# React

## 1. React components

React components are the building blocks of any React application. They are reusable, self-contained units of code that define the structure, behavior, and appearance of a user interface. A component can represent anything from a simple button to a complex dashboard. Components allow developers to break down the UI into smaller, manageable parts, making development more organized and scalable.

---

## 2. Differences between components and JavaScript functions

While both React components and JavaScript functions can perform tasks, there are key differences:

- **Purpose**: JavaScript functions are general-purpose code blocks used to perform specific tasks or return values. React components, on the other hand, are specialized functions or classes that return React elements to render UI.
- **Return Type**: A typical JavaScript function returns a value like a number, string, or object. A React component returns JSX (JavaScript XML), which is eventually rendered as HTML in the browser.
- **Usage**: React components can maintain their own state, handle lifecycle events, and respond to user inputs, while plain JavaScript functions do not have this capability unless integrated into a component.

---

## 3. Identify the types of components

There are two primary types of components in React:

1. **Class Components**: These are ES6 classes that extend from React.Component and can use state and lifecycle methods.
2. **Function Components**: These are JavaScript functions that return JSX. With the introduction of React Hooks, function components can also manage state and side effects, making them as powerful as class components.

---

## 4. Class components

A class component is a React component defined using a JavaScript class that extends the ***React.Component*** base class. It must include a render() method, which returns the JSX to be rendered on the screen. Class components can hold and manage their own state and make use of lifecycle methods such as componentDidMount() or componentWillUnmount() to handle operations at different stages of the component's lifecycle.

Example:

```javascript
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

---

## 5. Function component

A function component is a simpler way to write React components using JavaScript functions. It accepts props as arguments and returns JSX. With React Hooks like useState() and useEffect(), function components can now handle state and lifecycle logic, making them equally capable as class components but more concise and readable.

Example:

```javascript
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

---

## 6. Component constructor

The constructor in a class component is a special method used for initializing the component's state and binding event handlers. It is called automatically when a component is created. In React, the constructor is optional, but when used, it typically calls super(props) to ensure the component inherits from React.Component correctly.

Example:

```javascript
constructor(props) {
  super(props);
  this.state = { count: 0 };
}
```

---

## 7. Render() function

The render() function is a required method in class components. It is responsible for describing what the UI should look like. The method returns a React element (JSX) that represents the component's UI structure. It is called automatically whenever there is a change in the component’s state or props.

Example: