AW Academy

# React (Part 1)

Learning Outcomes

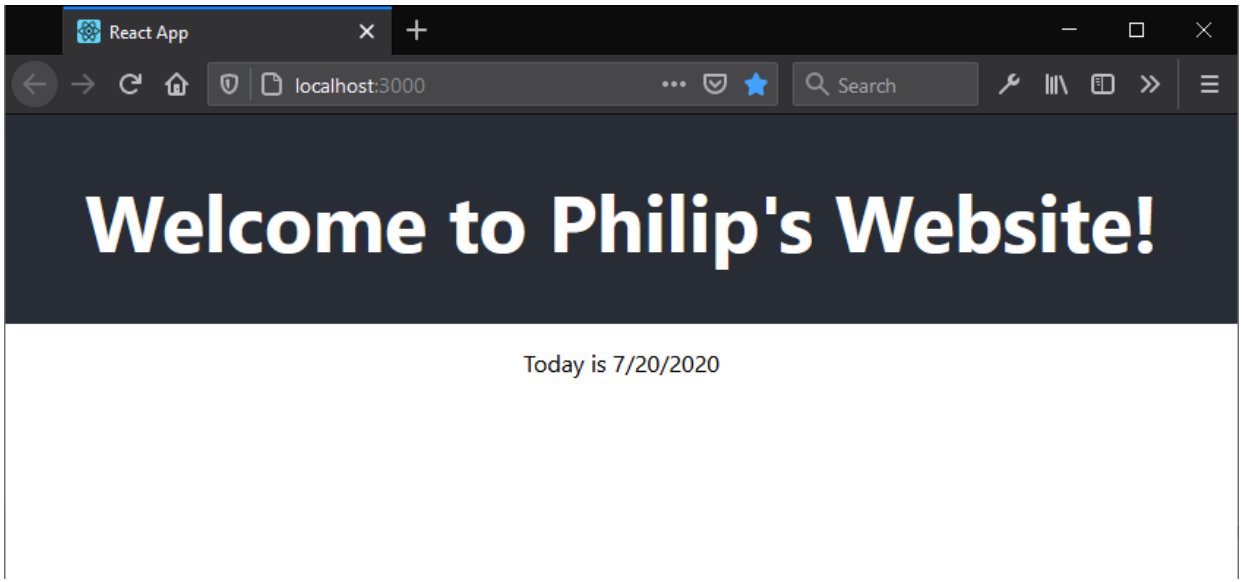
- Get to know React and JSX
- Understand functional components
- Understand Props

# React

JavaScript library for building user interfaces

# Building user interfaces with JavaScript

```
const name = "Philip's Website";
const today = new Date().toLocaleDateString();
```



# Getting started

Open the command line (PowerShell, Bash or Terminal) and change to a directory where you want your React app folder to be created. Then run this command:

```
npx create-react-app my-very-first-react-app
```

A folder "my-very-first-react-app" will be created. Open this folder in VSCode and look around the files.

Change into the directory (with your command line or a new Terminal in VSCode) and run

```
npm start
```

# Things to look at

- public/index.html
  - The div "root"
- src/index.js
- src/App.js
  - **`function App()`** is a *Component*

# HTML in JavaScript

- In React, we use code that looks like HTML inside our JavaScript to describe the desired user interface
- That is called JSX, more on it later

# Modifying the App Component

- Change the content of src/App.js:
- Notice how the Browser reloads, as soon as you save
- Change src/App.css to make the header element smaller and remove unused classes/animations

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

function App() {

  const name = "Philip's Website";
  const today = new Date().toLocaleDateString();

  return (
    <div className="App">
      <header className="App-header">
        <h1>Welcome to {name}!</h1>
      </header>
      <p>Today is {today}</p>
    </div>
  );
}

export default App;
```

# Components

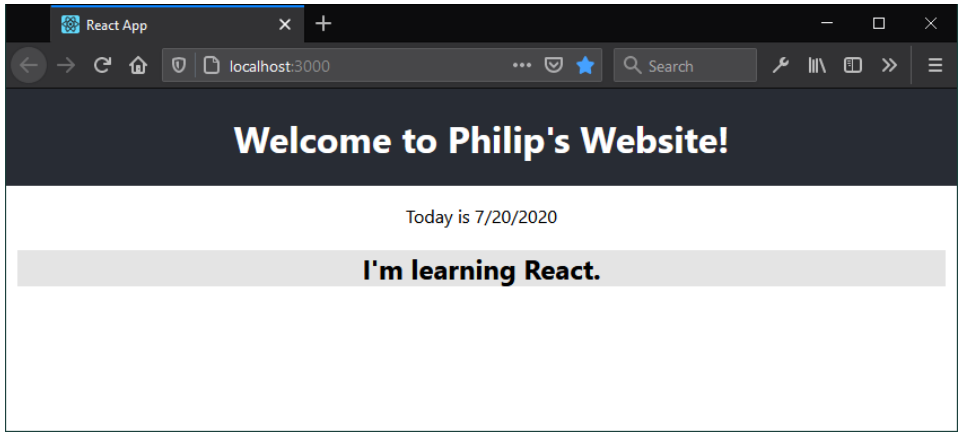
- Components are functions which return React elements describing what should appear on screen.
  - Function name is usually capitalized
  - "React elements": Written like HTML, thanks to JSX
  - Curly brackets used to place JS expressions inside HTML
- Our React app has only one component: **`App`**, which is rendered in src/index.js

# Create your first component

- Add this component function to App.js:

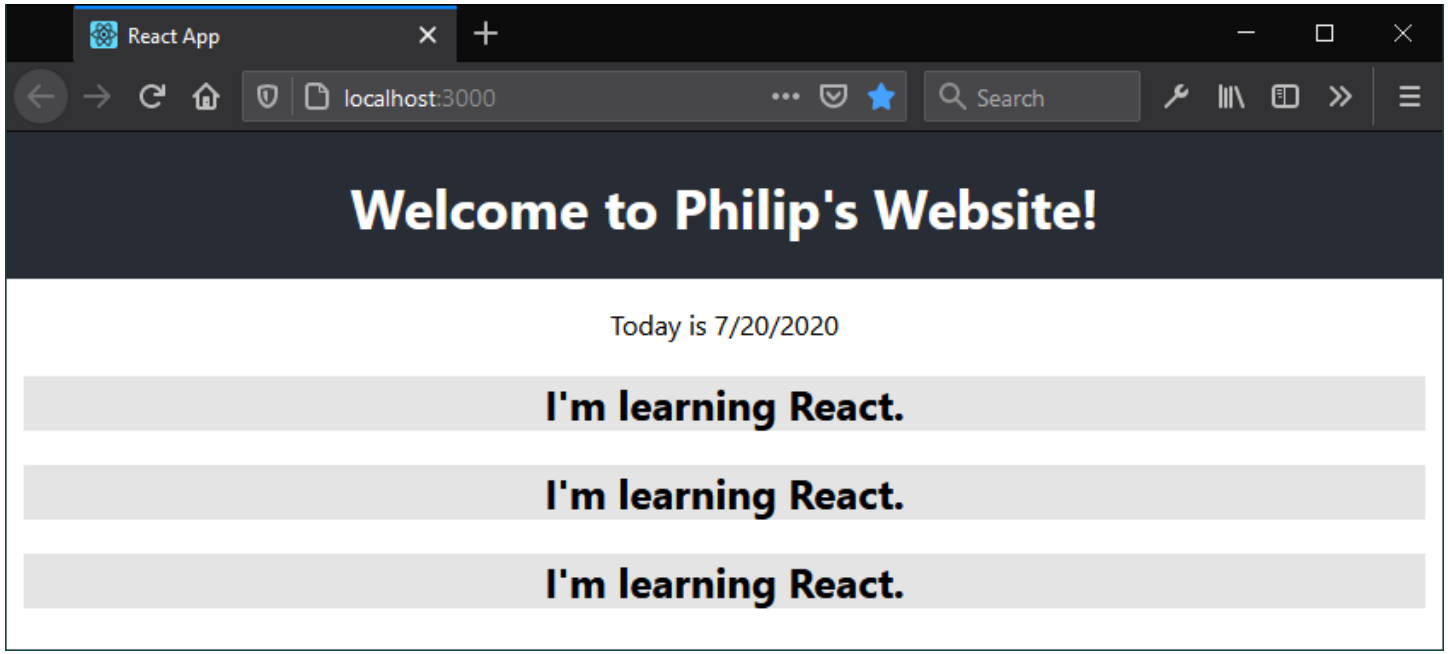
```
function Article() {
  return (
    <div className="Article">
      <h2>I'm learning React.</h2>
    </div>
  );
}
```

- Use the JSX expression **`<Article/>`** in your app component for this result:
  (Add some CSS to App.css)



# Re-Using Components

- Use the JSX expression **`<Article/>`** in your app component multiple times for this result:



# Using components

- Use components by their function name as "HTML Tag" in JSX
- Components can be re-used many times, just like regular functions
- Self-Closing tags with a slash before closing angle bracket

# Adding a prop to the component

- Add an argument **`props`** to the Article function
- Replace the fixed title *I'm learning React* with the expression **`props.title`**
- Define the prop **`title`** when using the component, somewhat like an HTML attribute

```
function Article(props) {
  return (
    <div className="Article">
      <h2>{props.title}</h2>
    </div>
  );
}
```

in *function App()*

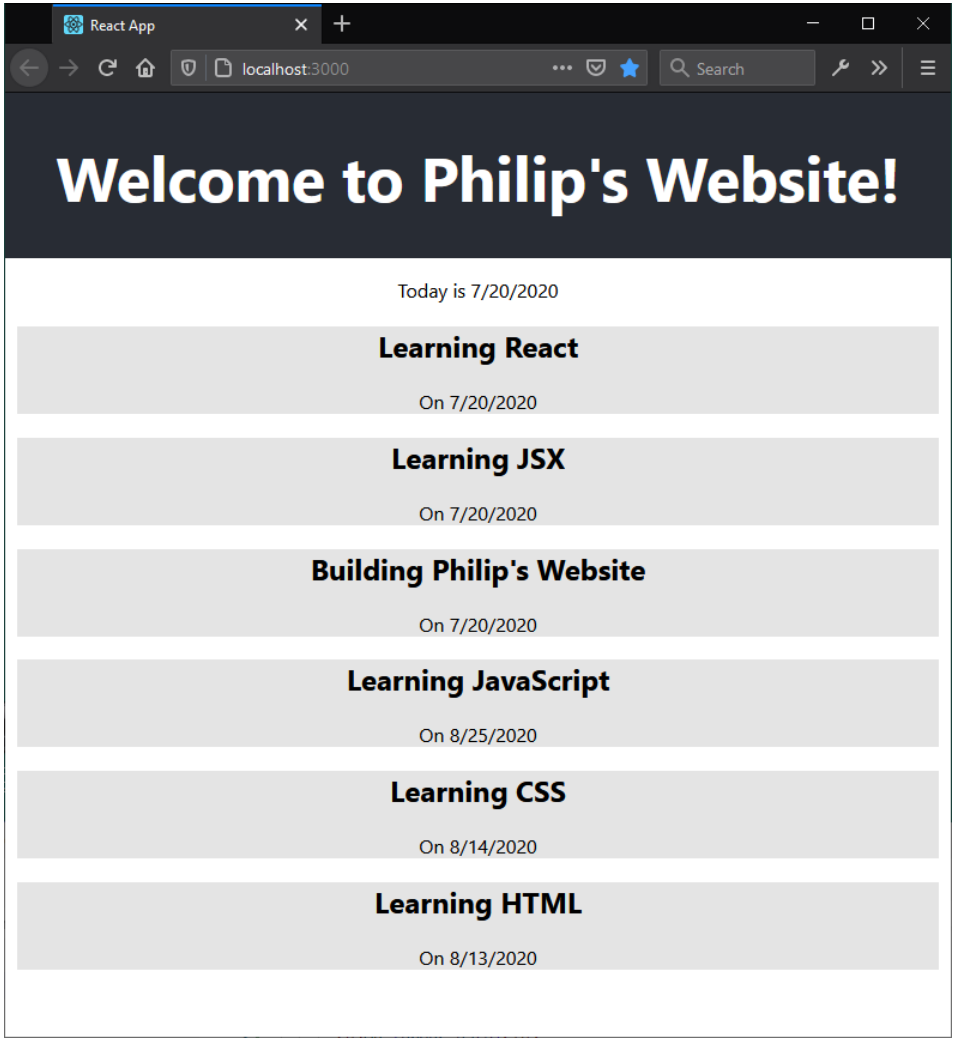
```
      <Article title="Learning React"/>
      <Article title="Learning JSX"/>
      <Article title={"Building " + name}/>
```

# Props

- Components accept a single object argument commonly named "props" (short for properties)
- Entries for the props object can be specified like HTML attributes when using the component
  - For fixed string values, use double quotes: `title="Learning React"`
  - For dynamic expressions, use curly brackets: `title={"Building "+name}`
- Props are used to pass values **into** the component.

# Change `Article` for this result

```
<Article date={today} title="Learning React"/>
<Article date={today} title="Learning JSX"/>
<Article date={today} title={"Building " + name}/>
<Article date="8/25/2020" title="Learning JavaScript"/>
<Article date="8/14/2020" title="Learning CSS"/>
<Article date="8/13/2020" title="Learning HTML"/>
```



# Children

- A special props entry **`children`** contains the component element's children

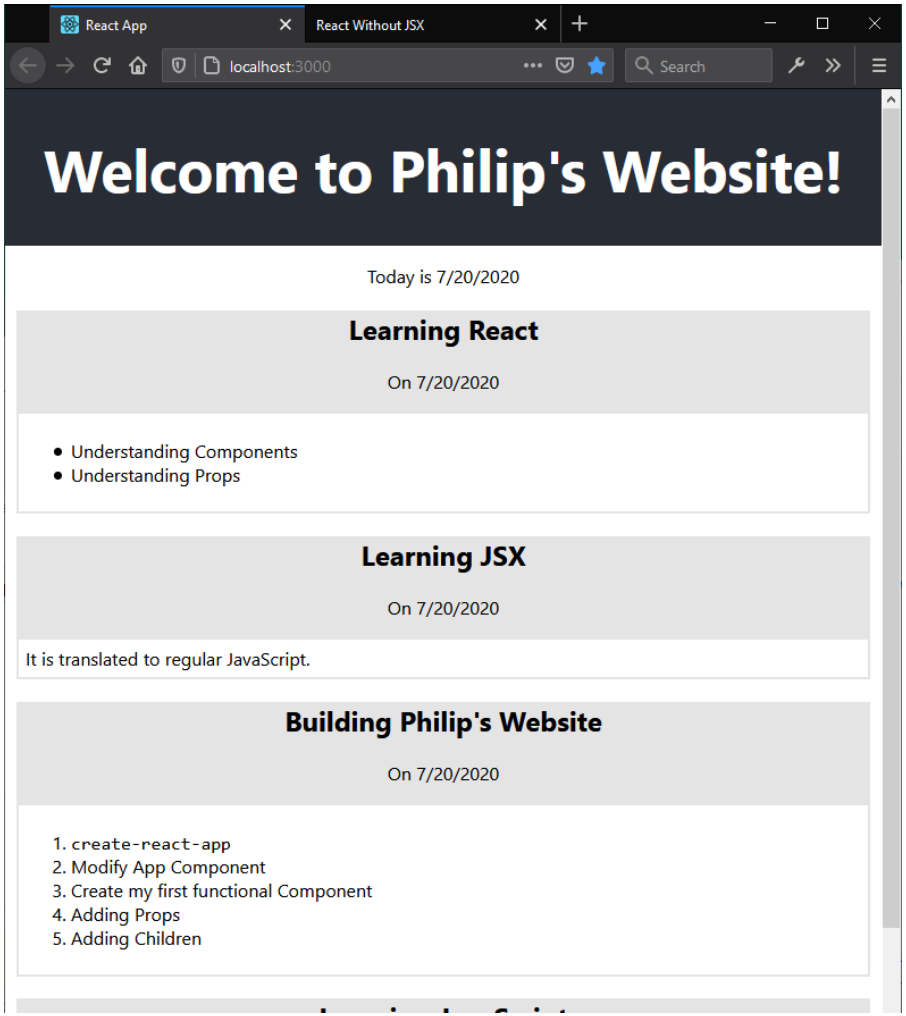
```
<MyComponent>
  Children go here, could also be <strong>HTML Elements</strong>
  or other components.
</MyComponent>
```

- Note: Element is no longer self-closing

# Change **App** component for this result

- Add children elements to Articles
- Add some CSS to make it look nice

```
function Article(props) {
  return (
    <div className="Article">
      <h2>{props.title}</h2>
      <p>On {props.date}</p>
      <div className="ArticleChildren">
        {props.children}
      </div>
    </div>
  );
}
```



# JSX in depth

https://reactjs.org/docs/jsx-in-depth.html

Look at **`react-without-jsx.html`** for an example of React without **`create-react-app`** and any JavaScript preprocessing

```
const myelement = <h1>I Love JSX!</h1>;

ReactDOM.render(myelement, document.getElementById('root'));
```

```
const myelement = React.createElement('h1', {}, 'I do not use JSX!‘);

ReactDOM.render(myelement, document.getElementById('root'));
```

# Things to try out

- Nested components: Article as child of Article
- Props with types other than string: number, Date, ...
- Assigning React Elements to variables
- Simple conditionals
  - Show placeholder content when Article has no children
- String values are escaped by React

Learning Outcomes Review

- Get to know React and JSX
- Understand functional components
- Understand Props