**Q.1. What is React Js?**

**Ans :** React.js, often simply referred to as React, is a popular open-source JavaScript library for building user interfaces, particularly single-page applications where you need a fast, interactive user experience. It was developed by Facebook and is now maintained by Facebook and a community of developers.

* **Component-Based Architecture:** React allows you to build encapsulated components that manage their own state and can be composed to create complex UIs. Each component represents a part of the UI and can be reused throughout the application.
* **Declarative Syntax:** React uses a declarative approach to describe the UI. You describe what the UI should look like for a given state, and React takes care of updating and rendering the UI to match that state.
* **Virtual DOM:** React uses a virtual DOM to optimize performance. Instead of manipulating the real DOM directly, React maintains a virtual DOM, which is a lightweight copy of the actual DOM. When the state of a component changes, React updates the virtual DOM first and then efficiently updates the real DOM based on the differences between the virtual DOM and the real DOM.
* **JSX (JavaScript XML):** React uses JSX, a syntax extension that allows you to write HTML-like code within JavaScript. JSX makes it easier to visualize the structure of your components and their UI.
* **Unidirectional Data Flow:** Data in React flows in a single direction, from parent components to child components. This makes it easier to understand how data changes affect the UI.
* **State and Props:** Components can manage their own state (internal data) and receive data through props (external data passed from parent components). State is mutable, while props are immutable.
* **Hooks:** Introduced in React 16.8, hooks are functions that let you use state and other React features without writing class components. Common hooks include useState, useEffect, and useContext.

**Q. 2. What is NPM in React Js ?**

**Ans :** NPM or Node Package Manager, is not specific to React but is widely used in the JavaScript ecosystem, including React development.

* **Package Manager:** NPM is a package manager for JavaScript. It allows developers to install, manage, and share code packages (libraries or modules) that are used in their projects.
* **Installing React:** You can use NPM to install React and its related libraries. For example, running npm install react react-dom will install React and ReactDOM into your project.
* **Scripts:** In React projects, you often define scripts in the package.json file for common tasks like starting the development server (npm start), building the project (npm run build), and running tests (npm test)
* **Registry:** NPM provides access to a vast repository of open-source libraries and tools through its online registry. This registry hosts packages that developers can include in their projects.
* **Managing Dependencies:** React projects typically depend on various libraries and tools (e.g., React Router, Redux, testing libraries). NPM helps manage these dependencies by listing them in your package.json file and handling their installation and updates.
* **Command Line Interface (CLI):** NPM comes with a command-line tool that you use to interact with the registry, install packages, manage dependencies, and run scripts

**Q.3. What is Role of Node Js in react Js?**

**Ans : Role of Node.js in React Development**

* **Development Server**: Node.js can be used to run a development server that serves your React application during development. Tools like Create React App use Node.js to set up and run a development server with features like hot reloading and error reporting.
* **Build Tools**: Node.js is used to run various build tools and task runners that are part of the React ecosystem. For example:
* **Webpack**: Bundles JavaScript, CSS, and other assets.
* **Babel**: Transpiles modern JavaScript (ES6+) into a format compatible with older browsers.
* **ESLint**: Provides linting to ensure code quality and consistency.
* **Server-Side Rendering (SSR)**: Node.js can be used to set up server-side rendering for React applications. This involves running React on the server to render the initial HTML of your application, which can improve performance and SEO. Frameworks like Next.js are built on top of Node.js and React to facilitate SSR.
* **Package Management**: Node.js includes npm (Node Package Manager), which is essential for managing JavaScript packages and dependencies in your React project. This includes React itself, related libraries, and build tools.
* **Tooling and Development Environment**: Many modern development tools and frameworks used with React are built on Node.js. This includes:
* **Create React App**: A tool for bootstrapping React projects with a preconfigured build setup.
* **Next.js**: A React framework that supports server-side rendering and static site generation.
* **React.js** is a JavaScript library for building user interfaces, typically focused on the client side
* **APIs and Backend Services**: Node.js can be used to build backend services and APIs that your React application interacts with. For example, a Node.js server might handle data storage, authentication, and other server-side logic, providing an API that your React frontend consumes.
* **Node.js** is a runtime environment that executes JavaScript on the server side and provides tools and libraries to help with tasks such as building, running, and managing React applications.

**Q.4. What is CLI command In React Js?**

**Ans :** In the context of React.js, CLI (Command Line Interface) commands are used to perform various tasks related to project management and development. These commands are typically executed in your terminal or command prompt and leverage tools like create-react-app, react-scripts, and other build tools.

**create-react-app**

create-react-app is a popular tool for bootstrapping a new React project with a pre-configured development environment. Once installed globally, you can use it to create new React applications.

**Create a new React app**

**npx create-react-app my-app**

or if create-react-app is installed globally:

**create-react-app my-app**

**Common Commands for React Projects**

Once you have set up a React project using create-react-app, you will have access to several CLI commands via react-scripts, which are included as part of the development dependencies. These commands help you manage and interact with your React application

**Start the development server**:

**npm start**

or if you are using Yarn:

**yarn start**

This command starts the development server, typically running at [http://localhost:3000](http://localhost:3000/), and enables features like hot reloading

**Build the project for production**:

**npm run build**

or with Yarn:

**yarn build**

This command creates an optimized production build of your React application, placing the build artifacts in the build directory

**Run tests**

**npm test**

or with Yarn:

**yarn test**

**Eject (advanced)**

**npm run eject**

**yarn eject**

**Q.5. What is Components in React Js?**

**Ans :** In React JS, a "component" is a fundamental building block of a React application. Components are reusable, self-contained pieces of the user interface (UI) that can be thought of as JavaScript functions or classes. They manage their own state and render the UI based on that state.

* **Class Components**: These are ES6 classes that extend from React.Component and include a render method to return React elements. They can also have state and lifecycle methods.

Example

class Welcome extends React.Component {

render() {

return <h1>Hello, {this.props.name}</h1>;

}

}

* **Functional Components**: These are simple functions that accept props (short for properties) and return React elements describing what should be rendered on the screen. They can use React hooks to manage state and side effects. Example:

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

3. **State**: Components can manage their own state. State is an object that holds data that may change over time and affect the rendering of the component. State is primarily used in class components and functional components using hooks like useState.

4. **Props**: Props are inputs to components. They are used to pass data from parent components to child components. Props are read-only and cannot be modified by the component receiving them.

5. **Lifecycle Methods**: In class components, lifecycle methods allow you to run code at specific points in a component’s lifecycle, such as when it mounts, updates, or unmounts. For functional components, similar behavior can be achieved using hooks like useEffect.

6. **Composition**: Components can be nested inside other components, allowing for a hierarchical structure of the UI. This promotes reusability and separation of concerns.

7. **JSX**: Components typically use JSX (JavaScript XML), a syntax extension that allows you to write HTML-like code within JavaScript. JSX gets compiled into React elements that the React library uses to build the UI.

8. **Hooks**: In functional components, hooks provide a way to use state and other React features without writing a class. Common hooks include useState, useEffect, and useContext.

**Q.6. What is Header and Content Components in React Js?**

**Ans :** In React.js, components are the building blocks of your user interface. They allow you to encapsulate and reuse code efficiently. The terms "Header" and "Content" refer to common types of components you might create in a React application:

**Header Component**

* **Purpose**: Typically, the Header component is used to display navigation elements, branding (like a logo), or any other global elements that appear at the top of your application or page.
* **Example Usage**: It might include a navigation bar, a logo, and a search bar.
* **Structure**: The Header component is usually placed at the top of the page layout and remains consistent across different views or pages.



**Content Component**

* **Purpose**: The Content component represents the main body or the primary information of the page. It is often used to display varying content depending on the current view or route.
* **Example Usage**: It could contain text, images, forms, or other elements relevant to the specific view or route.
* **Structure**: The Content component is generally dynamic and can change based on the application’s state or the data passed to it



**Integration in a Layout**

Here's how you might integrate these components into a basic layout:



**Q.7. How to install React Js on Windows, Linux Operating System? How to Install NPM and How to check version of NPM**

**Ans :** Installing Node.js and npm

**Windows:**

**Download Node.js:** Go to the [Node.js website](https://nodejs.org/).

Download the Windows installer (LTS version is recommended for most users).

**Run the Installer:**Open the downloaded .msi file and follow the installation prompts. This will install both Node.js and npm.

**Verify Installation:**Open Command Prompt (cmd) and run:

node –v npm –v

**Linux:**

**Update Package Index:** Open your terminal and update the package index: sudo apt update

**Install Node.js and npm:** You can install Node.js and npm from the NodeSource repository for the latest versions: sudo curl -fsSL <https://deb.nodesource.com/setup_18.x> | sudo -E bash - sudo apt install -y nodejs

Replace 18.x with the latest LTS version if needed.

**Verify Installation:** Check the installed versions:  *Creating a React App*

Once Node.js and npm are installed, you can use the npx tool (which comes with npm) to create a new React application.

* **Open your terminal (Command Prompt on Windows or terminal on Linux).**
* **Run the following command to create a new React app:**

npx create-react-app my-app Replace my-app with your desired application name.

**Navigate to your new app directory:**

cd my-app

**Start the React app:** npm start

**Checking npm Version** To check the version of npm installed on your system:

* **Open your terminal or Command Prompt.**
* **Run the following command:**

npm –v

**Q.8. How to check version of React Js?**

Ans Check package.json File

**Open the package.json File:**

Navigate to your project's root directory where the package.json file is located.

**Locate React Dependencies:**

Open package.json in a text editor or IDE and look for the dependencies section. You should see something lik. The version numbers for react and react-dom are listed here.

"dependencies": {

"react": "^18.2.0",

"react-dom": "^18.2.0"

}

**2. Using npm Commands**

* **Open Terminal or Command Prompt:**
* Navigate to your project directory where node\_modules is present.
* **Run npm List Command:**

npm list react

*  This command will show the installed version of React in your project, along with its dependencies.

**Run npm List Command with Depth Option:**

npm list react --depth=0

* Adding --depth=0 will limit the output to just the top-level packages, making it easier to find the version of React.

**3. Using Yarn Commands**

If you use Yarn as your package manager instead of npm:

* **Open Terminal:** Navigate to your project directory.
* **Run yarn list Command:**

**yarn list --pattern react** This command will list the version of React and other related packages.

**4. Programmatically Check Version**

If you want to check the React version programmatically from within your application:

**Create a JavaScript file (e.g., checkReactVersion.js):**

const React = require('react');

**Run the file with Node.js:** node checkReactVersion.js This will output the version of React being used.