**¿CÓMO NACE JAVASCRIPT?**

Nace con la necesidad de generar dinamismo en las páginas web y que a su vez los usuarios y las empresas pudieran interactuar unos con otros.

**¿QUÉ ES JAVASCRIPT?**

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

* **DÉBILMENTE TIPADO**

Se pueden hacer operaciones entre tipos distintos de datos (enteros con strings, booleanos con enteros, etc). Ejemplo:

4 + "7"; // 47

4 \* "7"; // 28

2 + true; // 3

false - 3; // -3

* **DINÁMICO**

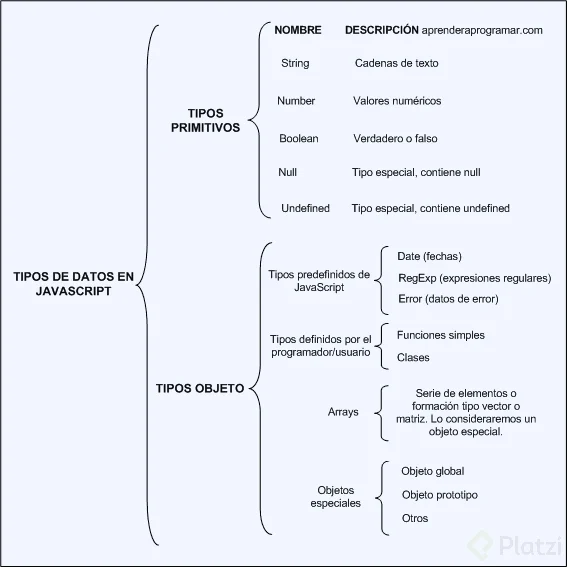
Corre directamente en la etapa de Runetime sin una etapa de compilación previa. Esto permite probar nuestro código inmediatamente; pero también es lo que hace que los errores se muestren hasta que se ejecuta el programa.

**¿REALMENTE ES JAVASCRIPT UN LENGUAJE INTERPRETADO?**

Si, y la razón es que le navegador lee linea por linea nuestro código el cuál le indica lo que tiene que hacer, sin la necesidad de compilar. Todo esto es controlado por el motor de Javascript V8 del navegador

**JAVASCRIPT ES BASCKWARDS COMPATIBLE**

Todas las funciones nuevas que salen de Javascript no dañarán el trabajo ya hecho, pero no se podrá utilizar en nuestro entorno de trabajo inmediatamente. Para solucionar esto está Babel que permite utilizar las nuevas características del lenguaje, pero lo transforma a una versión que el navegador pueda entender.

1. Si solo estuvieras interesado en trabajar aplicaciones web tienes muchos frameworks y librerías construidas en JavaScript que te van a ayudar a hacer proyectos de forma mucho mas rápida, eficiente y robusta (Angular, View, React,entre otros)
2. Si no quieres trabajar solo en aplicaciones Web puedes utilizar JavaScript con un framework que se llama React Native para poder construir aplicaciones nativas como Android y IOS.
3. Puedes construir aplicaciones de escritorio con JavaScript, usando un framework llamado Electron, pueden correr en Mac o Windows.
4. También puedes trabajar en la parte del Back-end o \*\*IOT \*\*(Internet Od Things) es un concepto que se refiere a una interconexion digital de objetos cotidianos con Internet. Esto con un Framework llamado NodeJS, el cual es un entorno de ejecución de JavaScript que corre directamente en el Back-end.

* **FUNCIONES**

Las funciones son las tareas que va a llevar a cabo el navegador. Existen 2 tipos de funciones

1) Declarativas

2) De expresión

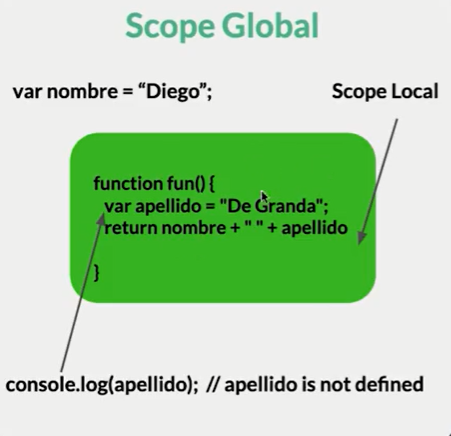
Ambas pueden llevar parámetros, que son los datos que necesitan para ejecutarse.

**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**

****

* **¿Qué es Hoisting?**

En JavaScript, las declaraciones (por ejemplo, de variables o funciones) se mueven al principio de su scope o ámbito. Este comportamiento se conoce como hoisting y es muy importante tenerlo en cuenta a la hora de programar para prevenir posibles errores.

Las funciones siempre se mueven arriba del scope. Por lo tanto, podemos elegir donde declararlas y usarlas.

La declaración de las variables se mueven arriba del scope, pero no la asignación. Antes de usar una variable, habrá que crearla y asignarla.

* **COERCIÓN**

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.

* **OPERADOR DE ASIGNACION**

= operador de asignacion

* **OPERADORES DE COMPARACION**

== igual que

=== estrictamente igual que

> or >= or >== mayor o mayor igual que

< or <= or <== menor o menor igual que

!= or !== diferente que

* **OPERADORES ARITMETICOS**

+ 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

* **OPERADORES LOGICOS**

! NOT niega un valor

&& AND

|| OR

* **ARRAY**

Un Array es un tipo de estructura de datos, objeto. Puede guardar distintos datos (del mismo tipo) dentro, guarda los datos en forma de lista.

**.lenght** devuelve la longitud del array.

**.push()** agrega elementos al final de array.

**.pop()** elimina el ultimo elemento del array.

**.unshift()** agrega un elemento al array, pero lo agrega en primer lugar.

**\*\*.shift() \*\***elimina el elemento que está en el inicio del array.

**\*\*.indexOf \*\*** devuelve la posición de un elemento del array.

* **LOOP**

Los bucles pueden ejecutar un bloque de código varias veces. JavaScript admite diferentes tipos de bucles:

* **for** - recorre un bloque de código varias veces
* **for/in** - recorre las propiedades de un objeto
* **for/of** - recorre los valores de un objeto iterable
* **while** - recorre un bloque de código mientras se cumple una condición específica
* **do/while** - también recorre un bloque de código mientras se cumple una condición específica
* **¿QUIÉN ES THIS?**

Es una variable que hace referencia al objeto. En este caso: this = miAuto.

**.FILTER :** nos permite filtrar solo los elementos que deseamos (según ciertos criterios) y devolverlos en un nuevo array.

**.MAP :** crea un nuevo array con los resultados de la llamada a la función indicada aplicados a cada uno de sus elementos.

**FOREACH()** : Ejecuta lo que le definamos una vez por cada elemento de nuestro array

**SOME()** : Comprueba si al menos un elemento del array cumple con la condición que le damos

El método **find ()** devuelve el primer valor que coincide de la colección. Una vez que coincida con el valor en los resultados, no verificará los valores restantes en la colección de matriz.

El método **filter ()** devuelve los valores coincidentes en una matriz de la colección. Verificará todos los valores de la colección y devolverá los valores coincidentes en una matriz.