React.createElement(tags, [props], [children]) :-

Tags :- a html tag like h1, div etc,
Props :- like class, id of that tag (can be an object or null, where null is same as an empty obj)
Children :- the content of the tag, like what we use in .appendChild()
It is a top-level API of ReactJS which is used to create an element like the document.createElement() in DOM.

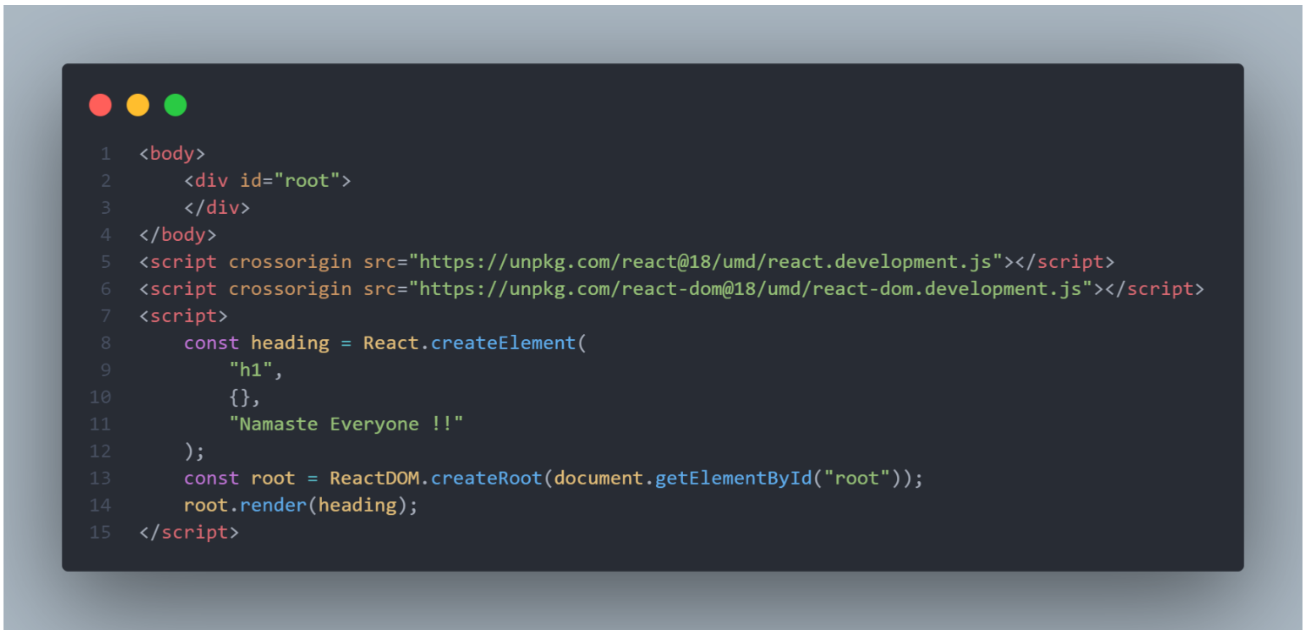
Now, in React, we have to create a root such that we can tell the JS Engine that we want to run our React application in that place. To do this, ReactDOM contains APIs to render React components on the client(on the browser).

ReactDOM.createRoot(document.getElementById("root")) :-

This API is a Client-side API that lets you create a root to display React components inside a browser DOM mode. It returns an object with 2 functions/methods :- render and unmount

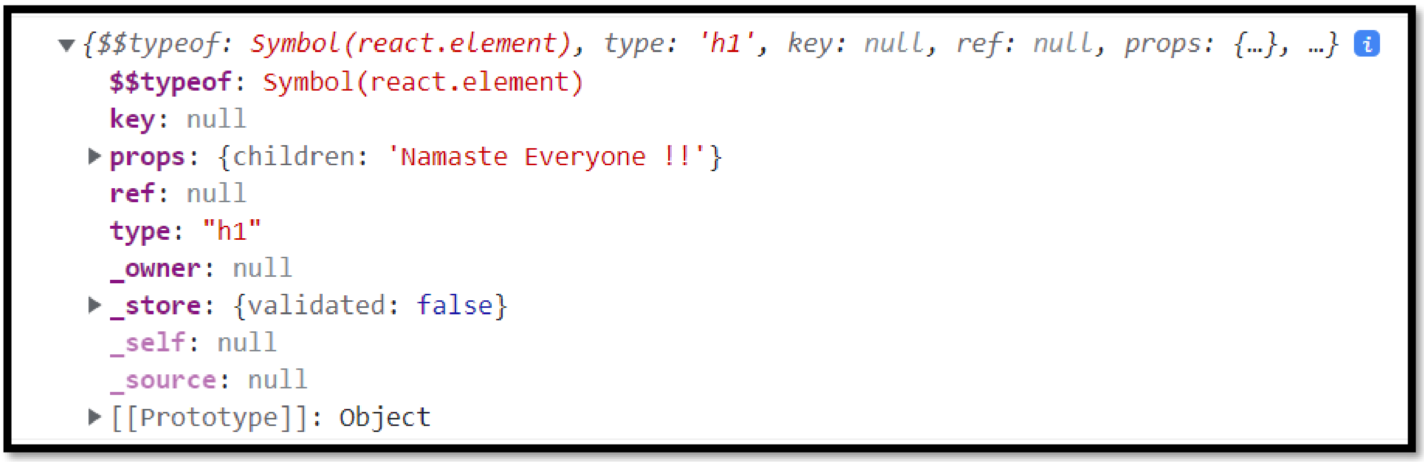
root.render(reactNode) :-

helps to display a piece of JSX("React node") or a React Element(created by react.createElement()) into the React root's browser DOM node.
It returns undefined.



```
<body>
    <div id="root">
    </div>
</body>
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
<script>
    const heading = React.createElement(
        "h1",
        {},
        "Namaste Everyone !!"
    );
    const root = ReactDOM.createRoot(document.getElementById("root"));
    root.render(heading);
</script>
```

If you do, console.log(heading), it will return an object :-

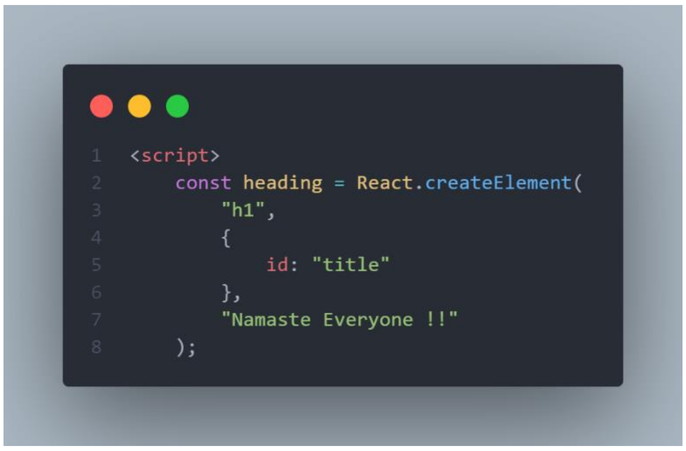


```
▼{$$typeof: Symbol(react.element), type: 'h1', key: null, ref: null, props: {…}, …}
   $$typeof: Symbol(react.element)
   key: null
  ▶props: {children: 'Namaste Everyone !!'}
   ref: null
   type: "h1"
   _owner: null
  ▶_store: {validated: false}
   _self: null
   _source: null
  ▶[[Prototype]]: Object
```

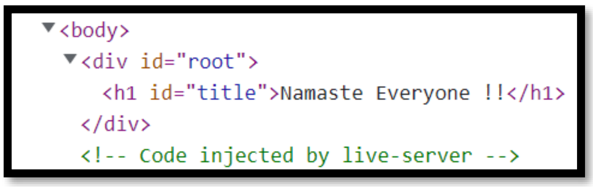
#NOTE:-
We can inject React into any existing projects too, without affecting the other areas. Like if we already have 2 div-s above and below the root div, the React will only affect the root div and the others will be displayed as usual.

Example 1:-

If you want to insert an element with id, then we do something like this :-



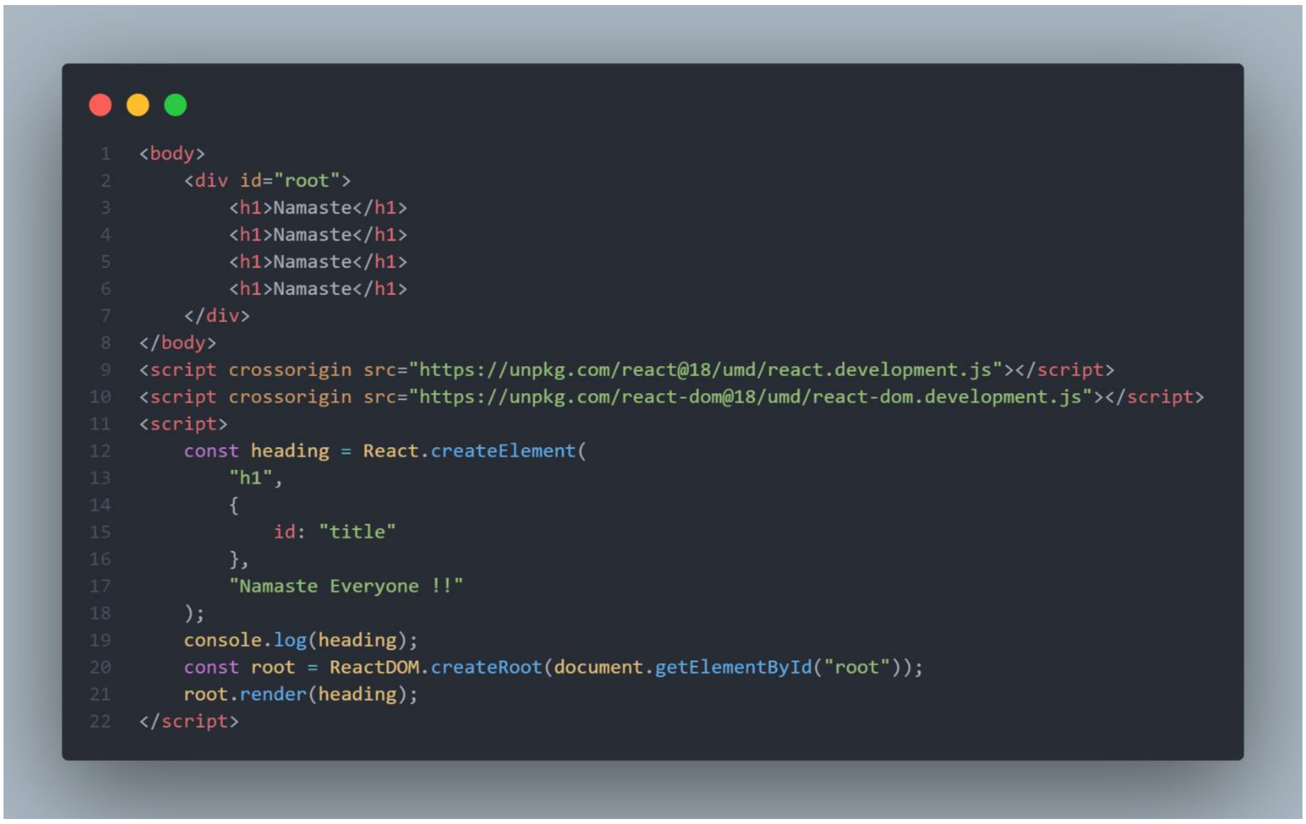
```
<script>
    const heading = React.createElement(
        "h1",
        {
            id: "title"
        },
        "Namaste Everyone !!"
    );
```



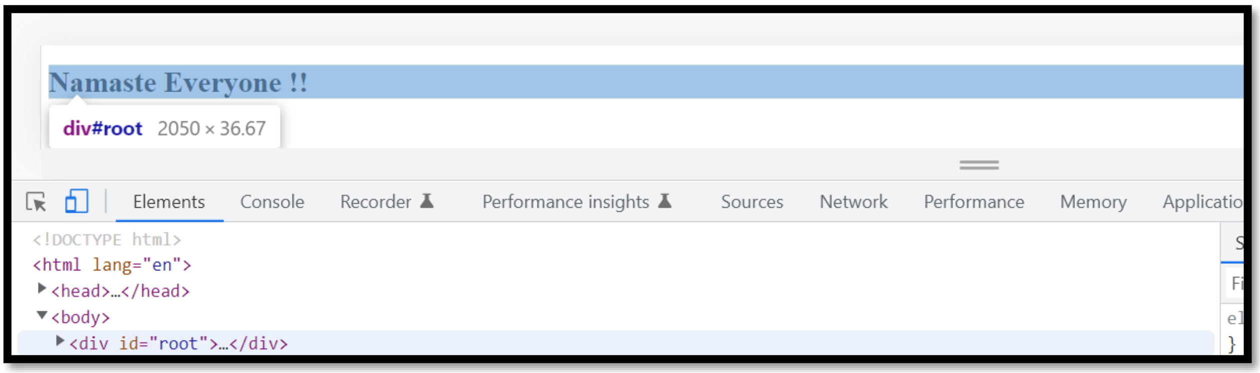
```
▼<body>
  ▼<div id="root">
     <h1 id="title">Namaste Everyone !!</h1>
   </div>
   <!-- Code injected by live-server -->
```

Example 2 :-

If there are already some children present in the root element of React, the React will override it :-



```
<body>
    <div id="root">
        <h1>Namaste</h1>
        <h1>Namaste</h1>
        <h1>Namaste</h1>
        <h1>Namaste</h1>
    </div>
</body>
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
<script>
    const heading = React.createElement(
        "h1",
        {
            id: "title"
        },
        "Namaste Everyone !!"
    );
    console.log(heading);
    const root = ReactDOM.createRoot(document.getElementById("root"));
    root.render(heading);
</script>
```



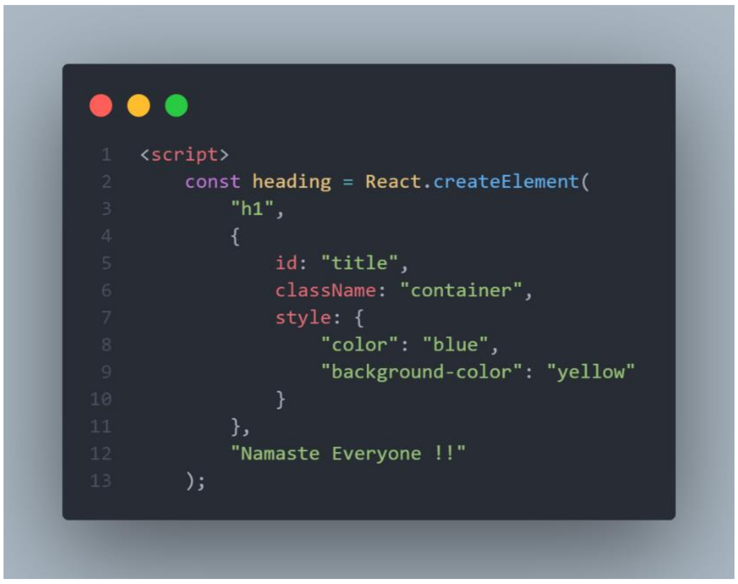
Namaste Everyone !!

div#root 2050 × 36.67

```
<!DOCTYPE html>
<html lang="en">
▶<head>…</head>
▼<body>
  ▶<div id="root">…</div>
```

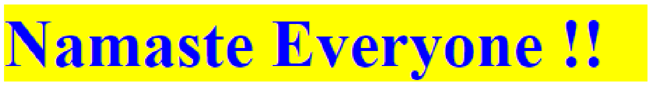
Example 3 :-

React tries to use HTML through JS with the help of JSX and in JSX syntax, class is called className and there are a few other changes.



```
<script>
    const heading = React.createElement(
        "h1",
        {
            id: "title",
            className: "container",
            style: {
                "color": "blue",
                "background-color": "yellow"
            }
        },
        "Namaste Everyone !!"
    );
```

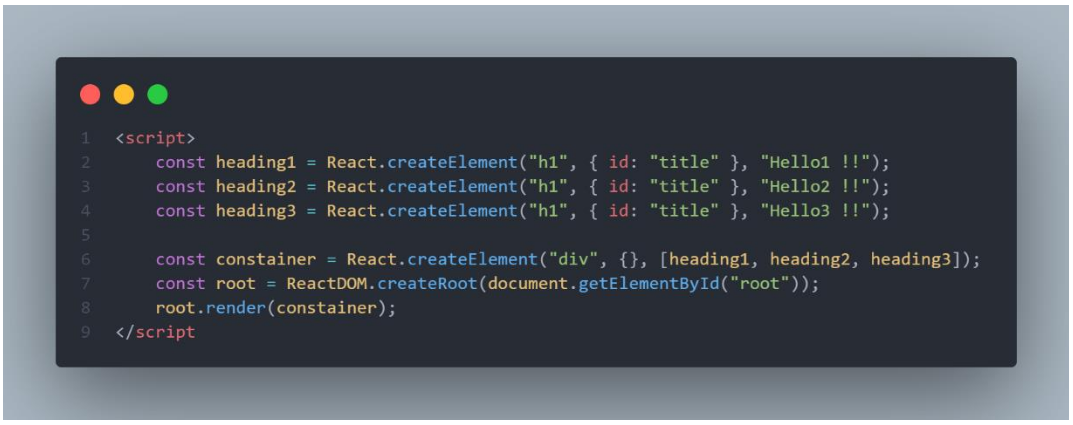
Output :-



Namaste Everyone !!

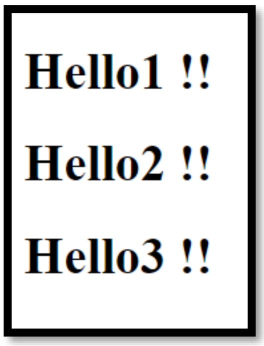
Example-4 :-

We can also insert multiple children elements inside the root element by creating an element which contains all those children elements in an array and rendering that element inside the React root like :-



```
<script>
    const heading1 = React.createElement("h1", { id: "title" }, "Hello1 !!");
    const heading2 = React.createElement("h1", { id: "title" }, "Hello2 !!");
    const heading3 = React.createElement("h1", { id: "title" }, "Hello3 !!");

    const constainer = React.createElement("div", {}, [heading1, heading2, heading3]);
    const root = ReactDOM.createRoot(document.getElementById("root"));
    root.render(constainer);
</script
```



Hello1 !!

Hello2 !!

Hello3 !!

Important points :-

- If we write the React script before the 2 links, then we will get an error in the console "React not defined" and the react elements will not be rendered properly
- Developers write :-

    <div id="root">Not Rendered Yet</div>

  Because, if something happens due to which react is not rendered properly in the root element, that will be easily detected.
- The heading1,2 and 3 in example-4, are all react elements and we know that they are nothing but objects and render function is used to overwrite the contents of root HTML element with those elements.