Java Script

**Clase 1 (info básica)**

Una vez vinculado con Java el html, crear en JAVA un “alert (“mensaje enviado al browser”);

De esa manera sabremos si el vinculo con el html quedó bien hecho.´

Una manera mas prolija de hacerlo es con console.log(“mensaje enviado a la consola del browser”);

De esa manera el mensaje aparecerá solamente en la consola del inspector de Google Chrome.

Es recomendable dejar en el código todas las variables juntas y todo el calculo abajo.

**Resumen**

. En JS podemos agregar comentarios de una sóla línea con la combinación “//”, o comentarios de *bloque*, de varias líneas, envolviendo el comentario con los signos “/\* \*/”:

Ejemplo:

*// comentario de una línea*

*/\*   comentario de  
     bloque de varias líneas \*/*

*Palabras reservadas de Java:*

Algunos ejemplos de estos términos son: break, case, catch, continue, default, let, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, while, with, etcétera. En el curso aprenderemos a utilizar los más importantes para construir aplicaciones.

Variables:

Una variable es un espacio reservado en la memoria dónde almacenamos algún tipo de dato, y que puede variar con el tiempo y transcurso de nuestra aplicación

Hay dos procesos en la declaración de variables:

1. Declaración: Es la creación de la variable. Se escribe let o const seguido del nombre de la variable. Ejemplo “let colores” (sin las comillas)
2. Asignación

Dos tipos de Variables:

let: Puede recibir multiples asignaciones en el trascurso de la aplicación. (Puede modificarse a lo largo del código, como por ejemplo una variable “total”)

const: Puede recibir una única asignación al momento de su declaración. (No se puede modificar a lo largo del código, como por ejemplo una variable de precio fijo)

**Ejemplo de uso:**

Let total = 0

Const torta = 1200

Total = total + torta --> (0 = 0 + 1200)

Total = total + torta --> (1200 = 1200 + 1200)

Console.log (total) --> 2400

**Tipos de valores:**

* Number: Es un valor numerico (“integer” si es un valor entero “float” si es un valor decimal)
* String: Compuesto por uno o mas caracteres y se define entre comillas.

Ambos tipos de valores se pueden convinar.

Para asignar la variable se hace por medio del = (igual).

let nombre = "Gabriel";

Entonces se escribiría asi: let nombre = "Gabriel"

**Operaciones con Variables numericas:**

Son operaciones comunes matemáticas, que pueden ser suma; resta; multiplicación; división.

Ejemplo:

let numeroa = 8;

let numerob = 2;

let numeroc = 1;

let suma = numeroa + numerob;

let resta = numeroa - numerob;

let division = numeroa / numerob;

let multiplicasion = numeroa \* numerob;

let calculo = ((numeroa + numerob) / numerob) + numeroc;

**Operaciones con Variables string:**

Estas variables pueden concatenarse y crear nuevas cadenas.

Por ejemplo:

let modelo ="Fiat UNO Fire "

let año = "2006 "

let cilindrada = "1.3"

let autodegabriel = modelo + año + cilindrada;

***IMPORTANTE:*** *Los espacios se ponen agregándole un espacio al finalizar el valor de la variable.*

Prompt, Consola y Alert

**Prompt():** Con ella se puede solicitar un dato al usuario para asignar el valor ingresado a una variable. Todo lo que ingrese por Prompt se lo conoce como “entradas”.

Ejemplo de uso:

let nombreingresado = prompt("Ingrese su nombre");

**Console():** Por medio de console.log() podremos mostrar el valor de una variable en la consola haciendolo de esta forma:

let segundonombre = "Alejandro"

console.log(segundonombre)

O mostrar un mensaje en particular escribiendo entre paréntesis de esta manera:

console.log("Mensaje de prueba")

**Alert():** Es conocido como “salida”, ya que salen de la aplicación como resultado de un procesamiento, siendo visibles por el usuario.

Por ejemplo:

**Algoritmo**

Es el conjunto de instrucciones ordenadas que programamos para realizar una tarea.

Basicamente es darle el paso a paso de como hacer algo a la computadora.

Ejemplo:

let entrada = prompt("Ingresa tu nombre")

let salida = "Hola" + " " + entrada + " " + "bienvenid@ a nuestra pagina!!"

alert(salida);

Definimos una variable y la asignamos como prompt, luego definimos otra variable como mensaje y por ultimo definimos un alert que use esa variable como salida.

**Clase 2 ( CONTROL DE FLUJO)**

**Condicionales:**

Vamos a encontrar los valores booleanos: Estos son TRUE or FALSE.

Para el armado de un condicional nos encontramos con las siguientes estructuras:

* **Estructura if:**  Se utiliza para ejecutar un bloque si la condición es verdadera.   
    
  Basicamente lo que se hace es  
    
  if(se pone lo que vamos a comparar){se pone lo que se va a ejecutar si la comparación es verdadera}
* **Estructura if…else:** Se utiliza para sumar un bloque de código que se ejecutara únicamente si la primer comparación (el if) es falso.   
  Siempre se escribe luego de cerrar el primer bloque con ;}  
    
  Basicamente lo que se hace es  
    
  if(se pone lo que vamos a comparar){se pone lo que se va a ejecutar si la comparación es verdadera}  
  else{se pone lo que se va a ejecutar cuando el if sea falso}
* **Estructura if…else if:** Se utiliza cuando se quiere comparar mas de un valor.   
    
  if(se pone lo que vamos a comparar){se pone lo que se va a ejecutar si la comparación es verdadera}  
  else if{se pone otra comparacion}

**Variable booleana:**

Este tipo de variable nos va a permitir utilizar una variable que sea igual a una función y luego utilizar el resultado de esa función en conjunto a las estructuras if; else; else if; antes vistas.

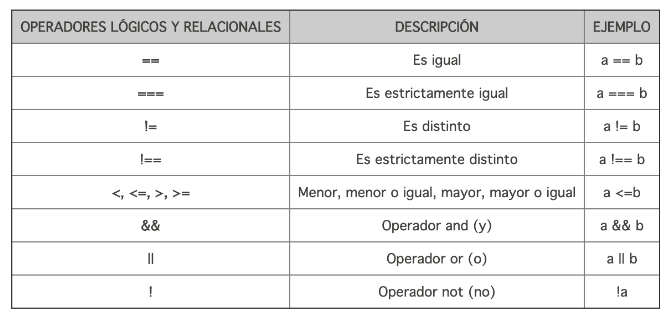
Ejemplo de uso:

Let booleano = true

if(booleano){se pone lo que se va a ejecutar si la comparación es verdadera ósea “true”}  
else{se pone otra comparacion}

**OPERADORES LÓGICOS Y RELACIONALES**

Son elementos que nos permiten evaluar valores y/o variables. Se utilizan en el paréntesis del if (). Y los valores a comparar se declaran dentro de un paréntesis.



**Operadores de igualdad: (==) (===):**

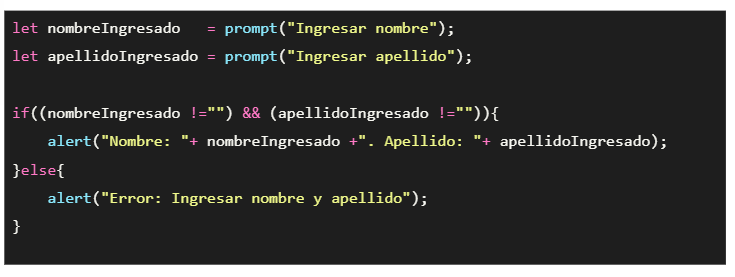
Sirve para preguntarnos si dos valores son iguales.  
  
(==): compara dos valores sin importar el tipo de dato, en este caso nos referimos que no importa si el valor ingresado es un numero o un string, si ambos son iguales, entonces la comparación será verdadera.   
  
(===): Compara valores string con string y valores numéricos con valores numéricos.

**Operadores de desigualdad: (!=) (!==):**Sirve para saber si dos variables o valores son iguales. Al igual que el caso anterior, con un igual compara sin importar el tipo de dato, con dos iguales compara estrictamente.

**Operadores lógicos (&&); ||; (!):**

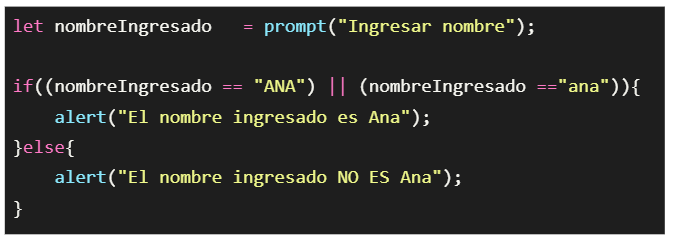
1. **&& (and):** Sirve para comparar dos valores booleanos. Es verdadera solamente si ambos valores son verdaderos.
2. **|| (or):** Sirve para comparar dos valores booleanos, la comparación es falsa si ambos valores son falsos.
3. **! (not):** Sirve para transformar e invertir el valor de verdadero o falso, ejemplo !true (es falso); !false (es verdadero)

**CONDICIONES COMPUESTAS CON && (AND)**



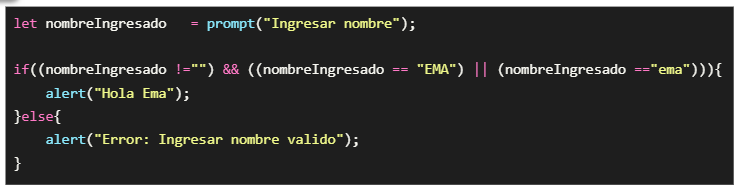
En este caso se esta comparando que ambas variables tengan algún valor cargado, ya que si alguna de ellas esta vacia, el navegador tirara el alert.

**CONDICIONES COMPUESTAS CON || (OR)**



Aca lo que se haces es decir, si el valor ingresado es alguno de estos dos, entonces, la variable es verdadera, sino, es falsa y tenes que mostrarme el mensaje de alerta.

**COMBINACIONES DE AND Y OR**

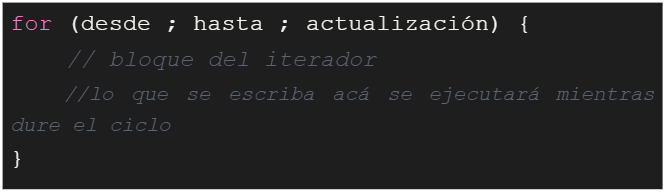
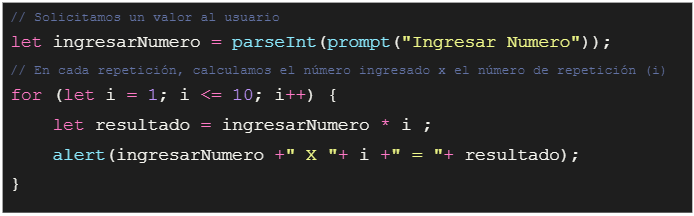
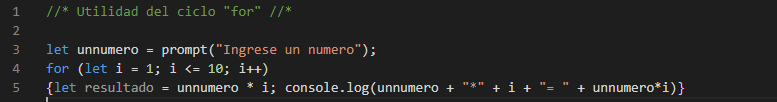
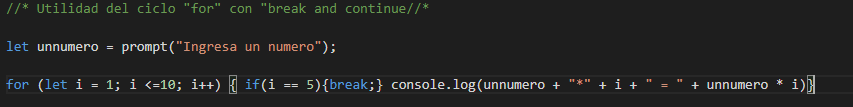
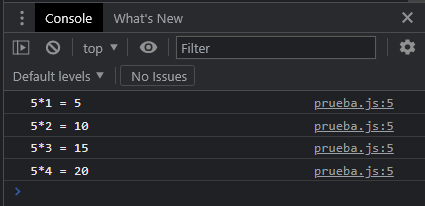
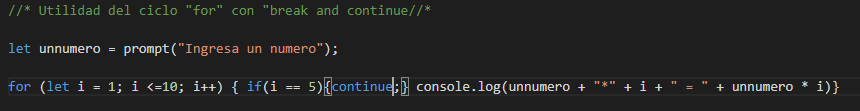
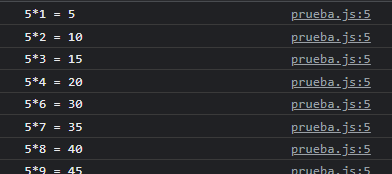
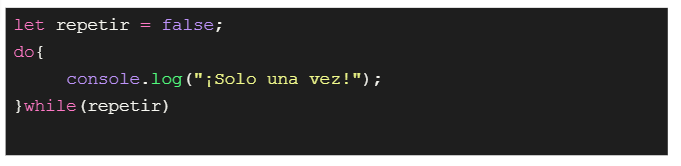


Lo que hacemos aca es primero asegurarnos que se haya ingresado un valor y que no hayamos dejado un campo vacio. El paso siguiente es fijarse si el valor ingresado es alguno de los dos admitidos como verdaderos, en caso contrario mostrara una aleta.

**Clase 3 ( CICLOS / ITERACIONES)**

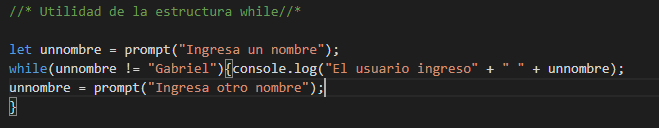
**CICLO:** Son estructuras que nos permiten repetir un conjunto de instrucciones (bloques) una cantidad determinada de veces.

Tipos de ciclos

* **Ciclos x conteo:** La estructura es **for**, la cual garantiza un numero de repeticiones hasta que una variable numérica tenga un valor especificoy se escribe de la siguiente manera  
    
    
    
    
  (la “i” significa índice); ++ significa sumar de a uno.  
    
    
    
    
  Estructura del for --> explicación  
    
  for(se declara una variable para ser comparada ; se hace la comparación ; es la operación que se le va a hacer a la variable definida en el primer “;” mientras que la comparación no se cumpla)  
    
    
  **Utilidad del Break y Continue**  
    
  Estas sentencias lo que nos permiten es cortar una iteración o saltear una iteración.  
    
    
  Ejemplo Break:  
    
    
    
    
  Ejemplo Continue  
    
    
  
* **Ciclos condicionales:** Implican comparaciones de todo tipo. De la misma forma que construimos los condicionales ***if***, las estructuras de este tipo interpretarán el bloque si la comparación es verdadera, sólo que a diferencia de ***if***, las instrucciones se repiten hasta que el condicional se evalúa como falso. Las estructuras ***while*** y ***do...while*** son de este tipo.  
    
    
  

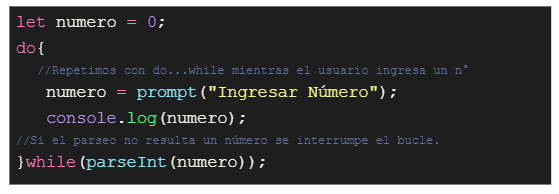
**While:**

Cuando existe la necesidad de utilizar un bucle, pero su repetición está condicionada por el valor de una o más variables no numéricas, la estructura ***while*** es más apropiada.  
El bucle continuara hasta que se ingrese un valor verdadero.

  
  
Es de suma importancia que luego de indicar que hacer si el valor es falso, indicar una acción siguiente, ya que sino se generara un bucle infinito.

Si el valor que se ingresa primero ya es correcto, entonces no se generará ninguna repetición.

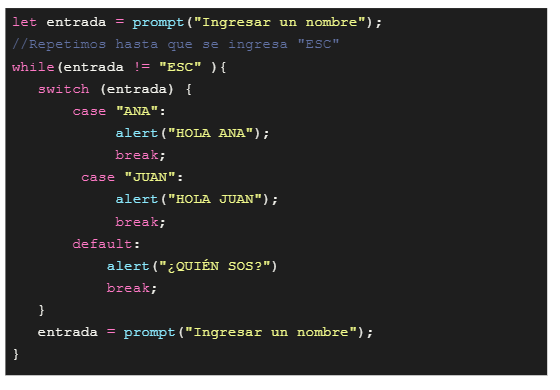
**Do while:** A diferencia del While, este permite evaluar el condicional al final de la iteración, por lo que garantiza al menos una iteración en caso de ser el valor falso.  
Un caso de ejemplo de uso es para el logeo de un usuario.



Parseint(string, base), Lo que hace esta función es convertir (parsear) un argumento de tipo cadena y devuelve un entero de la base especificada.  
Si encuentra un carácter que no es un numeral de la base especificada, lo ignora a el y a todos los caracteres correctos siguientes.

**Estructura Swhitch:** Lo que permite esta estructura es poder evaluar valores específicos y ejecutar un bloque de código puntual.   
Por medio de Shitch (variable) {case (escribimos el valor); break (para saltar a la siguiente comprobación; default (En caso que no sea ninguno de los case que hacer).

Ejemplo:

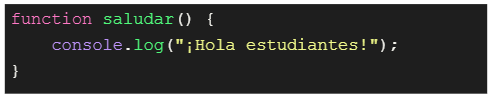


En conclusión es importante distinguir que en programación una cosa es la resolución del problema y otra es la resolución del problema de la manera mas apropiada.

Hay diferentes formas de afrontar el mismo problema. Requiere practica saber como combinar todo lo visto.

**Clase 4 ( PROGRAMACION CON FUNCIONES)**

Que son las funciones:   
  
Las funciones son elementos que creamos con la intención de agrupar instrucciones a ser utilizadas una o más veces, a lo largo del script. En programación, el uso de funciones permite separar un problema en partes más pequeñas, independientes y reutilizables.

Como declarar una función:  
  


Estructura básica de una función:

Function “nombre de la función” (Aca se escriben los parámetros (q son valores que usaremos en la funcion)) {Es lo que va a hacer la funcion}

Como llamar una función:

Vamos a codear el nombre de la función y abrimos paréntesis. Pepito ()

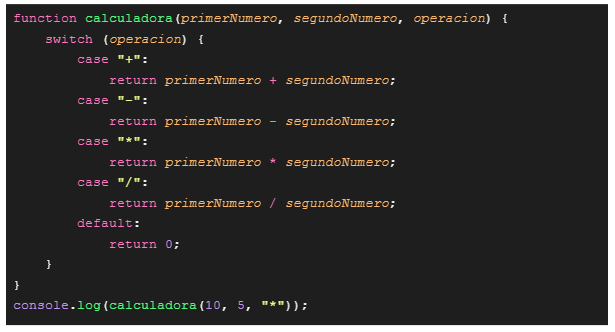
Parametros de la función:

Un parámetro es lo mismo que declarar una variable. Solo que es una variable que podremos usar en esa función.

Entonces por ejemplo, yo puedo dar dos parámetros y en el bloque de código llamarlos y asignarles una acción, igual que como haría con una variable.

**USO DEL VALOR “RETURN”**

Funciona como un total o mejor dicho muestra el resultado de la función asiganda. Ejemplo de uso:



**AMBITOS O SCOPE DE LAS VARIABLES:**

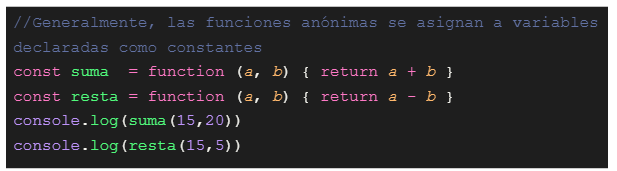
**Variables Globales:** Son las definidas por fuera de una función. Puedo utilizarlas en cualquier momento del código. (Existe en todo el código)

**Variables Locales:** Son las definidas dentro de una función y solamente voy a poder utilizarlas dentro de esa función. (Existe solo en la función)

**FUNCIONES ANONIMAS**

Se definen sin nombre y se usan para ser pasadas por parámetro o asignarla a una variable.

Ejemplo:

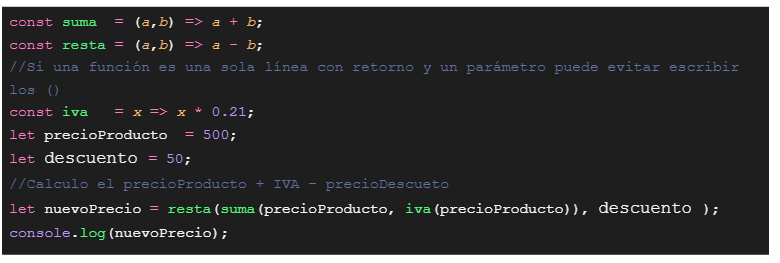
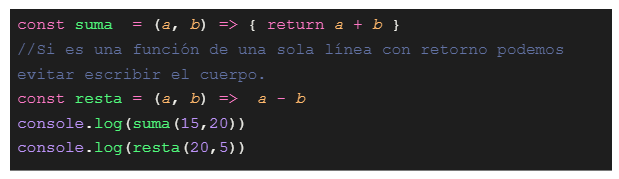


**FUNCIONES FLECHAS**

Es una forma mas resumida de declarar una función anónima.

Por ejemplo, si una función tiene solo un parámetro, puedo declararla sin uso del ()

Si una función es de una sola línea y solo tiene un retorno puedo codearla sin el uso de {}

Ejemplo: 

**Clase 5 ( OBJETOS)**

**OBJETOS:**  Son estructuras que podemos definir para agrupar valores bajo un mismo criterio, y asignarles comportamiento.   
Entendemos por “comportamiento” a funciones asociadas al objeto, las cuales podemos emplear para transformar los valores y estructura del mismo.

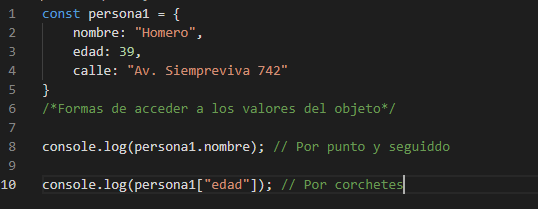
OBJETO

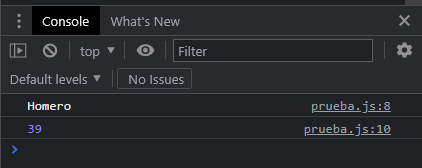
Funciones  
Variables  
Strings/Number

**Definicion de un objeto literal:** Define las propiedades y valores del objeto de forma explícita, asignando dicha estructura a una variable.

**Como lo definimos**

* Primero definimos una variable CONST con su nombre y la = {}
* Ahora abrir {}
* Ahora definimos dentro de las llaves las propiedades o identificadores y lo hacemos de la siguiente forma  
  Propiedad: “Nombre de la propiedad” ó   
  Propiedad: valor de la propiedad.   
  A una propiedad se le puede asignar cualquier tipo de valor (string, number, boolean, array, etcétera).

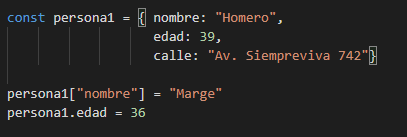


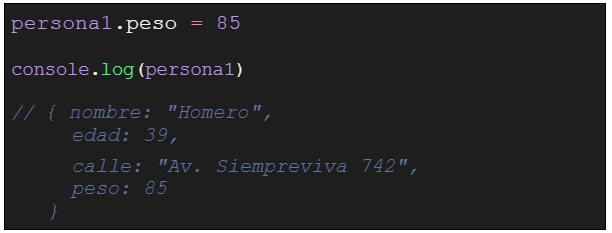


Lo bueno que ofrecen los corchetes es que permite que el string de referencia provenga de una variable.

**ASIGNACION DE VALORES:**

Se hace de la misma forma que para acceder a las propiedades. Es más podemos crear una nueva propiedad si la llamamos y la misma no existe.





**CONSTRUCTORES:**

Un constructor es una función que permite crear un objeto.

El constructor se define siguiendo las reglas de una función, pero teniendo presente que la declaración de propiedades implica usar un prefijo para asociar los valores al objeto correctamente.

El identificador de la función comienza en mayúscula “Persona”, puede considerarse una convención cuando definimos constructores. Por otro lado utilizamos la palabra reservada “this.”, la cual nos permite determinar las propiedades que conforman al objeto.

**Como definimos un constructor:**

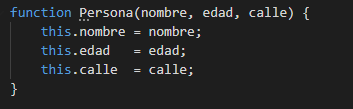
* El nombre del constructor debe comenzar en mayúscula
* Determinamos las propiedades del objeto por medio de la palabra “this”

Ejemplo:

*Creacion de una variable:* let variable1 = 5

*Creacion de un constructor:* this.Persona = 5;

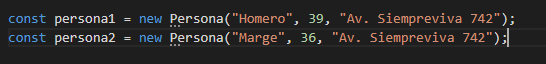
Basicamente cuando creamos una variable lo hacemos con **let nombre = valor** y cuando creamos un objeto lo hacemos con **this.nombre = valor**



Resumiendo, nuestra función constructora, en este ejemplo, creará un objeto con propiedad *nombre* (a través de *this.nombre)*, cuyo valor será el que reciba por el parámetro *nombre* en su llamado; lo mismo para el resto.

**INSTANCIA DE OBJETOS:**

Una vez declarada una función constructora, es posible crear nuevos objetos usándola. Veamos un ejemplo de ello:

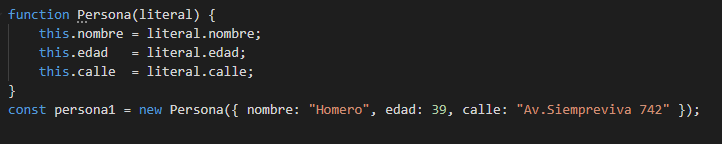


***Persona***, se llama como cualquier función, pasando sus respectivos parámetros, los cuales determinan los valores iniciales asignados a las propiedades del nuevo objeto.

En lo que hay que centrarse es en la palabra reservada ***new***, término que se codifica previamente a la llamada de la función constructora para indicar que estamos creando el nuevo objeto con ese constructor.

**THIS:**

Sirve para **autoreferenciar**, es decir, la usamos para señalar al elemento actual desde el cual estamos programando.



Pero cuando usamos ***this*** fuera de cualquier bloque, referencia al [contexto global](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Operadores/this#contexto_global); y si lo empleamos dentro de una función, su referencia dependerá de cómo la función es llamada.

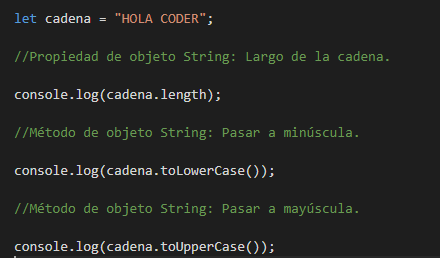
**METODOS Y OPERACIONES CON OBJETOS**

**Funciones y métodos:**Si bien técnicamente vamos a definir funciones para asociar comportamiento a los objetos, es importante identificar que cuando una función está vinculada a un objeto, se la conoce con el nombre de ***método.***Por ejemplo, si tenemos un objeto referenciado en la variable ***persona1*** y este tiene un método ***hola***, pero emplearlo debemos escribir  ***persona1.hola()***,

**Metodos en objetos Java Script:**

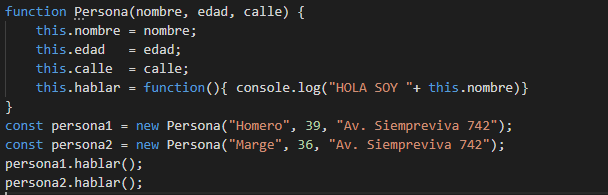
El lenguaje JavaScript cuenta con sus propios [objetos por defecto](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales).

El caso más común es emplear los métodos de string para crear nuevas cadenas con cierto formato.

Ejemplo:  
  


**Definición de métodos personalizados:**

Básicamente lo que se hace para esto es asignar una función a una propiedad en la función constructora.



Cosas a tener en cuenta:

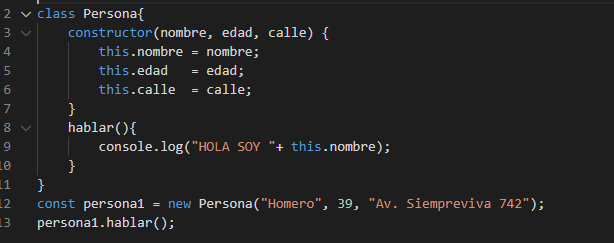
* El método personalizado se refiere a una propiedad del objeto: es necesario crear una propiedad, en este caso ***this.hablar***, a la cual le asociamos una función, definiendo así un método que se puede emplear desde todos los objetos instanciados con este constructor.
* Es posible referenciar una función anónima o una tradicional: podemos definir la función que representa el comportamiento del objeto como función anónima, declarandola en la asignación, o bien podemos detallar el nombre de una función convencional previamente creada. No obstante, para emplear el método es necesario llamarlo usando el identificador de la propiedad. En el ejemplo vemos ese uso en las últimas instrucciones:  persona1.hablar() y persona2.hablar()
* Para emplear las propiedades del objeto en los mètodos es necesario usar ***this***: como observamos en ***console.log("HOLA SOY "+ this.nombre)***, para referenciar el valor de la propiedad nombre del objeto desde el cual se efectúa la llamada al mètodo,  es necesario emplear la autorreferencia con this. De no utilizar el this, la función entendería que *nombre* es una variable a referenciar, y no una propiedad del objeto en cuestión.

**Clases:**

Las clases vienen a solucionar la desventaja que tienen las funciones constructoras, de requerir una propiedad por cada método.

Vemos a la clase como un equivalente de la función constructora, más ordenada en términos de definición.

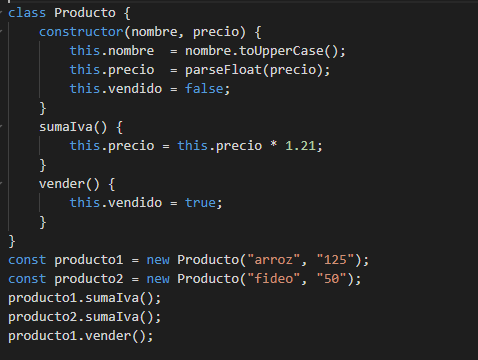
Ejemplo:



Lo importante de las clases es que los métodos se declaran aparte, sin la necesidad de asociarlos a una propiedad en el constructor.

Esto permite una declaración más ordenada, y al momento de crear un nuevo método personalizado podemos simplemente agregarlo sin modificar el constructor de la clase, como se observa en la siguiente declaración de la clase ***Producto***, y sus  dos métodos personalizado ***sumaIva()*** y ***vender()***:

Ejemplo de uso:

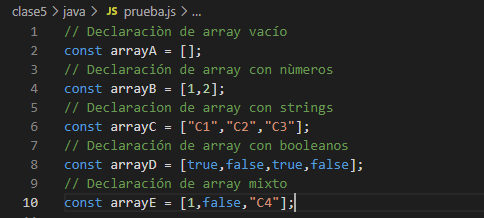


**Clase 6 ( ARRAYS)**

Los **arrays** (también llamados **colecciones**) son un tipo de objeto especial que nos permite agrupar elementos.

Se utilizan cuando necesitamos contar con un conjunto de valores asociados a un listado. Podemos guardar cualquier tipo de dato u objeto.

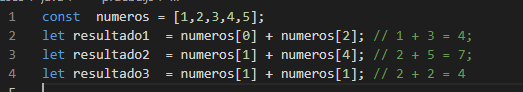
Para crear un ARRAY nos basta con usar los corchetes y especificar los elementos que componen la lista separados por coma (,), como podemos observar en el ejemplo:



Dado que el array es un objeto, es preferible declarar la variable con la palabra **CONST**

**Hay dos tipos de ARRAYS:**

* **Heterogéneos:** Contiene diferentes tipos de datos, por ejemplo, un booleano con un numero con un string.
* **Homogeneos:** Contienen todos los datos del mismo tipo, por ejemplo, un array numérico.



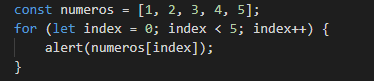
Para acceder a un elemento del array tenemos que emplear su posición indice, como ser:

* 1 está en la posición 0 del array.
* 2 está en la posición 1 del array.
* 3 está en la posición 2 del array.
* 4 está en la posición 3 del array.

Entonces lo hacemos de la siguiente manera: nombre del array [posición indice]

ACCEDER A UN ARRAY MEDIANTE ESTRUCTURA **FOR**

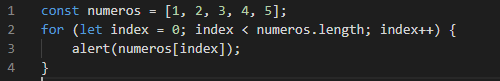
También es posible emplear la estructura **for** para acceder a un elemento del array en cada ciclo. A esta forma de acceso a los elementos se la conoce con el nombre de **“recorrido del array”**:



Lo que dice esto es que va a generar un bucle desde el comienzo del array hasta el número 5.

En caso de no saber la cantidad exacta de elementos dentro de un array utilizamos la propiedad **length.** Esta propiedad lo que va a hacer es contar dentro del array la cantidad de elementos que existen.

Por lo tanto vamos a definir de la siguiente manera:

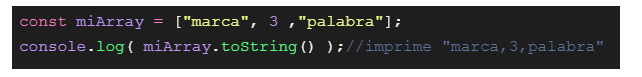
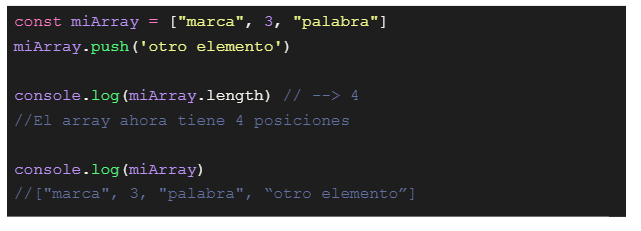
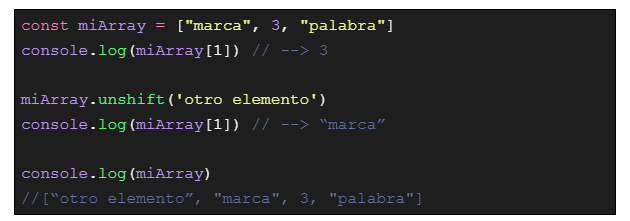
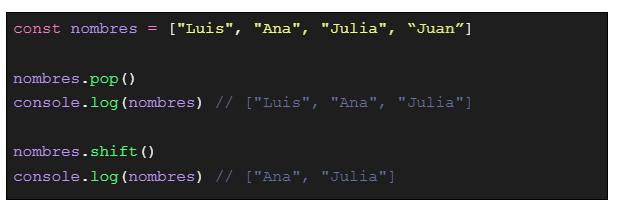
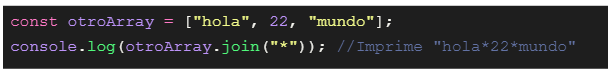
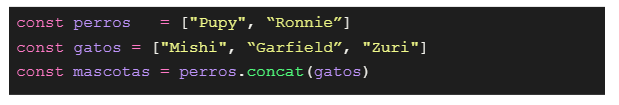
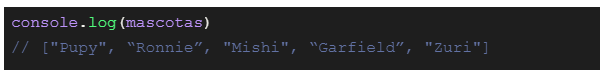
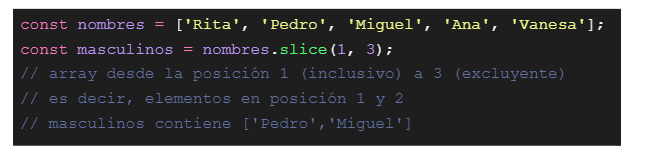
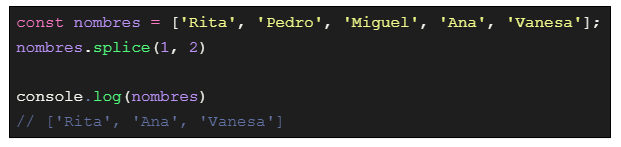
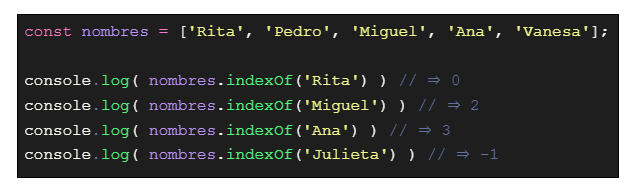
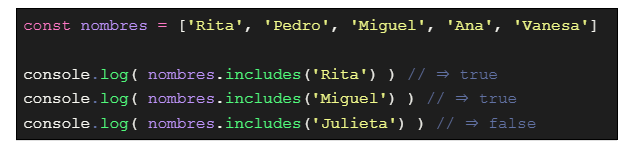
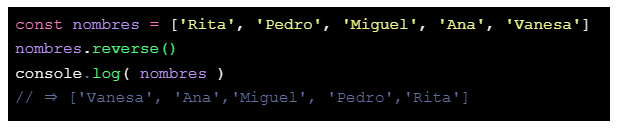


Basicamente que es lo que esta haciendo esta función?, Bueno esta repitiendo el bucle 5 veces. Porque? Porque con la propiedad **length** hicimos que busque la cantidad de elementos que se acumularon ahí dentro.

**METODOS PARA UTILIZAR CON ARRAYS:**

Los métodos sirven para hacer transformaciones de los valores o realizar búsquedas y filtros.

Estos son:

* **toString:**Convierte todos los elementos del array en string. En caso de que los elementos que hayan dentro del string sean mixtos, lo hacemos de la siguiente manera:   
    
  
* **Push**Sirve para sumar un elemento a un array ya existente. El elemento se agrega al final y si contamos con un length la cantidad de elementos a medida que agreguemos con push, el numero del length se va a incrementar en uno.  
  
* **Unshift**Sirve para sumar un elemento a un array ya existente. El elemento se agrega al inicio del array y corre a todos los elementos una posición.   
    
    
    
  Tanto el Unshift como el Push pueden recibir varios parámetros, es decir, podemos agregar mas de un elemento a la vez, separándolos con coma.
* **Pop** y **Shift**Sirven para eliminar elementos de los array. El **pop()** elimina el ultimo elemento, mientras que el **shift**() elimina el primero.  
    
  
* **Join**Sirve para juntar todos los elementos de un array en una cadena string, indicando como parámetro el separador.  
    
  
* **Concat**Sirve para unir dos arrays, es decir crear uno nuevo desde los elementos de dos arrays previamente declarados.   
    
     
    
  Tener en cuenta que esto va a funcionar siempre y cuando llamemos a **concat** desde el primer array y demos como parametro el segundo array.
* **Slice**Sirve para crear un nuevo array en base a recortes de otro. Se le pasan dos parámetros. Uno **indice inclusivo (desde donde)** otro **indice excluyente (hasta donde).**  
  
* **Splice**Sirve para eliminar uno o varios elementos de un array en cualquier posición. Recibe dos parámetros: **indice (donde se ubica el método para trabajar** y **cantidad de elementos a eliminar** desde esa posición.  
    
    
    
  En este ejemplo le digo al método **splice(1, 2)**,que se posicione en el índice 1 del array y desde allí elimine 2 elementos.
* **indexOf**Sirve para obtener el indice de un elemento en un array siempre y cuando este exista en el mismo. Basicamente nos va a decir en que posición del array se encuentra el elemento. En caso que el elemento no exista en el array, nos arroja **-1** (Eso significa que no existe el elemento buscado)  
    
  
* **includes**Sirve para saber si un elemento existe dentro del array, mediante el retorno de un valor booleano.  
    
  
* **reverse**Sirve para invertir el orden de los elementos dentro de un array. **OJO, ES DESTRUCTIVO, MODIFICA EL ARRAY ORIGINAL.**

**IMPORTANTICIMO: “\n”**, asi es como hacemos un salto de línea en el mensaje de un prompt o un alert.

**ARRAY DE OBJETOS:**

Iterador **for…of:** Lo que hace es acceder a cada elemento de un array de uno en uno.  
Se le hace la siguiente estructura: 

**Clase 7 (FUNCIONES DE ORDEN SUPERIOR)**