React.JS for Beginner

Maifee Ul Asad

Want to start React.js,
but don’t have that
much time?
What about a
kick-start?

Hope it helps!

Intro

What is React.JS

What is React.JS

A library, not a framework

What is React.JS

A library, not a framework

- V of MVC

Why React.js ?

Why React.js ?

Why not?

Why React.js ?

Why not?

- For building UI
- Reactive
- Easy to develop
- Easy to integrate
- Widely used & huge demand

Why React.js ?

Why not?

- For building UI
- Reactive
- Easy to develop
- Easy to integrate
- Widely used & huge demand

Why React.js ?

Why not?

- For building UI
- Reactive
- Easy to develop
- Easy to integrate
- Widely used & huge demand

Why React.js ?

Why not?

- For building UI
- Reactive
- Easy to develop
- Easy to integrate
- Widely used & huge demand

Why React.js ?

Why not?

- For building UI
- Reactive
- Easy to develop
- Easy to integrate
- Widely used & huge demand

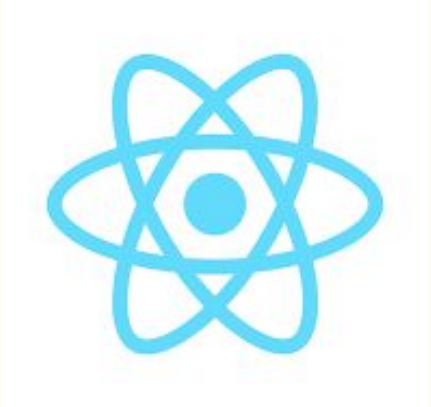
Why React.js ?

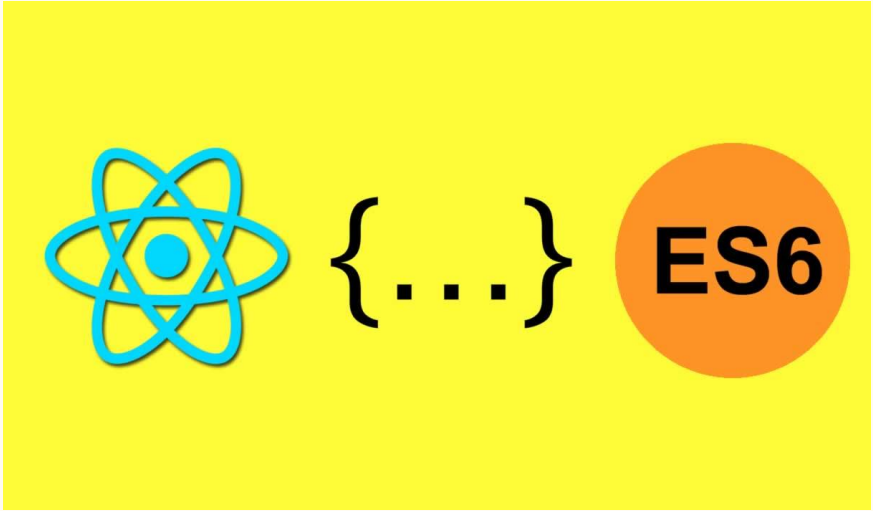
Why not?

- For building UI
- Reactive
- Easy to develop
- Easy to integrate
- Widely used & huge demand

ES6(All we need to know)

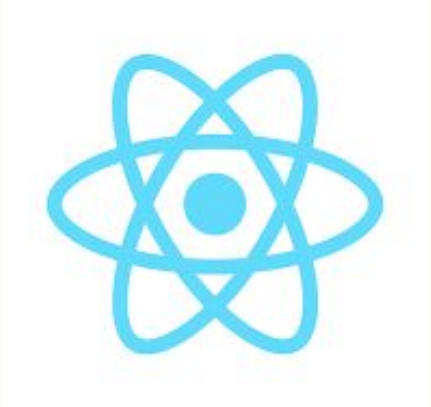
- let/const
- arrow function
- import/export
- class
- map
- promise
- destruction

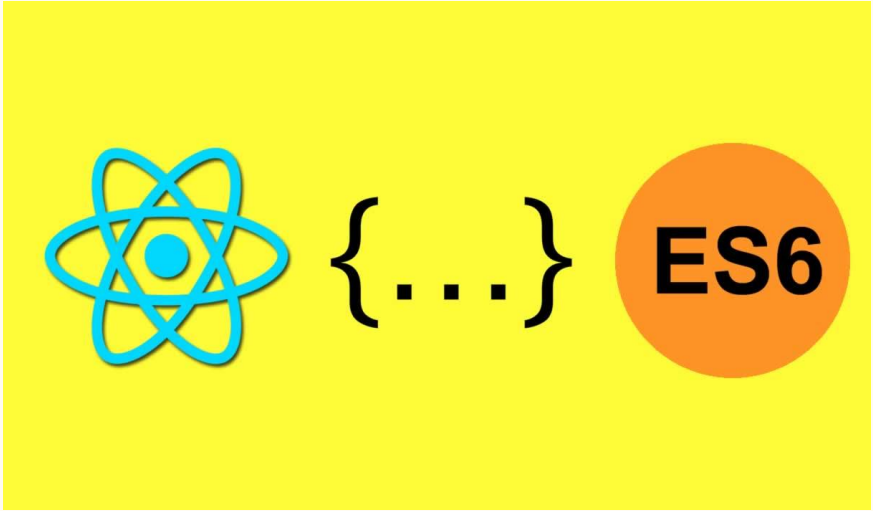




ES6(All we need to know)

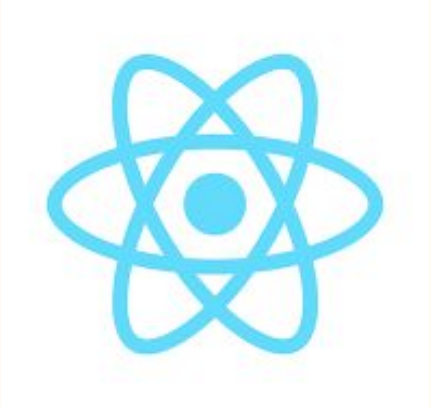
- let/const
- arrow function
- import/export
- class
- map
- promise
- destruction

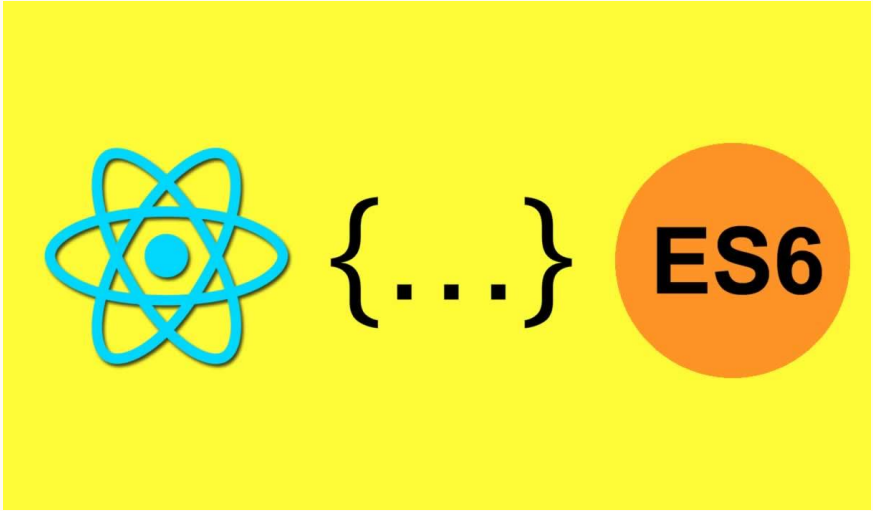




ES6(All we need to know)

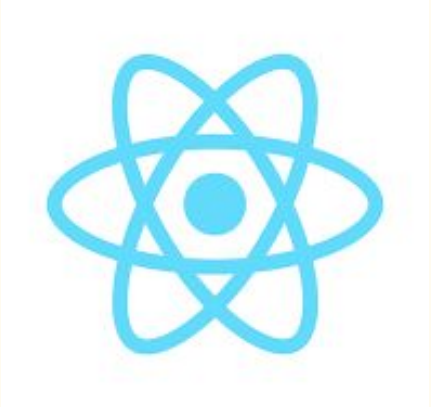
- let/const
- arrow function
- import/export
- class
- map
- promise
- destruction

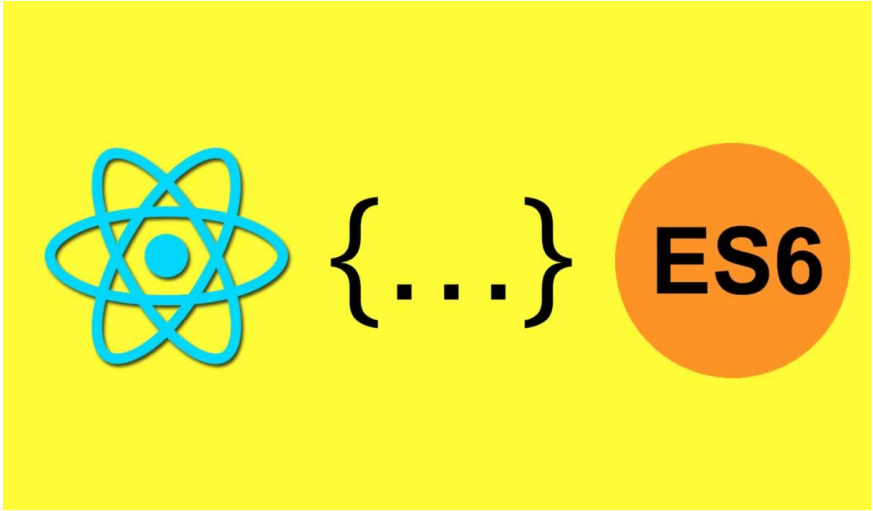




ES6(All we need to know)

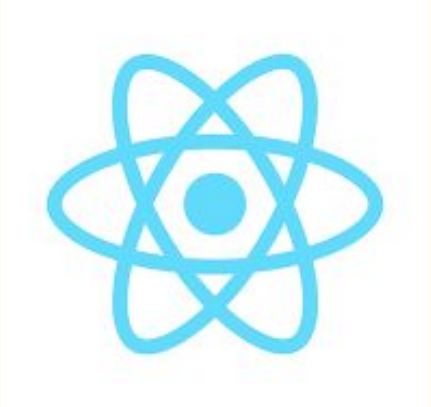
- let/const
- arrow function
- import/export
- class
- map
- promise
- destruction

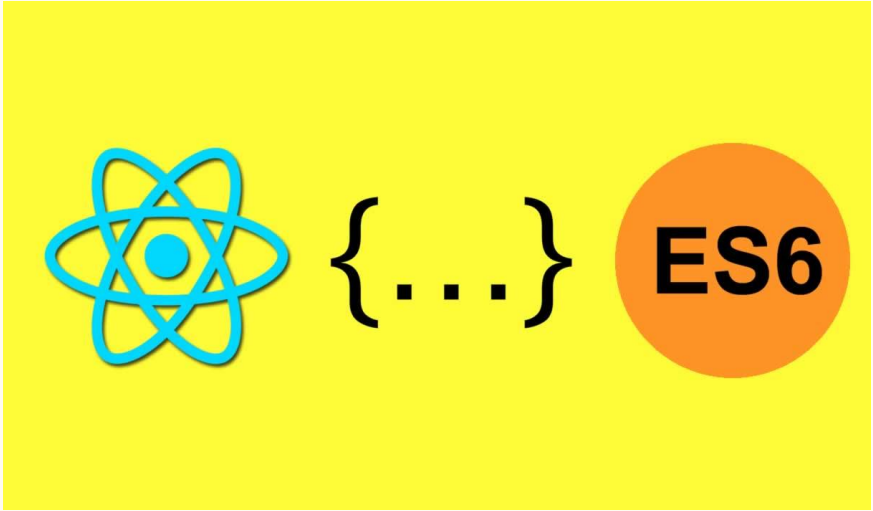




ES6(All we need to know)

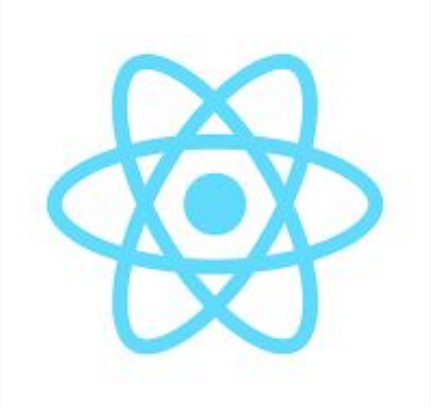
- let/const
- arrow function
- import/export
- class
- map
- promise
- destruction

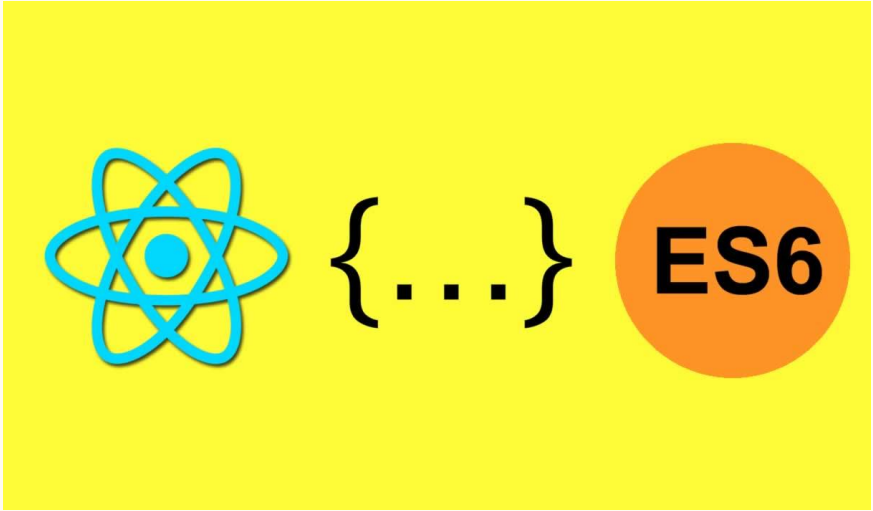




ES6(All we need to know)

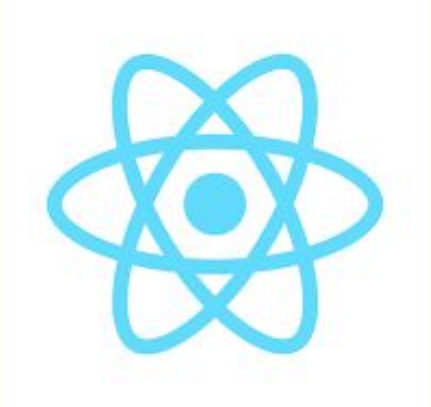
- let/const
- arrow function
- import/export
- class
- map
- promise
- destruction

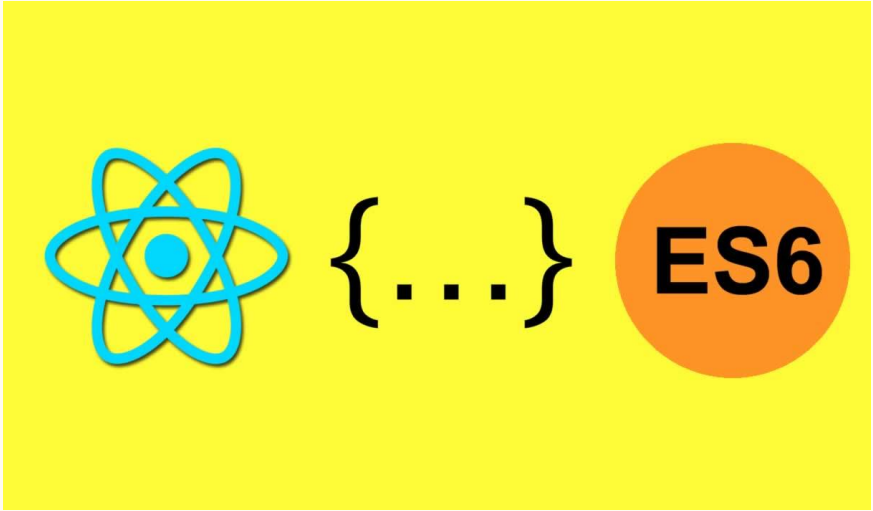




ES6(All we need to know)

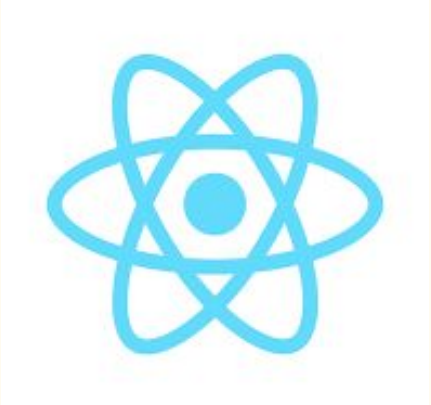
- let/const
- arrow function
- import/export
- class
- map
- promise
- destruction

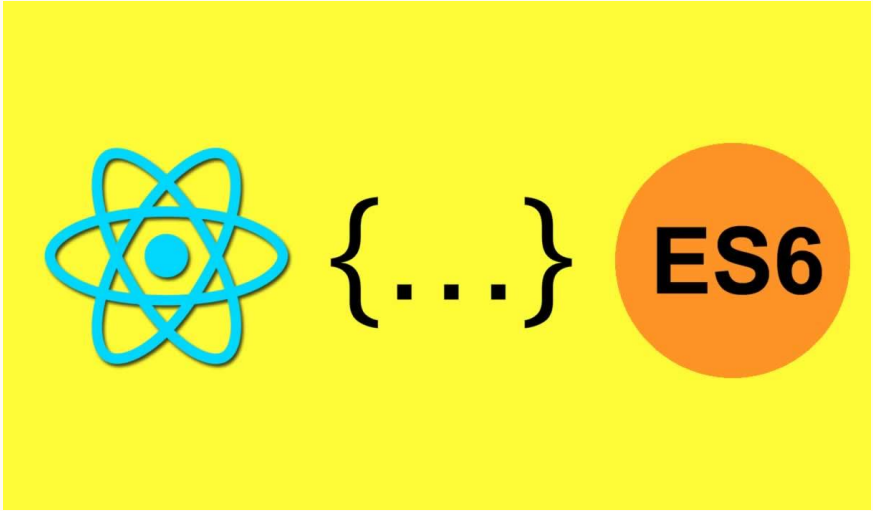




ES6(All we need to know)

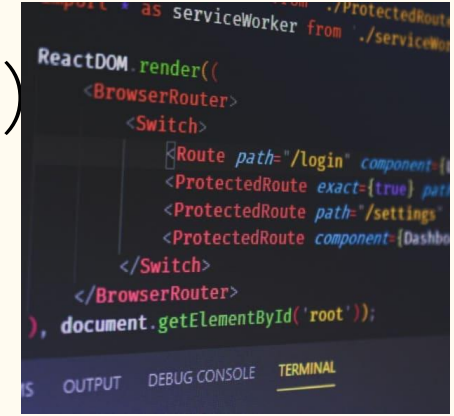
- let/const
- arrow function
- import/export
- class
- map
- promise
- destruction

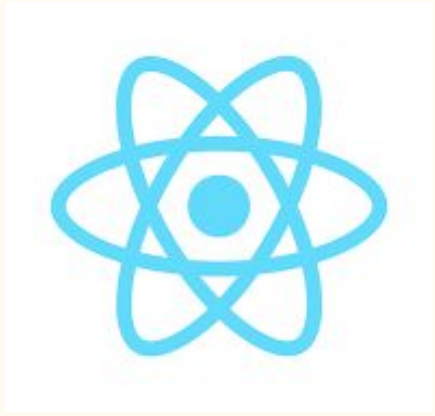


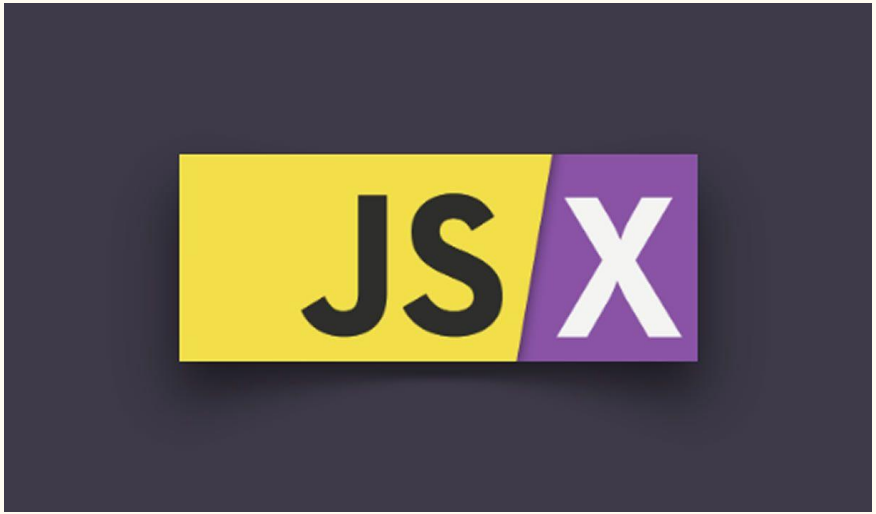


JSX(All we need to know)

- js + xml
- <something></something>
- <something/>
- They are actually object 😉
- Under the hood

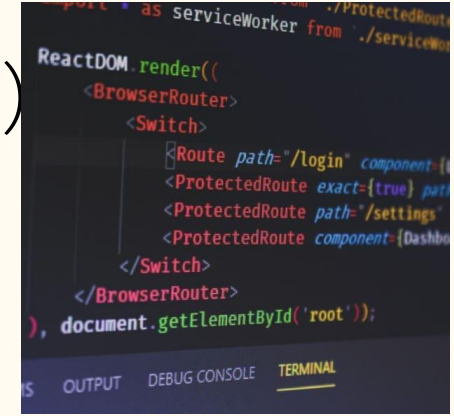


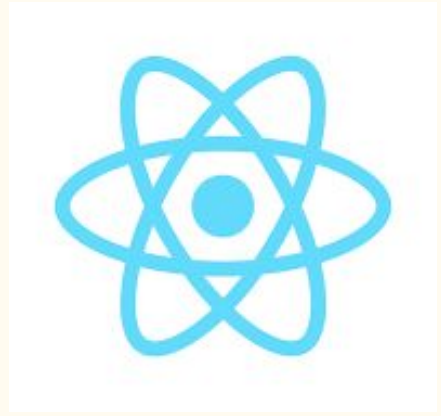


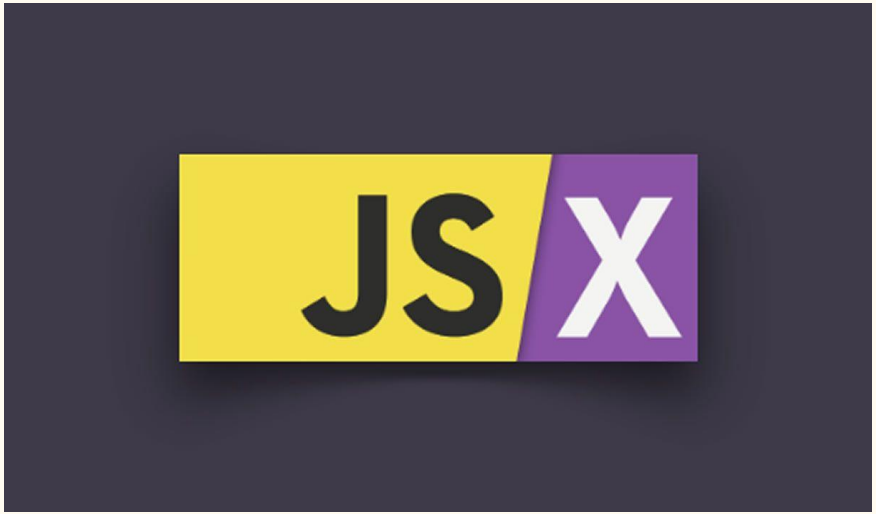


JSX(All we need to know)

- js + xml
- <something></something>
- <something/>
- They are actually object 😉
- Under the hood

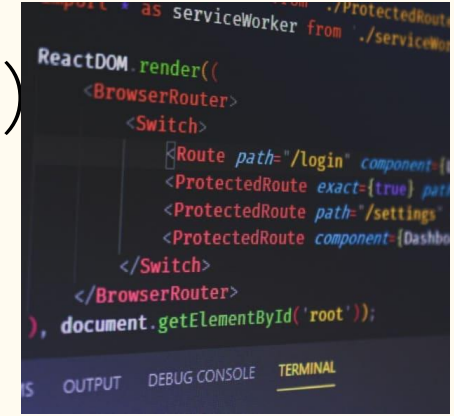


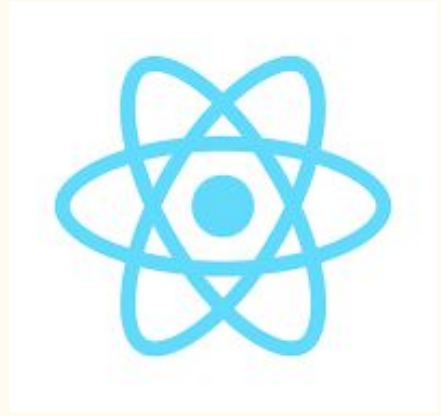


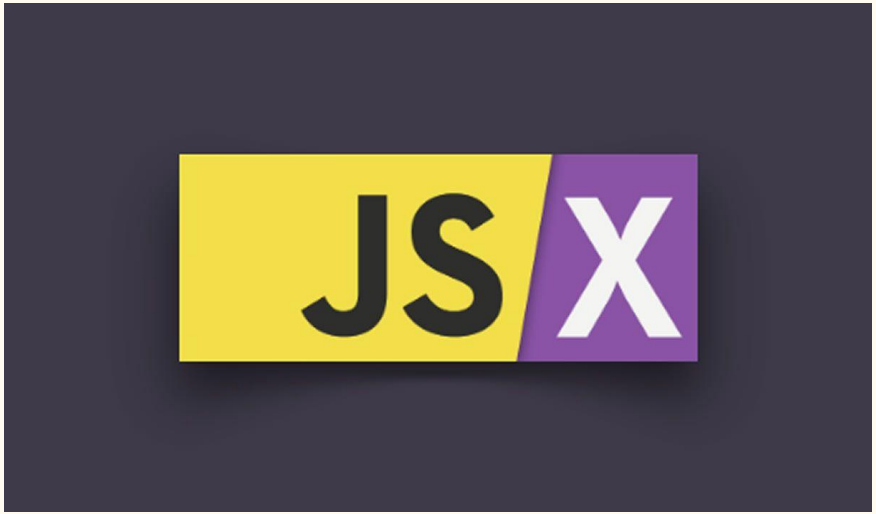


JSX(All we need to know)

- js + xml
- <something></something>
- <something/>
- They are actually object 😉
- Under the hood

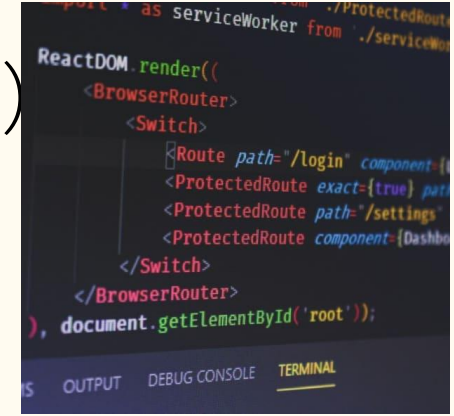


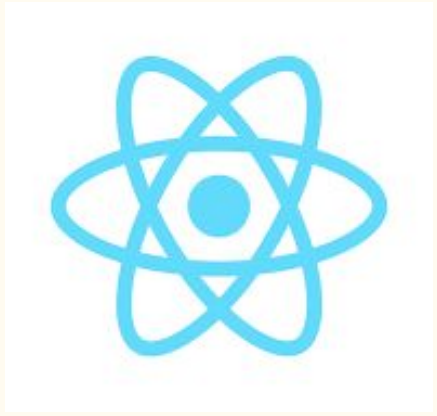


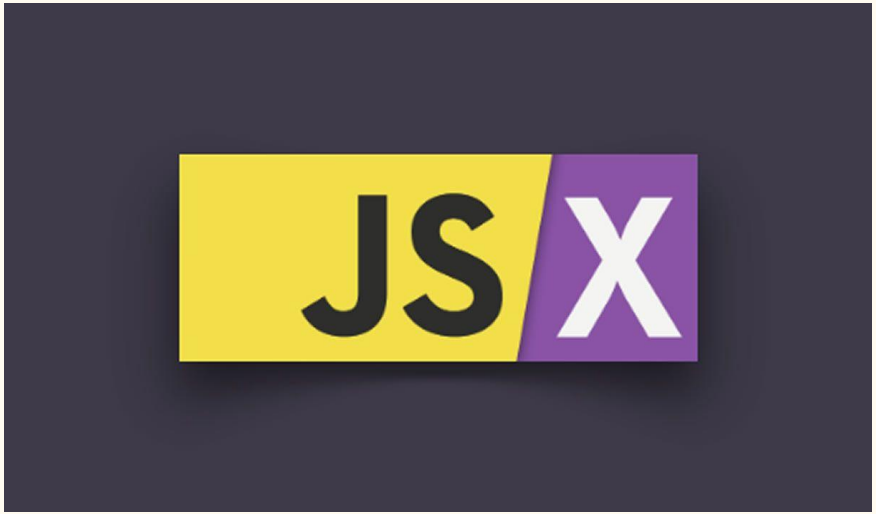


JSX(All we need to know)

- js + xml
- <something></something>
- <something/>
- They are actually object 😉
- Under the hood

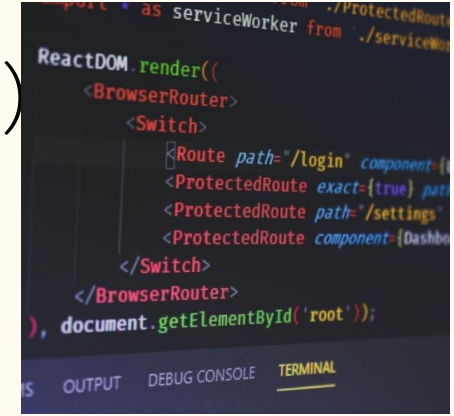


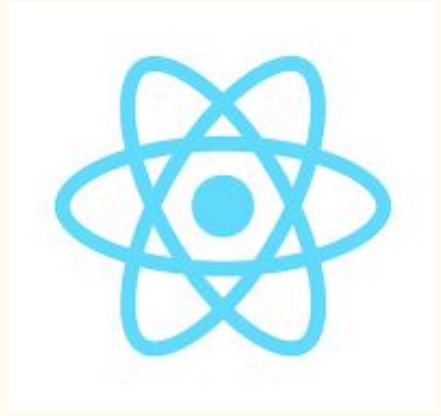


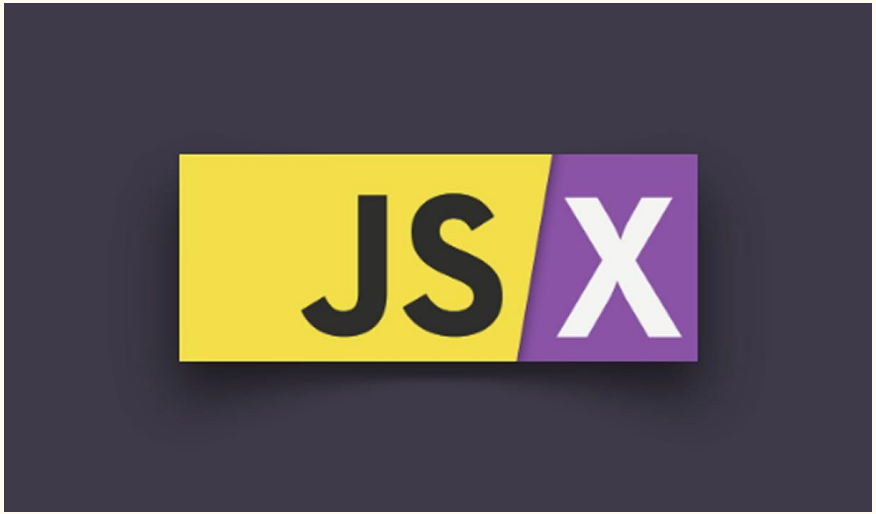


JSX(All we need to know)

- js + xml
- <something></something>
- <something/>
- They are actually object 😉
- Under the hood







State

What is ‘state’

Just some observable properties in a component

- Think like - they are just member of class, specifically field
- And they are reactive

Props

What is ‘props’

Something got passed to a component are props

- Just think we have a parent component which is calling some API, or somehow it has some data. And it will render some child component based on those data. To render those data, we need to pass those data to child first. We will be doing this with props.

Communication

Parent to Child

In one word `props`

Just pass whatever, and receive using props

- <Child name={“babu”}/>
- this.props.name

Child to Parent

In a few words `callback` as props

Instead of passing a variable, we will be passing an callback function

- <Child name={(xyz)=>{}}>
- this.props.name(‘shona’)

Child to Child

First pass to parent, then send it down to it's child

Instead of passing a variable, we will be passing an callback function

- <Child name={(xyz)=>{}} defaultName={'babu'}/>
- this.props.name('shona')

Anywhere?

Callback Hell

```
function hell(win) {
 // for listener purpose
 return function() {
   loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
     loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
         loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
           loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', function() {
             loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
               loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
                   loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
                     async.eachSeries(SCRIPTS, function(src, callback) {
                       loadScript(win, BASE_URL+src, callback);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```

Anywhere? Solution

- context
- store
- 3rd party libraries

Lifecycle

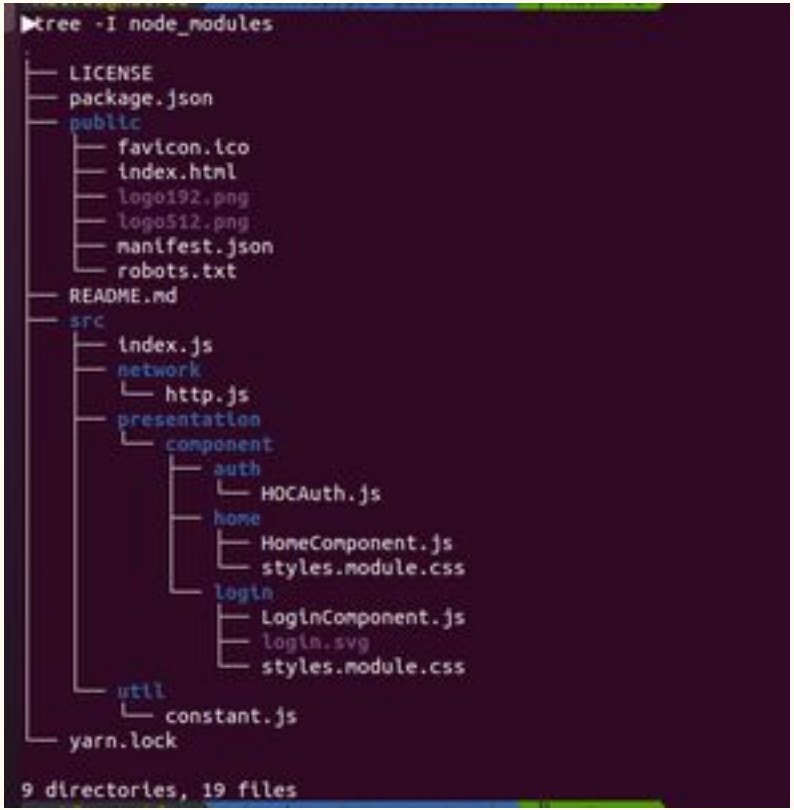
Lifecycle methods

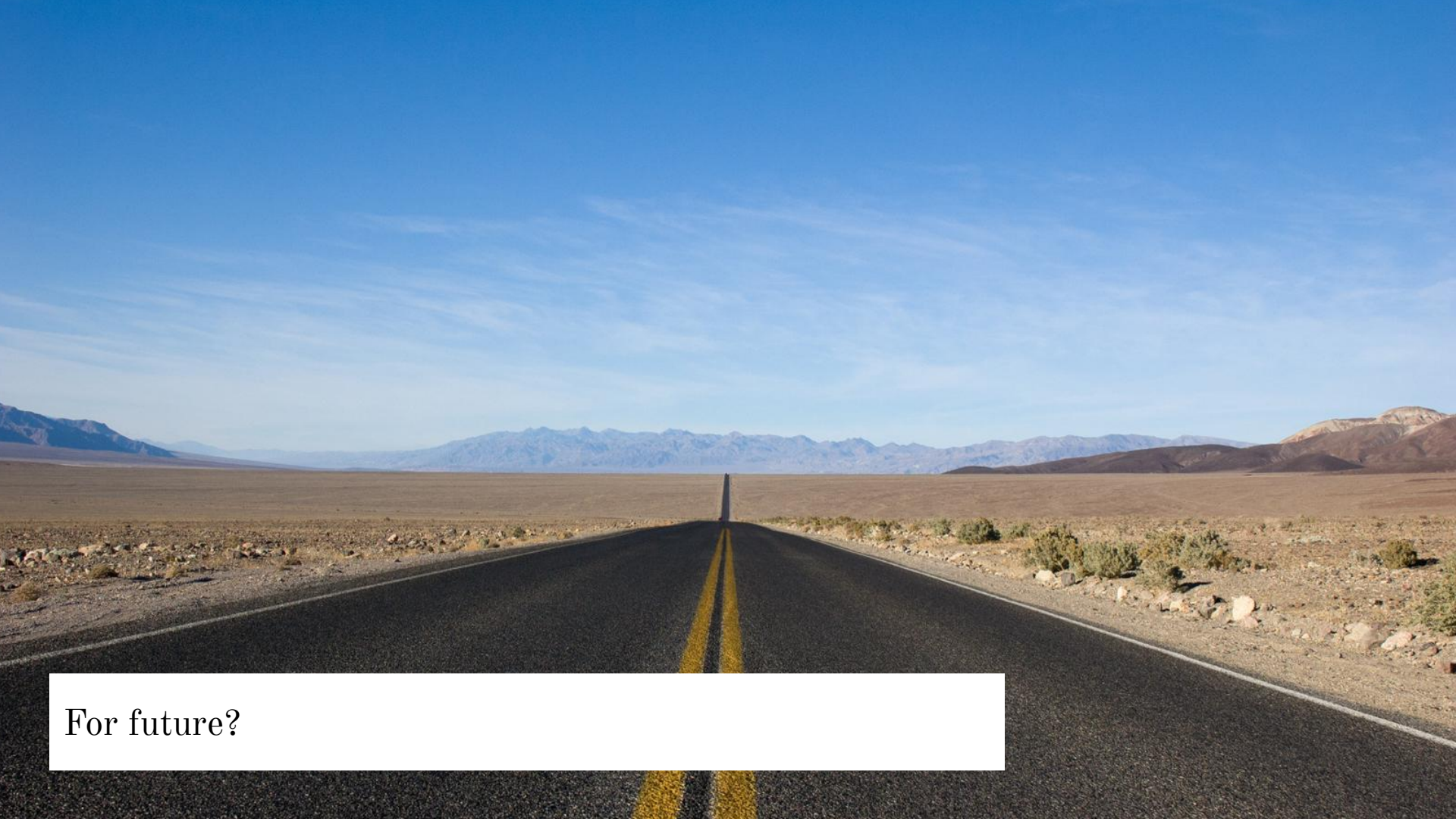
- componentDidMount
- componentDidUpdate
- componentWillUnmount

Project Structure

A suggested structure? No, not really

But a suggested one for Beginners





For future?

HOC & TypeScript & Redux & Deep Dive

These will be published, when they will be published.

Tada

Keep me in your prayer.

And I post randomly on social networks.

YouTube tutorials are "videos of, by and for the noob"

Peace out.