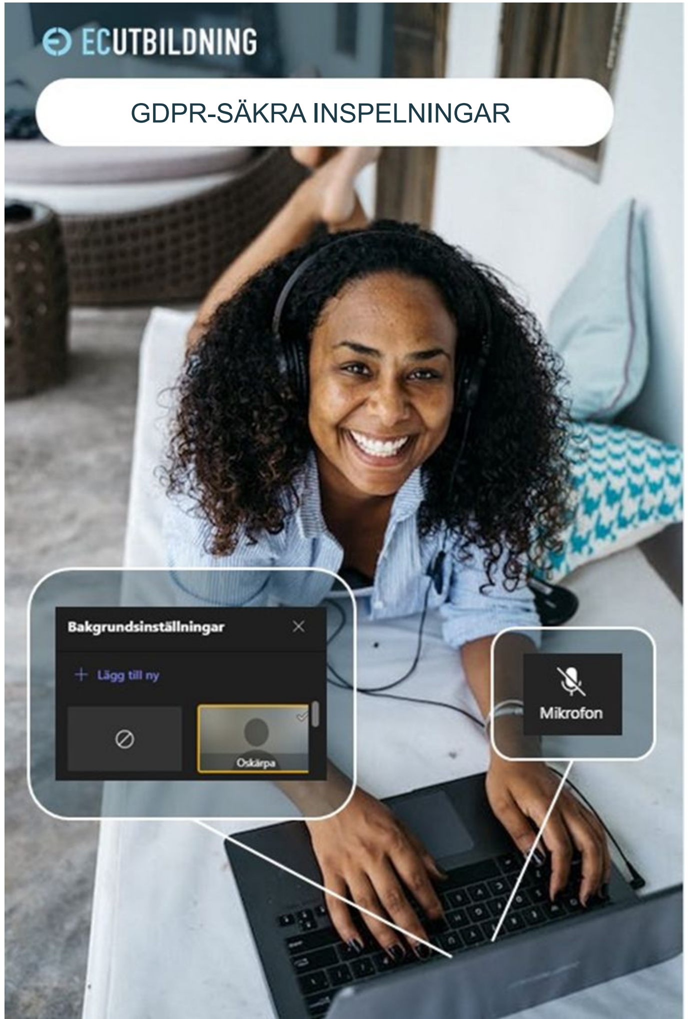
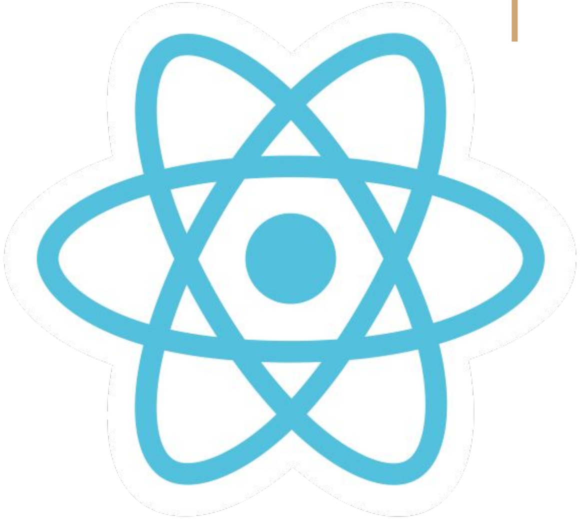
Roligt att just du är här idag!
För allas trevnad och för att jobba med GDPR-säkra inspelningar har EC Utbildning följande rutin kring inspelning av föreläsningar/lektioner.

- Blurra bakgrunden (använd oskärpa) genom att använda funktionen för bakgrundsfilter i Teams om du har kameran på, eller ha kameran avstängd.
- Mikrofonen skall vara avstängd när den inte behövs
- Inga privata samtal eller chatt medan inspelning pågår
- Sitt i en tyst miljö för att undvika bakgrundsljud
- Stäng av din kamera och mikrofon om du lämnar föreläsningen/lektionen tillfälligt och vid paus/rast
- Inspelade föreläsningar/lektioner får inte spridas utanför skolan.

Ha en lärorik dag!

ECUTBILDNING x gaddr





Introduction to **React.js**

[09/2022]

LECTURE 6

An interactive web application in React using TypeScript

ECUTBILDNING x gaddr

Rest of the course

- ❑ 14 / 15 sep: An interactive web application using TypeScript (Styling, Props, State); the different packages useful during development
- ❑ 20 sep: overview and questions + time for finishing your project and presentation
- ❑ 21 / 22 sep: ***presentations!*** Register your team: **https://docs.google.com/spreadsheets/u/2/d/1uTF1LXOrtJQXHiTx9_L7IqCcmRZmQXItXtRJ7to5P7k/edit#gid=653812220**

Dagens lektion

- Intro to TypeScript
- Creating a React application using TypeScript
- Information about packages which are helpful during development

Packages which will be mentioned today

- prettier
- linters (eslint, tslint, jslint)

TypeScript

JavaScript with types specified

- TypeScript offers all of JavaScript's features, and an additional layer on top of these: the type system.
- For example, JavaScript provides language primitives like `string` and `number`, but it doesn't check that you've consistently assigned these. TypeScript does.
- The main benefit of TypeScript is that ***it can highlight unexpected behavior in your code, lowering the chance of bugs***.

Important features of TypeScript

- mentioning types for (almost) every variable/constant
- interfaces
- optional parameters using “ ? ”
- for functions, mentioning types for input parameters and mentioning function return types

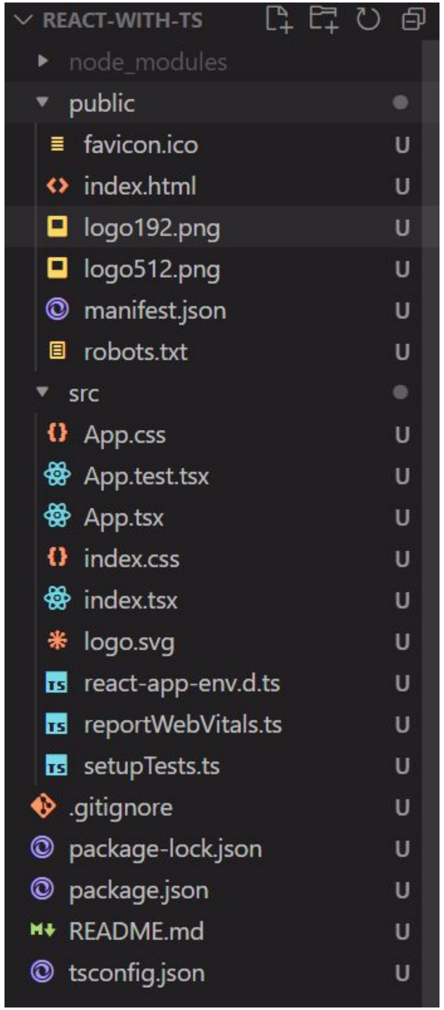
TypeScript with React

```
npx create-react-app react-with-ts --template typescript
```

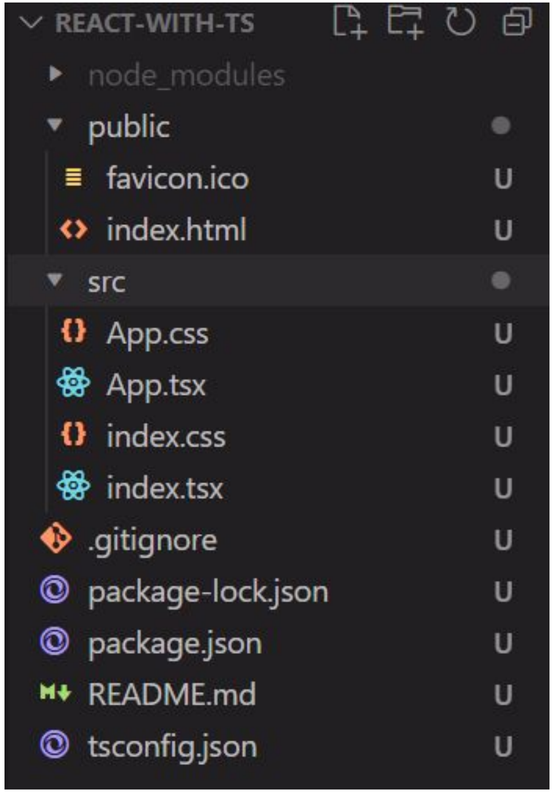
Instead of `.jsx`, it is `.tsx`

- to keep it simple, we are going to remove the extra items
- Removing:
  - `App.test.tsx` in src/
  - `logo.svg` in src/ and its references inside `App.tsx` (line 2, 9)
  - `reportWebVitals.ts` in src/ and its references inside `index.tsx` (line 5,16,17,18,19)
  - `react-app-env.d.ts, setupTests.ts` in src/
  - `logo192.png, logo512.png, manifest.json, robots.txt` in public/

We are now left with a very basic react app and can start editing it


```
REACT-WITH-TS
  node_modules
  public
    favicon.ico          U
    index.html           U
    logo192.png          U
    logo512.png          U
    manifest.json        U
    robots.txt           U
  src
    App.css              U
    App.test.tsx         U
    App.tsx              U
    index.css            U
    index.tsx            U
    logo.svg             U
    react-app-env.d.ts   U
    reportWebVitals.ts   U
    setupTests.ts        U
  .gitignore             U
  package-lock.json      U
  package.json           U
  README.md              U
  tsconfig.json          U
```

React project structure/architecture



```
REACT-WITH-TS
  node_modules
  public
    favicon.ico        U
    index.html         U
  src
    App.css            U
    App.tsx            U
    index.css          U
    index.tsx          U
  .gitignore           U
  package-lock.json    U
  package.json         U
  README.md            U
  tsconfig.json        U
```

- Main components:
  - src folder
  - node_modules folder
  - package. json
  - index.tsx
  - index.html
- Other major components:
  - README.md
  - index.css (and other css files)
  - package-lock.json
- Commonly used:
  - public folder to store static assets (images, etc)

ECUTBILDNING x gaddr

Linting

linters you may already have installed:
jslint, tslint, eslint

- Linting is the process of checking the source code for Programmatic as well as Stylistic errors.
- This is most helpful in identifying some common and uncommon mistakes that are made during coding.
- The *squiggly underlines* you see in your code are due to linting
- Basically they ensure no errors in the code during development

prettier

[**https://prettier.io/docs/en/install.html**]

```
npm install --save-dev --save-exact prettier
```

- create a .prettierrc.json file at the base of your project
- this is your configuration file
  [**https://prettier.io/docs/en/configuration.html**]
- more options: **https://prettier.io/docs/en/options.html**

Configuration files

.tsconfig, .prettierrc

A configuration file, often shortened to config file, defines the parameters, options, settings and preferences applied to your program

TypeScript “interfaces”

- basically like a “class”
- defines what an “object” should look like
- contains “required” and “optional” parts of the object
- can be nested for nested objects

```
interface IFunction {
   name: string;
   age: number | Date;
   password?: string | number | (() => void) | ((param: number) => string);
}

const myFunction = ({ name, age }: IFunction) => {
   // function logic
};
```

ECUTBILDNING x gaddr

Defining types for “useState”

```
const [exampleVariable, setExampleVariable] =
   useState<number | string>(0)
setExampleVariable("xyz")
```

Keywords for today

1. Linting
2. prettier
3. configuration files
4. interfaces
5. types
6. TypeScript

Today’s task

You may use this time to work on your projekt