**REACT**

**Crash Course**

**Components**

Components are one of the main building blocks of React. Think of them as sections of an application. You can put components together to make bigger components.

Class Bacon extends React.Component{

Render(){

Return(

<h1>Hello, World</h1>

);

}

}

ReactDOM.render(<Bacon />, document.getElementById(‘root’));

**Rendering multiple components**

Every single component can only return one parent element. So ensure if returning multiple elements to wrap them in a parent <div> tag.

e.g. ReactDOM.render(

<div>

<Bacon />

<Bacon />

</div>,

Document.getElementById(‘root’)

);

**Props**

Props allow you to customise your components:

Class Movie extends React.Component{

Render(){

Return(

<div>

<h1>{this.props.title}</h1>

<h2>{this.props.genre}</h2>

</div>

);

}

}

ReactDOM.render(

<div>

<Movie title=”Star Wars” genre=”Sci-Fi” />

<Movie title=”The Notebook” genre=”Romance” />

</div>,

Document.getElementById(‘root’)

);

//this.props refers to properties and must be wrapped in curly braces in the component.

//{this.props.children} – is a built in property which displays date between opening and closing tags of component. E.g. <Movie>children text here </ Movie>

**Event handling**

Class Comment extends React.Component{

Edit(){

Return(

Console.log(‘Edit btn clicked’)

);

};

Render(){

Return(

<div>

<button onClick={this.Edit}>Edit</button>

</div>

);

}

};

ReactDOM.render(Comment className=”test” />, document.getElementById(‘root’));

**State**

The difference between *state* and *properties* are; *states* can be changed, *props* cannot.

Class Practice extends React.Component{

Constructor(props){

Super(props);

This.state = {name: “Ali Issaee”};

}

Render() {

Return(

<h1>{this.state.name}</h1>

);

}

}

Whenever state changes on a component, the component will automatically re-render and update the DOM if there are any changes. If there are no changes, the DOM will not get touched at all. This allows our application to be extremely fast!

setState() is the one method that you need to know with states:

setTimeout(() => this.setState({name:”Will”}); );

//place this between render() and return(

State only gets used if a component has an internal value that only affects that component and not any of the rest of the appl. Aside from that, you want to use *props*.

Class Practice extends React.Component{

Constructor(props){

Super(props);

This.state = {

Age:27

};

This.newFunction = this.newFunction.bind(this);

};

newFunction() {

var n = this.state.age;

n = n+1;

this.setState({age:n})

};

Render() {

Return(

<div>

<h1>{this.props.name}</h1>

<button onClick={this.newFunction}> ADD </button>

<h4>{this.state.age}</h4>

</div>

);

}

}

ReactDOM.render(<Practice name=”Heading” />, document.getElementById(‘root’));

**Adding state to components**

Class Comment extends React.Component{

Constructor(props){

Super(props);

This.state = {editing: false};

This.edit = this.edit.bind(this);

This.save = this.save.bind(this);

};

Edit() {

This.setState({editing:true});

};

Save() {  
 this.setState({editing:false});

};

editForm() {

return(

<div>

<textarea defaultValue = {this.props.children}></textarea>

<button onClick = {this.save}>Save</button>

</div>

);

};

normalForm() {

return(

<div>

<h1>{this.props.children}</h1>

<button onClick={this.edit}>Edit</button>

</div>

);

};

Render() {

If(this.editing) {

Return this.editForm();

}else{

Return this.normalForm();

}

}

}

ReactDOM.render(

<div>

<Comment>Ali Issaee</Comment>

<Comment>Lauren Foster</Comment>

</div>,

Document.getElementById(‘root’)

);

**Refs**

Refs are used to get reference to a DOM node or an instance of a component in a React application. Do not overuse refs, especially for anything that can be done declaratively.

Class Practice extends React.Component{

Constructor(props){

Super(props);

This.testing = this.testing.bind(this);

}

Testing() {

Var val = this.get.value;

Console.log(val);

}

Render() {

Return(

<div>

<input type=”text” ref={ (getEl) => {this.get=getEl;} } />

<button onClick = {this.testing}>Test</button>

</div>

);

}

}