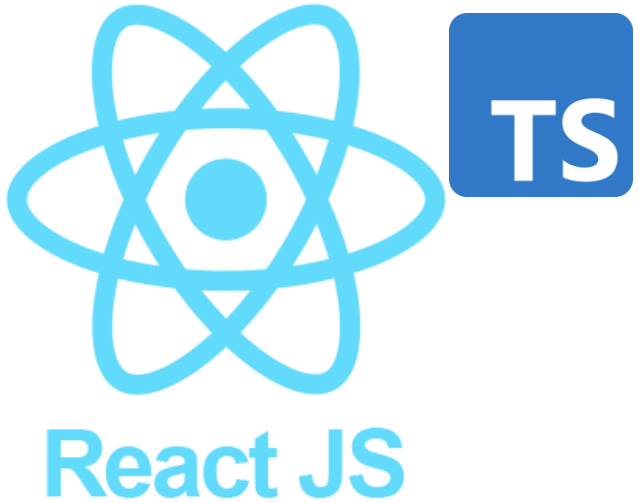
DS3103 Webutvikling
React og TypeScript

Rolando Gonzalez 2022



Innhold

- Hva er TypeScript?
- Hvorfor ønsker man å bruke TypeScript?
- Opprette en React boilerplate med TypeScript
- .ts vs. .tsx

- Eksempelkoder
  - Variabel
  - Funksjon
  - Komponenter med props

Hva er TypeScript?

- TypeScript er et superset av JavaScript.
- Superset: «A programming language that contains all the features of a given language and has been expanded or enhanced to include other features as well.» (encyclopedia2.thefreedictionary.com)
- TypeScript er JavaScript + muligheter for å definere datatyper og returtyper, interface, types m.m.

- TypeScript er et «under development»-språk. Det vil si blir ikke med i sluttresultatet, men kun mens utviklerne utvikler løsningen.

Hvorfor ønsker man å bruke TypeScript?

- Å bruke TypeScript gir utviklerne mer kontroll over løsningen.
- Man både ser mye enklere hva ting er (deres datatyper m.m.), mer intellisense, og får mer tilbakemelding fra boilerplate om man gjør ting feil.

Opprette en React boilerplate med TypeScript

- For å skape en React boilerplate med TypeScript kjører man følgende kommando:
  - npx create-react-app prosjekt-navn --template typescript

.ts vs. .tsx

- I React TypeScript-prosjektet vil det ligge .ts-filer og .tsx-filer.
- .ts-filer er filer for JavaScript med TypeScript.
- .tsx-filer er filer for React TypeScript-komponenter; eller mer konkret: der hvor man gjør bruk av JSX-kode.

Kodeeksempler

Variabler

- Med tanke på variabler kan man angi datatypen umiddelbart.
- Merk at man vil kunne ha litt splittede meninger om det er nødvendig eller ikke å angi datatypen når man setter variabelen med en gang.

```
let numberOfTypesOfCakes: number = 33;
let nameOfFirm: string = "Samson";
const internationalOffice: boolean = false;
const officesInNorway: string[] = ["Oslo", "Bergen", "Ålesund"];
```

En funksjon med TypeScript

- En funksjon kan i TypeScript definere datatypen for parameterne og retur-datatype.

```
const multiplyNumbers = ( number1: number, number2: number ) : number => {
    return number1 * number2;
}
```

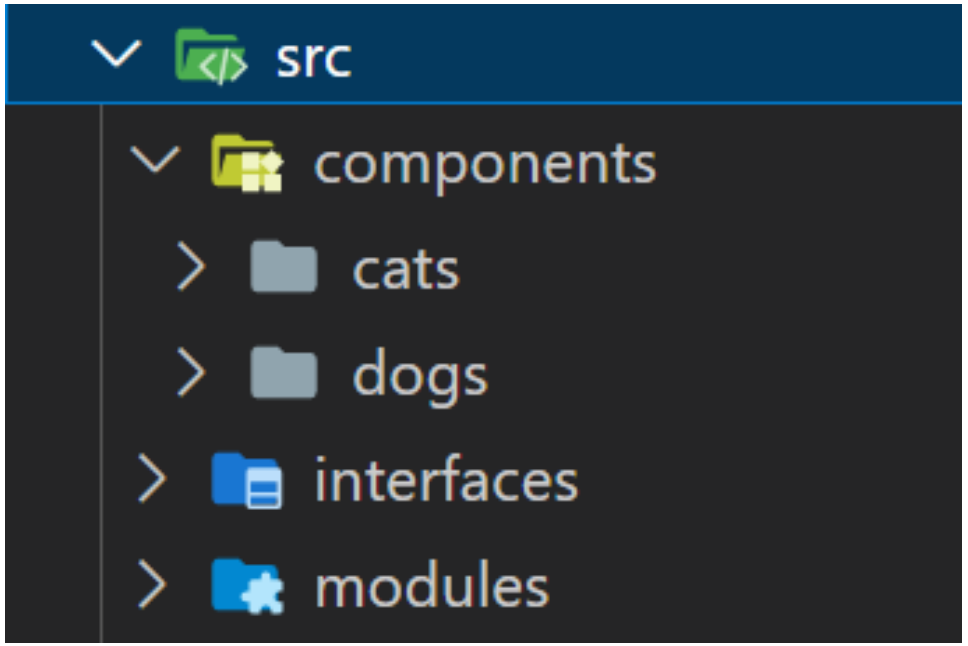
Komponenter med props

Oversikt over «Komponent med props»

- I de følgende slidene vil vi se en kombinasjon av flere filer som alle tar i bruk et «interface».
- De følgende filene vil være involvert:
  - Interface
  - Module
  - List-komponent
  - Item-komponent

Komponenter med props

- Når man lager komponenter med props vil man umiddelbart få feilmelding hvis man ikke bruker TypeScript for å definere datatypen til propsene.
- Det finnes flere måter å definere propsene til en komponent på. Én måte å definere dem på er å opprette interface.
- Eksempelvis har man en løsning for katter og hunder. Da kan man begynne med å lage interfaces i egen mappe for å definere hvordan informasjonen til en katt og en hund skal være



Interface

- “One of TypeScript’s core principles is that type checking focuses on the *shape* that values have. This is sometimes called “duck typing” or “structural subtyping”. In TypeScript, interfaces fill the role of naming these types, and are a powerful way of defining contracts within your code as well as contracts with code outside of your project.”

- TypeScript-håndboken:
  - https://www.typescriptlang.org/docs/handbook/interfaces.html

interface ICat

- Navnet til et interface dekoreres gjerne med stor i som prefix.
- ICat er interfacet som beskriver hvilken informasjon en katt skal inneholde.
- I koden ser vi at vi sier at en katt skal presenteres med navn, alder og katterase.
- Vi merker oss også det at breed har et spørsmålstegn etter seg. Det betyr at den er «optional» - må ikke legges til.

ICat.ts

src > interfaces > ICat.ts > ...

```
1   interface ICat {
2       name: string,
3       age: number,
4       breed?: string
5   }
6
7   export default ICat;
```

Module tar i bruk ICat

- CatModule – her en «database» med tilgangsfunksjoner til «databasen» tar også i bruk ICat for å definere hva slags informasjon den jobber med.
- Både array med informasjon og funksjoner dekoreres med ICat[]
- ICat[] betyr «array som inneholder objekter av typen Cat»

CatModule.ts

src > modules > CatModule.ts > ...

```ts
import ICat from "../interfaces/ICat";

const CatModule = (
    () => {

        const catArray: ICat[] = [
            {
                name: "Mr. Fluffy",
                age: 2,
                breed: "Norsk skogskatt"
            },
            {
                name: "Miawosy Lu",
                age: 5,
                breed: "Ragdoll"
            }
        ];

        const getAll = () : ICat[] => catArray;

        return {
            getAll
        }

    }
)();

export default CatModule;
```

CatItem tar i bruk ICat

- CatItem importer FC (Functional Component) fra react og ICat interfacet.
- På den måten som er vist på skjermbildet knyttes informasjonen i props til ICat.

ICat.ts | CatItem.tsx

src > components > cats > CatItem.tsx > ...

```tsx
import { FC } from 'react';
import ICat from '../../interfaces/ICat';

const CatItem: FC<ICat> = ({name, age, breed}) => {
    return (
        <article>
            <h3>{name}</h3>
            <p>Alder: {age}</p>
            <p>Rase: {breed}</p>
        </article>
    )
}

export default CatItem;
```

CatList tar i bruk ICat

- CatList vil være en komponent som er avhengig av flere ting. Den må ha tilgang til informasjonssenteret (Module), CatItem som den skal generere, ICat for å definere informasjonen og useState.

- Detalj: merk «cats?.» i getCatItems. Dette har med at koden oppdager at cats kan være null (dvs. ikke satt). Alternativet er at vi putter tom array i useState.

CatList.tsx

src > components > cats > CatList.tsx > ...

```tsx
import { useEffect, useState } from "react";
import CatItem from "./CatItem";
import ICat from "../../interfaces/ICat";
import CatModule from "../../modules/CatModule";

const CatList = () => {

    const [ cats, setCats ] = useState<ICat[]>();

    useEffect( () => {
        setCats( CatModule.getAll() );
    }, []);

    const getCatItems = () => {
        return cats?.map( (cat, i) => (
            <CatItem
                key={`cat-${i}`}
                name={cat.name}
                age={cat.age}
                breed={cat.breed}
                />
        ) );
    }

    return (
        <section>
            <h1>Katter</h1>
            <section>{getCatItems()}</section>
        </section>
    )
}

export default CatList;
```

Offisielle nettsider for TypeScript

- Diverse inforrmasjon om TypeScript:
  - https://www.typescriptlang.org/