**Research**

Binding

Two-way function binding

Map function

Function spreader? Actual syntax …varname

Example of creating React Component:

|  |
| --- |
| import React from "react"; |
|  | import Object from "ObjectName"; |
|  |  |
|  | import "./filename.css"; |
|  |  |
|  | export default class Display extends React.Component { |
|  | static propTypes = { |
|  | value: PropTypes.string, |
|  | }; |
|  |  |
|  | render() { |
|  | return ( |
|  | <div className="component-display"> |
|  | <div>{this.props.value}</div> |
|  | </div> |
|  | ); |
|  | } |
|  | } |

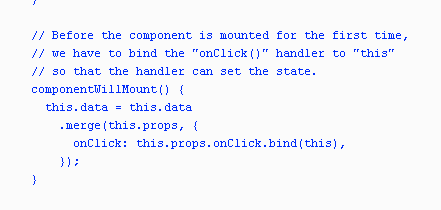
Method called:

React.createElement

COURSE: React: Master Design Patterns with React

* + 1. render – An element describes what you want to see on the screen
    2. componentDidMount – Invoked immediately after a component is mounted/inserted into the tree
       1. Initialization that requires DOM nodes should go here. I you need to load data from a remote endpoint, this is a good place to instantiate the network request.
       2. This method is a good place to set up any subscriptions. If created, then remember to use componentWillUnmount.
    3. componentWillMount – Invoked just before mounting occurs. Called before render, therefore calling setState synchronously in this method will not trigger an extra rendering. Generally we recommend using the constructor instead for initializing state. Avoid introducing any side-effects or subscriptions in this method. For those use cases, use componentDidMount instead. This is the only lifecycle method called on server rendering.
    4. componentWillUnmount – Invoked immediately before a component is unmounted and destroyed. Perform any necessary cleanup in this method, such as invalidating timers, canceling network requests, or cleaning up any subscriptions that were created in componentDidMount. Should not call setState here because component will never be re-rendered. Once a component is unmounted, it will never be mounted again.
    5. setState – Schedules an update to a components state object. When state changes, the component responds by re-rendering.
    6. Props – gets passed to the component whereas state is managed within the component.
    7. componentWillReceiveProps – Invoked before a mounted component receives new props. If you need to update the state in response to prop changes, you may compare this.props and nextProps and perform state transitions using this.setState in this method.
    8. shouldComponentUpdate – Use to let React know if a component’s output is not affected by the current change in state or props. The default behavior is to re-render on every state change, and in the vast majority of cases you should rely on the default behavior. Invoked before rendering when new props or state are being received. Defaults to true. Method is not called for the initial render or when forceUpdate is used. This method only exists as a performance optimization. For a shallow comparison, use PureComponent.
    9. Immutable.js – Data structure which provides immutable, persistent collections that work which makes tracking changes cheap. A change will always result in a new object so we only need to check if the reference to the object has changed.

**The below code allows dynamic creation by allowing a handler for the onclick to be assigned externally upon instantiation of the ladder**



This snippet creates an empty object or whatever depending upon what is passed. Look at Immutable library for more details.

fromJS({})