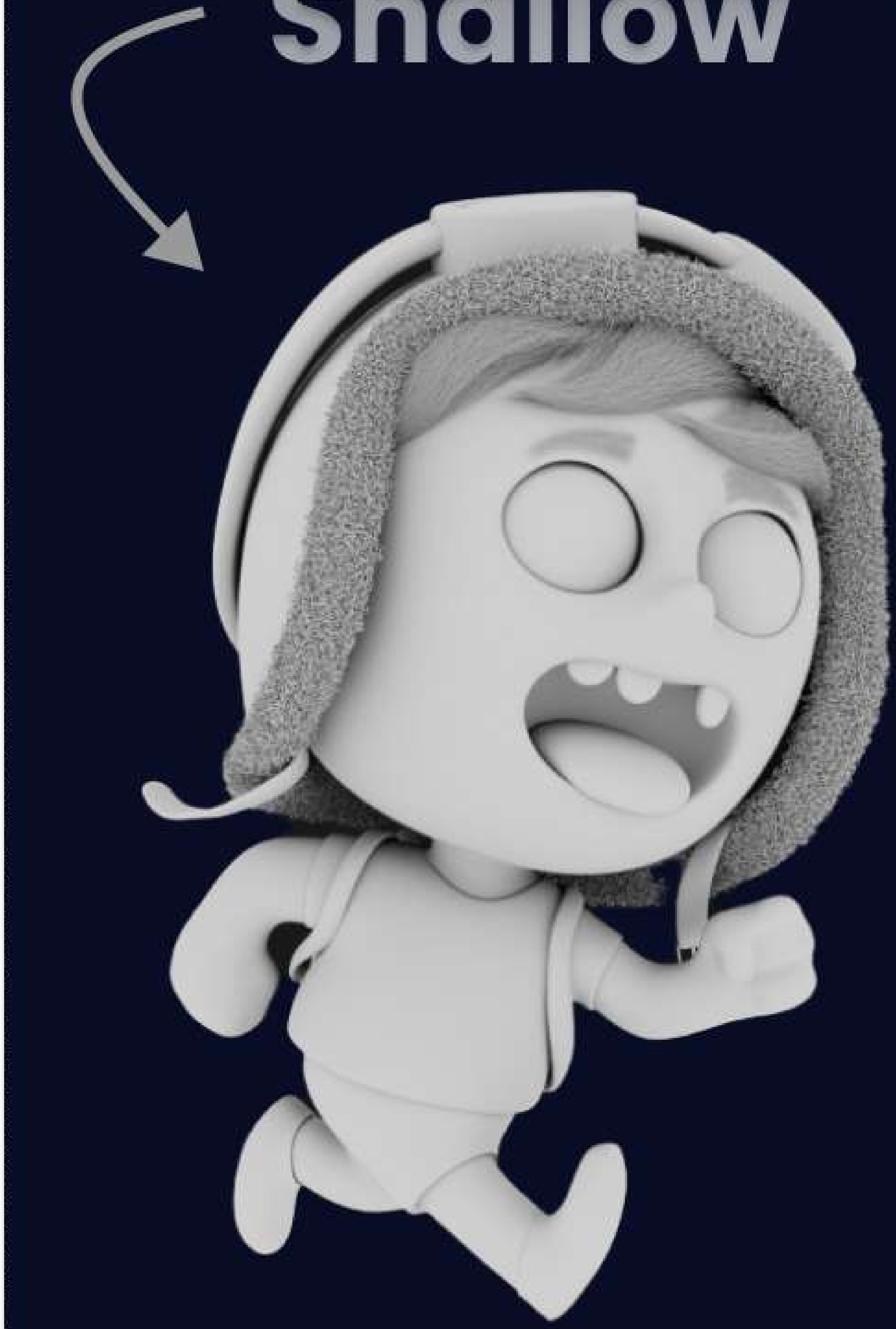
Javascript

Shallow Vs Deep copy



@startwithmani

@M.serisha Kothapalli

swipe

RECAP

Hey there! I'm back again ...

In my last post we went through object methods.
As you can remember I have skipped **Object.assign( )** method intentionally....So before we learn that in my upcoming posts, we need to have better understanding on this topic....

lets get to know about these **jargons**

what ***Shallow copy*** and ***Deep copy*** are??

@startwithmani
@M.serisha Kothapalli

← swipe

Deep copy:

A deep copy means that all of the values of a variable are copied into a new variable and this new variable is completely disconnected from the original variable.

Shallow copy:

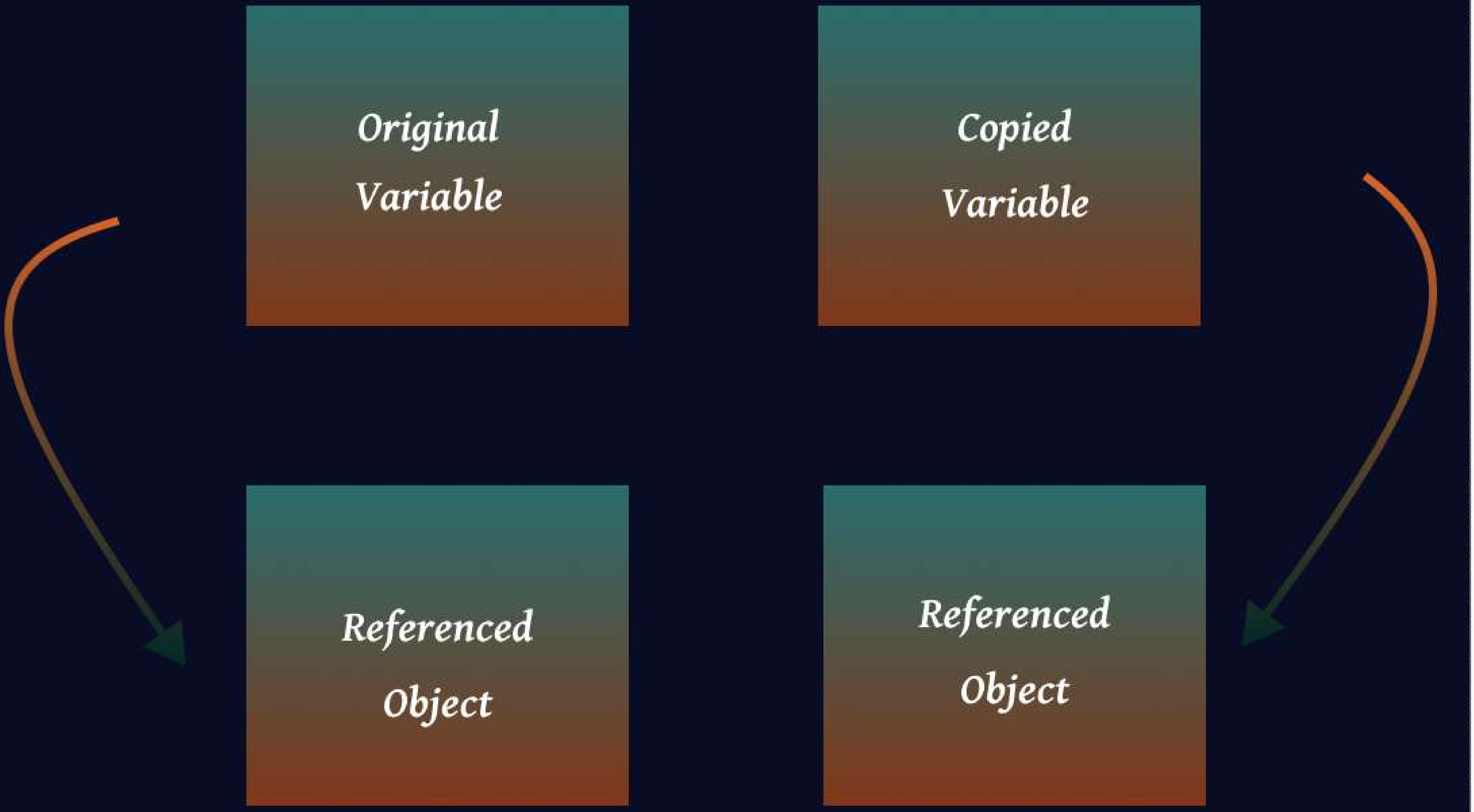
A shallow copy means that certain (sub-)values are still connected to the original variable.So the change made to the new variable will have the affect on original variable as well....

@startwithmani
@M.serisha Kothapalli

swipe

Deep copy:

As they refer to different references, changes made in one doesn't have affect on other.



Original Variable

Copied Variable

Referenced Object

Referenced Object

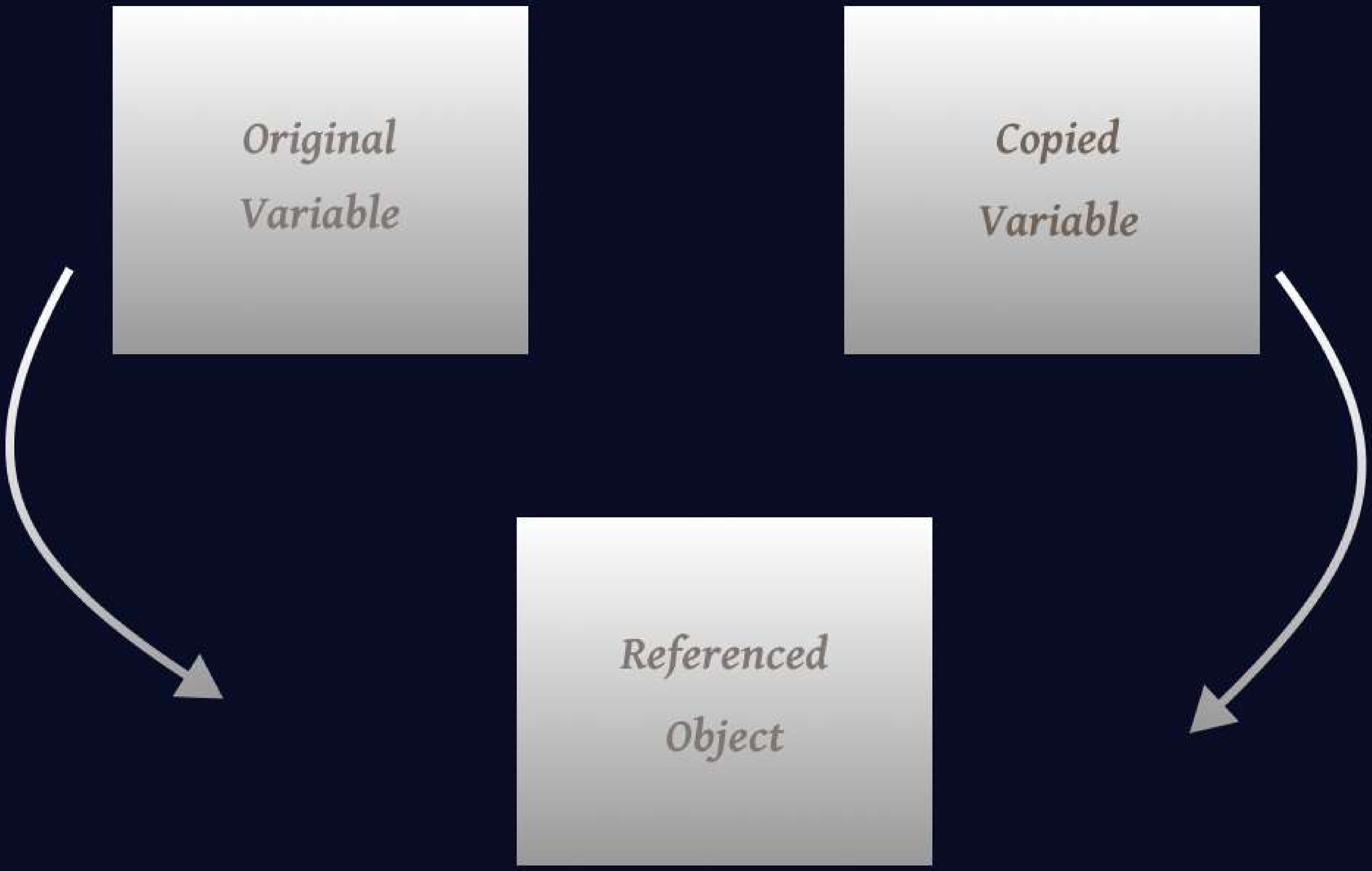
@startwithmani

@M.serisha Kothapalli

swipe

Shallow Copy:

As they refer to same object, changes made in one can affect other



Original Variable

Copied Variable

Referenced Object

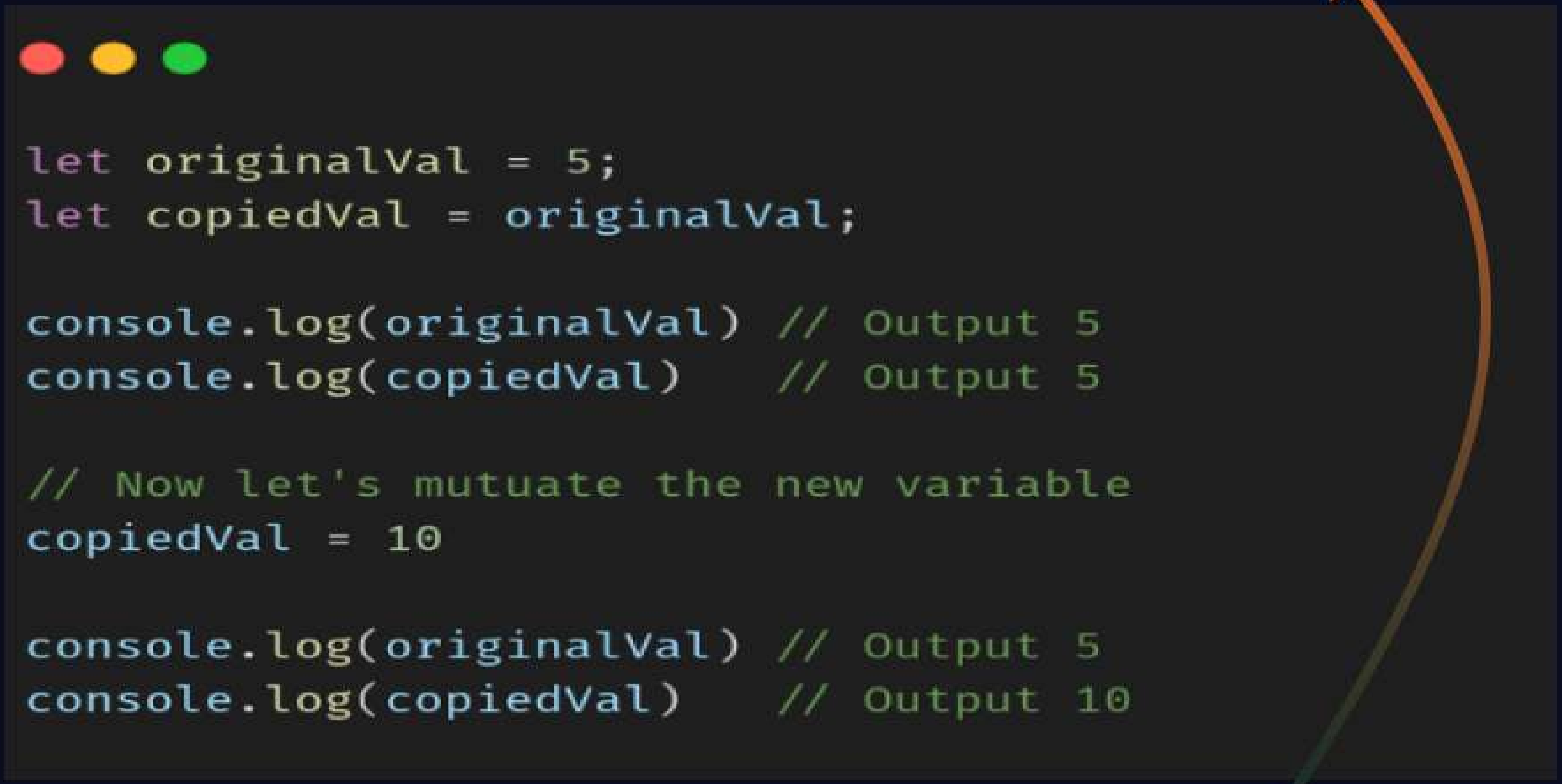
@startwithmani

@M.serisha Kothapalli

swipe

Deep copy:

Number, Boolean, String, Null,
Undefined, Symbol



```
let originalVal = 5;
let copiedVal = originalVal;

console.log(originalVal) // Output 5
console.log(copiedVal)   // Output 5

// Now let's mutuate the new variable
copiedVal = 10

console.log(originalVal) // Output 5
console.log(copiedVal)   // Output 10
```

Javascript always does a ***deep copy*** for ***primitive data types*** variable by default. So you don't have to worry about the shallow or deep copy for primitive datatypes. huhhh! Thank god!!!

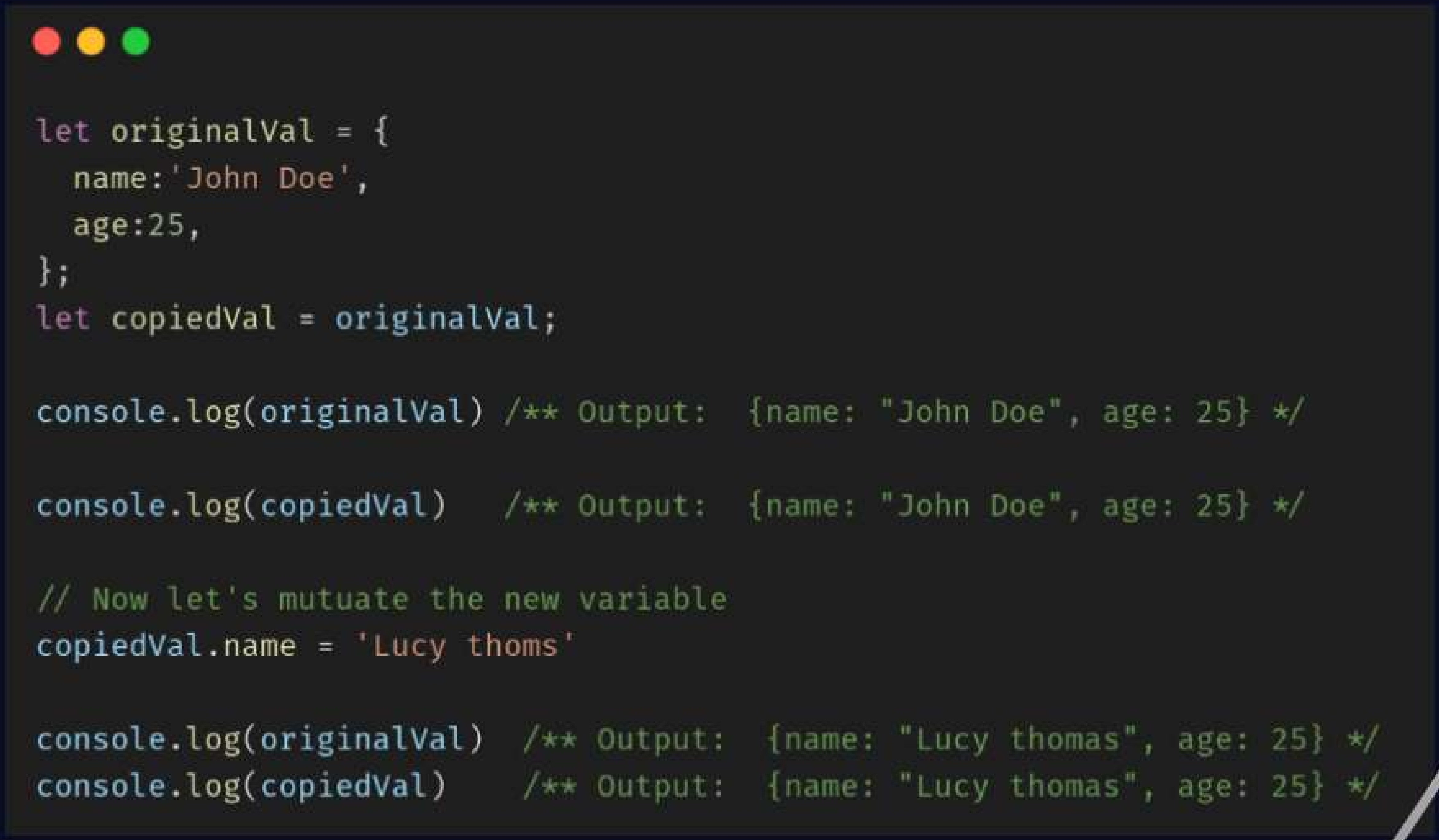
@startwithmani

@M.serisha Kothapalli

swipe

Shallow Copy:

Object, Array, Set, Functions, etc....



```
let originalVal = {
  name:'John Doe',
  age:25,
};
let copiedVal = originalVal;

console.log(originalVal) /** Output:  {name: "John Doe", age: 25} */

console.log(copiedVal)   /** Output:  {name: "John Doe", age: 25} */

// Now let's mutuate the new variable
copiedVal.name = 'Lucy thoms'

console.log(originalVal)  /** Output:  {name: "Lucy thomas", age: 25} */
console.log(copiedVal)    /** Output:  {name: "Lucy thomas", age: 25} */
```

Javascript does a ***shallow copy*** for the object data type (Non-primitive) automatically. So when I change the name(key) of copiedVal variable it also affects the originalVal and changed the original variable too.....

Thats much of it,

what if your requirement is not to change the original source which means want to do a deep copy.......🤔

Don’t worry Javascript has some methods available to achieve this.

We will be covering all of them in detail in upcoming posts....

So Stay Tuned !....😁

@startwithmani

@M.serisha Kothapalli

← swipe

Did you find it helpful??

♡ Like this post!

Share with your friends

Save it for later

Follow for more!

@startwithmani

in @M.serisha Kothapalli