REACT JS

1.

JS library for building single page apps.

Not plain old JS, but JSX, browser min does conversion/ transpilation

React can be used to generate HTML code, connecting to DBs etc.

2nd

render() takes bit of HTML & throws it into a target.

Tags inside render(<h1> My name </h1>, document.getElementById('example'));

returns reference to div example

3. React Components

Building block of • apps. Part of websites

Put components together to make a bigger component

Basically, a React Class

var Component = React.createClass({ render: function(){ return (<h1> Hello </h1>);

}});

ReactDOM.render(< Component />, document.getElementById('example'));

4. Rendering Multiple components

Every component can return one parent value

Eg:

var Comp = React.createClass({ render: function(){ return (

<h1> Here <h2> ← error

<p> Paragraph </p>); } });

To get around wrap them around div

< div> Returning one parent div

</ div>

ReactDOM. render ( ) can also display < Component /> once.

Again To render multiple components, < div) → 3 times </div>

5. <u>Props</u>

Properties allow to customize Components

```
var Movie = React.createClass ({
        render: function () {

            return (                        → {this.props.title}
             < div >
              < h1 > S1 </h2 >
              < h2 > S2 </h2 >              → { this.props.a }
             </ div >
            );
        }
    });

ReactDOM. render( < Movie /> , " doc. gEBId('cont'));
```

Can also do

```
                                          ↗ prop
ReactDOM. render ( < Movie  title= "A"  g = " a"  />
```

Can also put

```
ReactDOM. render(
      < div > < Mov   a= "a"   A = "A"  />
              < ⊖      "   "    "   " >
              < P      "   "    "   " >
      </ div>
dgEBId   );
```

6. <u>Events</u> :   class is reserved in JS, className

```
var Comment = React.createClass ({
        edit: function () {  alert ('A'); }
        delete: function () {  alert ('D'); }
        render: function () {
              < div  className = " cc" >
               < div  className = " cT" > {this.props.children}  </div>
               < button  onClick = {this. edit }  className=> Edit </button>
              </ div >

ReactDOM. render ( < div >  < Component > Hey </ Component > , docgetEle() )
                                          ↗ Child
```

7. State

Also helps to customize Components

States can change, Properties can't

getInitialState → Built-in Function, returns object of state (First thing if you want to use state)

React automatically watches change to Redraw DOMs

```
var checkBox = React.createClass({
    getInitialState: function() {
        return { checked: true }
    },
    handleChecked: function() {
        this.setState({ checked: !this.state.checked });
    },
    render: function() {
        var msg;
        if (this.state.checked) {
            msg = "checked";
        }
        else { msg = "unchecked"; }
        return (
            <div>
                <input type="checkbox" defaultChecked={this.state.
                    checked} onChange={this.handleChecked}/>
                <h1> Check is {msg} </h1>
            </div>
        )
    }
});
ReactDOM.render(<CB />, dgEBI(' '));
```

8. Adding State to Components

render → Always there for state & components & props.

Don't leave space between text area Tags [→ React takes it as null]

```
var Component = React.createClass({
        getInitialState() : function() {
            return { editing: false }
        },
        edit: function() {
            this.setState({ editing: true });
        },
        save: function() {
            this.setState({ editing: false });
        },
        delete: function() {
            console.log(`deleting`);
        }
        renderNormal$ : function() {
            return(
              <div className="commCont">
                <div className="commText">{this.props.children}</div>
                <btn>
                 <btn>
              </div>  );
            },
        renderForm: function() {
            return (
              <div className="comma">
                  <textarea defaultValue={this.props.children}></textarea>
                  <btn onClick={this.save}>Save</btn>
              </div> );
        }
        render: function() {
            if(this.state.editing) {
                    return this.renderForm();}
              else { return this.renderNormal(); }
```

9. Refs

Previous doesn't save edited text

Better than id, more efficient

```
< textarea ref = "newText" defValue = " " > </textarea>

save: function () {
    var value = this.refs . newText. value;
    console.log (value);
    }
```

10. Multiple Child Components

Make relations / connections between Components

```
var Comment = React . createClass ({ . . . });
var Board = React . createClass ({
        getInitialState: function () {
            return {
                comments : [ '1' , '2' , '3' ]
            }
        },
        render: function() {
            return (
                <div className = "board" >
                    { this . state . comments . map ( function( text , i ) {
                            return (< Comment key = { i } > { text } </Comment>);
                        })
                    }
                </div>
            );
        }
});
ReactDOM. render ( < board / > , doc. gEBId );
```

11. <u>Updating State & Removing Components</u>

In Board

```
eachComment : function(text, i) {
        return ( <Comment key={i} index{i}>
                              {text}
                 </Comment>); },

removeComment: function(i) {
        console.log('Remove'+i); var arr= this.state.comments;
}       arr.splice(i, 1) }.
                ↓     ↘ remove 1 after i
              index
        this. sho setState({comments: arr});
    },
                                new Text, i
updateComment : function (i) {
      arr[i] = newText;
       this.setState({comments: arr});
    },
```

12. <u>Passing F^n s as Props</u>

Can have property equal to F^n.

```
eachComment: function (text, i) {
           return (
               <Comment key={i} index={i} updateCommentText
                                    = {this.updateComment}
                 deleteFrom= {this.removeComment}.

           Rest of it
           );
   },

remove: function() {   // In Comment
        this.props.deleteFromBoard(this.props.index);
    }
save: function() {
    this.props.updateCommentText(value or this.refs.newText.value,
                                        this.props.index);
```

13 Creating new Components:

In Board

To pass values & text, use bind

```
render: function() {
    return (
      <div>
        <btn onClick={this.add.bind(null, "BT")}> Add </btn>
          <div class = >
            { this.state.comments.map(this.eachComment)}; }
          </div>
add: function(text) {
  var arr = this.state.comments;
  arr.push(text);
  this.setState({comments: arr});
},
```