**Q1. How do you copy by value a composite data type?**

**Using for loop.**

numbers = [1, 2, 3];

numbersCopy = [];

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

numbersCopy[i] = numbers[i];

}

numbersCopy.push(4);

console.log(numbers, numbersCopy);

// [1, 2, 3] and [1, 2, 3, 4]

**Q2. why there is a difference in behavior for copying contents in primitive and non primitive type?**

Primitive values are always passed by its value (copied). Non-primitive values however are passed by reference.

**Primitives** are known as being immutable data types because there is no way to change a primitive value once it gets created.

Non primitive values can also be referred to as reference types because they are being compared by reference instead of value. Two objects are only strictly equal if they refer to the same underlying object.

Hence, the behavior of copying contents varies.

**Q3. Use typeof in all the datatypes and check the result**

**\* typeof(1) - number**

**\* typeof(1.1) - number**

**\* typeof("1.1") - string**

**\* typeof(true) - Boolean**

**\* typeof(null) - object**

**\* typeof(undefined) - undefined**

**\* typeof([]) - object**

**\* typeof({}) – object**

**Q4.** **Write a blog about objects and its internal representation in JavaScript**

Conceptually, Objects are same in all programming languages i.e they represent real-world things that we want to represent inside our programs with characteristics/properties and methods.

For Eg. If your object is a student, it will have properties like name, age, address, id, etc and methods like updateAdress

Let’s see an example:-

const firstObj = {  
1: “Deepak”,  
“age”: 28  
}

first Obj is an object with 2 properties 1 and age and value as  Deepak and 28.

JavaScript objects are somewhat different in the way they are created. We can create objects in many ways in JavaScript,

1. Object literal (Direct way) — Object literals are a comma-separated list of key-value pairs wrapped in curly braces. Object literal property values can be of any data type, including array literals, functions, nested object literals or primitive data type.

var student = {  
id: 1,  
name: “Deepak”,  
age: “27”,  
updateAddress: () => {  
// logic to update address  
},  
grade: [‘A’, ‘A+’, ‘A’]  
}

The student object keys in the above can be accessed via Dot notation i.e. student.id, student.name or via a square bracket notation i.e. student[‘id’], student[‘name’], etc.

1. Object.create() — the method creates a new object with the specified prototype and properties of the old object.

var newStudent = Object.create(student);  
We can now add new properties and data newStudent object using the method we are learning here.

The newStudent will have access to the parent student object keys and value as it's been added to newStudent prototype chain and this is one way we do inheritance in JavaScript. That is, newStudent will store a link to student object. This parent object is also looked when a property is read.

The parent can have a parent and so on. This is repeated until we reach an object that does not have any parent i.e the parent is null.

1. Object Instance— The use of Object constructor in conjunction with the “new” keyword allows us to initialize new objects.

const newObj = new Object();  
newObj.name = ‘Deepak’;  
newObj.location = ‘Delhi, India’;

the above method using new Object() is not well suited to programs that require the creation of multiple objects of the same kind, as it would involve repeatedly writing the above lines of code for each such object.

To deal with this problem, we can use the next method.

1. Object Constructors — Constructors can be useful when we need a way to create an object “type” that can be used multiple times without having to redefine the object every time and this could be achieved using the Object Constructor function.

function Vehicle(name, model) {  
this.name = name;  
this.model = model;  
}  
  
let car1 = new Vehicle(‘Fiesta’, ‘2019’);  
let car2 = new Vehicle(‘DC avanti’, ‘2018’);

We created two objects with the same property but with different values.

1. Object.assign() —this is another method to create a new object from other objects.

It copies the values of all enumerable own properties from one or more source objects to a target object. It will return the target object.

const target = { a: 1, b: 2 };  
const source = { b: 4, c: 5 };

const returnedTarget = Object.assign(target, source);

console.log(target);  
// expected output: Object { a: 1, b: 4, c: 5 }

console.log(returnedTarget);  
// expected output: Object { a: 1, b: 4, c: 5 }

var obj = { a: 1 };  
var copy = Object.assign({}, obj);  
console.log(copy); // { a: 1 }

There are a lot of use cases for Object.assign() like Object cloning, Merging objects, etc.

1. Object.Entries() — method transforms a list of key-value pairs into an object.

const entries = new Map([  
[‘foo’, ‘bar’],  
[‘baz’, 42]  
]);

const obj = Object.fromEntries(entries);

console.log(obj);  
// expected output: Object { foo: “bar”, baz: 42 }

**Q5. execute and see at least 15 cli commands. like mkdir, ls etc.**

**ASSOC Displays or modifies file extension associations.**

**ATTRIB Displays or changes file attributes.**

**BREAK Sets or clears extended CTRL+C checking.**

**BCDEDIT Sets properties in boot database to control boot loading.**

**CACLS Displays or modifies access control lists (ACLs) of files.**

**CALL Calls one batch program from another.**

**CD Displays the name of or changes the current directory.**

**CHCP Displays or sets the active code page number.**

**CHDIR Displays the name of or changes the current directory.**

**CHKDSK Checks a disk and displays a status report.**

**CHKNTFS Displays or modifies the checking of disk at boot time.**

**CLS Clears the screen.**

**CMD Starts a new instance of the Windows command interpreter.**

**COLOR Sets the default console foreground and background colors.**

**COMP Compares the contents of two files or sets of files.**

**COMPACT Displays or alters the compression of files on NTFS partitions.**

**CONVERT Converts FAT volumes to NTFS. You cannot convert the**

**current drive.**

**COPY Copies one or more files to another location.**

**DATE Displays or sets the date.**

**DEL Deletes one or more files.**

**DIR Displays a list of files and subdirectories in a directory.**

**DISKPART Displays or configures Disk Partition properties.**

**DOSKEY Edits command lines, recalls Windows commands, and**

**creates macros.**

**DRIVERQUERY Displays current device driver status and properties.**

**ECHO Displays messages, or turns command echoing on or off.**

**Q6. What is the difference between window, screen, and document in JavaScript?**

**WINDOW:**

Window is a top level hierarchy in JavaScript and window consist of number of objects in it as properties and DOCUMENT and SCREEN are one of the objects in Window among all.

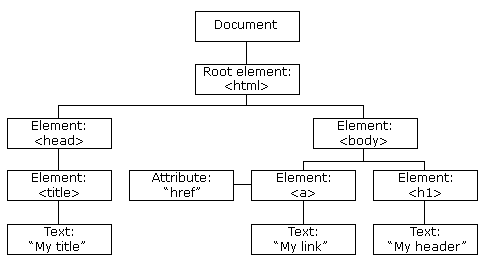
Window represent the complete things about browsers windowThere are lot of properties in window object. We can determine the size of the browser window by using below property.

window.innerHeight ->gives height of browser in pixels.

**DOCUMENT:**

DOCUMENT is the actual content of the page i.e. the html page you are loading is converted to the DOM object [Document object Model].

When a web page is loaded, the browser creates a Document Object Model of the page.



be created as TREE MODEL as above DOM TREE MODEL

document.getElementById(“header”);// gives the value of header

As document is property of window the above code is same as

window.document.getElementById(“header”);// gives the value of header.

**SCREEN:**

SCREEN is a property of WINDOW where we can see the content about the users screen. It can be used to display screen width, height etc.

Properties

screen.width  
screen.height  
screen.availWidth  
screen.availHeight

The window is the first thing that gets loaded into the browser.

The document object is your html, aspx, php, or other document that will be loaded into the browser.

The document gets loaded inside the window object and has properties available to it like title, URL, cookie, etc. If you want to access a property for the window it is window. Property, if it is document it is window.document.property which is also available in short as document.property.