Strings In JavaScript

Strings:-

Strings are the characters encoded within single quotes, double quotes, template literals(backticks)

Strings are primitive and immutable data types.

Now Let Us Know 🫠

🤔 What is Primitive and What is Immutable?

First Let us define what is immutability and what is mutability?

Mutability: -

Mutability refers to the changing of the value once it is created

Non-Mutability: -

Non-Mutability refers to the un-changing of the value once it is created

Now what is primitive? Also called Non-Reference Variables?

Primitive means, it is immutable, it holds a single and simple values and contains n number of characters.

In the concept of Strings, Strings are primitive until Strings are created using constructor which means creating a String with new keyword called as an object

How Primitives are immutable

For Example, we are creating a string like this:

let a=”hello”

let b=a

b=”bye”

console.log(a)

console.log(b)

Now what we think, we assigned b to a, so if we change the value of b then automatically the value of a will changes because of assigning a variable to the b variable.

But it is wrong assumption. Why?

- Because JavaScript creates a new copy of already initialized value and stored in new variable's memory location. JavaScript Performs actions on that new copy of variable instead of original variable.
- In the above example,
- First, we have created one variable:- let a="hello"
- We manually created another variable assigned with another variable which hold value 10 (copy) :- let b=a;
- Next we modified latest created variable which hold copied value of another variable: - b="bye"
- Now JavaScript changes the value of b not a because it is primitive we cannot change the value of original string variable.
- When you print: - console.log(a); you will get an output : hello because of primitive nature in JavaScript
- When you print: - console.log(b); you will get an output : bye because of JavaScript changes the value of variable which holds the copied value of another variable

Real Life Example: -

You and you are friend are writing one quotation, you copied your friend answer like your friend written "I am single", you have copied that answer and you have modified it in your way like "I am alone", is does mean your friend answer will also change from "I am single" to "I am alone", No right!, you have modified your copied answer not your friend answer right!. Same way in primitive also, when you create a String and assign it to a variable, JavaScript creates new copied variable and modify that variable if any operations performed on copied variable

Technical Explanation: -

Primitive Data Types are stored directly in the call stack (a part of memory that is fast and organized in a Last-in, First-Out manner) associated with the variable holding the value.

When you assign a primitive value to another variable, a new copy of that value is created and stored in the new variables's memory location.

Now What is non-primitive? Also called Reference Variables?

Non-Primitive means, it is mutable, it holds a collection of values and contains n number of types of data's

In the concept of Strings, Strings becomes non- primitive when Strings are created using constructor which means creating a String with new keyword called as an object

How Non-Primitives are mutable

For Example, we are creating a string in an object which is an non-primitive Data Type:

```
let a={
        name:"sri"
}
let b=a;
b.name="srinivas"
console.log(a.name)
console.log(b.name)
```

Now what we think, strings are immutable even we assigned b to a in the above case, then the value of a will not change.

But it is wrong assumption. Why?

- Because in the above case we are creating variables in the form of an objects which is non-primitive Data Type
- So, JavaScript instead of creating a new copy of new variable, it creates a reference which holds its memory address, when you change the value of b then it also reflect the value of a because of both references which memory addresses are equal
- In the above example,

- First we have created one variable in object Data Type:- let a and created property called name and assigned a value called “sri”
- We manually created another variable and assigned variable a to a b variable which hold reference memory address:- let b=a;
- Next we modified latest created variable which hold referenced memory address another variable:- b=”srinivas”
- Now JavaScript changes the value of b as well as a because it is non-primitive we can change the value of original string variable.
- When you print:- console.log(a); you will get an output : srinivas because JavaScript changes the value of variable which holds the same reference another variable
- When you print:- console.log(b); you will get an output : srinivas because of non-primitive nature in JavaScript

Real Life Example: -

You permanent address and reference address is same in aadhar card, when you change your permanent address, you also have to change your reference address in aadhar card right! If you not change your reference address in your aadhar card so many problems will occur, like that in non-primitive when you update one variable, it’s reference variable also gets updated automatically.

Technical Explanation:-

Non-Primitive Data Types are stored in heap memory, a larger pool for flexible storage. The variable holds a reference (memory address) to the data in the heap, not the data itself. When you assign a reference type variable to another, both variables points to the same underlying object in the memory. Changes made through one variable will be reflected in the other.