Baisc terminologies of react

React is a JavaScript library for building user interfaces. Here are some basic terminologies and concepts you should be familiar with when working with React:

1. **Component**: The building blocks of a React application. Components are reusable, self-contained pieces of code that define how a part of the user interface should appear and behave.
2. **JSX (JavaScript XML)**: JSX is a syntax extension for JavaScript used in React to describe what the UI should look like. It allows you to write HTML-like code within your JavaScript.
3. **Props (Properties)**: Props are short for properties and are used to pass data from a parent component to a child component. They are immutable and help make your components dynamic and reusable.
4. **State**: State is a JavaScript object that stores data relevant to a component. Unlike props, it is mutable and can be changed over time. State changes trigger component re-renders.
5. **Stateful Component**: Also known as class components, these components have state and lifecycle methods. They are created using ES6 classes and are used when a component needs to manage its own state.
6. **Functional Component**: These are stateless components created as JavaScript functions. They receive props and return JSX. Functional components are simpler and recommended when state and lifecycle methods are not needed.
7. **Render**: The `render()` method is called to determine what should be displayed on the screen. It returns JSX representing the component's output.
8. **Virtual DOM (Document Object Model)**: React uses a virtual representation of the DOM to optimize updates. When changes are made to the UI, React updates the virtual DOM first and then calculates the minimal changes needed to update the actual DOM.
9. **Component Lifecycle**: React components have a lifecycle that consists of various phases, such as mounting, updating, and unmounting. Lifecycle methods like `componentDidMount`, `componentDidUpdate`, and `componentWillUnmount` allow you to perform actions at specific points in a component's life.
10. **Event Handling**: React allows you to handle user interactions (e.g., clicks, input changes) by attaching event handlers to JSX elements. Event handlers are functions that execute when events occur.
11. **Conditional Rendering**: You can conditionally render parts of your component's UI based on conditions or state using constructs like `if` statements or the ternary operator.
12. **Lists and Keys**: React provides mechanisms for rendering lists of elements and using unique keys to help React efficiently update and re-render list items.
13. **Hooks**: Introduced in React 16.8, hooks are functions that allow functional components to manage state and use lifecycle features previously only available in class components. Common hooks include `useState`, `useEffect`, and `useContext`.
14. **Context**: Context is a way to share data between components without explicitly passing props through the component tree. It's useful for global data like themes, user authentication, or language preferences.
15. **Router**: React Router is a library for handling navigation and routing in React applications. It allows you to create single-page applications with multiple views.

16. **Redux**: Redux is a state management library for handling complex application state. It provides a centralized store and a predictable state management pattern.
17. **Props Drilling**: When you need to pass data through multiple levels of nested components, it's called "props drilling." This can be mitigated with context or state management solutions like Redux.

These are some of the fundamental concepts and terms you'll encounter when working with React. Understanding these concepts is crucial for building React applications effectively.