



<codoa
codo/>



Maurisandev



@MauriDeveloper



maurisan4011@gmail.com

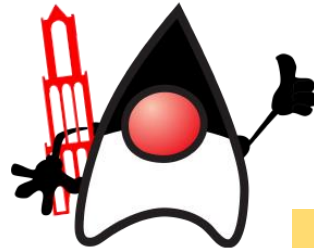
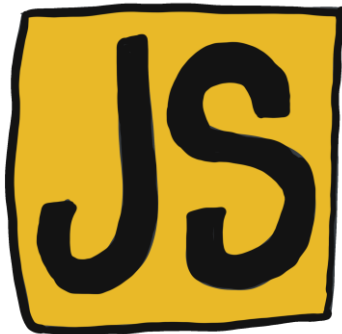


Curso Java FullStack

Codo a Codo
4.0

Clase-16

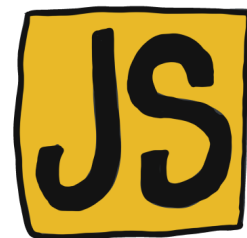
JavaScript





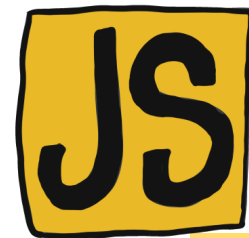
Tipos de Datos

- ✓ **String:** Secuencia de caracteres que representan un valor.
- ✓ **Number:** Valor numérico, entero, decimal, etc.
- ✓ **Boolean:** Valores true o false.
- ✓ **Null:** Valor nulo.
- ✓ **Undefined:** Valor sin definir.
- ✓ **Symbol:** Tipo de dato cuyos casos son únicos e inmutables.





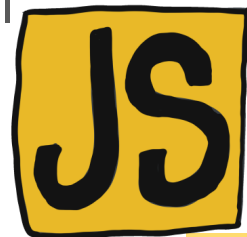
CONTROL DE FLUJOS





¿Que es una Estructura de Control?

- Es una instrucción que cambia el flujo de ejecución del código
- Nos permiten controlar qué líneas de código se ejecutan y en qué orden lo hacen
- Existen dos estructuras de control (if y switch) y su uso depende del problema

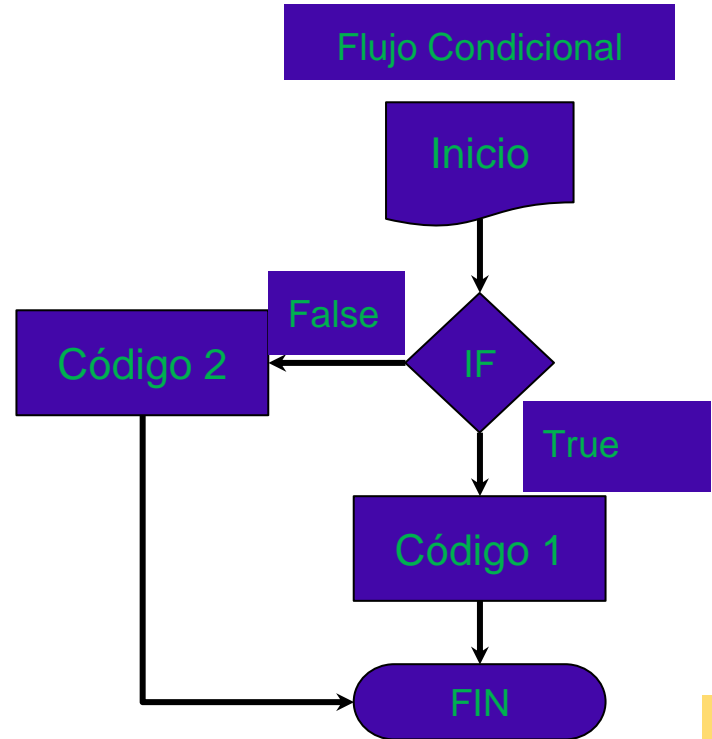
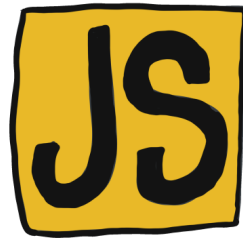
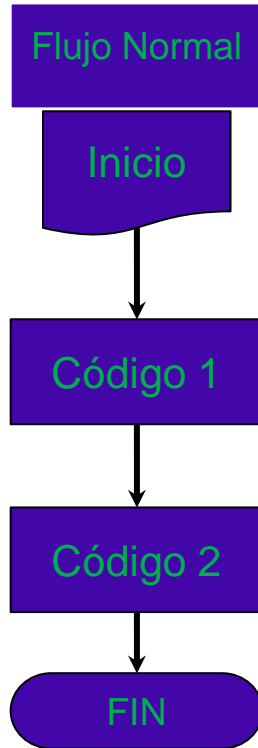




<codoo
codoo/>



Flujo de Ejecución

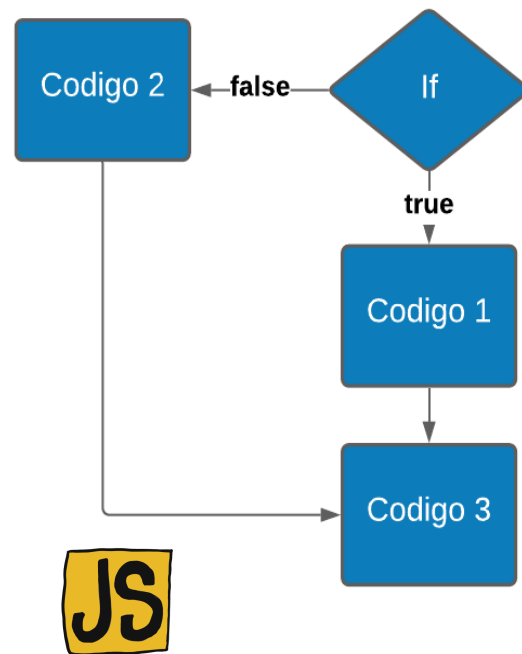




<codigo
codigo/>

Estructura de un IF

- Primero se evalúa la expresión en el if y se decide que camino tomar
- La expresión a evaluar tiene que dar como resultado un boolean, true o false
- Si el resultado de evaluar la expresión es true se ejecuta el código 1
- Si no (la expresión dio false) se ejecuta el código 2
- En cualquier caso se ejecuta el código 3



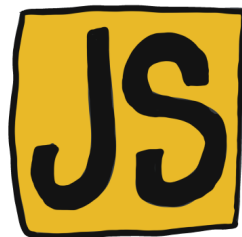


<codigo
codigo/>

Condicional: definición

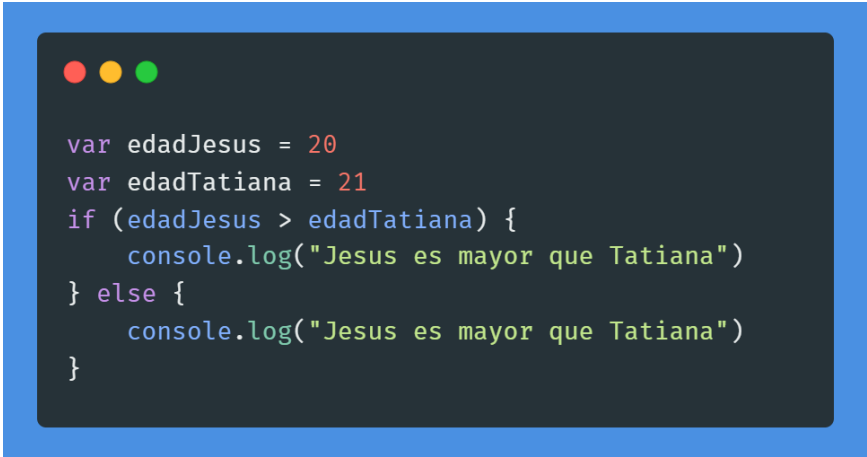
Cuando en programación hablamos de condicionales, hablamos de una estructura sintáctica que sirve para tomar una decisión a partir de una condición.

Si <condición> entonces <operación>



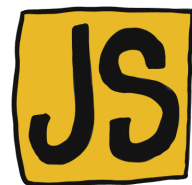
JS – Condicionales

Una sentencia condicional es un conjunto de comandos que se ejecutan si una condición es verdadera. JavaScript soporta dos sentencias condicionales: if...else y switch



```
var edadJesus = 20
var edadTatiana = 21
if (edadJesus > edadTatiana) {
    console.log("Jesus es mayor que Tatiana")
} else {
    console.log("Jesus es mayor que Tatiana")
}
```

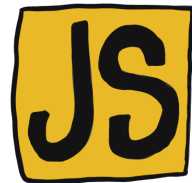
Se utiliza la sentencia if para comprobar si la condición lógica es verdadera. Se utiliza la opción else para ejecutar una sentencia si la condición es falsa.



JS - Condicionales / if-else if- else

También se pueden armar sentencias más complejas usando **else if** para tener múltiples condiciones, como se muestra a continuación:

```
if (edadJesus > edadTatiana) {  
  console.log("Jesus es mayor que Tatiana")  
} else if (edadTatiana > edadJesus) {  
  console.log("Tatiana es mayor que Jesus")  
  
} else {  
  console.log("Tatiana y Jesus tienen la misma edad")  
}
```

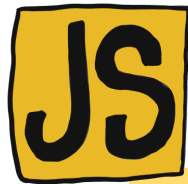




Condicionales Anidadas

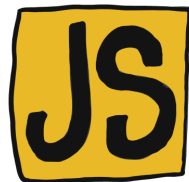


```
var edad = prompt("Porfavor ingresa tu edad para acceder a esta pagina")
if (edad<12) {
    alert("todavia sos muy pequeño :)");
    console.log("Edad ingresada :" + edad)
} else if (edad<19) {
    alert("Ya sos un Adolescente");
    console.log("Edad ingresada :" + edad)
} else if (edad<35) {
    alert("Juventud eterna")
    console.log("Edad ingresada :" + edad)
} else {
    alert("NUNCA DEJES DE SER NIÑO/A :)")
    console.log("Edad ingresada :" + edad)
}
```



JS – Condicionales – switch

Una sentencia switch permite a un programa evaluar una expresión e intentar igualar el valor de dicha expresión a una etiqueta de caso (case). Si se encuentra una coincidencia, el programa ejecuta la sentencia asociada. Una sentencia switch se describe como se muestra a continuación:



JS - Condicionales - switch

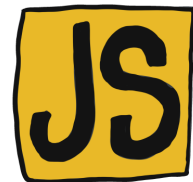
```
var dia = prompt("QUE DIA ES HOY : lunes , martes, miercoles o viernes?? escriba porfavor")

switch (dia) {
  case 'lunes':
    console.log("Hoy es lunes. Se arranca")

    break;
  case 'martes':
    console.log("Hoy es Martes. contodo")

    break;
  case 'miercoles':
    console.log("hoy es Miercoles. cortamos la semana")

    break;
  case 'viernes':
    console.log("Hoy es Viernes. Y tu codigo lo sabe!!")
    break;
  default:
    console.log("Che te cuento que hoy no es ni Lunes , ni martes, ni miercoles, ni viernes :(")
    console.log("Hoy es:" + dia + " Que tengas un gran dia!")
    break;
}
```

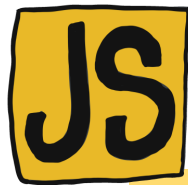




OPERADORES EN JS



OPERADORES LÓGICOS Y RELACIONALES	DESCRIPCIÓN	EJEMPLO
==	Es igual	a == b
===	Es estrictamente igual	a === b
!=	Es distinto	a != b
!==	Es estrictamente distinto	a !== b
<, <=, >, >=	Menor, menor o igual, mayor, mayor o igual	a <= b
&&	Operador and (y)	a && b
	Operador or (o)	a b
!	Operador not (no)	!a



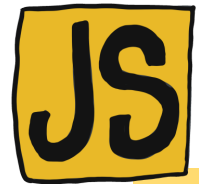


COMBINACIÓN DE OPERADORES



Ante una combinación de operadores AND será requisito para cumplir esa condición que todas las condiciones individuales se cumplan.

En caso de utilizar OR, la evaluación es parcial sobre cada condición y con que una se cumpla, se dará por válida la condición general.





COMBINACIÓN DE OPERADORES



Ya que las expresiones lógicas son evaluadas de izquierda a derecha, a veces es necesario agrupar las operaciones para asegurar que se cumplan como uno lo desea:



No es lo mismo:

```
if( ( mostrado && usuarioPermiteMensajes ) || texto != "test1" ) {
```

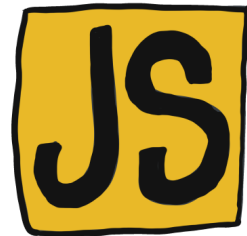
que:

```
if( mostrado && usuarioPermiteMensajes || texto != "test1" ) {
```

JavaScript - Funciones

Las funciones son uno de los pilares fundamentales en JavaScript. Una función es un procedimiento en JavaScript—un conjunto de sentencias que realizan una tarea o calculan un valor. Para usar una función, debe definirla en algún lugar del ámbito desde el cual desea llamarla.

```
function calcularEdad(anioNacimiento) {  
  return 2020 - anioNacimiento;  
}
```



JavaScript – Funciones Declaración

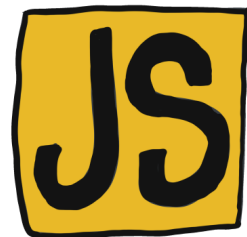
La definición de una función consiste de la palabra clave (reservada) `function`, seguida por:
El nombre de la función (opcional).

Una lista de argumentos para la función, encerrados entre paréntesis y separados por comas (,).

Las sentencias JavaScript que definen la función, encerradas por llaves, { }.

A continuación podemos observar la declaración de la función `calcularEdad`, la cual recibe un parámetro.

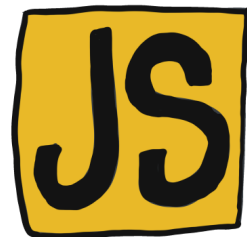
```
function calcularEdad(anioNacimiento) {  
    return 2020 - anioNacimiento;  
}
```





JavaScript – Funciones DRY

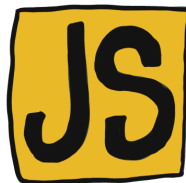
Sigamos siempre el principio DRY (Del inglés, “DO NOT REPEAT YOURSELF”). Es decir, con las funciones volvemos nuestro código modular y no nos repetimos.





challenge

<https://elated-joliot-3bf1a6.netlify.app/>



`<codoa
codo/>`



`<codoa
codo/>`



Maurisandev



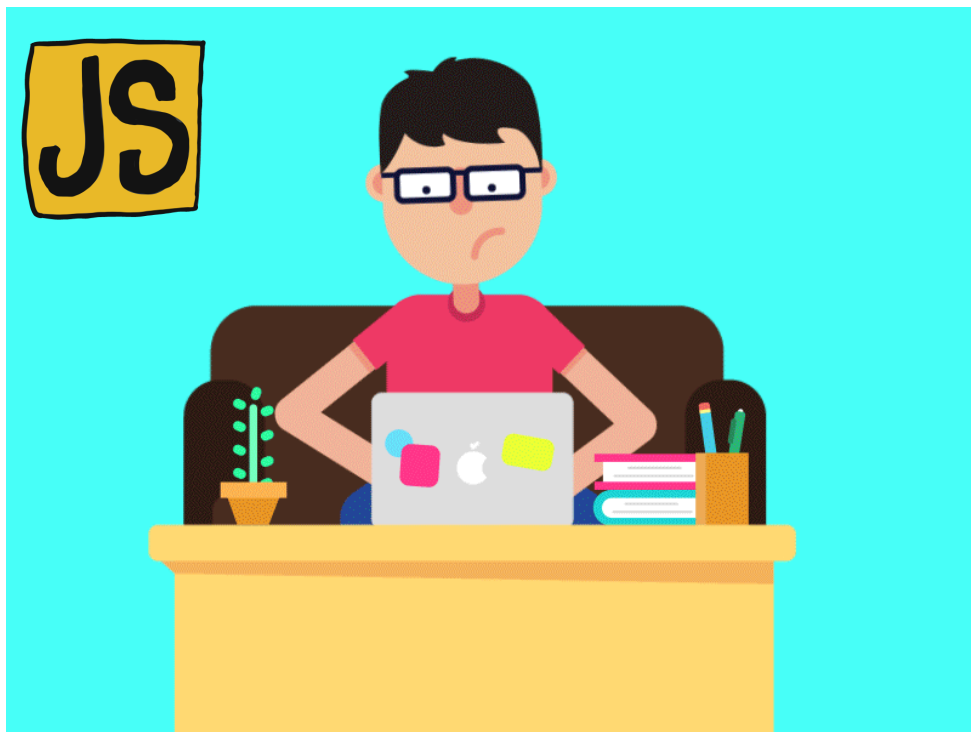
@MauriDeveloper



maurisan4011@gmail.com



MUCHAS GRACIAS!



**NOS VEMOS EL
martes!!**

