How to Render Multiple Elements inside render ():

if we want to render two or more elements, we have to **wrap them in another element or component**. Commonly, the element used for this is a <div> tag.

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  //  [
  //      <h1>Welcome to React App</h1>,
  //      <h2>This is 1st Way</h2>
  //  ]

  // <div>
  //      <h1>Welcome to React App</h1>
  //      <h2>This is 2nd Way</h2>
  // </div>

  // <React.Fragment>
  //      <h1>Welcome to React App</h1>
  //      <h2>This is 3rd Way</h2>
  // </React.Fragment>

  //JSX Fragment OR Shortex Syntax <></>
  <>
      <h1>Welcome to React App</h1>
      <h2>This is 4th Way</h2>
  </>
);
```

JSX:

- ✓ JSX stands for JavaScript extension or JavaScript XML
- ✓ JSX is used in React to easily write HTML and JavaScript together.
- ✓ JSX follows XML rules, and therefore HTML elements must be properly closed.
- ✓ babel JavaScript compiler is responsible to convert JSX into compatible version of JavaScript. (https://babeljs.io/)
- ✓ we can use Plain JavaScript instead of JSX but it becomes too complicated and lengthy.

e.g.

```
const myelement =<h1>This is JSX</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  myelement
);
```

Multiline JSX Expression:

```
const myList = (
  <ul>
     <li>item 1</li>    <li>item 2</li>    <li>item 3</li>
  </ul>
);
```

```
const root =
ReactDOM.createRoot(document.getElementById('root'));
root.render(
    myList
);
```

JSX Expression with Variables & Object inside HTML Element:

```
const name='Umesh'
const mobile=9876543210
const emp={ name:"Rakesh", mobile:9987654321 }

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <>
      <h1>Name : {name} & Mobile : {mobile} </h1>
      <h1>Employee Name: {emp.name} & Contact No. :{emp.mobile}</h1>
  </>
);
```

JSX with Map():

```
const emplist=[
  { name:"STV", age:30},  { name:"ABC", age:40},
  { name:"PQR", age:50},  { name:"XYZ", age:60}
];

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  emplist.map(e=>{
    return (      //<h1>Employee Name : {e.name},  Age : {e.age}</h1>
      <>
          <h1>Employee Name : {e.name}</h1>
          <h1>Employee Age : {e.age}</h1>
      </>
    )
  })
);
```

How to apply to style our Elements by using Inline Styling & CSS Stylesheet:

✓ In case of Inline styling, style attribute takes css properties in Object form {}.

```
import ReactDOM from 'react-dom/client';
import './style.css';

var eleStyle={color:"brown",fontStyle:"italic",textShadow:'3px 3px 3px orange'}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <>
      <h1 style={{color:'Navy',textShadow:'3px 3px 3px cyan'}}>
          CSS Styling with 1st Way
      </h1>
      <h1 style={eleStyle}>
          CSS Styling with 2nd Way
      </h1>
      <h1 className='headline'>CSS Stylesheet or External CSS</h1>
  </>
);
```

Style.css

```
.headline {
    color:darkgreen;
    text-shadow:3px 3px 3px greenyellow;
}
```

Bootstrap:

1. How install bootstrap:- **npm install bootstrap**
2. Import bootstrap.min.css file in index.js :

   import '../node_modules/bootstrap/dist/css/bootstrap.min.css'

React Components:

✓ A Component is the core building blocks of a React application. They can be reusable as per your need.

✓ we have mainly two types of components:

- Functional Components
- Class Components.

✓ **Functional Components:**

→ They are simply JavaScript functions that may or may not receive data as parameters.

→ We can create a functional component in React by writing a JavaScript Function.

→ In the functional components, the return value is the JSX Code to render to the DOM tree.

e.g.FunctionalComponent.js

```
function Headline(){
 return <h1 className="text-warning">This is Headline Functional Component</h1>
}
export default Headline;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import '../node_modules/bootstrap/dist/css/bootstrap.min.css';
import Headline from './MyComponents/FunctionalComponent';

const root = ReactDOM.createRoot(document.getElementById('root'));
```

```
root.render(<>
        <div className='container bg-light'>
            <h1 className='text-primary'>Welcome to React App</h1>
            <Headline/>
        </div>
 </>);
```

✓ **Class Components:**

→ You can create a class by defining a class that extends the Component and has a render function.

→ Class components must have a render() method. This method should return some React elements created with JSX.

→ The Class component is also known as a Stateful component because they can hold or manage local state.

e.g.ClassComponent.js

```
import { Component } from "react";

class Hello extends Component{
    render(){
        return (<>
           <h1 className="text-primary">This is Hello Class Component</h1>
           <button className="btn btn-success">Class</button>
        </>)
    }
}

export default Hello;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import '../node_modules/bootstrap/dist/css/bootstrap.min.css';
import Hello from './MyComponents/ClassComponent';
import Headline from './MyComponents/FunctionalComponent';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<>
        <div className='container bg-light'>
            <h1 className='text-primary'>Welcome to React App</h1>
            <Headline/>
            <Hello/>
        </div>
 </>);
```

Types of export:

1. Default export   2. Named export

1. export default: if file has only one object or component then, we can **import** that object or component with same name or any other name.

2. Named export: if file has multiple object or component then, we should have to use Named exports in curly braces {} and we can import that object or component with same name in curly braces {}.

State:

- ✓ The state is a built-in React object that is used to contain data or information about the component. State could only be used in class components.
- ✓ To define a state, add a class constructor which assigns an initial state using **this.state**.

```
import React, { Component } from 'react'

export class Student extends Component {

  //let city="Pune"; //Unexpected token. A constructor, method, property was
expected.

  constructor(){
    super();           //'this' is not allowed before 'super()'
    this.state={
           name:"Akash",
           course:"Java"
    }
  }

  render() {
    return (<>
        <h1>Name : {this.state.name} and Selected Course :
{this.state.course}</h1>
    </>)
  }
}
```