**JavaScript**

1. What do you understand about JavaScript?

JavaScript is a popular web scripting language and is used for client-side and server-side development. The JavaScript code can be inserted into HTML pages that can be understood and executed by web browsers while also supporting object-oriented programming abilities.

### 2. What are the various data types that exist in JavaScript?

These are the different types of data that JavaScript supports:

* Boolean - For true and false values
* Null - For empty or unknown values
* Undefined - For variables that are only declared and not defined or initialized
* Number - For integer and floating-point numbers
* String - For characters and alphanumeric values
* Object - For collections or complex values
* Symbols - For unique identifiers for objects

### 3. What are the features of JavaScript?

These are the features of JavaScript:

* Lightweight, interpreted programming language
* Cross-platform compatible
* Open-source
* Object-oriented
* Integration with other backend and frontend technologies
* Used especially for the development of network-based applications

### 4. What are the advantages of JavaScript over other web technologies?

These are the advantages of JavaScript:

#### **Enhanced Interaction**

JavaScript adds interaction to otherwise static web pages and makes them react to users’ inputs.

#### **Quick Feedback**

There is no need for a web page to reload when running JavaScript. For example, form input validation.

#### **Rich User Interface**

JavaScript helps in making the UI of web applications look and feel much better.

#### **Frameworks**

JavaScript has countless frameworks and libraries that are extensively used for developing web applications and games of all kinds.

5. What are some of the built-in methods in JavaScript?

|  |  |
| --- | --- |
| Built-in Method | Values |
| Date() | Returns the present date and time |
| concat() | Joins two strings and returns the new string |
| push() | Adds an item to an array |
| pop() | Removes and also returns the last element of an array |
| round() | Rounds of the value to the nearest integer and then returns it |
| length() | Returns the length of a string |

### 6. What are the scopes of a variable in JavaScript?

The scope of a variable implies where the variable has been declared or defined in a JavaScript program. There are two scopes of a variable:

#### **Global Scope**

Global variables, having global scope are available everywhere in a JavaScript code.

#### **Local Scope**

Local variables are accessible only within a function in which they are defined.

### 7. What is the ‘this’ keyword in JavaScript?

The ‘this’ keyword in JavaScript refers to the currently calling object. It is commonly used in constructors to assign values to object properties.

### 8. What are the conventions of naming a variable in JavaScript?

Following are the naming conventions for a variable in JavaScript:

* Variable names cannot be similar to that of reserved keywords. For example, var, let, const, etc.
* Variable names cannot begin with a numeric value. They must only begin with a letter or an underscore character.
* Variable names are case-sensitive.

### 09. What is Callback in JavaScript?

A callback is a JavaScript function that is passed to another function as an argument or a parameter. This function is to be executed whenever the function that it is passed to gets executed.

### 10. What are the ways of adding JavaScript code in an HTML file?

There are primarily two ways of embedding JavaScript code:

* We can write JavaScript code within the script tag in the same HTML file; this is suitable when we need just a few lines of scripting within a web page.
* We can import a JavaScript source file into an HTML document; this adds all scripting capabilities to a web page without cluttering the code.

### 11. What’s the difference between let and var?

Both let and var are used for variable and method declarations in JavaScript. So, there isn’t much of a difference between these two besides that while var keyword is scoped by function, the let keyword is scoped by a block.

### 12. What are the arrow functions in JavaScript?

Arrow functions are a short and concise way of writing functions in JavaScript. The general syntax of an arrow function is as below:

const helloWorld = () => {

  console.log("hello world!");

};

### 13. What are the different ways an HTML element can be accessed in a JavaScript code?

Here are the ways an HTML element can be accessed in a JavaScript code:

* getElementByClass(‘classname’): Gets all the HTML elements that have the specified classname.
* getElementById(‘idname’): Gets an HTML element by its ID name.
* getElementbyTagName(‘tagname’): Gets all the HTML elements that have the specified tagname.
* querySelector(): Takes CSS style selector and returns the first selected HTML element.

### 14. What are the ways of defining a variable in JavaScript?

There are three ways of defining a variable in JavaScript:

#### **Var**

This is used to declare a variable and the value can be changed at a later time within the JavaScript code.

#### **Const**

We can also use this to declare/define a variable but the value, as the name implies, is constant throughout the JavaScript program and cannot be modified at a later time.

#### **Let**

This mostly implies that the values can be changed at a later time within the JavaScript code.

### 15. What are some of the JavaScript frameworks and their uses?

JavaScript has a collection of many frameworks that aim towards fulfilling the different aspects of the web application development process. Some of the prominent frameworks are:

* React - Frontend development of a web application
* Angular - Frontend development of a web application
* Node - Backend or server-side development of a web application
* 16. What is the difference between Undefined and Undeclared in JavaScript?

|  |  |
| --- | --- |
| Undefined | Undeclared |
| Undefined means a variable has been declared but a value has not yet been assigned to that variable. | Variables that are not declared or that do not exist in a program or application. |

* 17. What is the difference between Undefined and Null in JavaScript?

|  |  |
| --- | --- |
| Undefined | Null |
| Undefined means a variable has been declared but a value has not yet been assigned to that variable. | Null is an assignment value that we can assign to any variable that is meant to contain no value. |

* 18. What is the difference between Session storage and Local storage?

|  |  |
| --- | --- |
| Session storage | Local storage |
| The data stored in session storage gets expired or deleted when a page session ends. | Websites store some data in local machine to reduce loading time; this data does not get deleted at the end of a browsing session. |

**React**

**1. What is React?**

* React is a front-end JavaScript library developed by Facebook in 2011.
* It follows the component-based approach which helps in building reusable UI components.
* It is used for developing complex and interactive web and mobile UI.
* Even though it was open sourced only in 2015, it has one of the largest communities supporting it.

### ****2. 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.

**3. 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

**4. 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

### **5. What is JSX?**

JSX is a syntax extension of JavaScript. It is used with React to describe what the user interface should look like. By using JSX, we can write HTML structures in the same file that contains JavaScript code.

### **6. Can web browsers read JSX directly?**

* Web browsers cannot read JSX directly. This is because they are built to only read regular JS objects and JSX is not a regular JavaScript object
* For a web browser to read a JSX file, the file needs to be transformed into a regular JavaScript object. For this, we use Babel

### **7. What is the virtual DOM?**

React keeps a lightweight representation of the real DOM in the memory, and that is known as the virtual DOM. When the state of an object changes, the virtual DOM changes only that object in the real DOM, rather than updating all the objects.

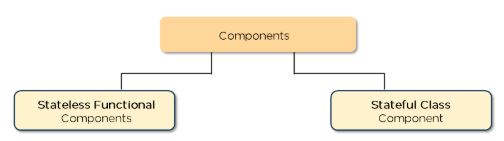
### ****8. How is React different from Angular?****

|  |  |  |
| --- | --- | --- |
| **React vs Angular** | | |
| **TOPIC** | **REACT** | **ANGULAR** |
| 1. ARCHITECTURE | Only the View of MVC | Complete MVC |
| 2. RENDERING | Server-side rendering | Client-side rendering |
| 3. DOM | Uses virtual DOM | Uses real DOM |
| 4. DATA BINDING | One-way data binding | Two-way data binding |
| 5. DEBUGGING | Compile time debugging | Runtime debugging |
| 6. AUTHOR | Facebook | Google |

### **9. What are the components in React?**

Components are the building blocks of any React application, and a single app usually consists of multiple components. A component is essentially a piece of the user interface. It splits the user interface into independent, reusable parts that can be processed separately.

There are two types of components in React:



* **Functional Components:**These types of components have no state of their own and only contain render methods, and therefore are also called **stateless components**. They may derive data from other components as props (properties).

|  |
| --- |
| function Greeting(props) {    return <h1>Welcome to {props.name}</h1>;  } |

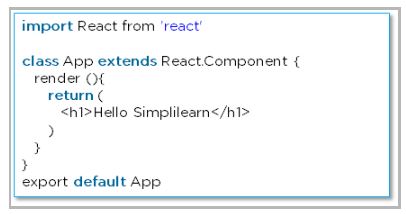
* **Class Components:**Class components are more complex than functional components. It requires you to extend from React. Component and create a render function which returns a React element. You can pass data from one class to other class components.

The class component is also known as a stateful component because they can hold or manage local state.

|  |
| --- |
| class Greeting extends React.Component {    render() {      return <h1>Welcome to {this.props.name}</h1>;    }  } |

### **10. What is the use of render() in React?**

* It is required for each component to have a render() function. This function returns the HTML, which is to be displayed in the component.
* If you need to render more than one element, all of the elements must be inside one parent tag like <div>, <form>.



### ****13. What is Props?****

* Props are short for Properties. It is a React built-in object that stores the value of attributes of a tag and works similarly to HTML attributes.
* Props provide a way to pass data from one component to another component. Props are passed to the component in the same way as arguments are passed in a function.

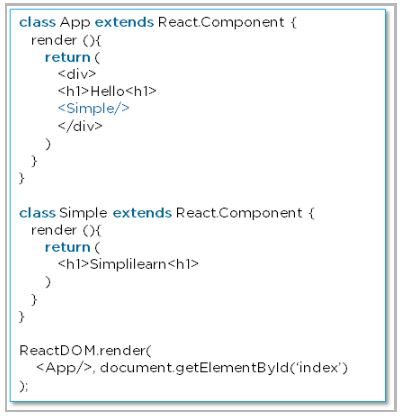
### **12. What is a state in React?**

* The state is a built-in React object that is used to contain data or information about the component. The state in a component can change over time, and whenever it changes, the component re-renders.
* The change in state can happen as a response to user action or system-generated events. It determines the behavior of the component and how it will render.
* 13. What are the differences between state and props?

|  |  |  |
| --- | --- | --- |
|  | State | Props |
| Use | Holds information about the components | Allows to pass data from one component to other components as an argument |
| Mutability | Is mutable | Are immutable |
| Read-Only | Can be changed | Are read-only |
| Child components | Child components cannot access | Child component can access |
| Stateless components | Cannot have state | Can have props |

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

We can embed two or more components into one using this method:



### **15. What is an arrow function and how is it used in React?**

* An arrow function is a short way of writing a function to React.
* It is unnecessary to bind **‘this’**inside the constructor when using an arrow function. This prevents bugs caused by the use of **‘this’**in React callbacks.

Arrow functions are mostly useful while working with the higher order functions.

### ****16. 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. |

### ****17. 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.

### ****18. 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.

### ****19. 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.

**20. 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>            );      }  }); |

### **21. What are synthetic events in React?**

* Synthetic events combine the response of different browser's native events into one API, ensuring that the events are consistent across different browsers.

### ****22. How**** ****are forms created in React?****

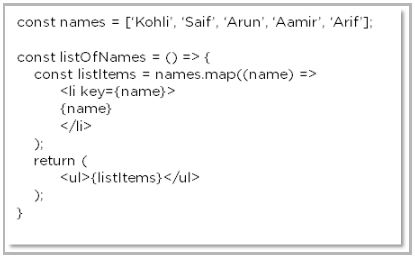
React forms are similar to HTML forms. But in React, the state is contained in the state property of the component and is only updated via setState(). Thus, the elements can’t directly update their state and their submission is handled by a JavaScript function. This function has full access to the data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

### **23. How do you update the state of a component?**

We can update the state of a component by using the built-in **‘setState()’**method:

### **24. Explain how lists work in React**

* We create lists in React as we do in regular JavaScript. Lists display data in an ordered format
* The traversal of lists is done using the map() function



### **25. Why is there a need for using keys in Lists?**

Keys are very important in lists for the following reasons:

* A key is a unique identifier, and it is used to identify which items have changed, been updated or deleted from the lists
* It also helps to determine which components need to be re-rendered instead of re-rendering all the components every time. Therefore, it increases performance, as only the updated components are re-rendered

### **26. What is Redux?**

Redux is an open-source, JavaScript library used to manage the application state. React uses Redux to build the user interface. It is a predictable state container for JavaScript applications and is used for the entire application’s state management.

### **27. What are the components of Redux?**

* **Store:** Holds the state of the application.
* **Action:** The source information for the store.
* **Reducer:** Specifies how the application's state changes in response to actions sent to the store.

### **28. What is the Flux?**

* Flux is the application architecture that Facebook uses for building web applications. It is a method of handling complex data inside a client-side application and manages how data flows in a React application.

### **29. What is React Router?**

React Router is a routing library built on top of React, which is used to create routes in a React application.

### **30. Why do we need to React Router?**

* It maintains consistent structure and behavior and is used to develop single-page web applications.
* Enables multiple views in a single application by defining multiple routes in the React application.

### ****31. 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.

**32. what is fragment In react?**

 React, whenever we want to render something on the screen, we need to use a render method inside the component. This render method can return **single** elements or **multiple** elements. The render method will only render a single root node inside it at a time. However, if we want to return multiple elements, the render method will require a '**div**' tag and put the entire content or elements inside it. This extra node to the DOM sometimes results in the wrong formatting of your HTML output and also not loved by the many developers.

To solve this problem, React introduced **Fragments** from the **16.2** and above version. Fragments allow you to group a list of children without adding extra nodes to the DOM.

**33. Why we use Fragments?**

The main reason to use Fragments tag is:

1. It makes the execution of code faster as compared to the div tag.
2. It takes less memory.

**What are hooks?**

Hooks are the new feature introduced in the React 16.8 version. It allows us to use state and other React features without writing a class. It does not work inside classes.

**Usestate ():**

* The React useState Hook allows us to track state in a function component.
* To use the useState Hook, we first need to import it into our component.
* We initialize our state by calling useState in our function component.
* useState accepts an initial state and returns two values:
  1. The current state.
  2. A function that updates the state.

Example: const[count, setcount] = usestate(“”);

**UseEffect ():**

* The useEffect Hook allows you to perform side effects in our components.
* Some examples of side effects are fetching data, directly updating the DOM, and timers.
* useEffect accepts two arguments. The second argument is optional.

useEffect(<function>, <dependency>).

**UseRef():**

* The useRef Hook allows us to persist values between renders.
* It can be used to store a mutable value that does not cause a re-render when updated.
* It can be used to access a DOM element directly.

If we tried to count how many times our application renders using the useState Hook, we would be caught in an infinite loop since this Hook itself causes a re-render.

To avoid this, we can use the useRef Hook.

**useReducer();**

* The useReducer Hook is similar to the useState Hook.
* It allows for custom state logic.
* If we want to keeping track of multiple pieces of state that rely on complex logic, useReducer may be useful.
* The useReducer Hook accepts two arguments.

useReducer(<reducer>, <initialState>)

The reducer function contains custom state logic and the initialState can be a simple value but generally will contain an object.

The useReducer Hook returns the current state and a dispatchmethod.

**useMemo():**

* useMemo is used for memorizing expensive computations within functional components.
* It caches the result of a function and returns the cached result when the inputs to the function remain unchanged.
* useMemo focuses solely on optimizing performance by avoiding unnecessary recalculations.

**Node js**

### 1. What is Node.js?

Node.js is an open-source, cross-platform JavaScript runtime environment and library to run web applications outside the client’s browser**.**It is used to create server-side web applications.

2. Why use Node.js?

Node.js makes building scalable network programs easy. Some of its advantages include:

* It is generally fast
* It rarely blocks
* It offers a unified programming language and data type
* Everything is asynchronous
* It yields great concurrency

### 3. 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 instead of the typical thread-based implementation.

### 4. 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.

5. What are the advantages of using promises instead of callbacks?

* The control flow of asynchronous logic is more specified and structured.
* The coupling is low.
* We've built-in error handling.
* Improved readability.

### 6. What is NPM?

NPM stands for Node Package Manager, responsible for managing all the packages and modules for Node.js.

### 7. What are the modules in Node.js?

Modules are like JavaScript libraries that can be used in a Node.js application to include a set of functions. To include a module in a Node.js application, use the **require ()** function with the parentheses containing the module's name.

Node.js has many modules to provide the basic functionality needed for a web application. Some of them include:

|  |  |
| --- | --- |
| Core Modules | Description |
| HTTP | Includes classes, methods, and events to create a Node.js HTTP server |
| util | Includes utility functions useful for developers |
| fs | Includes events, classes, and methods to deal with file I/O operations |
| url | Includes methods for URL parsing |
| query string | Includes methods to work with query string |
| stream | Includes methods to handle streaming data |
| zlib | Includes methods to compress or decompress files |

### 8. Which database is more popularly used with Node.js?

MongoDB is the most common database used with Node.js. It is a NoSQL, cross-platform, document-oriented database that provides high performance, high availability, and easy scalability.

### 9. What are some of the most commonly used libraries in Node.js?

There are two commonly used libraries in Node.js:

* **Express.JS** - Express is a flexible Node.js web application framework that provides a wide set of features to develop web and mobile applications.
* **Mongoose** - Mongoose is also a Node.js web application framework that makes it easy to connect an application to a database.

### 10. What are the pros and cons of Node.js?

|  |  |
| --- | --- |
| **Node.js Pros** | **Node.js Cons** |
| Fast processing and an event-based model | Not suitable for heavy computational tasks |
| Uses JavaScript, which is well-known amongst developers | Using callback is complex since you end up with several nested callbacks |
| Node Package Manager has over 50,000 packages that provide the functionality to an application | Dealing with relational databases is not a good option for Node.js |
| Best suited for streaming huge amounts of data and I/O intensive operations | Since Node.js is single-threaded, CPU intensive tasks are not its strong suit |

### 11. What is the command used to import external libraries?

The “require” command 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.

### 12. What is an Event Loop in Node.js?

Event loops handle asynchronous callbacks in Node.js. It is the foundation of the non-blocking input/output in Node.js, making it one of the most important environmental features.

13. What is an Event Emitter in Node.js?

* Event Emitter is a class that holds all the objects that can emit events
* Whenever an object from the Event Emitter class throws an event, all attached functions are called upon synchronously

14. What are the two types of API functions in Node.js?

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

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

### 15. What is the package.json file?

The package.json file is the heart of a Node.js system. This file holds the metadata for a particular project. The package.json file is found in the root directory of any Node application or module

This is what a package.json file looks like immediately after creating a Node.js project using the command: **npm init**

### 16. What is the Express.js package?

Express is a flexible Node.js web application framework that provides a wide set of features to develop both web and mobile applications.

17. How do you create a simple Express.js application?

* The request object represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on
* The response object represents the HTTP response that an Express app sends when it receives an HTTP request



### 18. What are streams in Node.js?

Streams are objects that enable you to read data or write data continuously.

There are four types of streams:

**Readable –** Used for reading operations

**Writable −** Used for writing operations

**Duplex −** Can be used for both reading and write operations

**Transform −** A type of duplex stream where the output is computed based on input

19. How do you create a simple server in Node.js that returns Hello World?



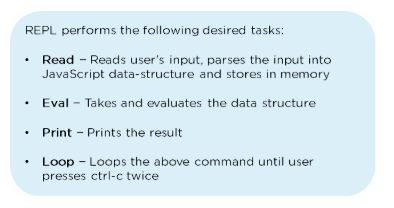
* Import the HTTP module
* Use createServer function with a callback function using request and response as parameters.
* Type “hello world."
* Set the server to listen to port 8080 and assign an IP address

20. Explain asynchronous and non-blocking APIs in Node.js.

* All Node.js library APIs are asynchronous, which means they are also non-blocking
* A Node.js-based server never waits for an API to return data. Instead, it moves to the next API after calling it, and a notification mechanism from a Node.js event responds to the server for the previous API call

### 21. What is REPL in Node.js?

REPL stands for Read Eval Print Loop, and it represents a computer environment. It’s similar to a Windows console or Unix/Linux shell in which a command is entered. Then, the system responds with an output



### 22. What is piping in Node.js?

Piping is a mechanism used to connect the output of one stream to another stream. It is normally used to retrieve data from one stream and pass output to another stream

### 23. What is a reactor pattern in Node.js?

A reactor pattern is a concept of non-blocking I/O operations. This pattern provides a handler that is associated with each I/O operation. As soon as an I/O request is generated, it is then submitted to a demultiplexer

24. Explain the concept of middleware in Node.js.

Middleware is a function that receives the request and response objects. Most tasks that the middleware functions perform are:

* Execute any code
* Update or modify the request and the response objects
* Finish the request-response cycle
* Invoke the next middleware in the stack

### 25. What are the different types of HTTP requests?

HTTP defines a set of request methods used to perform desired actions. The request methods include:

**GET:**Used to retrieve the data

**POST:**Generally used to make a change in state or reactions on the server

**HEAD:**Similar to the GET method, but asks for the response without the response body

**DELETE:** Used to delete the predetermined resource

**26. List down the various timing features of Node.js.**

Node.js provides a Timers module which contains various functions for executing the code after a specified period of time. Below I have listed down the various functions provided by this module:

* **setTimeout/clearTimeout** – Used to schedule code execution after a designated amount of milliseconds
* **setInterval/clearInterval** – Used to execute a block of code multiple times
* **setImmediate/clearImmediate** – Used to execute code at the end of the current event loop cycle
* **process.nextTick** – Used to schedule a callback function that needs to be invoked in the next iteration of the Event Loop

### ****27. What is the use of NODE\_ENV?****

If the project is in the production stage, Node.js promotes the convention of making use of NODE\_ENV variable to flag it. This helps in taking better judgment during the development of the projects. Also, when you set your NODE\_ENV to production, your application tends to perform 3 times faster.

### ****28. What do you understand by ESLint?****

ESLint is an open source project initially developed by Nicholas C. Zakas in 2013 which aims to provide a linting utility for JavaScript through a plug. Linters in Node.js are good tools for searching certain bug classes, especially those which are related to the variable scope.

### ****29. What do you understand by a test pyramid?****

A test pyramid basically is a diagram that describes the ratio of how many unit tests, integration tests, and end-to-end test are required to be written for the successful development of the project.

### ****30. Explain the concept of URL module.****

The **URL module** of Node.js provides various utilities for **URL** resolution and parsing. It is a built-in module that helps in splitting up the web address into a readable format:

|  |  |
| --- | --- |
| 1 | var url = require('url'); |

### 31. For Node.js, why does Google use the V8 engine?

The V8 engine, developed by Google, is open-source and written in [C++](https://www.simplilearn.com/tutorials/cpp-tutorial/learn-cpp-basics). Google Chrome makes use of this engine. V8, unlike the other engines, is also utilized for the popular Node.js runtime. V8 was initially intended to improve the speed of JavaScript execution within web browsers. Instead of employing an interpreter, V8 converts JavaScript code into more efficient machine code to increase performance. It turns JavaScript code into machine code during execution by utilizing a JIT (Just-In-Time) compiler, as do many current JavaScript engines such as SpiderMonkey or Rhino (Mozilla).

**Express.js**

### 1) What is Express.js?

Express.js, or simply Express, is a free, open-source, lightweight, and fast backend web application framework for Node.js. It is released as open-source software under the MIT License.

It is designed for building single-page, multi-page, and hybrid web applications and APIs. It is called the de facto standard server framework for Node.js. It was founded and developed by **TJ Holowaychuk** in 2010 and written in JavaScript.

### 2) What are some distinctive features of Express?

* It allows to set up middleware to respond to HTTP/RESTful Requests.
* It defines a routing table to perform different HTTP operations (method and URL).
* It allows to dynamically rendering HTML Pages based on passing arguments to templates.
* It provides high performance because of its ultra-fast I/O. It prepares a thin layer; therefore, the performance is adequate.
* Its MVC-like structure makes it organize the web application into MVC architecture.
* It provides good database support. It supports RDBMS as well as NoSQL databases.
* It is asynchronous and single-threaded.
* Its robust API makes routing easy.

### 3) Is Express.js front-end or backend framework?

Express.js or Express is a JavaScript backend framework. It is mainly designed to develop complete web applications (single-page, multi-page, and hybrid web applications) and APIs. Express is the backend component of the **MEAN stack** where **M stands for MongoDB**, which handles database; **E stands for Express,** which handles backend; **A stands for AngularJS**, which is for the front-end, and **N stands for Node**.

4) Which are the arguments available to an Express JS route handler function?

Following are the arguments that are available to an Express.js route handler-function:

* **Req:** the request object
* **Res:** the response object
* **Next (optional):** It is a function employed to pass management to one of the above route handlers.

### 5) What is the difference between Express.js and Node.js?

|  |  |  |
| --- | --- | --- |
| **Feature** | **Express.js** | **Node.js** |
| Definition | Express.js is a lightweight and fast backend web application framework for Node.js. | Node.js is an open-source and cross-platform that is used to execute JavaScript code outside of a browser. |
| Usage | Express.js is used to develop complete web applications such as single-page, multi-page, and hybrid web applications and APIs. It uses approaches and principles of Node.js. | Node.js is used to build server-side, input-output, event-driven apps. |
| Features | Express has more features than Node.js. | Node.js has fewer features as compared to Express.js. |
| Building Block | Express.js is built on Node.js. | Node.js is built on Google's V8 engine. |
| Written in | Express.js is written in JavaScript only. | Node.js is written in C, C++, and JavaScript language. |
| Framework/Platform | Express.js is a framework of Node.js based on its functionalities. | Node.js is a run-time platform or environment designed for server-side execution of JavaScript. |
| Controllers | Express.js is assigned with controllers. | Node.js is assigned with controllers. |
| Routing | Routing is provided in Express.js. | Routing is not provided in Node.js. |
| Middleware | Express.js uses middleware to arrange the functions systematically on the server-side. | Node.js doesn't use any such provision of middleware. |
| Coding | Express is easy to code and requires less coding time. | Node.js requires more coding time as compare to Express.js. |

### 4) How can you enable debugging in Express.js app?

**Use the following command on Windows: node app.js**

### 4) What is CORS?

### CORS stands for Cross-Origin Resource Sharing. It allows us to relax the security applied to an API. This is done by bypassing the Access-Control-Allow-Origin headers, which specify which origins can access the API.

5) What is Middleware in Express.js? What are the different types of Middleware?

Middleware is a function invoked by the Express routing layer before the final request handler.

Middleware functions are used to perform the following tasks:

* It is used to execute any code.
* It is also used to make changes to the request and the response objects.
* It is responsible for ending the request-response cycle.
* It can call the next middleware function in the stack.

**Type of Middleware**

Following are the main types of Middleware:

* Application-level Middleware
* Router-level Middleware
* Error-handling Middleware
* Built-in Middleware
* Third-party Middleware

### 6) Which template engines do Express support?

Express.js supports any template engine that follows the (path, locals, callback) signature.

**MongoDB**

### ****1. Define MongoDB.****

It is a document-oriented database that is used for high availability, easy scalability, and high performance. It supports the dynamic schema design.

**2. What are the key features of MongoDB?**

There are three main features of MongoDB:

* Automatic scaling
* High performance
* High availability

### ****3. What type of a DBMS is MongoDB?****

MongoDB is a document-oriented DBMS.

**4. What is CRUD?**

MongoDB provides CRUD operations:

C – Create – db.collection.insert();  
R – Read – db.collection.find();  
U – Update – db.collection.update();  
D – Delete – db.collection.remove({“fieldname” : ”value”});

### ****5. What is Aggregation in MongoDB?****

In MongoDB, aggregations are operations that process data records and return computed results.

### ****6. Define Namespace in MongoDB.****

It is the concatenation of the collection name and the name of the database.

### ****7. What is a Collection in MongoDB?****

In MongoDB, a collection is a group of MongoDB documents.

### ****8. Which command is used to create a database?****

To create a database, we can use the **Database\_Name** command.

### ****9. Which syntax is used to create a Collection in MongoDB?****

We can create a collection in MongoDB using the following syntax:

db.createCollection(name,options)

### ****10. Which command is used for inserting a document in MongoDB?****

The following command is used for inserting a document in MongoDB:

database.collection.insert (document)

### ****11. Which syntax is used to drop a Collection in MongoDB?****

We can use the following syntax to drop a collection in MongoDB:

db.collection.drop()

### ****12. Explain Replication.****

Replication is the process of synchronizing data across multiple servers.

### 13. By default, which replica sets are used to write data?

By default, MongoDB writes data only to the primary replica set.

### ****14. Which command is used to drop a database?****

The **db.dropDatabse()** command is used to drop a database.

### ****15. What is the use of the pretty() method?****

The pretty() method is used to show the results in a formatted way.

### ****16. What is the use of the limit() method?****

The limit() method is used to limit the records in the database.

db.COLLECTION\_NAME.find().limit(NUMBER)

### ****17. What is the use of the db command?****

The db command gives the name of the currently selected database.

### ****18. Which method is used to update documents into a collection?****

The update() and save() methods are used to update documents into a collection.

### ****19. What is the syntax of the skip() method?****

The syntax of the skip() methopd is as follows:

>db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER)

### ****20. Which language is used to write for MongoDB?****

C++ is used for writing and implementing MongoDB.

### ****21. In which format does MongoDB store data?****

MongoDB uses collections to store data rather than tables.

### 22. Does MongoDB support primary-key, foreign-key relationship?

No. By Default, MongoDB doesn't support primary key-foreign key relationship.

### 23. In which format MongoDB represents document structure?

MongoDB uses BSON to represent document structures.

**24. Explain the structure of ObjectID in MongoDB.**

ObjectID is a 12-byte BSON type with:

* 4 bytes value representing seconds
* 3 byte machine identifier
* 2 byte process id
* 3 byte counter

**25. How many indexes does MongoDB create by default for a new collection?**

By default, MongoDB created the \_id collection for every collection.

**26. Why MongoDB is not preferred over a 32-bit system?**

When running a 32-bit build of MongoDB, the total storage size for the server, including data and indexes, is 2 gigabytes. For this reason, do not deploy MongoDB to production on 32-bit machines.

If you're running a 64-bit build of MongoDB, there's virtually no limit to storage size.

**27. Which are the two storage engines used by MongoDB?**

MongoDB uses MMAPv1 and WiredTiger.

### 28 How does MongoDB provide concurrency?

MongoDB uses reader-writer locks for concurrency. Reader-writer locks allow concurrent readers shared access to a resource, such as a database or collection, but give exclusive access to a single write operation.

### 29) Do the MongoDB databases have schema?

Yes. MongoDB databases have dynamic schema. There is no need to define the structure to create collections.

### 30) By default, which index is created by MongoDB for every collection?

By default, the\_id collection is created for every collection by MongoDB.

**ES6**

**1) Define ECMAScript.**

It is the specification that is defined in the ECMA-262 standard to create a general-purpose scripting language.

**3) What are the new features introduced in ES6?**

The new features that are introduced in ES6 are listed as follows:

* Let and const keywords.
* Default Parameters.
* Arrow functions.
* Template Literals.
* Object Literals.
* Rest and spread operators.
* Destructuring assignment.
* Modules, Classes, Generators, and iterators.
* Promises, and many more.