## 1.How do you copy by value a composite data

Method 1: Using for loop

```
let user = {name: 'Sony', age: 30};
let user_copy = {};
for(let key in user)
   user_copy[key] = user[key];
user_copy.name = 'Nawaz';
console.log(user);
console.log(user_copy);
```

**Output:**

```
{ name: 'Sony', age: 30 }
{ name: 'Nawaz', age: 30 }
```

Method 2: Using the method Object.assign()

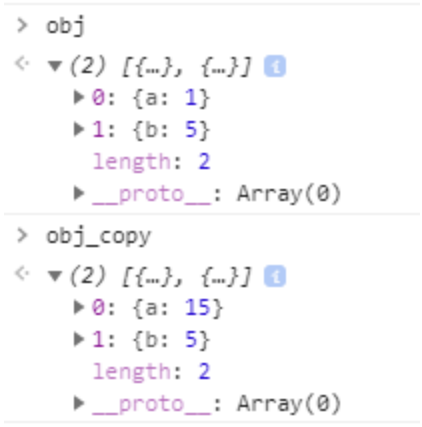
```
let user = {name: "John", age: 30};

let user_copy = Object.assign({}, user);

user_copy.name="Brad Lee"

console.log(user);
console.log(user_copy);
```

**Output:**

```
{ name: 'Sony', age: 30 }
{ name: 'Brad Lee', age: 30 }
```

Method 3: Using the method _.cloneDeep() for nested object

```
var obj=[{a:1},{b:5}];
var obj_copy=_.cloneDeep(obj);
obj_copy[0].a=15
```




## 2.Why there is a difference in behavior for copying contents in primitive and non-primitive types?

Primitive types are predefined in any programming language whereas Non-primitive types are created by the programmer and are not defined by the programming language itself. Because of this primitive types follows the pass-by-value technique whereas non-primitive types follow pass-by-reference.

In pass-by-value, the original value is copied from the actual parameter to the formal parameter, and any changes made in the formal parameter does not affect the actual parameter.

Whereas in pass-by-value, the references the object is not copied, it is shared (i.e., actual parameter and formal parameter are aliases) or in simple terms address of the actual parameter is shared with the formal parameter, and any changes made to the contents in the address hold by the formal parameter affect the actual parameter since they both points to the same address location.

## 3.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
```

## 4.Write a blog about objects and its internal representation in Javascript

Objects, in JavaScript, are the most important data-type and forms the building blocks for modern JavaScript. These objects are quite different from JavaScript's primitive data-types(Number, String, Boolean, null, undefined, and symbol) in the sense that while these primitive data-types all store a single value each (depending on their types).

Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types.

An object is a reference data type. Variables that are assigned a reference value are given a reference or a pointer to that value. That reference or pointer points to the location in memory where the object is stored. The variables don't actually store the value.

Loosely speaking, objects in JavaScript may be defined as an unordered collection of related data, of primitive or reference types, in the form of "key: value" pairs. These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.

An object can be created with figure brackets {...} with an optional list of properties. A property is a "key: value" pair, where a key is a string (also called a "property name"), and value can be anything.

let school = {

name: "NCET",

location: "Bangalore",

established: 1971

}

---

let school = {

name:  "NCET",

```
location : "Bangalore",
established : 1971,
displayInfo : function(){
console.log(`${this.name} was established in ${this.established} at ${this.location}`);
}
}
```

---

```
const school = new Object();
school.name = "NCET";
school.location ="Bangalore";
school.established = 1971;
school.displayInfo = () => {
console.log(`${this.name} was established in ${this.established} at ${this.location}`);
}
```

**5.Execute and see at least 15 CLI commands(windows 10).**

| Cmd command | Description |
|---|---|
| cls | Clear screen |
| cmd | Start command prompt |
| date | show/set date |
| dir | List directory content |
| echo | Text output |
| exit | Exit the command prompt or batch file |

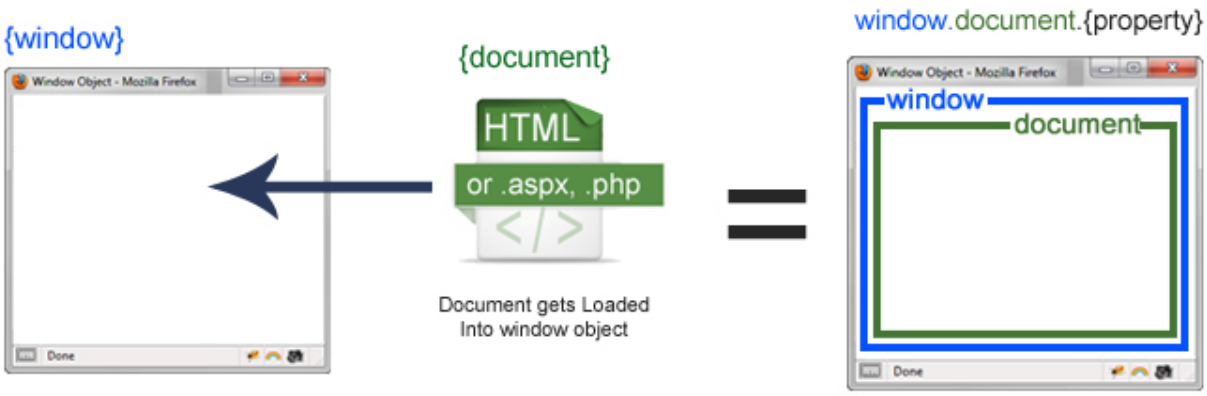
| | |
|---|---|
| find | Find files |
| hostname | Display hostname |
| pause | Pause the execution of the batch file and show a message |
| runas | Start a program as another user |
| shutdown | Shutdown the computer |
| start | Start a new window to execute a program or command |
| ver | Display OS version |
| ipconfig | Display IP network settings |
| mkdir | Create a new directory |
| del | Delete one or more files |
| rmdir | Delete directory |

## 6.What is the difference between window, screen, and document in Javascript?

window is the main JavaScript object root, aka the global object in a browser, also can be treated as the root of the document object model. You can access it as a window.

window.screen or just screen is a small information object about physical screen dimensions.

window.document or just document is the main object of the potentially visible (or better yet: rendered) document object model/DOM.
*Since* window *is the global object you can reference any properties of it with just the property name - so you do not have to write down* window*. - it will be figured out by the runtime.*


{window}
Window Object - Mozilla Firefox
Done
{document}
HTML
or .aspx, .php
Document gets Loaded
Into window object
window.document.{property}
Window Object - Mozilla Firefox
window
document
Done