Universidad CENFOTEC
SOMOS LO QUE SABEMOS

TECNICAS #1 JS
DESARROLLO WEB

Elaborado por: Msc. Francisco Jiménez Bonilla

Universidad CENFOTEC
SOMOS LO QUE SABEMOS

¿QUE ES JAVA SCRIPT?

JavaScript es un lenguaje de programación web que se utiliza para crear sitios web dinámicos y aplicaciones básicas.

Una web dinámica es aquella que incorpora efectos como animaciones, eventos o acciones que se activan por algún tipo de interacción con el usuario.

JavaScript es un lenguaje base interpretado, por lo que **NO** es necesario compilar los archivos para ejecutarlos, ya que se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

JS JavaScript

Universidad CENFOTEC
SOMOS LO QUE SABEMOS

SINTAXIS DEL LENGUAJE

Las normas básicas que definen la sintaxis de JavaScript son las siguientes:

- **NO se tienen en cuenta los espacios en blanco y las nuevas líneas**:
  El intérprete de JavaScript ignora cualquier espacio en blanco sobrante, por lo que el código se puede ordenar de forma adecuada para entenderlo mejor.

- **Se distinguen las mayúsculas y minúsculas**:
  El código JavaScript es sensible haciendo distinción entre mayúsculas y minúsculas.

- **No se define el tipo de las variables**:
  Al crear una variable, no es necesario indicar el tipo de dato que almacenará. De esta forma, una misma variable puede almacenar diferentes tipos de datos durante la ejecución del script.

SINTAXIS DEL LENGUAJE

- **No es necesario terminar cada sentencia con el carácter de punto y coma**

En la mayoría de lenguajes de programación, es obligatorio terminar cada sentencia con el carácter ; unque JavaScript no obliga a hacerlo, es recomendable hacerlo.

- **Se pueden incluir comentarios:**

Los comentarios se utilizan para añadir información en el código fuente del programa. JavaScript define dos tipos de comentarios: los de una sola línea y varias líneas.

COMENTARIOS

Existen 2 tipos de comentarios:

1. De una línea
2. De varias líneas (multilínea)

De una línea

```
<script>
  // Este un comentario de una única línea
  alert("Escribiendo comentarios en javascript!");
//Aquí puedo poner una nota de lo que hace esta línea
  // alert("Esto no se ejecuta");
</script>
```

COMENTARIOS

De varias líneas

```
<script>
    alert("Escribiendo comentarios multi-línea en javascript");
    /*
    alert("Esto no se ejecuta");
    alert("Esto no se ejecuta");
    alert("Esto no se ejecuta");
    alert("Esto no se ejecuta");
*/
</script>
```

FUNCIONES ÚTILES EN JAVASCRIPT

1. **Función alert**

   Esta función es un método del objeto Window y es muy utilizado en JavaScript, pues es la encargada de mostrar una pequeña ventana de aviso en la pantalla, así, si se requiere que aparezca un mensaje cuando ocurra determinada acción en el programa, podemos hacer uso de esta función. La función ***alert*** recibe como parámetro el mensaje que se debe mostrar en la ventana.

   Sintaxis:

   alert(“mensaje a mostrar”);

FUNCIONES ÚTILES EN JAVASCRIPT

2. **Función write**

   Esta función es un método del objeto **document** y lo que hace es escribir en la página web el texto que se ingresa como parámetro.

Sintaxis:

document.**write**(“texto”);

FUNCIONES ÚTILES EN JAVASCRIPT

3. **Función prompt**

Se utiliza cuando el usuario ingresa datos por medio del teclado. Con esta función aparece una ventana en la pantalla, con un espacio para el valor que se debe ingresar y un botón Aceptar para que la información sea guardada. Esta función recibe dos parámetros, el primero es el mensaje que se muestra en la ventana y el segundo es el valor inicial del área del campo texto.

Sintaxis:

letvariable = prompt(“mensaje”, “valor inicial”);

VARIABLES

Una variable es un elemento que se emplea para almacenar y hacer referencia a otro valor. Las variables también se conocen como identificadores y deben cumplir las siguientes normas:

- Sólo puede estar formado por letras, números y los símbolos $ (dólar) y _ (guión bajo).
- El primer carácter no puede ser un número.

Algunos ejemplos de nombres correctos:

```
let  nombre;
let  _nombre;
let  $nombre;
let  correoCliente;
```

TIPOS DE VARIABLES

Las variables son usadas para almacenar o asignar valores, todos los tipos de variables son declarados como el texto **let**

let numero= 200; *// tipo integer (Entero).*
let subtotal 4999.50; *// tipo float (Decimal / Flotante).*
let cadena= “Estamos en clases virtuales"; // tipo *string.*
let boolean = true; // tipo *boolean.*

TIPOS DE VARIABLES

- **boolean**, para los datos boleanos. (True o false)
- **number**, para los datos numéricos.
- **string**, para las cadenas de caracteres.
- **object**, para los objetos.
- **function**, para las funciones.
- **undefined**, para variables declaradas a las que no se les ha asignado valores.

Permite informar el tipo de una variable existente

```
console.log(typeof(nombre));

console.log(typeof(edad));
```

VARIABLES (INTEGER & FLOAT)

- Se utilizan para almacenar valores numéricos enteros (*integer*) o decimales (*float*).
- Los números decimales utilizan el carácter . (punto) en vez de , (coma) para separar la parte entera y la parte decimal.
- Para que una variable sea de tipo entero, se debe asignar un número sin ninguna comilla.

Por ejemplo:

```
let a = 2;
let b = 5;
```

VARIABLES (STRING)

- Se utilizan para almacenar caracteres, palabras y/o frases de texto. Para asignar el valor a la variable, se encierra el valor entre comillas dobles o simples, para delimitar su comienzo y su final.
- Un número dado entre comillas deberá ser considerado solo como un string y no como un entero.

Por ejemplo:

```
let nombre= “Juan”;
let apellido = “Rojas”;
```

VARIABLES (BOOLEAN)

- Las variables de tipo *boolean* o *booleano* también se conocen con el nombre de variables de tipo lógico.
- Una variable de tipo *boolean* almacena un tipo especial de valor que solamente puede tomar dos valores: true (verdadero) o false (falso).

Por ejemplo:

```
let usuarioRegistrado = false;
let aceptado = true;
```

DIFERENCIAS VARIABLES

- Durante la programación en Java Script se deben de utilizar 3 formas para declarar variables:

Var

Let

Const

¿Cuál es la diferencia que existe en las declaraciones var, let y const en JS?

const : Nos permite declarar variables inmutables, osea no se puede cambiar.

Var: pueden ser modificadas y re-declaradas dentro de su ámbito.

Let: pueden ser modificadas, pero no re-declaradas.

	var	let	const
Reasignación	✓	✓	✗
Redeclaración	✓	✗	✗
Propiedad del objeto global (window)	✓	✗	✗
Function Scope	✓	✓	✓
Block Scope	✗	✓	✓

DIFERENCIAS VARIABLES

VAR es la manera más antigua de JS. NO es muy estricta en cuanto al alcance, ya que al declarar variables de esta forma dichas variables podrán ser accedidas, e incluso modificadas, tanto dentro como fuera de los bloques internos en una función.

LET

Las variables declaradas NO pueden ser re-declaradas.
Las variables declaradas deben estar declaradas antes de ser usadas.
Las variables declaradas tienen un ámbito de bloque.

CONST al igual que LET se define en el contexto o alcance de un bloque, a diferencia de let y var, las variables definidas como constantes, ya no podrán ser modificadas ni declaradas nuevamente, en ninguna otra parte de la función o el contexto en el que ya existen.

La recomendación es reducir siempre al mínimo el alcance de nuestras variables, por lo que se debe usar let en lugar de var mientras sea posible.

OPERADORES

- Los operadores permiten manipular el valor de las variables, realizar operaciones matemáticas con sus valores y comparar diferentes variables. De esta forma, los operadores permiten a los programas realizar cálculos complejos y tomar decisiones lógicas en función de comparaciones y otros tipos de condiciones.

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
%	Residuo
++	Incremento
--	Decremento

Universidad CENFOTEC
SOMOS LO QUE SABEMOS

OPERADORES

1. Operadores aritmeticos
2. Operadores de cadena string
3. Operadores de asignación
4. Operadores relacionales
5. Operadores lógicos

Universidad CENFOTEC
SOMOS LO QUE SABEMOS

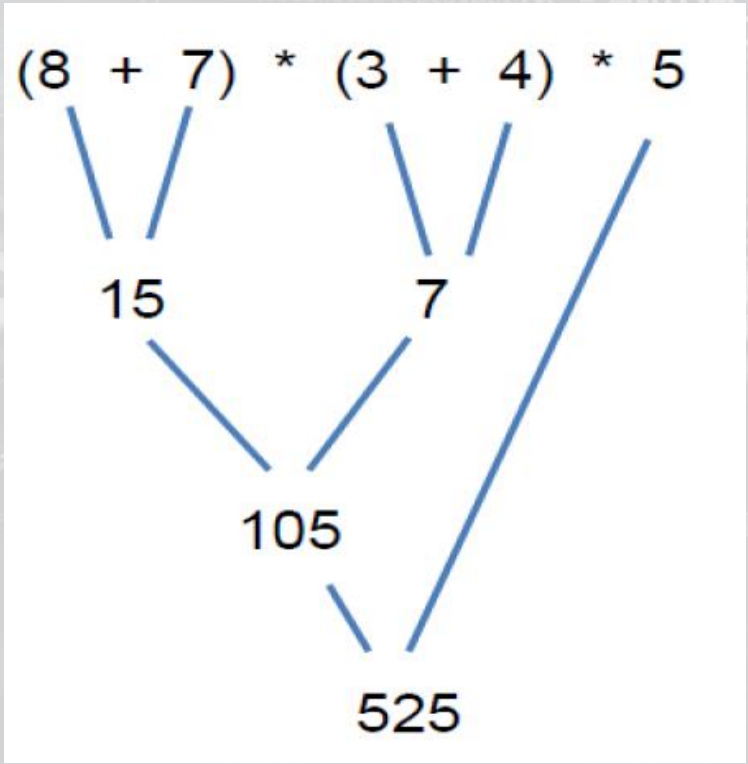
OPERADORES (MATEMÁTICOS)

- JavaScript permite realizar manipulaciones matemáticas sobre el valor de las variables numéricas. Los operadores definidos son: suma (+), resta (-), multiplicación (*) y división (/), residuo (%).

Operador	Nombre	Propósito	Ejemplo
+	Adición	Suma dos números juntos.	`6 + 9`
-	Resta	Resta el numero de la derecha del de la izquierda.	`20 - 15`
*	Multiplicación	Multiplica dos números juntos.	`3 * 7`
/	División	Divide el número de la izquierda por el de la derecha.	`10 / 5`
%	Sobrante (también llamado módulo)	Retorna el restante después de dividir el número de la izquierda en porciones enteras del de la derecha.	`8 % 3` (retorna 2, como tres está dos veces en 8, quedando 2 restantes.)

OPERADORES (JERARQUIA)

- La jerarquía de operadores inicia con las fórmulas que se encuentran entre los paréntesis para poder sacar los resultados.



OPERADORES (ASIGNACION)

- El operador de asignación es el más utilizado y el más sencillo. Este operador se utiliza para guardar un valor específico en una variable. El símbolo utilizado es = .

- A la izquierda del operador, siempre debe indicarse el nombre de una variable. A la derecha del operador, se pueden indicar variables, valores, condiciones lógicas, etc:

Por ejemplo:

```
let numero1 = 3;
let numero2 = 4;
```

Universidad CENFOTEC
SOMOS LO QUE SABEMOS

INCREMENTO Y DECREMENTO

- Estos dos operadores solamente son válidos para las variables numéricas y se utilizan para incrementar o decrementar en una unidad el valor de una variable.
- El operador de incremento se indica mediante el prefijo ++ en el nombre de la variable. El resultado es que el valor de esa variable se incrementa en una unidad.
- De forma equivalente, el operador decremento (indicado como un prefijo -- en el nombre de la variable) se utiliza para decrementar el valor de la variable.

Universidad CENFOTEC
SOMOS LO QUE SABEMOS

OPERADORES (RELACIONALES)

- Los operadores relacionales definidos por JavaScript son idénticos a los que definen las matemáticas: mayor que (>), menor que (<), mayor o igual (>=), menor o igual (<=), igual que (==) y distinto de (!=).

- Los operadores que relacionan variables son imprescindibles para realizar cualquier aplicación compleja. El resultado de todos estos operadores siempre es un valor booleano.

Universidad CENFOTEC
SOMOS LO QUE SABEMOS

OPERADORES (RELACIONALES)

Operador	Descripción
==	" Igual a" devuelve true si los operandos son iguales
===	Estrictamente "igual a" (JavaScript 1.3)
!=	" No igual a" devuelve true si los operandos no son iguales
!==	Estrictamente " No igual a" (JavaScript 1.3)
>	" Mayor que" devuelve true si el operador de la izquierda es mayor que el de la derecha.
>=	" Mayor o igual que " devuelve true si el operador de la izquierda es mayor o igual que el de la derecha.
<	" Menor que" devuelve true si el operador de la izquierda es menor que el de la derecha.
<=	"Menor o igual que" devuelve true si el operador de la izquierda es menor o igual que el de la derecha.

Universidad CENFOTEC
SOMOS LO QUE SABEMOS

OPERADORES LÓGICOS

Los operadores lógicos son importantes para realizar condicionales para toma de decisiones.

Operador	Descripción
&&	Lógico y (AND)
||	Lógico o(OR)
!	Lógico No (NOT)

OPERADORES (AND)

- La operación lógica AND obtiene su resultado combinando dos valores booleanos. El operador se indica mediante el símbolo && y su resultado solamente es true si los dos operandos son true:

variable1	variable2	variable1 && variable2
true	true	true
true	false	false
false	true	false
false	false	false

OPERADORES (OR)

- La operación lógica OR también combina dos valores booleanos. El operador se indica mediante el símbolo || y su resultado es true si alguno de los dos operadores es true:

variable1	variable2	variable1 || variable2
true	true	true
true	false	true
false	true	true
false	false	false

VARIABLES (ARRAY)

- También se les llama vectores, matrices e incluso *arreglos*. No obstante, el término array es el más utilizado y es una palabra comúnmente aceptada.
- Un array es una colección de variables, que pueden ser todas del mismo tipo o cada una de un tipo diferente.
- Una vez definido un array, es muy sencillo acceder a cada uno de sus elementos por su índice, las posiciones de los elementos empiezan a contarse en el hasta el numero de elementos menos 1.
- Los valores se almacenan en memoria.

Por ejemplo:

let dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];
Cual serial el valor de dias[3] ?