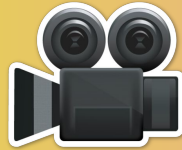


***RECUERDA PONER A GRABAR LA  
CLASE***





***¿DUDAS DEL ON-BOARDING?***

**MIRALO AQUI**



Clase 03. JAVASCRIPT

# ***CICLOS/ITERACIONES***



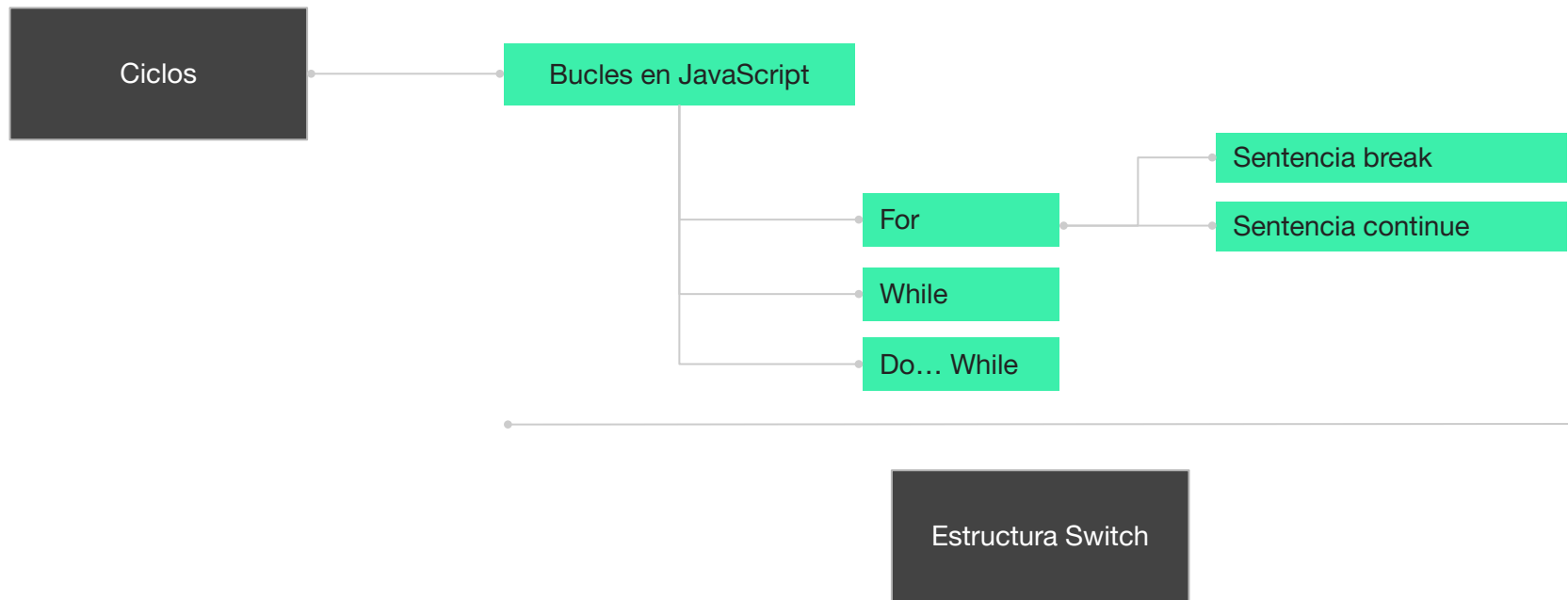
## ***OBJETIVOS DE LA CLASE***

- Comprender qué es un **ciclo** o **bucle** y cómo nos permite repetir operaciones similares fácilmente.
- Conocer qué **tipos de ciclos** podemos emplear y cuáles son sus diferencias.
- Analizar cómo combinar **operadores lógicos, ciclos y funciones** para resolver cada problema.

# ***MAPA DE CONCEPTOS***

# MAPA DE CONCEPTOS CLASE 3

¡Para  
recordar!



# MÓDULOS DE TRABAJO

## MÓDULO 0 NIVELACIÓN

**CLASE 0 -**  
INTRODUCCIÓN A JAVASCRIPT

## MÓDULO 1 CONCEPTOS BÁSICOS

**CLASE 1 -**  
CONCEPTOS GENERALES:  
SINTAXIS Y VARIABLES

**CLASE 2 -**  
CONTROL DE FLUJOS

**CLASE 3 -**  
CICLOS E ITERACIONES

**CLASE 4 -**  
FUNCIONES

- **Desafío entregable**

## MÓDULO 2 OBJETOS & ARRAYS

**CLASE 5 -**  
OBJETOS

**CLASE 6 -**  
ARRAYS

**CLASE 7 -**  
FUNCIONES DE ORDEN SUPERIOR

- **1ra pre-entrega**

# ***GLOSARIO:***

## ***Clase 2***

**Condicionales:** cuando en programación hablamos de condicionales, hablamos de una estructura sintáctica que sirve para tomar una decisión a partir de una condición.

**Estructura IF:** es la más utilizada en la mayoría de los lenguajes. Si la condición se cumple (es decir, si su valor es true) se ejecutan todas las instrucciones que se encuentran dentro de {...}. Si la condición no se cumple (es decir, si su valor es false) no se ejecuta ninguna instrucción contenida en {...} y el programa continúa ejecutando el resto de instrucciones del script.

**IF...ELSE:** en ocasiones, las decisiones que se deben realizar no son del tipo "si se cumple la condición, hazlo; si no se cumple, no hagas nada". Normalmente las condiciones suelen ser del tipo "si se cumple esta condición, hazlo; si no se cumple, haz esto otro".





# ***HERRAMIENTAS DE LA CLASE***

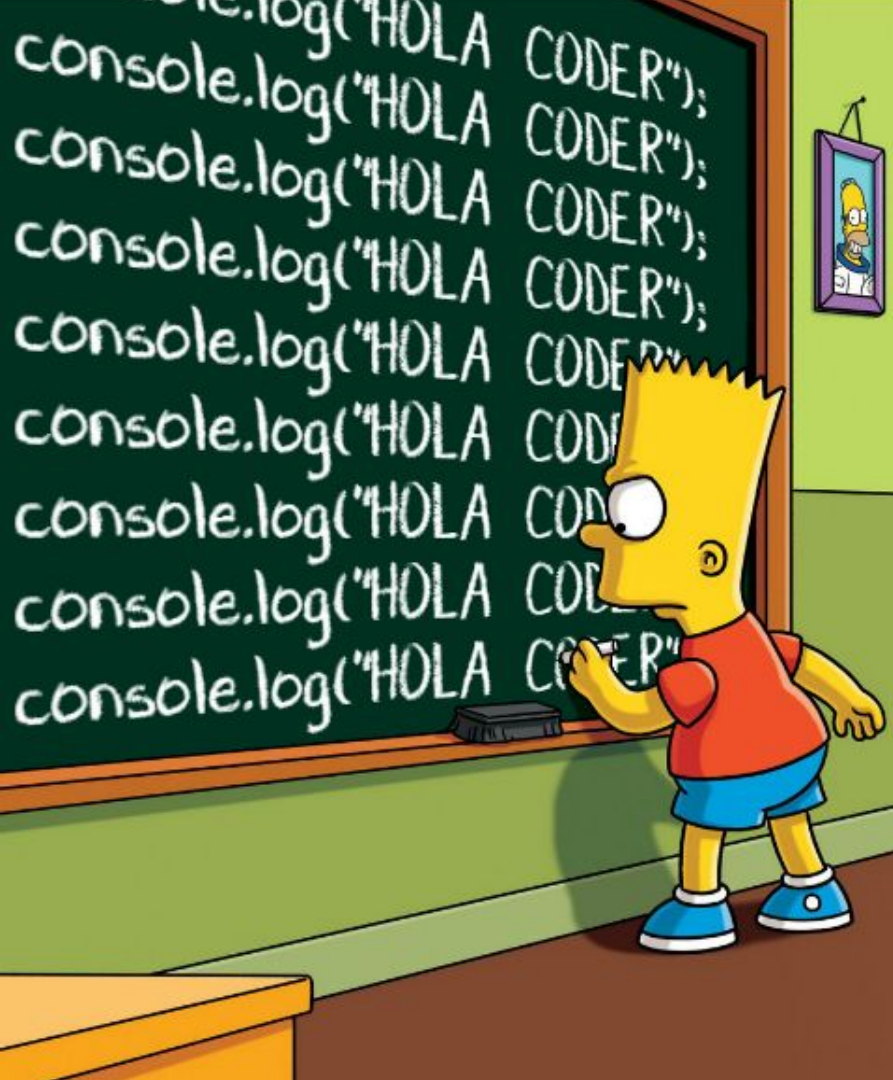
*Les compartimos algunos recursos para acompañar la clase*

- Guión de clase N° 3 [aquí](#).
- Quizz de clase N° 3 [aquí](#)
- Booklet de Javascript [aquí](#)
- FAQs de Javascript [aquí](#)

***EMPEZAMOS...***



***CICLOS***



## ***CICLOS EN JAVASCRIPT***

Los ciclos, también conocidos como bucles o iteraciones **son un medio rápido y sencillo para hacer algo repetidamente.**

Si tenemos que hacer alguna operación más de una vez en el programa, de forma consecutiva, usaremos las estructuras de bucles de JavaScript: **for**, **while** o **do...while**.

# ***TIPOS DE BUCLES***



- ***CICLOS POR CONTEO***

Repiten un bloque de código un número de veces específica.

Estructura **for**.

- ***CICLOS CONDICIONALES***

Repiten un bloque de código mientras la condición evaluada es verdadera. Estructuras **while** y **do...while**.

***FOR***

# ***ESTRUCTURA FOR***



```
for(desde; hasta; actualización) {  
    ... //lo que se escriba acá se ejecutará mientras dure el  
    ciclo  
}
```

El "**desde**" es la zona en la que se establecen los valores iniciales de las variables que controlan el ciclo.

El "**hasta**" es el único elemento que decide si se repite o se detiene el ciclo.

La "**actualización**" es el nuevo valor que se asigna después de cada repetición a las variables que controlan la repetición.

# EJEMPLO PRÁCTICO



En el siguiente ejemplo utilizamos un **for** para contar de 0 a 9.

```
for (let i = 0; i < 10; i++) {  
    alert(i);  
}
```

Ahora usamos **for** para contar de 1 a 10.

```
for (let i = 1; i <= 10; i++) {  
    alert(i);  
}
```



# ***EJEMPLO APLICADO FOR (1): TABLAS***

Algoritmo para calcular la tabla de multiplicar de un número.

```
// Solicitamos un valor al usuario
let ingresarNumero = parseInt(prompt("Ingresar Numero"));
// En cada repetición, calculamos el número ingresado x el número de repetición (i)
for (let i = 1; i <= 10; i++) {
    let resultado = ingresarNumero * i ;
    alert(ingresarNumero + " X " + i + " = " + resultado);
}
```

# ***EJEMPLO APLICADO FOR (2): TURNOS***

Algoritmo para dar turno del 1 al 20 a los nombres ingresados

```
for (let i = 1; i <= 20; i++) {  
  // En cada repetición solicitamos un nombre.  
  let ingresarNombre = prompt("Ingresar nombre");  
  // Informamos el turno asignado usando el número de repetición (i).  
  alert(" Turno  N° "+i+" Nombre: "+ingresarNombre);  
}
```

# ***SENTENCIA BREAK***

A veces, cuando escribimos una estructura **for**, necesitamos que bajo cierta condición el ciclo se interrumpa. Para eso se utiliza la **sentencia break**. Al escribir esa línea dentro de un ciclo **for**, el mismo se interrumpirá como si hubiera finalizado.

```
for (let i = 1; i <= 10; i++) {  
  //Si la variable i es igual 5 interrumpo el for.  
  if(i == 5){  
    break;  
  }  
  alert(i);  
}
```

# ***SENTENCIA CONTINUE***

A veces, cuando escribimos una estructura **for**, necesitamos que bajo cierta condición, el ciclo saltee esa repetición y siga con la próxima. Para eso se utiliza la **sentencia continue**.

```
for (let i = 1; i <= 10; i++) {  
    //Si la variable i es 5, no se interpreta la repetición  
    if(i == 5){  
        continue;  
    }  
    alert(i);  
}
```

Ejemplo  
en vivo



***¡VAMOS A PRACTICAR LO VISTO!***

***CODER HOUSE***



***BREAK***

**¡5/10 MINUTOS Y VOLVEMOS!**

***WHILE***

# WHILE

La estructura **while** permite crear bucles que se ejecutan **cero o más veces**, dependiendo de la condición indicada.

El funcionamiento del bucle **while** se resume en: ***mientras se cumpla la condición indicada, repite las instrucciones incluidas dentro del bucle.***



# WHILE



Cuando usamos **while**, asumimos que en algún momento la repetición va a finalizar; si la comparación no se realiza adecuadamente podemos generar el llamado “bucle infinito”:

```
let repetir = true;
while(repetir) {
    console.log("Al infinito y...¡Más allá!");
}
```

7818 Al infinito y...¡Más allá!

# ***EJEMPLO APLICADO WHILE: ESC***

Algoritmo que solicita una entrada al usuario hasta que ingresa “ESC”

```
let entrada = prompt("Ingresar un dato");  
//Repetimos con While hasta que el usuario ingresa "ESC"  
while(entrada != "ESC" ){  
    alert("El usuario ingresó "+ entrada);  
    //Volvemos a solicitar un dato. En la próxima iteración se evalúa si no es ESC.  
    entrada = prompt("Ingresar otro dato");  
}
```

***DO...WHILE***

# DO...WHILE

La estructura **do...while** permite crear bucles que se ejecutan una o más veces, dependiendo de la condición indicada.

A diferencia de while, **garantiza que el bloque de código se interpreta al menos una vez, porque la condición se evalúa al final.**

```
let repetir = false;
do{
    console.log("¡Solo una vez!");
}while(repetir)
```

# ***EJEMPLO APLICADO DO...WHILE: N°***

Algoritmo que solicita una entrada y se detiene cuando **NO** es un número

```
let numero = 0;
do{
    //Repetimos con do...while mientras el usuario ingresa un n°
    numero = prompt("Ingresar Número");
    console.log(numero);
    //Si el parseo no resulta un número se interrumpe el bucle.
}while(parseInt(numero));
```

***SWITCH***

# ***SWITCH***

La estructura **switch** está especialmente diseñada para manejar de forma sencilla **múltiples condiciones sobre la misma variable** (técnicamente se podría resolver con un **if**, pero el uso de **switch** es más ordenado).

Su definición formal puede parecer confusa, pero veamos un ejemplo para entender su simpleza.

# SWITCH

```
switch(numero) {  
  case 5:  
    ...  
    break;  
  case 8:  
    ...  
    break;  
  case 20:  
    ...  
    break;  
  default:  
    ...  
    break;  
}
```

Cada condición se evalúa y, si se cumple, se ejecuta lo que esté indicado dentro de cada **case**.

Normalmente, después de las instrucciones de cada case se incluye la sentencia **break** para terminar la ejecución del **switch**, aunque no es obligatorio.

*¿Qué sucede si ningún valor de la variable del **switch** coincide con los valores definidos en los **case**?*

En este caso, se utiliza el valor **default** para indicar las instrucciones que se ejecutan cuando ninguna condición anterior se cumplió.



# ***EJEMPLO APLICADO WHILE Y SWITCH:- ENTRADAS***

```
let entrada = prompt("Ingresar un nombre");  
//Repetimos hasta que se ingresa "ESC"  
while(entrada != "ESC" ){  
    switch (entrada) {  
        case "ANA":  
            alert("HOLA ANA");  
            break;  
        case "JUAN":  
            alert("HOLA JUAN");  
            break;  
        default:  
            alert("¿QUIÉN SOS?")  
            break;  
    }  
    entrada = prompt("Ingresar un nombre");  
}
```

Algoritmo que hace la operación según la entrada, pero ignora la ejecución de bloque si la entrada es en “ESC”.



# ***¡LO MÁS IMPORTANTE!***

Todas los temas que vimos (y los que vamos a ver), se pueden  
(y deben) combinar entre sí.

De forma que dentro de una función, pueda existir un **haya** un  
**condicional**, con un **for** adentro, y dentro de ese for, un  
**while**...así la combinación es infinita.

**¡Ahí es cuando la programación JavaScript empieza a  
volverse interesante!**

Ejemplo  
en vivo



***¡VAMOS A PRACTICAR LO VISTO!***

***CODER HOUSE***



# ***#FindTheBug***

***Encuentra el error***

Analizaremos el código para asegurarnos que corre bien.

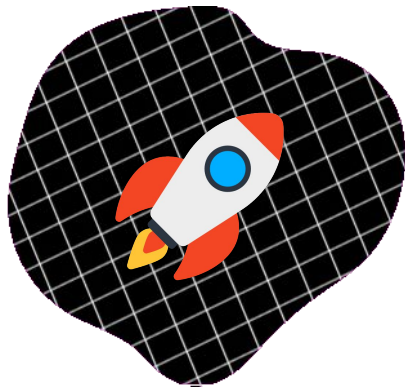
Si encontramos errores, ¡los solucionaremos!

**TIEMPO ESTIMADO: 5/10 MIN**

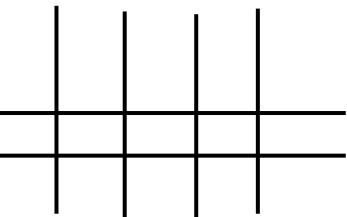
# ***PREPARÁNDONOS PARA EL DESAFÍO ENTREGABLE N° 1***

*En la clase 4 finaliza el primer módulo y se entregarán las consignas del **primer desafío entregable** del curso. El mismo, incluirá temas vistos en las clases 1, 2, 3 y 4.*





- El entregable se compone de temas vistos hasta el momento, más otros que verán durante el **módulo completo** 💪.
- Te recomendamos ir avanzando con los "Hands On" y "Desafíos Complementarios" ✨
- Recuerden que recién la consigna del desafío se entrega ¡en la clase N° 4! 🙌 **Y tendrán hasta 7 días para resolver el desafío y subirlo.**



# ***DESAFÍO ENTREGABLE N° 1***

*Compuesto por...*



- a) Crear un algoritmo con un condicional ✓
- b) **Crear un algoritmo utilizando un ciclo** ✓
- c) Armar un simulador interactivo, la estructura final de tu proyecto integrador 🕒



# ***CREAR UN ALGORITMO UTILIZANDO UN CICLO***

Tomando como base los ejemplos anteriores de la estructura **for**, **while** y **do...while**, crear un algoritmo que repita un bloque de instrucciones.



# CREAR UN ALGORITMO UTILIZANDO UN CICLO

**Formato:** Página HTML y código fuente en JavaScript. Debe identificar el apellido del alumno/a en el nombre de archivo comprimido por "claseApellido".

**Sugerencia:** Usamos la instrucción **for** para repetir un número fijo de veces. Mientras que usamos **while** cuando queremos repetir algo hasta que se deje de cumplir una condición.

Desafío  
Complementario



**>> Consigna:** Tomando como base los ejemplos anteriores de la estructura **for** y **while**, crear un algoritmo que repita un bloque de instrucciones. En cada repetición es necesario efectuar una operación o comparación para obtener una salida por alerta o consola.

**>>Aspectos a incluir en el entregable:**

Archivo HTML y Archivo JS, referenciado en el HTML por etiqueta `<script src="js/miarchivo.js"></script>`, que incluya la definición de un algoritmo en JavaScript que emplee bucles e instrucciones condicionales.

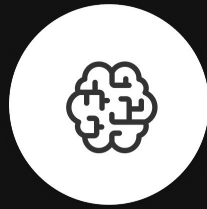
# ***CREAR UN ALGORITMO UTILIZANDO UN CICLO***

## **>>Ejemplo:**

- Pedir número mediante prompt y sumarle otro número en cada repetición, realizando una salida por cada resultado
- Pedir un texto mediante prompt, concatenar un valor en cada repetición, realizando una salida por cada resultado, hasta que se ingresa “ESC”.
- Pedir un número por prompt, repetir la salida del mensaje “Hola” la cantidad de veces ingresada.

***¿PREGUNTAS?***





## ***¡PARA PENSAR!***

*¿Te gustaría comprobar tus conocimientos de la clase?*

Te compartimos a través del chat de zoom  
el enlace a un breve quiz de tarea.

*Para el profesor:*

- *Acceder a la carpeta "Quizzes" de la camada*
  - *Ingresar al formulario de la clase*
  - *Pulsar el botón "Invitar"*
  - *Copiar el enlace*
- *Compartir el enlace a los alumnos a través del chat*



# ***RECURSOS:***

- Bucles |

***Los apuntes de Majo (Página 17 a 19).***

***Te lo explico con gatitos. Bucle FOR.***

***Te lo explico con gatitos. Bucle WHILE.***

- Funciones |

***Los apuntes de Majo (Página 20).***

***Te lo explico con gatitos. Parte 1.***

***Te lo explico con gatitos. Parte 2.***

- Documentación |

***Documentación FOR.***

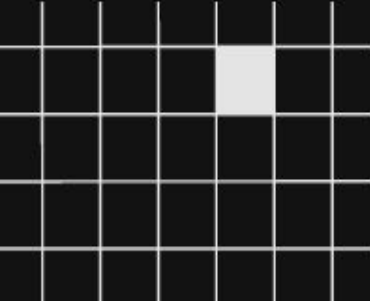
***Documentación WHILE.***

Disponible en [nuestro repositorio.](#)



# ***¡MUCHAS GRACIAS!***

Resumen de lo visto en clase hoy:

- Ciclos: for, while, do while.
  - Operador switch.
- 



***OPINA Y VALORA ESTA CLASE***

***#DEMOCRATIZANDO LA EDUCACIÓN***

***CODER HOUSE***