Assignment 8

Q1)[ReactJS](https://www.knowledgehut.com/web-development/react-js) is component-based JavaScript library created by Facebook. React makes it easier to create interactive UI using components and efficiently manage states of those components. Multiple components can be composed together to make complex applications without losing their state in DOM.

**Pros:**

**1. Component based architecture**

It is not uncommon for vanilla JavaScript based apps to get into a stage, when managing state of data at user’s browser becomes a headache for developers. As the data and complexity of an application grow, it becomes difficult to maintain using vanilla JavaScript.

The introduction of React components brings a highly sophisticated unit of a web page which can be independently created, maintained, and even reused. You can divide your web page into multiple components, and they can work independently.

**2. High Performance**

With component-based architecture, React allows to create highly scaled Single Page Application or SPA, in which content is dynamically loaded during user interaction without loading the entire page. However, this can turn into a trap. Imagine having to update DOM for every change caused by user’s interaction on web page. Every action might force DOM (which is a tree structure) to refresh itself. And if your web page is complex, having multiple UI components this can cause massive performance blockage.

To solve this, React uses the concept of Virtual DOM, which you can think of as a copy of your real DOM. Now all the changes caused by user’s interaction or other events are handled by the virtual DOM first, and only if it (the intelligence of React) thinks it is important enough to refresh the real DOM, the real DOM is refreshed. This saves us from massively repeating recreation of the DOM tree for every trivial change resulting in high performance application.

Cons:

1. **Not a full-featured framework:** Even though React is a rich JavaScript library with a set of interactive and useful features required for creating large scale applications, developers do not enjoy what they can have in a fully featured framework such as Angular (another popular JS Framework). If you look at the MVC (Model View Controller) architecture, React only handles the view part. For Controller and Model you need additional libraries, and tools. This can result in poor structure of code, and its patterns. Whereas frameworks like Angular provide the complete MVC featured ground, which is more structured, and well managed.
2. **High pace of Development:** This is arguably the most discussed con of using React. React is not just a rapidly growing library, it is also rapidly changing, which forces its developers to update the way they write code. Now this is obviously annoying for most of the developers who are not comfortable with adopting new ways every Monday they start or the ones who are working on an application where changes are critical to customers. There are many industries which are critical to change where customers look for more stable tools and technologies. But this again depends on how expert team members are and if they can convince their customers to trust them with React.

Q2) Virtual DOM: React uses Virtual DOM exists which is like a lightweight copy of the actual DOM(a virtual representation of the DOM). So for every object that exists in the original DOM, there is an object for that in React Virtual DOM. It is exactly the same, but it does not have the power to directly change the layout of the document. Manipulating DOM is slow, but manipulating Virtual DOM is fast as nothing gets drawn on the screen. So each time there is a change in the state of our application, the virtual DOM gets updated first instead of the real DOM.

Q3) **Real DOM:** The [DOM](https://www.geeksforgeeks.org/dom-document-object-model/) represents the web page often called a document with a logical tree and each  branch of the tree ends in a node and each node contains objects programmers can modify the content of the document using a scripting language like javascript and the changes and updates to the [DOM](https://www.geeksforgeeks.org/dom-document-object-model/) are fast because of its tree-like structure but after changes, the updated element and its children have to be re-rendered to update the application UI so the  re-rendering of the UI which make the [DOM](https://www.geeksforgeeks.org/dom-document-object-model/) slow all the UI components you need to be rendered for every [DOM](https://www.geeksforgeeks.org/dom-document-object-model/) update so real [DOM](https://www.geeksforgeeks.org/dom-document-object-model/) would render  the entire list and not only those item that receives the update

**Virtual DOM:**

* VDOM is the virtual representation of Real DOM
* React update the state changes in Virtual DOM first and then it syncs with Real DOM
* Virtual DOM is just like a blueprint of a machine, can do changes in the blueprint but those changes will not directly apply to the machine.
* Virtual DOM is a programming concept where a virtual representation of a UI is kept in memory synced with “Real DOM ” by a library such as ReactDOM and this process is called reconciliation
* Virtual DOM makes the performance faster, not because the processing itself is done in less time. The reason is the amount of changed information – rather than wasting time on updating the entire page, you can dissect it into small elements and interactions

Q4) **Components** enable the creation of complex applications by breaking them down into smaller, self- contained units.

**Functional Components**: it is also known as stateless components. It receive props as input and return JSX as output. These components are primarily focused on rendering UI based on the input props and do not manage their own state.

**Class Components**: it is also known as stateful components. Here the JS classes that extend a base component class provided by the framework. They have their own state and can perform lifecycle methods like mount and update.

**Higher-Order-Components**: these are functions that take a component as input and return as enhanced version of that component. They enable code reuse and add additional behaviors to the component.

Q5) Function based components with hooks have become the recommended approach in React due to their simplicity, readability and the ability to handle complex state and lifecycle logic.

Class based components are still supported and can be useful in certain scenarios, especially when working with older codebases or integrating with third-party libraries that rely on class components.

Q6) In ReactJS, every component creation process involves various lifecycle methods. These lifecycle methods are termed as component's lifecycle. These lifecycle methods are not very complicated and called at various points during a component's life. The lifecycle of the component is divided into four phases. They are:

1. Initial Phase
2. Mounting Phase
3. Updating Phase
4. Unmounting Phase

Each phase contains some lifecycle methods that are specific to the particular phase. Let us discuss each of these phases one by one.

1. Initial Phase

It is the birth phase of the lifecycle of a ReactJS component. Here, the component starts its journey on a way to the DOM. In this phase, a component contains the default Props and initial State. These default properties are done in the constructor of a component. The initial phase only occurs once and consists of the following methods.

* getDefaultProps()  
  It is used to specify the default value of this.props. It is invoked before the creation of the component or any props from the parent is passed into it.
* getInitialState()  
  It is used to specify the default value of this.state. It is invoked before the creation of the component.

2. Mounting Phase

In this phase, the instance of a component is created and inserted into the DOM. It consists of the following methods.

* componentWillMount()  
  This is invoked immediately before a component gets rendered into the DOM. In the case, when you call setState() inside this method, the component will not re-render.
* componentDidMount()  
  This is invoked immediately after a component gets rendered and placed on the DOM. Now, you can do any DOM querying operations.
* render()  
  This method is defined in each and every component. It is responsible for returning a single root HTML node element. If you don't want to render anything, you can return a null or false value.

3. Updating Phase

It is the next phase of the lifecycle of a react component. Here, we get new Props and change State. This phase also allows to handle user interaction and provide communication with the components hierarchy. The main aim of this phase is to ensure that the component is displaying the latest version of itself. Unlike the Birth or Death phase, this phase repeats again and again. This phase consists of the following methods.

* componentWillRecieveProps()  
  It is invoked when a component receives new props. If you want to update the state in response to prop changes, you should compare this.props and nextProps to perform state transition by using this.setState() method.
* shouldComponentUpdate()  
  It is invoked when a component decides any changes/updation to the DOM. It allows you to control the component's behavior of updating itself. If this method returns true, the component will update. Otherwise, the component will skip the updating.
* componentWillUpdate()  
  It is invoked just before the component updating occurs. Here, you can't change the component state by invoking this.setState() method. It will not be called, if shouldComponentUpdate() returns false.
* render()  
  It is invoked to examine this.props and this.state and return one of the following types: React elements, Arrays and fragments, Booleans or null, String and Number. If shouldComponentUpdate() returns false, the code inside render() will be invoked again to ensure that the component displays itself properly.
* componentDidUpdate()  
  It is invoked immediately after the component updating occurs. In this method, you can put any code inside this which you want to execute once the updating occurs. This method is not invoked for the initial render.

4. Unmounting Phase

It is the final phase of the react component lifecycle. It is called when a component instance is destroyed and unmounted from the DOM. This phase contains only one method and is given below.

* componentWillUnmount()  
  This method is invoked immediately before a component is destroyed and unmounted permanently. It performs any necessary cleanup related task such as invalidating timers, event listener, canceling network requests, or cleaning up DOM elements. If a component instance is unmounted, you cannot mount it again.

Q7) Prop drilling refers to the process of passing props through multiple levels of nested components, even when some components do not need those props. It can occur when a deeply nested child component needs to access data or function from its parent component.

To avoid it:

Use React Router for Routing: prop drilling can occur when passing route related props through nested components. Router gives access using hooks, reducing the need for prop drilling.

Use of Redux: we can greatly reduce prop drilling in React application. Instead of passing data and functions through component hierarchies manually, components can access the data they need from the Redux store directly.