1) var a = [];
2) var b =  new Array(0);

You are instantiating the var b with an object of type Array using a class Array.
Array - Constructor Function (Class)
Object
Function
String

Constructor function is a function which is similar to classes in other languages which used to create an instance of the type of the constructor function.

**Objects**

var a = {};
var b = new Object();
var c = Custom constructor function;

## Functions

## Function declaration

```
function a () {
	console.log(“a”);
}
```

**a()**

**function a (x) {**
**	console.log(x);**
**}**

**a(10);**

```
function ServeBeverage(drink, quantity){
    console.log('I want '+ quantity + " " + drink)
}
```

## Function Expression

**var fn = function  (x) {**
**	console.log(x);**
**}**

```
function  (x) {
      console.log(x);
}
```

**Anonymous function**

**Constructor functions-** Is a function that is used to create an instance of the type of the constructor function and usually it is named with a capital letter.

```
function Dog(name, breed) {
      this.name= name;
      this.breed = breed;
}
```

```
var d1 = new Dog();
```

## What is this ? (Discuss more about this)

This keyword refers to the current calling object of a function

**Reverse a string**

**Spread operator**

**…**
**Deep copy**
**var a = [12,3,2,1,4];**

**var b = [...a]; // [12,3,2,1,4];**

**Shallow copy**
**Deep copy**

**Assignment by copy(Primitive)**
**Assignment by reference(Object)**

**Shallow copy**
**Var a = 10;**
**Var b  = a;**

**…a**
**12,3,2,1,4**

var b = […a]

Var b = a;

**Execution Context in Javascript**

```
var a = 2
var b = 3
function add(num1, num2){
    var ans = num1+num2
    return ans;
}
var addition = add(4, 5)
console.log(addition) // print 9
add(addition, 4)
let add1 = addition(a, b)
let add2 = addition(5, 6)
console.log(add1) // prints 5
console.log(add2) // prints 11
```

There are two types of execution contexts one is the global context and second is function execution context

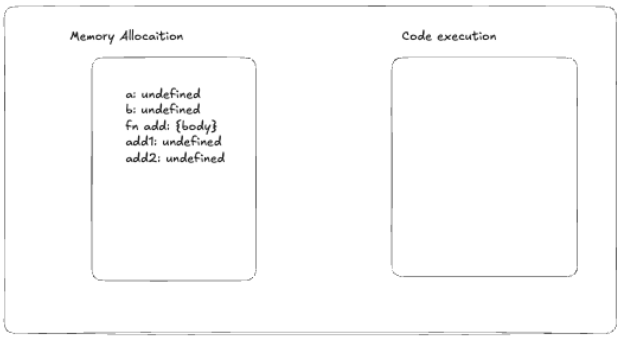
**Memory Allocation**: Every particular variable is assigned with undefined value initially. Initially, no value is assigned to the variable, only memory is allocated. And every function will get its whole body in this phase.

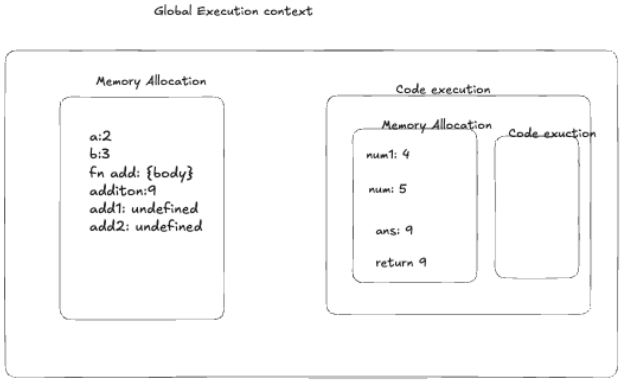
**Code Execution**: Code will get executed in this phase. Now the values of variables are assigned. When the function is called and it starts executing then another memory allocation and the code block is created For every function calling. basically another execution context will be created for the function. And execution context of the whole program is known as the global execution context, and for a function it is called function's Execution Context
Variables declared inside the function are written inside the execution context of that particular function.

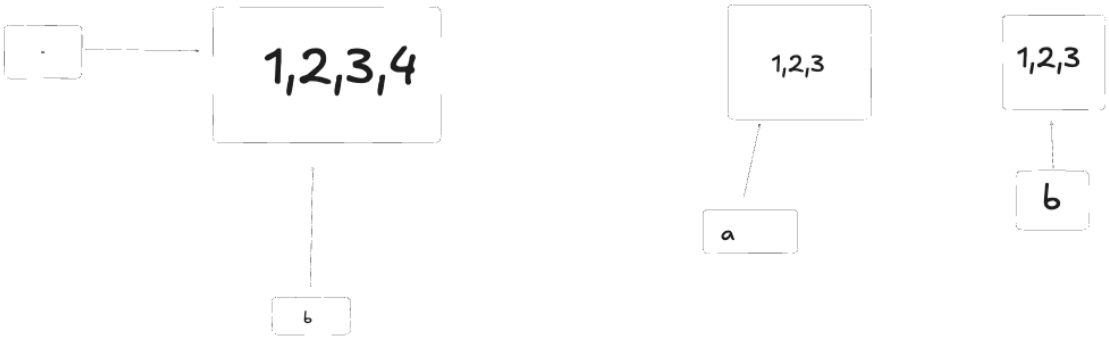
```
var a = 2
var b = 3
function add(num1, num2){
    var ans = num1+num2
    return ans;
}
var addition = add(4, 5)
console.log(addition) // print 9
add(addition, 4)
```

let add1 = add(a, b)

```
var a = 2
var b = 3
function add(num1, num2){
    var ans = num1+num2
    return ans;
}
var addition = add(4, 5)
console.log(addition) // print 9
add(addition, 4)
let add1 = addition(a, b)
let add2 = addition(5, 6)
console.log(add1) // prints 5
console.log(add2) // prints 11
```




let add2 = add(5, 6)
console.log(add1) // prints 5

console.log(add2) // prints 11

Global execution context

Memory allocation code execution
A: undefined
B: undefined
Fn: {body}
Addition: undefine
Add1: undefined
Add2: undefined

Code execution

Memory allocation code execution
A: 2
B: 3
Fn: {body}
Addition: 9
Add1: 5
Add2: 11

**Function execution context(**var addition = add(4, 5)**)**

Memory allocation code execution

Num1: undefined
Num2: undefined
Ans: undefined

Memory allocation code execution
Num1: 4
Num2: 5
Ans: 9
Return 9

Function Execution context (var add1 = add(a, b))

Memory allocation code execution
Num1: undefined
Num2: undefined
Ans: undefined

Memory allocation code execution
Num1: 2
Num2: 3
Ans: 5
Return 5

Function Execution context (var add2 = add(5, 6))

Memory allocation code execution
Num1: undefined
Num2: undefined
Ans: undefined

Memory allocation code execution
Num1: 5
Num2: 6
Ans: 11
Return 11

**Difference between let, const, var**

**Let**
Don't let you redeclare a variable

**Const**
Doesn't let you redeclare and doesn't even let you reassign a value

Let and const both are block scoped

Anything wrapped in curly braces is known as block scoped

redeclaration doesnt happen within the block but happens in another block?

But var is function scoped

**For next class**

Hoisting

Global scope

Local scope

Temporal dead zone

lexical env.