**¿Qué es JavaScript?**

*Clase 1*

¿Qué es JavaScript?  
Nace para dar dinamismo a la web para hacer las paginas estáticas mas dinámicas orientado a trabajar 100% web

Es un lenguaje interpretado, orientado a objetos, débilmente tirado y dinámico.

Orientado a objetos: permite generar objetos globales que son conjuntos de características para agrupar distintos elementos de nuestro sistema que a la vez forman parte de una base de datos y todos estos elementos pueden ser ocupados para mostrar información individual.

Débilmente tipado: los tipos de datos no están bien definidos en el leguaje y permite por ejemplo operaciones entre numerosos y letras porque el lenguaje asume tipos de datos que no necesariamente son los que se quiere representar.

++Dinámico: ++ el lenguaje corre directamente en el runtime(tiempo de ejecución).

Realmente es un lenguaje interpretado?  
Interpretado quiere decir que el leguaje va a ir leyendo el código linea por linea en el tiempo de ejecución y probando que todo funcione en el proceso con la interacción con el programa y si encuentra un error en ese momento lo marca.

Una vez que el navegador obtiene el código js el motor de js lo procesa convirtiéndolo a binario para que sea entendido por la maquina por lo que javascript es un leguaje compilado.

Backwards & Forwards

Forwards: Ser compatible con versiones futuras quiere decir que al implementar funcionalidades nuevas del leguaje no daña nuestro código y se puede usan en versiones anteriores del motor de compilación.

Backwards: Todas las funciones nuevas del lenguaje no rompen el entorno de trabajo actual pero no las puedes usar inmediatamente hasta que no se estandaricen .

Para hacer este cross versión del lenguaje se utilizan compiladores como BABEL que permite utilizar nuevas funcionalidades del lenguaje javascript dentro de nuestro entorno actual de trabajo traduciendo las nuevas versiones de javascript a versiones anteriores standard que el navegador entiende para que puedan ser utilizadas.

**Porqué JavaScript?**

3 Lenguajes estándares: **HTML5 - CSS3 - JS**

**HTML:** Para maquetar la información. Texto, imagenes, videos, etc.  
**CSS:** Para darle estilo a la información.  
**JS:** Programación para que la página sea dinámica o generar una app web.

En el 2019 la W3C decidió y subió como nuevo estandar y lenguaje **WA (WebAssembly)**. Es un lenguaje nuevo a utilizar para construir productos web. Con este no será necesario utilizar HTML, CSS y JS para hacer productos web.

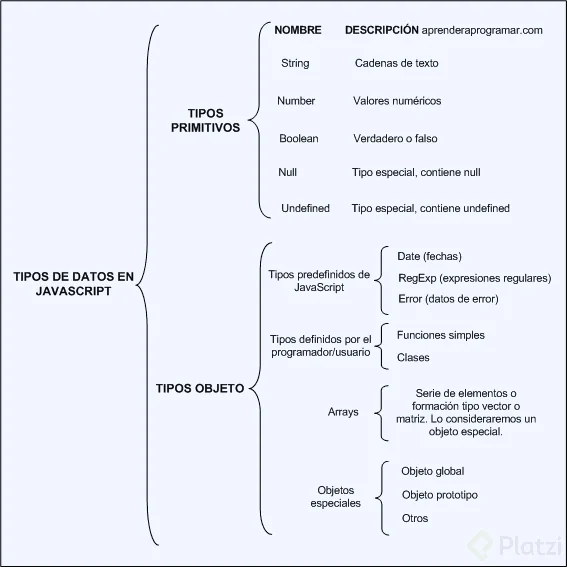
**JS** tiene una comunidad enorme de devs que ayudan a generar diferentes cosas. Para apps web hay muchos frameworks y librerías construidas en **JS**, que ayudan a desarrollar proyectos de una manera mucho más rápida, eficiente y robusta.  
**Angular, Vue, React** son algunos de los frameworks que podemos utilizar para hacer productos web.

Si no sólo quieres hacer productos web podemos utilizar un framework llamado **React Native**, para poder construir aplicaciones nativas para Android y IOS.  
**Electron:** framework JS que nos permite desarrollar aplicaciones para escritorio, tanto para Mac como para Windows.

En toda web y app tenemos 2 partes. El Front-end y el Back-end.

**Front-end:** Es todo lo que se ve en nuestra web/app y con lo que podemos interactuar.  
**Back-end:** Va manejando las bases de datos, las interacciones y peticiones que el Front-end le va a pedir.

**Node.JS** es un entorno de ejecución de JS que corre en el Back-end. Permite trabajar aplicaciones IOT (Internet de las cosas), hace inteligente ciertos dispositivos conectados a internet.



**VARIABLES**

Es la representación de algún lugar en memoria que vamos a reservar para poder guardar un valor, el cual puede tener diferentes tipos.

Javascript tiene una palabra reservada que utiliza para representar un valor. Ejem:

var nombre = “Oscar”;

El var le indica al programa que debe guardar un espacio en la memoria, el cual estará definido según el valor que le coloquemos, en el caso del ejemplo será “Oscar” (valor String).

**FUNCIONES**  
Son un conjunto de sentencias que nosotros podemos utilizar para generar ciertas acciones con los valores que ya guardamos antes en las variables.  
Tenemos dos tipos de funciones  
• Declarativas: va a inicializar un valor y lo guardamos en memoria.  
• Expresión (también conocidas como funciones anónimas): vamos a crear una variable donde guardamos la función en memoria.  
Las funciones también necesitan parámetros que estoy esperando recibir como valor para que la función pueda hacer algo.  
• Parámetros de la función: nos ayudan a generar cálculos o resultados de la función.

**Diferencias:**

A las funciones declarativas se les aplica hoisting, y a la expresión de función, no. Ya que el hoisting solo se aplica en las palabras reservadas var y function.

Lo que quiere decir que con las funciones declarativas, podemos mandar llamar la función antes de que ésta sea declarada, y con la expresión de función, no, tendríamos que declararla primero, y después mandarla llamar.

**Scope**

Es el alcance que tienen las variables en el código.

Existen 2 tipos de scopes.

* Global: Puede ser llamada a lo largo de nuestro programa.
* Local: Solo puede ser llamada dentro del bloque de código en el que se declaró.

**Hoisting**

El Hoisting es un proceso del compilador de JavaScript, que consiste en que la declaracion de las variables y las funciones son llevadas al inicio del codigo, sin importar su posicion, para su procesamiento, sin embargo, la inicializacion de las variables no es llevada al inicio del codigo para su compilacion, sino solo su declaracion, lo cual suele dar espacio a errores cuando se declara una variable sin inicializarla y se procesa en el codigo antes de haber llegado a su inicializacion, lo cual nos suele dar una variable con valor undefined, ya que la variable sí fue almacenada en memoria, pero no se le asigno un valor hasta despues de su ejecución.

Aqui un ejemplo de esto:

saludo();

function saludo() {

console.log("Hola " + nombre);

}

var nombre = "Aaron";

El output de este codigo seria el siguiente:

Hola undefined

**Coercion**

Coerción es la forma en la que podemos cambiar un tipo de valor a otro, existen dos tipos de coerción:  
Coerción implícita = es cuando el lenguaje nos ayuda a cambiar el tipo de valor.  
Coerción explicita = es cuando obligamos a que cambie el tipo de valor.

//Ejemplos de Coerción:

var a = 4 + "7"; //Convierte a 4 en un string y lo concatena con el "7", por esto regresa un string de valor "47"

4 \* "7"; //Convierte al "7" en un número y realiza la operación, por esto devuelve 28 var a = 20;

var b = a + ""; //Aquí concatenamos para convertir la variable a string (coerción implícita) console.log(b);

var c = String(a); //Aquí obligamos a la variable a convertirse en string (coerción explícita) console.log(c);

var d = Number(c); //Aquí obligamos a la variable a convertirse en número (coerción explícita) console.log(d);

**Valores: Truthy y Falsy**

*¿Que tipos por default son verdaderos y falsos?*

Usamos la función de JS que es *Boolean()* dentro del paréntesis ponemos el valor y nos dice si el mismo el False o True.

–> *Falsy*

* Boolean() —> sin ningun valor es false
* Boolean(0) —> false
* Boolean(null) —> false
* Boolean(NaN) —> false // NaN es Not and Number
* Boolean(Undefined) —> false
* Boolean(false) —> false
* Boolean("") —> false

–> *Truthy*

* Boolean(1) —> true //cualquier numero que no sea igual a cero es true
* Boolean(“a”) —> true
* Boolean(" ") —> true // siendo un espacio el valor es true
* Boolean([]) —> true // un array nos da un true
* Boolean({}) —> true // un objeto nos da el valor de true
* Boolean(function() {}) —> true //una funcion tambien es true
* Boolean(true) —> true

Todo esto lo vamos a usar en condiciones esto valida si es verdadero o falso para ejecutar cierta acción.

**Operadores de asignacion**

| **Simbolo** | **Descripcion** |
| --- | --- |
| **=** | **operador de asignacion** |

**Operadores de comparacion**

| **Simbolo** | **Descripcion** |
| --- | --- |
| **==** | **igual que** |
| **===** | **estrictamente igual que** |
| **> or >= or >==** | **mayor o mayor igual que** |
| **< or <= or <==** | **menor o menor igual que** |
| **!= or !==** | **diferente que** |

**Operadores aritmeticos**

| **Simbolo** | **Descripcion** |
| --- | --- |
| **+** | **operador suma este se utiliza para concatener dos cadenas de texto.** |
| **-** | **operador resta** |
| **\*** | **operador de multicacion** |
| **/** | **operador de division** |
| **%** | **operador de modulo** |
| **\*\*** | **operador de potenciacion** |

**tambien se les conoce como operadores binarios. por que toman dos valores y generan un resultado.**

**Operadores logicos**

| **Simbolo** | **Descripcion** |
| --- | --- |
| **!** | **NOT niega un valor** |
| **&&** | **AND** |
| **||** | **OR** |