**1) What is react?**

->React is powerfull Javascript developed by facebook in 2011.
-> React uses componenct based approach. This approach boost high reusabilty of code.
-> React is well known for developing website front end and mobile application.

**2) What are some important features of react?**

-> Uni directional data flow in react.
-> It can do server side data processing and handling.
-> It uses Virtual DOM.

**3) What is Virtual DOM?**

->Virtual DOM is a simple Javascript object that is the exact lightweight copy of real DOM.
-> It can be consider as NODE tree which have elements, attributes and other properties.

*How it helps->*

Here's what happens when we try to update the DOM in React:

1. The entire virtual DOM gets updated.
2. The virtual DOM gets compared to what it looked like before you updated it. React figures out which objects have changed.
3. The changed objects, and the changed objects only, get updated on the *real* DOM.
4. Changes on the real DOM cause the screen to change.

**4) Differentiate between Virtual DOM and Real DOM**

| **Virtual DOM** | **REAL DOM** |
|---|---|
| Changes can be made easily | Changes can be expensive |
| Minimum memory wastage | High demand of memory and more wastage |
| Can not update HTML directly | It will update HTML directly |
| Faster Update | Slow update |

**5) What is JSX?**

JSX is the abbreviation of Javascript xml
JSX allows us to write HTML in React.
JSX makes it easier to write and add HTML in React.

**6) Can browser read JSX?**

No. Browser can read javascript object. So for making JSX browser readable it has to be transformed in to Java script object.

**7) What are the disadvantages of React?**

It is a UI library only. As such when building something with React we will have to include other libraries to handle other parts of an application.

No predefined way to structure your app (such as services, controllers & views in Angular).

Writing code is complicated as it uses JSX, difficult to learn for biginer.

**8) Diffrence between React and Angular-**

| **Factor** | **Angular** | **React** |
|---|---|---|
| Created By | Google | Facebook |
| Render support | Client Side | Server Side |
| Architecture | Full MVC architecture | Only View |
| Data Flow | Bidirectional Data Flow | Uni directional data flow |

**9) How does rendring works in React?**

Rendering is important aspect in react.
Render function is used for rendering, when this function is called it returns a element, this elemnt will represent a DOM component
Every component must be render in React.

**10) What are the states in react?**

React has a special built-in object called state.
It allows components to create and manage their own data.
State should not be modified directly, but it can be modified with a special method called `setState()`.

**11) What are the props?**

Props are shorthand name given to properties in react.
Props are immutable, means read only
Props follows the hierarchy to pass from parent to child.
We can not pass props from child to parent, ie reversal not supported. This feature ensure react as a Unidirectional data flow.

**12) What is the difference between props and state?**

| Props | State |
|---|---|
| Props are read-only. | State changes can be asynchronous. |
| Props are immutable. | State is mutable. |
| Props allow us to pass data from one component to other components as an argument. | State holds information about the components. |
| Props can be accessed by the child component. | State cannot be accessed by child components. |
| Props are used to communicate between components. | States can be used for rendering dynamic changes with the component. |
| Stateless component can have Props. | Stateless components cannot have State. |
| Props are external and controlled by whatever renders the component. | The State is internal and controlled by the React Component itself. |

**13 ) What is a Higher order component?**

A **higher-order component** (HOC) is an advanced technique in React for reusing **component** logic
It is actually a function that takes one component and returns another component that wraps the original one.

**14) What is redux?**

Redux is a state container for JavaScript application.
Redux makes it easy to manage the state of your application.

**15) Explain the lifecycle methods of React components in detail.**

**The lifecycle of the component is divided into four phases. They are:**

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

## 1. Initial Phase-

It is the **birth** phase of the lifecycle of a ReactJS component.

In this phase, a component contains the default Props and initial State. These default properties are done in the constructor of a component.

This phase consist of 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-

In this phase 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.

**ComponentWillRecieveProps()**

It is invoked when a component receives new props. If we want to update the state in response to prop changes, we 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.

Source- https://www.javatpoint.com/react-component-life-cycle

**16) How routing in react is diffrent from convectional routing?**

Each view is considered as a new file in convectional routing while in react it is considered as a single HTML entity.

ON Navigation whole view got refreshed in conventional routing while in react routing view will not be refreshed only objects will refresh.

**17) Diffrence between flux and redux?**

**18) Can Ajax be used in React?**

Yes any Ajax library like axios, Jquery Ajax can be used with react.

**19) What is the meaning of synthetic events in react?**

**20) What are stateful component in react?**

Stateful components are the entities that stores the changes that happen and place them in memory. It can contains the state object and event handling function, user actions as well.

**21) What are refs in React?**

Refs provide a way to access DOM nodes or React elements created in the render method.

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.myRef = React.createRef();  }
  render() {
    return <div ref={this.myRef} />;  }
}
```

**22) What are the controlled components in React?**

The controlled component is a way that we can handle the form input value using the state.
To change the input value there is only one way to change it is using setState or useState if you are using React Hook

```
import React, { useState } from "react";

export default function App() {
  const [inputValue, setInputValue] = useState("");
  const handleInputChange = (e) => {
    setInputValue(e.target.value)
  }
  const handleSubmitButton = () => {
    alert(inputValue);
  };
  return (
    <div className="App">
      <input value={inputValue} onChange={handleInputChange} />
      <input type="submit" value="submit"
onClick={handleSubmitButton} />
    </div>
  );
}
```

In the above example, we use the controlled component to handle the form input value using React Hooks and every time you will type a new character, handleInputChange is called and it takes in the new value of the input and sets it in the state then you can use this value and print it inside alert when submitting use handleSubmitButton.

**23) What are the uncontrolled components in React?**

The uncontrolled component is like traditional HTML form inputs that we will not be able to handle the value but the DOM will take care of handling the value of the input.
We can get this value using `React Ref` and for example, print it inside alert when submitting or play with this value as we want

```
import React, { useRef } from "react";

export default function App() {
  const inputRef = useRef(null);
  const handleSubmitButton = () => {
    alert(inputRef.current.value);
  };
  return (
    <div className="App">
      <input type="text" ref={inputRef} />
      <input type="submit" value="submit" onClick={handleSubmitButton} />
    </div>
  );
}
```

**24) What is react router?**

React Router is a standard library for routing in React.
It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.

The main Components of React Router are:

**BrowserRouter:** BrowserRouter is a router implementation that uses the HTML5 history API(pushState, replaceState and the popstate event) to keep your UI in sync with the URL. It is the parent component that is used to store all of the other components.

**Route:** Route is the conditionally shown component that renders some UI when its path matches the current URL.

**Link:** Link component is used to create links to different routes and implement navigation around the application. It works like HTML anchor tag.

**Switch:** Switch component is used to render only the first route that matches the location rather than rendering all matching routes. Although there is no defying functionality of SWITCH tag in our application because none of the LINK paths are ever going to coincide. But let's say we have a route (Note that there is no EXACT in here), then all the Route tags are going to be processed which start with '/' (all Routes start with /). This is where we need SWITCH statement to process only one of the statements.

```
import React, { Component } from 'react';
import { BrowserRouter as Router, Route, Link, Switch } from 'react-router-dom';
import Home from './component/home';
```

```
import About from './component/about';
import Contact from './component/contact';
import './App.css';

class App extends Component {
  render() {
    return (
      <Router>
        <div className="App">
         <ul className="App-header">
           <li>
             <Link to="/">Home</Link>
           </li>
           <li>
             <Link to="/about">About Us</Link>
           </li>
           <li>
             <Link to="/contact">Contact Us</Link>
           </li>
         </ul>
         <Switch>
           <Route exact path='/' component={Home}></Route>
           <Route exact path='/about' component={About}></Route>
           <Route exact path='/contact' component={Contact}></Route>
         </Switch>
        </div>
      </Router>
    );
  }
}

export default App;
```

**25) What are the components of Redux in React?**

It has four components -
a. Action : An object that describe the call
b. Reducer: The state change storage unit.
c. Store: It stores state and object tree.
d. Display the data provided by the state.

**26) What Is a Pure Function in JavaScript?**

a. The function always returns the same result if the same arguments are passed in. It does not depend on any state, or data, change during a program's execution. It must only depend on its input arguments.

b. The function does not produce any observable side effects such as network requests, input and output devices, or data mutation.

Same Input => Same Output

```
const add = (x, y) => x + y;

add(2, 4); // 6
```

**Pure Functions = Consistent Results**

The example returns a value based on the given parameters, regardless of where/when you call it.

If you pass 2 and 4, you'll always get 6.

Nothing else affects the output.

**Impure Functions = Inconsistent Results**

```
let x = 2;

const add = (y) => {
  x += y;
};

add(4); // x === 6 (the first time)
```

The first time results in `6`, next time is `10` and so on.

**27) What are pure components in React?**

Pure Components in React are the components which do not re-renders when the value of state and props has been updated with the same values.
If the value of the previous state or props and the new state or props is the same, the component is not re-rendered.
Pure Components restricts the re-rendering ensuring the higher performance of the Component.
Pure Components are more performant in certain cases.

```
class ImpureComponent extends React.PureComponent {
  constructor() {
    super();
    this.state = {
      counter: 0
    }

    // The value of Counter is updated to same value during continues interval

    setInterval(() => {
      this.setState({
```

```
      counter: 0
    });
  }, 1000);
 }

 render() {

  // This function wont be re-rendered in case when the new state is same as previous

  return <b>Counter Value: {this.state.counter}</b>
 }
}
```

**28) What are keys in React?**

Keys are used to React to identify which items in the list are changed, updated, or deleted.
In other words, we can say that keys are used to give an identity to the elements in the lists.

**29) How to set React to production build?**

By setting process.env.NODE_ENV variable to production.

When we set React in production, warnings and other development features are not shown.

**30) What is the diffrence between cloneElement and createElement?**

**31) What is strict mode in React?**

`StrictMode` is a tool for highlighting potential problems in an application.
Like `Fragment`, `StrictMode` does not render any visible UI.
It activates additional checks and warnings for its descendants.

```
import React from 'react';

function ExampleApplication() {
 return (
  <div>
   <Header />
   <React.StrictMode>       <div>
     <ComponentOne />
     <ComponentTwo />
    </div>
   </React.StrictMode>      <Footer />
  </div>
 );
}
```

**32) What will you do if your React Application is rendering slow?**

The cause of slow rending is mostly unnecessary re-rendering operation, There are two main tools provided by React to help-

**React.memo():** This is used to prevent all of the unnecessary rendering in functional component.
**PureComponent:** This is used to prevent all of the unnecessary rendering in class component.

**33) What does calling super() in a React constructor do?**

super() is called inside a react component only if it has a constructor.

Below code doesn't require super:

```
class App extends React.component {
   render(){
      return <div>Hello { this.props.world }</div>;
   }
}
```

However if we have a constructor then super() is mandatory:

```
class App extends React.component {
   constructor(){
      console.log(this) //Error: 'this' is not allowed before super()

   }
}
```

The reason why this cannot be allowed before super() is because this is uninitialized if super() is not called. However even if we are not using this we need a super() inside a constructor because ES6 class constructors MUST call super if they are subclasses. Thus, you have to call super() as long as you have a constructor. (But a subclass does not have to have a constructor).

We call super(props) inside the constructor if we have to use this.props, for example:

```
class App extends React.component{
   constructor(props){
      super(props);
      console.log(this.props); // prints out whatever is inside props

   }
}
```

**33) Why is props passed to the super() function in React?**

There is only one reason when one needs to pass props to super():

When you want to access this.props in constructor.

**Passing:**

```
class MyComponent extends React.Component {
   constructor(props) {
      super(props)

      console.log(this.props)
      // -> { icon: 'home', … }
   }
}
```

**Not passing:**

```
class MyComponent extends React.Component {
   constructor(props) {
      super()

      console.log(this.props)
      // -> undefined

      // Props parameter is still available
      console.log(props)
      // -> { icon: 'home', … }
   }

   render() {
      // No difference outside constructor
      console.log(this.props)
      // -> { icon: 'home', … }
   }
}
```

**34) What are predefined props type in react?**

We can use the *propType* for validating any data we are receiving from props

There are mainly 5 diffrent type of predefined prop types in React-

React.PropTypes.bool
React.PropTypes.func
React.PropTypes.node
React.PropTypes.number
React.PropTypes.string

**35) What is Hook in react?**

**React Hooks** are functions that let us **hook** into the **React** state and lifecycle features from function components. By this, we mean that **hooks** allow us to easily manipulate the state of our functional component without needing to convert them into class components