1. Introduction

* React is open source JS library by Facebook
* Latest version 16.x
* Component based architecture
* Components are reusable
* React is declarative - User should specify how the UI should look like
* Uses virtual dom – It’s like a blueprint for actual dom
* The application will be rendered on div tag “root” in index.html

1. Component

* Reusable, part of UI
* It can be nested
* Component is placed in a JS file
* Functional Component
  + Plain JS function
* Class Component
  + ES6 class which extends React Component
  + Render() returns html
* JSX - JavaScript XML
* JSX - Reduce the complexity with syntax

1. State and Props

* Props – It’s immutable
* State – To overcome the problem with props
* Props is passed to the component – state is managed within the component
* Never modify the state directly. Use this.setState() method instead
* Changing this.state.variable will not re render the component
* SetState() will re render the component
* Calls to setState() is asynchronous – In case if any statement to be executed only after the state change, use a call back method as second param in setState()
* React will group multiple setState() into single setState(). Because of this instead of changing this.state.variable always use prevState.

1. Event Handling

* onClick – use function name instead of function call
  + onClick = {handleEvent()} -> will be rendered automatically when refreshed(wrong way)
  + onClick = {handleClick} -> correct way to trigger event

1. Ways to bind event handler

* Bind the handler in render method:
  + onClick = {this.handleClick.bind(this)} -> generate new event handler every time. This may cause performance issue in huge applications.
* Use arrow function:
  + onClick = {() => this.hanldeClick()} -> This also has performance issues. But it is the easiest way.
* Bind event handler in the constructor:
  + Add the line -> this.handleClick = this.handleClick.bind(this)
  + In click event -> onClick = {this.handleClick}
  + This approach is recommended by React
* Use arrow function in class property:
  + Use arrow function inside class
  + handleClick = () => {
  + this.setState = ({
  + // our code
  + })
  + }
  + This is also recommended by React

1. Lists and Keys:

* Use unique id in the list as key
* When to use index as key:
  + When the list don’t have a unique id
  + The list is static and won’t change in future
  + When the list will never be reordered or sorted

1. Life cycle methods
   1. Mounting

* When an instance of a component is being created and inserted into DOM
  + Constructor
    - Called when a new component is created
    - Used to initialize the state
    - To bind event handlers
    - Need to call super class constructor
  + static getDerivedStateFromProps
    - Rarely used as per React documentation
    - When the state of component depends on props
    - This.setState won’t work as it is static. Return the changed state as object instead
  + Render
    - It is a required method
    - Read props and state and return JSX
    - Don’t change the state or interact with DOM or make ajax call
  + componentDidMount
    - Invoked immediately after a component and it’s child components rendered to the DOM
    - Can make ajax calls, interact with DOM
* Execution order: constructor, getDerivedStatusFromProps, render, componentDidMount
* Execution order if the component has child component: parent constructor, parent getDerivedStatusFromProps, parent render, child constructor, child getDerivedStatusFromProps, child render, child componentDidMount, parent componentDidMount
  1. Updating
* When a component is re rendered by means of props or state
  + Static getDerivedStateFromProps
    - Called everytime when a component is re rendered
    - Rarely used
  + shouldComponentUpdate
    - Dictates if the component should re render or not
    - For performance optimization
    - Rarely used
  + Render
    - It is a required method
    - Read props and state and return JSX
    - Don’t change the state or interact with DOM or make ajax call
  + getSnapshotBeforeUpdate
    - Called right before the changes from virtual DOM are to be reflected in the DOM
    - Rarely used
    - Used to read the current DOM state
  + componentDidUpdate
    - Called only once in each re rendered cycle
    - Called after the render is finished in the re render cycle
    - Can make ajax calls after comparing the previous props with current props
  + Order of execution: parent getDerivedStateFromProps, parent shouldComponentUpdate, parent render, child getDerivedStateFromProps, child shouldComponentUpdate, child render, child getSnapshotBeforeUpdate, parent getSnapshotBeforeUpdate, child componentDidUpdate, parent componentDidUpdate
  1. Unmounting
* When a component is being removed from DOM
  + componentWillUnmount
    - Invoked immediately before a component is unmounted and destroyed
    - Can perform clean up activities
  1. Error handling
* When there is an error during rendering, life cycle method or constructor of a child component
  + Static getDerivedStateFromError
  + componentDidCatch

1. Pure component

* Pure component will implement the shouldComponentUpdate method and compares previous state and props with current props and state. If there is no change it will not re render the component at all. It improves the performance
* When to use: If there is no change in the props and state
* All the children component of pure component should be pure components
* Since it will perform shallow comparison, do not change the object. Because the address of the object remains same during mutation so it will not re render the component. Instead create a new object every time.

|  |  |
| --- | --- |
| Pure Component | Regular Component |
| Implement the shouldComponentUpdate method with a *shallow* state and props comparison | Will not implement shouldComponentUpdate. It always return true by default |

1. Memo component

* Function component version of pure component.
* Export default React.memo(functionComponentName)

1. Portals

* Used to mount a component outside of root div
* Useful when using pop up windows which has different styles
* Steps:
  + Add a div tag in index.html under root tag
  + In the function component instead of using simple return use,

Return ReactDom.createPortal(

<h1>Something</h1>, document.getElementById(‘portal-root’)

)

1. Error boundaries:

* It catches javascript error in their child component tree, log the errors and display fallback UI
* A class component become an error boundary by defining either or both of the getDerivedStateFromError and componentDidCatch methods.

1. Context API

* It prevents the props to be passed through every level
* Steps:
  + Create context
  + Provide context value
  + Consume the context value