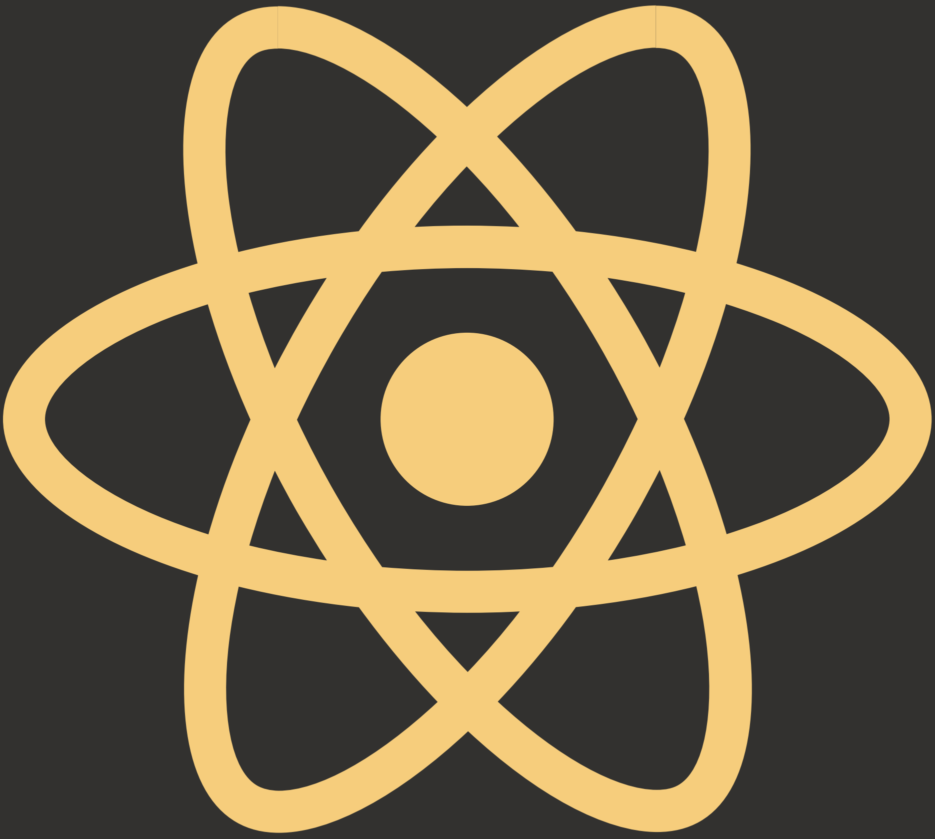
Special Offer!

70+ JavaScript & ReactJS Interview Questions



This Book includes:

- 10+ Resume Templates
- 100+ Recently Funded Startups List
- Official Discord Server for career guidance

Table Of Contents

JavaScript Important Interview Questions....4
- Basic JavaScript Interview Questions....5
  - What is JavaScript?....5
  - What is the difference between Java & JavaScript?....5
  - What are the data types supported by JavaScript?....5
  - Explain some features of JavaScript....5
  - Why do we use the word “debugger” in JavaScript?....6
  - Explain Hoisting in JavaScript....6
  - What are the different ways to declare functions in javascript?....7
  - Can you pass arrow function as parameter to another function?....8
  - What are the scopes of a variable in JavaScript?....9
  - Difference between “ == “ and “ === “ operators....10
  - Difference between var and let keyword in javascript....11
- Intermediate JavaScript Interview Questions....13
  - What is the purpose of ‘This’ operator in JavaScript?....13
  - Explain call(), apply() and, bind() methods....13
  - What is Callback?....14
  - What are the drawbacks of Callbacks?....15
  - What is the difference between exec () and test () methods in javascript?....16
  - What is the difference between Local storage & Session storage?....17
  - What is an event bubbling in JavaScript?....17
  - What is NaN in JavaScript?....18
  - What will be the output of 40 + 10 + “30”?....18
  - Explain Closures in JavaScript....18
- Advanced JavaScript Interview Questions....20
  - What do mean by prototype design pattern?....20
  - What is recursion in a programming language?....20
  - What is memoization?....21
  - What is currying in JavaScript?....22
  - What are the benefits of currying?....23
  - What is the use of promises in javascript?....23
  - Explain different states in a Promise....24
  - What are Async/Await in JavaScript?....25
  - What is a Temporal Dead Zone?....26
- JavaScript Coding Round Questions....27
  - Given a string, reverse each word in the sentence....27
  - Check if an object is an array or not....27
  - FizzBuzz Challenge....28
  - Given two strings, return true if they are anagrams of one another....28

Use Closure to Create a private counter....29
Perform right rotations on an array.... 29
Find missing number in O(n) time....30
Remove duplicates from Array....31
Largest difference between two elements in an Array.... 32
Isomorphic Strings.... 33
Balanced Parentheses....34
Enqueue and Dequeue operations using Stacks....35
Convert Decimal to binary....35
Recursive Binary Search.... 36
Guess the Output 1....36
Guess the Output 2....37
Guess the Output 3....37
Guess the Output 4....38
Reactjs Important Interview Questions.... 39
Basic Reactjs Interview Questions.... 40
What is React?....40
What are the major features of React?....40
What is Virtual DOM?.... 40
How Virtual DOM works?....40
What is JSX?.... 42
What is the difference between Element and Component?....43
What is Class Component in Reactjs?....43
What are the functional components in Reactjs?....44
When to use a Class Component over a Function Component?....45
What is state in React?....45
Why should we not update the state directly?.... 46
What are props in React?.... 46
What is props drilling in reactjs?.... 47
Intermediate Reactjs Interview Questions.... 49
What are Pure Components?.... 49
What is the purpose of callback function as an argument of setState()?....50
What are synthetic events in React?.... 50
What is the use of refs?.... 51
What is the difference between Shadow DOM and Virtual DOM?....52
What is React Fiber?.... 52
What are controlled components?.... 52
What are uncontrolled components?.... 53
What are the different phases of component lifecycle?.... 53
What are Higher-Order Components?....54
Advanced Reactjs Interview Questions.... 55

How to create props proxy for HOC component?... 55
What is context?.. 56
What is Lazy loading in Reactjs?... 57
Is lazy function supports named exports?...58
What is reconciliation?.. 59
What are React Mixins?...59
What is Shallow Renderer in React testing?...59
What are the advantages of Jest over Jasmine?..60
What is flux?.. 60
What is Redux?..61
What are the core principles of Redux?...61
What are the downsides of Redux compared to Flux?... 61
Can I dispatch an action in reducer?.. 62
How to access Redux store outside a component?..62
What is MVW pattern in ReactJs?.. 62
What are the drawbacks of MVW pattern?.. 63

JavaScript Important Interview Questions

JavaScript Coding Round Questions

Given a string, reverse each word in the sentence

Problem:
For example “Welcome to this Javascript Guide!” should be become “emocleW ot siht tpircsavaJ !ediuG”

Solution:
```
var string = "Welcome to this Javascript Guide!";

// Output becomes !ediuG tpircsavaJ siht ot emocleW
var reverseEntireSentence = reverseBySeparator(string, "");

// Output becomes emocleW ot siht tpircsavaJ !ediuG
var reverseEachWord = reverseBySeparator(reverseEntireSentence, " ");

function reverseBySeparator(string, separator) {
  return string.split(separator).reverse().join(separator);
}
```

Check if an object is an array or not

Problem:
Write a program to check whether a given object is array or not.

Solution:
In JavaScript, you can check if an object is an array or not using the Array.isArray() method or the instanceof operator.

1. Using Array.isArray() method:
   The Array.isArray() method is a built-in static method available in modern JavaScript environments. It returns true if the provided value is an array, and false otherwise. For eg:

   ```
   const myArray = [1, 2, 3];
   console.log(Array.isArray(myArray)); // Output: true

   const myObject = { name: "John", age: 30 };
   console.log(Array.isArray(myObject)); // Output: false
   ```

2. Using instanceof operator:

The instanceof operator checks if an object is an instance of a particular constructor function. While this approach can work, it may have issues when working with arrays from different frames or if the array is created using Object.create(null). For eg:

```
const myArray = [1, 2, 3];
console.log(myArray instanceof Array); // Output: true

const myObject = { name: "John", age: 30 };
console.log(myObject instanceof Array); // Output: false
```

FizzBuzz Challenge

Problem:
Create a for loop that iterates up to 100 while outputting "fizz" at multiples of 3, "buzz" at multiples of 5 and "fizzbuzz" at multiples of 3 and 5.

Solution:

```
for (let i = 1; i <= 100; i++) {
  let f = i % 3 == 0,
    b = i % 5 == 0;
  console.log(f ? (b ? 'FizzBuzz' : 'Fizz') : b ? 'Buzz' : i);
}
```

Given two strings, return true if they are anagrams of one another

Problem:
Return true if “Mary” is an anagram of “Army”

Solution:

```
var firstWord = "Mary";
var secondWord = "Army";

isAnagram(firstWord, secondWord); // true

function isAnagram(first, second) {
  // For case insensitivity, change both words to lowercase.
  var a = first.toLowerCase();
  var b = second.toLowerCase();

  // Sort the strings, and join the resulting array to a string. Compare the results
  a = a.split("").sort().join("");
  b = b.split("").sort().join("");
  return a === b;
}
```

Use Closure to Create a private counter

Problem:
Create a closure to create a private counter

Solution:
```
function counter() {
  var _counter = 0;
  // return an object with several functions that allow you
  // to modify the private _counter variable
  return {
    add: function(increment) { _counter += increment; },
    retrieve: function() { return 'The counter is currently at: ' + _counter; }
  }
}

// error if we try to access the private variable like below
// _counter;

// usage of our counter function
var c = counter();
c.add(5);
c.add(9);

// now we can access the private variable in the following way
c.retrieve(); // => The counter is currently at: 14
```

Perform right rotations on an array

Problem:
Implement a function that returns an updated array with r right rotations on an array of integers a.
For eg:
Given the following array: [2,3,4,5,7]
Perform 3 right rotations:
First rotation : [7,2,3,4,5] , Second rotation : [5,7,2,3,4] and, Third rotation: [4,5,7,2,3]

return [4,5,7,2,3]

Reactjs Important Interview Questions

What is reconciliation?

Ans. Reconciliation is the process through which React updates the Browser DOM and makes React work faster. React use a diffing algorithm so that component updates are predictable and faster. React would first calculate the difference between the real DOM and the copy of DOM (Virtual DOM) when there's an update of components. React stores a copy of Browser DOM which is called Virtual DOM. When we make changes or add data, React creates a new Virtual DOM and compares it with the previous one. This comparison is done by Diffing Algorithm. Now React compares the Virtual DOM with Real DOM. It finds out the changed nodes and updates only the changed nodes in Real DOM leaving the rest nodes as it is. This process is called Reconciliation.

What are React Mixins?

Ans. Mixins are a way to totally separate components to have a common functionality. Mixins should not be used and can be replaced with higher-order components or decorators.

One of the most commonly used mixins is PureRenderMixin. You might be using it in some components to prevent unnecessary re-renders when the props and state are shallowly equal to the previous props and state:

```
const PureRenderMixin = require("react-addons-pure-render-mixin");

const Button = React.createClass({
  mixins: [PureRenderMixin],
  // ...
});
```

What is Shallow Renderer in React testing?

Ans. Shallow rendering is useful for writing unit test cases in React. It lets you render a component one level deep and assert facts about what its render method returns, without worrying about the behavior of child components, which are not instantiated or rendered.

For example, if you have the following component:

```
function MyComponent() {
  return (
    <div>
      <span className={"heading"}>{"Title"}</span>
      <span className={"description"}>{"Description"}</span>
    </div>
  );
}
```

Then you can assert as follows:

```
import ShallowRenderer from "react-test-renderer/shallow";

// in your test
const renderer = new ShallowRenderer();
renderer.render(<MyComponent />);

const result = renderer.getRenderOutput();

expect(result.type).toBe("div");
expect(result.props.children).toEqual([
  <span className={"heading"}>{"Title"}</span>,
  <span className={"description"}>{"Description"}</span>,
]);
```

What are the advantages of Jest over Jasmine?

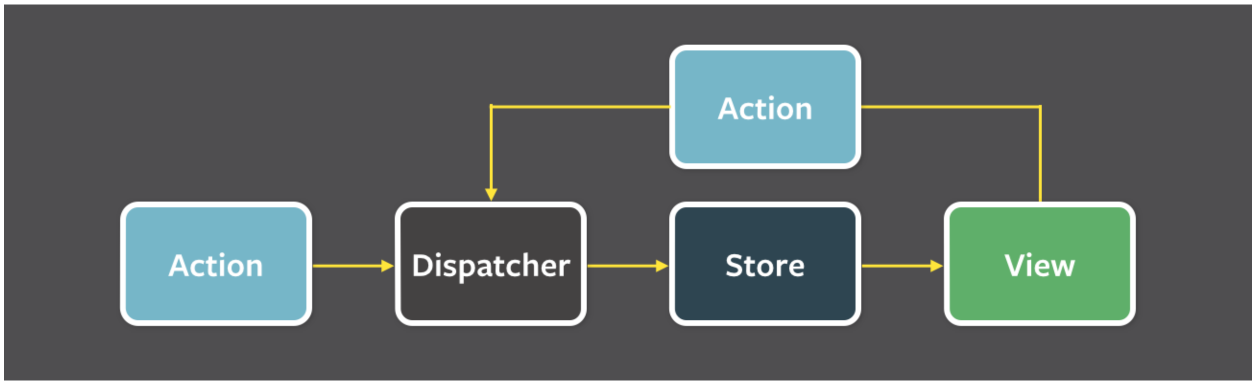
Ans. There are a couple of advantages compared to Jasmine:

- Automatically finds tests to execute in your source code.
- Automatically mocks dependencies when running your tests.
- Allows you to test asynchronous code synchronously.
- Runs your tests with a fake DOM implementation (via jsdom) so that your tests can be run on the command line.
- Runs tests in parallel processes so that they finish sooner.

What is flux?

Ans. Flux is an application design paradigm used as a replacement for the more traditional MVC pattern. It is not a framework or a library but a new kind of architecture that complements React and the concept of Unidirectional Data Flow. Facebook uses this pattern internally when working with React.

The workflow between dispatcher, stores and views components with distinct inputs and outputs as follows:



What is Redux?

Ans. Redux is a predictable state container for JavaScript apps based on the Flux design pattern. Redux can be used together with React, or with any other view library. It is tiny (about 2kB) and has no dependencies.

What are the core principles of Redux?

Ans. Redux follows three fundamental principles:

- **Single source of truth**: The state of your whole application is stored in an object tree within a single store. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.
- **State is read-only**: The only way to change the state is to emit an action, an object describing what happened. This ensures that neither the views nor the network callbacks will ever write directly to the state.
- **Changes are made with pure functions**: To specify how the state tree is transformed by actions, you write reducers. Reducers are just pure functions that take the previous state and an action as parameters, and return the next state.

What are the downsides of Redux compared to Flux?

Ans. Instead of saying downsides we can say that there are few compromises of using Redux over Flux. Those are as follows:

- **You will need to learn to avoid mutations**: Flux is un-opinionated about mutating data, but Redux doesn't like mutations and many packages complementary to Redux assume you never mutate the state. You can enforce this with dev-only packages like redux-immutable-state-invariant, Immutable.js, or instructing your team to write non-mutating code.