Programación Imperativa

Bimestre 1

Apuntes Rainvare

Pensamiento computacional

"El pensamiento computacional implica resolver problemas, diseñando sistemas, y entender el comportamiento humano a partir de los conceptos fundamentales a la informática. Pensamiento computacional incluye una gama de herramientas mentales que reflejan la amplitud del campo de la informática". -Jeanette Wing

Elementos:

- La descomposición: Este proceso consiste en tomar un problema complejo y seccionarlo en partes más pequeñas que sean manejables.
- Los patrones: una vez hemos seccionado el problema, se identifica cuáles son las partes que se repiten o relacionan, para aguparlas.
- La abstracción: eliminar la información irrelevante y enfocarse en analizar los elementos que son fundamentales para resolver el problema.
- Los algoritmos: expresión de pasos secuenciales que son necesarios para resolver el problema.

Pensamiento computacional es identificar procesos para resolver problemas

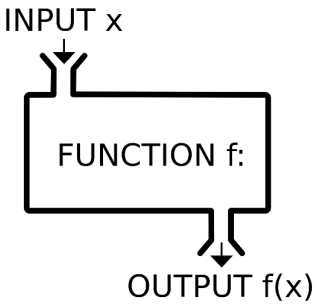
Juego para desarrollar el pensamiento computacional: lightbot

El código

Los lenguajes de programación se componen de códigos, estos se estructuran gracias a la sintaxis. Y auque cada lenguaje tiene una sintaxis propia; hay elementos comunes a todos los lenguajes.

- Los procedimientos son un serie de acciones que se le piden ejecutar a la computadora.
- Los programas son el conjunto de acciones a ejecutar
- Las funciones son un dipositivo que realiza una acción y devuelve un valor
- Los primitivos son una serie de valores o datos básicos

La computadora interpreta símbolos que son combinados en una estructura o sintaxis.



Operadores de comparación

```
function cuadrado(n){
    return n * n
}

let num = cuadrado(2);
```

Plataforma para desarrollar los conocimientos básicos de sintaxis Mumuki

```
program {
  Mover4AlNorte()
  Poner(Negro)
}

procedure Mover4AlNorte() {
  Mover(Norte)
  Mover(Norte)
  Mover(Norte)
  Mover(Norte)
}
```

Programa principal
Acá determinamos cuál será el comportamiento del personaje.

Primitivas
Acá ya no son símbolos iconográficos, son textos que contienen un significado diseñado por la persona que programó el juego.

Función o Procedimiento
Acá determinamos cuál será el comportamiento de nuestras funciones y, por lo tanto, del personaje o cursor, en este caso.

Javascript

Básico

IMPORTANT! Antes de iniciar, debes tener instalado un editor o IDE como VisualCode; y Node.JS para ejecutar el código en tu computadora.

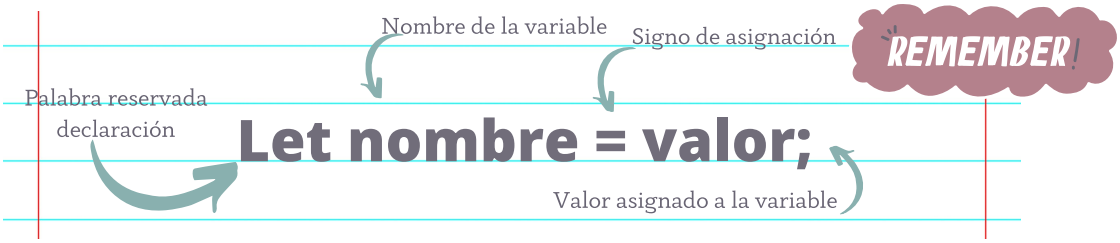
JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript.
Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Variables

Las variables son espacios de la memoria del computador, en donde se almacenan datos.
Hay diferentes tipos de variables:

- VAR *Es accesible de manera global
- LET *Solo funciona en un scope o bloque de código
- CONST *Al igual que LET, solo tiene alcance dentro de su bloque. No se puede reescribir una vez declarada y asignada

Estas tres palabras reservadas nos permiten "declarar" una variable al asignarle un nombre. Además, nos permiten indentificar el tipo de variable. Pero debemos recordar que, los nombres de las variables no deben empezar con un n°, ni deben contener acentos o signos especiales.



Una vez que la variable ha sido declarada; no es necesario usar la palabra reservada de nuevo. Si se llama de nuevo junto al signo =, esta variable se reescribe.

Tipos de datos

JavaScript tiene seis tipos de variables primitivas y, datos no-primitivos denominados también como tipos de datos objeto.

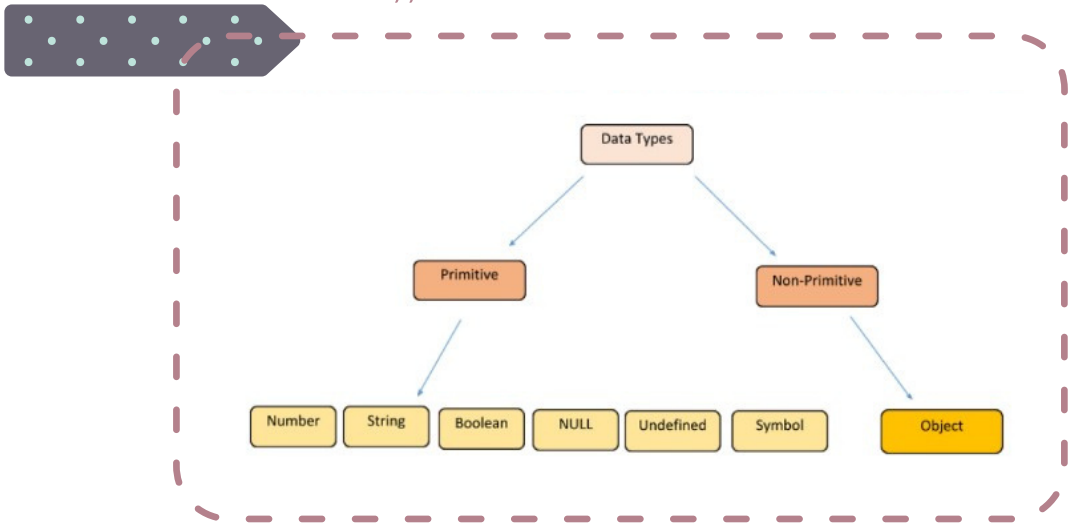
"Las variables en JavaScript no están asociadas directamente con ningún tipo de valor en particular, y a cualquier variable se le puede asignar (y reasignar) valores de todos los tipos:" MDN

- NUMBER Estos son datos númericos con los que podemos realizar operaciones.
- STRING Estos son datos de cadena de texto o caracteres
- BOOLEAN Estos son datos lógicos que, solo pueden tener dos valores: true y false.
- UNDEFINED Estos indican que el valor de la variable no ha sido definido.
- NULL Estos indican que el valor de la variable es nulo o desconocido.

NaN no es una variable; es un valor que indica que lo que contiene la variable no es un número.

Comentarios

Los comentarios en Javascript inician con dos barras inclinadas //



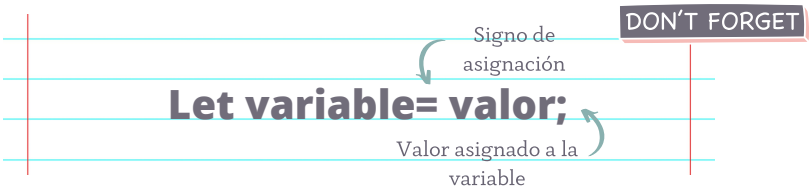
Operadores

Los operadores nos permiten manipular los datos de las variables, y realizar operaciones con estos datos. Existen los siguientes tipos de operadores:

la concatenación se realiza con el operador de suma: +
Ejem."string" + 3;

- Asignación (=)

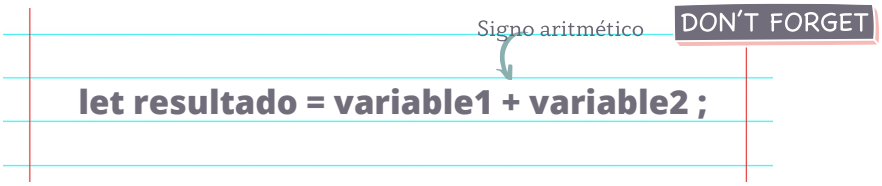
Este signo le da a la variables el valor situado a su derecha



- Aritméticos

Estos nos permiten hacer operaciones y obtener resultados

"+"suma,
"-" resta,
"*" multiplicación,
"/" división y
"%" módulo (que devuelve el resto de una división)



Los Aritméticos son incrementales. Demanera que podemos escribirlos dos veces para realizar operaciones. Ejem:

Let resultado = 10+1; o Let resultado =10++.
Ambas son iguales a 11.

Funciones

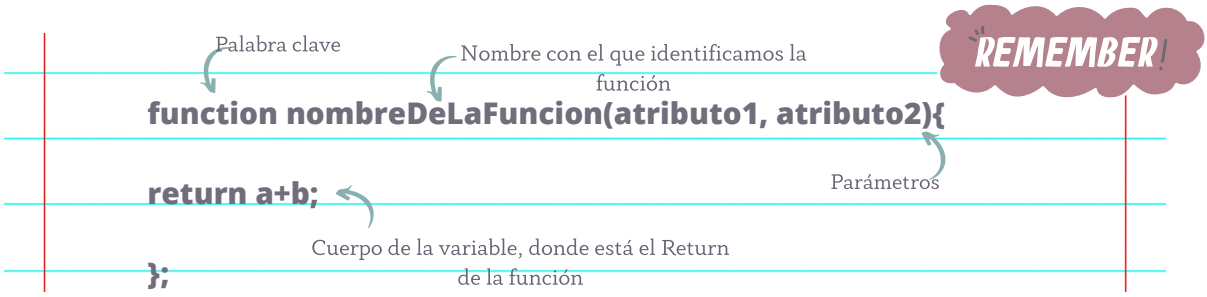
Las funciones son bloques de código que, realizan una tarea u operación; pues estas devuelven un resultado.
Las podemos invocar o llamar tantas veces como necesitemos realizar esa tarea. Estas definen el scope o alcance de las variables que utiliza. Las funciones pueden ser declaradas o expresadas (estas últimas son las funciones que se declaran como valor de una variable).

Estructura de una función

se usa la palabra clave function para declararla, luego se nombre y se indica cuáles son los valores de entrada o parámetros. Por último, se indica qué operación va a ejecutarse y cuál será el resultado o return que esta deberá devolver.

Si la función no lleva parámetros, debe ir con paréntesis vacíos:

Ejem.
function nombre( ){
return a+b
};



NOTES

- ¿Cómo definirías a Node.js y JavaScript? ¿En qué se diferencian?

Javascript es un lenguaje de programación y Node.js es un entorno de ejecución para ejecutar JS fuera del navegador. JS se puede ejecutar del lado del cliente; mientras que, Node.js se ejecuta del lado del servidor.

- ¿Cómo ejecuto mi archivo .js en Visual Studio Code? ¿Cómo me muevo hacia la carpeta donde se encuentra mi archivo?

En el editor de texto, se abre la terminal y con el código cd nos movemos hacia la carpeta que tiene el archivo; luego se ejecuta al llamarlo: "archivo.js + enter".
Si se ejecuta con node: "node archivo.js + enter".

- ¿Para qué sirve una variable?

Una variable sirve para almacenar y manipular un dato.

- En la vida real, ¿qué sería una variable?

En la vida real una variable puede ser una caja, un cajón o un contenedor, en donde guardamos cosas.

- ¿Cuáles son los distintos tipos de datos?

Hay datos de cadena de carácteres, numéricos, booleanos, nulos e indefinidos.

- ¿Con qué operadores trabaja JavaScript?

JS trabaja con operadores de asignación(=), concatenación (+) y aritméticos (+,-,/,*, %).

- ¿Para qué utilizamos las funciones? ¿Cuál es la diferencia entre ejecutar y declarar una función?

Usamos las funciones para realizar una tarea u operación. Al declarar una función, la denominamos por primera vez y definimos qué hará. Mientras que, al ejecutarla la llamanos por su nombre.

Lógica

Los operadores son elementos que nos permiten controlar ciertos aspectos de nuestro código y lo hacen más eficiente; porque reduce la cantidad de código que debemos escribir.

comparación
> mayor que
>= mayor o igual
< menor que
<= menor o igual
== igual

Operadores de comparación

- Comparación simple: Se comparan dos variables, y devuelve un booleano.

Comparación simple — DON'T FORGET

```
"variable1" == variable2 ;
```

- Comparación estricta: Se comparan dos variables en valor y tipo, y devuelve un booleano.

Comparación estricta — DON'T FORGET

```
variable1 === variable2 ;
```

Operadores lógicos

Los operadores lógicos combinan los valores booleanos.
Son tres: AND (&&), OR (||) y Negación (!).

- && Comparan dos valores, ambos deben ser verdaderos para que se aplique la condición.

Operador AND — REMEMBER!

```
(15>3) && (45<300);
```

- || Comparan dos valores, basta con que uno de ellos sea verdaderos para que se aplique la condición.

Operador OR — REMEMBER!

```
(15>3) || (45>=300);
```

- ! Indica que el valor es contrario o negado

Operador de Negación — REMEMBER!

```
! (45>300);
```

Falsy

Fasly indica que el valor de la variable es falso.

Hay valores que SIEMPRE son FALSO

IMPORTANT!

- **false**
- **0 (cero)**
- **"" (string vacío)**
- **null**
- **undefined**
- **NaN (Not a Number. Ejemplo: el resultado de 1 dividido por 0)**

Trully

Trully indica que el valor de la variable es verdadero.

Todo lo demás es truthy. Eso incluye:

- '0' (una cadena que contenga un simple 0)
- 'false' (un string que contenga el texto "false")
- [] (un arreglo vacío)*
- {} (un objeto vacío)*
- function(){} (una función vacía)

Comparaciones

Para realizar comparaciones, y saber si estos valores son Trully o False se debe serguir las siguientes reglas:

- son equivalentes:

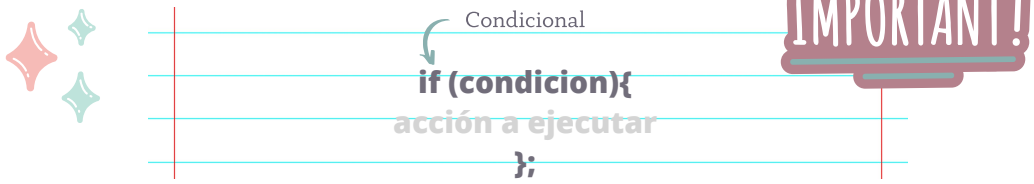
False = Null = " "	**Estas son equivalentes**
Null = Undefined	**Estas son equivalentes**
NaN	**No es equivalente a nada**

Condicionales

Las condicionales son evaluaciones que nos permiten filtrar las acciones a realizar.

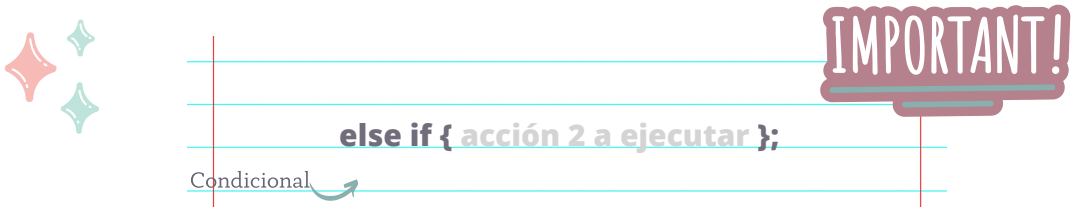
IF

Las condicional simple o IF nos permiten ejecutar un bloque de código solo si se cumple la condición.



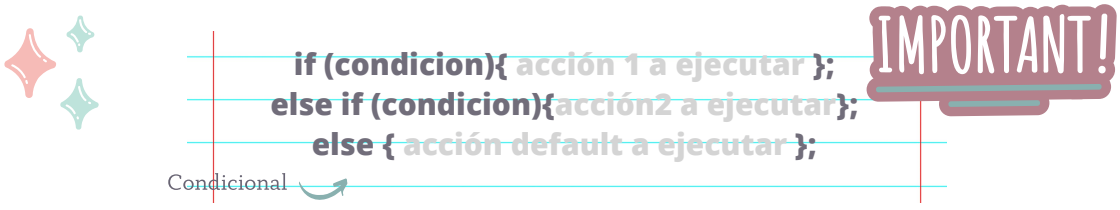
Else IF

Las condicional Else if nos permiten ejecutar un bloque de código adicional si no se cumple la condición del if.



Else

Las condicional Else nos permiten ejecutar otro bloque de código adicional si no se cumple la condición del if ni del else if.



If ternario

Las condicional if abreviada. Esta nos permiten ejecutar una condición en una línea de código.

REMEMBER!

Primera expresión - se ejecuta si se cumple la condición

```
condicion ? acción1 : acción2
```

operador

Segunda expresión - se ejecuta si no se cumple la condición

La segunda condición no es obligatoria; puede indicarse el vacío con " "

Switch

Las condicional Switch nos permite evaluar diferentes casos con un solo valor.

Y crear una acción por default en caso de que ninguno de los casos sea verdadero. Todos los switch deben terminar con un break.

Inicio de iteración

```
Switch (expresión){

Case (valor A):
 acción 1 a ejecutar
break;

Case (valor B):
 acción 1 a ejecutar
break;

Default:  acción 1 a ejecutar ;

};
```

IMPORTANT!

estructura del caso

estructura del deafult