

Shallow copy vs Deep Copy

By CODEMIND Technology

Contact us 966 5044 698
966 5044 598

Memory Allocation

Shallow copy Vs Deep Copy

- Constant object
- Object Freezing
- Object.assign() : Cloning and Merging an object
- Stack
- Stack operations
- Memory allocation
- Shallow copy vs Deep Copy

Defining Object with 'const keyword'

When we define object as 'const' then after defining object that reference variable will not be changed to point any other object

```
const student = {  
  name: "Mohit",  
  rollNo: 1234  
}  
student = { // Not allowed: TypeError: Assignment to constant variable.  
  city: "Pune"  
}
```

Object Freezing

Freezing an object does not allow new properties to be added to an object and prevents from removing or altering the existing properties.

Object.freeze(): We can use this method for freezing objects and arrays.

Freezing an object:

Note: Freezing an object and defining object with const keyword are two different things.

```
let student = {  
  name: "Mohit",  
  rollNo: 1234  
}  
  
Object.freeze(student);  
student.name = "Mohit Sharma"; // Not allowed to update  
student.city = "Pune"; // Not allowed to add new property  
console.log(student);
```

Freezing an Array

Freezing an array does not allow new element to be added to an array and prevents from removing or altering the existing elements.

Object.freeze(): We can use this method for freezing objects array.

Freezing an Array:

```
let array = [2, 3, 4, 5];  
Object.freeze(array);  
array.push(6); // TypeError: Cannot add property 4, object is not extensible  
array.shift(); // TypeError: Cannot assign to read only property '0' of object '[object Array]'
```

Note: When we declare array as const then that array reference variable can not point to any other array

Object.assign(): Syntax: Object.assign(target, ...sources);

This method is used for

1. To Clone an object

```
const emp = {  
  emp_name: "Anil",  
  company: "TCS"  
}  
  
// 1. Cloning an object  
const cloned_emp = Object.assign({}, emp);  
console.log(cloned_emp);
```

2. To merge an objects

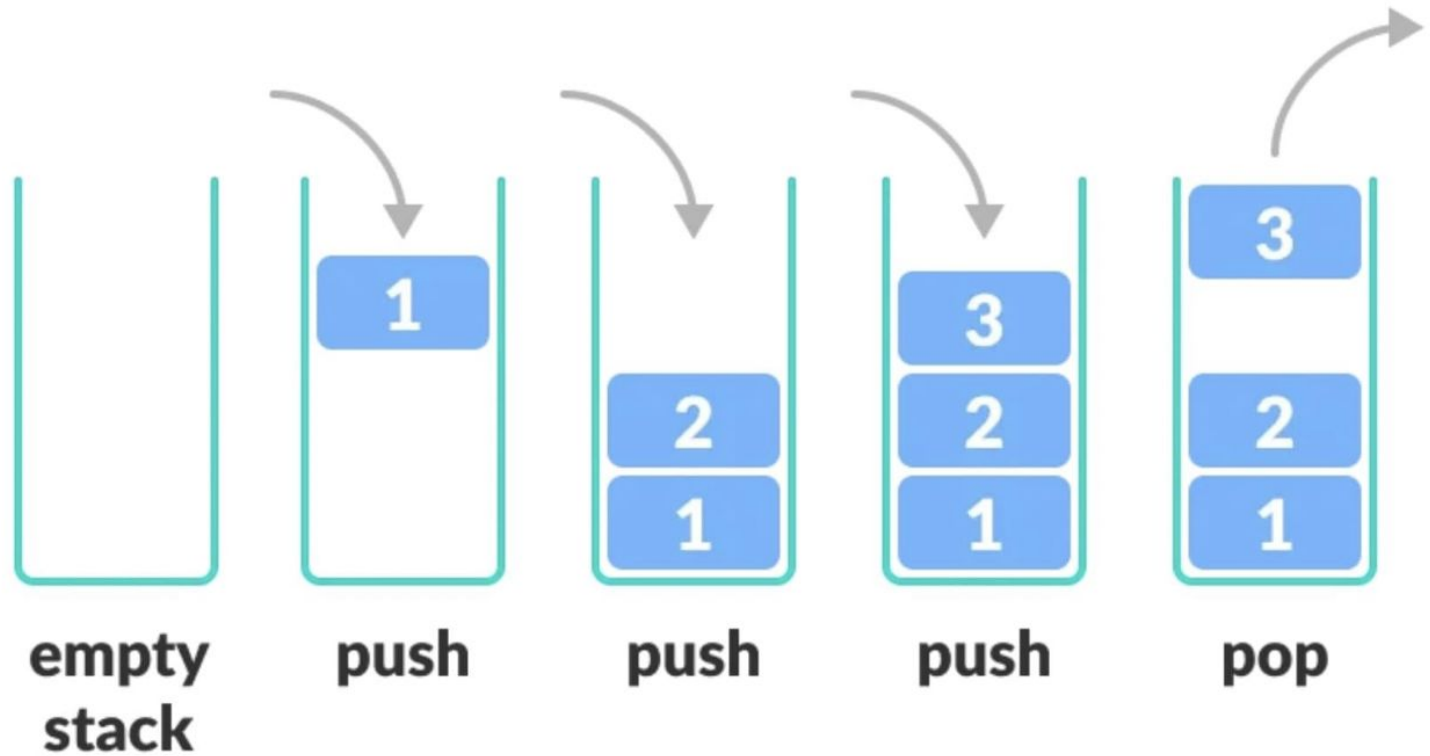
```
const emp = {  
  emp_name: "Anil",  
  company: "TCS"  
}  
  
const emp_address = {  
  city: "Pune",  
  pin: 431202  
}  
  
// 2. Merge an Object  
const merged_obj = Object.assign({}, emp, emp_address);  
console.log(merged_obj);
```

What is Stack ?


First in Last out → FILO



Stack data structure operations

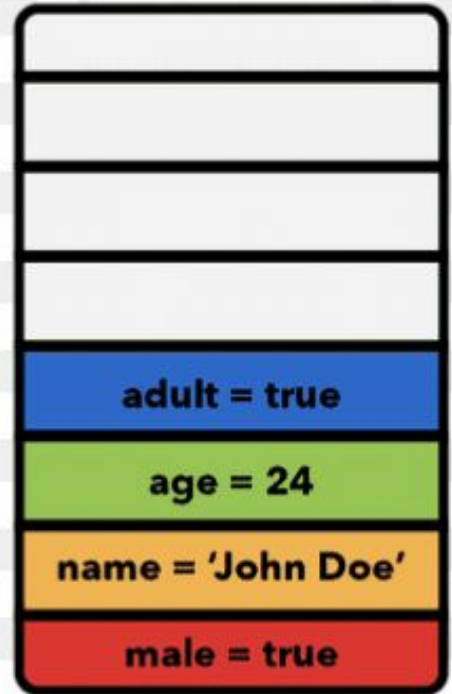


Memory allocation for primitive data types



```
const male = true  
const name = 'John Doe'  
const age = 24  
const adult = true
```

Stack





```
const person = {  
  id: 1,  
  name: 'John',  
  age: 25,  
}
```

```
const dog = {  
  name: 'puppy',  
  personId: 1,  
}
```

```
function getOwner(dog, persons) {  
  return persons.find((person) =>  
    person.id === dog.person  
  )  
}
```

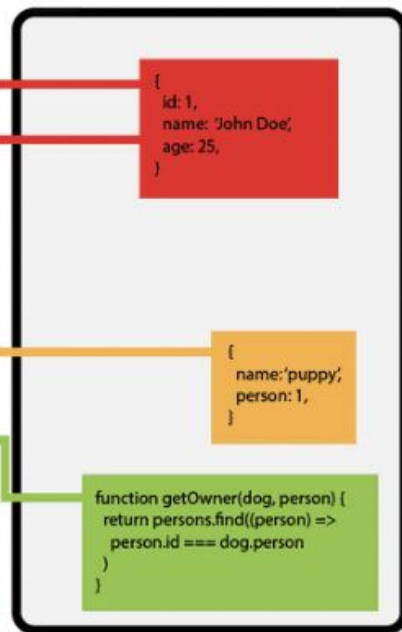
```
const name = 'John';
```

```
const newPerson = person;
```

Stack



Heap



Stack vs Heap

Stack

Primitive values and references

Size is known at compile time

Allocates a fixed amount of memory

Heap

Objects and functions

Size is known at run time

No limit per object

Shallow cloning or Copy

When a reference variable is copied into a new reference variable using the assignment operator (=), A shallow copy of the reference object is created

JS does a shallow copy for the object data types (Non primitive data types)
Automatically and not for the primitive data type values

As we used here assignment operator(=) and assigned 'car' object to 'vehicle' Hence it performed shallow copy.

So when we change the 'color' property of 'vehicle' object it also change the 'color' property of 'car' object. Not only that even we try vice versa it will change the other object value.

Opps this is the problem.. How to solve then ? Hey don't worry we can perform deep copy

```
// shallow copy

let car = {
  brand : "Tata",
  color : "Black",
}

let vehicle = car

console.log(car, vehicle)    // { brand: 'Tata', color: 'Black' } { brand: 'Tata', color: 'Black' }

vehicle.color = "Red"
console.log(car, vehicle)    // { brand: 'Tata', color: 'Red' } { brand: 'Tata', color: 'Red' }

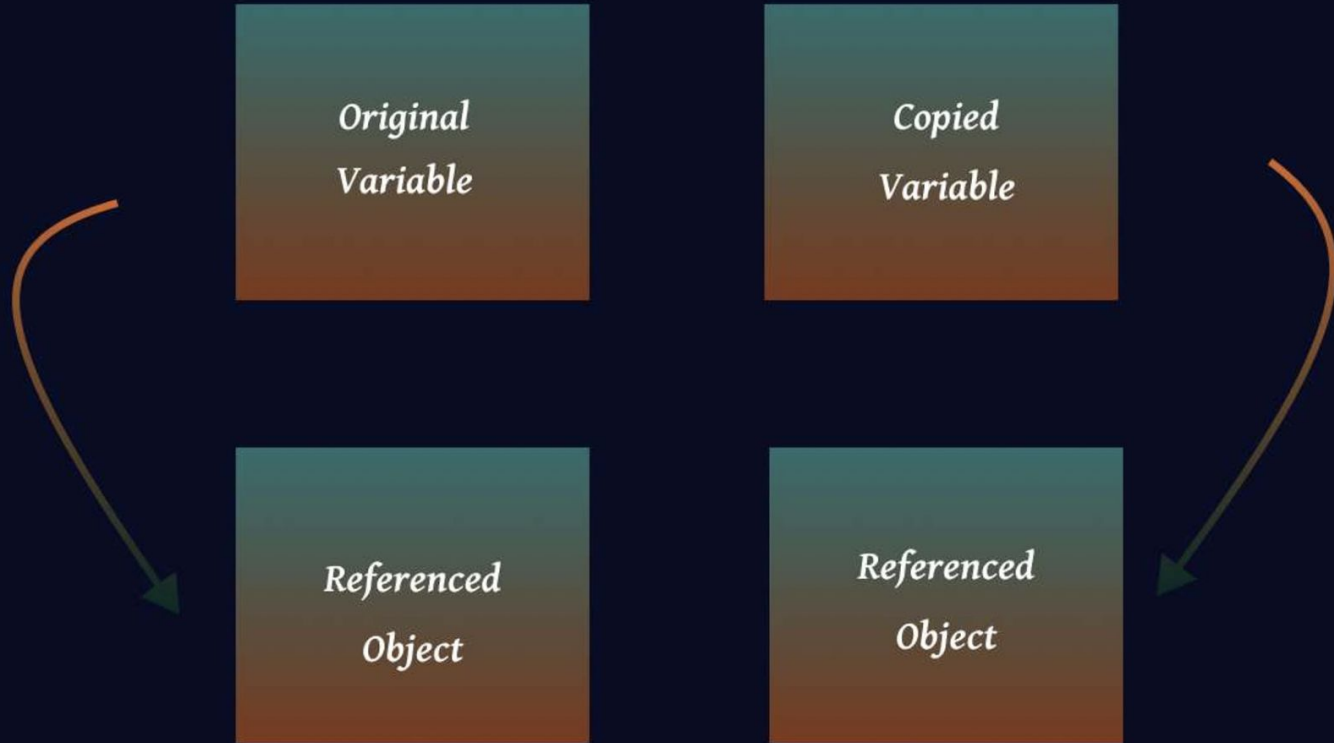
car.color = "Blue"
console.log(car, vehicle)    // { brand: 'Tata', color: 'Blue' } { brand: 'Tata', color: 'Blue' }
```

Assignment 0A: Object, Array Freeze and merge

1. Create an object using literals namely → personalDetails with your actual detail.
2. Create an object using literals → collegeDetails with your college details
3. Merge these two objects and log details on console with meaning message.
4. Create an array of your friend names and freeze it.
5. Iterate step 4 array using for of loop and log friend names on console
6. Given a string company = "Codemind Technology";
 - a. Write a code to reverse only → Technology word
 - b. Expected output → "Codemind ygolohceT"

Deep copy:

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



Deep Cloning or Copy

- A deep copy makes a copy of all the members of the original object, allocates separate memory location for the new object and then assigns the copied members to the new object .
- In this way both the objects are independent of each other and in case of any modification to either one the other is not affected.
- Also if one of the object is deleted the other still remains in the memory
- This new variable is completely disconnected from the original variable.

Note: JS always does a deep copy for primitive data types variables by default, So we don't have to worry about shallow copy or deep copy for primitive data types

There are multiple ways to perform deep copy

1. Spread Operator (...)

~~2.~~ JSON.parse() and JSON.stringify()

Let us perform the deep copy using spread operator

```
// Partial Deep Copy
```

```
let car = {  
  brand : "Tata",  
  color : "Black",  
}
```

```
let vehicle = {...car}
```

```
console.log(car, vehicle) // { brand: 'Tata', color: 'Black' } { brand: 'Tata', color: 'Black' }
```

```
vehicle.color = "Red"
```

```
console.log(car, vehicle) // { brand: 'Tata', color: 'Black' } { brand: 'Tata', color: 'Red' }
```

When we perform the deep copy using spread operator it fails when the object contains nested object.

In this we can use other method → `JSON.parse()` and `JSON.stringify()`

```
// deep copy
let car = {
  brand : "Tata",
  model:{
    type:"suv",
  }
}

let vehicle = JSON.parse(JSON.stringify(car))

console.log(car, vehicle)  // { brand: 'Tata', model: { type: 'suv' } } { brand: 'Tata', model: { type: 'suv' } }

vehicle.model.type="Hatchback"
console.log(car, vehicle); // { brand: 'Tata', model: { type: 'suv' } } { brand: 'Tata', model: { type: 'Hatchback' } }
```

Assignment B → Object cloning and Traversing, File→ 10_objectAssignB.js

1. Create the object → 'bankSbi' using literals with at least 4 properties
2. Create the object → 'bankLocation' using literals with properties: street, city, pinCode
3. Clone the step 1 ('bankSbi') and step 2 ('bankLocation') objects using
 - Object.assign()
 - Spread Operator

Note: Log the cloned object details on console with meaning message using strings template

4. Create the object using literals → rateOfInterest with properties
 - homeLoanInterest, personalLoanInterest, dueInterest
5. Merge the step 1, step 2 and step 4 objects into new object namely → sbiDetails
Log all the properties that sbiDetails got after merging with meaning message
6. Traverse this merged object - step 5 using loop

Assignment C: 07_ObjectCloneAssign.js, Please log output on console with step number and meaningful message

Given a array → `const arrayNums = [20, 3, 4, 56, 90, 400, 49]`

1. Perform shallow clone on `arrayNums`, Update cloned array with values → 55, 66 using `push()` and log original and cloned array on console
2. Perform deep clone using spread operator, Update original array i.e `arrayNums` with values 10, 25 at last position and log original and cloned array on console
3. Given other array → `arrayEven = [2, 30, 14, 8]`, Merge or concat this array with '`arrayNums`' using spread operator, log result on console after merge array operation
4. Create the `employee_info` object as shown in snippet→
5. Log the employee details on console like
 - a. Address: Locality, city, state and country
 - b. Mobile numbers
6. Perform deep copy using `JSON.stringify()`
 - a. Update property '`july_month`' salary to 80K on cloned object
 - b. Update property '`country`' to '`United Kingdom`' On original object
 - c. Log - Updated values for original & cloned object on console

```
const employee_info = {
  emp_id: 27,
  emp_name: "John Doe",
  salary: {
    july_month: "40,0000INR",
    aug_month: "50,0000INR",
    jun_month: "65,0000INR"
  },
  address: {
    locality: {
      colony: "OM Vrindavan Society",
      street: "Kanch Pokli, 431202",
    },
    city: "Mumbai",
    state: "Maharashtra",
    country: "India"
  },
  mobiles: ["+91 8600 3456 88", "1800- 4567 32", "+91- 9096 5678 77"]
}
```

Assignment 02: 07_ObjectMergeAssign2.js

Given a object car and carEngine merge or concat these two objects using

1. Object.assign()
2. Spread operator ...

Note:

- Log the step 1 merged object details with meaning message
- Log the step 2 merged object details

```
const car = {  
  carName: "Creta SX",  
  company : "Hundai",  
  launchYear: 2017  
}  
  
const carEngine = {  
  enginePower: "1499CC",  
  maxPower: "113 BHP"  
}
```

Defining Object with 'const keyword'

When we define object as 'const' then after defining object that reference variable will not be changed to point any other object

```
const student = {  
  name: "Mohit",  
  rollNo: 1234  
}  
student = { // Not allowed: TypeError: Assignment to constant variable.  
  city: "Pune"  
}
```