**Assignment 2**

**Date:06/11/202**

**Q-1)** **How do you copy by value a composite data type?**

**1.Using Spread(…)operator**

var a=[1,2,3]

var b=[…a]

console.log(a,b)

b[2]=6

console.log(a,b)

**2.Using Object.assign() function**

var a=[1,2,3]

var b=Object.assign([],a)

console.log(a,b)

b[2]=6

console.log(a,b)

3.**Using JSON.parse() and JSON.stringify() method**

var a=[1,2,3]

var b=JSON.parse(JSON.stringify(a))

console.log(a,b)

b[2]=6

console.log(a,b)

**Q-2)** **Why there is a difference in behavior for copying contents in primitive and non primitive type?**

In JavaScript, a variable may store two types of values: primitive and non primitive types. JavaScript provides primitive types as **undefined**, **null**, **boolean**, **number**, **string**, and **symbol**, and non primitive(reference) types as **objects**, **arrays**, and **functions**.

The size of a primitive value is fixed, therefore, JavaScript stores the primitive value on the stack.

Whereas the size of a reference value is dynamic so JavaScript stores the reference value on the heap. Since these types do not have a fixed size, their values cannot be stored directly in the fixed bytes of memory associated with each variable. Instead, the variable stores a reference to the value which is efficient in storage. This reference is some form of pointer or memory address. It is not the data value itself, but it tells the variable where to look to find the value.

When you assign a value to a variable, the JavaScript engine will determine whether the value is a primitive or reference value. If the value is a primitive value, when you access the variable, you manipulate the **actual value** stored in that variable. In other words, the variable that stores a primitive value is **accessed by value**. Unlike a primitive value, when you manipulate an object, you work on the **reference** of that object, rather than the actual object. It means a variable that stores an object is **accessed by reference**.

**Q-3) Use typeof in all the datatypes and check the result**

The typeof operator is used to get the data type (returns a string) of its operand. The operand can be either a literal or a data structure such as a variable, a function, or an object. The operator returns the data type.

**typeof(1)**

number

**typeof(1.1)**

number

**typeof("1.1")**

string

**typeof(true)**

boolean

**typeof(null)**

object

**typeof(undefined)**

undefined

**typeof([])**

object

**typeof({})**

object

**Q-4)** **Write a blog about objects and its internal representation in Javascript**

Objects can contain many values.

The values are written as name : value pairs (name and value separated by a colon).

**Eg.**

var emp={ id:101 name:”Gayatri”}

A JavaScript object is a collection of **named values**

The named values, in JavaScript objects, are called **properties**.

|  |  |
| --- | --- |
| Property | Value |
| id | 101 |
| name | Gayatri |

Objects are mutable: They are addressed by reference, not by value.

Objects also have a *prototype*. A prototype is another object that is used as a fallback source of properties. When an object gets a request for a property that it does not have, its prototype will be searched for the property, then the prototype’s prototype, and so on.

**Q-5)** **execute and see at least 15 cli commands. like mkdir, ls etc**

**1.ls**

ls is a utility for listing the files in a directory.

Syntax: ls

**2.mkdir**

mkdir is a utility for creating a directory.

Syntax: mkdir directory\_name

3.**cd**

**cd** changes the **c**urrent **d**irectory of the shell. This current directory will be used by other programs launched from the shell.

Examples

Change to 'foobar' directory:

$ **cd foobar**

Change to your home directory, cd command used without an option will drop you back into your home directory.

$ **cd**

~ (tilde) stores the path to your home directory, this command has same effect as the previous one.

$ **cd ~**

Change to parent directory:

$ **cd ..**

Change to the previous directory:

4.**pwd**

**pwd** (for Print Working Directory) shows the current directory that you are in.

*Examples*

$ **pwd**

/home/username

5.**cp**

**cp** copies a file

Most used options are:

**-r**

copies directories (recursively)

**-p**

preserves permissions, ownership, and timestamps

**-i**

prompt before overwrite

**-v**

verbose, show filenames as they are being copied

*Examples*

Makes a copy of file 'debian' and call it 'Debian' (assuming 'Debian' is not already a directory)

$ **cp -i debian Debian**

Makes a copy of file 'debian' and put it at /tmp/debian

$ **cp -i debian /tmp/debian**

**6.rm**

**rm** deletes a file from the filesystem, like the "del" command in DOS.

$ **rm filename**

**7.rmdir**

**rmdir** is a utility for deleting empty directories.

*Examples*

$ **rmdir directoryname**

If the directory is not empty, the correct way to remove the directory and all its contents recursively is to use

$ **rm -r directoryname**

**8.df**

**df** reports the amount of free disk space available on each partition.

$ **df**

Filesystem 1K-blocks Used Available Use% Mounted on

/dev/md0 5763508 207380 5263352 4% /

/dev/md1 78819376 13722288 61093296 19% /home

/dev/md4 23070564 4309572 17589056 20% /usr

/dev/md2 5763508 1757404 3713328 33% /var

/dev/md3 2877756 334740 2396832 13% /tmp

### 9.du

Shows disk usage.

**10.chroot**

**chroot** changes the root filesystem.

*Examples*

To change the root filesystem so /mnt/usbdrive/ becomes / and files outside of it cannot be seen:

# **chroot /mnt/usbdrive/**

Q-6) **What is the difference between window, screen, and document in Javascript**

**Window**

The JavaScript **window object** sits at the top of the JavaScript Object hierarchy and represents the browser window. The window object is supported by all browsers. All global **JavaScript objects** , functions, and variables automatically become members of the window object.

**Document**

The **Document interface** represents any web page loaded in the browser and serves as an entry point into the web page's content, which is the DOM tree. When an HTML document is loaded into a **web browser** , it becomes a document object. It is the root node of the HTML document. The document actually gets loaded inside the window object and has properties available to it like title, URL, cookie, etc. HTML documents, served with the **"text/html"** content type, also implement the HTMLDocument interface, whereas XML and SVG documents implement the XMLDocument interface.

**Screen**

Screen is a small information object about physical **screen dimensions** . It can be used to display screen width, height, colorDepth, pixelDepth etc. It is not mandatory to write **window prefix** with screen object. It can be written without window prefix.