React Element

- As React is Javascript Library, so we can use it directly using it's cdn link. Also at the end React is simply JS so we can caopy-paste the js code into **React.js** and **ReactDOM.js** file to see and visualise the methods more clearly.

```
<!-- Import "React" and "React-DOM" from their CDN link -->
<!-- <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> -->
<!-- Instead of using above cdn links,
we can directly open these links and copy-paste to React.js and ReactDOM.js and then use it.-->
<script src="React.js" defer></script>
<script src="ReactDOM.js" defer></script>
```

- **React.createElement** is simply just a function with three input parameters as shown.

```
React.createElement
ƒ createElementWithValidation(type, props, children) {
    var validType = isValidElementType(type); // We warn in this case but don't
throw. We expect the element creation to
    // succeed and ther…
```

Parameters

- `type`: The `type` argument must be a valid React component type. For example, it could be a tag name string (such as `'div'` or `'span'`), or a React component (a function, a class, or a special component like `Fragment`).
- `props`: The `props` argument must either be an object or `null`. If you pass `null`, it will be treated the same as an empty object. React will create an element with props matching the `props` you have passed. Note that `ref` and `key` from your `props` object are special and will *not* be available as `element.props.ref` and `element.props.key` on the returned `element`. They will be available as `element.ref` and `element.key`.
- **optional** `...children`: Zero or more child nodes. They can be any React nodes, including React elements, strings, numbers, portals, empty nodes (`null`, `undefined`, `true`, and `false`), and arrays of React nodes.

```
const h2 = React.createElement(
  "h2",
  { className: "sub-heading" },
  "Hello React"
);
```

- **React** will not rendor this h2 element on our web-page, to do this we need to use **ReactDOM** library which is a part of reactjs library. *We need to create a new element every time when we want to update something in that particular element.*
  First create root using ReactDOM library, this will return the element that you have passed as an argument in it's constructor function.

```
const root = ReactDOM.createRoot(document.querySelector("#root"));
```

Now root object is created so we will use render method of react-dom to render our component on web-page.

```
root.render(h2);
```
// After using this we can see our object on the page, & it's working fine as well.

- We can create children elements also using react.creatElement method by simply passing all children in array. As shown here :

```
const container = React.createElement("div", { className: "container" }, [
  h2,
  React.createElement("section", {}, [
    React.createElement(
      "p",
      {},
      "The library for web and native user interfaces"
    ),
  ]),
]);
```

- But Creating very big structures make it more complex (while we can simply do this in html with less code), as shown here:

```
const container = React.createElement(
  "div",
  { className: "container", id: "container" },
  [
    React.createElement("section", { key: 1 }, [
      React.createElement(
        "p",
        { key: 1 },
        "The library for web and native user interfaces"
      ),
      React.createElement("img", {
        key: 2,
        style: {
          width: 200,
          backgroundColor: "teal",
          borderRadius: 8,
          padding: 16,
        },
        src: "https://upload.wikimedia.org/wikipedia/commons/a/a7/React-icon.svg",
      }),
    ]),
    React.createElement("section", { key: 2 }, [
      React.createElement("form", { key: 1 }, [
        React.createElement("div", { className: "input-group", key: 1 }, [
          React.createElement(
            "label",
            { key: 1, htmlFor: "username" },
            "Username"
          ),
          React.createElement("input", { key: 2, id: "username" }),
        ]),
        React.createElement("div", { className: "input-group", key: 2 }, [
          React.createElement(
            "label",
            { key: 1, htmlFor: "password" },
            "Password"
          ),
          React.createElement("input", {
            key: 2,
            id: "password",
            type: "password",
          }),
        ]),
      ]),
    ]),
  ]
);
```

```
<div id="root">
  <div class="container" id="container">
    <section>
      <p>The library for web and native user interfaces</p>
      <img src="https://upload.wikimedia.org/wikipedia/commons
      adius: 8px; padding: 16px;">
    </section>
    <section>
      <form>
        <div class="input-group">
          <label for="username">Username</label>
          <input id="username" fdprocessedid="4abk6k">
        </div>
        <div class="input-group">
          <label for="password">Password</label>
          <input id="password" type="password" fdprocessedid="
        </div>
      </form>
    </section>
  </div>
</div>
<div id="app-container"></div>
```

- Therefore to solve this problem we use ***JSX and Babel.***