Lic. Gabriel Chaldú

Inmutabilidad y Mutabilidad

Tipos Primitivos: En JavaScript, los tipos primitivos incluyen “string”, “number”, “boolean”, “null”, y “undefined”. Estos son inmutables. Esto significa que una vez que se crea un valor primitivo, no se puede cambiar. Si realizas una operación sobre uno de estos tipos, obtendrás un nuevo valor en lugar de modificar el existente.

```
let a = 10;
let b = a; // b es una copia del valor de a
b = 20; // b se cambia a 20, pero a sigue siendo 10
```

Mutabilidad en Tipos de Datos Complejos

Tipos de Datos Complejos o Compuestos: Estos incluyen objetos, arrays, funciones, y otros como “Date”. Estos son mutables, lo que significa que su contenido o estructura puede cambiar después de su creación. Puedes modificar las propiedades de un objeto, los elementos de un array, etc., sin necesidad de crear un nuevo objeto o array.

```
let obj = { nombre: "Gabriel" };
obj.nombre = "Chaldu"; // Se modifica la propiedad nombre de obj
```

Inmutabilidad de Referencia

Variables con “const”: Cuando declaras una variable con “const”, la referencia a ese valor es inmutable, es decir, no puedes reasignar la variable para que apunte a otro valor u objeto. Sin embargo, el contenido interno de los objetos o arrays a los que apunta esa referencia puede cambiar.

```
const arr = [1, 2, 3];
    arr.push(4); // Esto es válido, modifica el contenido del array
    arr = [5, 6, 7]; // Esto dará un error, porque intentas cambiar la referencia de arr
```

Lic. Gabriel Chaldú

En resumen:

Inmutables:

Strings, números, booleanos, “null”, “undefined”: Estos tipos no se pueden modificar una vez creados. Cualquier operación que "modifique" estos valores crea un nuevo valor.

Entiendo tu confusión. La clave está en diferenciar la variable que almacena el valor y el valor en sí.

Explicación:

1. Inmutabilidad del valor:

Los strings, números, booleanos, “null”, y “undefined” son inmutables, lo que significa que el valor en sí no puede ser modificado una vez creado.

Ejemplo con un string: Si tienes “let nombre = 'Pepe';”, la cadena “'Pepe'“no puede ser alterada. Si intentas cambiar una letra de “'Pepe'“, no puedes hacerlo directamente.

2. Reasignación de variables:

Si escribes “nombre = 'Maria';” no es modificar el string “'Pepe'“, sino reasignar la variable “nombre” para que apunte a un nuevo string “'Maria'“.

La variable “nombre” ahora apunta a un nuevo espacio en memoria donde está almacenado “'Maria'“, mientras que “'Pepe'“permanece inmutable (aunque ya no esté referenciado por “nombre”).

La inmutabilidad se refiere al valor en sí y no a la variable que lo contiene.

Mutables:

Objetos, arrays, funciones, “Date”: Estos pueden ser modificados en su estructura o contenido después de haber sido creados. Puedes cambiar las propiedades de un objeto, agregar o eliminar elementos de un array, etc.

Inmutabilidad de Referencia con “const”:

“const” hace que la referencia a un objeto o array sea inmutable, pero no impide la mutabilidad del contenido del objeto o array al que hace referencia.