20 ReactJS Questions and Answers (Part 1)

If you are an aspiring front end developer and preparing for interviews, then this post is for you. This post is the perfect guide for you to learn all the concepts required to clear a React interview.

# ****1. What is ReactJS?****

ReactJS is an open-source frontend JavaScript library which is used for building user interfaces, specifically for single page applications. It is used for handling view layer for web and mobile apps. React was created by Jordan Walke, a software engineer working at Facebook. ReactJS was first deployed on Facebook’s newsfeed in 2011 and on Instagram.com in 2012.

# 2. What are the advantages and disadvantages of ReactJS?

**Advantages:**

1. Increases the application’s performance with Virtual DOM
2. JSX makes code is easy to read and write
3. It renders both on client and server side
4. Easy to integrate with other frameworks(Angular, BackboneJS) since it is only a view library
5. Easy to write UI Test cases and integration with tools such as JEST.

**Disadvantages:**

1. It is only a view layer, you have still to plug your code for Ajax requests, events and so on. Some people get surprised by that.
2. The library itself is pretty large.
3. The learning curve can be steep.

# 3. What is JSX?

JSX is a syntax notation for JavaScript XML(XML-like syntax extension to ECMAScript). It stands for JavaScript XML. It provides expressiveness of JavaScript along with HTML like template syntax. For example, the below text inside h1 tag return as javascript function to the render function

render() {  
 return (  
 <div>  
 <h1> Welcome to React world!!</h1>  
 </div>  
 );  
}

# 4. What are the differences between Props and State?

Both props and state are plain JavaScript objects. While both of them hold information that influences the output of render, they are different in their functionality with respect to component. Props get passed to the component similar to function parameters whereas state is managed within the component similar to variables declared within a function.

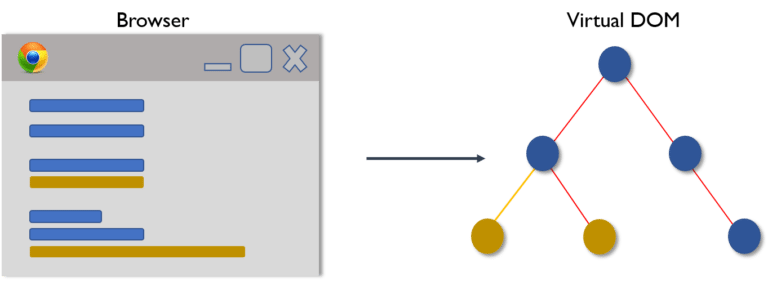
# 5. What are some lifecycle methods in a React component?

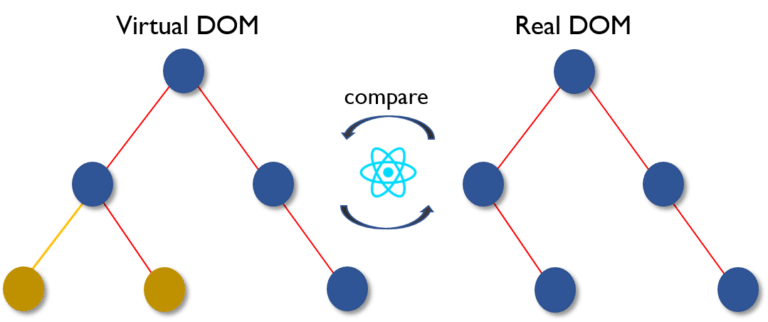
They are componentDidMount, componentWillMount, componentWillUpdate, componentDidUpdate, shouldComponentUpdate and componentWilReceiveProps.

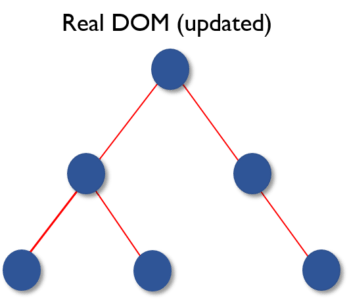


# 6. What is virtual DOM?

The virtual DOM (VDOM) is an in-memory representation of Real DOM. The representation of a UI is kept in memory and synced with the “real” DOM. It’s a step that happens between the render function being called and the displaying of elements on the screen. .







# 7. What are the differences between Stateless Components and Stateful Components?

**Stateless Components**: If the behavior is independent of its state then it can be a stateless component.You can use either a function or a class for creating stateless components. But unless you need to use a lifecycle hook in your components, you should go for stateless functional components. There are a lot of benefits if you decide to use stateless functional components: they are easy to write, understand, and test, and you can avoid the this keyword altogether.

**Stateful Components**: If the behavior of a component is dependent on the state of the component then it can be termed as stateful component.These Stateful components are always class components and have a state that gets initialized in the constructor.

# 8. What is Babel?

**Babel is a JavaScript compiler:**Babel is a toolchain that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript in current and older browsers or environments. Here are the main things Babel can do for you:

* Transform syntax
* Polyfill features that are missing in your target environment (through @babel/polyfill)
* Source code transformations
* And more! (check out these videos for inspiration)

// Babel Input: ES2015 arrow function   
[1, 2, 3].map(n => n + 1);// Babel Output: ES5 equivalent   
[1, 2, 3].map(function(n) {   
 return n + 1;  
});

# 9. What is Higher Order Component?

## From Higher-Order-Functions to Higher-Order-Components:

**Higher-order-functions** are defined as functions that accept a function as an argument

const map = (fn, array) => {  
 const mappedArray = [] for (let i = 0; i < array.length; i++) {  
 mappedArray.push(  
 // apply fn with the current element of the array  
 fn(array[i])  
 )  
 } return mappedArray  
}const square = (x) => x \* xconsole.log(map(square, [1, 2, 3, 4, 5]))  
//=> [ 1, 4, 9, 16, 25 ]

**Higher-order-component** (HOC) is a function that takes a component and returns a new component.

// A HOC to transforms it’s input to uppercase:  
const yell = (PassedComponent) =>  
 ({ children, ...props }) =>  
 <PassedComponent {...props}>  
 {children.toUpperCase()}!  
 </PassedComponent>const Title = (props) => <h1>{props.children}</h1>  
const AngryTitle = yell(Title)<AngryTitle>Whatever</AngryTitle>  
// => <h1>WHATEVER!</h1>

# 10. What areECMAScript, ES1, 2, 3, 4, 5, 6, 7, 8,…n? ES5 vs ES6?

**ECMAScript** is a standard. While **JavaScript**is the most popular implementation of that standard. JavaScript implements ECMAScript and builds on top of it.

**ES**is simply short for **ECMAScript**. Every time you see **ES** followed by a number, it is referencing an edition of ECMAScript. There are eight editions of ECMAScript published:

ES1, ES2, ES3, ES4, ES5, ES6 (ES2015), ES7 (ES2016), ES8(ES2017)

**ES5 VS ES6:**

**Arrow Function:**

ES5:

function greetings(name) {  
 return 'hello ' + name  
}

ES6:

const greetings = name => {  
 return `hello ${name}`;  
}

**Manipulating objects:**

ES5:

var obj1 = { a: 1, b: 2 }  
var obj2 = { a: 2, c: 3, d: 4}  
var obj3 = Object.assign(obj1, obj2)

ES6:

const obj1 = { a: 1, b: 2 }  
const obj2 = { a: 2, c: 3, d: 4}  
const obj3 = {...obj1, ...obj2}

ES5:

var obj1 = { a: 1, b: 2, c: 3, d: 4 }  
var a = obj1.a  
var b = obj1.b  
var c = obj1.c  
var d = obj1.d

ES6:

const obj1 = { a: 1, b: 2, c: 3, d: 4 }  
const {  
 a,  
 b,  
 c,  
 d  
} = obj1

ES5:

var a = 1  
var b = 2  
var c = 3  
var d = 4var obj1 = { a: a, b: b, c: c, d: d }

ES6:

var a = 1  
var b = 2  
var c = 3  
var d = 4var obj1 = { a, b, c, d }

**Promises vs Callbacks:**

ES5:

function isGreater (a, b, callback) {  
 var greater = false  
 if(a > b) {  
 greater = true  
 }  
 callback(greater)}isGreater(1, 2, function (result) {  
 if(result) {  
 console.log('greater');  
 } else {  
 console.log('smaller')  
 }  
})

ES6:

const isGreater = (a, b) => {  
 return new Promise ((resolve, reject) => {  
 if(a > b) {  
 resolve(true)  
 } else {  
 reject(false)  
 }  
 })  
}isGreater(1, 2)  
 .then(result => {  
 console.log('greater')  
 })  
 .catch(result => {  
 console.log('smaller')  
 })

**Exporting & Importing Modules:**

ES5:

var myModule = { x: 1, y: function() { console.log('This is ES5') }}module.exports = myModule;

ES6:

const myModule = { x: 1, y: () => { console.log('This is ES5') }}export default myModule;

ES5:

var myModule = require('./myModule');

ES6:

import myModule from './myModule';