# Teoría de JavaScript

Se ejecuta en los navegadores WEB (Es un lenguaje interpretado)

Se ejecuta dentro de las páginas HTML, se debe escribir dentro de las etiquetas (<script>)

Se ejecuta del lado del cliente, es decir, no depende de ningún servidor

Páginas webs interactivas, y efectos especiales dinámicos.

Frameworks de JS: Angular, React, NodeJS(Backend). Typescript

JavaScript se maneja con OOP. Clases no existen en JavaScript, solo prototipos.

Es débilmente tipado

Editor de texto: Brackets, Atom

Rutas relativas: Representa solo una parte de la ruta (Cuando se deben entregar los documentos de un usuario a otro)

Rutas absolutas: Ruta completa del recurso (Cuando no se mueven los datos)

Buenas prácticas:

* Priorizar la legibilidad (Usar espaciadores o sangrías, usar “Scopes”)
* Colocar comentarios
* Utilizar convenciones (JS es “Case Sensitive”, por lo que es recomendado utilizar el mismo estilo de escritura o política). Notación camello
* Coloca Scripts en la parte inferior de la página (Depende de lo requerido, pero para la certificación, siempre debe hacerse). Es cuestión de velocidad del sitio
* Realizar pruebas
* No usar la vía corta (Usar Scopes en JavaScript)

Implementar JavaScript

Interna: Embebido o incrustado, en línea (dentro del codigo), o por medio de atributos de JavaScript (usando etiqueta Script). Se usa para extensiones de código muy pequeñas, o cuando se requiere que cierto código se aplique solo a una página.

“Onload” significa que se ejecutará al cargar **TODO** el documento

Externa: En un documento con extensión JS. Se utiliza cuando se requiere que ciertas líneas de código se apliquen a muchas páginas. Es para reutilizar código.

**Puede afectar la clasificación de la pagina en los motores de búsqueda si JS ocupa la mayor parte del contenido en la página.**

Dentro de comillas dobles, utilizar comillas sencillas, o viceversa.

Variable: Nombres asignados a celdas o espacios en memoria, con el fin de identificar valores.

Tipos de variables:

* “var”: Variable global, sin scope
* “let”: Variable a la cual puede ser accedida solo en el mismo scope
* “const”: Variable de solo lectura

Yo puedo colocar el código JS después de la etiqueta HTML también, sin embargo, no es recomendado, debido a los buscadores. Estos pueden quitar del índice de búsquedas nuestro sitio si detectan mala legibilidad de programación.

Operadores aritméticos:

* Asignación (=)
* Aritméticos (+, -, \*, /, %)
* Incremento, decremento (++, --)
* Asignación compuesta (+=, -=, \*=, /=)

En JS no se puede volver a declarar una variable con el mismo nombre

¿Que se mostrará después de la ejecución del siguiente código? (1 2)

var a = 1;

var b = 2;

var c = a + " " + b

alert(c);

¿Que se mostrará después de la ejecución del siguiente código? (3)

var a = 1;

var b = 2;

var c = a + b + "";

alert(c);

Las operaciones son de izquierda a derecha (Según orden de operador, ya sea aritmético, incremento, decremento, etc.)

Tipos de datos:

* Indefinido (“undefinied”): No hay valor asignado:
* Booleano: Verdadero o falso
* “Number”: Valor numérico, ya sea entero, decimal, etc
* “String”: Cadena de texto
* “BigInt”: Numero entero grande, se le pone una “n” al final
* “Null”: Representa un valor vacío intencionalmente

Los cuadros de dialogo detienen la ejecución de la página.

¿Podemos usar JavaScript en cualquier archivo HTML? (Cierto)

La sentencia “DEBUGER” permite invocar funcionalidades de depuración. Funciona igual que un “BreakPoint”. Si el navegador no tiene ninguna funcionalidad de depuración, la sentencia no tendrá ningún efecto. Esta sentencia no toma en cuenta la posición en el documento.

¿Todas las funcionalidades de JS funcionan de la misma manera en todos los navegadores? (Falso)

"use strict";

Esta sentencia permite forzar la sintaxis correcta en JS. Debe ir al inicio del documento o función (local Scope). Permite, por ejemplo, obligar la declaración de variables. No permite eliminar variables ni funciones, o duplicar un parámetro en una función. Números octales tampoco están permitidos, ni cadenas octales.

Manejo de excepciones: Se pueden utilizar múltiples “TRY”, seguidos de un “CATCH”, “FINALLY”, o ambos. “CATCH” si hay errores, si no, se ejecuta “FINALLY”. Este último siempre se ejecuta.

NaN: “Not a Number”, puede arrojarlo si se intenta multiplicar un numero con una cadena, o asignar un valor que no sea un número, a una variable que ya se le asigno un valor numérico.

Las excepciones en JS pueden manejar dos tipos de propiedades:

* .name
* .message

Las excepciones son errores que se hayan encontrado en el código. Detecta también errores debido al “use strict”.

En JavaScript. Los booleanos al hacer conversiones, la ausencia de dígitos, o caracteres, así como el cero, es falso, de lo contrario, siempre será verdadero.

BOM (Browser object model)

Se trata del objeto que permite controlar la ventana navegación usando JS

Con la interfaz del navegador:

* outerHeight: Ancho de la ventana
* outerWidth: Alto de la ventana

Sin la interfaz del navegador

* innerHeight
* innerWidth

El objeto “WINDOW” nos permite utilizar estas funciones. Este es un objeto global, por lo que no es necesario usar la palabra “WINDOW”.

Caracteres de escape:

* “\n”: Salto de línea

¿Los botones de la función “confirm” son? (Aceptar y Cancelar)

¿Si el usuario hace “click” en el botón cancelar en una función “prompt()”, el valor devuelto es? (null)

En JavaScript, no todos los objetos se encargan de almacenar valores, otros, se encargan de brindar métodos y propiedades, como el objeto Math (No incluye constructor).

Funciones:

Bloques de código identificados con un nombre. Estas instrucciones se ejecutan una vez que la función es llamada.

**Las funciones que están fuera de una función se consideran en ámbito global.**

**No se recomienda crear variables globales en una función, lo mas adecuado es enviarle parámetros al ser llamadas.**

**Funciones anónimas:** Una función anónima es simplemente asignar el nombre de una función a una variable:

var nombreFuncion = function (){

    alert("Hola mundo");

}

En JS, las variables, de cierta manera, son objetos, por lo que es posible asignarles métodos, así como parámetros.

Un operador de comparación devuelve un valor, ya sea verdadero, o falso.

Texto

Descripción generada automáticamente

Operador ternario:



Operadores de comparación estrictas (JavaScript no hace conversiones de tipo cuando se usan estos operadores)

* “===”
* “!==”

Operadores lógicos

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Declaraciones Switch:

Toman solo una expresión, y luego, compara y selecciona la que coincida con la entrada.

La sentencia de descanso (**break**), termina o saca del bloque de código donde se insertó, por eso es importante colocarlo al final de una opción en la estructura “Switch - Case”, para que no ejecute lo restante (Esta sentencia transfiere el control a la siguiente declaración). Caso contrario, es la sentencia **continue**.

La diferencia entre el ciclo “do”, y el ciclo “while”, es que el ciclo do, se ejecutara al menos una vez.

## Document Object Model (DOM)

Generalmente, la mayoría de los navegadores tienen directrices DOM, por lo que no es necesario importar nada para interactuar con él, utilizando JS.

No es necesario escribir la palabra “window”, antes de utilizar un objeto del navegador, ya que este es un objeto global.

La mayoría de los objetos del DOM, son del tipo “element”.

No se recomienda el uso de atributos de eventos en HTML, porque es contrario al propósito principal de HTML5, es decir, proveer de una tarea especifica a cada lenguaje involucrado (Recordemos, HTML solo es la estructura, no la funcionalidad). Por eso existen los oyentes.

**Los oyentes de eventos pertenecen al objeto WINDOW**

Objeto evento: Sirve para acceder al elemento que desencadenó un evento dentro de la función oyente.