

# Reminders

- ▶ Lab 4 assigned, Due Tuesday, Nov 28
- ▶ Quiz 10 posted and due end of day Thursday, Nov 16
- ▶ Quiz 11 will come next weekend to cover week 12 materials
- ▶ Tuesday, November 21 will be remote class. Attendance will be taken so please zoom into the class at your scheduled time.



ITMD 441/541

Web Application Foundations

# Week 12

FALL 2023 – NOVEMBER 13, 2023

# Weekly's Agenda

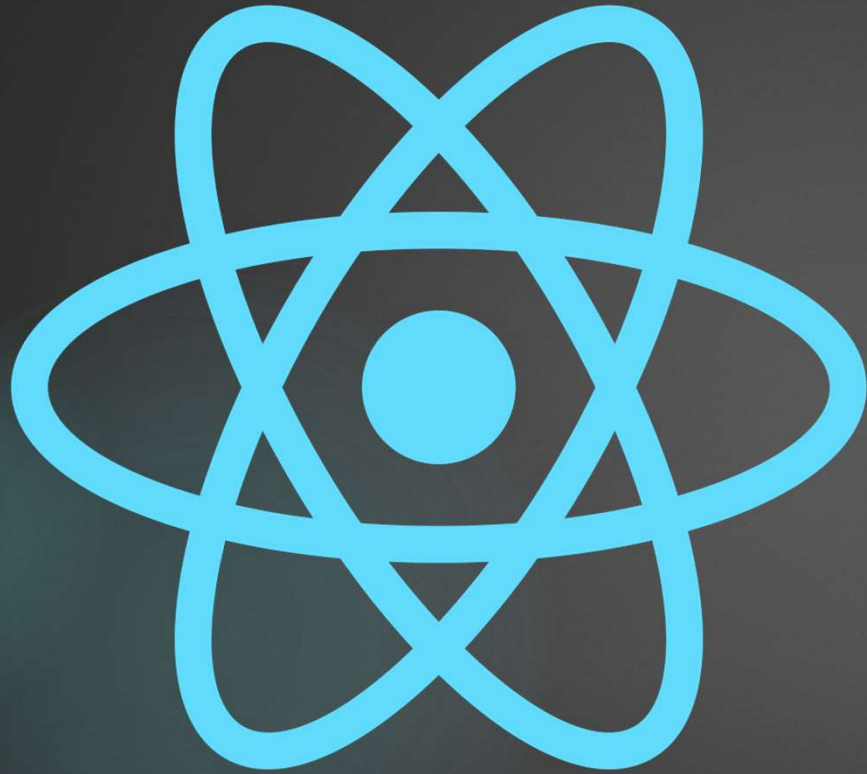
- ▶ Lab 4 discussion
- ▶ Babel for prototyping
- ▶ Front End Frameworks – Fundamentals of React

# Babel

- ▶ Babel is used to convert JSX to plain JavaScript React calls.
- ▶ Babel can also compile our ES6+ to ES5
- ▶ Babel can be directly used in the browser for development purposes

```
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
<!-- Your custom script here -->
<script type="text/babel">
  const getMessage = () => "Hello World";
  document.getElementById('output').innerHTML = getMessage();
</script>
```

- ▶ Need to configure the babel presets or it won't do anything. Typically, in a .babelrc or babel.config.json file. Need to load modules for the presets you want to use and then add them to the config.
- ▶ <https://babeljs.io/setup#installation>



# ReactJS

JAVASCRIPT LIBRARY FOR BUILDING  
USER INTERFACES

# React JS

Watch 100 second video first

- ▶ Created at Facebook by software engineer Jordan Walke
- ▶ An early prototype in 2011 (FaxJS) shipped as a search element on Facebook. Jordan then worked on a prototype for the hard to manage Facebook Ads which became react.
- ▶ Deployed in newsfeed in 2011 and instagram.com in 2012 after Facebook acquired them.
- ▶ Open sourced at JSConf US May 2013
- ▶ Used to build interactive user interfaces for applications that have frequent data changes.
- ▶ When the components internal data changes it automatically updates the markup.
- ▶ Can be used to build UI components, single page applications (SPA), and iOS, Android or UWP when using React Native.
- ▶ [https://en.wikipedia.org/wiki/React\\_\(JavaScript\\_library\)](https://en.wikipedia.org/wiki/React_(JavaScript_library))
- ▶ <https://reactjs.org/>

# React JS

- ▶ What is React? A library for building user interfaces.
- ▶ **Composition, Unidirectional Dataflow, Explicit Mutations, Just JavaScript**
- ▶ You compose a user interface with components
- ▶ Component Based - Components are encapsulated chunks and manage their own state and can be nested inside each other
- ▶ Component logic is all in JavaScript and there are no HTML templates used
- ▶ Data is passed to the component through **properties** or **props** with a one-way data flow
- ▶ Properties or props are a set of **immutable** values that gets passed to the components render function
- ▶ Components should not directly modify the values in props. Instead pass callback functions from the parents via props.

# React JS

- ▶ Uses a “Virtual DOM” in the background and only updates the visible DOM when needed, including only updating the parts that need to.
- ▶ Can be used to build single page application or just components on an existing website or web application
- ▶ It can be directly used in the browser in a script tag or with a bundler to package all the JS files in one big file.
- ▶ Works well as a node js project. All modules used are typically node modules.
- ▶ There is a Create React App node package that will produce a skeleton app as a starting point with a preconfigured build pipeline.
- ▶ <https://reactjs.org/docs/getting-started.html>
- ▶ <https://reactjs.org/docs/react-api.html>
- ▶ Stopped here on 11/14/2023



# React JS - Composition

- ▶ You compose the UI out of multiple components. It looks like HTML but it is how something might be built out of react components.

```
<Container>  
  <NavBar />  
  <Header />  
  <DatePicker>  
    <Calendar />  
  </DatePicker>  
  <Footer />  
</Container>
```



# React JS – Unidirectional Dataflow

- ▶ In a React app data flows from parent component to child component
- ▶ Data is passed as **props** to the component and the props are **immutable**
- ▶ If a component needs to save data/state it needs to do that in itself or get that data as props from a parent
- ▶ Props are added to a component as attributes of the component (in JSX) or as the second argument to a createElement() function call
- ▶ Different than a normal JS/Jquery app that is all event driven and just listening for events and modifying the DOM directly
- ▶ Remember Data flows down to the component and is immutable
- ▶ `<HeaderComponent name="brian" />`. This is considered a prop. A prop looks like an attribute on an HTML.
- ▶ If you need to pass data back up the chain, you need to pass down a function to run and call that function within the component

# React JS – Explicit mutations

- ▶ You will explicitly change or mutate the components state to update the component. If data needs to change in the app you have to explicitly tell it that the data is changing.
- ▶ This makes mutations explicit
- ▶ You use the `setState()` function to trigger this state mutation and pass it the updated state items

```
this.setState({  
  name: 'brian',  
  class: 'itmd4565'  
})
```

Can also pass `setState()` a function that looks at the known good previous state

```
this.setState(prevState => ({  
  seconds: prevState.seconds + 1  
}))
```

# React JS - JSX

- ▶ Can use an optional JSX syntax in your JavaScript code although typically JSX is used
- ▶ JSX is a JavaScript syntax extension
- ▶ This allows us to use what looks like HTML tag syntax to render our components.
- ▶ If we don't use JSX we need use the `React.createElement()` function in its place
- ▶ JSX allows you to evaluate JavaScript expressions in your JSX with curly brackets `{ }`
- ▶ Custom HTML attributes are passed in but need to start with data-
- ▶ Since JSX is not supported by browsers or runtimes you need to convert it to plain JS with a tool like babel.
- ▶ <https://reactjs.org/docs/introducing-jsx.html>

# React JS – React Elements

- ▶ Fundamental object and concept in React apps
- ▶ A React element is a JavaScript object representation that describes how you would like it to appear on the screen. This is a JavaScript representation of a DOM Node.
- ▶ Use `React.createElement()` to create an element. Most fundamental piece.
- ▶ All JSX will be converted to React Elements by babel

`React.createElement( type, [props], [...children] )`

```
const element = React.createElement(  
  'div',  
  {id: 'button'},  
  'Login Now'  
)
```

- ▶ If you are not passing props use `null`, props should be an object

# React JS – ReactDOM

- ▶ React does not provide a way to use elements in a web page by itself
- ▶ react-dom provides the DOM-specific methods that you can use to interact with the DOM in a web page
- ▶ We use the ReactDOM.createRoot().render() method to render a React element into the web DOM

```
const container = document.getElementById('root');
```

```
ReactDOM.createRoot(container).render(<App />);
```

- ▶ <https://reactjs.org/docs/react-dom.html>
- ▶ Click on createRoot. Example near the bottom
- ▶ <https://reactjs.org/docs/react-dom-client.html>

# React JS - Components

- ▶ **A component is a function or a Class which optionally accepts input and returns a React element.**
- ▶ React components can be function based or class based
- ▶ React component names should start with a capital letter
- ▶ Optionally input via props are passed into the function or class
- ▶ A component always returns a React element
- ▶ If your component needs to save any data or state within the component, you need to use a class and save the data in the components state.
- ▶ No DEMOS. Great reading material especially for render()
- ▶ <https://reactjs.org/docs/react-component.html>

# React JS – Functional Components

- ▶ Stateless functional components can be written as a simple function

```
function HelloHeader(props){  
  return (  
    <h1>Hello World!</h1>  
  );  
}
```

Or

```
const HelloHeader = (props) => {  
  return (  
    <h1>Hello World!</h1>  
  );  
}
```

```
ReactDOM.createRoot(container).render(<HelloHeader />);
```



# React JS - Components

- ▶ Components should be designed as small possibility reusable elements that are added to a react app.
- ▶ **The only required method in the component class is the render() function**
- ▶ A basic example is below

```
Class HelloHeader extends React.Component {  
  render() {  
    return <h1>Hello World!</h1>;  
  }  
}
```

# React JS - Components

- ▶ The same basic component is below without JSX

```
Class HelloHeader extends React.Component {  
  render() {  
    return React.createElement('h1', null, 'Hello World!');  
  }  
}
```

<https://reactjs.org/docs/react-without-jsx.html>

# React JS - Components

- Components can use other components.

```
Class HelloHeader extends React.Component {  
  render() {  
    return <h1>Hello World!</h1>;  
  }  
}
```

```
Class Page extends React.Component {  
  render(){  
    return(  
      <div>  
        <HelloHeader />  
      </div>  
    );  
  }  
}
```

# React JS – Data Flow

- Data flows from parent to child via properties and are available in the child with `this.props`. And properties are immutable.

```
Class HelloHeader extends React.Component {  
  render() {  
    return <h1>Hello {this.props.name}</h1>;  
  }  
}
```

```
Class Page extends React.Component {  
  render(){  
    return(  
      <div>  
        <HelloHeader name='Brian' />  
      </div>  
    );  
  }  
}
```

# React JS – Data Flow

- ▶ Inverse Data Flow – Child to Parent
- ▶ Pass a parent's event handler as a property on the child then in the child call that handler to pass data back to the parent for setting state or something else.

```
Class Page extends React.Component {  
  render(){  
    return (  
      <MainHeader />  
      <Greeting searchFunction={this.searchHandler}>Hello  
World!</Greeting>  
      <MainFooter />  
    );  
  }  
  
  searchHandler(e){  
    console.log(e);  
  }  
}
```

# React JS – State

- ▶ State is place we can store data in a component and change the data to see a reflection in the UI.
- ▶ It is available in the `this.state` object in class components
- ▶ Must add a constructor to your class and initialize the state there.

```
class App extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = { header: "Header from state...", content: "Content  
from state..." }  
  }  
  render() {  
    return (  
<div><h1>{this.state.header}</h1><h2>{this.state.content}</h2></div> );  
  }  
}
```

- ▶ Try to make the state as simple as possible and keep as many components stateless as possible. If you have many components that need state, you should make a parent component that has the state in it and pass it down through props.

# React JS – State

- ▶ Do not try to modify the state object directly

```
// Wrong  
this.state.comment = 'Hello';
```

```
// Correct  
this.setState( { comment: 'Hello' } );
```

**Or** `this.setState((currentState) => { return new state});`

- ▶ If you need to set state in one component from another you need to pass handlers down to the child component that then calls `setState()` in the parent that has the state object.
- ▶ The React library watches this state and when it detects changes it compares it to the browser DOM and updates only what is necessary
- ▶ Great Demo in CodePen
- ▶ <https://reactjs.org/docs/state-and-lifecycle.html>

# Controlled Components

- ▶ In React when a form controls data and value is managed by the component's state is called a controlled component.
- ▶ This leads to a single source of truth, the value in state. The value in state is the value you would submit to any server processes.
- ▶ You use the value in state to populate the controls value attribute and update the value in state when the user makes a change.
- ▶ <https://reactjs.org/docs/forms.html#controlled-components>
- ▶ DEMOS:
- ▶ <https://www.w3schools.com/react/default.asp>
- ▶ [https://www.w3schools.com/react/tryit.asp?filename=tryreact\\_getstarted\\_class](https://www.w3schools.com/react/tryit.asp?filename=tryreact_getstarted_class)





# Assignments

# Reading/Assignments

- ▶ Lab 4 assigned, Due Tuesday, Nov 28
- ▶ Quiz 10 posted and due end of day Thursday, Nov 16
- ▶ Quiz 11 will come next weekend to cover tonight's materials