**Differentiate between Real DOM and Virtual DOM.**

|  |  |
| --- | --- |
| Real DOM vs Virtual DOM | |
| **Real DOM** | **Virtual  DOM** |
| 1. It updates slow. | 1. It updates faster. |
| 2. Can directly update HTML. | 2. Can’t directly update HTML. |
| 3. Creates a new DOM if element updates. | 3. Updates the JSX if element updates. |
| 4. DOM manipulation is very expensive. | 4. DOM manipulation is very easy. |
| 5. Too much of memory wastage. | 5. No memory wastage. |

### ****3. What are the features of React?****

Major features of React are listed below:

1. It uses the **virtual DOM** instead of the real DOM.
2. It uses **server-side rendering**.
3. It follows **uni-directional data flow** or data binding.

**4. List some of the major advantages of React.**

Some of the major advantages of React are:

1. It increases the application’s performance
2. It can be conveniently used on the client as well as server side
3. Because of JSX, code’s readability increases
4. React is easy to integrate with other frameworks like Meteor, Angular, etc
5. Using React, writing UI test cases become extremely easy

**5. What are the limitations of React?**

Limitations of React are listed below:

1. React is just a library, not a full-blown framework
2. Its library is very large and takes time to understand
3. It can be little difficult for the novice programmers to understand
4. Coding gets complex as it uses inline templating and JSX

**6. What is JSX?**

JSX is a shorthand for JavaScript XML. This is a type of file used by React which utilizes the expressiveness of JavaScript along with HTML like template syntax. This makes the HTML file really easy to understand. This file makes applications robust and boosts its performance. Below is an example of JSX:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | render(){      return(    <div>    <h1> Hello World from Edureka!!</h1>             </div>        );  } |

**7. What do you understand by Virtual DOM? Explain its working.**

A virtual DOM is a lightweight JavaScript object which originally is just the copy of the real DOM. It is a node tree that lists the elements, their attributes and content as Objects and their properties. React’s render function creates a node tree out of the React components. It then updates this tree in response to the mutations in the data model which is caused by various actions done by the user or by the system.  
This Virtual DOM works in three simple steps.

1. Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation.
2. Then the difference between the previous DOM representation and the new one is calculated.
3. Once the calculations are done, the real DOM will be updated with only the things that have actually changed. 

**8. Why can’t browsers read JSX?**

Browsers can only read JavaScript objects but JSX in not a regular JavaScript object. Thus to enable a browser to read JSX, first, we need to transform JSX file into a JavaScript object using JSX transformers like Babel and then pass it to the browser.

### ****12. Explain the purpose of render() in React.****

Each React component must have a **render()**mandatorily. It returns a single React element which is the representation of the native DOM component. If more than one HTML element needs to be rendered, then they must be grouped together inside one enclosing tag such as **<form>, <group>,<div>** etc. This function must be kept pure i.e., it must return the same result each time it is invoked.

### ****13. How can you embed two or more components into one?****

We can embed components into one in the following way:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | class MyComponent extends React.Component{      render(){          return(    <div>    <h1>Hello</h1>                    <Header/>              </div>            );      }  }  class Header extends React.Component{      render(){          return    <h1>Header Component</h1>       };  }  ReactDOM.render(      <MyComponent/>, document.getElementById('content')  ); |

### ****14. What is Props?****

Props is the shorthand for Properties in React. They are read-only components which must be kept pure i.e. immutable. They are always passed down from the parent to the child components throughout the application. A child component can never send a prop back to the parent component. This help in maintaining the unidirectional data flow and are generally used to render the dynamically generated data.

### ****15. What is a state in React and how is it used?****

States are the heart of React components. States are the source of data and must be kept as simple as possible. Basically, states are the objects which determine components rendering and behavior. They are mutable unlike the props and create dynamic and interactive components. They are accessed via **this.state().**

### ****16. Differentiate between states and props.****

|  |  |  |
| --- | --- | --- |
| States vs Props | | |
| **Conditions** | **State** | **Props** |
| 1. Receive initial value from parent component | Yes | Yes |
| 2. Parent component can change value | No | Yes |
| 3. Set default values inside component | Yes | Yes |
| 4. Changes inside component | Yes | No |
| 5. Set initial value for child components | Yes | Yes |
| 6. Changes inside child components | No | Yes |

### ****17. How can you update the state of a component?****

State of a component can be updated using this.setState().

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | class MyComponent extends React.Component {      constructor() {          super();          this.state = {              name: 'Maxx',              id: '101'          }      }      render()          {              setTimeout(()=>{this.setState({name:'Jaeha', id:'222'})},2000)              return (    <div>    <h1>Hello {this.state.name}</h1>    <h2>Your Id is {this.state.id}</h2>                       </div>                );          }      }  ReactDOM.render(      <MyComponent/>, document.getElementById('content')  ); |

### ****18. What is arrow function in React? How is it used?****

Arrow functions are more of brief syntax for writing the function expression. They are also called ‘fat arrow‘ (**=>**) the functions. These functions allow to bind the context of the components properly since in ES6 auto binding is not available by default. Arrow functions are mostly useful while working with the higher order functions.

### ****19. Differentiate between stateful and stateless components.****

|  |  |
| --- | --- |
| Stateful vs Stateless | |
| **Stateful Component** | **Stateless Component** |
| 1. Stores info about component’s state change in memory | 1. Calculates the internal state of the components |
| 2. Have authority to change state | 2. Do not have the authority to change state |
| 3. Contains the knowledge of past, current and possible future changes in state | 3. Contains no knowledge of past, current and possible future state changes |
| 4. Stateless components notify them about the requirement of the state change, then they send down the props to them. | 4. They receive the props from the Stateful components and treat them as callback functions. |

**20. What are the different phases of React component’s lifecycle?**

There are three different phases of React component’s lifecycle:

1. *Initial Rendering Phase:* This is the phase when the component is about to start its life journey and make its way to the DOM.
2. *Updating Phase:*Once the component gets added to the DOM, it can potentially update and re-render only when a prop or state change occurs. That happens only in this phase.
3. *Unmounting Phase:*This is the final phase of a component’s life cycle in which the component is destroyed and removed from the DOM.

**21. Explain the lifecycle methods of React components in detail.**

Some of the most important lifecycle methods are:

1. ***componentWillMount()***–Executed just before rendering takes place both on the client as well as server-side.
2. ***componentDidMount()***–Executed on the client side only after the first render.
3. ***componentWillReceiveProps()***– Invoked as soon as the props are received from the parent class and before another render is called.
4. ***shouldComponentUpdate()***–Returns true or false value based on certain conditions. If you want your component to update, return **true** else return **false**. By default, it returns false.
5. ***componentWillUpdate()***– Called just before rendering takes place in the DOM.
6. ***componentDidUpdate()***–Called immediately after rendering takes place.
7. ***componentWillUnmount()***– Called after the component is unmounted from the DOM. It is used to clear up the memory spaces.

**22. What is an event in React?**

In React, events are the triggered reactions to specific actions like mouse hover, mouse click, key press, etc. Handling these events are similar to handling events in DOM elements. But there are some syntactical differences like:

1. Events are named using camel case instead of just using the lowercase.
2. Events are passed as functions instead of strings.

The event argument contains a set of properties, which are specific to an event. Each event type contains its own properties and behavior which can be accessed via its event handler only.

**23. How do you create an event in React?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | class Display extends React.Component({      show(evt) {          // code      },      render() {          // Render the div with an onClick prop (value is a function)          return (    <div onClick={this.show}>Click Me!</div>            );      }  }); |

**24. What are synthetic events in React?**

Synthetic events are the objects which act as a cross-browser wrapper around the browser’s native event. They combine the behavior of different browsers into one API. This is done to make sure that the events show consistent properties across different browsers.

**25. What do you understand by refs in React?**

Refs is the short hand for References in React. It is an attribute which helps to store a reference to a particular React element or component, which will be returned by the components render configuration function. It is used to return references to a particular element or component returned by render(). They come in handy when we need DOM measurements or to add methods to the components.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | class ReferenceDemo extends React.Component{       display() {           const name = this.inputDemo.value;           document.getElementById('disp').innerHTML = name;       }  render() {      return(    <div>              Name: <input type="text" ref={input => this.inputDemo = input} />              <button name="Click" onClick={this.display}>Click</button>    <h2>Hello <span id="disp"></span> !!!</h2>          </div>      );     }   } |

**26. List some of the cases when you should use Refs.**

Following are the cases when refs should be used:

* When you need to manage focus, select text or media playback
* To trigger imperative animations
* Integrate with third-party DOM libraries

**27. How do you modularize code in React?**

We can modularize code by using the export and import properties. They help in writing the components separately in different files.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | //ChildComponent.jsx  export default class ChildComponent extends React.Component {      render() {          return(    <div>    <h1>This is a child component</h1>               </div>            );      }  }    //ParentComponent.jsx  import ChildComponent from './childcomponent.js';  class ParentComponent extends React.Component {      render() {          return(    <div>                  <App />              </div>            );      }  } |

**28. How** **are forms created in React?**

React forms are similar to HTML forms. But in React, the state is contained in the state property of the component and is only updated via setState(). Thus the elements can’t directly update their state and their submission is handled by a JavaScript function. This function has full access to the data that is entered by the user into a form.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | handleSubmit(event) {      alert('A name was submitted: ' + this.state.value);      event.preventDefault();  }    render() {      return (    <form onSubmit={this.handleSubmit}>              <label>                  Name:                  <input type="text" value={this.state.value} onChange={this.handleSubmit} />              </label>              <input type="submit" value="Submit" />          </form>        );  } |

**29. What do you know about controlled and uncontrolled components?**

|  |  |
| --- | --- |
| Controlled vs Uncontrolled Components | |
| **Controlled Components** | **Uncontrolled Components** |
| 1. They do not maintain their own state | 1. They maintain their own state |
| 2. Data is controlled by the parent component | 2. Data is controlled by the DOM |
| 3. They take in the current values through props and then notify the changes via callbacks | 3. Refs are used to get their current values |

**30. What are Higher Order Components(HOC)?**

Higher Order Component is an advanced way of reusing the component logic. Basically, it’s a pattern that is derived from React’s compositional nature. HOC are custom components which wrap another component within it. They can accept any dynamically provided child component but they won’t modify or copy any behavior from their input components. You can say that HOC are ‘pure’ components.

**31. What can you do with HOC?**

HOC can be used for many tasks like:

* Code reuse, logic and bootstrap abstraction
* Render High jacking
* State abstraction and manipulation
* Props manipulation

**32. What are Pure Components?**

*Pure*components are the simplest and fastest components which can be written. They can replace any component which only has a **render().**These components enhance the simplicity of the code and performance of the application.

**33. What is the significance of keys in React?**

Keys are used for identifying unique Virtual DOM Elements with their corresponding data driving the UI. They help React to optimize the rendering by recycling all the existing elements in the DOM. These keys must be a unique number or string, using which React just reorders the elements instead of re-rendering them. This leads to increase in application’s performance.

**React Redux – React Interview Questions**

**34. What were the major problems with MVC framework?**

Following are some of the major problems with MVC framework:

* DOM manipulation was very expensive
* Applications were slow and inefficient
* There was huge memory wastage
* Because of circular dependencies, a complicated model was created around models and views

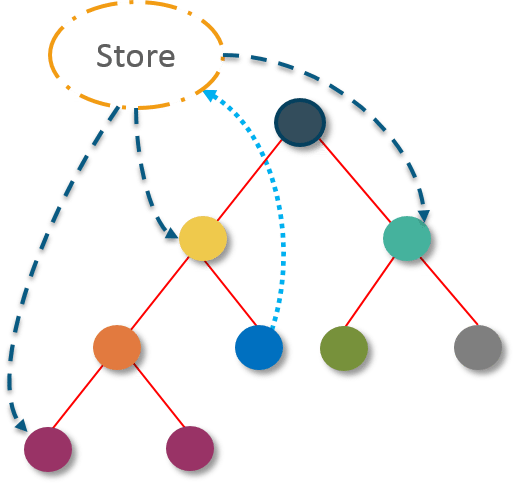
**35. Explain Flux.**

Flux is an architectural pattern which enforces the uni-directional data flow. It controls derived data and enables communication between multiple components using a central Store which has authority for all data. Any update in data throughout the application must occur here only. Flux provides stability to the application and reduces run-time errors.

**36.** **What is Redux?**

Redux is one of the hottest libraries for front-end development in today’s marketplace. It is a predictable state container for JavaScript applications and is used for the entire applications state management. Applications developed with Redux are easy to test and can run in different environments showing consistent behavior.

**37. What are the three principles that Redux follows?**

1. ***Single source of truth:***The state of the entire application is stored in an object/ state 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.
2. ***State is read-only:***The only way to change the state is to trigger an action. An action is a plain JS object describing the change. Just like state is the minimal representation of data, the action is the minimal representation of the change to that data.
3. ***Changes are made with pure functions:*** In order to specify how the state tree is transformed by actions, you need pure functions. Pure functions are those whose return value depends solely on the values of their arguments.

**38. What do you understand by “Single source of truth”?**

Redux uses ‘Store’ for storing the application’s entire state at one place. So all the component’s state are stored in the Store and they receive updates from the Store itself. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.

**39. List down the components of Redux.**

Redux is composed of the following components:

1. **Action** – It’s an object that describes what happened.
2. **Reducer**–  It is a place to determine how the state will change.
3. **Store** – State/ Object tree of the entire application is saved in the Store.
4. **View** – Simply displays the data provided by the Store.

### ****47. Why**** ****is switch keyword used in React Router v4?****

Although a **<div>** is used to encapsulate multiple routes inside the Router. The ‘switch’ keyword is used when you want to display only a single route to be rendered amongst the several defined routes. The **<switch>**tag when in use matches the typed URL with the defined routes in sequential order. When the first match is found, it renders the specified route. Thereby bypassing the remaining routes.

**48. Why do we need a Router in React?**

A Router is used to define multiple routes and when a user types a specific URL, if this URL matches the path of any ‘route’ defined inside the router, then the user is redirected to that particular route. So basically, we need to add a Router library to our app that allows creating multiple routes with each leading to us a unique view.

|  |  |
| --- | --- |
| 1  2  3  4  5 | <switch>      <route exact path=’/’ component={Home}/>      <route path=’/posts/:id’ component={Newpost}/>      <route path=’/posts’   component={Post}/>  </switch> |

**49. List down the advantages of React Router.**

Few advantages are:

1. Just like how React is based on components, in React Router v4, the API is *‘All About Components’*. A Router can be visualized as a single root component (**<BrowserRouter>**) in which we enclose the specific child routes (**<route>**).
2. No need to manually set History value: In React Router v4, all we need to do is wrap our routes within the **<BrowserRouter>** component.
3. The packages are split: Three packages one each for Web, Native and Core. This supports the compact size of our application. It is easy to switch over based on a similar coding style.

**50. How is React Router different from conventional routing?**

|  |  |  |
| --- | --- | --- |
| Conventional Routing vs React Routing | | |
| **Topic** | **Conventional Routing** | **React Routing** |
| **PAGES INVOLVED** | Each view corresponds to a new file | Only single HTML page is involved |
| **URL CHANGES** | A HTTP request is sent to a server and corresponding HTML page is received | Only the History attribute is changed |
| **FEEL** | User actually navigates across different pages for each view | User is duped thinking he is navigating across different pages |

#### 4. What do you mean by Props and State in React?

**Answer:**  
This is the basic ReactJs Interview Questions asked in an interview. Props mean the properties of the argument which is passed in javascript function. A state is used for creating a dynamic and interactive component.

#### 5. What are the refs in React?

**Answer:**  
For focus management, trigger animation we use refs in React. It also contains the third party libraries.

#### 6. What is the difference between ReactJs and AngularJs?

**Answer:**  
Both [ReactJs and AngularJs](https://www.educba.com/reactjs-vs-angularjs/) are a very powerful and diverse approach for a front-end web application. It supports all small medium and large enterprise application. One side the ReactJs application SEO friendly, simple and easy to understand and the other side Angular Js applications are easy to develop and supports a different form of testing.

#### 8. What do you mean a functional component in React?

**Answer:**  
Functional component is those components which returns react elements as an element.

#### 9. What do you mean by reacting routing?

**Answer:**  
In React Routing only single HTML page will be involved.

#### 11. What are the lifecycle methods of React Components in detail?

**Answer:**  
Some of the most important lifecycles methods are given below:  
componentWillMount()  
componentDidMount()  
componentWillRecieveProps()  
shouldComponentUpdate()  
componentWillUpdate()

#### 12.What are the lifecycle of ReactJs?

**Answer:**  
Initialization, State/Property updates, Destruction are the lifecycle of ReactJs.

Let us move to the next ReactJs Interview Questions

#### 13. What are the advantages of ReactJs?

**Answer:**  
Application performance will be increased.  
It can be used on both Client and Server side.  
Because of JSX code will become or reliable.  
Testing will become easy.

#### 14. Why we use ReactJs?

**Answer:**  
This is the frequently asked ReactJs Interview Questions in an interview. For handling the view part of the mobile application we use React.

#### 17. What is the basic difference between pros and state?

**Answer:**  
The basic difference is: State is mutable and Pros are immutable.

#### 18. When you will use the class component over a functional component?

**Answer:**  
When your component carrying a state or lifecycle then we will use the Class component.

### ****5. What is the difference between createElement and cloneElement?****

**A:** This is one of the quite logical React interview questions and answers. Basically, as the name suggests, createElement what React uses to create React Elements, cloneElement, on the other hand, is used to clone an element and pass it new props. Might seem quite obvious, but it is still one of the top 50 interview questions and answers.

### ****15. What is render() in React? Explain its purpose.****

**A:** We should start by explaining that each React component must have a render() because it returns a single React element which is the representation of the native DOM component. If more than one HTML element needs to be rendered, then they must be grouped together inside one enclosing tag such as <form>, <group>, <div>etc. This function must be kept pure and return the same result each time it is invoked.

### ****18. What is the difference between DOM and virtual DOM in React.js?****

**A:** DOM aka [Document Object Model](https://www.w3.org/TR/WD-DOM/introduction.html) is an abstraction of structured code (HTML). Dom and HTML code are interrelated as the elements of HTML are known as nodes of DOM. It defines a structure where users modify the content present in the structure in any way they want (create, edit, alter, modify etc.). Basically, HTML is a text, DOM is an in-memory representation of this text.

Virtual DOM is a representation of DOM objects like a lightweight copy. It is used and provided for free by React.js

### ****19. What are the controlled components and uncontrolled components in React?****

**A:** When answering React interview questions, you should know that the form data is handled by the React components.

A controlled input accepts values as props and callbacks to change that value. The uncontrolled component, on the other hand, is a substitute for controlled components. In these cases, DOM itself handles the form data.

### ****20. What’s the difference between an Element and a Component in React?****

**A:** React element is an object representation of some UI. Basically, it describes what you want to see on the screen. A React component, on the other hand, is a function or a class that optionally accepts input and returns a React element. This is also one of the common interview questions on ReactJS.

### ****21. Explain the difference between functional and class components.****

**A:** The components that return React elements as a result are called functional components. They are basically just simple JavaScript functions. They, however, haven’t been around for long. In fact, they have been introduced with React 0.14.

Class components, on the other hand, have been around for quite some time. They use plain Java objects for creating pages. With the React’s create-a-class-factory method, a literal is passed in defining the methods of a new component.

### ****23. What is the use of the arrow function in React?****

**A:** The arrow function is something you should be familiar when preparing for React interview questions because this function is very important for React operations. It allows you to predict the behavior of bugs when passed as callbacks hence it prevents bugs caused by this altogether.

Top of Form

**24. List some of the cases when you should use Refs.**

**A:**Refs should be used in the following cases:

* When you need to manage focus, select text or media playback
* To trigger imperative animations
* Integrate with third-party DOM libraries

**25. What are Pure Components?**

**A:** Pure components are the simplest and fastest components which can replace any component with only a render(). Pure components enhance the simplicity of the code and performance of the application. Consider learning more about them when preparing for React native interview questions.

**26. What is the significance of keys in React?**

**A:** When you want to identify the unique virtual DOM elements with their corresponding data driving the UI – that is where the keys come in. By recycling all the existing elements in the DOM, the keys help React optimize the rendering. They allow React to reorder the elements instead of re-rendering them, which increases the app’s performance. Definitely add the keys on your list of ReactJS interview questions and answers.

### ****28. Describe how events are handled in React.****

**A:** The event handlers in React will be passed instances of [SyntheticEvent](https://reactjs.org/docs/events.html) to solve cross-browser compatibility issues. As we mentioned earlier, SyntheticEvent is React’s cross-browser wrapper around the browser’s native event. The synthetic events have the same interface as the native ones but they work identically across all browsers.

However, React doesn’t actually attach events to the child nodes themselves. Instead, it uses a single event listener in order to listen to all events at the top level which. Not only is this great for the performance but it also means that React doesn’t have to keep track of the event listeners when updating the DOM.

### ****30. Is setState() async? Why?****

**A:** setState() actions are indeed asynchronous. setState() doesn’t immediately mutate this.state. Instead, it creates a pending state transition. Accessing this.state after calling this method can potentially return the existing value. There is no guarantee of synchronous operation of calls to setState and calls may be batched for performance gains.

The reason behind is the way setState alters the state and causes rerendering. Making it synchronous might leave the browser unresponsive. That being said, the setState calls are asynchronous as well as batched for better UI experience and performance. Keep this in mind as this is definitely among the most popular 50 interview questions and answers when it comes to React.

### 32. What is Context?

Context enables passing data through the component tree without having to pass props down manually at every level. Which means you can deep-nest items without issue.

It is primarily used when some data needs to be accessible by many components at different nesting levels.

**34. What are synthetic events in React?**

**A:** When talking about React js interview questions and answers, we should mention the synthetic events. They act as a cross-browser wrapper around the browser’s native event thus combining the behavior of different browsers into one API. Their purpose is to ensure that the events show consistent properties across different browsers.

**35. What can you do with HOC?**

**A:** You can use HOC for:

* Code reuse, logic and bootstrap abstraction
* Render hijacking
* State abstraction and manipulation
* Props manipulation

**How else can the JavaScript code below be written using Node.Js to produce the same output?**

console.log("first");

setTimeout(function() {

console.log("second");

}, 0);

console.log("third");

Output:

first

third

second

In Node.js version 0.10 or higher, setImmediate(fn) will be used in place of setTimeout(fn,0) since it is faster. As such, the code can be written as follows:

console.log("first");

setImmediate(function(){

console.log("second");

});

console.log("third");

Why is Node.js Single-threaded?

Node.js is single-threaded for async processing. By doing async processing on a single-thread under typical web loads, more performance and scalability can be achieved as opposed to the typical thread-based implementation.

### Explain callback in Node.js.

A callback function is called after a given task. It allows other code to be run in the meantime and prevents any blocking.  Being an asynchronous platform, Node.js heavily relies on callback. All APIs of Node are written to support callbacks.

### Explain the role of REPL in Node.js.

As the name suggests, REPL (Read Eval Print Loop) performs the tasks of – Read, Evaluate, Print and Loop. The REPL in Node.js is used to execute ad-hoc Javascript statements. The REPL shell allows entry to javascript directly into a shell prompt and evaluates the results. For the purpose of testing, debugging, or experimenting, REPL is very critical.

### Name the types of API functions in Node.js.

There are two types of functions in Node.js.:

* + **Blocking functions** - In a blocking operation, all other code is blocked from executing until an I/O event that is being waited on occurs. Blocking functions execute synchronously.

For example:  
const fs = require('fs');  
const data = fs.readFileSync('/file.md'); // blocks here until file is read  
console.log(data);  
// moreWork(); will run after console.log  
  
The second line of code blocks the execution of additional JavaScript until the entire file is read. moreWork () will only be called after Console.log

* + **Non-blocking functions** - In a non-blocking operation, multiple I/O calls can be performed without the execution of the program being halted. Non-blocking functions execute asynchronously.

For example:

const fs = require('fs');  
fs.readFile('/file.md', (err, data) => {  
  if (err) throw err;  
  console.log(data);  
});  
// moreWork(); will run before console.log

Since fs.readFile () is non-blocking, moreWork () does not have to wait for the file read to complete before being called. This allows for higher throughput.

1. Which is the first argument typically passed to a Node.js callback handler?

Typically, the first argument to any callback handler is an optional error object. The argument is null or undefined if there is no error.

Error handling by a typical callback handler could be as follows:

function callback(err, results) {  
    // usually we'll check for the error before handling results  
    if(err) {  
        // handle error somehow and return  
    }  
    // no error, perform standard callback handling  
}

1. What are the functionalities of NPM in Node.js?

NPM (Node Package Manager) provides two functionalities:

* 1. An online repository for Node.js packages.
  2. Command-line utility for installing packages, version management and dependency management of Node.js packages.

1. What is the difference between Node.js and Ajax?

Node.js and Ajax (Asynchronous JavaScript and XML) are the advanced implementations of JavaScript. They all serve entirely different purposes.

Ajax is primarily designed for dynamically updating a particular section of a page’s content, without having to update the entire page.

Node.js is used for developing client-server applications.

1. Explain chaining in Node.js.

Chaining is a mechanism whereby the output of one stream is connected to another stream creating a chain of multiple stream operations.

### ****1. Differentiate between JavaScript and Node.js.****

|  |  |  |
| --- | --- | --- |
| **Features** | **JavaScript** | **Node.js** |
| Type | Programming Language | Interpreter and environment for JavaScript |
| Utility | Used for any client-side activity for a web application | Used for accessing or performing any non-blocking operation of any operating system |
| Running Engine | Spider monkey (FireFox), JavaScript Core (Safari), V8 (Google Chrome), etc. | V8 (Google Chrome) |

### ****List down the major benefits of using Node.js?****

|  |  |
| --- | --- |
| **Features** | **Description** |
| ***Fast*** | Node.js is built on Google Chrome’s V8 JavaScript Engine which makes its library very fast in code execution |
| ***Asynchronous*** | Node.js based server never waits for an API to return data thus making it asynchronous |
| ***Scalable*** | It is highly scalable because of its event mechanism which helps the server to respond in a non-blocking way |
| ***Open Source*** | Node.js has an extensive open source community which has contributed in producing some excellent modules to add additional capabilities to Node.js applications |
| ***No Buffering*** | Node.js applications simply output the data in chunks and never buffer any data |

**8. How many types of API functions are there in Node.js?**

There are two types of API functions in Node.js:

* Asynchronous, non-blocking functions
* Synchronous, blocking functions

**What is the difference between Asynchronous and Non-blocking?**

|  |  |
| --- | --- |
| **Asynchronous** | **Non-blocking** |
| Asynchronous means not synchronous. Using these we can make asynchronous HTTP requests that do not wait for the server to respond. These functions continue to respond to the request for which it has already received the server response. | Non-blocking functions are used in regards with I/O operations. They immediately respond with whatever data is available and keeps on running as per the requests. In case, any answer couldn’t be retrieved then the API returns immediately with an error. |

### ****11. What do you understand by Event-driven programming?****

Event-driven programming is a programming approach that heavily makes use of events for triggering various functions. An event can be anything like a mouse click, key press, etc. When an event occurs, a call back function is executed that is already registered with the element. This approach mainly follows the publish-subscribe pattern. Because of [event-driven programming](https://www.edureka.co/blog/nodejs-tutorial/#events), Node.js is faster when compared to other technologies.

### ****13. Explain  REPL in the context of Node.js.****

REPL in Node.js stands for **R**ead, **E**val, **P**rint, and **L**oop. It represents a computer environment such as a window console or Unix/Linux shell where any command can be entered and then the system can respond with an output. Node.js comes bundled with a REPL environment by default. REPL can perform the below-listed tasks:

* **Read:** Reads the user’s input, parses it into JavaScript data-structure and then stores it in the memory.
* **Eval:** Receives and evaluates the data structure.
* **Print:**Prints the final result.
* **Loop:** Loops the provided command until *CTRL+C* is pressed twice.

### ****14. List down the tasks which should be done asynchronously using the event loop?****

Below is the list of the tasks which must be done asynchronously using the event loop:

* I/O operations
* Heavy computation
* Anything requiring blocking

### ****15. List down the steps using which “Control Flow” controls the function calls in Node.js?****

1. Control the order of execution
2. Collect data
3. Limit concurrency
4. Call the next step in the program

### [2) Explain CLI in Node.js?](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled2)

CLI stands for Command Line Interface. It is a utility or program on your computer where users type commands to perform some action or run some script rather than clicking on the screen.  
There are different types of command line interfaces depending on which operating system you are using. We have listed some of them below.

* [**9) Explain What is NPM ?**](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled9)
* NPM stands for node package manager. It is default Package Manager for JavaScript programming language. NPM is used for installing/updating packages and modules of Javascript.

[**10) Explain Modules in Node Js ?**](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled10)

**Modules** are reusable block of code whose existence does not impact other code in any way. It is not supported by Javascript. Modules are introduced in ES6. Modules are important for Maintainability, Reusability, and Namespacing of Code.

[**12) For what require() is used in Node Js ?**](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled12)

**require()** is used to include modules from external files in Node Js. It is the easiest way to include a module in Node. Basically require is a function that takes a string parameter which contains the location of the file that you want to include. It reads the entire javascript file, executes the file, and then proceeds to return the exports object.  
Syntax:

require('path')

[**14) Is Node Js Single-threaded ?**](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled14)

Yes, Node Js is single threaded to perform asynchronous processing. Doing async processing on a single thread could provide more performance and scalability under typical web loads than the typical thread-based implementation.

[**17) How to create a simple server in Node js that returns Hello World ?**](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled17)

By writing following line of code, you can **create a server in Node Js**.

var http =require('http');

http.createServer(function(req,res){

res.writeHead(200,{'Content-Type':'text/plain'});

res.end('Hello World\n');

}).listen(1320,'127.0.0.3');

### [22) List some features of Express JS.](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled22)

Some of the main features of Express JS are listed below: –

* It is used for setting up middlewares so as to provide a response to the HTTP or RESTful requests.
* With the help of express JS, the routing table can be defined for performing various HTTP operations.
* It is also used for dynamically rendering [HTML](https://www.onlineinterviewquestions.com/html-interview-questions/)pages which are based on passing arguments to the templates.
* It provides each and every feature which is provided by core Node JS.
* The performance of Express JS is adequate due to the presence of a thin layer prepared by the Express JS.
* It is used for organizing the web applications into the MVC architecture.
* Everything from routes to rendering view and performing HTTP requests can be managed by Express JS.

### [23) Write the steps for setting up an Express JS application.](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled23)

Following are the steps used to set up an express JS application: –

1. A folder with the same name as the project name is created.
2. A file named package.json is created inside the folder created.
3. “npm install” command is run on the command prompt. It installs all the libraries present in package.json.
4. A file named server.js is created.
5. “Router” file is created inside the package which consists of a folder named index.js.
6. “App” is created inside the package which has the index.html file.

This way, an express JS application is set up.

### [24) What do you mean by Express JS?](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled24)

Express JS is an application framework which is light-weighted node JS. A number of flexible, useful and important features are provided by this JavaScript framework for the development of mobile as well as web applications with the help of node JS.

[**26) What is the use of Express JS?**](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled26)

Express.js is a lightweight web application which helps in organizing the web application into MVC architecture on the server side.

### [27) What function are arguments available to Express JS route handlers?](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled27)

The arguments which are available to an Express JS route handler-function are-

* Req – the request object
* Res – the response object
* Next (optional) – a function which is used to pass control to one of the subsequent route handlers.

The third argument is optional and may be omitted, but in some cases, it is useful where there is a chain of handlers and control can be passed to one of the subsequent route handlers skipping the current one.

### [28) How to config properties in Express JS?](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled28)

In Express JS, there are two ways for configuring the properties:

* With process.ENV:
* A file with the name “.env” is to be created inside the project folder.
* All the properties are to be added in the “.env” file.
* Any of the properties can be used in server.js.
* With require JS:
* A file with the name “config.json” is to be created in the config folder inside the project folder.
* The config properties are to be added in the config.json file.
* Now, require should be used to access the config.json file.
* [**30) How to authenticate users in express JS?**](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled30)
* Since authentication is an opinionated area which is not ventured by express JS, therefore any authentication scheme can be used in express JS for the authentication of users.

[**31) Which template engine is supported by express JS?**](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled31)

Express JS supports any template engine that conforms to the (path, locals, callback) signature.

### [33) Why to use Express.js?](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled33)

Below are the few reasons why to use Express with Node.js

* Express js is built on top of Node.js. It is the perfect framework for ultra-fast Input / Output.
* Cross Platform
* Support MVC Design pattern
* Support of NoSQL databases out of the box.
* Multiple templating engine support i.e. Jade or EJS which reduces the amount of HTML code you have to write for a page.
* Support Middleware, basic web-server creation, and easy routing tools.

### [35) List types of Http requests?](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled35)

Http defines a set of request methods to perform the desired actions. These request methods are:

1. **GET**: The GET method asked for the representation of the specifies resource. This request used to retrieve the data.
2. **POST**: The POST technique is utilized to present an element to the predetermined resource, generally causing a change in state or reactions on the server.
3. **HEAD**: The HEAD method is similar to the GET method but asks for the response without the response body.
4. **PUT**: This method is used to substitute all current representations with the payload.
5. **DELETE**: It is used to delete the predetermined resource.
6. **CONNECT**: This request is used to settle the TCP/IP tunnel to the server by the target resource
7. **OPTION**: This method is used to give back the HTTP strategies to communicate with the target resource.
8. **TRACE**: This method echoes the message which tells the customer how much progressions have been made by an intermediate server.
9. **PATCH**: The PATCH method gives partial modifications to a resource.

### [36) What is difference between put and patch?](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled36)

The main difference between the put and the patch is

|  |  |
| --- | --- |
| Put | Patch |
| The embedded entity is believed to the modified version of the resources that are deposited on the original server. It is requested to the client to replace the stored is substituted. | In this, the information regarding the way of modifying the original server which has the resources to produce a new version is found. |
| At the time of updating the resource, you need to forward full payload as the request. | At the time of updating the resource, you only need to send the parameter of the resource which you want to update. |

[**39) Write a simple code to enable CORS in Node js?**](https://www.onlineinterviewquestions.com/node-js-interview-questions/#collapseUnfiled39)

**CORS** stands for Cross-Origin Resource Sharing. It a is a mechanism that uses additional HTTP headers to tell a browser to let a web application running at one origin (domain) have permission to access selected resources from a server at a different origin.

**Use below code to enable CORS on NodeJS**

app.use(function(req, res, next) {

res.header("Access-Control-Allow-Origin", "\*");

res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

next();

});

**24)   Mention the framework most commonly used in node.js?**

“Express” is the most common framework used in node.js

**15)   What are the two arguments that async.queue takes?**

The two arguments that async.queue takes

a)      Task function

b)      Concurrency value

**16)   What is an event loop in Node.js ?**

To process and handle external events and to convert them into callback invocations an event loop is used. So, at I/O calls, node.js can switch from one request to another .

**17)   Mention the steps by which you can async in Node.js?**

By following steps you can async Node.js

a)      First class functions

b)      Function composition

c)       Callback Counters

d)      Event loops

**18)    What are the pros and cons of Node.js?**

**Pros:**

a)      If your application does not have any CPU intensive computation, you can build it in Javascript top to bottom, even down to the database level if you use JSON storage object DB like MongoDB.

b)      Crawlers receive a full-rendered HTML response, which is far more SEO friendly rather than a single page application or a websockets app run on top of Node.js.

**Cons:**

a)       Any intensive CPU computation will block node.js responsiveness, so a threaded platform is a better approach.  
b)      Using relational database with Node.js is considered less favourable

**19)   How Node.js overcomes the problem of blocking of I/O operations?**

Node.js solves this problem by putting the event based model at its core, using an event loop instead of threads.

**20)   What is the difference between Node.js vs Ajax?**

The difference between Node.js and Ajax is that, Ajax (short for Asynchronous Javascript and XML) is a client side technology, often used for updating the contents of the page without refreshing it. While,Node.js is Server Side Javascript, used for developing server software. Node.js does not execute in the browser but by the server.

**21)   What are the Challenges with Node.js ?**

Emphasizing on the technical side, it’s a bit of challenge in Node.js to have one process with one thread to scale up on multi core server.

**22)** **What does it mean “non-blocking” in node.js?**

In node.js “non-blocking” means that its IO is non-blocking.  Node uses “libuv” to handle its IO in a platform-agnostic way. On windows, it uses completion ports for unix it uses epoll or kqueue etc. So, it makes a non-blocking request and upon a request, it queues it within the event loop which call the JavaScript ‘callback’ on the main JavaScript thread.

**23)   What is the command that is used in node.js to import external libraries?**

Command “require” is used for importing external libraries, for example, “var http=require (“http”)”.  This will load the http library and the single exported object through the http variable.

**10)   Why Node.js is single threaded**?

For async processing, Node.js was created explicitly as an experiment. It is believed that more performance and scalability can be achieved by doing async processing on a single thread under typical web loads than the typical thread based implementation.

**11)   Does node run on windows?**

Yes – it does. Download the MSI installer from <http://nodejs.org/download/>

**12)   Can you access DOM in node?**

No, you cannot access DOM in node.

**13)   Using the event loop what are the tasks that should be done asynchronously?**

a)      I/O operations

b)      Heavy computation

c)       Anything requiring blocking

**What does event-driven programming mean?**

In computer programming, event driven programming is a programming paradigm in which the flow of the program is determined by events like messages from other programs or threads. It is an application architecture technique divided into two sections 1) Event Selection 2) Event Handling