**Technical Interview QNA**

# **Algorithms**

**1)** **Can Binary Search be used for linked lists?**

Yes, Binary search is possible on the linked list if the list is ordered, and you know the count of elements in list. But while sorting the list, you can access a single element at a time through a pointer to that node i.e., either a previous node or next node.

**2)** **When does the worst case of QuickSort occur?**

The worst case occurs when the partition process always picks greatest or smallest element as pivot. If we consider above partition strategy where last element is always picked as pivot, the worst case will occur when the array is already sorted in increasing or decreasing order.

**3)** **A sorted array is rotated at some unknown point, how to efficiently search an element in it?**

First, finding the pivot point (the point where start the rotated part of the list) by comparing the half element with edge element to find out in which half the pivot point is located. Narrow down in which to halves between the pivot the element is located and start a binary search.

**4)** **Given a big string of characters, how to efficiently find the first unique character in it?**

Create a frequency hashmap and increment the value of every character key that is found. Return the first key that has the value equal to 1.

**5)** **Given a big array, how to efficiently find k’th largest element in it?**

Sort the array with any sorting algorithm that meets the complexity of n log n in a descending order. Return the kth element without counting the 0 index as the first one.

**6)** **How is an Array different from Linked List?**

An array is a collection of elements of a similar data type. A linked list is a collection of objects known as a node where node consists of two parts, i.e., data and address. Array elements store in a contiguous memory location. Linked list elements can be stored anywhere in the memory or randomly stored.

**7)** **What is Stack and where it can be used?**

A stack is a data structure used to store a collection of objects. Individual items can be added and stored in a stack using a push operation. Objects can be extracted in the order of lasted added with pop operation. It can be used to verified if the parenthesis or any type of brackets are being closed before opening a new one by pushing the opening one and popping it in case, they it finds a closing one.

**8)** **What is a Queue, how it is different from stack and how is it implemented?**

A queue is a container of objects (a linear collection) that are inserted and removed according to the first-in first-out (FIFO) principle unlike the stack that it removes the last item pushed. This can be used in a list of clients that are waiting to be served.

**9)** **What is a Linked List and What are its types?**

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list. Singly Linked List, Doubly Linked List, Circular Linked List, Doubly Circular linked list, Header Linked List.

**10)** **How to implement a stack using a queue?**

By using 2 queues, it can make sure to swap the elements, so they be in order needed to be a stack.

**11)** **How to implement a queue using a stack?**

To enqueue an item into the queue, first move all elements from the first stack to the second stack, push the item into the first stack, and finally move all elements back to the first stack. This ensures that the new item lies at the bottom of the stack and hence would be the last one to be removed. To dequeue an item from the queue, return the top item from the first stack.

**12)** **Which Data Structure Should be used for implementing LRU cache?**

We use two data structures to implement an LRU Cache.

**Queue** which is implemented using a doubly linked list. The maximum size of the queue will be equal to the total number of frames available (cache size). The most recently used pages will be near rear end and least recently pages will be near front end.

A **Hash** with page number as key and address of the corresponding queue node as value.

**13)** **How to check if a given Binary Tree is BST or not?**

All nodes in the left subtree of a node have values less than the node's value. All nodes in the right subtree of a node have values greater than the node's value. Both left and right subtrees are also binary search trees.

**HTML and CSS**

**1)** **Describe the difference between a cookie, sessionStorage and localStorage.**

SessionStorage is only available for the duration of the browser session (and is deleted when the tab or window is closed) - it does, however, survive page reloads.

If the data, you are storing needs to be available on an ongoing basis then localStorage is preferable to sessionStorage. Cookies are text files that save small pieces of information used to identify the computer as it is connected to a network.

**2)** **Describe the difference between <script>, <script async> and <script defer>.**

<script> is used to define a client-side script.

<script async>: If async is present: The script is executed asynchronously with the rest of the page (the script will be executed while the page continues the parsing) If async is not present and defer is present: The script is executed when the page has finished parsing.

<script defer> The defer attribute tells the browser to only execute the script file once the HTML document has been fully parsed.

**3)** **Why is it generally a good idea to position CSS <link>s between <head></head> and JS <script>s just before </body>? Do you know any exceptions?**

CSS is better to be placed before the body so that the elements render with the design in the moment that the page is loading. If the user’s internet is slow, the browser would show the raw version of the page. Script is hardly recommended to be placed at the end because the code may need the page’s element already rendered to work on it.

**4)** **What is progressive rendering?**

Progressive Rendering is the technique of sequentially rendering portions of a webpage in the server and streaming it to the client in parts without waiting for the whole page to rendered.

**5)** **What is CSS selector specificity and how does it work?**

If there are two or more conflicting CSS rules that point to the same element, the browser follows some rules to determine which one is most specific and therefore wins out. Think of specificity as a score/rank that determines which style declarations are ultimately applied to an element.

**6)** **What's the difference between "resetting" and "normalizing" CSS? Which would you choose, and why?**

Normalize CSS is mainly a set of styles, based on what its author thought would look good, and make it look consistent across browsers. Reset basically strips styling from elements so you have more control over the styling of everything.

**7)** **Describe z-index and how stacking context is formed.**

The Z-index is a property that allows the developer to stack elements in the CSS. It's basically a 3-d property so it allows the developer to choose how close the element appears. This is how stacking context is formed.

**8)** **What are the various clearing techniques, and which is appropriate for what context?**

Empty div method - <div style="clear:both;"></div>.

Clearfix method — Refer to the .clearfix class above.

overflow: auto or overflow: hidden method - Parent will establish a new block formatting context and expand to contains its floated children.

In large projects, I would write a utility. clearfix class and use them in places where I need it. overflow: hidden might clip children if the children is taller than the parent and is not very ideal.

**9)** **How would you approach fixing browser-specific styling issues?**

After identifying the issue and the offending browser, use a separate style sheet that only loads when that specific browser is being used. This technique requires server-side rendering though.

Use libraries like Bootstrap that already handles these styling issues for you.

Use autoprefixer to automatically add vendor prefixes to your code.

Use Reset CSS or Normalize.css.

**10)** **What are the different ways to visually hide content (and make it available only for screen readers)?**

visibility: hidden. However, the element is still in the flow of the page, and still takes up space.

width: 0; height: 0. Make the element not take up any space on the screen at all, resulting in not showing it.

position; absolute; left: -99999px. Position it outside of the screen.

text-indent: -9999px. This only works on text within the block elements.

**11)** **Have you ever used a grid system, and if so, what do you prefer?**

**12)** **How do you implement media queries or mobile specific layouts/CSS?**

**13)** **What are the advantages/disadvantages of using CSS preprocessors?**

Advantages:

CSS is made more maintainable.

Easy to write nested selectors.

Variables for consistent theming. Can share theme files across different projects.

Mixins to generate repeated CSS.

Splitting your code into multiple files. CSS files can be split up too but doing so will require a HTTP request to download each CSS file.

Disadvantages:

Requires tools for preprocessing. Re-compilation time can be slow.

**14)** **Describe pseudo-elements and discuss what they are used for.**

A CSS pseudo-element is a keyword added to a selector that lets you style a specific part of the selected element(s). They can be used for decoration (:first-line, :first-letter) or adding elements to the markup (combined with content: ...) without having to modify the markup (:before, :after).

:first-line and :first-letter can be used to decorate text.

Used in the .clearfix hack as shown above to add a zero-space element with clear: both.

Triangular arrows in tooltips use :before and :after. Encourages separation of concerns because the triangle is considered part of styling and not really the DOM, but not possible to draw a triangle with just CSS styles.

**15)** **Explain your understanding of the box model and how you would tell the browser in CSS to render your layout in different box models.**

The CSS box model is responsible for calculating:

How much space a block-level element takes up.

Whether or not borders and/or margins overlap, or collapse.

A box’s dimensions.

The box model has the following rules:

The dimensions of a block element are calculated by width, height, padding, borders, and margins.

If no height is specified, a block element will be as high as the content it contains, plus padding (unless there are floats, for which see below).

If no width is specified, a non-floated block element will expand to fit the width of its parent minus padding.

The height of an element is calculated by the content's height.

The width of an element is calculated by the content's width.

By default, paddings and borders are not part of the width and height of an element.

**16)** **What is the CSS display property, and can you give a few examples of its use?**

The display property specifies the display behavior (the type of rendering box) of an element. It can be used to hide elements for users, show them, block them, show them inline and inline-block.

**17)** **What's the difference between a relative, fixed, absolute and statically positioned element?**

Static - this is the default value, all elements are in order as they appear in the document.

Relative - the element is positioned relative to its normal position.

Absolute - the element is positioned absolutely to its first positioned parent.

Fixed - the element is positioned related to the browser window.

**18)** **Have you played around with the new CSS Flexbox or Grid specs? Could you explain them to me?**

The Flexible Box Layout Module makes it easier to design flexible responsive layout structures without using float or positioning.

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

**19)** **Can you explain the difference between coding a web site to be responsive versus using a mobile-first strategy?**

Responsive web design is a design and development philosophy whose objective is to adapt the appearance of web pages to the device that is being used to visit them.

Being mobile-first refers to the decision to shape your business with mobile users as the priority. More specifically, a mobile-first strategy can mean prioritizing your businesses' mobile app and mobile web capabilities over desktop.

**20)** **Have you ever worked with retina graphics? If so, when and what techniques did you use?**

Retina graphics lay emphasis on the double density pixels screen of its devices. There are different CSS techniques you can use… like CSS pixel. It is an abstract unit used by the browsers to draw images and other content on a web page. CSS pixels are DIPs which means they are device independent pixels. They readjust themselves according to the pixel density of the screen they are rendered in. You can also resize images in media queries for higher resolution.

**21)** **What is progressive enhancement?**

Progressive enhancement is a strategy in web design that puts emphasis on web content first. This strategy involves separating the presentation semantics from the content, with presentation being implemented in one or more optional layers, activated based on aspects of the browser or Internet connection of the user.

**22)** **What are the differences between vector graphics and raster (bitmap) graphics?**

Vector graphics are digital art that is rendered by a computer using a mathematical formula. Raster images are made up of tiny pixels, making them resolution dependent and best used for creating photos.

**23)** **What is semantic HTML?**

Semantic HTML is the use of HTML markup to reinforce the semantics, or meaning, of the information in web pages and web applications rather than merely to define its presentation or look. Semantic HTML is processed by traditional web browsers as well as by many other user agents. CSS is used to suggest its presentation to human users.

**24)** **What is accessibility? How do you make your web application the most accessible?**

Web accessibility means that websites, tools, and technologies are designed and developed so that people with disabilities can use them.

Choose a content management system that supports accessibility

Use headings correctly to organize the structure of your content

Include proper alt text for images

Give your links unique and descriptive names

Use color with care

Design your forms for accessibility

Use tables for tabular data, not for layout

Ensure that all content can be accessed with the keyboard alone in a logical way

Use ARIA roles and landmarks (but only when necessary)

Make dynamic content accessible

**JavaScript**

**1)** **What is the significance of, and reason for, wrapping the entire content of a JavaScript source file in a function block?**

In order to keep variables private and not pollute the global scope. Global variables are generally considered bad. They encourage error-prone patterns and make it more difficult to reason about your programs.

**2)** **What is the significance, and what are the benefits, of including 'use strict' at the beginning of a JavaScript source file?**

It is a literal expression used to indicate that the code should be executed in "strict mode". It helps you to write cleaner code by changing previously accepted "bad syntax" into real errors; like preventing you from using undeclared variables, or using reserved words as variable names.

**3)** **What is NaN? What is its type? How can you reliably test if a value is equal to NaN?**

It’s a Global property that represents a "Not-a-Number" value. It’s type is “number”. And you can test if a value is equal to NaN with: *isNaN(“variableOrCharacter”)*

**4)** **What is a “closure” in JavaScript? Provide an example.**

A closure is the bundling of functions where you have access to an outer function’s scope from an inner function.

*function init() {*

*var name = 'Mozilla'; // name is a local variable created by init*

*function displayName() { // displayName() is the inner function, a closure*

*alert(name); // use variable declared in the parent function*

*}*

*displayName();*

*}*

*init();*

**5)** **How do you clone an object?**

Using the assignment operator (=): *let object = { a: 2, b: 3 }; let clone = object;*

**6)** **How do you add an element at the begining of an array? How do you add one at the end?**

Add at the beginning: *array.unshift(element);* Add at the beginning: *array.push(element);*

**7)** **What is the difference between undefined and not defined in JavaScript?**

Undefined is a Global property that represents the primitive value ‘undefined’. Not defined is a term typically used when trying to use a variable that has not been previously defined.

A variable that has not been assigned a value is of type undefined. A method or statement also returns undefined if the variable that is being evaluated does not have an assigned value. A function returns undefined if a value was not returned.

**8)** **How do you check if an object is an array or not?**

With the following method: *Array.isArray(objectToCheck);*

**9)** **What is function hoisting in JavaScript?**

Hoisting basically allows variables to be initialized and used before they are declared.

JavaScript Hoisting refers to the process whereby the compiler allocates memory for variable and function declarations prior to execution of the code. Declarations that are made using var are initialized with a default value of undefined. Declarations made using let and const are not initialized as part of hoisting.

**10)** **Explain how `this` works in JavaScript**

When this is defined in a global context, this refers by default to a Global object. In case of browsers, this global object is the window object.

In Node.js, the global object is a special object called global. This means that, in a global scope, this will refer to this global.

You can set the value of this when you invoke a function so it is not undefined. To do this you can use call(), apply() or bind() methods. This is called “explicit function binding”. When you use one of these methods you pass the value of this as an argument. The first two, call() and apply() are almost the same. The difference is that apply() accepts a list of arguments while the call() accepts arguments array. apply() also allows you to use an array literal.

The bind() method to create a new “bound” function. After that, you invoke the new “bound” function, not the original. Now, the value of this will be what you wanted it to be. USed for a local scope.

**11)** **Explain how prototypal inheritance works**

The core idea of Prototypal Inheritance is that an object can point to another object and inherit all its properties. The main purpose is to allow multiple instances of an object to share common properties, hence, the Singleton Pattern.

**12)** **Explain why the following doesn't work as an IIFE: "function foo(){ }();". What needs to be changed to properly make it an IIFE?**

The brackets have to be inside of the first set of parentheses which is the Grouping Operator; like so: *"( function foo() { } )();"*. And it doesn’t work like that because the Grouping Operator encloses the lexical scope, prevents accessing variables within the IIFE idiom as well as polluting the global scope.

**13)** **What's the difference between a variable that is: null, undefined or undeclared? How would you go about checking for any of these states?**

A variable is undeclared if it has not been declared with an appropriate keyword (i.e. var, let or const).

A variable is undefined if it hasn't been assigned a value. undefined is a primitive data type in JavaScript and represents the absence of a value, intentional or otherwise.

A variable is assigned a value of null like any other value. null is also primitive data type in JavaScript and always represents the intentional absence of a value.

For undefined and null, with a conditional (==) as they are both falsy values. For undefined with: *typeof myVariable === 'undefined'*

**14)** **Can you describe the main difference between a .forEach loop and a .map() loop and why you would pick one versus the other?**

The main difference is that the map method returns an array with the results of calling a provided function on every element in the calling array. When you need to populate a new or existing array with something done to the values of another array you use best .map.

**15)** **What's a typical use case for anonymous functions?**

When you are only going to call them once.

**16)** **How do you organize your code? (module pattern, classical inheritance?)**

An option I mostly use is Object Literal Notation for encapsulating and organizing my code, but upon further readings, Module Pattern using Anonymous Closures, Global Import, and Module Export have sparked my interest as it provides more features for private and public var/methods. It still uses object literal but as to return values from the scoping function.

**17)** **Difference between: function Person(){}, var person = Person(), and var person = new Person()?**

Function Declaration: *function Person(){}*, declares a function statement but does not execute, however, it does get registered into the global namespace.

Function Expression: *var person = Person()*, defines a variable ‘var person’ and contains a value reference to a ‘Person’ function. Any JavaScript Expressions (including Function Expressions) always returns a value.

Function Constructor: *var person = new Person()*, this instantiates a new object of the Person class constructor. A function declaration is just a regular function unless it has been instantiated, it then becomes a class constructor.

**18)** **What's the difference between .call and .apply?**

The difference is that apply lets you invoke the function with arguments as an array; call requires the parameters be listed explicitly. A useful mnemonic is *"A for array and C for comma."*

**19)** **Explain Function.prototype.bind.**

bind returns a function that provides a new this and prepends arguments, and works the same on all functions. The simplest possible semantic.

**20)** **When would you use document.write()?**

**21)** **Explain Ajax in as much detail as possible.**

**22)** **What are the advantages and disadvantages of using Ajax?**

**23)** **What's the difference between an "attribute" and a "property"?**

**24)** **What is the difference between == and ===?**

**25)** **Create a for loop that iterates up to 100 while outputting "fizz" at multiples of 3, "buzz" at multiples of 5 and "fizzbuzz" at multiples of 3 and 5**

**26)** **Why is it, in general, a good idea to leave the global scope of a website as-is and never touch it?**

**27)** **Explain what a single page app is and how to make one SEO-friendly.**

**28)** **What are the pros and cons of using Promises instead of callbacks?**

**29)** **What tools and techniques do you use debugging JavaScript code?**

**30)** **What language constructions do you use for iterating over object properties and array items?**

**31)** **Explain the difference between mutable and immutable objects.**

**32)** **Explain the difference between synchronous and asynchronous functions.**

**33)** **What is event loop? What is the difference between call stack and task queue?**

**34)** **What are the differences between variables created using let, var or const?**

**35)** **What are the differences between ES6 class and ES5 function constructors?**

**36)** **What is the definition of a higher-order function?**

**37)** **Can you give an example of a curry function and why this syntax offers an advantage?**

**38)** **Can you describe the Document Object Model in JavaScript?**

**39)** **What is the difference between function scope and block scope in JavaScript?**

**40)** **What will this do and why? var foo = 10 + '20';**

**Ruby**

**1)** **Explain each of the following operators and how and when they should be used: ==, ===, eql?, equal?.**

The == operator, also known as equality or double equal, will return true if both objects are equal and false if they are not.

The eql? method returns true if obj and other refer to the same hash key. This is used by Hash to test members for equality.

Unlike the == operator which tests if both operands are equal, the equal method checks if the two operands refer to the same object.

The === returns true if the object on the right “belongs to” or “is a member of” the object on the left. For instance, it can be used to test if an object is an instance of a class (or one of its subclasses).

**2)** **Can you call a private method outside a Ruby class using its object?**

No, it is not possible to call private methods. Those are exclusive to use by other methods inside of the class.

**3)** **What is the difference between extend and include in Ruby?**

Include is used for importing module code. Ruby will throw an error when we try to access the methods of import module with the class directly because it gets imported as a subclass for the superclass. So, the only way is to access it through the instance of the class.

Extend is also used to importing module code but extends import them as class methods. Ruby will throw an error when we try to access methods of import module with the instance of the class because the module gets import to the superclass just as the instance of the extended module. So, the only way is to access it through the class definition.

**4)** **How do you remove nil values in array using Ruby?**

Using the filter\_map method we could leave just the values that are different to nil.

**5)** **What is the difference between the Object methods clone and dup?**

Clone copies any singleton methods of an object but Dup does not support this.

**6)** **Ruby provides four types of variables. List them and provide a brief explanation for each.**

Local Variables: These variables are local to the code construct in which they are declared. A local variable is only accessible within the block of its initialization. Local variables are not available outside the method. There is no need to initialize the local variables.

Instance Variables: They are similar to Class variables but their values are local to specific instances of an object.

Class Variables: A class variable belongs to the class and it is a characteristic of a class. They need to be initialized before use. Another way of thinking about class variables is as global variables within the context of a single class. A class variable is shared by all the descendants of the class. An uninitialized class variable will result in an error.

Global Variables: If you want to have a single variable, which is available across classes, you need to define a global variable. Its scope is global, means it can be accessed from anywhere in a program.

**7)** **What are some advantages of using Ruby?**

Cost-effective. The Ruby on Rails framework is 100% free and runs on Linux, which is an open-source framework.

Built on Model-View-Controller (MVC) architecture

Easy to manage changes.

Performance.

Flexibility.

Productivity.

**8)** **Name the three levels of access control for Ruby methods.**

Public, private, protected.

**9)** **Explain the role of modules and mixins in Ruby.**

Mixins in Ruby allows modules to access instance methods of another one using the include method. Mixins provides a controlled way of adding functionality to classes. The code in the mixin starts to interact with code in the class. In Ruby, a code wrapped up in a module is called mixins that a class can include or extend.

**10)** **What are blocks and procs?**

Ruby blocks are little anonymous functions that can be passed into methods.

A Proc object is an encapsulation of a block of code, which can be stored in a local variable, passed to a method or another Proc, and can be called.

**11)** **What is a class?**

A class is a blueprint from which objects are created. The object is also called as an instance of a class.

**12)** **What is the difference between a class and a module?**

Modules are collections of methods and constants. Classes may generate instances (objects), and have per-instance state (instance variables). Modules may be mixed in to classes and other modules.

**13)** **How would you declare and use a constructor in Ruby?**

Is like declaring a method inside a class but it has to be called ‘initialize’. It can include the parameters needed to create the class object and save the data that the object will use. To create an object with the constructor we only have to use the new word before writing the class name and include the parameters that the constructor needs.

**14)** **How would you create getter and setter methods in Ruby?**

By creating 2 methods with the name of the attribute. The one with parameters would be the setter and the one without parameters would be the getter.

**15)** **Describe the difference between class and instance variables?**

**16)** **What does ‘self’ mean?**

**17)** **Explain how (almost) everything is an object in Ruby.**

**18)** **Explain what singleton methods are. What is Eigenclass in Ruby?**

**19)** **What is RubyGems? How does it work?**

**20)** **Can you give me some examples of your favorite gems besides Ruby on Rails?**

**Ruby on Rails**

**1)** **What is Asset Pipeline?**

**2)** **Describe CRUD verbs and actions.**

**3)** **How should you test routes?**

**4)** **How should you use filters in controllers?**

**5)** **What do we need to test in controllers?**

**6)** **How should you use content\_for and yield?**

**7)** **Explain the Active Record pattern.**

**8)** **What is Object-Relational Mapping?**

**9)** **Explain the Migrations mechanism.**

**10)** **Describe types of associations in Active Record.**

**11)** **Explain what a session mechanism is. How does it work?**

**12)** **What is the difference between SQL Injection and CSS Injection?**

**13)** **Why do we need to use HTTPS instead of HTTP?**

**14)** **What is unit testing (in classical terms)?**

**15)** **What are your favorite tools for writing unit tests and why?**

**16)** **Explain the MVC pattern.**

**React and Redux**

**1)** **Explain the Virtual DOM, and a pragmatic overview of how React renders it to the DOM.**

**2)** **Explain the standard JavaScript toolchain, transpilation (via Babel or other compilers), JSX, and these items’ significance in recent development. What sort of tools might you use in the build steps to optimize the compiled output React code?**

**3)** **What are pure functional Components?**

**4)** **How might React handle or restrict Props to certain types, or require certain Props to exist?**

**5)** **Which feature can we use to cause a component to render only when its ID changes?**

**6)** **What is React?**

**7)** **List some of the major advantages of React.**

**8)** **What are the limitations of React?**

**9)** **What is JSX?**

**10)** **Why can’t browsers read JSX?**

**11)** **What do you understand from “In React, everything is a component.”?**

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

**13)** **What is Props?**

**14)** **What is a state in React and how is it used?**

**15)** **What is an event in React?**

**16)** **Explain Flux.**

**17)** **What is Redux?**

**18)** **In Redux, what do you understand by “Single source of truth”?**

**19)** **Explain the role of Reducer.**

**20)** **What is the significance of Store in Redux?**

**System Design**

**1)** **How would you design a URL Shortener?**

**2)** **How would you design a Collaborative Editor?**

**3)** **How would you design a Photo Sharing App?**

**4)** **How would you design a Social Network Feed?**

**5)** **How would you design a Trending Algorithm?**

**6)** **How would you design a Facebook Chat?**

**7)** **How would you design a Key Value Store?**

**8)** **How would you design a Recommendation System?**

**9)** **How would you design a Cache System?**

**10)** **How would you design a E-commerce Website?**

**11)** **How would you design a Web Crawler?**

**12)** **How would you design a YouTube clone?**

**13)** **How would you design a Hit Counter?**