

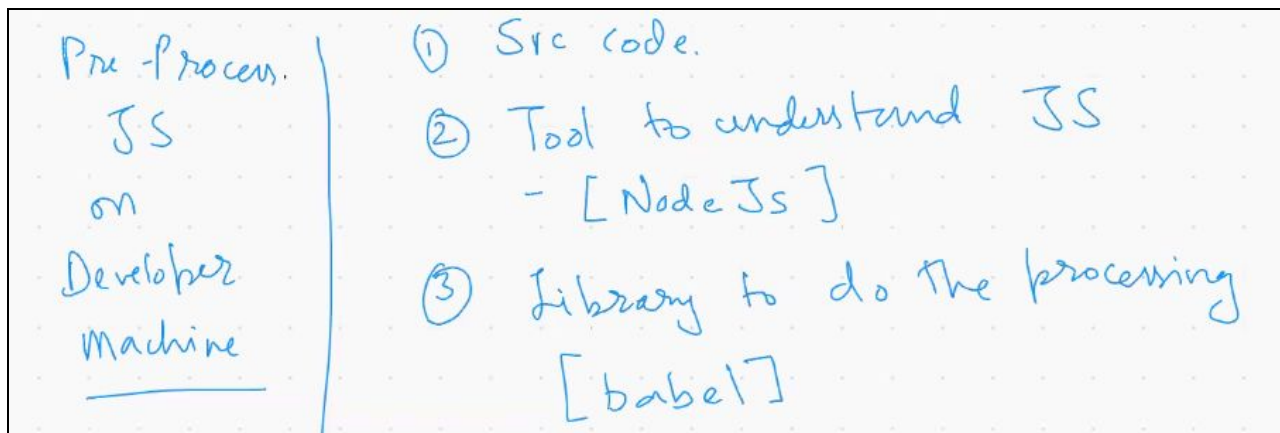
Day:

Title: Tic Tac Toe in React

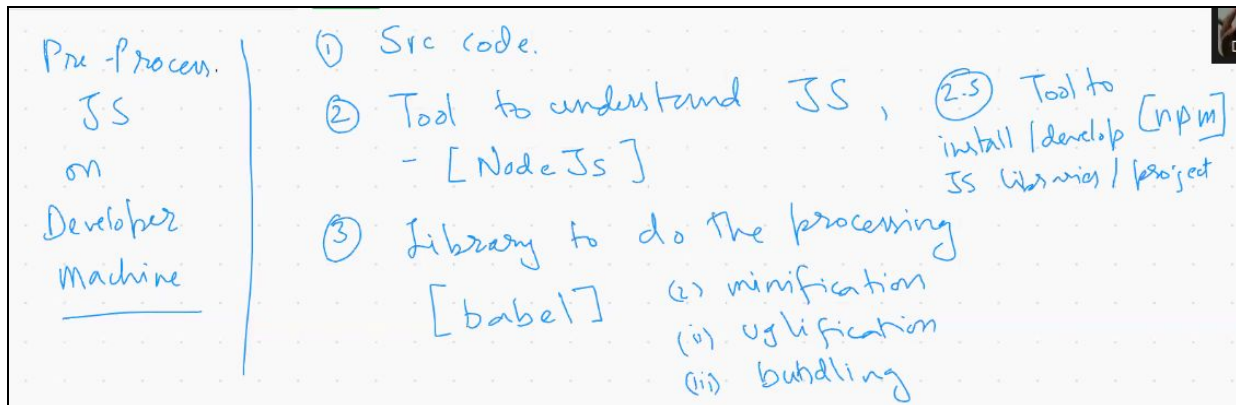
Topics covered:

- Preprocessing in babel
- React
- Virtual DOM
- JSX

For preprocessing which things were required, these three below!.



Then we studied npm which is a tool to install / develop a JS library/project

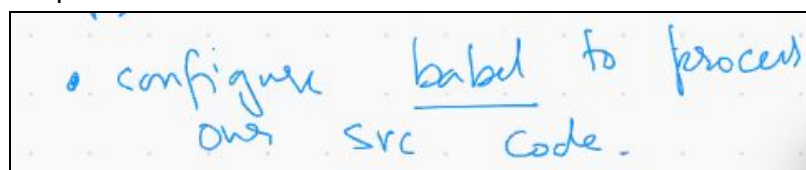


Till now this much was covered.

Babel has plugin for

- Minification plugin
- Uglification plugin
- Bundling plugin

Plugins are also called as presets



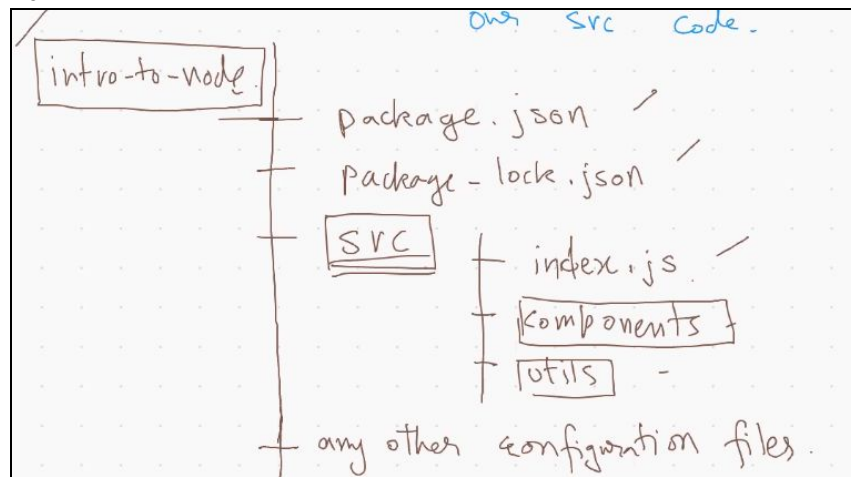
This we will do using create react app

Configuration is standard

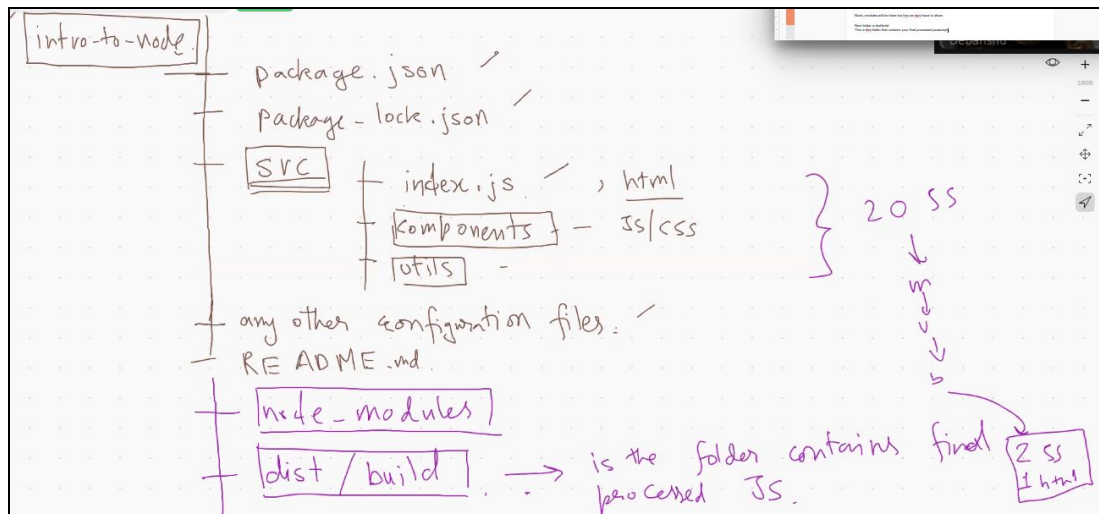
- This is for babel.
- Babel configuration doesn't change that much in the company also.

Now We are talking about creating project and performing preprocessing in babel.
You need to have src directory and in that index.js file should be there.

- This 2 things are compulsory
- Src directory index.js



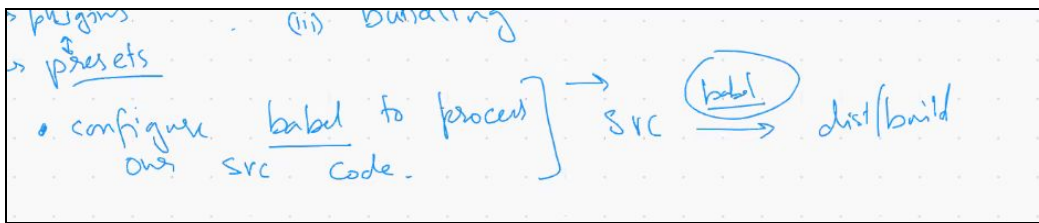
- Configuration files are at outermost level. These files are to be zipped, while distributing your source code
- NEXT FILE IS README.MD -> Informative file
- Node_modules will be there but this we don't have to share
- Next folder is dist/build. This is the folder that contains your final processed javascript.



These are the total things present in our project, a basic structure

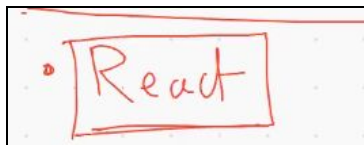
- It's a basic structure that we will follow through.

Even int



Configure babel will convert the src code(original code) to dist/build(pre-processed code)

What is React

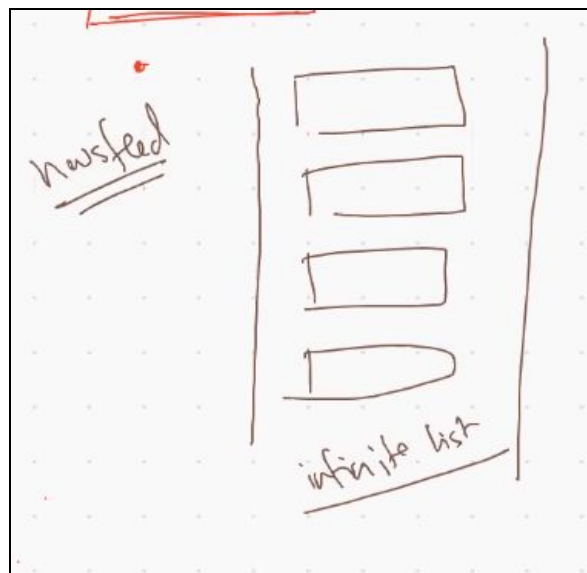


- Its a library.
- Its a UI framework for developing front end application.
- Angular is also used for front end.



- Like bootstrap makes life easier to create layout
- Similarly react makes life easier to create a website
- React is created/developed by facebook.

History: From where react came.

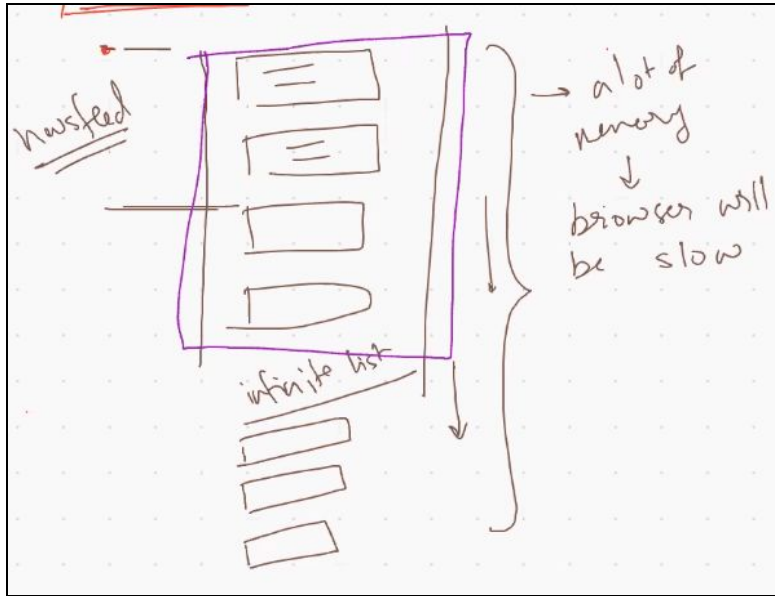


Newsfeed in facebook

- Newsfeed has an Infinite list of updates from your friends on facebook.
- If you keep adding items in the bottom, as this list is very large, it will take so much memory.
- And the browser will lag.

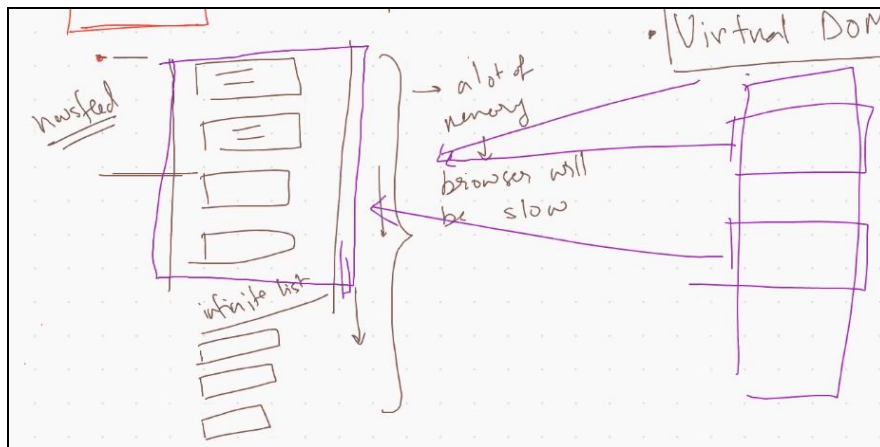
This was the motivation to build react. To optimise the browser lag.

- Fundamental concept was to make a virtual DOM. Developers thought, they will still maintain an infinite list but they won't add them (items/elements) to the DOM.



From this page what part you could see (only the block marked by purple/violet color)

- Only the part of the window that you can see will be added to actual dom
- And remaining things will be present in the virtual dom

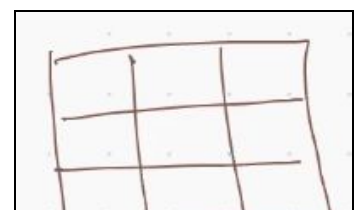


Elements will be added from virtual dom, and only those which are needed not all.

- This was how virtual DOM started off
- Now only the part needed to see are added to dom that's why speed is increased a lot.
- Virtual dom takes less memory than the actual dom. Virtual dom takes browser memory but not much memory as the memory required to add this much elements in the dom.
- If virtually stored json data structure is just a text.
- Virtual dom used took to make much less memory.

Next step (Next Advantage of using DOM)

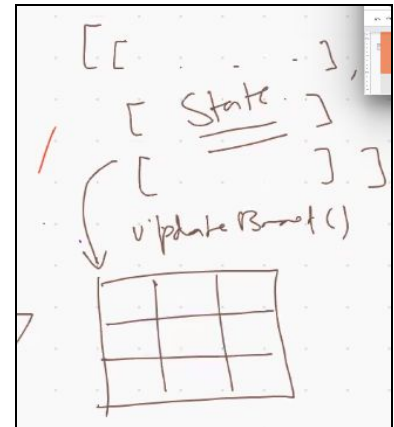
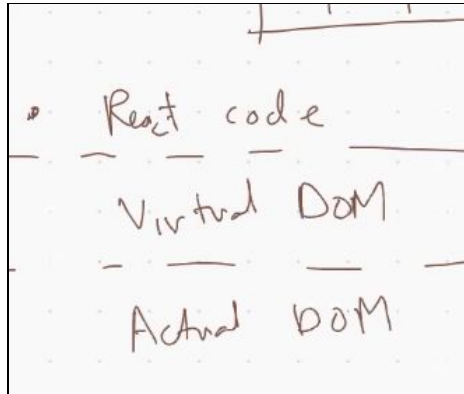
Example



Everytime you re render the entire board in tic tac toe example

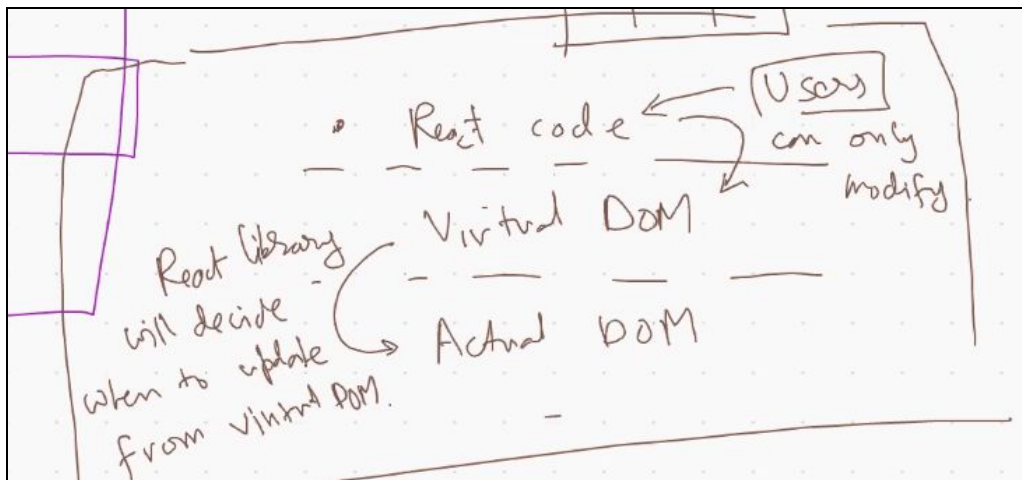
We were keeping the board data on the array. We were rerendering the data
When you detected the update we updated the DOM Element
So using react you actually don't update the code in dom

React made separation

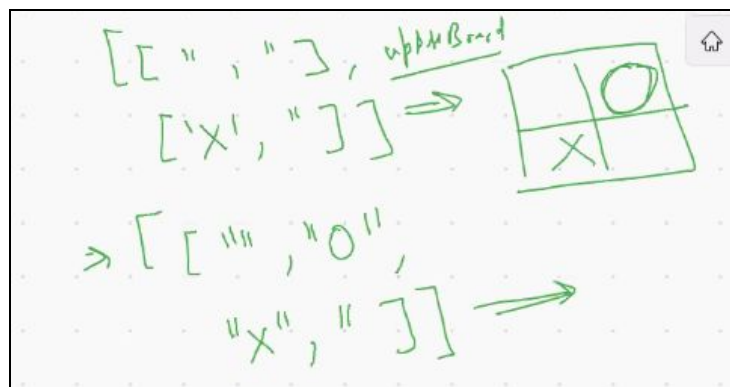


It's this way

- User creates react code
- Eact code will only modify the virtual dom not the actual dom.
- Only the react library will decide when to update the actual dom from the virtual dom.



An example

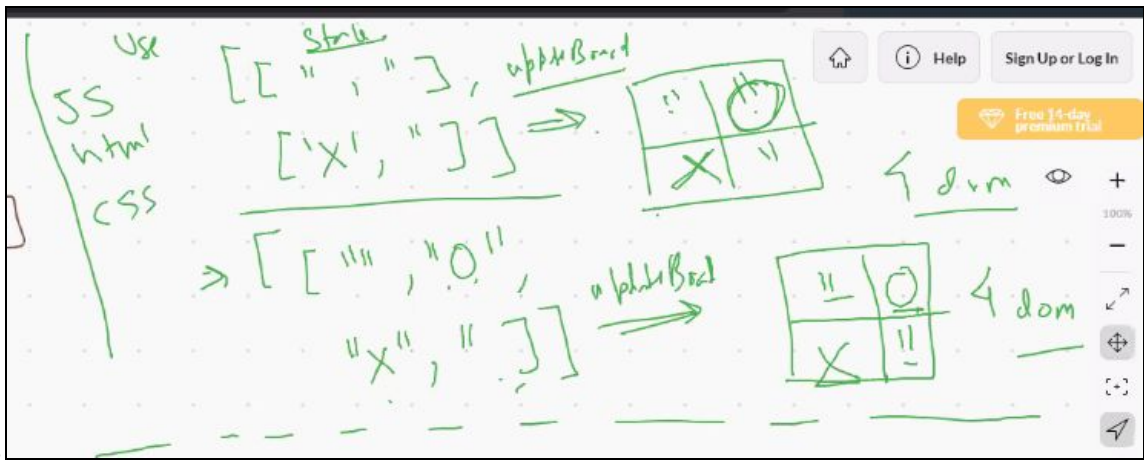


When user click

We will update this (x/o) only in array not in dom, this is the analogy for react.

Normally Everytime you change the state you are changing in actual DOM, so 4 changes in array and then 4

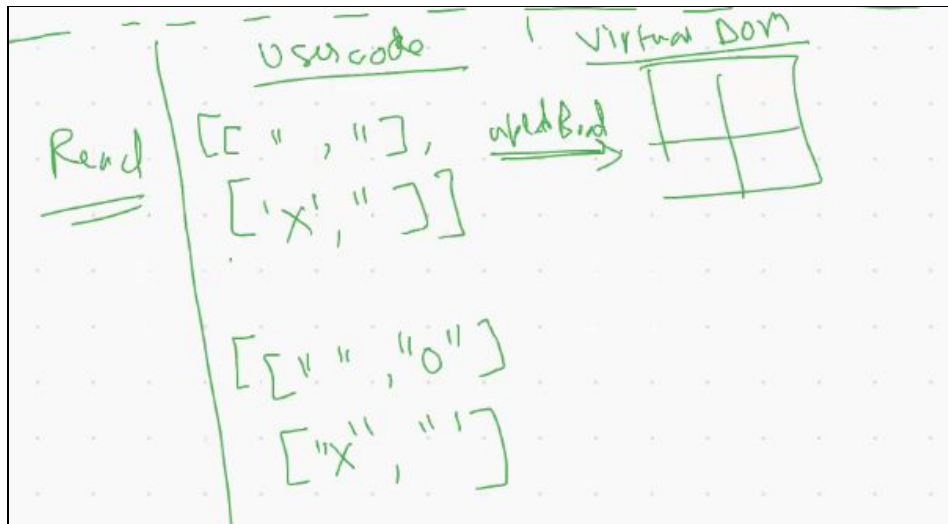
updates



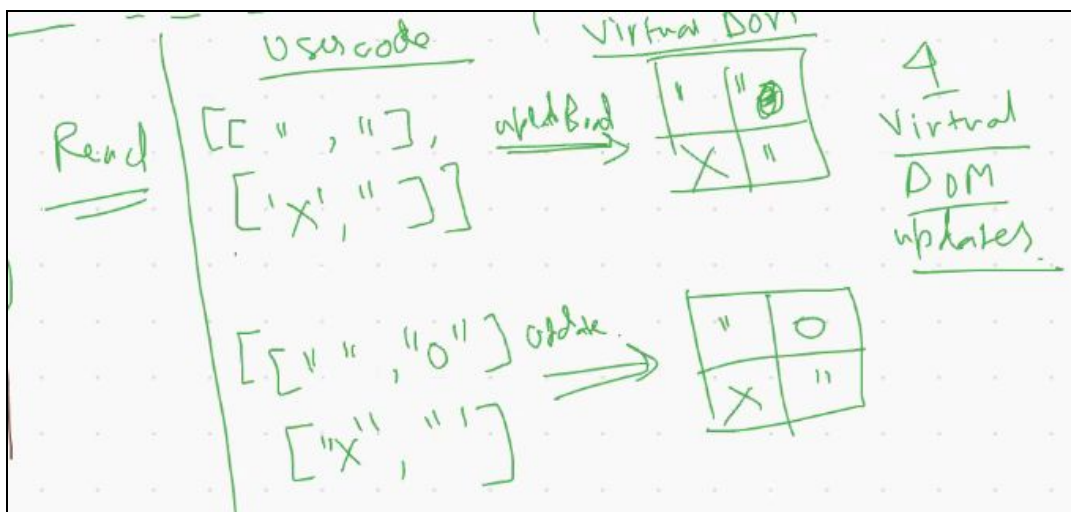
But in react

React code we are maintaining the same state and writing the exact same code

Main change that would happen react code is modifying the virtual dom



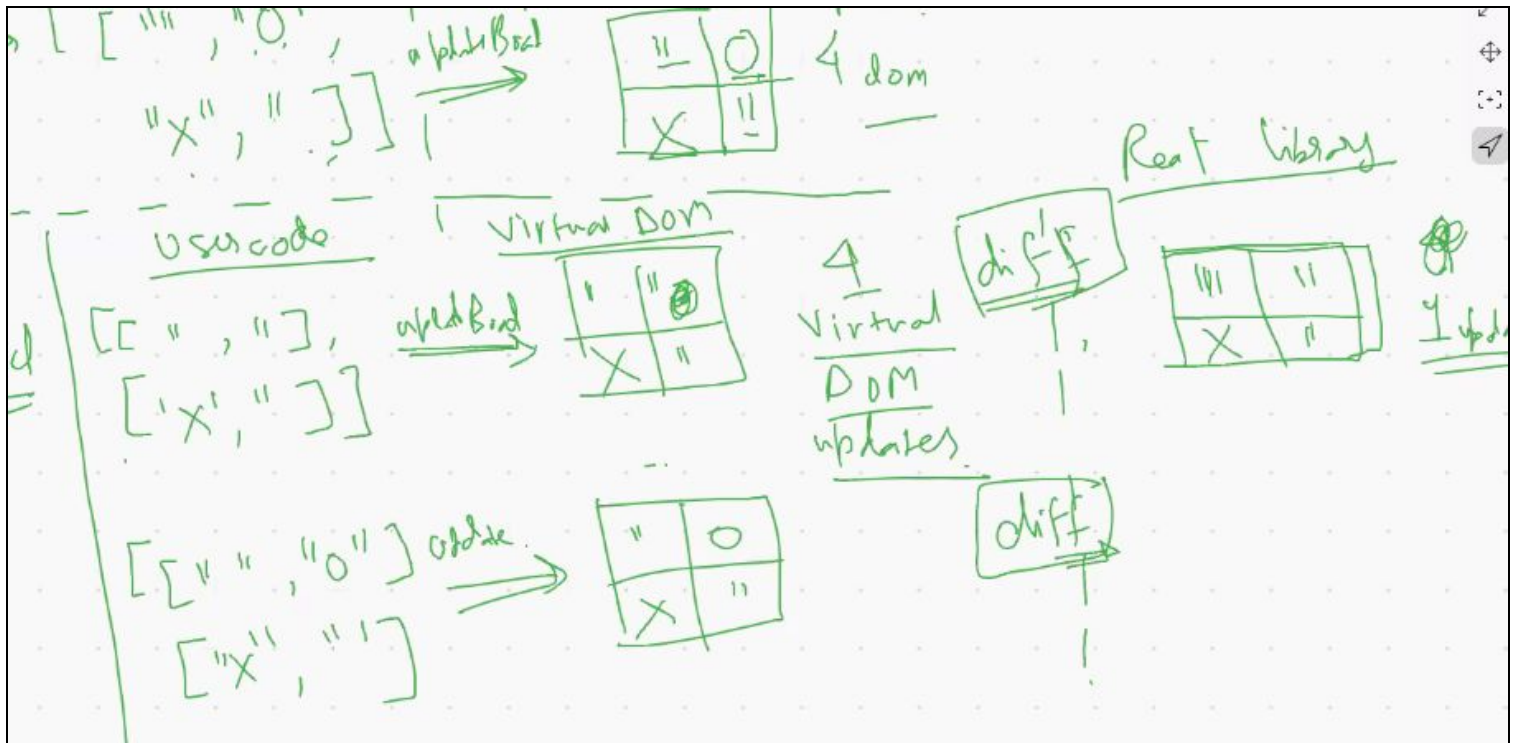
now updates will happen in virtual dom



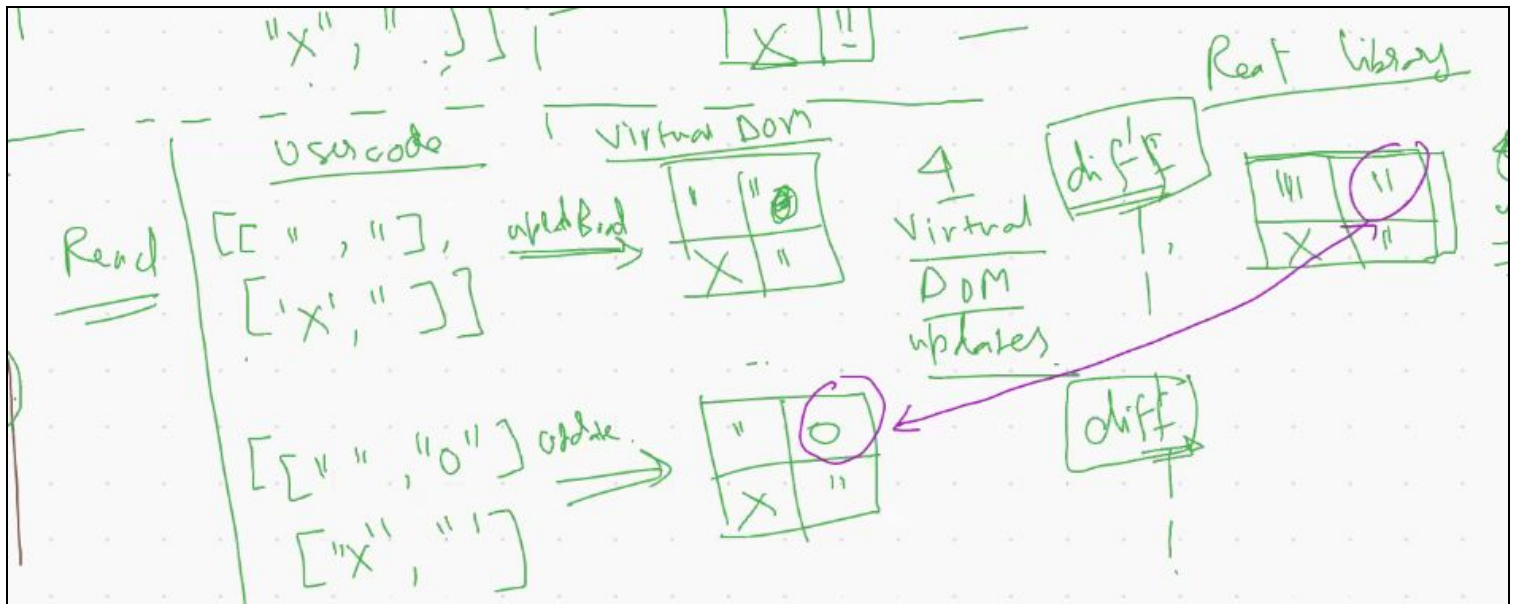
after this react library will decide when it want to update on the actual dom

Find the difference present in the actual dom and virtual dom and will only update the difference in them.

It will update the actual dom and do it in single update



Only change is



Normally(without react) 4 updates in the dom, But here only 1or2 updates

This is the major optimization that react library started to do

Other optimatoin

When can a change is perceived by the user

An example

You changed 100 things in a sec. Can our eyes understand 100 frames per second. We can't understand 100Hz. Human eyes only understand 60 frames per second. React can understand And can make sure and update the dom only 60 times in one sec

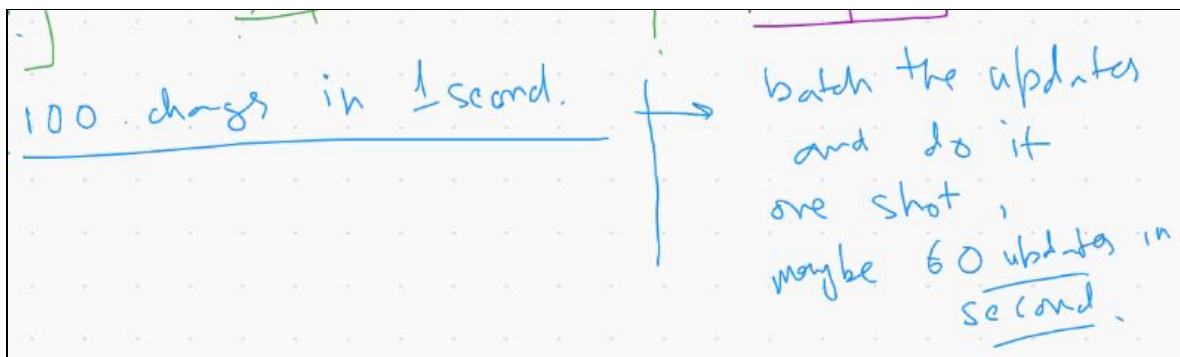
Even if 100 changes are done.

Because the user can't perceive 100 changes in the dom,

React will do it in less changes

How many updates(changes) per sec?

Completely depends on the device. If 12hz device it cannot perform 60 updates in a sec. It can only do 12 in a sec.



No need to concern about how react is doing this thing

Just know how we can do it i mean how we can use react to optimise our website

React will make sure that it will only update the change when needed

This is the biggest advantage of using react. Because of its virtual dom.

What is react native

Version of mobile

React native converts down to to the native languages of the mobile platform

Whats imp is how to create a virtual dom how can i update virtual dom

How can i update interactivity in the virtual dom

And the rest things will be taken care by react

How to create element using react in virtual dom?

When we modified the actual dom

First thing we learnt was `document.createElement`. This allowed us to create actual dom elements

how to create element in the virtual dom

- create Element \Rightarrow for Virtual DOM

React has a special syntax for this

The syntax of react is in below screenshot, we will see the shortcut syntax afterwards.

This is the original one but we will always use shortcut

- React, create Element (' ' ' ' ' ' ' ')
 - ↓
 - ↓
 - ↓
 tagName attribute inner Dom.

If you want to create a virtual dom

An example

we want to create this in virtual dom

```
<div class="abc" id="d1">  
Some Text  
</div>
```

How to do this using react syntax

Ans is

```
React.createElement('div', { class: "abc", id: "d1", "Some Text" });
```

how to create more complicated element?

How to append child?

```

<div>
  <ul class="mylist">
    <li> item 1 </li>
    <li> item 2 </li>
  </ul>
</div>

```

Tryout this using `React.createElement`

`React.createElement('div', '', 'React.createElement('ul', '{class : "mylist"}', react.createelement('li', '', item1)))`

This is the way

But we have shorthand syntax

Innerdom elements can take multiple parameters

Its like rest element(rest operator)

`React.createElement`

↓

tagName

↓

attribute object

↓

inner Dom

First writing in separate lines

```

const li1 = R.cE('li', null, 'item 1')
const li2 = R.cE('li', null, 'item 2')
const ul = R.cE('ul', {class: "mylist", li1, li2})
const div = R.cE('div', null, ul)

```

Writing in single call

```

R.cE('div', null, R.cE('ul', {class: "..."},
  R.cE('li', "..."),
  R.cE('li', "...")
))

```

If content is constant do in this way. "Some text" is a const.

object

```

const div = R.cE('div', {
  class: "abc",
  id: "d1",
  "Some Text"
})
const divValue = sm

```

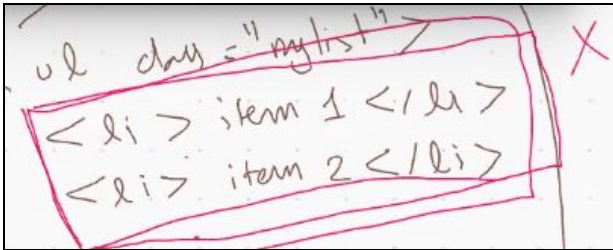
Observations

- We have to create everything just using JS
- It's painful to create an element using `React.createElement("","","")`.

Maintains everything in one place, First option is an advantage, No html, Only js. For css We will keep a separate file

2nd observation is R.CE is painful to write the simpler syntax is using "JSX". Javascript with XML

That jsx is just a shortcut for create element of react syntax. If there's any limitation in createelement syntax that limitation is also there in the jsx.



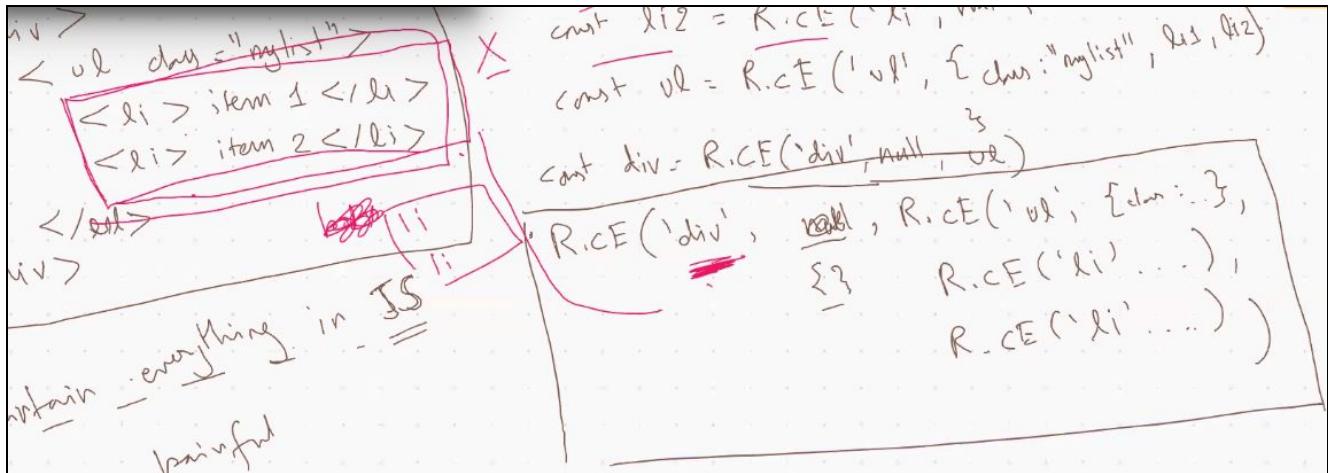
if we want to do only this

We can't do this using create element

No top level tag name is present

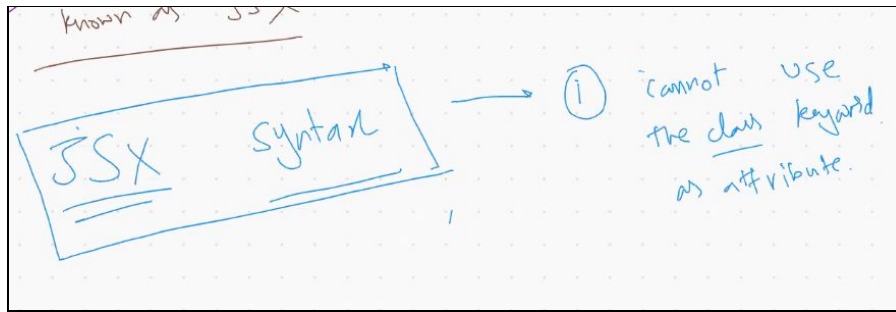
Create element needs a top level tag name

Like div is the top level tag name in the previous example

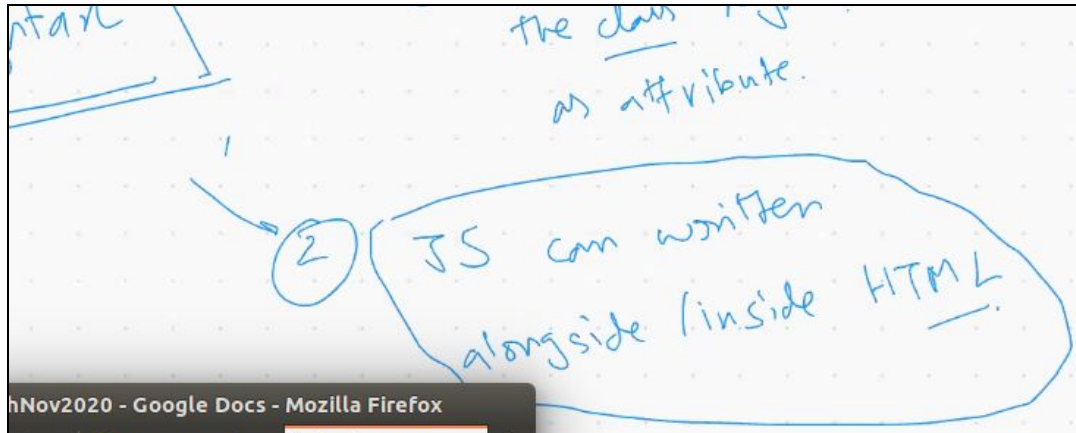


This is the limitation

You can't use the class keyword in JSX syntax

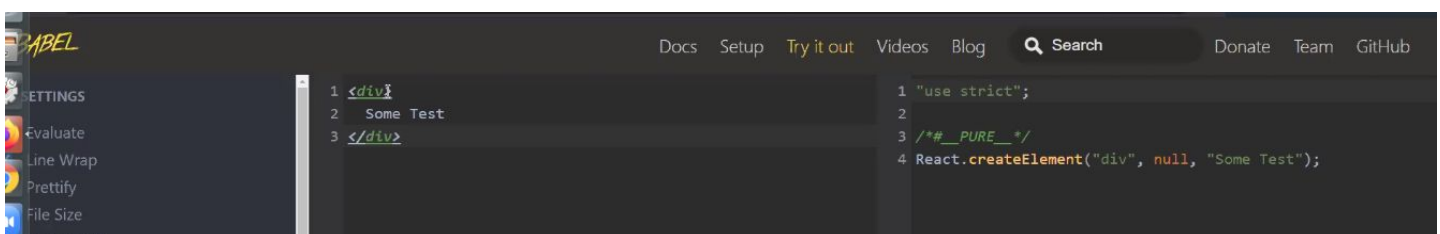
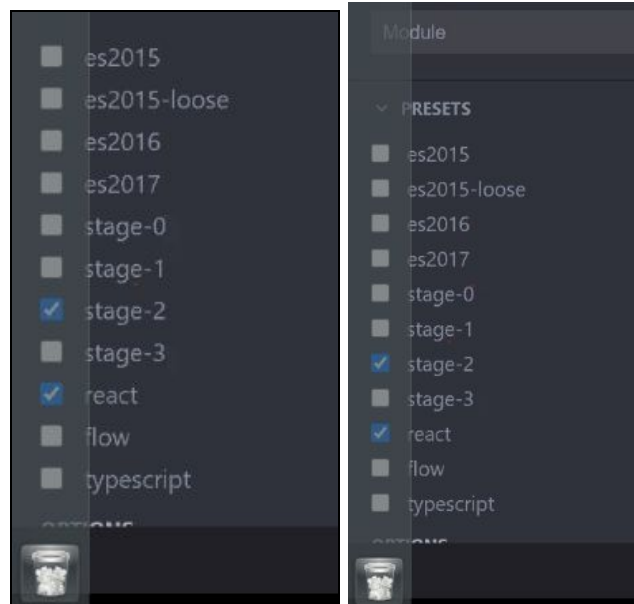


JS can be written alongside inside HTML

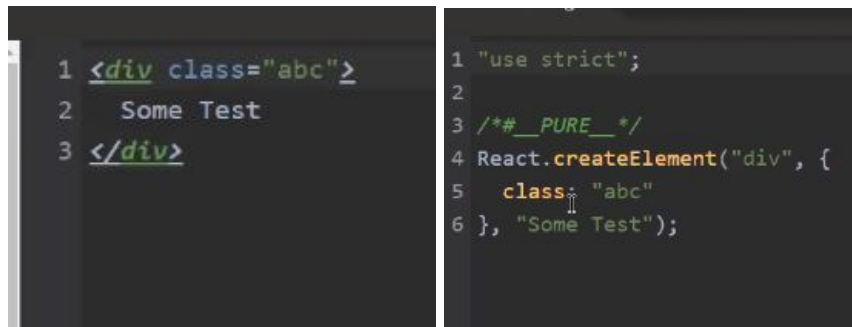


Writing first jsx using babel

click on react preset



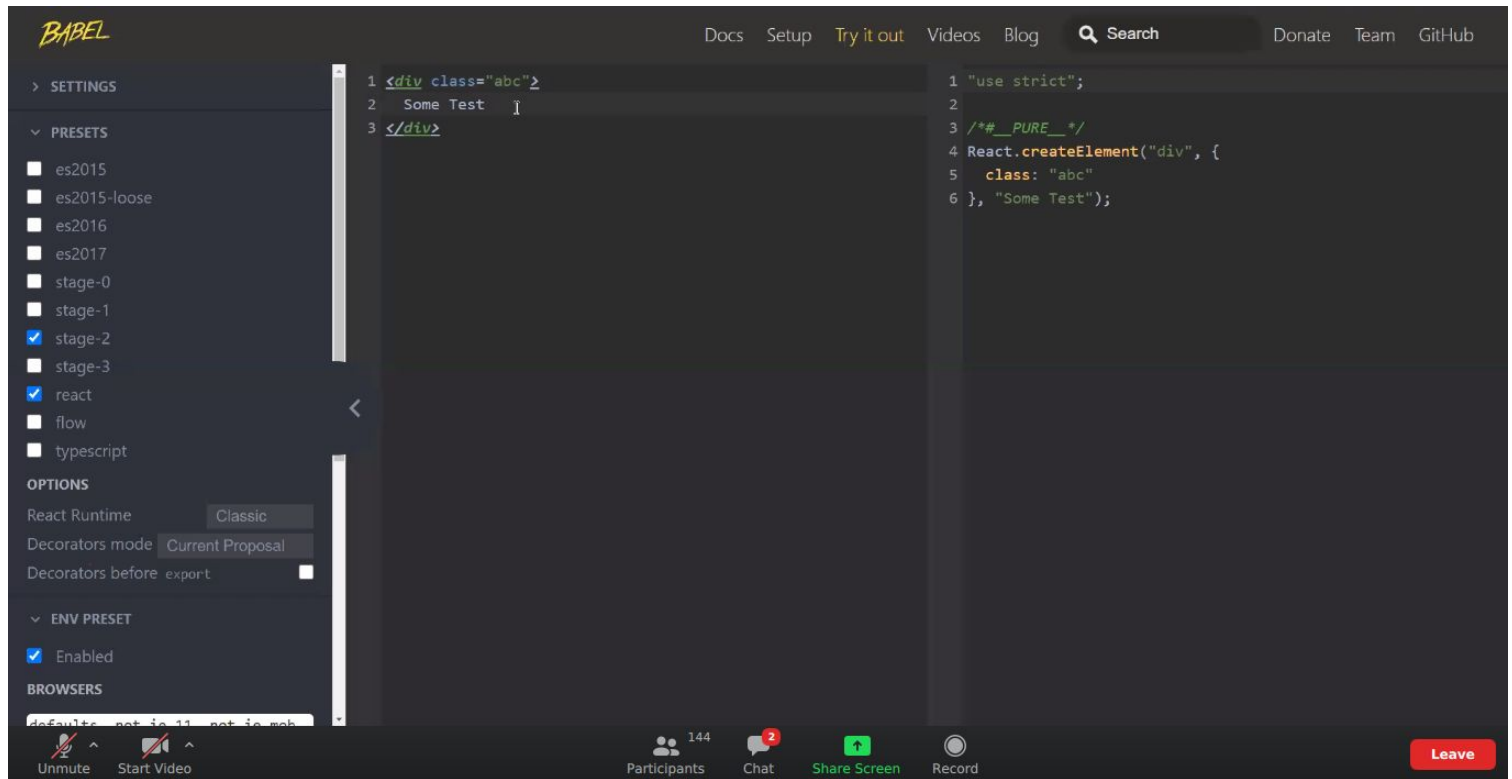
Babel used to convert JSX to React code



```
1 <div class="abc">
2   Some Test
3 </div>
```

```
1 "use strict";
2
3 /*__PURE__*/
4 React.createElement("div", {
5   class: "abc"
6 }, "Some Test");
```

Some Text -> innerText is added

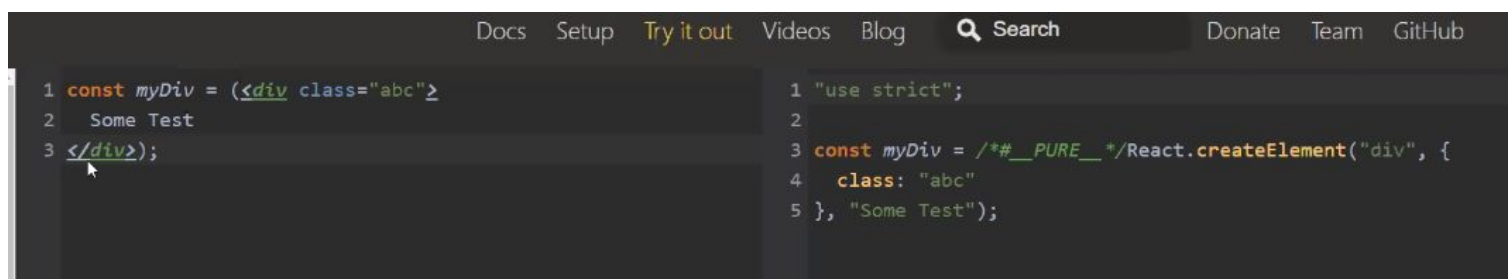


This is how it looks like

Using class keyword can be problematic.

`class` is a keyword in javascript and JSX is an extension of javascript. That's the principal reason why React uses `className` instead of `class`.

Reference -> <https://stackoverflow.com/questions/46989454/class-vs-classname-in-react-16/46991278>



```
1 const myDiv = (<div class="abc">
2   Some Test
3 </div>);
```

```
1 "use strict";
2
3 const myDiv = /*__PURE__*/ React.createElement("div", {
4   class: "abc"
5 }, "Some Test");
```



```
Docs Setup Try it out Videos Blog Search Donate Team GitHub

1 const myDiv = (<div class="abc">
2   Some Test
3 </div>);

1 "use strict";
2
3 const myDiv = /*#__PURE__*/React.createElement("div", {
4   class: "abc"
5 }, "Some Test");
```

Whatever we want to write using create element we can directly link html and assign it to some const instead of class keyword

```
Docs Setup Try it out Videos Blog Search Donate Team GitHub

1 const myDiv = (<div className="abc">
2   Some Test
3 </div>);

1 "use strict";
2
3 const myDiv = /*#__PURE__*/React.createElement("div", {
4   className: "abc"
5 }, "Some Test");
```

Instead of using class use className

Class can conflict with rest of JS

Babel is converting JSX syntax to React

CSS will never apply on virtual dom

Next example

```
1 const li1 = <li>item1</li>;
2 const li2 = <li>item2</li>;
3
1 "use strict";
2
3 const li1 = /*#__PURE__*/React.createElement("li", null,
  "item1");
4 const li2 = /*#__PURE__*/React.createElement("li", null,
  "item2");
```

```
1 const li1 = <li>item1</li>;
2 const li2 = <li>item2</li>;
3
4 const ul = <ul className="myList">
5   </ul>;
6
1 "use strict";
2
3 const li1 = /*#__PURE__*/React.createElement("li", null,
  "item1");
4 const li2 = /*#__PURE__*/React.createElement("li", null,
  "item2");
5 const ul = /*#__PURE__*/React.createElement("ul", {
6   className: "myList"
7 });
```

Use of className

New syntax observe we will add li1 and li2 in ul

Single curly brace

```

1 const li1 = <li>item1</li>;
2 const li2 = <li>item2</li>;
3
4 const ul = <ul className="myList">
5   { li1 }
6   { li2 }
7 </ul>;
8

```

```

1 "use strict";
2
3 const li1 = /*#__PURE__*/React.createElement("li", null,
  "item1");
4 const li2 = /*#__PURE__*/React.createElement("li", null,
  "item2");
5 const ul = /*#__PURE__*/React.createElement("ul", {
6   className: "myList"
7 }, li1, li2);

```

Creating outer div

```

1 const li1 = <li>item1</li>;
2 const li2 = <li>item2</li>;
3
4 const ul = <ul className="myList">
5   { li1 }
6   { li2 }
7 </ul>;
8
9 const div = <div>{ul} Some text</div>
10
11

```

```

1 "use strict";
2
3 const li1 = /*#__PURE__*/React.createElement("li", null,
  "item1");
4 const li2 = /*#__PURE__*/React.createElement("li", null,
  "item2");
5 const ul = /*#__PURE__*/React.createElement("ul", {
6   className: "myList"
7 }, li1, li2);
8 const div = /*#__PURE__*/React.createElement("div", null,
  ul, " Some text");

```

Putting some other value in div as well

Babel is properly converting

```

1 const li1 = <li>item1</li>;
2 const li2 = <li>item2</li>;
3
4 const ul = <ul className="myList">
5   { li1 }
6   { li2 }
7 </ul>;
8
9 const someOtherValue = 'some Other Value';
10
11 const div = <div>{ul} Some text {someOtherValue}</div>
12
13

```

```

1 "use strict";
2
3 const li1 = /*#__PURE__*/React.createElement("li", null,
  "item1");
4 const li2 = /*#__PURE__*/React.createElement("li", null,
  "item2");
5 const ul = /*#__PURE__*/React.createElement("ul", {
6   className: "myList"
7 }, li1, li2);
8 const someOtherValue = 'some Other Value';
9 const div = /*#__PURE__*/React.createElement("div", null,
  ul, " Some text ", someOtherValue);

```

Jsx is converting it to react.createElement

- So that second limitation is gone

Now you can use jsx to write the code and babel will convert it to react.creteelement

How to use for loop to add

100 elements inside it (forloop)

Div with 100 ili inside it

```
Docs Setup Try it out

1 const li1 = <li>item1</li>;
2 const li2 = <li>item2</li>;
3
4 const ul = <ul className="myList">
5   { li1 }
6   { li2 }
7   </ul>;
8
9 const someOtherValue = 'some Other Value';
10
11 const div = <div>{ul} Some text {someOtherValue}</div>;
12
13 const div100 = <div>
14   {
15     const liArr = [];
16     for(let i =0; i<100;i++) {
17       liArr.push(<li>Item {i+1}
18     }
19   }
20   </div>;

/repl.jsx: Unexpected token (15:10)

13 | const div100 = <div>
14 |   {
> 15 |     const liArr = [];
    |     ^
16 |     for(let i =0; i<100;i++) {
17 |       liArr.push(<li>Item {i+1}
18 |     }
19 |   }
20 |   </div>;
```

Writing the the code using standard for loop

Observe unexpected token

You cannot write for loop using this curly braces

```
Docs Setup Try it out videos blog Search Donate learn Github

1 const li1 = <li>item1</li>;
2 const li2 = <li>item2</li>;
3
4 const ul = <ul className="myList">
5   { li1 }
6   { li2 }
7   </ul>;
8
9 const someOtherValue = 'some Other Value';
10
11 const div = <div>{ul} Some text {someOtherValue}</div>;
12
13 let liArr = [li1, li2];
14 const div100 = <div>
15   {
16     liArr
17   }
18   </div>;

1 "use strict";
2
3 const li1 = /*#__PURE__*/React.createElement("li", null,
  "item1");
4 const li2 = /*#__PURE__*/React.createElement("li", null,
  "item2");
5 const ul = /*#__PURE__*/React.createElement("ul", {
6   className: "myList"
7 }, li1, li2);
8 const someOtherValue = 'some Other Value';
9 const div = /*#__PURE__*/React.createElement("div", null,
  ul, " Some text ", someOtherValue);
10 let liArr = [li1, li2];
11 const div100 = /*#__PURE__*/React.createElement("div",
  null, liArr);
```

indi

Inside

```
let liArr = [li1, li2];  
const div100 = <div>  
  {  
    liArr  
  }  
</div>;
```

inside this curly braces you can just write limited amount of javascript

Extension of this file is .jsx

.jsx

Learning how to use virtual dom

```
<html>  
  <body>  
    <div id="myApp">  
    </div>  
  </body>  
</html>
```

You have to tell react that you have to add the where to add the element

```
<html>  
  <body>  
    <div id="myApp">  
    </div>  
  </body>  
</html>
```

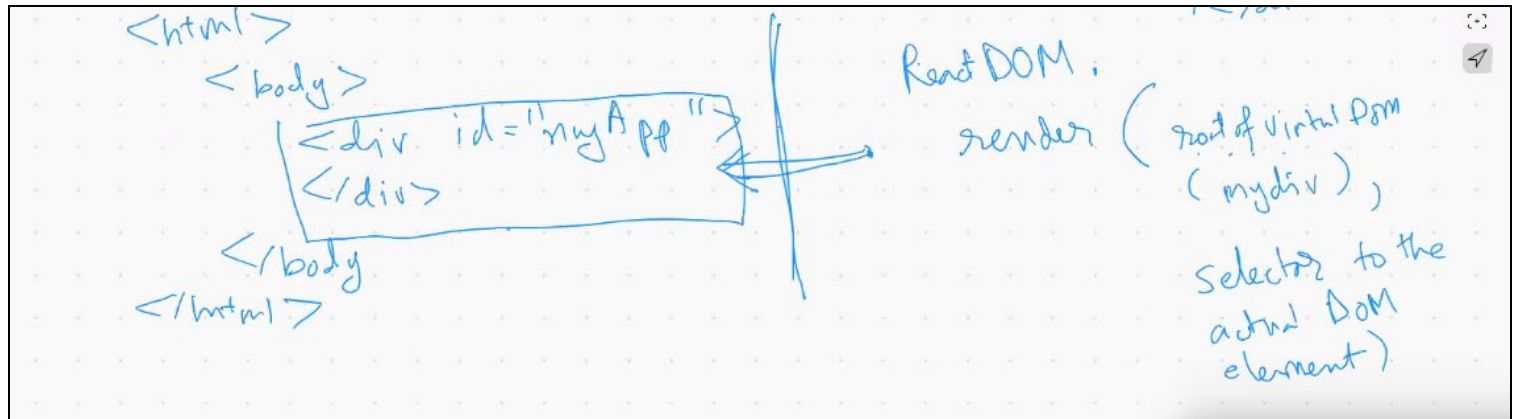
Because sometimes react replaces the entire html

We don't want that

All the virtual dom updates will happen in this div(myApp) this is the way

React will only modify the dom inside this div

do it for the first time and you never have to do it again.



first thing is root of virtual dom element i.e mydiv

Second thing is selector to actual dom element

ReactDOM.render (~~my div~~, ^{element} document.getElementById("myApp"))

This is an example

This line is written just once

Its like start updating only in this dom ie myapp div

Outer tag will always have the

Creating very simple html

This is in codesandbox

Inside your src ignore app.js
Focus on index.js

```
<> index.html x
1  <html>
2    <head>
3      <title>First React App</title>
4    </head>
5    <body>
6      <div id="myApp"></div>
7    </body>
8  </html>
```


Before virtual dom

```
1  <html>
2    <head>
3      <title>First React App</title>
4    </head>
5    <body>
6      <div>
7        Before Virtual DOM
8      </div>
9      <div id="myApp"></div>
10   </body>
11 </html>
12
```

No js till now

Starting react
I mean writing react

```
<> index.html  JS index.js
1  import React from 'react';
2  import ReactDOM from 'react-dom';
```

This library is just to hook on to dom once

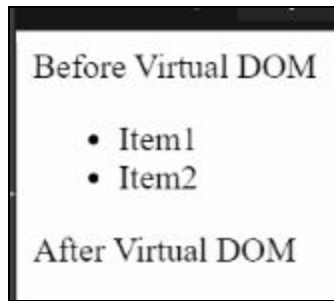
This

We are adding things to virtual dom

```
<> index.html x  JS index.js
1  <html>
2    <head>
3      <title>First React App</title>
4    </head>
5    <body>
6      <div>
7        Before Virtual DOM
8      </div>
9      <div id="myApp"></div>
10     <div>
11       After Virtual DOM
12     </div>
13   </body>
14 </html>
```

```
<> index.html  JS index.js
1  import React from "react";
2  import ReactDOM from "react-dom";
3
4  const vDom = (
5    <div>
6      <ul>
7        <li>Item1</li>
8        <li>Item2</li>
9      </ul>
10    </div>
11  );
```

```
12  
13 ReactDOM.render(vDom, document.getElementById("myApp"));  
14
```



```
< index.html  js index.js  ●  Browser  Tests  
1  import React from "react";  
2  import ReactDOM from "react-dom";  
3  
4  const vDom = (  
5    <div>  
6      <ul>  
7        <li>Item1</li>  
8        <li>Item2</li>  
9      </ul>  
10   </div>  
11 );  
12  
13 ReactDOM.render(vDom, document.getElementById("myApp"));
```

Before Virtual DOM

- Item1
- Item2

After Virtual DOM

Configuration of babel is remaining now (which converts the jsx to react)

Create react app

That will make the configuration and it is easier