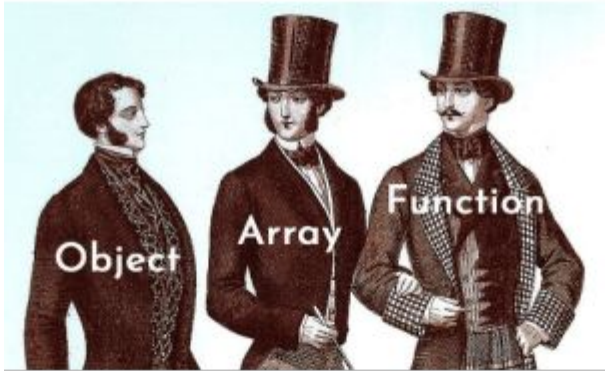
How Js works

Key points to remember

There are two types of data types

Primitive Data Types

Non Primitive Data Types



Difference between primitive and non-primitive data type

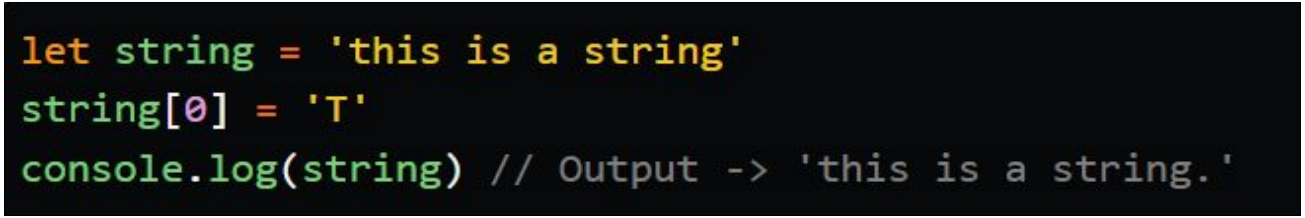
Primitive

1. They hold the actual value.(in js its called as by value)

Non-primitive

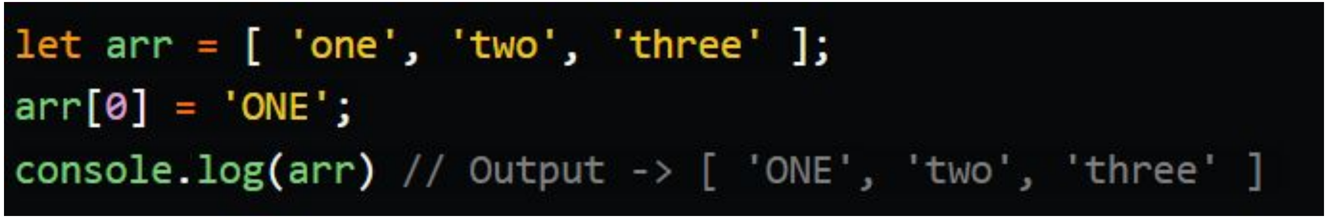
1. They hold the location of value.(in js its called as by reference).

For primitive data type - By value- The original STRING is not modified on changes in other variables.



```
let string = 'this is a string'
string[0] = 'T'
console.log(string) // Output -> 'this is a string.'
```

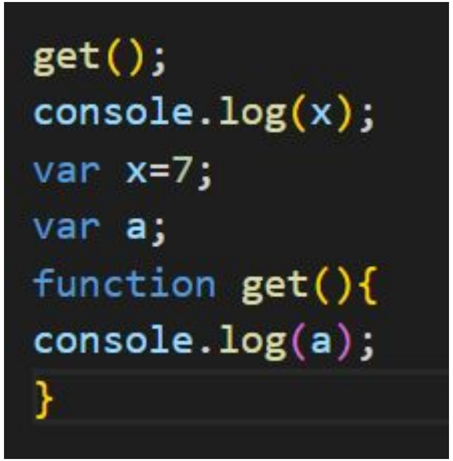
For non primitive data type- By reference- the original variable gets modified on changes in other variables.



```
let arr = [ 'one', 'two', 'three' ];
arr[0] = 'ONE';
console.log(arr) // Output -> [ 'ONE', 'two', 'three' ]
```

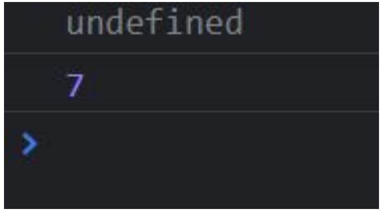
Hoisting

It means Js can access all variables and function even before they were initialized.



```
get();
console.log(x);
var x=7;
var a;
function get(){
console.log(a);
}
```

output



```
  undefined
  7
>
```

MEMORY or VARIABLE CREATION PHASE

- Here HOISTING happens
- MEMORY IS ALLOCATED for variables
- Here value of variables is UNDEFINED

CODE EXECUTION PHASE

Here variable values are assigned

Here code is executed

Execution context

Everything in Js happens inside an execution context.

It has two phases

1. Creation phase
2. Code execution phase

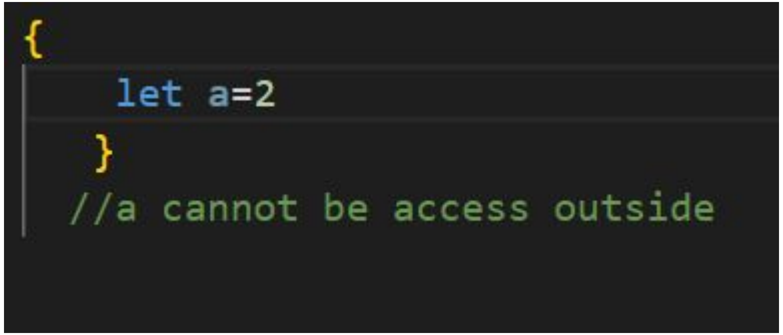
Scopes

There are 3 types of scopes.

1. Block scope.
2. Local scope.
3. Global scope.

Block scope

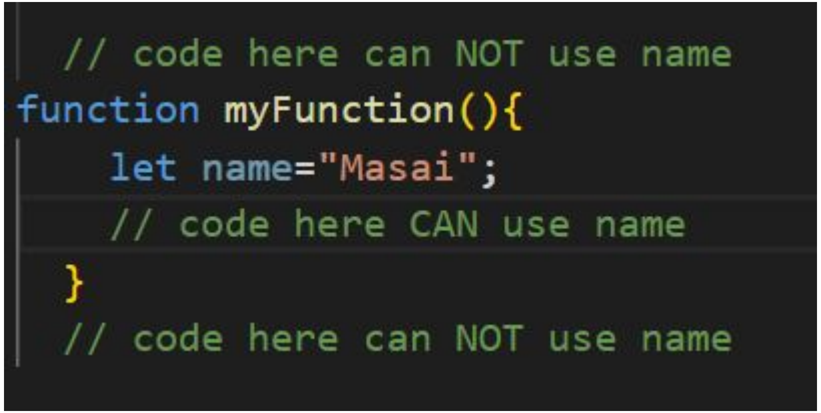
Variables declared inside a { } block cannot be accessed from outside the block.



```
{
    let a=2
  }
 //a cannot be access outside
```

Local scope

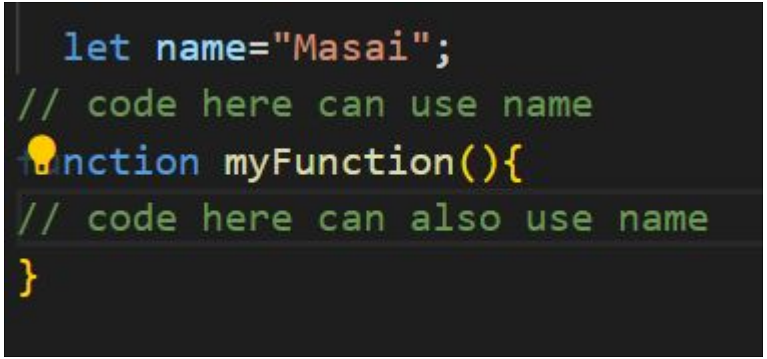
Variables declared within a JavaScript function, become LOCAL to the function.



```
  // code here can NOT use name
function myFunction(){
    let name="Masai";
    // code here CAN use name
  }
  // code here can NOT use name
```

Global scope

A variable declared outside a function, becomes GLOBAL.



```
  let name="Masai";
// code here can use name
function myFunction(){
// code here can also use name
}
```