Developing Components with React + Typescript

Components are independent and reusable bits of code and are like JavaScript functions. Components are of two types: Class components and Function components. Always start component names with a Capital Letter as React treats components starting with lowercase letters as DOM tags.

Function components: The code below creates a component called Welcome with a function argument called props (Note: A function may or may not have arguments). A component has a return statement which indicates what should appear on the screen when a component is called.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

To use this component in your application, use a HTML syntax **<Welcome />**
We display the Welcome component in the "root" element:

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const root = ReactDOM.createRoot(document.getElementById('root'));
const element = <Welcome name="Sara" />;
root.render(element);
```

We call root.render() with the <Welcome name="Sara" /> element. Always pass the function arguments within the Welcome component call. React calls the Welcome component with {name: 'Sara'} as the props. Our Welcome component returns a <h1>Hello, Sara</h1> element as the result.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}
```

Components can refer to other components in their output. We created an App component that renders Welcome many times. As you can see, we have re-used the Welcome component with different names each time. This will be useful in our Uni Portfolio app as for example when displaying connections for a user we can create one component for a connection and re-use the code n-times for the number of connections by providing a different name, profile picture (etc) as the argument. A component can also be used in the Recommendations feature. By using components, it saves time for developers as they don't have to write various codes for the same feature. Furthermore, if any changes are to be made within a code it can be made once within the component rather than editing several lines of code.

React is all about re-using code and it is recommended to split your components into separate files. Therefore, in the src folder, add a folder called components. This will hold any react components and within this folder create a new file with a .tsx file extension and place your component code inside it and make sure to add this statement after creating a component
export Welcome;
Thereafter to be able to use your component you must import the file in your application.
For example for the Welcome component , if we created a Welcome.tsx then the import statement will be
import Welcome from './Welcome.tsx' (File name should be starting with a capital letter)

Resources :

https://reactjs.org/docs/components-and-props.html

https://www.w3schools.com/react/react_components.asp#:~:text=Components%20are%20independent%20and%20reusable,will%20concentrate%20on%20Function%20components.

https://www.w3schools.io/learn/react-typescript-functional-components/?utm_content=cmp-true

https://beta.reactjs.org/learn/your-first-component