

# JavaScript Frameworks

Day 2

# Agenda

- Assignment 1 Walkthrough
- Components

# React Components

# What is a Component?

- Think of a component as the building blocks for your application
- Some components can be made up of simple static HTML elements such as a header or footer element
- Some components will be more dynamic with their template changing with the applications state and properties (props) passed into the component

# Creating a React Component

- Two ways to create a React Component
  - Class components
  - Functional components
- For most components we create in this course we will use functional components
  - For details on the differences between class components and functional components, see this article:
    - <https://www.twilio.com/blog/react-choose-functional-components>
  - Some older tutorials on React will use Class components
  - For most situations in a modern React app I would recommend sticking with functional components

# Creating a React Component

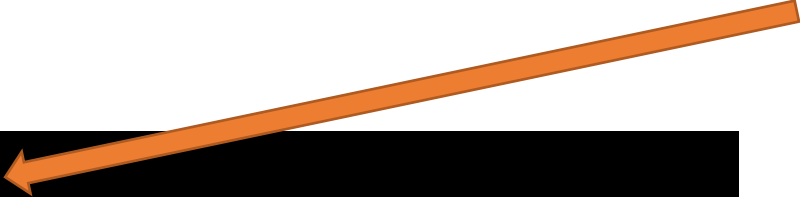
React functional components are just regular JS functions. You can use function declarations or function expressions for React components. The arrow functions syntax also works for React components

```
function Header(){  
  return (  
    <header>  
      <h1>Things To Do</h1>  
    </header>  
  );  
}
```

The only requirement for a React functional component is that it must return JSX

# Start your component name with a capital letter

To avoid name collisions with real HTML elements it is recommended that you start your component names with a capital letter



```
function Header(){  
  return (  
    <header>  
      <h1>Things To Do</h1>  
    </header>  
  );  
}
```

# Adding a Component to your App

```
// Exported component
function App() {
  return (
    <div className="App">
      <Header />
      <TopNav />
      <Main />
      <Footer />
    </div>
  );
}
export default App;
```



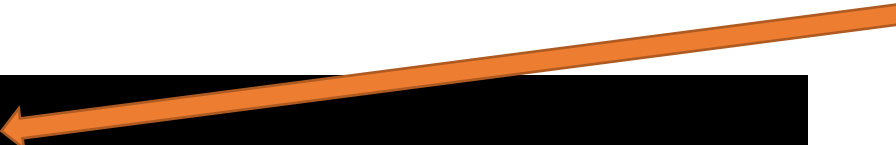
To add your components to your app, simply add them to your main entry component by writing them as JSX elements

This main “App” component is being exported and then imported in the “index.js” file where it is outputted by React. More on imports and exports in the upcoming slides



# React Functional Component Shorthand

If your component only outputs JSX then you can use arrow function shorthand syntax to create the components




```
const Header = () => (  
  <header>  
    <h1>Things To Do</h1>  
  </header>  
) ;
```

# React Component Files

- It is usually preferable to break up your React application into smaller component files
- Use export statements to “export” your components, variables or functions
- Use import statements to “import” a component file (or any JS file) into another component file (or even just a JS file)

# Exporting a React Component

```
const Header = () => (  
  <header>  
    <h1>ThingsTo Do</h1>  
  </header>  
);  
  
export default Header;
```



This statement exports our Header component. The “default” part of the statement makes importing our component a bit easier as we do not have to specify the Header function when importing this component


# Exporting General functions or other values

```
function getYear(){
  const d = new Date();
  return d.getFullYear();
}

function makeDollar(num){
  num = num.toFixed(2);
  num = `$$${num.toLocaleString()}`;
  return num;
}

export { getYear, makeDollar };
```

Import and export statements are part of the JavaScript language. This means you can use export and import statements with regular functions and variables



In this example we are exporting the “getYear” and the “makeDollar” function. Since we do not have a default export, we will have to add these function names in our import statement

# Importing a React Component

```
import Header from './Header';  
import Main from './Main';  
import Footer from './Footer';
```

Use an import statement to import a component.

```
function App() {  
  return (  
    <div className="App">  
      <Header />  
      <Main />  
      <Footer />  
    </div>  
  );  
}
```

The “from” part of the statement tells the compiler where to load the component from. The file path is relative to the importing file

```
export default App;
```

You can add the imported components to your component by writing them as JSX elements

# VS Code ES7 React Snippets Extension

- Consider using a React Code Snippet extension to speed your workflow and reduce the amount of repetitive typing of similar code
- Snippets work by allow you to type a few short characters that then output a block of standard React Code
- There are several React Code Snippet extensions for VS Code
  - ES7 React Snippets
    - <https://marketplace.visualstudio.com/items?itemName=dsznajder.es7-react-js-snippets>
    - To create basic React component
      - Create a new React component file -> example: Widget.js
      - In the Widget.js file type "rfce" ...then the [TAB] key...
    - Optional setting:
      - If you want the outputted basic react component to omit the "import React..." line then add the following line to your "settings.json" file in VS Code
        - "reactSnippets.settings.importReactOnTop": false

# React Props

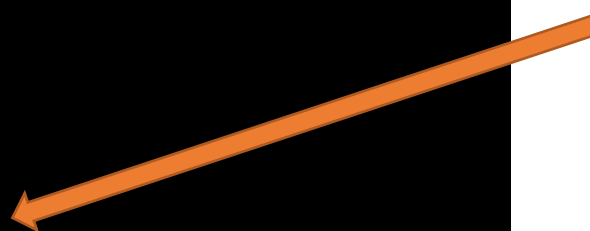
# React Props

- Props allow us to pass data from parent components to child components via HTML like attributes
- See syntax on next page



# React Props Syntax on the Parent Component

```
function App() {  
  const appInfo = {  
    title: 'Things To Do',  
    tasks: [  
      'Buy milk',  
      'Do the dishes',  
      'Wash the car'  
    ]  
  }  
  
  return (  
    <div className="App">  
      <Header title={appInfo.title} />  
      <Main tasks={appInfo.tasks} />  
      <Footer />  
    </div>  
  );  
}
```



Pass data into components using HTML like attributes. You can then access this data in the component via the "props" object which is created for you by React...See the next slide for details

# React Props Syntax on the Child Component

```
const Header = ({ title }) => (  
  <header>  
    <h1>{title}</h1>  
  </header>  
);
```

Grab the passed in data from the parent component via the "props" object which is created for you by React

The word "title" is equal to the prop name we assigned when we called the component in the parent component

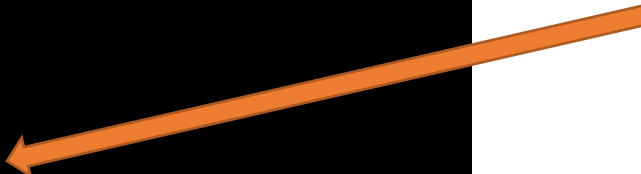
In this example we are outputting the value of the prop in our component

# Default Props


- You can assign prop default values
- If a component has props, but they have not been set when the component is called then React will load the default prop values
- Props passed into a component will always override the default prop values

# Default Props

```
function Header({ title, slogan }) {  
  return (  
    <header>  
      <h1>{title}</h1>  
      <h2>{slogan}</h2>  
    </header>  
  );  
}  
  
Header.defaultProps = {  
  title: 'My First App',  
  slogan: 'The Best App Ever!!!'  
}  
  
export default Header;
```



If no prop values are passed into the component when it is called, React will assign the props their default values, if the default values have been set.



The “defaultProps” object sets the default values for the props of the component.