React Study Guide

# 1. What is React?

React is a JavaScript library used for building user interfaces, primarily for single-page applications where you want a dynamic and highly interactive experience for users. It allows developers to create reusable UI components, leading to more efficient development and a consistent user experience.

- Component-Based Architecture: Breaks down the UI into small, reusable pieces called components.  
- Virtual DOM: Instead of manipulating the browser's DOM directly, React uses a virtual representation of the DOM that is lightweight and efficient.  
- Declarative: React allows you to describe how the UI should look based on the current application state, and it handles the rendering efficiently.  
- Unidirectional Data Flow: Data flows in one direction, making it easier to manage and debug.

# 2. Prerequisites of React JS

Before learning React, it’s helpful to have a foundational understanding of:  
- HTML/CSS: Understanding how to structure and style web pages.  
- JavaScript (ES6+): Key concepts such as arrow functions, classes, modules, `this` keyword, and promises are commonly used in React code.  
- Basic Node.js and npm: React development involves Node.js and npm (Node Package Manager) for installing dependencies and running tools.  
- Version Control: Familiarity with Git helps manage source code.

# 3. What is TypeScript? How is it helpful? Explain with an example.

TypeScript is a superset of JavaScript that adds static typing to the language. It helps by allowing developers to specify types, leading to fewer bugs and more readable code. TypeScript is particularly useful in large-scale projects where managing data types across different parts of the application is essential.

- Static Typing: Types are enforced at compile time, preventing runtime errors.  
- Code Readability: Improves maintainability by making the code easier to understand and work with.  
- Better Tooling: TypeScript provides better IntelliSense, error detection, and autocompletion in modern editors like VS Code.  
- Cross-platform: Compiles down to plain JavaScript, making it compatible with any JavaScript environment.

Example:  
```typescript  
function add(a: number, b: number): number {  
 return a + b;  
}  
  
const result = add(5, 10);  
console.log(result); // Output: 15  
```

# 4. What is a Component in React?

In React, a component is a building block of the user interface. It encapsulates how the UI should look and behave. Components can be classified into two main types:  
- Functional Components: These are stateless components defined as functions. They receive data through props and return the UI.  
- Class Components: These are stateful components that can handle lifecycle methods and state management.

Example of a Functional Component:  
```jsx  
function Welcome(props) {  
 return <h1>Hello, {props.name}</h1>;  
}  
export default Welcome;  
```

Example of a Class Component:  
```jsx  
class Welcome extends React.Component {  
 render() {  
 return <h1>Hello, {this.props.name}</h1>;  
 }  
}  
export default Welcome;  
```

# 5. How to Set Up Node.js

Node.js is a runtime environment that allows you to run JavaScript code outside of a browser. It’s often used in React development to manage packages and run build tools.  
Steps to Set Up Node.js:  
1. Download Node.js: Go to the official website [Node.js](https://nodejs.org/en/) and download the installer for your operating system.  
2. Install Node.js: Run the downloaded installer and follow the installation prompts.  
3. Install npm: npm comes preinstalled with Node.js. To check if npm is installed, run `npm -v`.  
4. Create a React App: Run the command `npx create-react-app my-app` to create a new React application.