Components

- Components are the building blocks of any React app.
- Components let you split the UI into independent, reusable pieces, and think about each piece in isolation.
- Components are like JavaScript functions. They accept arbitrary inputs (called “props”) and return React elements describing what should appear on the screen.
- Always start component names with a capital letter.
- React treats components starting with lowercase letters as DOM tags. For example, represents an HTML div tag, but <App /> represents a component requires App to be in scope.

Function Components

It is a JavaScript function which accepts a single “props” object argument with data and returns a React Element.

Syntax:-

```
function func_name ( ) {
   return React Element;
}
```

Ex:-

```
function Student( ){
   return <h1>Hello Rahul</h1>
}
```

```
const Student = ( ) =>{
   return <h1>Hello Rahul</h1>
}
```

Function Components

Syntax:-

```
function func_name (props) {
  return React Element;
}
```

Ex:-

```
function Student(props){
  return <h1>Hello Rahul</h1>
}
function Student(props){
  return <h1>Hello {props.name}</h1>
}
```

```
const Student = (props) =>{
  return <h1>Hello {props.name}</h1>
}
```

Class Component

A class component requires you to extend from React.Component. The class must implement a render() member function which returns a React component to be rendered, similar to a return value of a functional component. In a class-based component, props are accessible via this.props.

Syntax:-

```
class class_name extends Component {
  render( ){
      return React Element
   }
}
```

Ex:-

```
class Student extends Component {
  render( ){
      return <h1>Hello Rahul</h1>
   }
}
```

```
class Student extends Component {
  render( ){
      return <h1>Hello {this.props.name}</h1>
   }
}
```

Rendering a Component

```
ReactDOM.render(<Student />, document.getElementById("root"));
ReactDOM.render(<Student  name="Rahul"/>, document.getElementById("root"));
```

Ex:-

```
function Student(props){
   return <h1>Hello {props.name}</h1>
}
ReactDOM.render(<Student name="Rahul"/>, document.getElementById("root"));
```

When React sees an element representing a user-defined component, it passes JSX attributes to this component as a single object. We call this object "props".

Composing Components

Components can refer to other components in their output. This lets us use the same component abstraction for any level of detail.

Ex:-

```
function Student(){
  return <h1>Hello Rahul</h1>
}
function App( ){
  return (
   <div>
         <Student / >
         <Student / >
        <Student / >
  </div>
);   }
ReactDOM.render(<App />, document.getElementById(“root”));
```

Functional vs Class Component

Use functional components if you are writing a presentational component which doesn't have its own state or needs to access a lifecycle hook. You cannot use setState() in your component because Functional Components are plain JavaScript functions,

Use class Components if you need state or need to access lifecycle hook because all lifecycle hooks are coming from the React.Component which you extend from in class components.