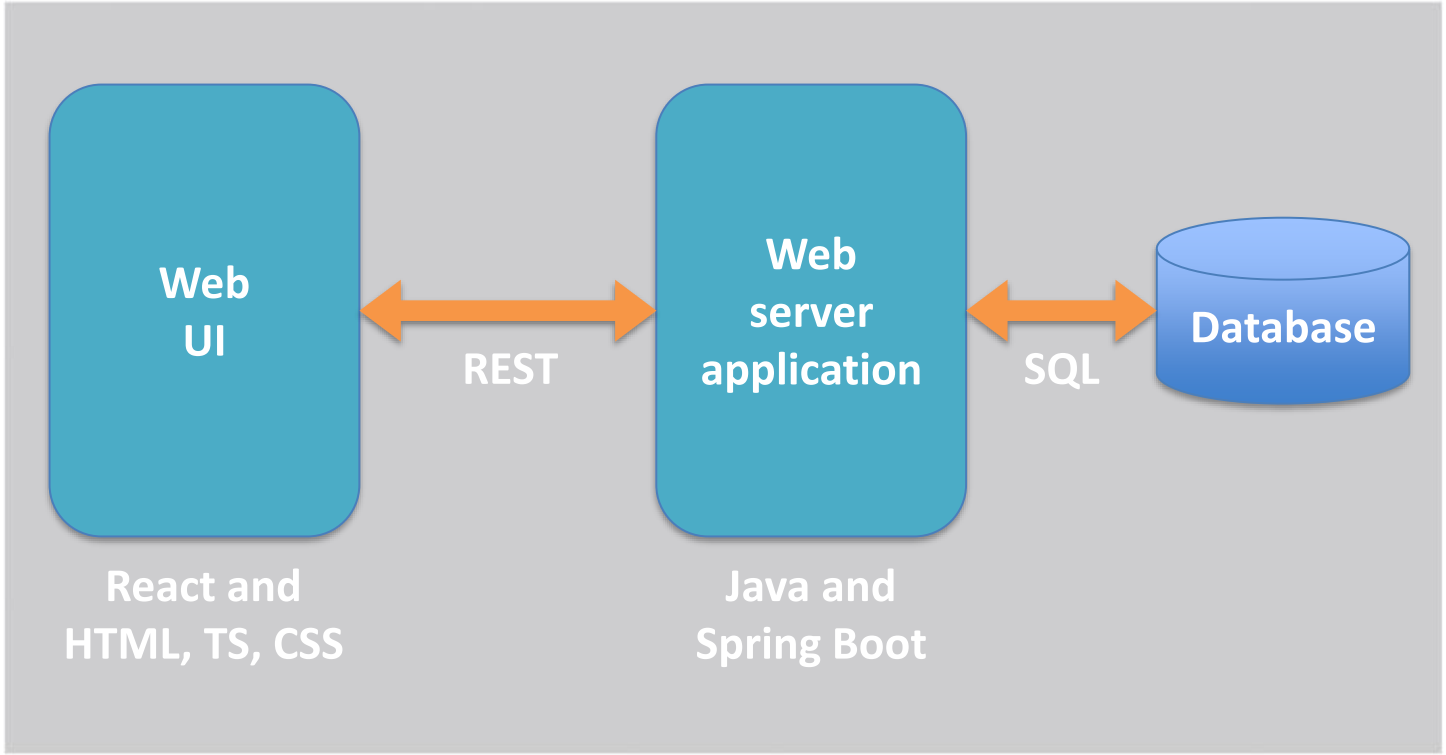
Creating a React Front End

1. Creating a simple React app
2. Working with components

1. Creating a Simple React App

- Recap our technology stack
- How to create a React app
- Reviewing the React app
- The home page
- The source code entry point
- The `App` component
- Running the application

Recap our Technology Stack



How to Create a React App

- There are various tools available to create a React app
  - We're going to use **Create React App**
  - See https://create-react-app.dev/

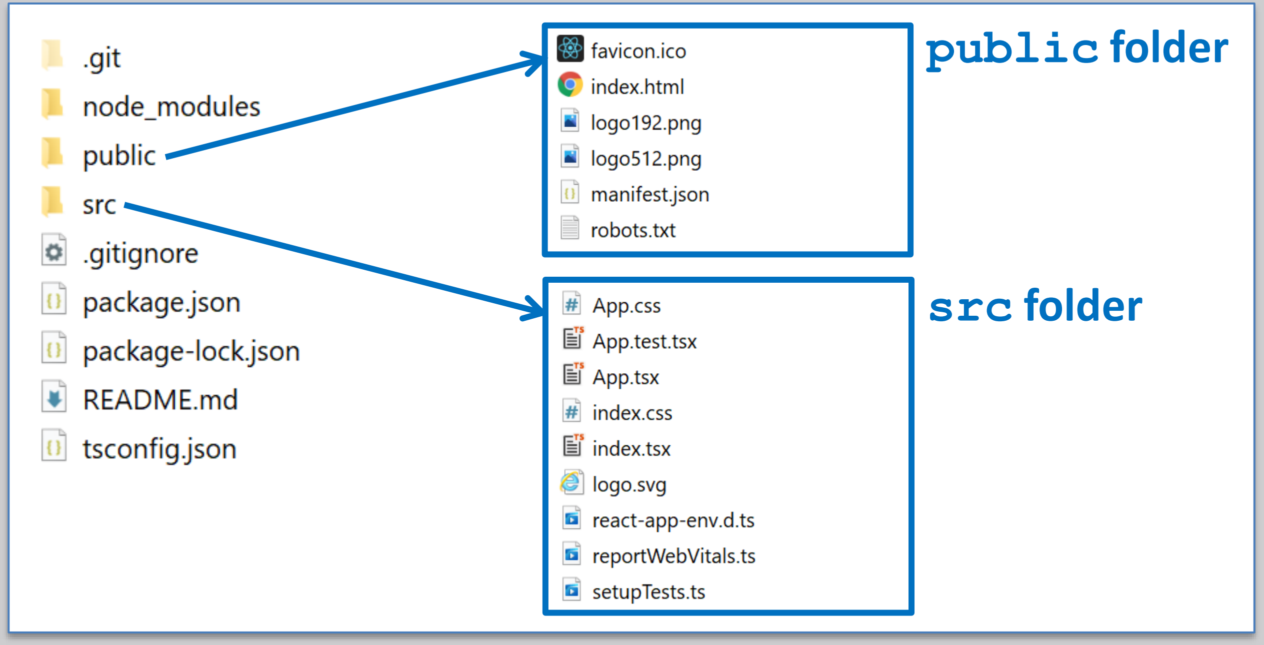
- You can create a React app as follows:

```
npx create-react-app my-demo-app --template typescript
```

  - Creates a simple React app in TypeScript
  - Downloads React libraries to the `node_modules` folder

Reviewing the React App

- Here's the structure of the generated React app:



.git
node_modules
public
src
.gitignore
package.json
package-lock.json
README.md
tsconfig.json

favicon.ico
index.html
logo192.png
logo512.png
manifest.json
robots.txt

App.css
App.test.tsx
App.tsx
index.css
index.tsx
logo.svg
react-app-env.d.ts
reportWebVitals.ts
setupTests.ts

The Home Page

- The application home page is `public/index.html`

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>React App</title>
    …
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

This is where your React content will be added

public/index.html

The Source Code Entry Point

- The source code entry point is `src/index.tsx`

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
…
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

This creates an `App` component (see next slide)

src/index.tsx

- Aside: For info about React "strict mode", see:
  - https://reactjs.org/docs/strict-mode.html

The App Component

- App is a "functional component"
  - i.e. a function that returns HTML (XML actually ☺)

```
import React from 'react';
import './App.css';
…

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        … Plus other HTML content …
      </header>
    </div>
  );
}

export default App;
```

A .tsx file is a mixture of TypeScript and XML

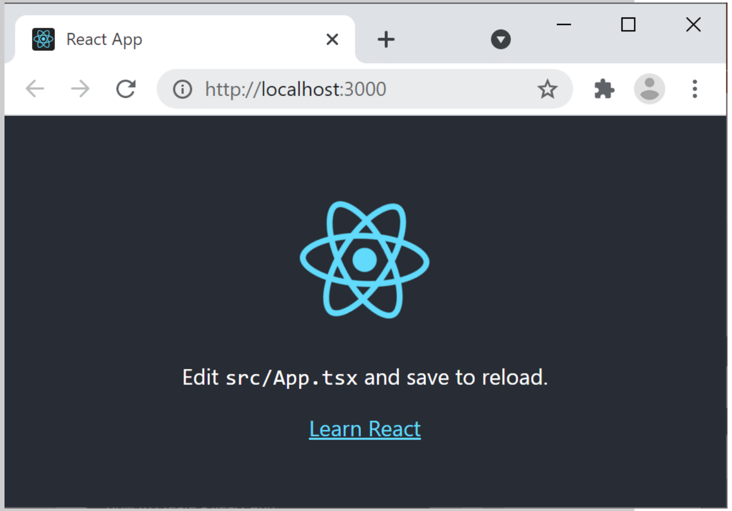
src/App.tsx

Running the Application

- You can run the application as follows:

```
npm start
```

  - Builds the app in memory
  - Starts a server to host app on http://localhost:3000



Edit `src/App.tsx` and save to reload.

Learn React

2. Working with Components

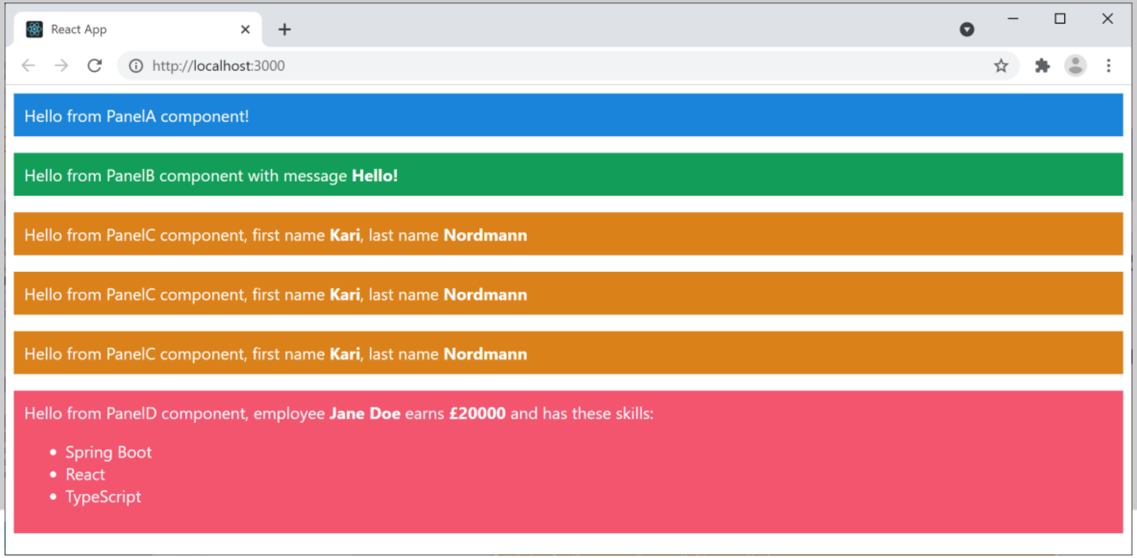
- Overview
- Defining and instantiating a component
- Passing a property to a component
- Passing multiple properties
- Working with complex data

```
React demo:  demo-react-app2
To install:  npm install
To run:      npm start
```

Pearson

Overview

- Let's see a React app that has multiple components
  - The root component, named `<App>`
  - Plus other components, to render portions of HTML



Hello from PanelA component!

Hello from PanelB component with message **Hello!**

Hello from PanelC component, first name **Kari**, last name **Nordmann**

Hello from PanelC component, first name **Kari**, last name **Nordmann**

Hello from PanelC component, first name **Kari**, last name **Nordmann**

Hello from PanelD component, employee **Jane Doe** earns **£20000** and has these skills:

- Spring Boot
- React
- TypeScript

Defining and Instantiating a Component

- Define a component as a function that returns markup:

```
export default function PanelA() {
  return (
    <div className="panelA">
      Hello from PanelA component!
    </div>
  );
}
```

PanelA.tsx

- Instantiate the component wherever you need it:

```
export default function App() {
  return (
    <React.Fragment>
      <PanelA />
      … … … …
    </React.Fragment>
  );
}
```

App.tsx

Passing a Property to a Component

- A component can receive a "properties" object:

```
export default function PanelB(props: any) {
  const msg = props.msg
  return (
    <div className="panelB">
      Hello from PanelB component with message <b>{msg}</b>
    </div>
  );
}
```

Note: You must enclose TS code in { }

PanelB.tsx

- You can pass in a property value when you create the component, as follows:

```
<PanelB msg="Hello!" />
```

App.tsx

Passing Multiple Properties (1 of 2)

- A component can receive multiple properties:

```
export default function PanelC(props: any) {
  const fname = props.fname
  const lname = props.lname
  return (
    <div className="panelC">
      Hello from PanelC component, first name <b>{fname}</b>, last name <b>{lname}</b>
    </div>
  );
}
```

PanelC.tsx

Passing Multiple Properties (2 of 2)

- You can pass in properties one-by-one as follows:

```
<PanelC fname="Kari" lname="Nordmann" />                                           App.tsx
```

- You can also use objects, if appropriate:

```
{/* Create an object containing some useful properties */}
const person = {fname: 'Kari', lname: 'Nordmann'}

{/* Either pass in the object's properties one-by-one */}
<PanelC fname={person.fname} lname={person.lname} />

{/* Or use the EcmaScript spread operator to achieve the same effect */}
<PanelC {...person} />                                                             App.tsx
```

Working with Complex Data

```
export default function PanelD({name, salary, skills}: any) {
  return (
    <div className="panelD">
      Hello from PanelD component,
      employee <b>{name}</b> earns <b>&pound;{salary}</b> and has these skills:
      <ul>
        {skills.map((skill: string, i: number) =>
          <li key={i}>{skill}</li>
        )}
      </ul>
    </div>
  );
}
```

PanelD.tsx

```
const employee = {
  name: 'Jane Doe',
  salary: 20_000,
  skills: ['Spring Boot', 'React', 'TypeScript']
}
<PanelD {...employee} />
```

App.tsx

Summary

- Creating a simple React app
- Working with components