**HTML FLAGS**

**1.What are the differences between html4 and html5?**

HTML 4 is the 4th and older version of HTML, with less features and tags, whereas HTML 5 is an extension of HTML 4 and the 5th version of HTML, with new and simple functionality as well as several new tags. HTML 4 does not allow for multimedia, however HTML 5 does.

Only !DOCTYPE html> is written in HTML 5. Because html4 has less functionalities, we must put some lengthy code in the doctype. Html5 has a better error handling functionality than HTML4 . New tags in HTML5 include canvas, audio, and video, which are not available in HTML4.

**2.What are semantic tags in html? Give me some examples.**

HTML code written with semantic tags (correct and descriptive tags) is referred to as semantic HTML. Semantic HTML tags tell us—and our browser—the intended purpose of the text contained in the tag, rather than just indicating the start of a new section or indicating which area of a page would carry a certain tag or ID. On both the block and inline levels, the HTML5 specifications added several additional semantic components to the syntax.

**Examples**: <Header>, <footer>, <audio>, and <video> .

**3. What is the purpose of Article, div, section, nav, aside?**

<Header>, nav>, <section>, article>, aside>, and footer> elements behave similarly to <div> elements. They divide the page into parts by grouping other items together. But semantic tags are much easier to read. It has greater accessibility. Overall, semantic elements also lead to more consistent code.

**4.Why will you use Meta tag?**

Users cannot see meta tags, but they can be found in the HTML source code and are visible to search engines. Meta tags define and aid in informing search engines about the content of a web page.

There are three key aspects of Meta tags that you might adopt:

The title tag is the text that appears in search engine results pages.

The meta description tag is where you should insert the summary of your website.

Keywords — Put all of the keywords you use on your site in the meta keywords element.

**5.What is the difference between inline, inline-block, and block?**

**Inline Components:**

Inline components are placed literally side by side with other inline and inline-block elements on a single line.You cannot specify a precise height, width, margin-top, or margin-bottom. Consequently, without any paddings or side margins, inline elements will only be as wide as their inner content.

**Inline-block:**

The difference between an inline element and an inline-block element is that an inline-block element can take up specified width and height. But, it will also not start on a new line within its parent or cause a line break after it.

**Block:**

Any element styled with display: block is the polar opposite of display:inline. A block element starts on a new line and occupies the available width of its parent element or its specified width.

The block elements always start on a new line. They will also take space of an entire row or width. It means that there can be no other HTML elements that can stand side by side with block level elements.

**6.Difference between strong, b, em, i?**

These are inline properties.

**Strong**: It is used to show text bold or highlight it semantically.

**Bold or b:** This bold tag is just offset text conventionally styled in bold.

**em:** <em> tag semantically emphasizes the important word.

**i :** <i> tag just offset text conventionally styled in italic.

**7. What are properties and attributes in HTML?**

When the browser parses the HTML, it creates a tree data structure wich basically is an in memory representation of the HTML. It the tree data structure contains nodes which are HTML elements and text. Attributes and properties relate to this is the following manner:

**Attributes** are additional information which we can put in the HTML to initialize certain DOM properties.

**Properties** are formed when the browser parses the HTML and generates the DOM. Each of the elements in the DOM have their own set of properties which are all set by the browser. Some of these properties can have their initial value set by HTML attributes. Whenever a DOM property changes which has influence on the rendered page, the page will be immediately re rendered.

**8.What is a Viewport?**

Viewport meta tag for Responsive Web Design. A Browser's viewport is the area of web page in which the content is visible to the user. The viewport does not have the same size, it varies with the variation in screen size of the devices on which the website is visible. Viewport is commonly described as the visible area of a web page, which changes based on the size of the device used to access the page. For example, desktop computers, tablets and smartphones all have a different viewports.

**9.Have you used Audio and Video tags? How does they work?**

The HTML5 audio> and video> tags make adding media to a website straightforward. You must set the src attribute to identify the media source and provide the controls attribute to allow the user to play and pause media. The HTML5 video element can contain a variety of properties to manage its appearance and functionality, such as autoplay, autobuffer, height, and loop. The HTML5 audio tag can contain a variety of properties to manage its appearance and functionality, such as autoplay, autobuffer, height, loop, preload, etc.

**10.What is hyperlink in html? what tag and attribute will you use for hyperlink?**

In HTML, a hyperlink is denoted by the <a> tag, which is also referred to as the anchor tag. This is the basic tag that can be used to generate links within web pages. When it comes to determining where a link will lead, the "href" property of the "a" element is the most crucial piece of information to have. The "target" element of the a tag specifies where the link will open (the same page, a new page, or a new window), however the default behavior of the target tag is to open the link on the same page as its parent.

**11.What is the difference between HTML elements and tags?**

Start and end tags are often the components that make up an HTML element. Everything between the start tag and the end tag is considered to be part of the HTML element.

The terms "elements" and "tags" are sometimes mixed up with one another. Documents written in HTML include tags, but they do not include the elements themselves. These tags are not used to generate the elements until after the parsing phase has been completed.

For example <p> is starting tag of a paragraph and </p> is closing tag of the same paragraph but <p>This is paragraph</p> is a paragraph element.

**12.What is charset in html? why will you use it?**

A charset, also known as a character set in its complete form, is essentially a collection of characters that can be identified by a computer in the same way that a calculator can. This establishes a communication channel that may be used for encoding and decoding content, with each of these characters being represented by a number that is referred to as a code point.

As a result, a character set is comprised of individual characters that are designed to fulfill a certain function. The characters are saved on the computer as either one byte or many bytes. One example of this would be the ASCII character set, which assigns numerical values ranging from 0 to 127 to each of the English characters and special control characters.The character encoding needs to be understood in advance in order to avoid any issues that may arise when this data is eventually decoded. One illustration of this type can be observed in internet browsers whenever a webpage is being viewed. Either the server itself or the developer will directly write the information about the character set that is being used in the application. Regrettably, there are a plethora of character sets, and as a result, there are multiple ways in which binary codes can be matched to characters and bytes.

**CSS FLAGS**

1. **What Flex layout? Difference Flex and grid layout?**

The Flexbox (or Flexible Box) layout model is a one-dimensional CSS layout model that enables you to design responsive layouts for your web application. You can also refer to this model as the Flexible Box layout model.

CSS Grid and Flexbox are layout models that share similarities and can be used together. The most important distinction is that Flexbox can only be used to build one-dimensional layouts, but CSS Grid can be used to generate both one-dimensional and two-dimensional layouts. This means that you can arrange components along the X-axis and the Y-axis in CSS Grid, while Flexbox only allows you to place components along one axis.

1. **Explain CSS position property? What are some differences between absolute position and relative position?**

CSS position: relative; represents that the element will be positioned relative to its default position in the document. Here, the default position means, the position of the element according to the normal flow of the layout. So, if we apply any of the top, right, bottom, or left properties on a relatively positioned element, the element will move away from its original or default position.

CSS position: absolute; is totally different from it. If we apply position: absolute; on an element, the element will be positioned relative to its nearest positioned ancestor(i.e. any parent element having position other than static ).

1. **What is a box model? And what are the different elements of a box model?**

The CSS box model is a container that holds several properties, such as borders, margins, padding, and the content itself. These attributes are all contained within the box model. It is utilized in the process of designing the look and structure of web sites. The CSS box model dictates that the web browser must provide each element in the form of a square box. For example, if you add a <p> element to your webpage and load it, it initially looks like it's just loose text. If you add some CSS colors, though, you can see the square box shape:

1. **What is a Hover effect? What is the purpose of the active class? or What is pseudo-class?**

Both :hover and :active are examples of pseudo-classes that are created based on the activities of a user. They each correlate to a very particular moment in the process by which a user will interact with an element on a website, such as a link, a button, or an input field. Examples of such elements include input fields, buttons, and links. The :hover pseudo-class, also called the “pointer hover pseudo-class”, applies when a pointing device interacts with an element without necessarily activating it. The :active pseudo-class applies during the period in which the element is being activated.

1. **What are the different types of Selectors in CSS? or Differences between Class selector from ID selector?**

CSS selectors are divided into the following five categories:

· Simple selectors

· Selectors for combinatorics

· Pseudo-class selectors

· Pseudo-elements selectors

· Selectors of particular attributes

The class selector selects elements with a specific class attribute. It matches all the HTML elements based on the contents of their class attribute

The ID selector matches an element based on the value of its id attribute. In order for the element to be selected, its ID attribute must exactly match the value given in the selector

1. **What is CSS Specificity?**

The CSS specificity property is a set of criteria that browsers use in order to decide which of the styles that were defined by the developer will be applied to a particular element. The developer needs to adhere to the guidelines in order for the browser to understand how to apply a style to a specific element, and this is necessary so that the style may be applied. When there are two or more styles that target the same element, the one that has the highest level of specificity will be the one that is used.

Hierarchy of CSS Specificity

· Inline Styles

· ID Selectors

· Classes, Attributes, and Pseudo Classes

· Elements and Pseudo Elements

1. **What is a CSS Preprocessor? What are some benefits of Sass?**

According to my knowledge, this is an add-on for the standard CSS that expands the basic capabilities, adds some functions, and enables you to write code that is more readable and straightforward to comprehend. When the result is generated, the source code is compiled into the conventional CSS.

Benefits of Sass

Sass is a programming concept that enables you to build CSS that is more readable, simpler, and requires less lines of code. Because it has less codes, you will be able to write CSS much more quickly. Because it is an extension of CSS, it is more stable, powerful, and elegant than other options. Therefore, it is simple for designers and developers to work in a more effective and expedient manner. It works with all different versions of the CSS standard. You are free to use any of the CSS libraries that are currently available.

1. **What is a Pseudo element? Give an example of pseudo element**

The addition of a pseudo-element is analogous to targeting an additional element, but it does not need the addition of any additional HTML. They serve a multitude of purposes, all of which are explained in this section of the module.

Example:

p:after{

content: " -Read more… ";

color:#f00;

}

1. **How will you use media queries to make a website responsive?**

The CSS media queries were initially implemented as part of the CSS2 specification, which included the CSS media rule as part of its library.

If you're just starting out as a web developer, you might find it a little strange that the syntax for CSS media queries is similar to that of TestNG annotations.

The following is an example of how the media query can be done using the word "media":

As an example:

@media only screen and (max-width: 480px) {

/\* CSS rules to apply /\*

}

@media only screen and (max-width: 480px) {

/\* CSS rules to apply /\*

}

1. **How will you make font size responsive?or What are some CSS Measuring units? Which one will you use?**

If you want to design web pages with responsive text, you should be familiar with relative units in addition to absolute units.

Here are the most common relative units that you can use for responsive text:

· Percentage (%)

· Viewport Width (vw)

· Viewport Height (vh)

· em

· rem

1. **Difference between transition and transform**

CSS Transitions allows property changes in CSS values to occur smoothly over a specified duration.

CSS transforms allows elements styled with CSS to be transformed in two-dimensional or three-dimensional space.

1. **How will you horizontally and vertically center a div inside a div**

Absolute + Calc

In the new CSS standard, CSS introduces the calc function, which allows us to write code similar to 50% — 50px.

.father {

position: relative;

}

.child {

position: absolute;

top: calc(50% - 50px);

left: calc(50% - 50px);

}

**JAVASCRIPT FLAGS**

**1.What is the JavaScript Event Loop?**

The execution of the code, the collection and processing of events, and the completion of queued subtasks are all handled by JavaScript's event-driven runtime model, which is responsible for the runtime model's core responsibilities.

The sole responsibility of the Event Loop is to keep an eye on the Call Stack and the Callback Queue at all times. If the Call Stack is empty, the Event Loop will take the first event from the queue and push it to the Call Stack, which will, in effect, run the event. If the Call Stack is not empty, the Event Loop will repeat this process. In the Event Loop, an iteration of this kind is referred to as a tick. Every event is really just a callback to a function.

**2. How does JavaScript code is executed in Browser?**

When we speak about a JavaScript engine, we are typically referring to the compiler. The compiler is a piece of software that takes source code that is readable by humans (in this case, JavaScript) and generates instructions that are readable by machines, such as your computer.

The following are the possible steps in a straightforward compilation process:

1. The lexical analyzer will read your source code and break it down into tokens, which are smaller than atomic units.

2. The code that has been tokenized is next put through a parser, which analyzes the code to determine its structure and scope, and encodes this information into a syntax tree.

3. This structure, which resembles a graph, is then sent through a translator so that it may be converted into bytecode.

4. The bytecode is then rendered after being processed by a bytecode interpreter, which converts it into native code.

**3. What are the differences between “==” and “===” ?**

To determine whether or not two expressions are equal to one another, the '==' operator is a standard component of virtually every programming language. If the expressions on both sides of the == operator are the same or evaluate to the same value, then the result will be TRUE. If the expressions on both sides of the == operator are different or evaluate to different values, then the result will be FALSE.

If there are two operands to the === operator and both are the same but have different datatypes, then === will return FALSE. However, if the datatypes are same is, if the two operands are exactly the same in everything—then only === will evaluate to TRUE.

**4. What is a callback function?**

A callback is a function given as an argument to another function, and that is to be called at the end of the containing function , hence the name “callback”.

As an exemple if you want to sort an arry of numbers you need to provide the sort method with the appropiate compare method (by default it would compare the strings instead of number values)

**5. How will you return more than one value from a function?**

We can do this from ECMAScript 6 onwards using arrays and "destructuring ".

function myfunction()

{

var n=0;var s=1;var w=2;var e=3;

return [n,s,w,e];

}

**6. Tell me about bind, call and apply.**

Call invokes the function immediately and allows you to pass in arguments one by one

Apply invokes the function immediately and allows you to pass in arguments as an array.

Bind returns a new function, and you can invoke/call it anytime you want by invoking a function.

**7. What is a Closure in JavaScript? How does it work?**

A closure is a type of function in JavaScript that refers to variables that are defined in an outer scope from within its own scope. The closure ensures that the outer scope remains intact within the context of the inner scope. Because of closure, an inner function will always have access to the variables and arguments of its outer function, even after the outer function has returned. This access will remain even after closure has been applied.

**8. What does the “this” keyword indicate in JavaScript?**

The this keyword behaves differently in JavaScript compared to other languages. In JavaScript the value of this is determined by the invocation context of function (context.function()) and where it is called.

**Different use of this keyword.**

· When used in global context

· When used inside object method

· When invoking context-less function

**9. What is Event bubbling in js? Or How does event delegate work in JS?**

Event delegation is the process of managing a bubbling event by adding an event handler to a container element, but only activating the behavior of the event handler if the bubbling event occurred on an element within the container that satisfies a given condition. This is done through the use of an event handler. This can make it easier to handle events that occur on items included within the container.

Take, for instance, the scenario in which you need to respond to a click on any table cell inside a large table. You could build a loop to attach a click handler to each each cell, or you could attach a click handler to the table itself and then use event delegation to restrict its activation to table cells only (and not table headers, or the whitespace within a row around cells, etc.).

**10. Explain hoisting in JavaScript.**

JavaScript The term "hoisting" refers to the process by which the interpreter seems to relocate the declaration of functions, variables, or classes to the top of their scope prior to the execution of the code. This occurs before the code is actually executed. Hoisting makes it possible for functions to be utilized securely in code prior to their being declared. The declarations of variables and classes are hoisted as well, so that they, too, can be referenced before they are really declared. Be aware that doing so can lead to errors that were not anticipated, and that doing so is not normally encouraged.

**11. What is a recursive function.**

The technique of calling itself is known as recursion. A recursive function is a function that calls itself several times inside the same scope. It is necessary for a recursive function to have a condition that tells it when to stop calling itself. In that case, the function will be called on and on forever. After the condition has been satisfied, the function will no longer call itself. A base condition is another name for this. You can avoid an infinite recursion by utilizing an if...else statement (or another approach of a similar nature) in which one branch of the statement makes the recursive call and the other branch does not.

**12. Difference between undefined and null**

The undefined value is a primitive value used when a variable has not been assigned a value.

The null value is a primitive value that represents the null, empty, or non-existent reference.

**13. What are the different data types in JavaScript?**

There are seven different types in JavaScript. The values that JavaScript can take are referred to as types. The following is a list of the data types that can be used with JavaScript:

· Number

· String

· Boolean

· Undefined

· Null

· Object

· Symbol

**14. What is DOM**

The abbreviation for "Document Object Model" is "DOM." It is a programming interface that gives us the ability to add new elements to the document, modify existing ones, or delete existing ones. To make our page more dynamic, we can also add events to the elements that make up the page.

The Document Object Model (DOM) sees an HTML document as a tree made up of nodes. A node represents an HTML element.

**15. Is JavaScript a static type or a dynamic type?**

JavaScript is a dynamically typed language, dynamically typed languages all type checks are performed in a runtime, only when your program is executing. So this means you can just assign anything you want to the variable and it will work.

**ES6 FLAGS**

**1. Tell me about Es6**

A newer version of JavaScript, known as JavaScript ES6 (also known as ECMAScript 2015 or ECMAScript 6), was released in 2015. ECMAScript 2015 is another name for JavaScript ES6. The programming language JavaScript conforms to the guidelines established by ECMAScript. ECMAScript is the source of the specification that details how the JavaScript programming language ought to function.

Now, let's take a look at some of the more significant improvements that ES6 adds to JavaScript.

· Constants

· Block-Scoped Variables and Functions

· Arrow Functions

· Default Function Parameters

· Rest Function Parameters

· String Templating

**2. What are the differences between var, let, and const?**

Declarations of var can either have the global scope or function scope. Therefore, a variable is considered to be in the global scope if it is declared with the var keyword outside of a function (accessible anywhere). In any other case, if they are included within a function, we will only be able to access them while we are inside the function.

let is a variable that is only accessible within this block; hence, we cannot re-declare it but it can be modified (re-assigned).

Const declarations are somewhat similar to let declarations, however, const variables cannot be modified or re-declared. let variables, on the other hand, can be updated. It ensures that the constant values are maintained.

In contrast to var and let, the const keyword requires that the variable be initialized at the same time that it is declared in order for it to be valid.

**3. Why will you use default parameters?**

You are now able to assign a default value to any parameter thanks to the advent of default parameters in ES2015. This value will be used by the function in place of undefined whenever the function is called without any arguments.

In the absence of default parameters, the process of setting defaults would require the user to perform an explicit check for values that have not been defined.

Using default parameters, on the other hand, achieves the same effect while requiring far less code. You can assign a default value to the parameter in the cube by using the equality assignment operator, which is represented by an equal sign (=).

**4. How does the Spread operator work?**

The collection of operators in JavaScript ES6 has been updated with the addition of a new one called the spread operator. It takes an iterable as its input (for example, an array) and breaks it down into its component parts. The spread operator is frequently utilized to produce shallow copies of JS objects. By utilizing this operator, the code can be made more concise while simultaneously improving its readability.

Example:

const numbersOne = [1, 2, 3];

const numbersTwo = [4, 5, 6];

const numbersCombined = [...numbersOne, ...numbersTwo];

**5. Difference between class and object**

A class is responsible for defining object properties, which may include a valid range of values and a default value. The behavior of an item is also described by a class. An object is a member of a class, often known as a "instance" of that class. There is a state that an object can be in where all of its attributes have values that are either explicitly defined by you or are defined by the default settings for the object. Moreover, everything in JavaScript is an object (not primitive values of course), so an instance of a class is an object. An object is simply the smallest building block in OO-Programming in JavaScript.

**6. What is a Prototype chain?**

The ability to access object properties from another object is what's meant when we talk about prototype inheritance. When extending the functionality of an existing object constructor with additional attributes and methods, we make use of a JavaScript prototype. After that, we will be able to instruct our JS code to effectively inherit properties from a prototype. Through the usage of a reference pointer function, we are able to reuse the attributes or methods of one JavaScript object while inheriting them into another object thanks to prototypical inheritance.

**7. Explain Call by value vs call by reference**

When a variable is used as a parameter in a function, any changes to the parameter will not affect the original variable. This is called "call by value," and it works for all primitive data types.

When a variable's reference (address) instead of its value is passed to a function's parameter, any changes made to the parameter will update the original variable reference. This is called "call by reference," and it happens with all non-primitive data types.

**8. What is the scope in JavaScript?**

Scope refers to the sections of the code where variables and functions can be used.

In JavaScript, there are two types of scope for a variable:

· Global Scope

· Local Scope

A global scope variable is one that is set at the beginning of a program or outside of a function.

A variable can also only be accessible within a function, which is called a "local scope."

**9. What is a Higher-order Function?**

In JavaScript, functions are considered to be a data type in the same way as arrays, strings, and numbers are considered to be data types. As a result, functions can have their values assigned to variables, but they are distinct from all other sorts of data in that they can be called upon directly. From that way higher-order functions are those that either receive or return another function as their input or output respectively. Higher-order functions operates not on strings, numbers, or Booleans but rather on functions, which is why this type of function is referred to as higher order functions.

**10. What is API? Difference between Get vs post?**

The abbreviation API refers to the "Application Programming Interface."

An application programming interface, or API, is a collection of coding that facilitates data transmission from one software product to another.

The GET method is used to add form data to the URL as a pair of name and value. If you use GET, the URL length will stay the same. It makes it easier for users to send in the bookmark result. GET is better for information that doesn't need to be secure or have images or Word documents.

HTTP supports the POST method, which means that a web server will accept the data in the message's body. When you upload a file or send user-generated data to a web server, the World Wide Web often uses POST.

**11. Difference between local storage and Session storage**

Local storage: It keeps storing the user information data without expiration date this data will not be deleted when a user closed the browser windows it will be available for day, week, month and year. Local storage can store 5-10MB of offline data.

Session Storage: It is same as local storage date except it will delete all windows when browser windows closed by a web user.In Session storage can store upto 5 mb data

**12. What are cookies? And why will you use it?**

Cookies (web cookie, browser cookie) Cookies are data, stored in small text files as name-value pairs, on your computer.

New cookies can also be created via JavaScript using the Document. cookie property, and if the HttpOnly flag is not set, existing cookies can be accessed from JavaScript as well.

Cookies are often used in a web applications to identify a user and their authenticated session

When receiving an HTTP request, a server can send a Set-Cookie header with the response. The cookie is usually stored by the browser, and then the cookie is sent with requests made to the same server inside a Cookie HTTP header.

**13. What is object-oriented programming?**

Object-oriented programming, also known as OOP, is a paradigm of computer programming that organizes the design of software on data. One definition of an object is a data field that possesses special behaviors and characteristics in its own way.

In Object-Oriented Programming (OOP), the focus is on the objects themselves, rather than the logic that is necessary to handle those objects. This method of programming works particularly effectively for applications that are extensive, complicated, and frequently updated or maintained.

These are the four main principles of the object-oriented programming paradigm. Understanding them is essential to becoming a successful programmer.

· Encapsulation

· Inheritance

· Abstraction

· Polymorphism

**14. Difference between Array vs LinkedList.**

The term "array" refers to a group of elements that all have the same data type. Using the array index, it is possible to retrieve the members of the array in random order. Memory is organized so that data elements are stored in consecutive locations. Because memory locations are consecutive and cannot be changed, performing insertion and deletion operations incurs additional costs.

Linked List: A linked list is a collection of elements of the same type that is ordered and in which each entry is connected to the next member in the collection through the use of pointers. Accessing the items in a linked list at random is not possible. It will be necessary to access the items in the specified order. Any location is suitable for storing newly formed elements, and pointers are used to generate a reference to each new element. In a linked list, the insertion and deletion operations can be performed quickly and with little effort.

**15. How will you debug a JavaScript application**

In computer programming, "debugging" refers to the act of analyzing a program in order to locate and correct an error. Your JavaScript software can be debugged using a variety of different approaches.

Using console.log() : With the help of console.log(): In order to debug the code, you can make use of the console.log() method. You may verify that the data is accurate by using the console.log() method

Using debugger: The debugger keyword stops the execution of the code and calls the debugging function.

Setting Breakpoints: The debugger window allows you to put in breakpoints for the JavaScript code that you are developing. At each breakpoint, JavaScript will halt its execution and allow you to inspect the values in the variable. After that, you will be able to restart the process of executing the code.

**REACT JS FLAG**

**1.What is reactjs? Tell us about advantages and disadvantages of using react js.**

React is developed by Facebook in March 2013.

It is maintained by Facebook and a community of individual developers and companies. It can be used as a base in the development of single-page or mobile applications.

Advantages

· Component base Architecture

· Reusable code

· React is declarative

Disadvantages

· Lack of Documentation.

· It covers only UI hence we need to use other languages to make a full application.

· The high pace of development makes developer life difficult.

**2.Why will you select ReactJS?**

With over 100,000 websites currently using React, so there should be a good reason for it.

the major reason of choosing React for web development include the following:

· Development efficiency –

· Flexible and easy to maintain

· High performance

· SEO friendly

· Open, rich toolset

· Strong community

· Easy transition to React Native.

**3.What is Virtual dom? What are the differences between virtual and real dom?**

In simple words, virtual DOM is just a copy of the original DOM kept in the memory and synced with the real DOM by libraries such as ReactDOM. This process is called Reconciliation. So when there is a update in the virtual DOM, react compares the virtual DOM with a snapshot of the virtual DOM taken right before the update of the virtual DOM.With the help of this comparison React figures out which components in the UI needs to be updated. This process is called diffing. The algorithm that is used for the diffing process is called as the diffing algorithm. Once React knows which components has been updated, then it replaces the original DOM nodes with the updated DOM node.

**4.Differences between props and state?**

**State** :Use state to store the data your current page needs in your controller-view.

· state should be managed in view-controller

· state is top level component

· state is mutable

· state has worse performance

· state should not be accessed from child components

· state pass it down with props instead

· state Use props to pass data & event handlers down to child components.

**Props**:Use props to pass data & event handlers down to your child components.

· props are immutable

· props lets React do fast reference checks

· props are used to pass data down from your view-controller

· props are top level component

· props have better performance

· props use to pass data to child components

**5. What is the purpose of useState? When and why will you use it?**

useState() is an example built-in React hook that lets you use states in your functional components. This was not possible before React 16.7.The useState function is a built in hook that can be imported from the react package. It allows you to add state to your functional components. Using the useState hook inside a function component, you can create a piece of state without switching to class components.

**6. What is a context API? How does it work?**

Using the React Context API, a React application can effectively construct global variables that can be passed around to other application components. This is an alternative to "prop drilling," which is passing props down through generations from grandparents to children to parents, etc. The use of context is also promoted as an easier and more lightweight method of managing states while using Redux.

Simply using React.createContext() is sufficient. It gives you back a customer as well as a provider. The term "provider" refers to a component that, as its name suggests, gives its offspring access to the state. It will be the parent of all the components that would require that store, as well as the container for the "store." Consumer, as it happens, is a component that makes use of the state while also consuming it. There is additional information available on the documentation page for React.

**7.Difference between useEffect and useState?**

For useState() First, we have the functional component, which was a stateless component because it did not support state. Another way to put it is that a functional component does not store any state. Now, thanks to Hooks, we have the functional component while still maintaining its stateful nature.

To utilize and effect ()

To begin, while we were using stateless functional components, there were no component lifecycle hooks available. To put it another way, whenever you wish to use component lifecycle hooks, you should think about utilizing class component instead of component. We no longer need to make use of the class component in order to make use of component lifecycle hooks. Utilizing the useEffect syntax will get the job done. To put it another way, whenever we wish to use component lifecycle hooks, we already have two options: either using class component or using Hooks with useEffect. Either way, we can choose to use component lifecycle hooks.

**8.What is JSX? How does it work**

The abbreviation for JavaScript syntax extension is JSX. It is an extension to JavaScript that gives us the ability to define the object tree that React uses in a syntax that is similar to that of an HTML template. Simply put, it is an extension that is similar to XML that enables us to create JavaScript code that has the appearance of markup and have that code returned from a component.

The creation of particularly specific JavaScript objects can be simplified by using JSX, which is nothing more than syntactic sugar. When you start producing layered content, not only is it difficult to code, but it also becomes highly inconvenient to manage a codebase with this kind of structure. JSX allows you to combine the efficiency of HTML and the flexibility of JavaScript in this way.

**9.Tell us about React Component lifecycle**

The various phases that a component goes through during its lifetime are referred to as "life cycle" in the React programming language. In each and every stage of the React life cycle, a callback function is provided by React to attach functionality. In this chapter, we will learn about the life cycle of a React component, as well as the API that is associated with it.

Life cycle API

Each individual React component can be broken down into one of three steps.

Mounting -Mounting Mounting denotes the process of rendering the React component within the DOM node that has been provided.

Updating -The term "updating" refers to the process of re-rendering the React component in the specified DOM node when the state of the application is changed or updated.

Unmounting -The process of unmounting, also known as detaching, signifies the removal of the React component.

**10. What is the purpose of a custom hook? How will you create a custom hook? Give us an example.**

When we wish to communicate logic between different JavaScript methods, we can do so by utilizing something called a Custom Hook, which is a JavaScript function that we develop on our own. It makes it possible for you to reuse the same piece of code in multiple locations throughout your program.

Example:

const useDocumentTitle = (title) => {

useEffect(() => {

document.title = title;

}, [title])

}

**11.How would you optimize a react js application**

**Performance Optimization Techniques for React Apps**

1. Using Immutable Data Structures.

2. Function/Stateless Components and React.

3. Multiple Chunk Files.

4. Use React.

5. Avoid Inline Function Definition in the Render Function.

6. Throttling and Debouncing Event Action in JavaScript.

7. Avoid using Index as Key for map.

**REACT JS FLAGS 2**

**1.How will you send data from a Child Component to the parent component?**

To pass data from child to parent component in React:

· Pass a function as a prop to the Child component.

· Call the function in the Child component and pass the data as arguments.

· Access the data in the function in the Parent.

**2.What is the best way to send 4 or more props to a child component?**

When you have a React component who received more than 4 propos from its parent component, and you want to pass all of those props on to this component’s child, then you need to pass the entire props object.

So the best option is using spread operator The spread operator {...props} “spreads out” the properties of the props’ object as arguments (and thus, props) for each child component.

Example:

<DisplayAllProps {...props}></DisplayAllProps>

**3.What is Redux and what is the purpose of Redux?**

Redux is a state container that facilitates the creation of JavaScript applications that operate consistently across client, server, and native contexts and are simple to test. While it is most commonly used as a state management tool with React, it is compatible with any JavaScript framework or library. It is small at 2KB (dependencies included), thus you need not worry about it increasing the asset size of your application.

Benefits of Redux:

1. Centralization

2. Predictability

3. Time-Travel Debugging

**4. What is React native? What do you know about React Native?**

React Native is an open-source JavaScript framework developed for building applications on multiple platforms, such as iOS, Android, and the web, with the same code base. It is React-based and brings all of its splendor to mobile app development. Facebook developed both the ReactJS (web) and React Native frameworks. React Native was a Hackathon project designed to alleviate the primary pain point for the organization, which was maintaining two code bases for their app. Maintaining two code bases for such a large application? Work duplication and, at times, two distinct solutions to the same problem. React Native provides a clear solution to these issues.

**5.What are Higher order components? Give us an example.**

A function that accepts a component and returns a new component is a higher-order component. Higher-order component (HOC) is an advanced way for reusing component functionality in React.js. Reacts API does not include Higher-Order Components. They are the pattern resulting from the compositional nature of React. The component converts props into UI, whereas a higher-order component converts one component into another. Redux's connect and Relay's createContainer are HOC instances.

const EnhancedComponent = higherOrderComponent(WrappedComponent);

Whereas a component transforms props into UI, a higher-order component transforms a component into another component.

**6.Is there any reason to return something from a UseEffect hook?**

Why did we return a function from our effect? This is the optional cleanup mechanism for effects. Every effect may return a function that cleans up after it. This lets us keep the logic for adding and removing subscriptions close to each other. They’re part of the same effect!

When exactly does React clean up an effect? React performs the cleanup when the component unmounts. However, as we learned earlier, effects run for every render and not just once. This is why React also cleans up effects from the previous render before running the effects next time. We’ll discuss why this helps avoid bugs and how to opt out of this behavior in case it creates performance issues later below.

**7.How will you optimize a react application?**

**Performance Optimization Techniques for React Apps**

1. Using Immutable Data Structures.

2. Function/Stateless Components and React.

3. Multiple Chunk Files.

4. Use React.

5. Avoid Inline Function Definition in the Render Function.

6. Throttling and Debouncing Event Action in JavaScript.

7. Avoid using Index as Key for map.

**8.What are the different ways to manage state in a React Application**

There are four main types of state you need to properly manage in your React apps:

Local state: local state is data we manage in one or another component.

Global state: Global state is data we manage across multiple components.

Server state: Data that comes from an external server that must be integrated with our UI state.

URL state: Data that exists on our URLs, including the pathname and query parameters.

**9.Why do we inject dependency inside a UseEfect hook?**

dependencies argument of useEffect(callback, dependencies) lets you control when the side-effect runs:

Not provided: the side-effect runs after every rendering.

function MyComponent() {

useEffect(() => {

// Runs after EVERY rendering

});

}

B) An empty array []: the side-effect runs once after the initial rendering.

function MyComponent() {

useEffect(() => {

// Runs ONCE after initial rendering

}, []);

}

C) Has props or state values [prop1, prop2, ..., state1, state2]: the side-effect runs only when any depenendecy value changes.

function MyComponent({ prop }) {

const [state, setState] = useState('');

useEffect(() => {

// Runs ONCE after initial rendering

// and after every rendering ONLY IF `prop` or `state` changes

}, [prop, state]);

}

**10.How will you prevent re-render in react applications?**

shouldComponentUpdate is the first method used to prevent a component from rendering in React. It is a lifecycle method accessible to React class component instances. Instead of Square being a stateless functional component as it was previously:

const Square = ({ number }) => <Item>{number \* number}</Item>;

A class component can be used with the componentShouldUpdate method:

class Square extends Component {

shouldComponentUpdate(nextProps, nextState) {

...

}

render() {

return <Item>

{this.props.number \* this.props.number}</Item>;

}

}

Before re-rendering a component, the shouldComponentUpdate class method has access to the subsequent props and state, as demonstrated. By returning false from this method, you can prevent a re-render. If true is returned, the component is rerendered.

**11.Tell me some disadvantages of Reactjs**

**The pace of development:** This disadvantage is aptly described by developers Michael Jackson and Ryan Florence on Modern Web

**Poor documentation:**

The problem with documentation traces back to constant releases of new tools. Different and new libraries like Redux and Reflux are promising to accelerate the work of a library or improve the entire React ecosystem

**HTML in my JavaScript!’ – JSX as a barrier:**

ReactJS uses JSX. It’s a syntax extension, which allows mixing HTML with JavaScript. JSX has its own benefits (for instance, protecting code from injections), but some members of the development community consider JSX to be a serious disadvantage as it looks like old spaghetti code.

**Additional SEO hassle:**

There have been concerns that Google and other search engines can’t index or poorly index dynamic web pages with client-side rendering. These concerns haven’t been fully proven and there are debunking materials around

**12.Does React perform one-way data binding or two way data binding?**

React perform one way binding.

React doesn't have a mechanism to allow the HTML to change the component. The HTML can only raise events that the component responds to. The typical example is by using onChange.

the value of the <input /> is controlled entirely by the render function. The only way to update this value is from the component itself, which is done by attaching an onChange event to the <input /> which sets this.state.value to with the React component method setState. The <input /> does not have direct access to the components state, and so it cannot make changes. This is one-way binding.

**NODE JS FLAGS**

**1.What is Nodejs?**

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project! Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.A Node.js app runs in a single process, without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.

**2.Node vs JavaScript**

JavaScript is a simple programming language that runs in any browser JavaScript Engine. Whereas Node JS is an interpreter or running environment for a JavaScript programming language that holds many excesses, it requires libraries that can easily be accessed from JavaScript programming for better use.

JavaScript is normally following Java Programming language standard. There may have some different way of writing code, but at the same time, we can say it following the Java Programming language standard. Whereas node JS is written in C++ and provides a V8 engine base browser javascript running engine, it helps us run a written javascript program in any browser environment.

**3.Nodejs single threaded or multi threaded?**

Node.js architecture is single-threaded

Node.js was created explicitly as an experiment in async processing. The theory was that doing async processing on a single thread could provide more performance and scalability under typical web loads than the typical thread-based implementation.

**4.Is Nodejs working as blocking or nonblocking?**

Node.js uses an event-driven, asynchronous non-blocking I/O model

Blocking refers to operations that block further execution until that operation finishes while non-blocking refers to code that doesn’t block execution. Or as Node.js docs puts it, blocking is when the execution of additional JavaScript in the Node.js process must wait until a non-JavaScript operation completes.

**5.What is Npm**

NPM – or "Node Package Manager" – is the default package manager for JavaScript's runtime Node.js.

NPM consists of two main parts:

1. a CLI (command-line interface) tool for publishing and downloading packages, and

2. an online repository that hosts JavaScript packages

**6.What is the purpose of a database?**

The database is used to store the data in the organised way and it can be accessible by electronically, where you can store the large amount of data, for example if you want to find something in the general storage devices where you stored large amount of data it is difficult to retrieve the data what you want and if you want to do some operations on the data it is also difficult for us.So to overcome these type of problems with small commands in database you can retrieve what you want and also backup facilities will also be very nice for database.And also for analysing the data the database is used.If database management system is not there, then for retrieving the data from storage devices would be difficult and we have to write the programs for making the data in an organised way.

**7.Difference between SQL and Nosql? Which one is more popular?**

The five critical differences between SQL vs NoSQL are:

1. SQL databases are relational, NoSQL databases are non-relational.

2. SQL databases use structured query language and have a predefined schema. NoSQL databases have dynamic schemas for unstructured data.

3. SQL databases are vertically scalable, while NoSQL databases are horizontally scalable.

4. SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores.

5. SQL databases are better for multi-row transactions, while NoSQL is better for unstructured data like documents or JSON.

SQL is more popular

**8.Why mongodb is used as a primary database for react and node applications?**

Node.js is popularly being used in web applications because it lets the application run while it is fetching data from the backend server. It is asynchronous, event-driven and helps to build scalable web applications. Even though Node.js works well with MySQL database, the perfect combination is a NoSQL like MongoDB wherein the schema need not be well-structured. MongoDB represents the data as a collection of documents rather than tables related by foreign keys. This makes it possible for the varied types of data dealt over the internet to be stored decently and accessed in the web applications using Node.js. Another option is using CouchDB that also stores the data as JSON/BSON environment.

**9.What is Database design and database schema design ?**

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored.The main objectives of database design in DBMS are to produce logical and physical designs models of the proposed database system.The term "database schema" can refer to a visual representation of a database, a set of rules that govern a database, or to the entire set of objects belonging to a particular user. Read on to find out more about database schemas and how they are used.

**10.What will you do during server side crashed?**

If you store any state on your servers that you don't want to loose when a server is down then store your state on the two specific database servers - a master and a replica. If you don't know which database server to pick then let me know and I will ask you additional questions to help.

If a client code that is using your servers is under your control then implement all the fault tolerance logic in there. The simplest way of doing this is the "round robin" - you connect to a random server until you find a server which serves your request. Google "round robin" for more details.

If a client code is not under your control then you should use BGP load balancing. The main idea of that is that once a server is down another server gets the IP address of the first one, such as on a client side everything is perfect. For more details just google "bgp load balancing".

**11.How api works?**

APIs work by sharing data and information between applications, systems, and devices—making it possible for these things to talk with each other.Sometimes the easiest way to think about APIs is to think about a metaphor, and a common scenario that a lot of folks use is that of the customer, a waiter, and a restaurant kitchen: A customer talks to the waiter and tells the waiter what she wants. The waiter takes down the order and communicates it to the kitchen. The kitchen does their work, creating the food, and then the waiter delivers the order back to the customer.In this metaphor, a customer is like a user, who tells the waiter what she wants. The waiter is like an API, receiving the customer’s order and translating the order into easy-to-follow instructions that the kitchen then uses to fulfill that order—often following a specific set of codes, or input, that the kitchen easily recognizes. The kitchen is like a server that does the work of creating the order in the manner the customer wants it, hopefully! When the food is ready, the waiter picks up the order and delivers it to the customer.

**12.What is CRUD**

CRUD Meaning: CRUD is an acronym that comes from the world of computer programming and refers to the four functions that are considered necessary to implement a persistent storage application: create, read, update and delete. Persistent storage refers to any data storage device that retains power after the device is powered off, such as a hard disk or a solid-state drive. In contrast, random access memory and internal caching are two examples of volatile memory - they contain data that will be erased when they lose power.

**13.Get vs post**

**GET**:

GET is the simplest type of HTTP request method; the one that browsers use each time you click a link or type a URL into the address bar. It instructs the server to transmit the data identified by the URL to the client. Data should never be modified on the server side as a result of a GET request. In this sense, a GET request is read-only.

**POST**:

The POST verb is mostly utilized to create new resources. In particular, it's used to create subordinate resources. That is, subordinate to some other (e.g. parent) resource.

On successful creation, return HTTP status 201, returning a Location header with a link to the newly-created resource with the 201 HTTP status.

**14.Difference between PUT and Patch ?**

**PUT**:

PUT is most-often utilized for update capabilities, PUT-ing to a known resource URI with the request body containing the newly-updated representation of the original resource.

**PATCH**:

Patch request says that we would only send the data that we need to modify without modifying or effecting other parts of the data. Ex: if we need to update only the first name, we pass only the first name

**15.How will you secure an API**

JSON Web Token or JWT is a secure open standard way, which securely helps in transmitting all the information between two respective parties. JWT can be signed with the help of any secret key with a proper algorithm. When systems exchange confidential data, the JSON secure app is used, which helps in identifying the user without any private credentials. JWT is currently the latest technology used by app development services, which helps in securing the APIs. API is the application programming interface, which helps in communication between two applications. API is used in the iPhone applications, which can be designed by the iPhone app Development Company. When you use an application and you get a response from it, it is due to this software programming interface. Now, there is a need to secure APIs, which can be done with the help of JWT. The app development services help you to secure the API with the help of JWT.

**16.What is mongose and how it works?**

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.

**17.What is webpack?**

Webpack is a module bundler. Webpack can take care of bundling alongside a separate task runner. However, the line between bundler and task runner has become blurred thanks to community-developed webpack plugins. Sometimes these plugins are used to perform tasks that are usually done outside of webpack, such as cleaning the build directory or deploying the build although you can defer these tasks outside of webpack.