



JSX

To make react work, the two libraries is needed which is

- React - Responsible for creating components / views
- React-DOM - Responsible for rendering.

2

We will see with below example.

```
JS
1 class Message extends React.Component{
2   render()
3   {
4     return(
5       React.createElement('div',null,
6         React.createElement('h1',null,"React"),
7         React.createElement('h1',null,"React DOM"),
8       ) )
9   }
10 }
11 ReactDOM.render(React.createElement(Message),
12   document.getElementById('reactcontainer'));
```

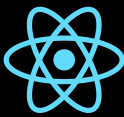
JS CODE

```
HTML
1 <div>
2   <h4>Hello</h4>
3 </div>
4 <div id = 'reactcontainer'>
5
6 </div>
```

HTML

- I imported two library react and react dom in codepen via CDN.
- In reactcontainer, we inserted two h1 tag grouped under div like this.

```
<div>
  <h1>React</h1>
  <h1>React Dom</h1>
</div>
```



- The **react.createElement** method construct the above structure.
- The **ReactDOM** is taking responsibility of rendering under destination.



- The `react.createElement` will look simple for this example. But imagine it for large projects. Things become difficult there. 😱
- Think like instead of coding like this tree structure, what if we do like our HTML code (Like Below). The thing becomes easier now. 🤔
- It is what is called **JSX**

```
return
(  
  <div>  
    <h1>React</h1>  
    <h1>React Dom</h1>  
  </div>  
)
```

- It looks like HTML, but not. It is a JSX.
- JSX will not be understandable by browser.
- In the end, it will be compiled to normal React JS code that we write before.

Who is behind it?

BABEL

BABEL.

- It is a JavaScript compiler that transforms advanced JS code into browser-understandable format.
- Babel is not only for JSX. Used for many other things like ECMA2016 JS code into older format.
- It makes life easier.



Here you can see what babel can do. It translate our JSX into JS code.

```
class Message extends React.Component {  
  render()  
  {  
    return(  
      <div>  
        <h1>React</h1>  
        <h1>React Dom </h1>  
      </div>  
    )  
  }  
}
```

```
1 "use strict";  
2  
3 class Message extends React.Component {  
4   render() {  
5     return /*#__PURE__*/React.createElement("div", null,  
        /*#__PURE__*/React.createElement("h1", null, "React"),  
        /*#__PURE__*/React.createElement("h1", null, "React Dom "));  
6   }  
7 }
```

BABEL SANDBOX

- As earlier said JSX is not supported by browser. It required some compiler translate into browser understandable format.
- Here in code pen without babel you can see the code is giving error.

```
JS  
1 class Message extends React.Component{  
2   render()  
3   {  
4     return(  
5       <div>  
6         <h1>React</h1>  
7         <h1>React Dom </h1>  
8       </div>  
9     )  
10  }  
11 }  
12 ReactDOM.render(<Message/>,document.getElementById('reactcontainer'));
```

Uncaught SyntaxError: Unexpected token '<'

WITHOUT BABEL

AFTER BABEL INCLUDE

Pen Settings

- HTML
- CSS
- JS
- Pen Details
- Privacy PRO
- Behavior
- Editor
- Template
- Screenshot PRO

JavaScript Preprocessor

Babel

Babel includes JSX processing.

Add External Scripts/Pens

Any URL's added here will be added as <script>s in order, and run before the JavaScript in the editor. You can use the URL of any other Pen and it will include the JavaScript from that Pen.

Search CDNjs (jQuery, Lodash, React, Angular, Vue.js, Ember..)

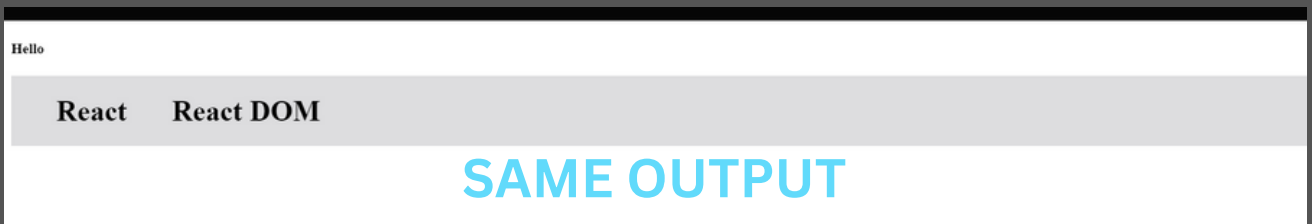
Powered by algolia

https://cdnjs.cloudflare.com/ajax/libs/react/18.2.0/umd/react.prod...

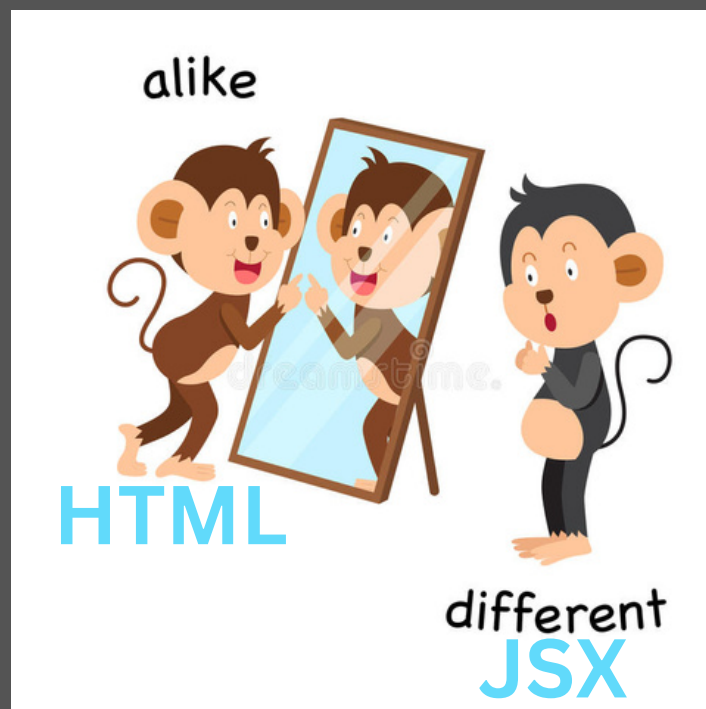
https://cdnjs.cloudflare.com/ajax/libs/react-dom/18.2.0/umd/react-...

Upgrade to PRO and unlock Privacy, Custom Thumbnails, Asset Hosting, and more.

Save & Close



- As earlier said JSX is not HTML. It look like HTML but not.
- Simple Example. We will use class in HTML.
- But in JSX if we use this it will throw error. Because class is a reserved keyword in JS.
- We have to use className. This is one example, so some other difference also there in JSX.





CSS

```

padding-top: 20px;
padding-bottom: 20px;
padding-left: 50px;
padding-right: 50px;
}
h1 {
display: inline;
padding-left: 50px;
}
.red {
background-color: #red
}

```

JS (Babel)

```

class Message extends React.Component {
  render() {
    return (
      <div class="red">
        <h1>React</h1>
        <h1>React Dom </h1>
      </div>
    )
  }
}
ReactDOM.render(<Message/>, document.getElementById('reactcontainer'));

```

USING CLASS IN JSX.

JSX ask hey what is class?



HTML

```

<div>
  <h4>Hello</h4>
</div>
<div id = 'reactcontainer'>
</div>

```

CSS

```

padding-top: 20px;
padding-bottom: 20px;
padding-left: 50px;
padding-right: 50px;
}
h1 {
display: inline;
padding-left: 50px;
}
.red {
background-color: red
}

```

JS (Babel)

```

class Message extends React.Component {
  render() {
    return (
      <div className = "red">
        <h1>React</h1>
        <h1>React Dom </h1>
      </div>
    )
  }
}
ReactDOM.render(<Message/>, document.getElementById('reactcontainer'));

```

ello

AFTER USING CLASS NAME

React	React Dom

- Anyways the ClassName is translated into class at final.
- In simple words, in one thing it is called class. And on another thing it is called ClassName. It is Like Language difference. But meaning is same.

