}></Route>
20                     </Routes>
21                 </CityProvider>
22             </BrowserRouter>
23         </div>
24     );
25 }
26
27 export default App;
```

Resultat

- Når AddCity.tsx legger til ny by oppdateres state i Context og CityList (som også er koblet til Context) oppdateres.
- På denne måten er altså 2 uavhengige komponenter koblet opp mot samme datakilde og oppdateres samtidig ved endringer i state i Context.

