# CSS

## Position

1. Static: HTML elements are positioned static by default. Static positioned elements are not affected by the top, bottom, left, and right properties. An element is not positioned in any special way; it is always positioned according to the normal flow of the page.
2. Relative: Element is positioned relative to its normal position. Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.
3. Fixed: element gets positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.
4. Absolute: Element is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed). However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
5. Sticky: Element is positioned based on the user's scroll position. A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

## Display

1. Block-Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width
2. None- The element is completely removed
3. Inline- Displays an element as an inline element (like <span>). Any height and width properties will have no effect
4. Inline-block- Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values
5. Initial- Sets this property to its default value.
6. Inherit- Inherits this property from its parent element.

# Animaton

1. specify CSS styles inside the @keyframes rule, using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).
2. animation-duration
3. animation-delay
4. animation-iteration-count: Infinite
5. Animation-direction:
   1. Normal
   2. Reverse
   3. Alternate
   4. Alternate-reverse
6. animation-timing-function property specifies the speed curve of the animation
   1. ease - Specifies an animation with a slow start, then fast, then end slowly (this is default)
   2. linear - Specifies an animation with the same speed from start to end
   3. ease-in - Specifies an animation with a slow start
   4. ease-out - Specifies an animation with a slow end
   5. ease-in-out - Specifies an animation with a slow start and end
   6. cubic-bezier(n,n,n,n) - Lets you define your own values in a cubic-bezier function
7. animation-fill-mode property specifies a style for the target element when the animation is not playing (before it starts, after it ends, or both).
   1. None
   2. Forward
   3. Backwards
   4. Both

# HTML

## Diff btw html 4 and html 5

<https://www.go4expert.com/articles/html4-vs-html5-comparison-t30141/>

1. Simple doctype declaration
2. Integrated multimedia elements
3. Html 5 can access users geo location
4. Not yet compatible with all browsers
5. Tags:
   1. <applet> replaced by <object>
   2. <acronym> replaced by <abbr>
   3. Usage of <hr> changed
   4. Schema removed from <meta>
   5. <table> now contains only **border** attribute, value from 0 to 1. Unlike many other attributes previously.
6. New Tags:
   1. <canvas> used to draw graphics using JS
   2. New multimedia tags:
      1. <audio>- formats mp3, wav, Ogg
      2. <video>- formats mp4, webM, Ogg
      3. <embed>
      4. <source>
      5. <track>
   3. <article> contains complete and independent data within a page
   4. <main>
   5. <mark> allows you highlight text
   6. <header>
   7. <footer>

# Javascript

## Difference between es5 and es6

1. Default arguments within functions
2. Template literals: new syntax ${name} inside backtick string.
3. Multiline string using backtick
4. Arrow functions
5. Promise
6. Const
7. symbol
8. Block scopes variables and functions
9. …rest
10. …spread
11. parseInt
12. export import
13. default export
14. Class and class inheritance
15. Object.**assign**
16. New methods:
    1. Findindex
    2. Join
    3. Startwith
    4. Endswith
    5. Includes
    6. isNan
    7. isFinite
    8. isSafeInteger
    9. math.trunc
    10. math.sign
17. numer, currency, datetime formatting

## Reduce, filter, map

The **reduce()** method executes a **reducer** function (that you provide) on each member of the array resulting in a single output value.

array.reduce(*function(total, currentValue, currentIndex, arr), initialValue*)

If initialValue isn't provided, reduce() will execute the callback function starting at index 1, skipping the first index. If initialValue is provided, it will start at index 0.

The **map()** method creates a new array with the results of calling a provided function on every element in the calling array.

array.map(*function(currentValue, index, arr), thisValue*)

The **filter()** method creates a new array with all elements that pass the test implemented by the provided function.

array.filter(*function(currentValue, index, arr), thisValue*)

## GET vs POST

1. get url (query string) is cached in the browser, whereas post cannot be cached
2. get can be used to send small data,

# Important Questions

## > local storage vs session storage

Both are only read on client side

Session storage ends when the tab closes.

Local storage=browser cache, has large memory, no expiration date. Cannot be accessed by server. Storage -at least 5mb, much larger than cookie

## > cookies

Can be access by the server (including creation deletion and updation)

Size must be less than 4kb

## > steps to improve performance in webpage

1. Lazy loading
2. reduce image size using new formats such as webP jpeg XR.
3. Using CDN- (content delivery network uses distributed servers to deliver websites). Delivering thru servers that are closer to users physical location.
4. Cache- temporary storage of webpages.
5. less http calls
6. enable gzip compression
7. minify and combine files
8. asynchronous loading of jS and css files
9. deferred javascript
10. avoid using inline css
11. reduce the number of pluins used
12. reduce redirects
13. reduce external scripts
14. place js at the bottom of html, not in head.

## > css minification

involves removing unnecessary formatting, whitespace, and code.

## > css sprites

## > es6 functionality

## > cookie read / write

## > cross domain call

## > jsonp

## > GET vs POST vs PUT

## > Ajax

## > Promise

## > callback

## > event loop

## > closures

## > js hoisting

## > user level vent

## > ways of calling function

## > self invoking function

## > extend class in JS

## > ptrototype in js

## > arrow function

## > data types

## > CSS

## > pseudo class

## > sprite image

## > transformation

## > position

## > display, visibility

## > flex in css

## > box model

## > align div in center

## > React

## > Life cycle

## > Pure component

## > Why react

## > react vs angular

## > virtual dom

## > store

## > redux

## > webpack

## > Map

# Queries

## 1

**function** quux **(**strings**,** **...**values**)** **{**

strings**[**0**]** **===** "foo\n"

strings**[**1**]** **===** "bar"

strings**.**raw**[**0**]** **===** "foo\\n"

strings**.**raw**[**1**]** **===** "bar"

values**[**0**]** **===** 42

**}**

quux`foo\n$**{** 42 **}**bar`

String**.**raw`foo\n$**{** 42 **}**bar` **===** "foo\\n42bar"

# TO DO Sun, Mon(Javascript)

## Sorting Algorithm:

Bubble- compares 2 adjacent values and swaps

Quick – picks a mean value within the given array. Loops through the entire array placing the values less than mean value is the left array and the values greater than mean value in the right array. Then sorts the left and right array in a similar way. Lastly joins the left array, mean value and right array.

Merge- divides the array into 2 parts- left and right array. Sorts the 2 arrays individually by any method. Then checks the corresponding values in left and right arrays, pushing the smaller value into the sorted array. Once one of the left or right arrays are completed, it pushes the remaining values of the remaining array to sorted array.

Insertion- starts with the 1st value checks it with 0th value and swaps if required. Then picks 2nd value and compares it with 1st and 0th values and shifts 2nd val to the required position. Continues till the end of the array in a similar way.

Tree

Binary search tree

Depth first search

Breadth first search

Preorder

Postorder

Inorder

Stack-

Array

List(Queue)

Array

Geekforgeek

Dontknowjs

# Architecture based questions

1. how does browser read files file js, html, css
2. what are the checks to make a library