**1. What is the difference between == and === in JavaScript?**

== is just comparing the two values and if they are of different types, type conversion is done

=== compares the values and also their types, no type conversion is done.

Ex: (8 == 1) //Returns false

(1 == 1) //Returns true

(1 === '1') //Returns false

(1 === 1) //Returns true

**2. What is Function Call and what is Function Extend?**

The code in a function is not executed when the function is **defined**. It is executed when the function is **invoked** or Called.

function myFunction(a, b) {  
    return a \* b;  
}  
myFunction(10, 2);

Function Extend is to extend the initialize function.

window.onload = init();

function init(){

something();

}

function extends init(){

doSomething();

}

**3. What is the difference between for, while, do while and for-in loops?**

**For:** For loop is used when we know how many times we want to loop. When we have some sort of counter

for (*statement 1*;*statement 2*;*statement 3*) {  
    *code block to be executed*  
}

**While:** While loop is used when we are not sure how many times we are going to loop until the condition is true.

while (condition){  
*code block to be executed*  
}

**Do While:** when we want to loop at least once without checking the condition is true.

do {  
*code block to be executed*}  
while (condition);

**For-In:** when we are looping over the properties of an object.

*for (var* in *object) {  
    code block to be executed  
}*

**4. What are slice, splice and push?**

**Slice()** method selects the elements starting at the given start argument, and ends at, but does not include, the given end argument.

*array*.push(item1, item2, ..., itemX)

**Splice()** method adds/removes items to/from an array, and returns the removed items.

*array*.splice(index,howmany,item1,.....,itemX)

* index: It specifies the at what position we have to add/remove the items and we have to use the negative values when we are coming from the end of the array.
* Howmany: No of items to be removed
* Item1,…,itemX : new items that are added to the array.

**Push()** method adds new items to the end of an array, and returns the new length. It adds new items to the end of an array, and returns the new length.

*array*.push(item1, item2, ..., itemX)

**5. In a given paragraph I want to identify all the possible occurrences of "the" please mention the pseudo code for that?**

String, x, count

for(i=0;i<string.length;i++)

{

x=string.search(“the”);

If(x!= -1) {

Count++;

String=substring(x+4,string.length);

}

}

**6. What are recursive functions?**

Recursive function is a function that makes a call to itself. To prevent infinite recursion, you need an if-else statement where one branch makes a recursive call, and the other branch does not.

**7. Check given a string is palindrome or not by using recursive functions?**

<html>

<h2>Palindrome</h2>

<body>

<script>

function palindrome(myString)

{

var Char = myString.replace(/[^A-Z0-9]/ig, "").toLowerCase();

var checkPalindrome = Char.split('').reverse().join('');

if(Char === checkPalindrome)

{

document.write("<div>"+ myString + " is a Palindrome <div>");

}

else

{

document.write("<div>" + myString + " is not a Palindrome </div>");

}

}

palindrome('"I saw you,who?"');

palindrome('"121"');

palindrome('"Ramya"');

palindrome('"2,3,4"');

</script>

</body>

</html>

**8. What are immediate functions and give the syntax?**

An immediate function is a function that executes as soon as it is defined.

(function() {

var days = ['Sun','Mon'];

alert(msg);

}());

**9. What is eval in javascript?**

The eval() function evaluates or executes an argument, expression, statements.

eval(*string*)

var x = 10;  
var y = 20;  
var a = eval("x \* y") + "<br>";  
var b = eval("2 + 2") + "<br>";  
var res = a + b ;

**10. var arr=[{"abc":1234},{"def":3456"}]... now can i use arr.abc?, arr.def?**

Yes, Javascript object literals are written as **name:value** pairs(properties).

**11. what are object literals and what is the advantage/disadvantage of using object literals vs objects?**

A JavaScript object literal is a comma-separated list of name-value pairs wrapped in curly braces. Object literals minimize the use of global variables when combining code.

Advantages:

* No need to invoke constructors directly or maintain the correct order of arguments passed to functions.
* Object literals are also useful for unobtrusive event handling
* They can hold the data that would otherwise be passed in function calls from HTML event handler attributes.

Disadvantages:

* If you are unfamiliar with the syntax, it can be very easy to introduce errors which cause the code to stop working.

Using new Object() allows you to pass another object. The obvious outcome is that the newly created object will be set to the same reference. The function will set the type accordingly. If the object is a copy of a child class of object, we could add the property without the type conversion.

**12. What is the use of "this" keyword in Java Script?**

**This:**

* When used in a function, is the object that "owns" the function.
* When used in an object, is the object itself.
* When used in an object constructor does not have a value. It is only a substitute for the new object.
* The value of this will become the new object when the constructor is used to create an object.

**13. What are events in Java Script what kind of events available in js?**

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page. When the page loads, it is called an event. When the user clicks a button, that click too is an event.

* onclick Event Type - when a user clicks
* onsubmit Event type - when you try to submit a form(form validation)
* onmouseover and onmouseout - when you bring your mouse over any element and when you move your mouse out from that element.

**14. How many ways an event can be triggered?**

When events happen to an HTML element in a web page, it checks to see if any event handlers are attached to it. There are two types of event order: ***event capturing* and *event bubbling*.**

**15. What is event bubbling in javascript?**

The concept of event bubbling deal with situations where a single event, such as a mouse click, may be handled by two or more event handlers defined at different levels of the Document Object Model (DOM) hierarchy. The event bubbles up from parent to parent until it is handled, or until it reaches the document object.

event.bubbles

Return Value: A Boolean, indicating whether the specified event is a bubbling event.

**<div>**

**<ul>**

**<li></li>**

**</ul>**

**</div>**

In the bubbling model, the event will be first handled by the li, then by the ul, and at last by the div element.

**16. event.preventDefault, event.stopPropagation?**

**Event.preventDefault**

The preventDefault method prevents an event from carrying out its default functionality. For example, you would use preventDefault on an A element to stop clicking that element from leaving the current page:

While the element's default functionality is bricked, the event continues to bubble up the DOM.

**Event.stopPropagation**

The second method, stopPropagation, allows the event's default functionality to happen but prevents the event from propagating. stopPropagation effectively stops parent elements from knowing about a given event on its child.

**17. What are "args" in javascript write a simple programs to use the args?**

In JavaScript, a function may be called with any number of arguments, no matter how many of them are listed. There is a special pseudo-array inside each function called [arguments](https://developer.mozilla.org/en/JavaScript/Reference/functions_and_function_scope/arguments). It contains all parameters by their number: arguments[0], arguments[1] etc.

function disp() {

for(var i=0; i<arguments.length; i++) {

alert("Hello, " + arguments[i])

}

}

disp("Ramya", "G");

**18. How many was an control can be accessed in javascript?// getElementById etc...**

The most common way to access an HTML element is to use the id of the element. The easiest way to get the content of an element is by using the **innerHTML** property.

document.getElementById(id) Find an element by element id

document.getElementsByTagName(name) Find elements by tag name

document.getElementsByClassName(name) Find elements by class name

element.innerHTML = new html content Change the inner HTML of an element

element.attribute = new value Change the attribute value of an HTML element

element.setAttribute(attribute, value) Change the attribute value of an HTML element

element.style.property = new style Change the style of an HTML element

document.createElement(element) Create an HTML element

document.removeChild(element) Remove an HTML element

document.appendChild(element) Add an HTML element

document.replaceChild(element) Replace an HTML element

document.write(text) Write into the HTML output stream

**19. How to change the css property of a control by using javascript?**

document.getElementById('id').style.width = value;

var sheet = document.createElement('style')

sheet.innerHTML = "div {border: 2px solid black; background-color: blue;}";

document.body.appendChild(sheet);

**20. How to add a class dynamically to a control by using javascript?**

document.getElementById("MyElement").className += " MyClass"

**Questions on CSS**

**## What are the different new features of HTML5.**

<!DOCTYPE html>

<meta charset="UTF-8">

HTML5 New Semantic/Structural Elements

* <article> Defines an article in the document
* <aside> Defines content aside from the page content
* <bdi> Defines a part of text that might be formatted in a different direction from other text
* <details> Defines additional details that the user can view or hide
* <dialog> Defines a dialog box or window
* <figcaption> Defines a caption for a <figure> element
* <figure> Defines self-contained content, like illustrations, diagrams, photos, code listings, etc.
* <footer> Defines a footer for the document or a section
* <header> Defines a header for the document or a section
* <main> Defines the main content of a document
* <mark> Defines marked or highlighted text
* <menuitem> Defines a command/menu item that the user can invoke from a popup menu
* <meter> Defines a scalar measurement within a known range (a gauge)
* <nav> Defines navigation links in the document
* <progress> Defines the progress of a task
* <rp> Defines what to show in browsers that do not support ruby annotations
* <rt> Defines an explanation/pronunciation of characters (for East Asian typography)
* <ruby> Defines a ruby annotation (for East Asian typography)
* <section> Defines a section in the document
* <summary> Defines a visible heading for a <details> element
* <time> Defines a date/time
* <wbr> Defines a possible line-break

New Form Elements

* <datalist> Defines pre-defined options for input controls
* <keygen> Defines a key-pair generator field (for forms)
* <output> Defines the result of a calculation

New Input Types

Color, date, datetime, datetime-local, email, month, number, range, search, tel, time, url, week

New Input Attributes

Autocomplete, autofocus, form, formaction, formenctype, formmethod, formnovalidate, formtarget, height and width, list, min and max, multiple, pattern (regexp), placeholder, required, step

HTML5 - New Attribute Syntax

* Empty <input type="text" value="John" disabled>
* Unquoted <input type="text" value=John>
* Double-quoted <input type="text" value="John Doe">
* Single-quoted <input type="text" value='John Doe'>

HTML5 Graphics

* <canvas> Defines graphic drawing using JavaScript
* <svg> Defines graphic drawing using SVG

New Media Elements

* <audio> Defines sound or music content
* <embed> Defines containers for external applications (like plug-ins)
* <source> Defines sources for <video> and <audio>
* <track> Defines tracks for <video> and <audio>
* <video> Defines video or movie content

**## What are the new features available in CSS3**

* Selectors
  + attribute selectors
* Box Model
  + overflow-x, overflow-y
* Backgrounds and Borders
  + background-origin and background-clip
  + background-size
  + multiple backgrounds
  + border-color
  + border-image
  + border-radius
  + box-shadow
* Image Values and Replaced Content
* Text Effects
  + text-shadow
  + text-overflow
  + word-wrap
* 2D/3D Transformations
* Animations
* Multiple Column Layout
* User Interface
  + box-sizing
  + resize
  + outline
  + nav-top, nav-right, nav-bottom, nav-left

Color

* HSL colors
* HSLA colors
* opacity
* RGBA colors

Other modules

* CSS3 Transitions
* media queries
* multi-column layout
* Web fonts
* speech

**1. What is the box model?**

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.

* **Content** - The content of the box, where text and images appear
* **Padding** - Clears an area around the content. The padding is transparent
* **Border** - A border that goes around the padding and content
* **Margin** - Clears an area outside the border. The margin is transparent

**2. What is the problem with float and how you are going to rectify the float issues?**

Problems with Floats

* Pushdown is a symptom of an element inside a floated item being wider than the float itself (typically an image). Most browsers will render the image outside the float, but not have the part sticking out affect other layout. IE will expand the float to contain the image, often drastically affecting layout.
* Double Margin Bug - Another thing to remember when dealing with IE 6 is that if you apply a margin in the same direction as the float, it will double the margin.

Quick fix: set display: inline on the float

* The 3px Jog is when text that is up next to a floated element is mysteriously kicked away by 3px like a weird force field around the float.

Quick fix: set a width or height on the affected text.

* In IE 7, the Bottom Margin Bug is when if a floated parent has floated children inside it, bottom margin on those children is ignored by the parent.

Quick fix: using bottom padding on the parent instead.

**3. What is a grid layout?**

CSS Grid Layout is a whole module and not a single property, it gathers a bunch of properties meant to be used together. Some are meant to be set on the parent (grid) element, others on the children.

* grid-row / grid-column
* grid-row-align / grid-column-align
* grid-row-span / grid-column-span
* grid-rows / grid-columns

display: grid;

grid-columns: 200px 1% 1fr;

grid-rows: auto 15px auto 15px auto;

**4. What are SVG?**

* SVG stands for Scalable Vector Graphics
* SVG is used to define vector-based graphics for the Web
* SVG defines the graphics in XML format
* SVG graphics do NOT lose any quality if they are zoomed or resized
* Every element and every attribute in SVG files can be animated
* SVG is a W3C recommendation
* SVG integrates with other W3C standards such as the DOM and XSL

Advantages of using SVG over other image formats (like JPEG and GIF) are:

* SVG images can be created and edited with any text editor
* SVG images can be searched, indexed, scripted, and compressed
* SVG images are scalable
* SVG images can be printed with high quality at any resolution
* SVG images are zoomable (and the image can be zoomed without degradation)
* SVG is an open standard
* SVG files are pure XML

**5. What is a 2-D transform ?**

* translate() : The translate() method moves an element from its current position

translate(50px,100px);

* rotate() : The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

rotate(20deg);

* scale() : The scale() method increases or decreases the size of an element.

scale(2,3);

* skewX(): The skewX() method skews an element along the X-axis by the given angle.

skewX(20deg);

* skewY() : The skewY() method skews an element along the Y-axis by the given angle.

skewY(20deg);

* matrix() : The matrix() method combines all the 2D transform methods into one.

matrix(1, -0.3, 0, 1, 0, 0);

**6. What is a 3-D transform ?**

* rotateX()
* rotateY()
* rotateZ()
* transform Applies a 2D or 3D transformation to an element
* transform-origin Allows you to change the position on transformed elements
* transform-style Specifies how nested elements are rendered in 3D space
* perspective Specifies the perspective on how 3D elements are viewed
* perspective-origin Specifies the bottom position of 3D elements
* backface-visibility Defines whether or not an element should be visible when not facing the screen

**7. What are media queiries?**

A **media query** consists of a media type and at least one expression that limits the style sheets' scope by using media features, such as

* width and height of the viewport
* width and height of the device
* orientation (is the tablet/phone in landscape or portrait mode?)
* resolution

<link rel="stylesheet" media="(max-width: 800px)" href="example.css" />

<style>

@media (max-width: 600px) {

.facet\_sidebar {

display: none;

}}

</style>

**8. What are form factors for the different devices remember the range?**

“Form-factor” refers to a device’s size, shape, and style (“class of device” or device profile).It includes specifying what type of device it is such as a tablet, a mini-tablet, a smartphone, a legacy phone, a feature phone (something in between old dinosaur phones and smartphones), a PC, a smart-TV / Xbox, a wearable device.

<input type=range min=10 max=30 value=10 step=1>

<input type=range multiple min=0 max=10 step="0.2" value="3,5">

**9. How to achieve responsive web layout.**

* Responsive Web Design makes your web page look good on all devices (desktops, tablets, and phones).
* Responsive Web Design is about using CSS and HTML to resize, hide, shrink, enlarge, or move the content to make it look good on any screen
* Another way to create a responsive design, is to use a responsive style sheet, like W3.CSS.

**10. What is canvas and what it will be used for?**

The HTML <canvas> element is used to draw graphics. Canvas has several methods for drawing paths, boxes, circles, text, and adding images. A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

<canvas id="myCanvas" width="200" height="100"></canvas>

* Canvas can draw colorful text, with or without animation.
* Canvas has great features for graphical data presentation with an imagery of graphs and charts.
* Canvas objects can move. Everything is possible: from simple bouncing balls to complex animations.
* Canvas can respond to JavaScript events.
* Canvas can respond to any user action (key clicks, mouse clicks, button clicks, finger movement).
* Canvas' methods for animations, offer a lot of possibilities for HTML gaming applications.