**1.  Differentiate between Real DOM and Virtual DOM ?**

|  |  |
| --- | --- |
|  | |
| **Real DOM** | **Virtual  DOM** |
| 1. It updates slow. | 1. It updates faster. |
| 2. Can directly update HTML. | 2. Can’t directly update HTML. |
| 3. Creates a new DOM if element updates. | 3. Updates the JSX if element updates. |
| 4. DOM manipulation is very expensive. | 4. DOM manipulation is very easy. |
| 5. Too much of memory wastage. | 5. No memory wastage. |

**2. What is React ?**

* React is a front-end JavaScript library developed by Facebook in 2011.
* It follows the component based approach which helps in building reusable UI components.
* It is used for developing complex and interactive web and mobile UI.
* Even though it was open-sourced only in 2015, it has one of the largest communities supporting it.

**3. What are the features of React ?**

Major features of React are listed below:

1. It uses the **virtual DOM** instead of the real DOM.
2. It uses **server-side rendering**.
3. It follows **uni-directional data flow** or data binding.

**4. List some of the major advantages of React ?**

Some of the major advantages of React are:

1. It increases the application’s performance.
2. It can be conveniently used on the client as well as server side.
3. Because of JSX, code’s readability increases.
4. React is easy to integrate with other frameworks like Meteor, Angular, etc.
5. Using React, writing UI test cases become extremely easy.

**5. What are the limitations of React ?**

Limitations of React are listed below:

1. React is just a library, not a full-blown framework
2. Its library is very large and takes time to understand
3. It can be little difficult for the novice programmers to understand
4. Coding gets complex as it uses inline templating and JSX

**6. What is JSX ?**

JSX is a shorthand for JavaScript XML. This is a type of file used by React which utilizes the expressiveness of JavaScript along with HTML like template syntax. This makes the HTML file really easy to understand. It is an example of JSX:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | render(){      return(    <div>    <h1> This is a JSX Code</h1>             </div>        );  } |

**7. What do you understand by Virtual DOM ? Explain its working ?**

A virtual DOM(Document Object Model) is a lightweight JavaScript object which originally is just the copy of the real DOM. It is a node tree that lists the elements, their attributes and content as Objects and their properties. React’s render function creates a node tree out of the React components.

This Virtual DOM works in three simple steps.

1. Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation.



1. Then the difference between the previous DOM representation and the new one is calculated.
2. Once the calculations are done, the real DOM will be updated with only the things that have actually changed. 

**8. Why can’t browsers read JSX ?**

Browsers can only read JavaScript objects but JSX in not a regular JavaScript object. Thus to enable a browser to read JSX, first, the file needs to be transform JSX file into a JavaScript object using JSX transformers like Babel and then pass it to the browser.

**9. How is React different from Angular ?**

|  |  |  |
| --- | --- | --- |
| React vs Angular | | |
| **TOPIC** | **REACT** | **ANGULAR** |
| *1. ARCHITECTURE* | Only the View of MVC | Complete MVC |
| *2. RENDERING* | Server-side rendering | Client-side rendering |
| *3. DOM* | Uses virtual DOM | Uses real DOM |
| *4. DATA BINDING* | One-way data binding | Two-way data binding |
| *5. DEBUGGING* | Compile time debugging | Runtime debugging |
| *6. AUTHOR* | Facebook | Google |

**React Components**

**10. “In React, everything is a component.” Explain ?**

Components are the building blocks of a React application’s UI. These components split up the entire UI into small independent and reusable pieces. Then it renders each of these components independent of each other without affecting the rest of the UI.

**11. What is the purpose of render() in React ?**

Each React component must have a **render()**mandatorily. It returns a single React element which is the representation of the native DOM component. If more than one HTML element needs to be rendered, then they must be grouped together inside one enclosing tag such as **<form>, <group>,<div>** etc. This function must be kept pure i.e., it must return the same result each time it is invoked.

**12. What is a state in React and how is it used ?**

States are the heart of React components. States are the source of data and must be kept as simple as possible. Basically, states are the objects which determine components rendering and behavior. They are mutable unlike the props and create dynamic and interactive components. They are accessed via **this.state().**

**13. How can you embed two or more components into one ?**

We can embed components into one in the following way:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | class MyComponent extends React.Component{      render(){          return(    <div>    <h1>Hello</h1>                    <Header/>              </div>            );      }  }  class Header extends React.Component{      render(){          return    <h1>Header Component</h1>       };  }  ReactDOM.render(      <MyComponent/>, document.getElementById('content')  ); |

**14. What is Props ?**

Props is the shorthand for Properties in React. They are read-only components which must be kept pure. They are always passed down from the parent to the child components throughout the application. A child component can never send a prop back to the parent component. This help in maintaining the unidirectional data flow and are generally used to render the dynamically generated data.

**14. Differentiate between states and props ?**

|  |  |  |
| --- | --- | --- |
| States VS Props | | |
| **Conditions** | **State** | **Props** |
| 1. Receive initial value from parent component | Yes | Yes |
| 2. Parent component can change value | No | Yes |
| 3. Set default values inside component | Yes | Yes |
| 4. Changes inside component | Yes | No |
| 5. Set initial value for child components | Yes | Yes |

**15. What is arrow function in React? How is it used ?**

Arrow functions are more of brief syntax for writing the function expression. They are also called *‘fat arrow*‘ (**=>**) the functions. These functions allow to bind the context of the components properly since in ES6 auto binding is not available by default. Arrow functions are mostly useful while working with the higher order functions.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | //General way  render() {      return(          <MyInput onChange={this.handleChange.bind(this) } />      );  }  //With Arrow Function  render() {      return(          <MyInput onChange={ (e) => this.handleOnChange(e) } />      );  } |

**16. List some of the cases when you should use Refs ?**

Following are the cases when refs should be used:

* When you need to manage focus, select text or media playback
* To trigger imperative animations
* Integrate with third-party DOM libraries

**17. What are the different phases of React component’s lifecycle ?**

There are three different phases of React component’s lifecycle:

1. **Initial Rendering Phase:** This is the phase when the component is about to start its life journey and make its way to the DOM.
2. **Updating Phase:**Once the component gets added to the DOM, it can potentially update and re-render only when a prop or state change occurs. That happens only in this phase.
3. **Unmounting Phase:**This is the final phase of a component’s life cycle in which the component is destroyed and removed from the DOM.

**18. What is an event in React ?**

In React, events are the triggered reactions to specific actions like mouse hover, mouse click, key press, etc. Handling these events are similar to handling events in DOM elements. But there are some syntactical differences like:

1. Events are named using camel case instead of just using the lowercase.
2. Events are passed as functions instead of strings.

The event argument contains a set of properties, which are specific to an event. Each event type contains its own properties and behavior which can be accessed via its event handler only.

**19. How do you create an event in React?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | class Display extends React.Component({      show(evt) {          // code      },      render() {          // Render the div with an onClick prop (value is a function)          return (    <div onClick={this.show}>Click Me!</div>            );      }  }); |

**20. How** **are forms created in React ?**

React forms are similar to HTML forms. But in React, the state is contained in the state property of the component and is only updated via setState(). Thus the elements can’t directly update their state and their submission is handled by a JavaScript function. This function has full access to the data that is entered by the user into a form.

**21. What do you know about controlled and uncontrolled components?**

|  |  |
| --- | --- |
| Controlled vs Uncontrolled Components | |
| **Controlled Components** | **Uncontrolled Components** |
| 1. They do not maintain their own state | 1. They maintain their own state |
| 2. Data is controlled by the parent component | 2. Data is controlled by the DOM |
| 3. They take in the current values through props and then notify the changes via callbacks | 3. Refs are used to get their current values |

**22. What are Higher Order Components(HOC) ?**

Higher Order Component is an advanced way of reusing the component logic. Basically, it’s a pattern that is derived from React’s compositional nature. HOC are custom components which wrap another component within it. They can accept any dynamically provided child component but they won’t modify or copy any behavior from their input components. You can say that HOC are ‘pure’ components.

**23. What can you do with HOC ?**

HOC can be used for many tasks like:

* Code reuse, logic and bootstrap abstraction
* Render High jacking
* State abstraction and manipulation
* Props manipulation

**24. What are Pure Components ?**

*Pure*components are the simplest and fastest components which can be written. They can replace any component which only has a **render().**These components enhance the simplicity of the code and performance of the application.

**25. What is the significance of keys in React ?**

Keys are used for identifying unique Virtual DOM Elements with their corresponding data driving the UI. They help React to optimize the rendering by recycling all the existing elements in the DOM. These keys must be a unique number or string, using which React just reorders the elements instead of re-rendering them. This leads to increase in application’s performance.

**26. What were the major problems with MVC framework ?**

Following are some of the major problems with MVC framework:

* DOM manipulation was very expensive
* Applications were slow and inefficient
* There was huge memory wastage
* Because of circular dependencies, a complicated model was created around models and views

**27. Explain Flux ?**

Flux is an architectural pattern which enforces the uni-directional data flow. It controls derived data and enables communication between multiple components using a central Store which has authority for all data.  Flux provides stability to the application and reduces run-time errors.



**28.** **What is Redux ?**

Redux is one of the most trending libraries for front-end development in today’s marketplace. It is a predictable state container for JavaScript applications and is used for the entire applications state management. Applications developed with Redux are easy to test and can run in different environments showing consistent behavior.

**29. What are the three principles that Redux follows ?**

1. ***Single source of truth:***The state of the entire application is stored in an object/ state tree within a single store. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.
2. ***State is read-only:***The only way to change the state is to trigger an action. An action is a plain JS object describing the change. Just like state is the minimal representation of data, the action is the minimal representation of the change to that data.
3. ***Changes are made with pure functions:*** In order to specify how the state tree is transformed by actions, you need pure functions. Pure functions are those whose return value depends solely on the values of their arguments.

**30. List down the components of Redux ?**

Redux is composed of the following components:

1. **Action** – It’s an object that describes what happened.
2. **Reducer**–  It is a place to determine how the state will change.
3. **Store** – State/ Object tree of the entire application is saved in the Store.
4. **View** – Simply displays the data provided by the Store.

**31. How are Actions defined in Redux ?**

Actions in React must have a type property that indicates the type of ACTION being performed. They must be defined as a String constant and you can add more properties to it as well. In Redux, actions are created using the functions called Action Creators. Below is an example of Action and Action Creator:

|  |  |
| --- | --- |
| 1  2  3  4  5 | function addTodo(text) {         return {                  type: ADD\_TODO,                   text      }  } |

**32. How is Redux different from Flux ?**

|  |  |
| --- | --- |
| Flux vs Redux | |
| **Flux** | **Redux** |
| 1. The Store contains state and change logic | 1. Store and change logic are separate |
| 2. There are multiple stores | 2. There is only one store |
| 3. All the stores are disconnected and flat | 3. Single store with hierarchical reducers |

**33. What are the advantages of Redux ?**

Advantages of Redux are listed below:

* **Predictability of outcome,**
* **Maintainability**.
* **Server-side rendering**
* **Developer tools**
* **Community and ecosystem**.
* **Ease of testing Organization**

**34. What is React Router ?**

React Router is a powerful routing library built on top of React, which helps in adding new screens and flows to the application. This keeps the URL in sync with data that’s being displayed on the web page. It maintains a standardized structure and behavior and is used for developing single page web applications. React Router has a simple API.

**35. Why** **is switch keyword used in React Router v4 ?**

Although a **<div>** is used to encapsulate multiple routes inside the Router. The ‘switch’ keyword is used when you want to display only a single route to be rendered amongst the several defined routes. The **<switch>**tag when in use matches the typed URL with the defined routes in sequential order. When the first match is found, it renders the specified route. Thereby bypassing the remaining routes.

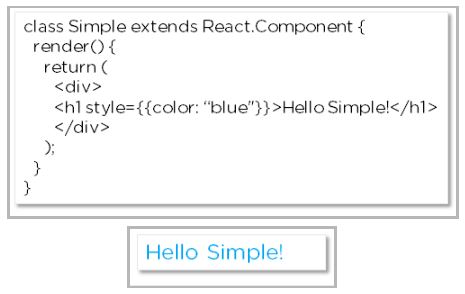
**36. How is React Router different from conventional routing ?**

|  |  |  |
| --- | --- | --- |
| Conventional Routing vs React Routing | | |
| **Topic** | **Conventional Routing** | **React Routing** |
| **PAGES INVOLVED** | Each view corresponds to a new file | Only single HTML page is involved |
| **URL CHANGES** | A HTTP request is sent to a server and corresponding HTML page is received | Only the History attribute is changed |
| **FEEL** | User actually navigates across different pages for each view | User is duped thinking he is navigating across different pages |

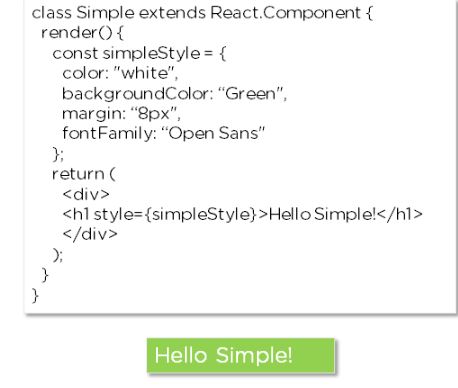
### ****37. How do you style React components ?****

There are several ways in which we can style React components:

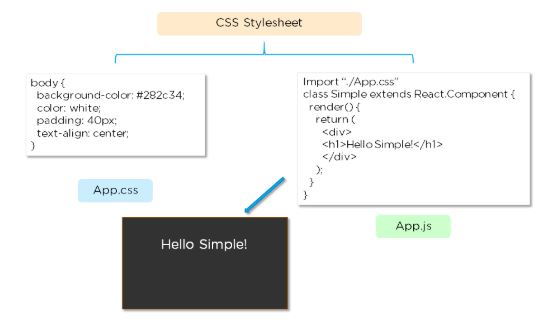
* **Inline Styling**



* **JavaScript Object**



* **CSS Stylesheet**



### ****38. How is React different from React Native ?****

|  |  |  |
| --- | --- | --- |
| Properties | React | React Native |
| Release | 2013 | 2015 |
| Platform | Web | Android/ IOS |
| HTML | Yes | No |
| CSS | Yes | No |
| Prerequisites | Html/CSS/JavaScript | React.JS |