JS

PROFE VIDERMID

Programación en Java Script

(Parte 2)

Facilitador: Ing. Esp. Vidermid Sánchez

@vidermid +584147106623

@ingenieriadigitalsc

+584147464801

Ingeniería Digital

HOISTING

PROFE VIDERMID

Hoisting (elevación), es un término que no encontrará utilizado en ninguna especificación previa a ECMAScript® 2015 Language Specification. El concepto de Hoisting fue pensado como una manera general de referirse a cómo funcionan los contextos de ejecución en JavaScript (específicamente las fases de creación y ejecución).

En JavaScript, hoisting permite usar funciones y variables antes de que se hayan declarado. En este post, aprenderemos qué es el hoisting y cómo funciona.

Para el caso de var permite tener acceso al valor desde antes de la línea que la declara. Mientras que para const y let muestra como que no es posible tener acceso al valor. En ambos casos deberá existir la declaración de la variable, de no ser así, dirá que la variable no existe o no esta definida.

Ejemplo 1:

```
console.log(nombre);          // undefined
var nombre = 'Vidermid';
console.log(nombre);          // Vidermid
```

Ejemplo 2:

```
console.log(nomb); // Uncaught ReferenceError: Cannot access 'nomb' before initialization
let nomb = 'ProfeVidermid';  // lo mismo para variables declaradas con const
```

HOISTING

PROFE VIDERMID

La zona muerta temporal o TDZ, se refiere al estado donde las variables son inaccesibles. Se encuentran en el scope, pero no han sido declaradas todavía. Las variables let y const existen en la TDZ desde el inicio de su ámbito de aplicación, su scope, hasta que son declaradas.

```
console.log(nombre);          // ReferenceError porque estamos en la TDZ de nombre
let nombre = 'Vidermid';      // Final de TDZ de nombre
```

El Hoisting también se aplica a las funciones.

```
mostrar('Vidermid');

function mostrar (nombre){
    console.log('Hola'+nombre);
}
```

Nota: El Hoisting no aplica para clases que no han sido declaradas o definidas.

ACUMULADORES Y CONTADORES

PROFE VIDERMID

Contadores, variable numérica que funciona dentro de ciclo repetitivo y se encarga de almacenar la cantidad de veces de algún evento. Su incremento es fijo.

Sintaxis: c=0;

```
ciclo
{
c=c+incremento fijo;
}
alert('La cantidad es: '+c);
```

Acumuladores, variable numérica que funciona dentro de ciclo repetitivo y se encarga de almacenar la sumatoria de montos numéricos. Su incremento es variable.

Sintaxis: a=0;

```
ciclo
{
a=a+incremento variable;
}
alert('El total es: '+a);
```

EJERCICIOS

PROFE VIDERMID

Ejemplo de Validación con el ciclo do-while:

```
let numero;
do{
   numero=parseInt(prompt('Ingrese un número: '));
}while(isNaN(numero));
alert(`${numero} es ${isNaN(numero)}`);
```

```
let cadena;
do{
   cadena=prompt('Ingrese una cadena: ');
}while(cadena.length<1);
alert(`${cadena} es correcto`);
```

EJERCICIOS

PROFE VIDERMID

Elabore un Script en Js que permita solicitar para N clientes los siguientes datos:
Rif, Apellido, Nombre, Edad, Genero, cantidad de productos a llevar, precio único de los productos.

La Empresa otorga los siguientes descuentos sobre el monto bruto:

- Si el cliente lleva mas de 20 productos se le aplicará el 25%.
- Si el cliente lleva entre 1 y 20 productos se le aplicará el 15%.

Calcular y mostrar por pantalla lo siguiente:

- Cantidad de clientes de genero “F” o “f”.
- Cantidad de clientes con mas de 49 años.
- Cantidad de Clientes con menos de 50 años.
- Monto neto a cancelar por cada cliente.
- Monto general que ingresó a la Empresa por todas las ventas realizadas.

MANEJO DE ARREGLOS Y ESTRUCTURAS

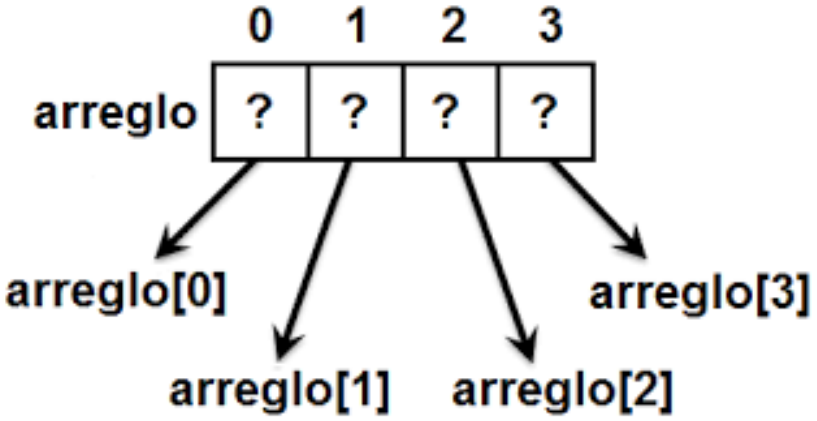
PROFE VIDERMID

En JS los Arrays se definen con los símbolos de corchetes [ ]

Su declaración puede ser:

```
let  a=[];
console.log (a);
document.write(a.length);

var n=[35,22,100,7,88,44];
const f=[“Hola”,”Hi”,“Epale”,”Que onda”];
let g=[true,”Saludo”,4,[22,63,24,10],”a”];
document.write(g[2]);
```



Ingeniería Digital

EJERCICIOS

PROFE VIDERMID

```
let z=Array.of(33,56,43,22);
console.log('Arreglo original: ');
console.log(z);
console.log('A continuación vemos Ejemplos de salidas y operaciones con arreglos: ');
console.log('Arreglo creado como objeto con of: '+z);
console.log('Valor especifico en la posición "2" del arreglo: '+z[2]);
z.push(88);
console.log('Arreglo con un nuevo elemento al final: '+z);
z.pop();
console.log('Arreglo menos el elemento de la ultima celda: '+z);
```

EJERCICIOS

PROFE VIDERMID

```
let z=Array.of(33,56,43,22);
document.write('<h2>Valores almacenados en el Arreglo: </h2>');

z.forEach(function(ele,ind){
        document.write('El elemento '+ele+' se almacena en la posición: '+ind+'<br>');
});
```

EJERCICIOS

PROFE VIDERMID

```
let z=Array.of(33,56,43,22);
let x=0,pos=0;
document.write('<h2>Valores almacenados en el Arreglo: </h2>');

for(x=0;x<z.length;x++){
        document.write('El elemento '+z[x]+' se almacena en la posición: '+x+'<br>');
}
```

EJERCICIOS

PROFE VIDERMID

```
let nota=[15,18,10,08,17,20];
let i=0,n=nota.length;
for(i=0;i<n;i++)
{
   document.writeln('El elemento '+i+' tiene un valor de '+nota[i]+'<br>');
 }
```

EJERCICIOS

PROFE VIDERMID

```
let nota=[15,18,10,08,17,20];
let i=0,n=nota.length;

document.writeln('-----------------ORIGINAL----------------------- <br>');
for(i=0;i<n;i++)
{
   document.writeln('El elemento '+i+' tiene un valor de '+nota[i]+'<br>');
 }

document.writeln('-----------------ORDENADO DE MAYOR A MENOR----------------------- <br>');
nota.sort(function(a, b){return b - a});
for(i=0;i<n;i++)
{
   document.writeln('El elemento '+i+' tiene un valor de '+nota[i]+'<br>');
}
```

Ingeniería Digital

EJERCICIOS

PROFE VIDERMID

```
let edad=[];
let x=0,n=0,s=0,p=0;
n=prompt('Ingrese cantidad de personas: ');
n=parseInt(n);
for(x=0;x<n;x++)
{
   edad[x]=prompt('Ingrese Edad de la persona '+(x+1)+':');

}
document.write('Las Edades son: <br>');
for(x=0;x<n;x++)
{
   document.write('Persona '+(x+1)+' tiene: '+edad[x]+' años<br>');
   s=s+parseInt(edad[x]);
}
p=parseInt(s)/parseInt(n);
document.write('El promedio de edades de las personas encuestadas es de: '+p.toFixed(2));
```

Ingeniería Digital

MANEJO DE ARREGLOS Y ESTRUCTURAS

PROFE VIDERMID

Las estructuras tipo listas utilizan la nomenclatura de { } para cada registro y puede llevar N atributos, pero todos con un valor asignado, observemos el siguiente ejemplo:

```
let seccion =[{ dni: '52741357',
    apellidos: 'Restrepo Palomino',
    nombres: 'Bryan Alejandro',
    genero: 'Masculino',
    edad: '21',
    estatus: true
},
{   dni: '63842159',
    apellidos: 'Vergara Madeira',
    nombres: 'Fatima Maria',
    genero: 'Femenino',
    edad: '18',
    estatus: true
}];
console.log(`Datos del integrante 1:\nApellidos: ${seccion[0].apellidos}\nEstatus: ${seccion[0].estatus}`);
```

Ingeniería Digital

FUNCIONES

PROFE VIDERMID

En JS existen tres tipos de Funciones:

Funciones declaradas:

```
function bienvenido(nom){
    document.write('Hola '+nom);
}
bienvenido('Maria');
```

Expresión funcional: (puede tener nombre o ser anónima)

```
const comentario= function(nom){
            document.write('Hola '+nom);
}
comentario('María');
```

Funciones flecha:

```
const comentario = (nom) =>{
            document.write('Hola '+nom);
}
comentario('María');
```

nota: las funciones también pueden retornar valores con "return"

FUNCIONES

PROFE VIDERMID

```
function verificar(numero)
{
   let xn=parseInt(numero);
    if (xn>0)
    {
        document.write('Es un número Positivo');
    }else{
        if(xn<0)
        {
            document.write('Es un número Negativo');
        }else{
        document.write('Es un número Neutro');}
    }
}

let valor=prompt('Ingrese un número: ');
verificar(valor);
```

FUNCIONES

PROFE VIDERMID

Elabore un Script JS con el uso de funciones para leer los datos de una persona (apellido, nombre, año de nacimiento, año actual. Calcular la edad y determinar si es mayor o menor de edad (tipo booleano). Muestre por pantalla los datos de identificación y un mensaje que indique si es menor o mayor de edad.

Elabore un Script JS con el uso de funciones y validaciones para registrar N veces el nombre, apellido, nota de matemática, nota de física, nota de química y nota de programación de los estudiantes de una sección. la escala de calificaciones es del 1 al 20. Se desea obtener el promedio de la sección y el promedio individual de cada persona.

Elabore un Script JS con el uso de funciones y validaciones que permita mostrarle al usuario un menú de opciones para seleccionar la operación básica de calculo que desea realizar (sumar, restar, multiplicar o dividir), posteriormente debe ingresar dos números positivos. Al finalizar el proceso de registro y calculo el usuario deberá responder si desea o no realizar otra operación.

EJERCICIOS

PROFE VIDERMID

```
function leer()
{
    let nombre=prompt('Ingrese Nombre: ');
    let apellido=prompt('Ingrese Apellido: ');
    let edad=prompt('Ingrese Edad: ');
    return[nombre,apellido,edad];
}

let [nom,ape,ed] = leer();
```

EJERCICIOS

PROFE VIDERMID

```
function analizar(edad){
    let mayor;
    if (parseInt(edad)>17)
    {
        mayor=true;
    }
    else{
        mayor=false;
    }
    return mayor;
}

let persona=analizar(ed);
```

EJERCICIOS

PROFE VIDERMID

```
function mostrar(nombre,apellido,edad,individuo)
{
   document.write('Su Nombre es:'+nombre+'<br>');
   document.write('Su Apellido es:'+apellido+'<br>');
   document.write('Su Edad es:'+edad+'<br>');
   if (individuo)
   {
    document.write('Usted es Mayor de edad <br>');
   }
   else
   {
    document.write('Usted es Mayor de edad <br>');
   }

}
mostrar(nom,ape,ed,persona);
```