JS questions

1. null vs undefined

Question: What are the differences between `null` and `undefined`?

Answer: JavaScript has two distinct values for nothing, null and undefined.

undefined

`undefined` means, value of the variable is not defined. JavaScript has a global variable `undefined` whose value is "undefined" and `typeof undefined` is also "undefined". Remember, undefined is not a constant or a keyword. undefined is a type with exactly one value: undefined. Assigning a new value to it does not change the value of the type undefined.

8 Ways to get Undefined:

- A declared variable without assigning any value to it.
- Implicit returns of functions due to missing return statements.
- return statements that do not explicitly return anything.
- Lookups of non-existent properties in an object.
- Function parameters that have not passed.
- Anything that has been set to the value of undefined.
- Any expression in the form of void(expression)
- The value of the global variable `undefined`

null

`null` means empty or non-existent value which is used by programmers to indicate “no value”. null is a primitive value and you can assign null to any variable. null is not an object, it is a primitive value. For example, you cannot add properties to it. Sometimes people wrongly assume that it is an object, because typeof null returns "object".

Btw, `null == undefined` ref: history of typeof null

2. == Vs ===

Question: What are the differences between `==` and `===`?

Answer: The simplest way of saying that, == will not check types and === will check whether both sides are of same type. So, == is tolerant. But under the hood it converts to its convenient type to have both in same type and then do the comparison.

=== compares the types and values. Hence, if both sides are not same type, answer is always false. For example, if you are comparing two strings, they must have identical character sets. For other primitives (number, boolean) must share the same value.

Rule for implicit coercion: Comparison by using == does implicit type conversion under the hood. And rules for implicit coercion are as follows-

- If both operands are same type use ===
- undefined == null
- If one operands is string another is number, convert string to number
- If one is boolean and another is non-boolean, convert boolean to number and then perform comparison
- While comparing a string or number to an object, try to convert the object to a primitive type and then try to compare

Be careful while comparing objects, identifiers must reference the same objects or same array.

```
var a = {a: 1};
var b = {a: 1};
a == b //false
a === b //false
```

```
var c = a;
a == c//true
a === c //true
```

Special note: NaN, null and undefined will never === another type. NaN does not even === itself.

Homework: Read confusing special cases of NaN

3. Object Equality

Question: How would you compare two objects in JavaScript?

Basics: JavaScript has two different approaches for testing equality. Primitives like strings and numbers are compared by their value, while objects like arrays, dates, and user defined objects are compared by their reference. This means it compares whether two objects are referring to the same location in memory.

Answer: Equality check will check whether two objects have same value for same property. To check that, you can get the keys for both the objects. If the number of properties doesn't match, these two objects are not equal. Secondly, you will check each property whether they have the same value. If all the properties have same value, they are equal.

```
function isEqual(a, b) {
    var aProps = Object.getOwnPropertyNames(a),
        bProps = Object.getOwnPropertyNames(b);

    if (aProps.length != bProps.length) {
        return false;
    }

    for (var i = 0; i < aProps.length; i++) {
        var propName = aProps[i];

        if (a[propName] !== b[propName]) {
            return false;
        }
    }
    return true;
}
```

Limitation:

- If one of the property values is itself an object
- If one of the property values is NaN (the only value in JavaScript that is not equal to itself?)
- If one object has a property with value undefined, while another object doesn't have the property (which thus evaluates as undefined). Btw, you can solve this problem by using hasOwnProperty

ref: object equality in JS, Underscore.js isEqual function

4. True Lies

Question: 11+ true false related questions that will trick you.

falsy: In javascript 6 things are falsy and they are- false, null, undefined, '', 0, NaN

truthy: There are only two truthy things- true and everything that is not false

True False Rapid Fire

Question: Is `'false'` is false?

Answer: No. Because, it's a string with length greater than 0. Only empty string is false.

Question: Is `' '` is false?

Answer: No. Because, it's not an empty string. There is a white space in it.

Question: What about `{}`?

Answer: true. It's an object. An object without any property is an object can't be falsy.

Question: Tell me about `[]`?

Answer: This is also truthy. It's an array object (array is child of object) is truthy.

Question: You talked bout `''` to be falsy. What about `new String('')`?

Answer: Though you are passing empty string to the string constructor, it is creating an String object. More precisely a instance of String object. It becomes an object. Hence, it is not false. so, it is truthy.

Question: Tell me about `new Boolean(false)`

Answer: truthy. As it creates an instance of the Boolean object which is an object. Object is truthy.

Question: `Boolean(function(){})`

Answer: `true` if you pass a truthy value to Boolean, it will be true.

Question: `Boolean(/foo/)`

Answer: `true`

Question: `true%1`

Answer: 0. When you are trying to find reminder of true, true becomes 1 and reminder of 1 while dividing by 1 is 0. you will get same result if you doe `false%1`

Question: `''%1`

Answer: 0

5. Truthy isn't Equal to true

Question: As `[]` is true, `[]==true` should also be true. right?

Answer: You are right about first part, `[]`, empty array is an object and object is always truthy. Hence, if you use `if([]){console.log('its true')}` you will see the log.

However, special case about `==` (double equal) is that it will do some implicit coercion.

Since left and right side of the equality are two different types, JavaScript can't compare them directly . Hence, under the hood, JavaScript will convert them to compare. first right side of the equality will be cooereced to a number and number of `true` would be 1.

After that, JavaScript implementation will try to convert `[]` by using`toPrimitive` (of JavaScript implementation). since `[].valueOf` is not primitive will use `toString` and will get `""`

Now you are comparing "" == 1 and still left and right is not same type. Hence left side will be converted again to a number and empty string will be 0.

Finally, they are of same type, you are comparing `0 === 1` which will be false.

ref: angus croll: truth and eqality in JS, ref: truthy and falsy

6. Extend Core Object

Question: How could you write a method on instance of a date which will give you next day?

Answer: I have to declare a method on the prototype of Date object. To get access to the current value of the instance of the date, i will use `this`

```
Date.prototype.nextDay = function(){
  var currentDate = this.getDate();
  return new Date(this.setDate(currentDate +1));
}
```

```
var date = new Date();
date; //Fri May 16 2014 20:47:14 GMT-0500 (Central Daylight Time)
date.nextDay();//Sat May 17 2014 20:47:14 GMT-0500 (Central Daylight Time)
```

Show Similar Question

7. bind

Question: If you want to use an arbitrary object as value of `this`, how will you do that?

Answer: There are at least three different ways to doing this by using bind, call and apply. For example, I have a method named deductMontlyFee in the object monica and by default value of this would be monica inside the method.

```
var monica = {
  name: 'Monica Geller',
  total: 400,
  deductMontlyFee: function(fee){
     this.total = this.total - fee;
     return this.name + ' remaining balance is '+ this.total;
  }
}
```

If I bind the deductMontlyFee of monica with another object `rachel` and pass rachel as first parameter of the bind function, rachel would be the value of `this`.

```
var rachel = {name: 'Rachel Green', total: 1500};
var rachelFeeDeductor = monica.deductMonthlyFee.bind(rachel, 200);

rachelFeeDeductor(); //"Rachel Green remaining balance is 1300"
rachelFeeDeductor(); //"Rachel Green remaining balance is 1100"
```

`bind` allows you to borrow a method and set the value of `this` without calling the function. It simply returns an exact copy of the function with new value of `this`. You can reuse the same function with new value of this without harming the old one.

```
var ross = {name:'Ross Geller', total:250};
var rossFeeDeductor = monica.deductMonthlyFee.bind(ross, 25);
rossFeeDeductor(); //"Ross Geller remaining balance is 225"
rossFeeDeductor(); //"Ross Geller remaining balance is 200"

rachelFeeDeductor(); //"Rachel Green remaining balance is 900"
```

Question: If an older browser dont have bind function, how will you shim it

Answer: Look at the code below and use your brain.

```
Function.prototype.bind = Function.prototype.bind || function(context){
  var self = this;
  return function(){
    return self.apply(context, arguments);
  };
}
```

8. arguments and call

Question: Write a simple function to tell whether 2 is passed as parameter or not?

Basics: arguments is a local variable, available inside all functions that provides a collection of all the arguments passed to the function. `arguments` is not an array rather an array like object. It has length but doesn't have the methods like forEach, indexOf, etc.

Answer: First convert arguments to an array by calling slice method on an array and pass arguments. After that simply use indexOf.

```
function isTwoPassed(){
  var args = Array.prototype.slice.call(arguments);
```

```
  return args.indexOf(2) != -1;
}

isTwoPassed(1,4) //false
isTowPassed(5,3,1,2) //true
```

Danger: Don't name any argument as "arguments" or dont create any local variable named as "arguments", this will override build in arguments object.

9. apply

Question: How could you use Math.max to find the max value in an array?

Answer: Use apply on Math.max and pass the array as apply takes an array of arguments. Since we are not reading anything from this or using it at all. We can simply pass null as first parameter.

```
function getMax(arr){
  return Math.max.apply(null, arr);
}
```

Extra: call and apply, both takes the value of this as first parameter. However, call takes a collection of arguments after first parameter whereas apply use an array of arguments as second parameter.

Tip: If you have weaker memory like me, you can remember apply starts with "a" and array starts with "a"

10. this

Question: What the heck is `this` in JavaScript?

Answer: At the time of execution of every function, JavaScript engine sets a property to the function called `this` which refer to the current execution context. `this` is always refer to an object and depends on how function is called. There are 7 different cases where the value of `this` varies.

1. In the global context or inside a function this refers to the window object.
2. Inside IIFE (immediate invoking function) if you use "use strict", value of this is undefined. To pass access window inside IIFE with "use strict", you have to pass this.
3. While executing a function in the context of an object, the object becomes the value of this

4. Inside a setTimeout function, the value of this is the window object.
5. If you use a constructor (by using new keyword) to create an object, the value of this will refer to the newly created object.
6. You can set the value of this to any arbitrary object by passing the object as the first parameter of bind, call or apply
7. For dom event handler, value of this would be the element that fired the event

ref: Understand JavaScript this in a crystal clear way

11. Rapid Fire

Question: What is `typeof []`

Answer: Object. Actually Array is derived from Object. If you want to check array use `Array.isArray(arr)`

Question: What is `typeof arguments`

Answer: Object. arguments are array like but not array. it has length, can access by index but can't push pop, etc.

Question: What is `2+true`

Answer: 3. The plus operator between a number and a boolean or two boolean will convert boolean to number. Hence, true converts to 1 and you get result of 2+1

Question: What is `'6'+9`

Answer: 69. If one of the operands of the plus (+) operator is string it will convert other number or boolean to string and perform a concatenation. For the same reason, `"2"+true`will return "2true"

Question: What is the value of `4+3+2+"1"`

Answer: 91 . The addition starts from the left, 4+3 results 7 and 7+2 is 9. So far, the plus operator is performing addition as both the operands are number. After that 9 + "1" where one of the operands is string and plus operator will perform concatenation.

Question: What is the value of `"1"+2+4`

Answer: "124". For this one "1" + 2 will produce "12" and "12"+4 will generates "124".

Question: What is the value of `-'34'+10`

Answer: -24. minus(-) in front of a string is an unary operator that will convert the string to a number and will make it negative. Hence, -'34' becomes, -34 and then plus (+) will perform simple addition as both the operands are number.

Question: What is the value of `+'dude'`

Answer: NaN. The plus (+) operator in front of a string is an unary operator that will try to convert the string to number. Here, JavaScript will fail to convert the "dude" to a number and will produce NaN.

Question: If you have `var y = 1, x = y = typeof x;` What is the value of x?

Answer: "undefined"

Question: for `var a = (2, 3, 5);` what is the value of a?

Answer: 5. The comma operator evaluates each of its operands (from left to right) and returns the value of the last operand. ref: MDN

Question: for `var a = (1, 5 - 1) * 2` what is the value of a?

Answer: 8

Question: What is the value of `!'bang'`

Answer: false. `!` is NOT. If you put `!` in front of truthy values, it will return false. Using !! (double bang) is a tricky way to check anything truthy or falsy by avoiding implicit type conversion of == comparison.

Question: What is the value of `parseFloat('12.3.4')`

Answer: 12.3

Question: What is the value of `Math.max([2,3,4,5]);`

Answer: NaN

Question: `3 instanceof Number`

Answer: false

Question:`null == undefined`

Answer: true

Question:What is the value of `!!function(){};`

Answer: true

Question: What is the value of `typeof bar`

Answer: "undefined"

Question: What is the value of `typeof null`

Answer: "object"

Question: If `var a = 2, b =3` what would be value of `a && b`

Answer: 3

Question: What would be consoled `var foo = 'outside'; function logIt(){console.log(foo); var foo = 'inside';} logIt();`

Answer: undefined

Question: What is `-5%2`

Answer:-1. the result of remainder always get the symbol of first operand

Question: Why `.1+.2 != .3`

Answer:

Question: `42..toString()`

Anwser: `"42"`

Question: `4.2..toString`

Anwser: //SyntaxError: Unexpected token .

Question:`42 . toString()`

Anwser: `"42"`

Question: `typeof(NaN)`

Anwser:"number"

Question: `2 in [1,2]`

Anwser: false. Because "in" returns whether a particular property/index available in the Object. In this case object has index 0 and 1 but don't have 2. Hence you get false.

12. log prefix

Question: How could you set a prefix before everything you log? for example, if you `log('my message')` it will log: "(app) my message"

Answer: Just get the arguments, convert it to an array and unshift whatever prefix you want to set. Finally, use apply to pass all the arguments to console.

```
function log(){
  var args = Array.prototype.slice.call(arguments);
  args.unshift('(app)');
  console.log.apply(console, args);
}

log('my message'); //(app) my message
log('my message', 'your message'); //(app) my message your message
```

ref: This is a really good materials, walks you through an interview process.

13. Scope and hoisting

Question: What will you see in the console for the following example?

```
var a = 1;
function b() {
    a = 10;
    return;
    function a() {}
}
b();
console.log(a);
```

Answer: 1

Explanation:

- function declaration `function a(){}` is hoisted first and it behaves like `var a = function () {};`. Hence in local scope variable `a` is created.
- If you have two variables with same name (one in global another in local), local variable always get precedence over global variable.
- When you set `a = 10;`, you are setting the local variable `a` , not the global one. Hence, the value of global variable remain same and you get, 1 in the log. ref: js hoisting/scope
- Extra: If you didnt have a function named as "a", you will see 10 in the log.

Show Tough Example

ref: watch this video or learn more about scope and hoisting

14. Closures Inside Loops

Question: Look at the code below, you have a for loop if you have setTimeout inside it. If log the loop counter inside setTimeout, what will be logged?

```
for(var i = 0; i < 10; i++) {
  setTimeout(function() {
    console.log(i);
  }, 10);
}
```

Answer: The above will not output the numbers 0 through 9, but will simply print the number 10 ten times.

Explanation: The console log is inside the anonymous function of setTimeout and setTimeout is executed when current call stack is over. So, the loop finishes and before setTimeout get the chance to execute. However, anonymous functions keep a reference to i by creating a closure. Since, the loop is already finished, the value i has been set to 10. When setTimeout use the value of i by reference, it gets the value of i as 10. Hence, you see 10 ten times.

Solution: You can fix it by avoiding closure. Just create a IIFE (Immediately Invoked Function Expression), it will create its own scope and you can pass i to the function. In that case i will be a local variable (will not refer to i in the closure) and value of the i in every loop will be preserved.

```
for(var i = 0; i < 10; i++) {
```

```
    setTimeout((function(i) {
      console.log(i);
    })(i), 10)
}
```

Alternative Solution: Look at the code below and use your brain (if any).

```
for(var i = 0; i < 10; i++) {
  setTimeout(console.log.bind(console, i), 10);
}
```

15. delete can delete but

Question: Look at the code below, I have a property in a object and I am creating a new object where I am setting it to a new value. If I delete that property what will i get if I try to access that property?

```
var myObject = {
    price: 20.99,
    get_price : function() {
        return this.price;
    }
};
var customObject = Object.create(myObject);
customObject.price = 19.99;

delete customObject.price;
console.log(customObject.get_price());
```

Answer: You will see 20.99

Explanation: This is very interesting question. When you create `object.create(myObject)` you are creating new object where the `myObject` will be the parent of the newly created object. Hence the price property will be at the parent.

When you are assigning some value to `customObject.price`, you are creating a new property on the child. This means, when you `delete customObject.price` it deletes the `price`property in the customObject (in the child). However, when you call the method getprice, first it looks for `this.price` in the child since the customObject doesn't have price property, JavaScript executor walks through the prototype chain towards the parent. Since customObject was inherited from myObject and myObject has a `price` property, the get_price method returns the price from parent. Hence, you are getting 20.99

16. Pass by value or by reference

Question: Does JavaScript pass parameter by value or by reference?

Answer: Primitive type (string, number, etc.) are passed by value and objects are passed by reference. If you change a property of the passed object, the change will be affected. However, you assign a new object to the passed object, the changes will not be reflected.

```
var num = 10,
    name = "Addy Osmani",
    obj1 = {
      value: "first value"
    },
      obj2 = {
     value: "second value"
      },
      obj3 = obj2;

function change(num, name, obj1, obj2) {
    num = num * 10;
    name = "Paul Irish";
    obj1 = obj2;
    obj2.value = "new value";
}

change(num, name, obj1, obj2);

console.log(num); // 10
console.log(name);// "Addy Osmani"
```

```
console.log(obj1.value);//"first value"
console.log(obj2.valuee);//"new value"
console.log(obj3.valuee);//"new value"
```

ref: Snook: passing by value or reference

17. memoization

Question: How could you implement cache to save calculation time for a recursive fibonacci function?

Answer: You could use poor man's memoization with a global variable. If fibonacci is already calculated it is served from the global memo array otherwise it is calculated.

```
var memo = [];

function _fibonacci(n) {
   if(memo[n]){
    return memo[n];
   }
   else if (n < 2){
     return 1;
   }else{
     fibonacci(n-2) + fibonacci(n-1);
   }
}
```

Better Implementation: implement memoization in JavaScript

18. Cache function execution

Question: How could you cache execution of any function?

Answer: You could have a method where you will pass a function and it will internally maintain a cache object where calculated value will be cached. When you will call the function with same argument, the cached value will be served.

```
function cacheFn(fn) {
    var cache={};

    return function(arg){
        if (cache[arg]){
            return cache[arg];
        }
        else{
            cache[arg] = fn(arg);
             return cache[arg];
        }
    }
}
```

Question: What if you are passing more than one argument?

Answer: First we have to use arguments to get all the parameters passed to the function and then we can generate key for the cache object. Generating key for the cache object could be tricky and one solution could be just get the all the parameters and concatenate those. Look at the code below.

```
return function(){
  var args = arguments;
  var key = [].slice.call(args).join('');
  if(cache[key]){
      return cache[key];
  }
  else{
      cache[key] = fn.apply(thi, args);
      return cache[key];
  }
}
```

19. JQuery style chaining

Question: If you need to implement the following chaining with call back, how will you implement it?

```
function slow(callback) {
    setTimeout(function(){
        if (Math.random() > 0.5) {
            return callback("Error 417",null)
        }
        callback(null, {id:123})
    },500);
}

function exec(fn){
//write your code here
}

exec(slow).done(function(data){
    console.log(data);
}).fail(function(err){
    console.log("Error: " + err);
})
```

Too much sleepy now. will try to put it up tomorrow.

```
var obj = {   // every method returns obj---------v
    first: function() { console.log('first');   return obj; },
    second: function() { console.log('second'); return obj; },
    third: function() { console.log('third');   return obj; }
}

obj.first().second().third();
```

ref: jquery like chaining or jquery like chaining

20. Animation

Question: How could you implement moveLeft animation?

Answer: Use setInterval that will place the element to the left position by some pixels in every 10ms. Hence, you will see the element moving towards the desired position. When you call setInterval, it returns a timeId. After reaching the desired location, you have to clear the time interval so that function will not be called again and again in every 10ms.

```
function moveLeft(elem, distance) {
  var left = 0;

  function frame() {
    left++;
    elem.style.left = left + 'px';

    if (left == distance)
      clearInterval(timeId)
  }

  var timeId = setInterval(frame, 10); // draw every 10ms
}
```

21. Currying

Question: How would you implement currying for any functions?

What is curring: Curring is partial invocation of a function. Currying means first few arguments of a function is pre-processed and a function is returned. The returning function can add more arguments to the curried function. It's like if you have given one or two spice to the curry and cooked little bit, still you can add further spice to it. A simple example will look like-

```
function addBase(base){
  return function(num){
    return base + num;
```

```
    }
}

var addTen = addBase(10);
addTen(5); //15
addTen(80); //90
addTen(-5); //5
```

Explanation: You are creating a closure that return a function. When you are curring with a new number, new number is added to the base you have provided.

Answer: You can add a curry method to the prototype of Function. If now parameters is passed to curry, you simply return the current function. If you have provided arguments to curry there are two steps

- Step-1: Concatenate old arguments (provided while creating curry), with new arguments (added after cooking little bit) by using `args.concat(toArray(arguments))`
- Step-2: Pass all the arguments to the function by using apply.
- Extra: Just be careful to retain the value of this.

```
Function.prototype.curry = function() {
    if (arguments.length<1) {
        return this; //nothing to curry. return function
    }
    var self = this;
    var args = toArray(arguments);
    return function() {
        return self.apply(this, args.concat(toArray(arguments)));
    }
}

function toArray(args) {
    return Array.prototype.slice.call(args);
}
```

To use it: Just pass the argument to the function.curry method and a function will be returned. Use returned function for further currying

```
function converter = function(factor, symbol, input){
  return input * factor + symbol;
}

var milesToKm = converter.curry(1.62, 'km');
mileToKm(3); //result here

var kgToLb = converter.curry(2.2, 'lb');
kgToLb(3); //result here
```

ref: Favoring Curry, curry: cooking up tastier functions

Deleted Questions

- In JavaScript `isNaN(undefined)` returns true. how could you fix it? Answer: use `function isReallyNaN (x){return x!==x;}`
- What are differences between host object and native object? read answer here
- Why extending build in JavaScript object is a bad idea? Answer: google it
- How will you get query string in a browsers URL? detail answer or `window.location.search;`
- Why does nearly every object have a toString method?
- Why Everything in JavaScript acts like an object, with the only two exceptions being null and undefined?
- How would you perform inheritance in JavaScript?
- How would you apply asynchronous call without any help of library
- What is the difference between slice, substr, substring?

1. window vs document

Question: Is there any difference between window and document?

Answer: Yes. JavaScript has a global object and everything runs under it. window is that global object that holds global variables, global functions, location, history everything is under

it. Besides, setTimeout, ajax call (XMLHttpRequest), console or localStorage are part of window.

document is also under window. document is a property of the window object. document represents the DOM and DOM is the object oriented representation of the html markup you have written. All the nodes are part of document. Hence you can use getElementById or addEventListener on document. These methods are not present in the window object.

```
window.document === document; //true
window.getElementById; //undefined
document.getElementById; //function getElementById() { [native code] }
```

ref: document and window

JS DOM Questions

2. window.onload vs document.onload

Question: Does document.onload and window.onload fire at the same time?

Answer: `window.onload` is fired when DOM is ready and all the contents including images, css, scripts, sub-frames, etc. finished loaded. This means everything is loaded.

`document.onload` is fired when DOM (DOM tree built from markup code within the document)is ready which can be prior to images and other external content is loaded.

Think about the differences between `window` and `document`, this would be easier for you.

Bonus:`document.readyState` Returns "loading" while the Document is loading, "interactive" once it is finished parsing but still loading sub-resources, and "complete" once it has loaded. The `readystatechange` event fires on the Document object when this value changes.

ref: MDN: readyState

3. attribute vs property

Question: Is attribute similar to property?

Answer: attributes are just like attribute in your html tag (XML style attribute) inside the starting tag. html attributes are exposed to the DOM via property. Hence, a property is created when DOM is parsed for each attribute in the html tag. If you change an attribute only the value of the property will change. However, the value of attribute will remain same.

Show more

4. DOM Query

Question: What are the different ways to get an element from DOM?

Answer: You can use the following methods in `document`

- `getElementById` to get a element that has the provided Id.
- `getElementsByClassName` to get a nodelist (nodelist is not an array, rather it is array-like object) by providing a class name.
- `getElementsByTagName` to get a nodelist by the provided tag name.
- `querySelector` you will pass css style selector (jquery style) and this will retrurn first matched element in the DOM.
- `querySelectorAll` will return a non-live nodelist by using depth-first pre order traversal of all the matched elements. Non-live means, any changes after selecting the elements will not be reflected.

There are two more options but I dont use them frequently-

- `getElementsByName` returns the list of elements by the provided name of the html tag
- `getElementsByTagNameNS` returns elements with particular tag name within the provided namespace

5. Fastest way to Query DOM

Question: What is the fastest way to select elements by using css selectors?

Answer: It depends on what you are selecting. If you have an ID of an element `getElmentById` is the fastest way to select an element. However, you should not have so many ID in you document to avoid style repetition. css class `getElementsByClassName` is the second quickest way to select an element

Here is the list. As we go downwards through the list, it takes more time to select elements.

- ID (#myID)
- Class (.myClass)

- Tag (div, p)
- Sibling (div+p, div~p)
- child (div>p)
- Descendant (div p)
- Universal (*)
- Attribute (input[type="checkbox"])
- Pseudo (p:first-child)

If you have crazy long selectors to select element, think about your selectors and check whether you can use a class instead.

Show deeper question

6. Use forEach on NodeList

Question: How come, I can't use forEach or similar array methods on a NodeList?

Answer: Yeah. Both array and nodeList have length and you can loop through elements but they are not same object.

Both are inherited from Object. However array has different prototype object than nodeList. forEach, map, etc are on array.prototype which doesn't exist in the NodeList.prototype object. Hence, you don't have forEach on a nodeList

```
myArray --> Array.prototype --> Object.prototype --> null
```

```
myNodeList --> NodeList.prototype --> Object.prototype --> null
```

Question: How could you solve this problem?

Answer: To solve this problem, you can simply loop through a nodeList and do whatever you wanted to inside forEach or you can call method on array to convert nodelist to an array. After that you will have access to all array.prototype methods

```
var myNodeList = document.querySelectorAll('.my-class');
var nodesArray = Array.prototype.slice.call(myNodeList);

//use array method on nodeList
nodesArray.forEach(function(el, idx){
```

```
  console.log(idx, el);
});
```

ref: MDN: nodelist

7. getElementsByAttribute

Question: If you need to implement getElementByAttribute, how would you implement it?

Answer: First, get all the elements in the DOM. You can either get it by Tag Name '*' and then check whether they have the particular attribute. In this case, even if attribute is null that will be captured. If you need to check the value, you should be able to do it by passing one extra parameter and comparing it in the if block.

```
function getElementsByAttribute(attribute){
  var allElements = document.getElementsByTagName('*'),
      elm,
      found=[];
  for (var i = 0; i < allElements.length; i++)
    {
    elm = allElements[i];
    if (elm.getAttribute(attribute))
    {
      found.push(elm);
    }
  }
  return found;
}
```

Show better algorithm

Question: Can I add this `getElementsByAttribute` to `document`?

Answer: Mr. Interviewer, please read this article: whats wrong with extending the dom

8. add class

Question: How would you add a class to an element by query selector?

Answer: Very simple. Just get the element and add the classname to the classlist.

```
function addClass(selector, className){
   var elm = document.querySelector(selector);
   if (elm){
      elm.classList.add(className);
   }
}
```

Similarly, you can implement removeClass, toggleClass and hasClass

```
//IE9+
el.classList.remove('my-class'); //removing a class
el.classList.toggle('my-class');  // toggling a class
el.classList.contains('my-class'); // checking whether class exists
```

ref: you might not need jquery.

9. Check Descendant

Question: How could I verify whether one element is child of another?

Answer: First check whether the passed parent is the direct parent of the child. If not, keep moving upward to the root of the tree.

```
function isDescendant(parent, child){
  while(child.parentNode ){
    if(child.parentNode == parent)
      return true;
    else
      child = child.parentNode;
```

```
  }
  return false;
}
```

10. innerHTML vs appendChild

Question: What is the best way to create a DOM element? Set innherHTML or use createElement?

Answer: When you set innerHTML property, browser removes all the current children of the elements. Parse the string and assign the parsed string to the element as children.

For example, if you want to add a list item to an unorderedList. You can get the element and set the innerHTML of the element like

```
var ul = document.getElementById('myList');

el.innerHTML = '<li>Only one item</li>';
```

Extra Credit: innerHTML could be slow while parsing a string. Browser has to deal with the string and if you have crappy html (invalid html).

appendChild

On the other hand, while using appendChild, you create a new Element. Since you are creating it, browser doesnt have to parse string and there is no invalid html. And you can pass the child to the parent and child will be appended to the parent.

```
var li = document.createElement("li");
var text = document.createTextNode('Only one Item');

li.appendChild(text);
ul.appendChild(li);
```

Extra Credit: If you pass a reference to be appended as child other than creating a new element. The reference you are passing will be removed from the current parent and will be added to the new parent you are appending to.

Though you would be writing couple more lines of JavaScript, it makes browsers life easier and your page faster.

11. CrateDocumentFragment

Question: What is createDocumentFragment and why you might use it?

Answer: `documentFragment` a very lightweight or minimal part of a DOM or a subtree of a DOM tree. It is very helpful when you are manipulating a part of DOM for multiple times. It becomes expensive to hit a certain portion of DOM for hundreds time. You might cause reflow for hundred times. Stay tuned for reflow.

If you are changing dom that cause expensive reflow, you can avoid it by using `documentFragment` as it is managed in the memory.

```
//bad practice. you are hitting the DOM every single time
var list = ['foo', 'bar', 'baz', ... ],
    el, text;
for (var i = 0; i < list.length; i++) {
    el = document.createElement('li');
    text = document.createTextNode(list[i]);
    el.appendChild(text);
    document.body.appendChild(el);
}
```

```
//good practice. you causing reflow one time
var fragment = document.createDocumentFragment(),
    list = ['foo', 'bar', 'baz', ...],
    el, text;
for (var i = 0; i < list.length; i++) {
    el = document.createElement('li');
    text = document.createTextNode(list[i]);
```

```
    el.appendChild(text);
    fragment.appendChild(el);
}
document.body.appendChild(fragment);
```

ref: W3: spec

12. reflow

Question: What is reflow? What causes reflow? How could you reduce reflow?

Answer: Aha... so many questions at the same time. Take a breathe Mr. Interviewer.

reflow: When you change size or position of an element in the page, all the elements after it has to change their position according to the changes you made. For example, if you change height on an element, all the elements under it has to move down in the page to accomodate a change in height. Hence, flow of the elements in the page is changed and this is called reflow.

Why reflow is bad: Reflows could be very expensive and it might have a performance hit specially in the smaller devices like phone. As it might causes changes in the portion (or whole) layout of the page.

Show more explanation

Reasons to reflow: The following cases causes reflow

- change layout (geometry of the page)
- resize the window
- change height/width of any element
- changing font
- change font size
- move DOM element (animation)
- adding or removing stylesheet
- calculating offset height or offset width
- `display: none;`

How to avoid: To avoid reflow, try to avoid doing things in the above list and some more in the below

- avoid setting multiple inline style

- apply animation to the elements that are positioned fixed or absolute
- avoid tables for layout

ref: reflow and repaint: css performance makes your JS slow

13. repaint

Question: What is repaint and when does this happen?

Answer: repaint happens when you change the look of an element without changing the size and shape. This doesn't cause reflow as geometry of the element didn't changed.

How it happens:

- change background color
- change text color
- visibility hidden

Show more explanation

If possible, prefer repaint over reflow.

14. DOM ready

Question: How could you make sure to run some javaScript when DOM is ready like `$(document).ready`?

Answer: There are four different ways-

option-1: Put your script in the last tag of html body element. DOM would be ready by the time browser hits the script tag.

option-2: Place your code inside a `DOMContentLoaded` handler. This event will be fired when DOM is completely loaded.

```
document.addEventListener('DOMContentLoaded', function(){
    //put your script here
});
```

option-3: Watch changes in the readyState of the document. And the last state is "complete" state, you can put your code there.

```
document.onreadystatechange = function () {
  if (document.readyState == "complete") {
    //put your script here
  }
}
```

option-4: Search jquery source code and copy dom ready function. In that case you have a ready function that works in the older browsers as well without loading the whole jquery library.

15. Event bubble

Question: What is event bubble? How does event flows?

Answer: To understand event bubble, you have to understand what happen when you click on anything on a page.

Where you clicked: If you have a table with multiple rows and multiple columns and you click in one of the cell

- You will think that you have clicked on the cell and browser will know that you have a click handler with the cell that will be fired immediately.
- You are completely wrong. Right away, browser doesn't know where you have clicked.

The way browser find out where you have clicked are as follows-

1. Capture: When you clicked, browser knows a click event occurred. It starts from the window (lowest level/root of your website), then goes to document, then html root tag, then body, then table... its trying to reach the the as lowest level of element as possible. This is called capture phase (phase -1).
2. Target: When browser reach the lowest level of element. In this case, you have clicked on a table cell (table data) hence target would be "td" tag. Then browser checks whether you have any click handler attached to this element. If there is any, browser executes that click hander. This is called target phase (phase -2).
3. Bubbling: After firing click hander attached to "td", browser walks toward root. One level upward and check whether there is any click handler attached with table row ("tr" element). If there is any it will execute that. Then it goes to tbody, table, body, html,

document, window. In this stage its moving upward and this is called event bubbling or bubbling phase (phase-3). Please note that, you clicked on cell but all the event handler with parent elements will be fired. This is actually very powerful (check event delegation)

Show a picture of bubble

Show live demo: Go to this link and click any layer to see event propagation.

16. Event Delegate

Question: How would you destroy multiple list items with one click handler?

Easy Solution: If you have one hundred list items that have similar event to handle. You can write one hundred event handler (actually copy paste) same code in 99 places. This works but if something changes in the event handler, you have to change in one hundred places. This doesn't call job security. This is called screwed up.

Second problem is if you want to dynamically add a new element, you have to make sure event handler is attached with the new element. More JavaScript code!

Answer: We can actually leverage event bubbling. You can have only one event handler attached to the parent element of one hundred list items. In this case, you can attach the event handler to the "ul" tag. After you click on a list item (list item does not have an event), event will bubble and "ul" has a handler. That handler will be fired.

```
<ul id="listToDestroy">
    <li><a href="#">first item</a></li>
    <li><a href="#">second item</a></li>
    <li><a href="#">third item</a></li>
    <li><a href="#">forth item</a></li>
    <li><a href="#">Fifth item</a></li>
</ul>
```

```
document.getElementById('listToDestroy').addEventListener('click', function (e) {
    var elm = e.target.parentNode;
```

```
        elm.parentNode.removeChild(elm);
        e.preventDefault();
});
```

Show demo

17. destroy button

Question: Create a button that is destroyed by clicking on it but two new buttons are created in it's place.

Answer: One way of solving is to attach a event handler with the button to destroy itself and append two. However, we can leverage event delegate. If we attach the event hander to the parent div instead of the button itself. We don't have to add the event handler to each button we create. So, we will take advantage of event bubbling.

Try to be Smart: If both the newly created button is identical to one we are destroying, why interviewer wants to destroy exactly similar one and then create one. Why not just add one. And end result would be same, you will have two two buttons.

Interviewer: I just want to see whether you can destroy any element. Make sure when you are destroying, there is no reference to the element, otherwise, you will have memory leak. If interviewer, says ok, just create one more button on click, then use your brain to change the following code.

```
<div id="doubleHolder">
  <button class="double">double</button>
</div>
```

```
document.getElementById('doubleHolder').addEventListener('click', function (e) {
   if(e.target.classList.contains('double')){
      var btn = document.createElement('button');
      btn.setAttribute('class', 'double');
      btn.innerHTML = 'double';

      var btn2 = document.createElement('button');
```

```
      btn2.setAttribute('class', 'double');
      btn2.innerHTML = 'double';

      this.appendChild(btn);
      this.appendChild(btn2);
      this.removeChild(e.target);
    }
  });
```

Show clickable demo

ref: destroy button

18. Capture all click

Question: How could you capture all clicks in a page?

Answer: You can leverage, event bubble to capture all the clicks. As all the clicks will be bubbled up to the body.

```
document.querySelector('body').addEventListener('click', function(e){
  console.log('body clicked', e.target);
});

//or
window.onclick =function(e){
  console.log('someone clicked', e.target)
}
```

However, if event bubbling is canceled by `stopPropagation()` your code will not work. Think about it.

19. All text in a page

Question: How can you get all the texts in a web page?

Answer: The easiest way to get all the text is to get the innerText of body tag.

```
document.body.innerText;
```

Question: Can you do by accessing DOM?

Answer: Make it recursive. I used closure. There could be many other ways to implement.

```
function getAllText(node){
  var allText = [];

  function getNodeText(node){
      if(node && node.childNodes && node.childNodes.length){
          for(var i = 0, len = node.childNodes.length; i<len; i++){
              getNodeText(node.childNodes[i]);
          }
      }
      else{
          allText.push(node.nodeValue);
      }
  }
  getNodeText(node);
  return allText.join('');
}
```

disclaimer: I didn't explicitly tested these two methods.

20. defer vs async

Question: What is defer and async keyword does in a script tag?

Answer: HTML parser will ignore defer and async keyword for inline script (script that does not have a src attribute).

normal: When you have a plain script tag (no defer or async keyword), parser will pause parsing, script would be downloaded and exectuted. After that parsing resume.

defer: defer keyword in the script tag will defer the execution of the script. Hence script will be executed when DOM is available. Important point is, defer is not supported by all major major browsers.

async: If possible, set the execution of the script, asynchronously. `async` keyword has no effect on inline script.

Image copied from JS script execution

Extra: async injected scripts are considered harmful

21. Difference between classical and prototypal inheritance

classical inheritance: you write a class(an abstract definition or blueprint of object) and create object instances from this. Extend this class in new class to reuse its behavior (Inheritance) so that the objects instantiated from the new class has behaviour of both classes.

Prototypical Inheritence: No classes. To make a new object you make a copy of an existing one!. The original object serves as the prototype for the copy, and the copy inherits the object's original behaviors. The new object is almost empty with basically a pointer to it's prototype (called delegation of behaviour)

22. Rapid fire

Question: How could you prevent a click on an anchor from going to the link?

Answer: `preventDefault()` inside event handler. However, this doesnt stop further propagation.

Question: How could you stop further propagation of an event?

Answer: Call `event.stopPropagation();`

Question: Can you remove an event handler from an element?

Answer: Yes. `target.removeEventListener('click', handler)`

Question: How could you run event handler in the capturing phase not in bubble phase?

Answer: There is a third (optional) parameter in addEventListener and removeEventLister. You can pass true or false to useCapture phase.

Question: How could you prevent multiple event handler to be fired for an event?

Answer: If event listeners are attached for the same type event (click, keydown, etc.) of an element for the same event type, you can call `event.stopImmediatePropagation()` in the first event handler. No other event handler will be executed.

Question: What are the cancelable events?

Answer: Go to wiki find the right most column cancelable.

Question: How could I check whether an event is cancelable or not?

Answer: Use `event.cancelable` to get true or false return. However, you have to `preventDefault()` to prevent the event.

Question: Is there anything you have to be careful when using `node.cloneNode()`?

Answer: While cloning, make sure you didnt duplicate ID.

Question: What are different nodeTypes?

Answer: ELEMENT_NODE (1), TEXT_NODE (3), COMMENT_NODE(8), DOCUMENT_NODE(9), DOCUMENT_TYPE_NODE(10), DOCUMENT_FRAGMENT_NODE(11), etc.

Question: What are the differences between node and element?

Answer: read here.

HTML Questions

1. doctype

Question: What is doctype? Why do u need it?

Answer: doctype is an instruction to the browser to inform about the version of html document and how browser should render it.

It ensures how element should be displayed on the page by most of the browser. And it also makes browser's life easier. otherwise, browser will guess and will go to quirks mode. Moreover, doctype is required to validate markup.

```
<!DOCTYPE html>
<meta charset="UTF-8">
```

extra: this the first tag of html file, don't need a closing tag and not case sensitive.

ref: don't forget doctype, Sparse vs Dense Array

2. data-*

Question: What is the use of data- attribute?

Answer: allow you to store extra information/ data in the DOM. u can write valid html with embedded private data. You can easily access data attribute by using javascript and hence a lot of libraries like knockout uses it.

```
<div id="myDiv" data-user="jsDude" data-list-size="5" data-maxage="180"></div>
```

ref: MDN: data-*, use data attribute

3. keygen

Question: How can u generate public key in html?

Answer: html has a keygen element that facilitate generation of key and submission via a form.

```
    <keygen name="name" challenge="challenge string" keytype="type" keyparams="pqg-params">
```

extra: keygen has to be used inside a form.

ref: MDN: keygen

4. bdo

Question: How can u change direction of html text?

Answer: use bdo (bidirectional override) element of html.

```
<!-- Switch text direction -->
<p><bdo dir="rtl">This text will go right to left.</bdo></p>
```

result:

This text will go right to left.

extra: rtl: right to left. and alternatively you can use, ltr: left to right.

5. mark

Question: How can u highlight text in html?

Answer: use mark element.

```
<p>Some part of this paragraph is <mark>highlighted</mark> by using mark element.</p>
```

result:Some part of this paragraph is highlighted by using mark element.

6. scoped

Question: Can u apply css rule to a part of html document?

Answer: yes. by using "scopped" in the style tag.

ref MDN: style

7. http request

Question: Does the following trigger http request at the time of page load?

```
<img src="mypic.jpg" style="visibility: hidden" alt="My photo">
```

Answer: yes

```
 <div style="display: none;">
    <img src="mypic.jpg" alt="My photo">
</div>
```

Answer: yes

ref: David Shariff: quiz

8. download order

Question: Does style1.css have to be downloaded and parsed before style2.css can be fetched?

```
<head>
    <link href="style1.css" rel="stylesheet">
    <link href="style2.css" rel="stylesheet">
</head>
```

Answer: No

Question: Does style2.css have to be downloaded and parsed before Paragraph 1 is rendered on the page?

```
<head>
    <link href="style1.css" rel="stylesheet">
</head>
<body>
    <p>Paragraph 1</p>
    <p>Paragraph 2</p>
```

```
    <link href="style2.css" rel="stylesheet">
</body>
```

Answer: yes

ref: David Shariff: quiz

9. self closing tag

Question: What are optional closing tag? and why would u use it?

Answer: p, li, td, tr, th, html, body, etc. you don't have to provide end tag. Whenever browser hits a new tag it automatically ends the previous tag. However, you have to be careful to escape it.

reason: you can save some byte and reduce bytes needs to be downloaded in a html file.

```
<p>Some text
<p>Some more text
<ul>
 <li>A list item
 <li>Another list item
</ul>
```

the above html will be parsed as the following blocks.

```
<p>Some text</p>
<p>Some more text</p>
<ul>
 <li>A list item</li>
 <li>Another list item</li>
</ul>
```

ref: W3.org: index of elements

Old School questions

10. span vs div

Question: What is the difference between span and div?

Answer: div is a block element and span is inline element.

Extra: It is illegal to put block element inside inline element. div can have a p tag and a p tag can have a span. However, span can't have a div or p tag inside.

ref: Stackoverflow: div vs span

11. div, section & article

Question: When should you use section, div or article?

Answer:

<section>, group of content inside is related to a single theme, and should appear as an entry in an outline of the page. It's a chunk of related content, like a subsection of a long article, a major part of the page (eg the news section on the homepage), or a page in a webapp's tabbed interface. A section normally has a heading (title) and maybe a footer too.

<article>, represents a complete, or self-contained, composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.

** , on the other hand, does not convey any meaning, aside from any found in its class, lang and title attributes.**

Good Summary:div, section & article

Extra: Authors are strongly encouraged to view the div element as an element of last resort, for when no other element is suitable. Use of more appropriate elements instead

of the div element leads to better accessibility for readers and easier maintainability for authors.

ref: (copied from) W3C: section, W3C: div, W3c: article

12. svg vs canvas

Question: What are the difference between svg and canvas?

Answer: Read this one. (I am tired of copy-pasting)

13. multiple languages

Question: How to serve a page content in multiple languages?

Answer: CMS could be used to deliver content in different language with same structure and style.

ref: john polacek

14. standard & quirks mode

Question: Difference between standard/ strict mode and quirks mode?

Answer: quirks mode in browser allows u to render page for as old browsers. This is for backward compatibility.

15. semantic

Question: What is semantic HTML?

Answer: Semantic HTML, or "semantically-correct HTML", is HTML where the tags used to structure content are selected and applied appropriately to the meaning of the content.

for example, (for bold), and (for italic) should never be used, because they’re to do with formatting, not with the meaning or structure of the content. Instead, use the replacements and (meaning emphasis), which

by default will turn text bold and italic (but don't have to do so in all browsers), while adding meaning to the structure of the content.

Question: Why you would like to use semantic tag?

Answer: Search Engine Optimization, accessibility, repurposing, light code.

Many visually impaired person rely on browser speech and semantic tag helps to interpret page content clearly.

Search engine needs to understand page content to rank and semantic tag helps.

ref: why important, Wiki: semantic HTML

Question: What does "semantically correct" mean?

Answer: read it in Stackoverflow

CSS Questions

1. float

Question: What does float do?

Answer: float pushes an element to the sides of a page with text wrapped around it. you can create entire page or a smaller area by using float. if size of a floated element changes, text around it will re-flow to accommodate the changes. You can have float left, right, none or inherit.

if you set, 'float: left;' for an image, it will move to the left until the margin, padding or border of another block-level element is reached. The normal flow will wrap around on the right side.

Hide example

```
<style>
   .floatContainer{
         width: 200px;
         height: 100px;
         border: 2px solid purple;
   }
   .box{
         float: left;
         width: 50px;
         height: 30px;
         border: 2px solid gray;
         margin: 5px;
   }
</style>
<div class="container">
    <div class="box"><span>1</span></div>
    <div class="box"><span>2</span></div>
    <div class="box"><span>3</span></div>
    <div class="box"><span>4</span></div>
    <div class="box"><span>5</span></div>
  </div>
```

1

2

3

4

5

If interviewer wants to ask one question about css, that would be most likely about float.

extra: read the positioning constraints in W3.org: floating elements.

ref: css-tricks: float, float 101

2. clear

Question: How can you clear sides of a floating element?

Answer: If you clear a slide of an element, floating elements will not be accepted on that side. With 'clear' set to 'left', an element will be moved below any floating element on the left side. clear is used to stop wrap of an element around a floating element.

Hide example

```
<style>
  .clearLeft{
     clear: left;
  }
</style>
<div class="container">
   <div class="box"><span>1</span></div>
   <div class="box clearLeft"><span>2</span></div>
   <div class="box"><span>3</span></div>
   <div class="box"><span>4</span></div>
   <div class="box"><span>5</span></div>
 </div>
```

as clear left is applied to the second box. no other elements would be on the left of that box and hence it is placed in a new row.

1

2

3

4

5

Question: How can you fix, "floated points don't add up to the height of a parent"?

Answer: You can use `clear: both` in an empty div `<div style="clear: both;"></div>`, you can use overflow hidden or scroll and you can float the parent as well.

What the heck? Sorry. if you didn't get the question or answer, please read "Techniques for clearing floats" in css-tricks: all about floats

ref: W3.org: clear

3. rapid fire

Question: Does css properties are case sensitive?

Answer: no.

Question: Why css selectors mixed up with cases don't apply the styles?

Answer: because, html ID and classes are case sensitive.

Question: Does margin-top or margin-bottom has effect on inline element?

Answer: no.

Question: Does padding-top or padding-bottom has effect on inline element?

Answer: no.

Question: Does padding-left or padding-right or margin-left or margin-right has effect on inline element?

Answer: yes.

Question: If you have a <p> element with font-size: 10rem, will the text be responsive when the user resizes / drags the browser window?

Answer: no.

Question: The pseudo class :checked will select inputs with type radio or checkbox, but not <option> elements.

Answer: False

Question: In a HTML document, the pseudo class :root always refers to the <html> element.

Answer: True

Question: The translate() function can move the position of an element on the z-axis.

Answer: False

4. units

Question: Which one would you prefer among px, em % or pt and why?

Answer: it depends on what you are trying to do.

`px` gives fine grained control and maintains alignment because 1 px or multiple of 1 px is guaranteed to look sharp. px is not cascade, this means if parent font-size is 20px and child 16px. child would be 16px.

`em` maintains relative size. you can have responsive fonts. em is the width of the letter 'm' in the selected typeface. However, this concept is tricky. 1em is equal to the current font-size of the element or the browser default. if u sent font-size to 16px then 1em = 16px. The common practice is to set default body font-size to 62.5% (equal to 10px). em is cascade

`%` sets font-size relative to the font size of the body. Hence, you have to set font-size of the body to a reasonable size. this is easy to use and does cascade. for example, if parent font-size is 20px and child font-size is 50%. child would be 10px.

`pt`(points) are traditionally used in print. 1pt = 1/72 inch and it is fixed-size unit.

ref: css-tricks: length, css-tricks: px, em, %, css font-size

5. position

Question: How absolute, relative, fixed and static position differ?

Answer:

absolute, place an element exactly where you want to place it. absolute position is actually set relative to the element's parent. if no parent available then relatively place to the page itself.

relative, is position an element relative to itself (from where the element would be placed, if u don't apply relative positioning). for example, if u set position relative to an element and set top: 10px, it will move 10px down from where it would be normally.

fixed, element is positioned relative to viewport or the browser window itself. viewport doesn't changed if u scroll and hence fixed element will stay right in the same position.

static, element will be positioned based on the normal flow of the document. usually, u will use position static to remove other position might be applied to an element.

ref: css positioning in ten steps, css position, MDN: position

6. display vs visibility

Question: What are the differences between visibility hidden and display none?

Answer: `display: none` removes the element from the normal layout flow and allow other elements to fill in. `visibility: hidden` tag is rendered, it takes space in the normal flow but doesn't show it.

if u want to say it smartly, `display: none` causes DOM reflow where is `visibility: hidden` doesn't. btw, what is re-flow? answer. sorry i wont tell you, google it.

ref: visibility vs Display

7. inline block

Question: What are the differences between inline, block and inline-block?

Answer:

`inline`, elements do not break the flow. think of span it fits in the line. Important points about inline elements, margin/ padding will push other elements horizontally not vertically. Moreover, inline elements ignores height and width.

`block`, breaks the flow and dont sits inline. they are usually container like div, section, ul and also text p, h1, etc.

`inline-block`, will be similar to inline and will go with the flow of the page. Only differences is this this will take height and width.

ref: display

8. box model

Question: What are the properties related to box model?

Answer: Technically, height, width, padding and border are part of box model and margin is related to it.

Extra: Everything in a web page is a box where you can control size, position, background, etc. Each box/ content area is optionally surrounded by padding, border and margin. When you set height and width of an element, you set content height and width.

ref: W3: box model, css box model, Whats wrong with box model

Hide example

ref: image taken from google search

9. overflow

Question: Does overflow: hidden create a new block formatting context?

Answer: yes

Extra: overflow property deals with the content if content size exceeds the allocated size for the content. You can make extra content visible, hidden, scroll or auto (viewport default behavior).

Hide example

`overflow: visible`This is some simple content to see how overflow works when you have too many things to say in a too small space. But don't be the person who just say rather be the person who does. does something for himself and others.

`overflow: hidden`This is some simple content to see how overflow works when you have too many things to say in a too small space. But don't be the person who just say rather be the person who does. does something for himself and others.

`overflow: scroll`This is some simple content to see how overflow works when you have too many things to say in a too small space. But don't be the person who just say rather be the person who does. does something for himself and others.

`overflow: auto`This is some simple content to see how overflow works when you have too many things to say in a too small space. But don't be the person who just say rather be the person who does. does something for himself and others.

ref: overflow (read the link and add something from it)

10. media queries

Question: How could you apply css rules specific to a media?

Answer: `@media (max-width: 700px){...}` means you want to apply rules to those media whose max-width is 700 px. this means every smaller device will have this rule.

`@media (max-width: 700px) and (orientation: landscape){...}` will apply rules for media smaller than 700px and in landscape orientation.

Question: What is the use of only?

Answer: to hide style sheets from older user agents.

Question: Does the screen keyword apply to the device's physical screen or the browser's viewport?

Answer: Browser's Viewport

ref: how to use media queries, css media queries, W3: media queries

11. pseudo class

Question: What are the some pseudo classed u have used?

Answer: pseudo class tells you specific state of an element. allow to style element dynamically. The most popular one is `:hover`. Besides i have used `:visited`, `:focus`,`:nth-child`, `nth-of-type`, `:link`, etc.

pseudo classes is better if you don't want to mess up with javaScript however, pseudo-classes is slow to process and apply rules.

ref: How to use pseudo classes, meet pseudo classes, list of pseudo classes

pseudo elements

pseudo elements helps you to add cosmetics contents. pseudo elements generates content where as pseudo class deals with state of the element. for example, you can style`:first-letter` of every paragraph. similarly, `:first-line` and fancy stuff with `:before`, `:after`

Hide example

```
<style>
   p:before{
      content: "Starting of Paragraph: ";
      font-weight: bolder;
      color: purple;
   }
</style>
<p>First Paragraph with some content.</p>
<p>Second Paragraph with something else.</p>
```

result:

First Paragraph with some content.

Second Paragraph with something else.

ref: intro to css pseudo element, :before :after, css content, W3: generate content

12. vertical Center

Question: How do you align a p center-center inside a div?

Answer: `text-align: center`will do the horizontal alignment but `vertical-align: middle` will not work here. there are couple of different ways to solve this problem and one of them are positioning. You make the parent as relative position and child as absolute positioning. And then define all position parameter as sero and width 50% and height 30% (sounds messy look at the example and read ref)

Hide example

```
<style>
   .divContainer{
      height: 100px;
      border: 2px solid gray;
      text-align: center;
      position: relative;
   }
   p{
```

```
      position: absolute;
      top: 0;
      bottom: 0;
      left: 0;
      right: 0;
      width: 50%;
      height: 30%;
      margin: auto;
    }
</style>
<div class = "divContainer">
  <p>content example</p>
</div>
```

result:

content example

ref: 6 methods for vertical center, Absolute horizontal and vertical centering

13. optimize selector

Question: How do you optimize css selectors?

Answer: This is very open and depend on what you are trying to achieve. If i order selectors in terms of render speed it would be like id, class, tag, siblings, child, descendant, universal, attribute, pseudo. Speed of ID and class is very close. However your code should be readable, maintainable and DRY along with highly performant.

The best practices in general are: avoid universal selectors, don't repeat yourself, remove redundant selectors, be as specific as possible, and keep learning.

ref: Efficient CSS selectors, efficiently rendering

14. @import

Question: How can you load css resources conditionally?

Answer: @import allows you to load/ import stylesheet by using a path (uri) representing the location of the file. You can define one or more media by comma separation for which you want to load the stylesheet. If browser dont support the media stylesheet will not be loaded.

ref: be careful while using @import (don't use @import)

15. sprite

Question: Why would you use sprites?

Answer: When you have multiple images/ icons, browser makes separate call to the server for each one of them. sprite is a technique to combine all/ some of them (usually similar one in terms of type of image. For example, you will put jpg in one sprite) in one image. To display the icon you set height, width and background position.

popular libraries like bootstrap use this technique. If you repeat the image. want to scale you have to be careful with sprite.

ref: css sprites, generate sprites

16. specificity

Question: What is specificity? How do u calculate specificity?

Answer: is a process of determining which css rule will be applied to an element. it actually determines which rules will take precedence.

inline style usually wins then ID then class value (or pseudo-class or attribute selector), universal selector (*) has no specificity.

ref: css specificity: things you need to know, specifishity, specificity calculator

17. shadow DOM

Question: What is shadow DOM?

Answer: encapsulate part of a DOM. hide subtree. you can have same ID in different shadow DOM. Polymers uses it. This way your DOM becomes reusable. if interviewer is not happy with your answer give him the links and tell him to spend a weekend on reading.

ref: W3: shadow-DOM, html5rock: shadow DOM

18. transition

Question: What do you know about transition?

Answer: transition allows to add an effect while changing from one style to another. You can set the which property you want to transition, duration, how you want to transit (linear, ease, ease-in, ease-out, cubic-bezier) and delay when transition will start. you can transition more than one property by comma separation

Hide example

```
.transition {
    transition: [transition-property] [transition-duration] [transition-timing-function]
[transition-delay];
}
.element {
  width: 50px;
  background-color: purple;
  transition: width  2s linear 0.5s, background-color 2s linear 0.5s;
}

.element:hover {
  width: 100px;
  background-color: red;
}
```

Hover over the purple box and wait. also move the out ur mouse.

Advanced: you can check transform: css 2D 3D transform, flip an image

ref: all you need to know about css transition, transition: tutorial, css transition, transition and 3D

19. filter

Question: What are the different css filter you can use?

Answer: css filter allows u to render DOM element, image, or video. u can choose from: grayscale, blur, opacity, brightness, contrast.

Show example

ref: Understanding css filter effect

20. pre processor

Question: What are the reasons to use preprocessor?

Answer: you write css in high level with some special syntax (declaring variable, nested syntax, mathematical operations, etc.) and that is compiled to css. Preprocessor helps you to speed up develop, maintain, ensure best practices and also confirms concatenation, compression, etc.

ref: css preprocessor, working with preprocessor

21. see & tell

Question: What would be the color of text "I am awesome" for he following rules?

html: for questions a-d.

```
<ul class="shopping-list" id="awesome">
    <li><span>Milk</span></li>
    <li class="favorite" id="must-buy"><span class="highlight">I am awesome</span></li>
</ul>
```

a.

```
<style>
  ul#awesome {
    color: red;
  }
  ul.shopping-list li.favorite span {
    color: blue;
  }
```

```
</style>
```

Answer: blue

b.

```
<style>
 ul#awesome #must-buy {
    color: red;
 }
 .favorite span {
    color: blue!important;
 }
</style>
```

Answer: blue

c.

```
<style>
  ul.shopping-list li .highlight {
    color: red;
  }
  ul.shopping-list li .highlight:nth-of-type(odd) {
    color: blue;
  }
</style>
```

Answer: blue

d.

```
<style>
  #awesome .favorite:not(#awesome) .highlight {
    color: red;
  }
  #awesome .highlight:nth-of-type(1):nth-last-of-type(1) {
    color: blue;
  }
</style>
```

Answer: red

Position related

Question: What will happen to the position of #myDude?

```
<style>
  #myDude {
    margin-bottom: -5px;
  }
</style>
<p id="myDude">Dude</p>
```

Answer: All elements succeeding #myDude will move 5px updward.

reason: .

```
<style>
  #myDude {
    margin-left: -5px;
  }
</style>
```

```
<p id="myDude">Dude</p>
```

Answer: #myDude will move 5px left.

download resources

Question: On page load, will mypic.jpg get downloaded by the browser?.

```
<style>
  #test2 {
    background-image: url('mypic.jpg');
    display: none;
  }
</style>
<div id="test1">
    <span id="test2"></span>
</div>
```

Answer: yes.

Question: On page load, will mypic.jpg get downloaded by the browser?

```
<style>
  #test1 {
    display: none;
  }
  #test2 {
    background-image: url('mypic.jpg');
    visibility: hidden;
  }
</style>
<div id="test1">
    <span id="test2"></span>
</div>
```

Answer: No.

read selector

Question: What will this selector do?

```
[role=navigation] > ul a:not([href^=mailto]) {
}
```

Answer: This selects anchor links that are not email links that are decedents of an unordered list that is the direct child of any element with a role attribute of 'navigation'. this answer copied from css tricks

Backbone JS

1) Explain what is backbone.js ?

Backbone.js is a JavaScript client-side (front end) framework, which helps to organize your code and makes it easier to develop single page applications. It allows you to structure JavaScript code in an MVC (Model, View , Controller) fashion

- **Model: It is a part of your code that populates and retrieves the data**
- **View: It is the HTML representation of this model**
- **Controller: It enables you to save your javascript application via a hashbang URI**

2) What are the main components of Backbone.js ?

The main component of Backbone.js are

- **Model**
- **View**
- **Collection**
- **Router**
- **Event class object**

3) Explain what is Backbone.js collections ?

An ordered set of models are represented by Backbone.js collections. Any event in model will trigger an event in collection directly. For example, you can bind "change" event to be notified in a case when any model in the collection has been modified.

4) Explain what is Backbone.js router is used for ?

Whenever an application want to change their URL fragment in order to provide bookmarkable and shareable URLs for an Ajax heavy application, backbone.js router is used.

5) What is Backbone events ?

Backbone events is a module that can be mixed in to any object, giving the object the ability to bind and trigger custom named events. Events are not declared before they are bound to any object . Events reflects the state of the model.

6) What are the keypoints of Backbone ?

- **It has hard dependency with underscore.js to make it more functional and supporting a range of useful collection based operations**
- **With jQuery it has a soft dependency**
- **When the model changes it can update the HTML of your application automatically**
- **It uses client-side rendering framework or Javascript templating to render html which avoid you to embed HTML code inside JavaScript code**
- **For UI updates and DOM manipulations if offers a significantly clean and elegant way**

7) Why you have to use Backbone ? Advantages ?

- By using JavaScript with the minimal set of data-structuring (models & collections) and user interface (views & URLs) it enables you to develop a web application
- Backbone is best useful to develop MVC like web applications, single page web applications or complex JavaScript web applications in an organized and structured manner without JavaScript code mixing with HTML
- Provides key value binding and custom events
- API with tons of functions
- Robust event handling
- API connetion over a RESTful JSON interface

8) What are the three js files that you require to setup a working environment for backbone?

you are required following three js files to setup a working environment for backbone

- jQuery
- Backbone
- Underscore

In your application put these files within js folder and use it in your index.html page

9) Explain when you require Backbone.js ?

Backbone.js is required in following condition

- When developing a web application that requires a lot of JavaScript
- It is required when you want to give structure to your code, if your application needs to be scalable
- Backbone is useful when a web application has to work with jQuery to traverse the DOM or give animations

10) Explain what is view in Backbone.js?

Backbone view is a Javascript object that manages a specific DOM element and descendants.

- Views are not HTML
- It is a description of a model
- The HTML code comes from templates
- Works with any template system

11) Explain what is Backbone.js Models?

Backbone.js models are object and core of backbone.js. It contains an array of attributes and listens for events. To represent your data, Backbone provides a **model object**. For example, you have a **to do list**, you would have a model representing each item on that list.

12) Explain how you can use backbone.js for multiple page web app ?

For multiple page web app in backbone.js there are lots of consideration but here are two which can be useful

- **Serving the page :** In this, where you want to have your web server route everything to the server route everything to serve the same static page. That means that everything in http://guru99.com/* will serve /var/www/guru99.com/index.html. once the static page is loaded, the JS on that page will decide what to do given the url
- **Push State :** You can still use backbone routing to do your routing, but don't use hashbangs. This will allow you to navigate to URLs without actually needing a page refresh.

13) Explain what is Modelbinder in Backbone.js ?

To make synchronization process of views and models together, ModelBinder class is used.

14) What is the most powerful capabilities of the ModelBinder ?

The most powerful capabilities of ModelBinder class is that it enables you to define scope when you create your bindings using jQuery.

- If your views are simple, you can rely on default scoping rules that are based off of the html “name” attribute.
- You can define scoping with jQuery selectors if your views are complex.

15) Explain what is Converter in Backbone.js ?

A function is called when model’s attribute is copied to an html element or when an html element value is copied into a model’s attribute, this function is referred as Converter in Backbone.js

16) What is model.attributes ?

The attributes property is the internal hash containing the model’s state, usually a form of the JSON object representing the model data on the server. It is often a straightforward serialization of a row from the database

17) What is the function of toJSON ?

It returns a shallow copy of the model’s attribute for JSON stringification. This function is used for persistence, serialization and for augmentation before being sent to the server. This does not return a JSON string

18) Explain when you can use Unbinding function in Backbone.js ?

When you want to remove the validation binding on the model or all models , removing all events hooked up on the collection, you can use Unbinding function.

For example : **Backbone.Validation.Unbind(view)** [This will remove the validation binding]

19) What are the configuration options available ?

The configuration options available are

- InitialCopyDirection
- modelSetOptions
- change Triggers
- boundAttribute
- suppressThrows
- converter

20) Mention what are the typical problems you might face with the Backbone view code ?

- Application models don't change very often
- Application pages are frequently refreshed from scratch from the server
- Between different view models are not shared

21) What is the function of escape ?

It gets the current value of an attribute from the model but returns the HTML-escaped version of a model's attribute. It is helpful in preventing XSS attacks, if you are interpolating data from the model into HTML

22) Explain what is the function of parse ?

Whenever a model's data is returned by the server, in fetch and save , this data is called **parse.** It is called by Backbone whenever a collection's models are returned by server, in fetch.

23) What is Backbone.sync is used for ?

When Backbone wants to save or read a model to the server it calls out a function called as Backbone.sync.

24) In Backbone View, what is the use of setElement ?

setElement function is used when Backbone view has to be applied to a different DOM element.

25) Explain what is model.cid ?

Model.cid works as a unique identifier. It is a special property of models, the cid or client id is automatically assigned to all models when they are first created. This property is useful when the model is not saved to the server, but needs to be visible in the UI. It takes the from c1,c2….