<https://github.com/Mayankgupta688/DecemberSession.git>

<https://github.com/Mayankgupta688/SessionNovember>

<https://medium.technofunnel.net/>

<https://prod.liveshare.vsengsaas.visualstudio.com/join?85733FE73BA357BB37228FC9BAB13A11B6C5>

javascript is purely functional, class is syntactical sugar

javascript prototype Chaining

(https://dzone.com/articles/prototype-amp-prototype-chaining-with-object-orien)

Reacts promotes functional programming

JavaScript hoisting (happens when it’s a function or var defined variables)

JavaScript Scoping - [JavaScript | Hoisting - GeeksforGeeks](https://www.geeksforgeeks.org/javascript-hoisting/)

[How V8 compiles JavaScript code ? - GeeksforGeeks](https://www.geeksforgeeks.org/how-v8-compiles-javascript-code/#:~:text=%20How%20V8%20compiles%20JavaScript%20code%20%3F%20,Stack%20of%20the%20V8%20engine%E2%80%99s%20runtime...%20More%20)

JavaScript Interpreted or Complied (double interpretation)

Kyle Simpson - Author

**Javascript - Async or Sync**

http://latentflip.com/loupe/?code=%3D%3D!!!

var data = 10;

var sum = 0;

setTimeout(function First() {

console.log("Data for SetTimeout 1 is: " + data)

}, 5000)

setTimeout(function Second() {

console.log("Data for SetTimeout 2 is: " + data)

}, 15000)

setTimeout(function Third{

console.log("Data for SetTimeout 3 is: " + data)

}, 10000)

console.log("Data from Outside" + data);

// 10 seconds

for(let i = 0 ; i < 100 ; i++) {

sum = sum + i

}

Not Single Threaded Architecture

Single Threaded Execution Model with multiple thread in the background for Async

(Call Stack,Web Api,Callback Queue)

<https://medium.com/technofunnel>

<https://www.w3schools.com/js/default.asp>

Context API, Closures

**ES6 – Features**

<https://www.boardinfinity.com/blog/top-10-features-of-es6/>

**let** - block scope/Lexical scope

**var** - function scope

const userDetails = {

name: "Raj"

age: 20

}

userDetails.age = 37

console.log(userDetails);

const dataArray = [1,2,3]

dataArray.push(4) // data can be changed,

//but new memory allocation is not allowed

console.log(dataArray);

const obj = {

prop: 42

};

Object.freeze(obj);

obj.prop = 33; // Throws an error in strict mode

**Spread and rest operators**

* Spread works with comma separated values
* copies and creates a new objects

const userDetails = {

name: "Raj",

age: 20,

designation: manager

}

var newDetails = {

…userDetails;

};

*Spread operator for an complex object*



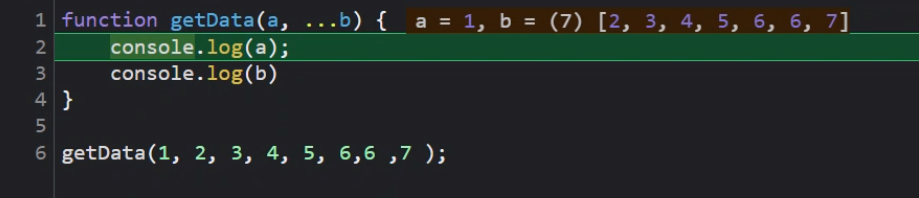
**Deep and Shallow Copies**

* <https://medium.com/@gamshan001/javascript-deep-copy-for-array-and-object-97e3d4bc401a>
* <https://www.freecodecamp.org/news/copying-stuff-in-javascript-how-to-differentiate-between-deep-and-shallow-copies-b6d8c1ef09cd/>

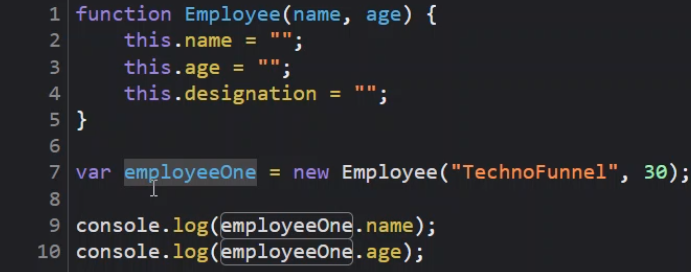
For in loop: <https://www.w3schools.com/jsref/jsref_forin.asp>

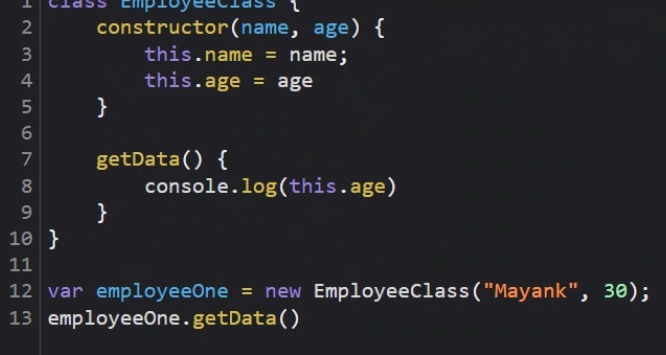
**Rest Operator**

//b will become will an array as the more two parameter



**Classes (prototype internally)**

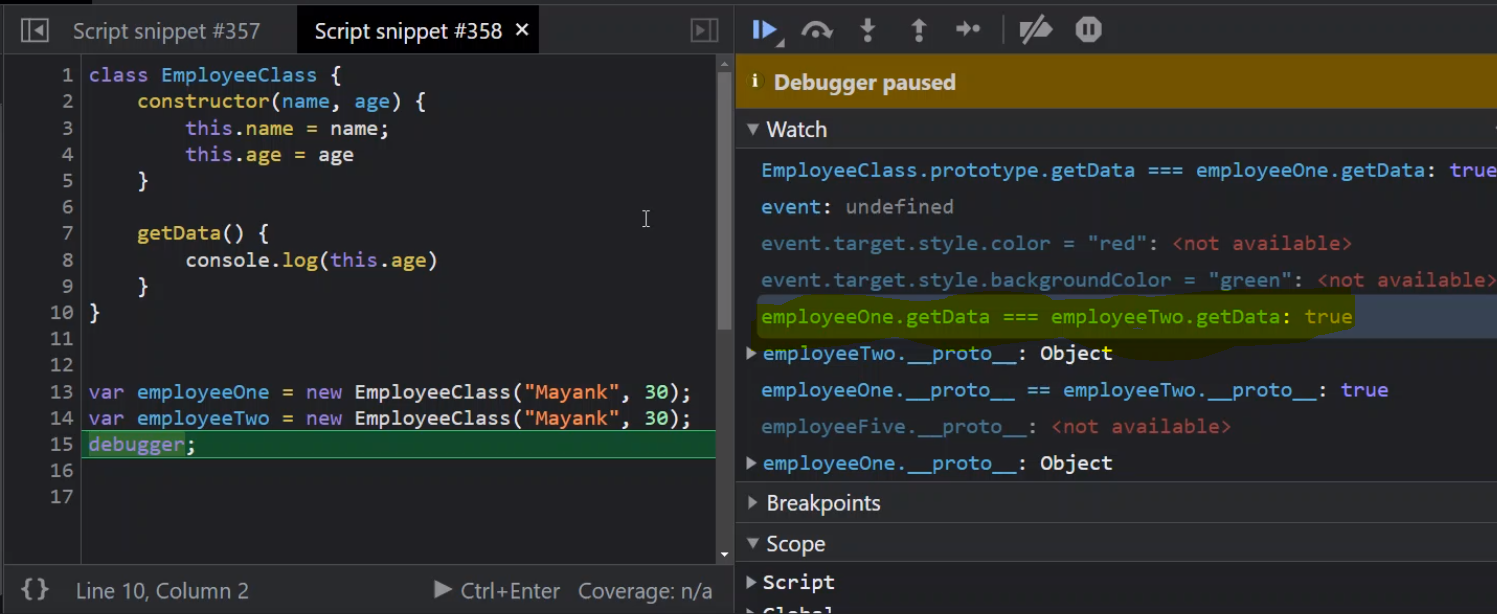




<https://www.typescriptlang.org/play>

Getdata function’s memory is allocated with Employee class not employeeOne object, this saves memory (prototype).

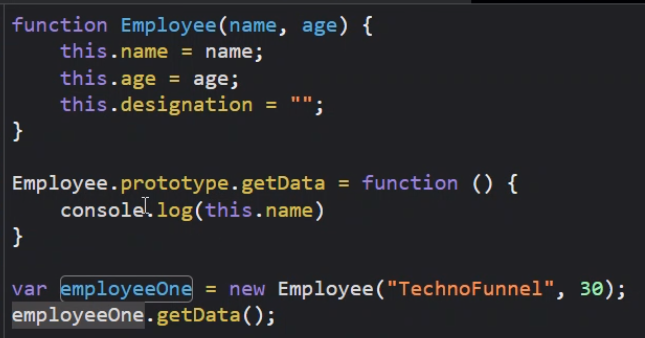
i.e class memory not object memory



Prototype chaining

[**https://medium.com/developers-arena/javascript-classes-inheritance-and-prototype-chaining-es5-and-es6-way-4b8e9416702b**](https://medium.com/developers-arena/javascript-classes-inheritance-and-prototype-chaining-es5-and-es6-way-4b8e9416702b)

**Example:**



**Getting started – create react app**

npm install -g create-react-app

create-react-app employee-details

cd employee-details

npm start

**debugging steps**

npm cache clean –force

npm update npx

npm install npm@latest -g

Why Jsx – JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement()  and/or appendChild() methods. JSX converts HTML tags into react elements.

Babel - Babel can convert JSX syntax!

<https://www.geeksforgeeks.org/jsx-full-form/#:~:text=Advantages%20of%20JSX%3A,()%20or%20createElement()%20method>.

Every component name must start with Capital letter.

import React from "react";

import ReactDOM from "react-dom";

import component, { HeaderComponent } from "./components/HeaderComponent";

import ContentComponent from "./components/ContentComponent";

import FooterComponent from "./components/FooterComponent";

alert(component)

ReactDOM.render((

    <div>

        <h1>These are the Application Components</h1><hr/>

        <HeaderComponent></HeaderComponent><hr/>

        <ContentComponent></ContentComponent><hr/>

        <FooterComponent></FooterComponent><hr/>

    </div>

), document.getElementById("root"));

Without JSX code will look like – html code becomes too complex and tedious to write. Example below:

import React from "react";

export default function HeaderComponent() {

    return React.createElement("div", null, [

            React.createElement("div", null,

                React.createElement("h1", null,

                    "This is my header")

            ),

            React.createElement("div", null,

                [

                    React.createElement("h1", null,

                        "This is my header"),

                    React.createElement("p", null,

                        React.createElement("h2", null,

                        "This is Non-Critical Header")

                        )

                ]

            )

        ]

    )

}

<div>

    <div>

        <h1>This is My Header</h1>

    </div>

    <div>

        <h1>This is My Header</h1>

        <p>

            <h2>This is Non Critical Header</h2>

        </p>

    </div>

</div>

**Interpolation aka Curly Braces**

export default function InterpolatingDataComponent() {

    function getData(){

            return "Raj"

    }

    var name ="Thyagaraju";

    var lname="Govardhan";

    return (

        <div>

            <h1>Hello World.... { getData()+ " " +  1 + 1 } </h1>

            <h1> { name +" "+lname } </h1>

        </div>

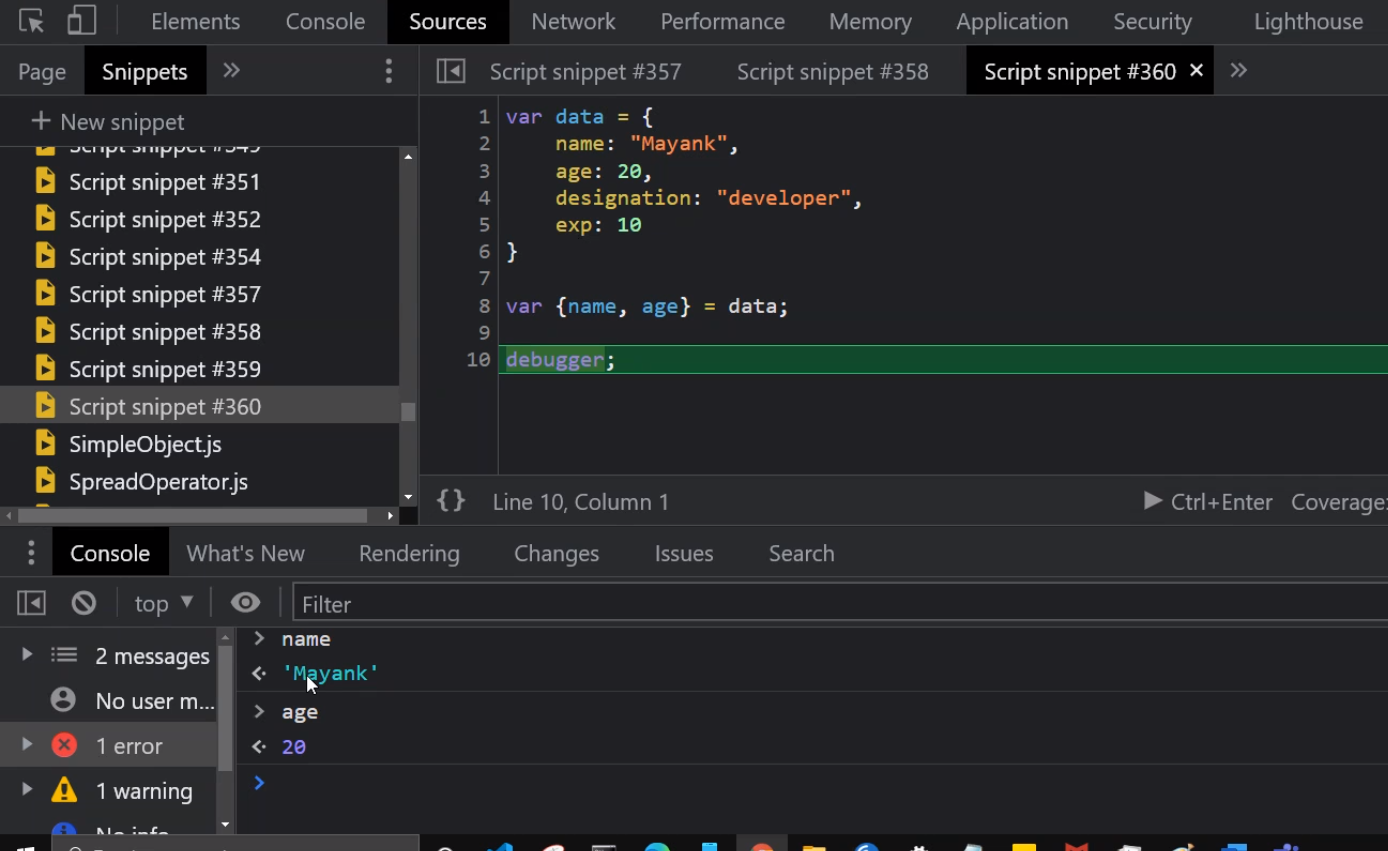
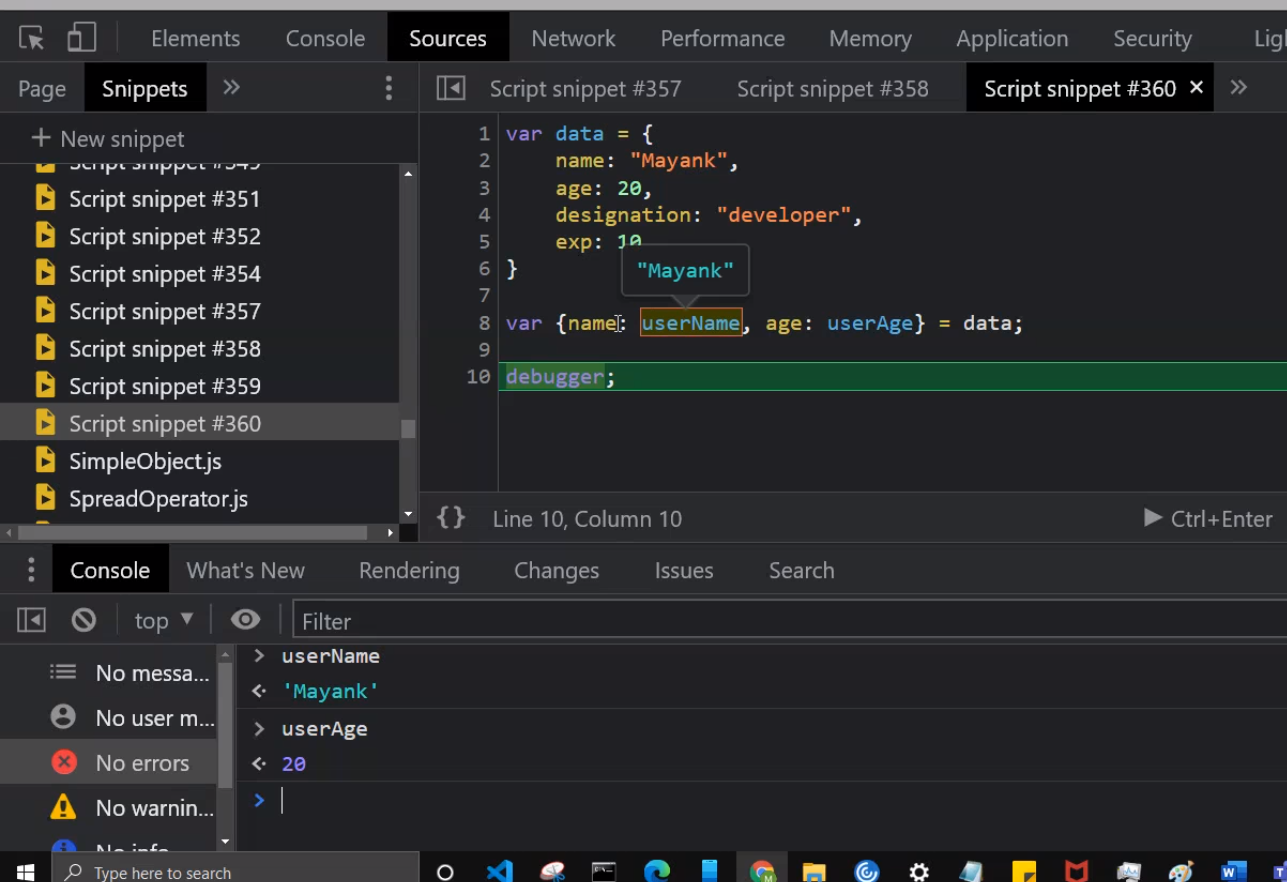
    )

}

**Read:** RxJs (Observables) / Generator

**Presentational Component**

**Destructing -** [**https://medium.com/@pyrolistical/destructuring-nested-objects-9dabdd01a3b8**](https://medium.com/@pyrolistical/destructuring-nested-objects-9dabdd01a3b8)

**Higher Order Functions**

A function that accepts and/or returns another function is called a **higher-order function**.

It’s higher-order because instead of strings, numbers, or booleans, it goes higher to operate on functions. Pretty meta.

With functions in JavaScript, you can

1. Store them as variables
2. Use them in arrays
3. Assign them as object properties (methods)
4. Pass them as arguments
5. Return them from other functions

<https://www.freecodecamp.org/news/a-quick-intro-to-higher-order-functions-in-javascript-1a014f89c6b/>

**Use of Map function**

export default function EmployeeListComponent() {

    return (

        <div>

            { employeeList.map((emp,index) => {

                return (

                    <div className="card">

                        <img src={emp.avatar} class="card-img-top" alt="..." />

                        <div className="card-body">

                            <h5 className="card-title">{emp.name}</h5>

                            <p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

                            <input type="button" value={"Delete " + emp.name} className="btn btn-primary" />

                        </div>

                    </div>

                )

            })}

        </div>

    )

}

**Use of Filter function**

 function getEventDetails(event, employeeId) {

        //debugger;

        employeeList = employeeList.filter((emp) => {

            return emp.id !== employeeId

        })

        alert(employeeList.length)

    }

**Promises -** [**https://javascript.info/promise-basics**](https://javascript.info/promise-basics)

**State Component:**

import React from "react";

export default class TimerComponent extends React.Component {

    constructor() {

        super();

        this.state = {

            time: this.getCurrentTime()

        }

        setInterval(() => {

            this.setState({

                time : this.getCurrentTime()

            })

            console.log(this.state.time);

        }, 1000)

    }

     getCurrentTime() {

        var currentDate = new Date();

        return currentDate.getHours() + " : " + currentDate.getMinutes() + " : " + currentDate.getSeconds();

    }

    render() {

        return (

            <div>

                <h1>Current Time is: {this.state.time}</h1>

            </div>

        )

    }

}

**This keyword and bind function**

**Read: call, bind and apply**

**Arrow function preserves this current context**

getEventDetails = (event) => {

        debugger;

        var updatedList = this.state.employeeList.filter((emp) => {

            return emp.id !== event.target.id

        })

        this.setState({

            employeeList: updatedList

        })

    }

<input type="button" id={emp.id} value={"Delete " + emp.name} onClick={this.getEventDetails} className="btn btn-primary" />

**Or**

export default class EmployeeListingClass extends React.Component  {

    constructor() {

        super();

        this.state = {

            employeeList: empList

        }

        this.sampleData= {

            name: "adfhakgkjs"

        }

        this.getEventDetails = this.getEventDetails.bind(this);

    }

**Or**

**Use bind function on**

<input type="button" id={emp.id} value={"Delete " + emp.name} onClick={this.getEventDetails.bind(this)} className="btn btn-primary" />

**Typescript – Provides type safety**

**Read – Controlled and Uncontrolled element in relation with Dom Difference**

**Uncontrolled elements are the ones not managed/controlled by React. i.e not having JSX markup**

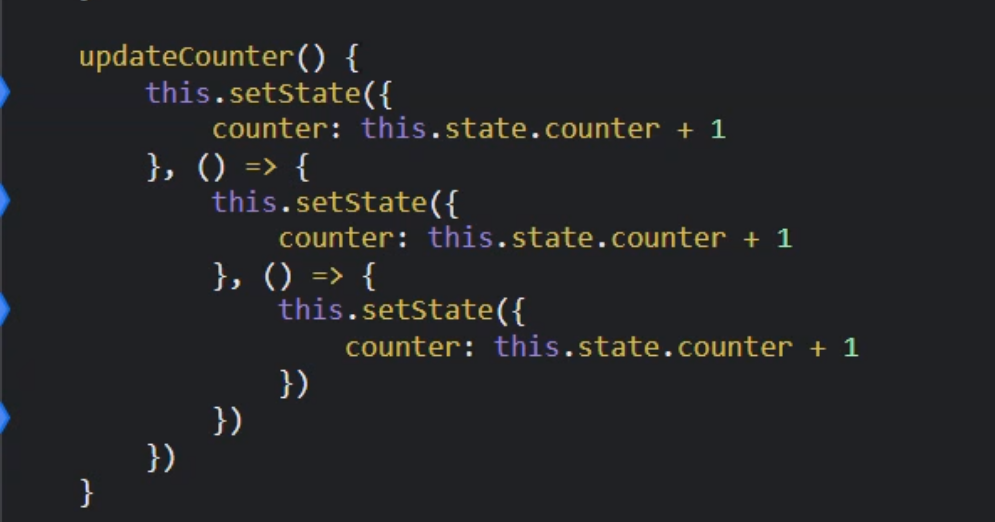
[**https://reactjs.org/docs/uncontrolled-components.html**](https://reactjs.org/docs/uncontrolled-components.html)

**SetState is Async**

[**https://stackoverflow.com/questions/33613728/what-happens-when-using-this-setstate-multiple-times-in-react-component**](https://stackoverflow.com/questions/33613728/what-happens-when-using-this-setstate-multiple-times-in-react-component)

**React detects if setState called multiple time, then will batch setState operations, this is done for performance.**

**Chaining setState:**



**Axios mock api -** [**https://5a530e1477e1d20012fa066a.mockapi.io/login**](https://5a530e1477e1d20012fa066a.mockapi.io/login)

[**https://priceapi.moneycontrol.com/pricefeed/nse/equitycash/NAC**](https://priceapi.moneycontrol.com/pricefeed/nse/equitycash/NAC)**​**

[**https://priceapi.moneycontrol.com/pricefeed/bse/equitycash/SBI**](https://priceapi.moneycontrol.com/pricefeed/bse/equitycash/SBI)

shouldComponentUpdate

componentDidMount

**Read:**

[**https://www.geeksforgeeks.org/reactjs-pure-components/**](https://www.geeksforgeeks.org/reactjs-pure-components/)

[**https://stackoverflow.com/questions/42756354/should-i-use-react-purecomponent-everywhere**](https://stackoverflow.com/questions/42756354/should-i-use-react-purecomponent-everywhere)

[**https://medium.com/welldone-software/react-when-should-pure-components-be-used-56c4428fe970**](https://medium.com/welldone-software/react-when-should-pure-components-be-used-56c4428fe970)

[**https://medium.com/technofunnel/working-with-react-pure-components-166ded26ae48**](https://medium.com/technofunnel/working-with-react-pure-components-166ded26ae48)

[**https://stackoverflow.com/questions/54374046/what-is-the-meaning-of-mutable-state-in-react-forms-or-generally**](https://stackoverflow.com/questions/54374046/what-is-the-meaning-of-mutable-state-in-react-forms-or-generally)

"Mutable" in programming refers to anything that can change during the running of a program. The [dictionary definition](https://www.dictionary.com/browse/mutable) is "liable or subject to change or alteration".

The opposite would be "[Immutable](https://en.wikipedia.org/wiki/Immutable_object)" (i.e. something that cannot change). In the context of a React component, this would be the props.

So, in the context of a form, the mutable state would be made up of the contents of the inputs that make up the form. (Unless of course you set one of those inputs to be read only!)

**Observables – RxJs**

npm install --save rxjs

Both Observables are promises are Aysnc

<https://stackoverflow.com/questions/37364973/what-is-the-difference-between-promises-and-observables>

**RxJs** –

Offering a powerful, functional approach for dealing with events and with integration points into a growing number of frameworks, libraries, and utilities, the case for learning Rx has never been more appealing.

Read Subjects

A Subject is a special type of Observable which shares a single execution path among observers.

Very important –

[Subjects - Learn RxJS](https://www.learnrxjs.io/learn-rxjs/subjects)

[Introduction - Learn RxJS](https://www.learnrxjs.io/)

[Understanding RxJS Subjects, Behavior Subjects, and Replay Subjects Using a Deck of Cards | by Benni Russell | Medium](https://medium.com/@bennirus/understanding-rxjs-subjects-behaviour-subjects-replay-subjects-with-a-deck-of-cards-5e5a3aac096f)

Subject vs Behavioural subject – Subject does not provide initial value, Behavioural subject

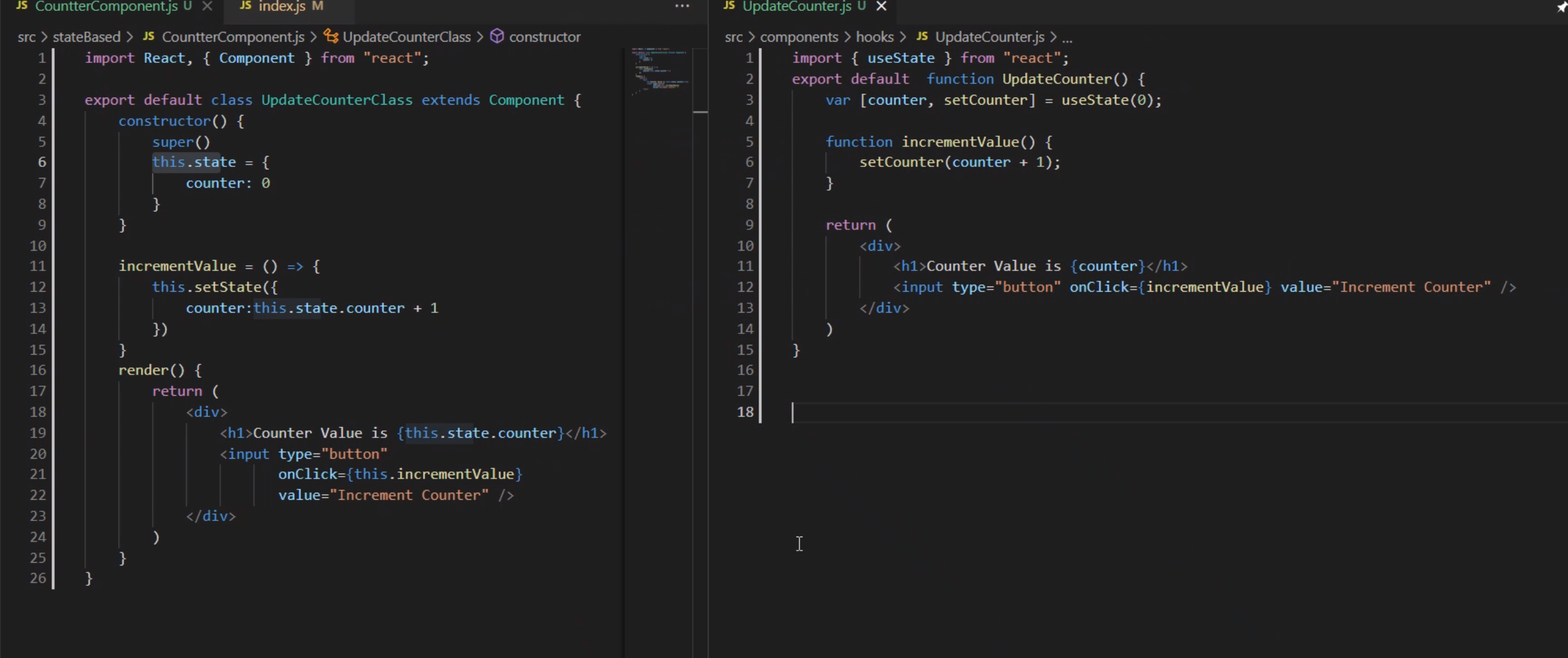
Provides first initial value to the subscriber.

Using subject you can wire communication between parent, child and siblings also

**React Hooks**

<https://reactjs.org/docs/hooks-intro.html>

Functional component ability to own data i.e., State



[Making setInterval Declarative with React Hooks — Overreacted](https://overreacted.io/making-setinterval-declarative-with-react-hooks/)

SetInterval is called once and is not destroyed. SetTimeout is calls itself after an interval and is destroyed immediately

[React Class Component vs Functional Component: How To Choose (telerik.com)](https://www.telerik.com/blogs/react-class-component-vs-functional-component-how-choose-whats-difference)