**1.What is React JS**  
React is an open source, JavaScript library for developing user interface (UI) in web application. React is developed and released by Facebook. Facebook is continuously working on the React library and enhancing it by fixing bugs and introducing new features.

**2.What are the major updates for react versions so far.**  
**React JS 0.x**: This was the initial release of React JS, which was released in 2013. This version introduced the concept of virtual DOM, which helped in improving the performance of web applications.

**React JS 15.x**: This was a major release of React JS, which was released in 2016. It introduced several new features like React.PureComponent, improved server-side rendering, and improved performance.

**React JS 16.x**: This version was released in 2017 and introduced several new features like error boundaries, improved server-side rendering, and support for returning an array of elements from a component's render method.

**React JS 17.x**: This version was released in 2020 and focused mainly on improving the existing APIs and making them more stable. It also introduced some minor improvements like improved handling of focus events and improved support for Web Components.

**React JS 18.x**: This version is currently in development and is expected to be released in 2022. It is expected to focus on improving the developer experience by introducing new features like automatic batching of state updates, improved support for server-side rendering, and more efficient handling of large-scale applications.

**3.What are three important concepts of react architecture?**  
**React component:** React component is the primary building block of the React application. It uses React elements and JSX to design its user interface. React component is basically a JavaScript class (extends the React.Component class) or pure JavaScript function. React component has properties, state management, life cycle and event handler. React component can be able to do simple as well as advanced logic.  
**React elements:** JavaScript representation of HTML DOM. React provides an API, React.createElement to create React Element.  
**JSX:** A JavaScript extension to design user interface. JSX is an XML based, extensible language supporting HTML syntax with little modification. JSX can be compiled to React Elements and used to create user interface.

**4.Explain JSX**  
JSX stands for JavaScript XML. It is simply a syntax extension of JavaScript. It allows us to directly write HTML in React (within JavaScript code). It is easy to create a template using JSX in React, but it is not a simple template language instead it comes with the full power of JavaScript.

Example:   
render(){

return( <div> <h1>JavaTpoint</h1> </div> );

}

**5.Explain React components.**A React component represents a small chunk of user interface in a webpage. The primary job of a React component is to render its user interface and update it whenever its internal state is changed. In addition to rendering the UI, it manages the events belongs to its user interface.   
To summarize, react component provides below functionalities.

Initial rendering of the user interface.  
Management and handling of events.  
Updating the user interface whenever the internal state is changed.  
  
**React component accomplish these features using three concepts −**

Properties − Enables the component to receive input.

Events − Enable the component to manage DOM events and end-user interaction.

State − Enable the component to stay stateful. Stateful component updates its UI with respect to its state.

**6.What are the types of react components? Explain the difference.**  
React library has two component types. The types are categorized based on the way it is being created.

1)Function component − Uses plain JavaScript function.

2)ES6 class component − Uses ES6 class.

**The core difference between function and class component are −**

Function components are very minimal in nature. Its only requirement is to return a React element.

function Hello() {

return '<div>Hello</div>'

}

**The same functionality can be done using ES6 class component with little extra coding.**

class ExpenseEntryItem extends React.Component {

render() {

return ( <div>Hello</div> ); }

}

Class components supports state management out of the box whereas function components does not support state management. But, React provides a hook, useState() for the function components to maintain its state.

Class component have a life cycle and access to each life cycle events through dedicated callback apis. Function component does not have life cycle. Again, React provides a hook, useEffect() for the function component to access different stages of the component.

**7.What are the important ways of writing CSS styles in react.**  
Some of the top options are as follows −

1. CSS stylesheet − Normal CSS styles along with className
2. Inline styling − CSS styles as JavaScript objects along with camelCase properties.
3. CSS Modules − Locally scoped CSS styles.
4. Styled component − Component level styles.
5. Sass stylesheet − Supports Sass based CSS styles by converting the styles to normal css at build time.
6. Post processing stylesheet − Supports Post processing styles by converting the styles to normal css at build time.

Let use learn how to apply the three important methodologies to style our component in this chapter.

**CSS Stylesheet, Inline Styling, CSS Modules**

**8.Explain how to write CSS Stylesheet method.**

CSS stylesheet is usual, common and time-tested methodology. Simply create a CSS stylesheet for a component and enter all your styles for that particular component. Then, in the component, use className to refer the styles.Now, open ExpenseEntryItem.css file and add few styles.

div.itemStyle {

color: brown;

font-size: 14px;

}  
Next, open ExpenseEntryItem.js and add className to the main container.

import React from 'react';

import './ExpenseEntryItem.css';

render() {

return (

<div className="itemStyle"> </div> );

}}

function App() {

  return (

    <div className="App">

    </div>

  );

}

export default App;

**9.Explain how to write inline styling.**  
Inline Styling is one of the safest ways to style the React component. It declares all the styles as JavaScript objects using DOM based css properties and set it to the component through style attributes.

Open expense-manager application in your favorite editor and modify ExpenseEntryItem.js file in the src folder. Declare a variable of type object and set the styles.

**itemStyle = {**

**color: 'brown',**

**fontSize: '14px'**

**}**

Then use style={this.itemStyle} inside HTMl element.

export default class LifeCycleMethod extends React.Component{

    constructor(){

        super();

    }

    itemStyle = {

        color: 'brown',

        fontSize: '20px'

     }

    render(){

        console.log('render method')

        return(

            <div style={this.itemStyle}>Life Cycle Method</div>

        )

    }

}

Also, can use this:

 return (

    <div className="App" style={

      {

         color: 'brown',

         fontSize: '14px'

      }

   }>

**10.Explain CSS module declaration type:**Css Modules provides safest as well as easiest way to define the style. It uses normal css stylesheet with normal syntax. While importing the styles, CSS modules converts all the styles into locally scoped styles so that the name conflicts will not happen. Let us change our component to use CSS modules

Open expense-manager application in your favorite editor.

Next, create a new stylesheet, ExpenseEntryItem.module.css file under src/components folder and write regular css styles.

div.itemStyle {

color: 'brown';

font-size: 14px;

}

Here, file naming convention is very important. React toolchain will pre-process the css files ending with .module.css through CSS Module. Otherwise, it will be considered as a normal stylesheet.

Next, open ExpenseEntryItem.js file in the src/component folder and import the styles.

import styles from './ExpenseEntryItem.module.css'

Next, use the styles as JavaScript expression in the component.

<div className={styles.itemStyle}>

Now, we have successfully used the CSS modules in our application.

The final and complete code is −

import React from 'react';

import './ExpenseEntryItem.css';

import styles from './ExpenseEntryItem.module.css'

class ExpenseEntryItem extends React.Component {

render() {

return (

<div className={styles.itemStyle} >

<div><b>Item:</b> <em>Mango Juice</em></div>

<div><b>Amount:</b> <em>30.00</em></div>

<div><b>Spend Date:</b> <em>2020-10-10</em></div>

<div><b>Category:</b> <em>Food</em></div>

</div>

);

}

}

export default ExpenseEntryItem;

**11. Explain props with an example.**  
React enables developers to create dynamic and advanced component using properties. Every component can have attributes similar to HTML attributes and each attribute’s value can be accessed inside the component using properties (props).

For example, Hello component with a name attribute can be accessed inside the component through this.props.name variable.

<Hello name="React" />

// value of name will be "Hello\* const name = this.props.name

React properties supports attribute’s value of different types. They are as follows,

Data type support: String, Number, Datetime, Array, List, Object

**Example:**  
**Parent Component:** Trying to send name and amount values to child component

import React from 'react';

import ExpenseEntryItem from './components/ExpenseEntryItem'

const name = "Grape Juice"

const amount = 30.00

ReactDOM.render(

   <React.StrictMode>

      <ExpenseEntryItem name={name} amount={amount}/>

   </React.StrictMode>,

   document.getElementById('root')

**Child Component**: it tries to accessing the props values provided by parent component in the next code snippet:

import React from 'react'

class ExpenseEntryItem extends React.Component {

   constructor(props) {

      super(props);

   }

   render() {

      return (

         <div>

            <div><b>Item:</b> <em>{this.props.name}</em></div>

            <div><b>Amount:</b> <em>{this.props.amount}</em></div>>

         </div>

      );}

}

export default ExpenseEntryItem;

);

**12.Explain event handling in react.**

Define a method for the event:

Types of function Declaration:  
  log() {

    console.log("Event is fired");

 }

**Lambda Function**: - React provides an alternative syntax using lambda function to define event handler. The lambda syntax is

 log = () => {

    console.log("Event is fired");

 }

If you want to know the target of the event, then add an argument e in the handler method. React will send the event target details to the handler method.

 log(e) {

    console.log("Event is fired");

    console.log(e.target);

 }

The alternative lambda syntax is −

 log = (e) => {

    console.log("Event is fired");

    console.log(e.target);

 }

If you want to send extra details during an event, then add the extra details as initial argument and then add argument (e) for event target.

  log(extra, e) {

    console.log(e.target);

    console.log(extra);

    console.log(this);

 }

The alternative lambda syntax is as follows –

  log = (extra, e) => {

    console.log(e.target);

    console.log(extra);

    console.log(this);

 }

Function declaration and accessing:

**Class Component:**

export default class FunClass extends React.Component{

   classFunction=(arg,e)=>{

     console.warn('Hi I am Class Function',arg);

     console.log(e);

    }

    classFunction2=()=>{

        console.warn('Hi I am Class Function 2');

    }

    render(){

    return(

    <div>

    <button onClick={()=>{this.classFunction(30,Event)}}>Click Me</button>

           <button onClick={()=>{this.classFunction2()}}>Click Me</button>

     </div>

        )

    }

}

**Functional Component:**

import React from 'react';

export default function FunFun(){

     let functionalFunction = (args) =>{

        console.warn('Hi I am from Functional Component',args)

     }

     let functionalFunction2 = () =>{

        console.warn('Hi I am from Functional Component')

     }

    return(

        <div>

            <button onClick={()=>{functionalFunction(10)}} >Click Me</button>

            <button onClick={()=>{functionalFunction2()}} >Click Me</button>

        </div>

    )

}

**13.Explain State management in React.**

State represents the value of a dynamic properties of a React component at a given instance. React provides a dynamic data store for each component. The internal data represents the state of a React component and can be accessed using this.state member variable of the component. Whenever the state of the component is changed, the component will re-render itself by calling the render() method along with the new state.

import React from 'react';

export default class ClassState extends React.Component{

    constructor(){

        super();

        //declaring a state

        this.state = {

            name:"Nikkun",

            surname:"Swain"

        }

    }

    //updating a state

    updateState(){

        let i = 10;

        this.setState({

        name:"Rakesh"

        })

    }

    render(){

        return(

            <div>

                <h2>{this.state.name}</h2>

                <button onClick={()=>{this.updateState()}}>Change State Value</button>

            </div>

        )

    }

}

**14.Explain React http programming or API service call.**

import react from 'react';

export default class APIfetch extends react.Component{

    constructor(props){

        super(props)

        //declare variable to store API data

        this.state ={

            items:[]

        }

    }

    fetchRemoteItems(){

        fetch("https://jsonplaceholder.typicode.com/todos")

        .then((result)=>{return result.json()})

        .then(

            (result)=>{

                var itemsArr = [];

                result.forEach(element => {

                    let itemOb = {

                        userId: element.userId,

                        id:element.id,

                        title:element.title,

                        completed:element.completed

                    }

                    itemsArr.push(itemOb)

                });

                this.setState({

                    items:itemsArr

                })

            },

            (error)=>{

                console.log(error)

            }

        )

    }

    render(){

        return(

            <div>

                <h3>API Fetch</h3>

                <button onClick={()=>{this.fetchRemoteItems()}}>Get Result</button>

                <table border="1" style={{margin:'auto',width:'1000px'}}>

                <tr>

                    <td>Id</td>

                    <td>UserId</td>

                    <td>Title</td>

                    <td>Completed</td>

                </tr>

                {

                    this.state.items.map((item)=>

                        <tr>

                            <td>{item.id}</td>

                            <td>{item.userId}</td>

                            <td>{item.title}</td>

                            <td>{item.completed}</td>

                        </tr>

                    )

                }

                </table>

            </div>

        )

    }

}

**15.Explain how can we use basic form in React.**

import react from 'react';

export default class Form extends react.Component{

    constructor(){

        super()

        this.state={

            firstName:null,

            lastName:null

        }

    }

    formSubmit(){

        console.log(this.state)

    }

    render(){

        return(

            <div>

                <h3>Forms</h3>

                <input type='text' name="firstName"

                onChange={(e)=>{this.setState({firstName:e.target.value})}}/>

                <input type='text' name="lastName"

                onChange={(e)=>{this.setState({lastName:e.target.value})}}/>

                <button onClick={()=>{this.formSubmit()}}>Submit</button>

            </div>

        )

    }

}