Import and Export

A.js

```
export let name="Rahul Chauhan";
//export make this variable outside the file
```

B.js

```
import { name } from "./A.js";
// .js is optional in React
```

Note: Variable, function and class can be exported and imported.

React Render

- Render means providing HTML in a web page.
- React renders HTML to the web page by using a function called **createRoot()** and its function **render().**

The createRoot Function

This function returns the main element(can say root element), where all the HTML content goes.

The render Function

This function is used to provide the HTML content.

React Render

index.js

```
//Getting the main container from index.html page.
const root = createRoot(document.getElementById('root'));
//Providing the HTML content to that container.
root.render(<p>Hello</p>);
```

index.html

```
<body>
        <div id="root"></div>
</body>
```

JSX

- JSX stands for **JavaScript XML**.
- JSX allows us to write HTML in React.

With JSX

```
const ele = <h1>JSX is Best!</h1>;
const root = createRoot(document.getElementById('root'));
root.render(ele);
```

Expression in JSX

- With JSX you can write expressions inside curly braces { }.
- The expression can be a React variable, or property, or any other valid JavaScript expression.
- JSX will execute the expression and return the result:

```
const ele = <h1>React is {2*4} times better with JSX</h1>;
```

JSX Fragment

The HTML code must be wrapped in *ONE* top level element.
So if you like to write *two* paragraphs, you must put them inside a parent element, like a **div** element.

```
const ele = (
<div>
        <p>I am a paragraph 1.</p>
        <p>I am a paragraph 2.</p>
</div> );
```

Alternatively, can use a **fragment** "<><>" to wrap multiple lines.

```
const ele = (
<>
        <p>I am a paragraph 1.</p>
        <p>I am a paragraph 2.</p>
</> );
```

JSX Fragment

JSX follows XML rules, and therefore HTML elements must be properly closed.

```
const ele = <input type="text" />;
```

JSX will throw an error if the HTML is not properly closed.

Attribute class = className

The class attribute is a much used attribute in HTML, but since JSX is rendered as JavaScript, and the class keyword is a reserved word in JavaScript, you are not allowed to use it in JSX.

```
const ele = <h1 className="myclass">Hello INCAPP</h1>;
```

JSX Conditional Statement

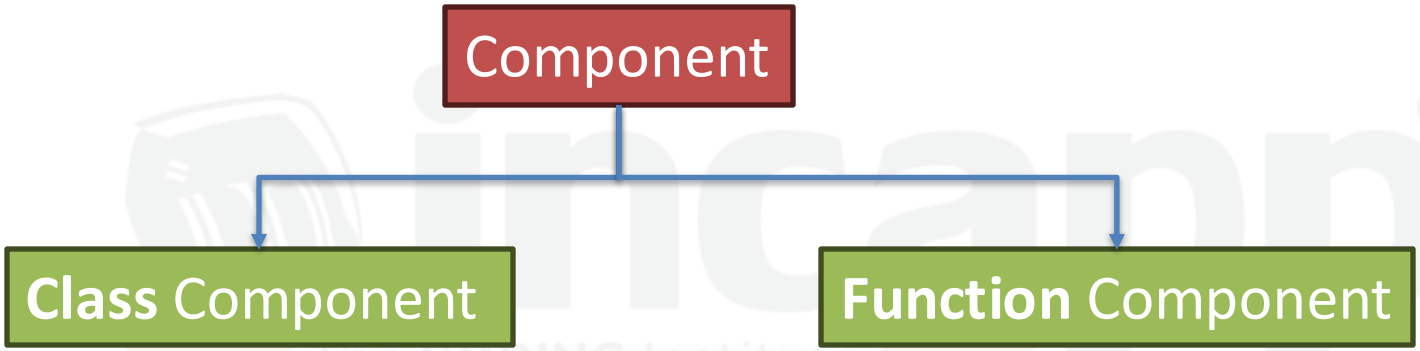
- React does not support if statements *inside* JSX.
- To be able to use conditional statements in JSX, you should put the if statements outside of the JSX, or you could use a ternary expression instead:

```
const a = 2;
let b = "Incapp";
if (a > 0)
{
    b = "India";
}

const ele = <h1>{b}</h1>;
```

React Component

- Component is like function that return HTML content.
- Types of components:



Class Component

- When creating a React component, the component's name *MUST* start with an upper case letter.
- A class component must include the extends ***React.Component*** statement. This statement creates an inheritance to ***React.Component***, and gives your component access to React.Component's functions. The component also requires a ***render()*** method, this method returns HTML.

```
import React from "react";

class Person extends React.Component {
        render() {
                return <h3>Hello Person!</h3>;
        }
}
```

Function Component

- A Function component also returns HTML, and behaves much the same way as a Class component, but Function components can be written using much less code, are easier to understand.

```
function Person() {
        return <h3>Hello Person!</h3>;
}
```

Rendering Component

- Display the **Person** component in the "root" element:

```
const root = createRoot(document.getElementById('root'));
root.render(<Person />);
```

Component inside Component

```
function Person() {
        return <h2>Hello Person!</h2>;
}
function World() {
        return (
        <>
                <h1>Hello World</h1>
                <Person />
                <Person />
        </> );
}
```

```
const root = createRoot(document.getElementById('root'));
root.render(<World />);
```

StrictMode & createRoot

Code	Purpose
StrictMode	Helps detect potential problems in development
createRoot	Enables new features like concurrent rendering in React 18+

Term	Real World Example	Purpose
StrictMode	A supervisor who warns during practice	Helps find bugs and follow best practices
createRoot	Smartboard in your classroom	Gives power to use modern features

Props with function

- Props stands for properties.
- Props are like function arguments, and you send them into the component as attributes.

```
function Person(props) {
            return <h3>Hello Person of Age: {props.age}!</h3>;
}
```

```
const root = createRoot(document.getElementById('root'));
root.render(<Person age="23"/>);
```

Note: props can be changed to any name.

Props with function

```
function Person(props) {
        return <h3>
        Hello Person {props.age}    {props.text.name}
        </h3>;
}
```

```
const root = createRoot(document.getElementById('root'));
root.render(<Person age="23" text={{name: 'ram'}} />);
```

Props with class

```
import React from "react";

class Person extends React.Component {
        render() {
        return <h3>
        Hello Person {this.props.age}    {this.props.text.name}
        </h3>;
        }
}
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Person age="23" text={{name: 'ram'}} />);
```

Class vs Function Component

Class Components	Function Components
Class components need to extend the component from “React.Component” and create a render function that returns the required element.	Functional components are like normal functions which take “props” as the argument and return the required element.
They are also known as stateful components.	They are also known as stateless components.
They implement logic and the state of the component.	They accept some kind of data and display it in the UI.
Lifecycle methods can be used inside them.	Lifecycle methods cannot be used inside them.
Format of Hooks inside class components: `constructor(props) { super(props); this.state = { color: ‘ ‘ } }`	Format of Hooks inside function components: `const [color,SetColor]= React.useState('')`
It needs to store state therefore constructors are used.	Constructors are not used in it.
It has to have a “render()” method inside that.	It does not require a render method.