**JS and React Interview Questions**

* **Synchronous** – Everything will go in the sync, so the page will take more time to load. Will block all the instructions until the task is completed.
* **Async**- It will never wait for each operation to complete, instead will complete all the operations in one go.
* **Spread Operator**- It essentially takes either an object or an array and expands into its set of items. Let’s us essentially drop an array and get its values.
* **Rest Operator**- Collects the rest of list of the arguments into an array. When occurs in a function call, it’s called a spread operator.
* **Slice()-** returns the selected element in an array as a new array object. Original array will not be changed.
* **Splice()-** add/removes item to/from an array and returns the removed items . This method changes the original array.
* **. append ()-** insert the specified content at the end of the selected elements.
* **. call ()-** a predefined JS method, an object can use a method belonging to another object.
* **Event capturing-** the element is first captured by the outermost element and propagate to the inner elements.
* **Event binding-** allows you to add an event handler for a specified event so that your chosen JS function will be invoked when that event is triggered for the associated DOM element.
* **Event bubbling-** the event is captured and handled by the innermost element and propagate to the outer elements.
* **Event delegation-** Mechanism of responding to UI events via a single common parent rather than each child, through the magic of event “bubbling”.
* **Singleton-** an object which can only be instantiated one time. By using this, one can ensure they don’t accidentally create two users in a single user application.
* **Promise-** Completion or failure of async operation and its resulting value. Have two possible values either resolve or reject. .then() is used to pass handlers which can take the resolved or rejected value. Provide better error handling than callbacks.
* **Difference between promise and observable:**

Promise is like an asynchronous get operation whereas observable is like an asynchronous iterable.

Main difference is that promise emits only a single value whereas observable emits multiple values.

Often observable is preferred over promise because it provides features of Promise and more.

* **Callback hell-** Any code where the use of callbacks in async code becomes difficult to follow. Also known as pyramid of doom. It consists of multiple nested callbacks which makes code hard to read and debug.
* **Shallow copy-** A bit-wise copy of an object. A new object is created that has an exact copy of the values in the original object. If any of the fields of an object are references to other objects, just the reference addresses are copied i.e. only memory address is copied.
* **Deep copy-** A deep copy copies all fields and make copies of dynamically allocated memory pointed to by the fields. It occurs when an object is copied along with the object to which it refers.
* **Debounce functions-** included in many JS libraries. The goal behind each implementation is to reduce overhead by preventing a function from being called several times in succession.

The general idea is:

1. Start with no timeout.
2. If the produced function is called clear and reset the timeout.
3. If the timeout is hit call the original function.

* **Fetch()-** Promises are often used for network requests. Used to load the information about the user from the remote server. Method is quite complex and has many optional parameters, but basic usage is quite simple.

**Let promise= fetch(url);**

* **Set-** To remove duplicates from an array.
* **Reduce()-** reduces the array to a single value. Executes a provided function for each value of the array(from left-to-right). Return value of the function is stored in an accumulator(result/total).
* **Map()-** creates an array by calling a specific function on each element present in the parent array. It is a non-mutating method. Used to iterate over an array and calling function on every element of an array.
* **Filter()-** creates an array filled with all array elements that pass a test. Does not execute the function for array elements without values. Does not change the original array.
* **Curring-** Helps to avoid passing the same variable again and again. Technique of evaluating function with multiple arguments into sequence of function with single argument.
* **Call-** will invoke the function immediately and modify the context
* **Apply-** apply usually works when it has the second argument as an array.
* **Bind-** returns the function that later executed, will returns the accurate context for calling the original function

**React**

* **State:** Owned locally and updated by the component itself. Component itself will update the state using setState function. State is mutable. Cannot be accessed from child components. Managed in the view controller.
* **Prop:** Owned by a parent component and are read only in the child component that receives them. Props are immutable. Are basically kind of global variable or object. Can only be updated if a callback function is passed to child to trigger the upstream change.
* **Functional Component-** A plain JS which accepts props as an argument and returns a React element. Absence of this keyword. Provide solution without using the state. Also called dumb or presentational or stateless components. Mainly responsible for the UI.
* **Class Component-** Requires you to extend from React.Component and creates a Render function which returns React element. More feature rich. Can work with each other. Can pass data from one class component to another. Manage their own private data. Provide lifecycle hooks. Stateful/Smart/Container.
* **Stateless Component-** When component just receives props and renders them to the page is a stateless component, for which a pure function can be used. These are also called dumb or presentational components.
* **Synthetic events-** Cross browser wrapper around the browser’s native event. Basically, are event objects that are created as opposed to real events.
* **Super(props)-** used to call base class constructor, to get properties from parent class.
* **React router-** allows us to build a single web page application with navigation without the page refreshing as the user navigates. Uses component structure to call the components.
* **Redux-** The basic idea of redux is that the entire application state is kept in a single store. The store is simply a JS object. The only way to change state is by firing actions from your application and then writing reducers for these actions that modify the state.
* **Redux-thunk-** is a middleware that allows us to write action creators that return a function instead of an action. The thunk can be used to display the dispatch of an action if a certain condition is met.
* **Store-** In which the state of the application lives.
* **Reducers-** used to modify the store, only way to trigger reducers is to dispatch an action.
* **Providers-** A redux component. Purpose is to provide store to its child components.
* **Connect-** to map the stores state and dispatch to the props of the component. mapStateToProps and mapDispatchToprops are both pure functions that are provided the stores state and dispatch resp.
* **Actions-** A message that we send to our central redux store. Are payloads of information that send data from your application to the store. **Store.dispatch().**
* The inner function receives the store methods dispatch and getState as parameters.
* To enable Redux Thunk, we need to use **applyMiddleware()**.
* **Lifecycles-** Mounting, Updating and Unmounting

**Mounting-** putting elements into the DOM. Has four built-in methods that gets called, in the order below:

* Constructor()
* getDerivedStateFromProps()
* render()
* componentDidMount()

**ReactJS Interview Questions**

* What is React?

React is an open source JS library created by Facebook for building complex, interactive UI’s in web and mobile applications. Follows unidirectional flow and involves one-way data binding. Renders only parts of the page that need it.

Built with separate chunks of code that can communicate with each other.

* Why do we use redux for react?

When using redux with react, states will no longer need to be lifted-up, hence it makes it easier for us to trace which action cause any change. So, the component does not need to provide any state or method for its children components to share data among themselves. Everything is handled by Redux.

* How react works? How virtual DOM works in react?

React creates a virtual DOM.

**First step: “Diffing”**

When state changes in a component it first runs a **diffing** algorithm, which identifies what has changed in a virtual DOM.

**Second step: “Reconciliation”**

Updates the DOM with the results of the diff.

The virtual DOM is an abstraction of the HTML DOM. It is light weight and detached from the browser specific implementation details.

**React Elements-** lives in the virtual DOM. They make the basic nodes here. Once we defined the elements, React Elements can be render into the real DOM.

* What is diffing in React?

When the state of a component changes so that an element needs to be changed from one type to another, React unmounts the whole tree and builds a new one from the scratch. This causes every node in that tree to be destroyed.

* What is JSX?
* JSX is a syntax extension to JS.
* JSX produces React elements.
* We can embed only JavaScript expression in JSX by wrapping it in curly braces.
* After compilation, JSX expressions become regular JS objects.
* What is render() in react?

Process of transforming react components into DOM that our browser can understand and display it on the screen.

* What are components in react?

Core building blocks of react. UI broken down into multiple individual pieces called components and work on them independently ad merge them all in a parent component which will be our final UI.

* What are axios in react?

A promise based and take advantage of async and await. For more readable async code, easy to use API.

* What is React.createClasss?

It allows us to create component 'classes’. But with ES6, React allows us to implement component classes that used ES6 JS classes.

When using a React’s createclass() method, we pass in an object as an argument.

Using ES6 class to write the same component is a little different. Instead of using the method from the library, we extend an ES6 class that the library defines, **Component**.

* What is constructor in JS?

Constructor() is a special function in a JS class. JS invokes constructor() whenever an object is created via a class.

* What is ReactDOM and what is the difference between the ReactDOM and React?

Prior to v0.14, all ReactDOM functionality was part of React.

But later, React and ReactDOM were split into two different libraries.

ReactDOM is the glue between React and the DOM.

Often, we use it only for one single thing: mounting with the ReactDOM.

Another feature of ReactDOM is **ReactDOM.findDOMNode()** which we can use to gain direct access to a DOM element.

For everything else, we use React, to define and create our elements, for lifecycle hooks etc.

* What are the differences between class and functional component?
* Difference between state and props?
* How to use setState in JS?

Only place we can directly write to this.state should be the components constructor. setState is **asynchronous**. This allows us to have multiple calls to setState in a single scope.

setState accepts a function as its parameter, if we pass a function as the first argument of setState , React will call it with the at-call-time-current state and expect you to return an Object to merge into state.

* What are refs in React?

According to React documents, refs are used to get reference to a DOM(Document Object Model)node or an instance of a component in a React application.

* What are **controlled and uncontrolled components** in React?

**Controlled Components**

A form element whose value is controlled by React is called controlled component.

In HTML, form elements such as <input>, <textarea> and <select> typically maintain their own state and update it based on the user input. But with react, it works differently. The component containing the form will keep track of the value of the input in its state and will re-render the component each time the callback function

**Uncontrolled Components**

Form data is handled by the DOM itself. To write an uncontrolled component, instead of writing an event handler for every state update, we can use a ref to get form values from the DOM.

* What is a higher order component?

An advanced technique in react for reusing component logic. They are not part of the react API. A pattern that emerges from the React’s compositional nature.

It is a function that takes a component and returns a new component. They are common in third-party react libraries. The most common is probably Redux’s connect function. These are the best way to share behavior between react components.

* What is create-react-app?

It is the official CLI(Command Line Interface) for react to create react apps with no build configuration.

We don’t need to install or configure tools like Webpack or Babel. They are preconfigured and hidden so that we can focus on the code. We can install easily just like any other node modules.

**Command:** create-react-app my-app

* What is pure component?

A react component can be considered pure if it renders the same output for the same state and props. Basically, it takes an input value and depending on that input produce an output value. For class components, React provides the PureComponent base class. Class components that extend the React.pureComponent class are treated as pure components.

* What is Redux?
* Redux is a predictable state container for JS apps.
* Redux is a library for JS applications.
* Redux is not tied to React, can be used with Angular, Vue or Vanilla JS.
* It is a state container, stores the state of the application.