**Tecnologías para el Desarrollo de Aplicaciones Web.**

**Declaración de variables**: var, let y const.

Una variable es un espacio en memoria donde se almacena un dato. Los nombres de las variables se construyen caracteres alfanuméricos, el carácter subrayado (\_) y el carácter $, y no pueden comenzar por un carácter numérico. No se pueden utilizar símbolos o caracteres operacionales, como el signo +, el signo -, o el espacio en blanco. Algunos ejemplos son los siguientes:

* Email
* nombreDePila
* \_edad
* $valor
* Algo$\_Cosa

Se evitan nombres reservados en variables, como return o for. Algunos nombres de variables no permitidos, por ejemplo, son:

* 15años
* Mi correo
* return
* for
* buenos-días
* Ox%xo

Los nombres de las variables en Javascript son sensibles a mayúsculas y minúsculas, es decir, no es igual la variable minombre que la variable miNombre.

La declaración de variables en Javascript, consiste en definir el uso de una variable, sin embargo no es necesario, como sucede con otros lenguajes como Java, C, C# y otros.

**Uso de var**

El alcance o ámbito de var significa dónde estas variables están disponibles para su uso. Las declaraciones de var tienen un alcance global o alcance local en una función.

El alcance es global cuando una variable var se declara fuera de una función. Esto significa que cualquier variable var que se declare  fuera de un bloque de funciones está disponible para su uso en toda la ventana. var tiene un alcance local cuando se declara dentro de una función, es decir, que sólo se puede acceder a ella dentro de esa función. Por ejemplo:

var greeter = "hey hi";

function newFunction() {

var hello = "hello";

}

Allí, greeter tiene un alcance global porque existe fuera de la función, mientras que hello tiene un alcance de función. No se puede acceder a la variable hello fuera de una función. Por ejemplo, se genera un error en el siguiente código:

var tester = "hey hi";

function newFunction() {

var hello = "hello";

}

console.log(hello); // error: hello is not defined

Las variables declaradas con var se pueden volver a declarar y actualizar. Por ejemplo:

var greeter = "hey hi";

var greeter = "say Hello instead";

var greeter = "hey hi"; // y esto también

greeter = "say Hello instead";

El Hoisting (elevación) es un mecanismo de JavaScript en el que las variables y las declaraciones de funciones se mueven a la parte superior de su alcance antes de la ejecución del código. Esto significa que si se hace lo siguiente:

console.log (greeter);

var greeter = "say hello"

Se interpreta así:

var greeter;

console.log(greeter); // greeter no está definido.

greeter = "say hello"

Así, las variables var se elevan a la parte superior de su alcance y se inicializan con un valor de undefined.

La declaración con var tiene un problema. Por ejemplo:

var greeter = "hey hi";

var times = 4;

if (times > 3) {

var greeter = "say Hello instead";

}

console.log(greeter) // "say Hello instead"

Como times>3 devuelve true, greeter se redefine a say Hello instead. Si bien esto no es un problema si desea redefinir greeter, se convierte en un problema cuando la variable greeter ya se definió antes. Si no se ha utilizado greeter en otras partes de su código, es posible que el resultado no sea el esperado.

**Uso de let**

Una mejora de la declaración var es con let. Resuelve el problema se comentó anteriormente con var.

El ámbito o alcance de let es de bloque. Un bloque se define con {}. Una variable declarada con let en un bloque sólo está disponible para ese bloque. Por ejemplo:

let greeting = "say Hi";

let times = 4;

if (times > 3) {

let hello = "say Hello instead";

console.log(hello);// "say Hello instead"

}

console.log(hello) // hello is not defined

Al usar hello fuera de su bloque se devuelve un error.

Al declarar una variable con let se le puede actualizar, pero no volver a declarar. Al igual que var, una variable declarada con let puede actualizarse dentro de su alcance, pero a diferencia de var, una  variable let no se puede volver a declarar dentro de su alcance. Por ejemplo:

let greeting = "say Hi";

greeting = "say Hello instead";

Lo siguiente devolverá un error:

let greeting = "say Hi";

let greeting = "say Hello instead"; //error: Identifier 'greeting' has already been declared

Sin embargo, si la misma variable se define en diferentes ámbitos, no habrá ningún error:

let greeting = "say Hi";

if (true) {

let greeting = "say Hello instead";

console.log(greeting); // "say Hello instead"

}

console.log(greeting); // "say Hi"

El error se debe a que ambas instancias se tratan como variables diferentes ya que tienen alcances diferentes. Al usar let, no hay problema si ha usado un nombre para una variable antes, ya que una variable existe solo dentro de su alcance. Una variable no se puede declarar más de una vez dentro de un ámbito.

El Hoisting con let, al igual que con var,  las declaraciones let se elevan a la parte superior. A diferencia  de var lo que se inicializa como undefined, la  palabra clave let no se inicializa. Entonces, si intenta usar una  variable let antes de la declaración, obtendrá un Reference Error.

**Uso de const**

Las declaraciones const comparten algunas similitudes con las  declaraciones let. Las declaraciones const tienen un alcance de bloque. Al igual que las  declaraciones let, const sólo se puede acceder a las declaraciones dentro del bloque en el que se declararon.

Con const no se puede actualizar ni volver a declarar. Esto significa que el valor de una variable declarada con const sigue siendo el mismo valor dentro de su alcance. No se puede actualizar ni volver a declarar. Entonces, si se declara una variable con const, tampoco se puede hacer lo siguiente:

const greeting = "say Hi";

greeting = "say Hello instead";// error: Assignment to constant variable.

Tampoco lo siguiente:

const greeting = "say Hi";

const greeting = "say Hello instead";//error:Identifier 'greeting' has already been declared

Cada  declaración const, por lo tanto, debe inicializarse en el momento de la declaración. Este comportamiento es algo diferente cuando se trata de objetos declarados con const. Si bien un  objeto const no se puede actualizar, las propiedades de estos objetos se pueden actualizar. Por lo tanto, si declaramos un  objeto const como el siguiente:

const greeting = {

message: "say Hi",

times: 4

}

Pero que no se puede hacer lo siguiente:

greeting = {

words: "Hello",

number: "five"

} // error: Assignment to constant variable.

Sin embargo, si se puede hacer esto:

greeting.message = "say Hello instead";

Esto actualizará el valor de greeting.message sin devolver errores.

En el Hoisting de const, al igual que let,  las declaraciones const se elevan a la parte superior, pero no se inicializan.

Por lo que las diferencias son las siguientes:

* las declaraciones var tienen un alcance global o un alcance de función, mientras que let y const tienen un alcance de bloque.
* las variables var se pueden actualizar y volver a declarar dentro de su alcance;  las variables let se pueden actualizar pero no volver a declarar;  las variables const no se pueden actualizar ni volver a declarar.
* Todos ellas, var, let y const, están elevados a la parte superior de su ámbito de aplicación. Pero mientras que  las variables var se inicializan con undefined,  las variables let y const no se inicializan.
* Si bien var y let pueden declararse sin inicializarse, const debe inicializarse durante la declaración.