1) What is React?

1) React is an open-source JavaScript library for creating user interfaces (UIs).

2) It was developed by Facebook.

3) React is a component-based library. Components are re-usable.

4) React is used for creating dynamic web applications, react can update and render data efficiently when the data changes.

5) React uses virtual representation of DOM to efficiently update and render data in UI. Instead of directly manipulating the actual DOM, react calculates the minimal set of changes needed and updates the virtual DOM, which is efficiently applied to actual DOM.

6) React utilizes JSX (JavaScript XML), It allows developers to write HTML-like syntax within JavaScript code. This helps developers for developing UI fast and easily.

7) React follows a uni-directional data flow pattern, where data flows from parent components to child components via props.

8) React can do only one thing that is it can render data fastly and efficiently in webpages.

9) React provides the facility of integrating with other libraries, frameworks and languages easily to develop additional functionality.

10) React works as a declarative library.

11) React can develop single page applications by using react-router-dom library.

2) DOM Manipulation

In JavaScript

```
<script>
    var h1 = document.createElement('h1')
    h1.innerHTML = 'Hello react'
    var root = document.getElementById('root')
    root.appendChild(h1)
</script>
```

In React

```
--------<script>
    var h1 = React.createElement('h1', {}, 'hello world')
     var root = ReactDOM.createRoot(document.getElementById('root'))
     root.render(h1)
 </script>
```

3) How to install react

Use React and ReactDOM library

1) How to use react by using CDN links

```
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>

<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
```

2) How to install react by using create-react-app tool

=> Here by default webpack will be installed (This is also a build tool)

=> npx create-react-app app-name

=> cd app-name

=> npm start

3) How to install react by using vite tool

- Vite is one of the most popular build tool in market.
- Vite is a build tool that aims to provide a faster development environment.
- Vite also allows us to select the framework we want to work.

=> npm create vite

=> npm i

=> npm run dev

4) Folder structure in react

node modules

- "node_modules" contains external libraries and packages that your React application depends on. These modules are managed by a package manager like npm.
- With a package manager like npm, you can use a single command (npm install) to automatically download and install all the dependencies specified in your project's package.json file.
- No need to send this file to github, it can be generated by using npm install command. So put this folder in .gitignore file.

public folder

- In React.js, the "public" folder is a special directory that contains static assets and files that should be publicly accessible to the client-side of your application.
- The purpose of the "public" folder is to provide a place for assets that don't need to go through the build process. Unlike the "src" folder, which contains the source code of your React application and gets processed by Webpack and Babel, the "public" folder assets are copied as-is to the build output directory during the build process.

index.html file: - This is the main HTML file that serves as the entry point to your React application. It contains a <div> element with an id='root', which acts as the mount point for the React components.

src folder

it's important to note that the "src" folder contains the source code that will be processed and bundled by build tools like webpack, vite and others to create a production-ready version of your application.

- During development, you work in the "src" folder, and the build process outputs the optimized and minified code into a separate "build" or "dist" folder that you can deploy.

- App.js file:

It is the root/ parent component created by default.

- main/index.js file:

This is the linking file, in this file we link root(App) component to index.html file. App component will be rendered in div element of index.html file.

package.json

- It contains a dependencies that lists all the external libraries and packages that your project depends on. When you or someone else clones your project and runs npm install, npm reads the package.json file and installs all the listed dependencies along with their specified versions.

package.lock.json

- To summarize, the package-lock.json file and the package.json file work together to manage dependencies, ensure version consistency, provide reproducibility, and enhance the security and integrity of your React.js project.

- It's important to commit both the package-lock.json and the package.json files to version control systems like Git, so others can have a consistent development environment when working on the project.

.gitignore

- The .gitignore file in a React project is used to specify which files and directories should be ignored by the version control system, such as Git.

In a React project, some typical entries you might find in the .gitignore file include:

- Build Output: Ignore the output directories where the bundled and minified code is generated. These directories are usually named "build" or "dist."

- Node_Modules: Ignore the "node_modules" directory, which contains all installed dependencies. Since dependencies can be easily re-installed using the package manager there is no need to include them in the version control system.

- Environment Variables: Ignore files that store sensitive information, such as API keys, passwords, or configuration files specific to your development environment.

5) Components

= 1) What is component?

1) A component is a reusable block of code, it contains a piece of user interface (UI).

2) User Interface (UI) is a collection of components in react.

3) Components are re-usable.

4) Components can maintain state in it and can receive props from parent and return JSX.

5) Components return JSX, JSX contains UI.

6) Components can render dynamic data in UI by using props and state.

7) Components can be composed together by nesting them within each other or passing them as props to other components, creating a hierarchy of UI elements. This modular approach allows for reusability, maintainability, and separation of concerns in React applications.

2) Types of components in react?

In React there are two types of components.

1) Class component:

- A class component is a type of component that is defined using ES6 classes and

extends the React.Component class from React.

- It can maitain state in it and can receive props from parent and return JSX.

- It has a constructor where the initial state can be defined.

- The render method is a compulsory method in a class component. Render method returns the JSX that defines the component's UI.

- In class components we have to bind "this" keyword when we handle with events.

- Class components can't undersand by browser which need to be converted into pure javascript by using Babel (Transpiler).

2) Functional component:

A functional component is a type of component that is defined using JavaScript function.

- It can maitain state in it and can receive props from parent and return JSX.

- The function body returns the JSX that defines the component's UI.

- No need of constructor and render() method and "this" key word in functional components.

- Functional component is a javascript function which can be undersand by browser easily. No need of conversion.

- After introduction of hooks functional components are not stateless. By using useSate hook we can maintain state in functional component.

- Functional components are simpler and more lightweight compared to class components.

6) JSX (Javascript & XML)

- JSX (JavaScript & XML) is used in React.js for defining and rendering the UI.

- It allows developers to write HTML-like syntax within JavaScript code. This helps developers for developing UI fast and easily.

- Under the hood, JSX is transformed into regular JavaScript code by a process called transpiling. Tools like Babel are used to transpile JSX code into JavaScript code that the browser can understand.

- The transpiled code uses React.createElement() function to create and update the actual DOM elements.

- JSX allows you to embed JavaScript expressions within curly braces {}.

- Using of JSX is a common practice in React development, it is not mandatory.

  React can also work without JSX by using the React.createElement() function directly.

- Take a look at the below code:

```
let jsx = <h1>This is JSX</h1>
```

- This is simple JSX code in React. But the browser does not understand this JSX because it's not valid JavaScript code. This is because we're assigning an HTML tag to a variable that is not a string but just HTML code.

- So to convert it to browser understandable JavaScript code, we use a tool like Babel which is a JavaScript transpiler.

  The React.createElement has the following syntax:

```
React.createElement(type, [props], [...children])
```

  Let's look at the parameters of the createElement function.

  *type can be an HTML tag like h1, div or it can be a React component.

  *props are the attributes you want the element to have.

  *children contain other HTML tags or can be a component.

- When we have two or more jsx sibling elements, JSX will through an error.

```
const App = () => {
 return (
  <p>This is first JSX Element!</p>
  <p>This is another JSX Element</p>
  );
 };    Here We will get an error
```

Solutions to resolve issue:

1) To make it work, the obvious solution is to wrap both of them in some parent element, most probably a div.

2) You can try returning it as an array as shown below:

```
const App = () => {
  return (
      [<p>This is first JSX Element!</p>,<p>This is another JSX Element</p>]
     )
    };
```

3) The other way to fix it is by using the React.Fragment component:

```
const App = () => {
   return (
     <React.Fragment>
     <p>This is first JSX Element!</p>
     <p>This is another JSX Element</p>
     </React.Fragment>
     );
   };
```

- Fragment let you group a list of children without adding extra nodes to the DOM.

**Following are the valid things you can have in a JSX Expression:

- A string like "hello"
- A number like 10
- An array like [1, 2, 4, 5]
- An object property that will evaluate to some value
- A function call that returns some value which may be JSX
- A map method that always returns a new array
- JSX itself

**Following are the invalid things and cannot be used in a JSX Expression:

- Loops
- variable declaration
- function declaration

- An object
- undefined, null, and boolean are not displayed on the UI when used inside JSX.

Summary:

- Every JSX tag is converted to React.createElement call and its object representation.
- JSX Expressions, which are written inside curly brackets, allow only things that evaluate to some value like string, number, array map method and so on.
- In React, we have to use className instead of class for adding classes to the HTML element
- All attribute names in React are written in camelCase.
- undefined, null, and boolean are not displayed on the UI when used inside JSX.

7) Props

1. In ReactJS, "props" short for properties, it is a way for passing data from a parent component to its child components.

2. Props allow you to make your components dynamic and reusable by providing them with the necessary data from their parent components.

3. Props are read-only (immutable) meaning that the child components should not modify the props directly. If a child component needs to modify the data, ((it should be done by sending a callback function from the parent component as a props.

4. Props promote the flow of data from top to bottom in the component hierarchy, following the unidirectional data flow principle of React.

Here's how it works:

- Parent Component: In the parent component, you define a child component and pass data to it using attributes. These attributes are referred to as props in the child component.
- Child Component: In the child component, you can access the data passed from parent through the props object.

1) Parent to child

2) Child to parent

3) Child to child (between siblings)

   1) Child to parent & parent to child

   2) Context API

   3) Redux