educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 2 - Java Web Developer

Módulo 17

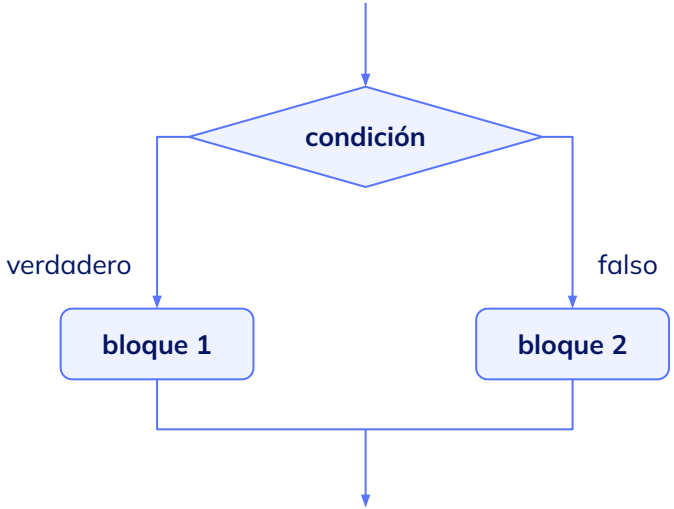
Estructuras de control de flujo

Estructuras de control de flujo

Si solamente se trabaja con operadores y variables, se genera una sucesión de pasos lineales. No se puede **decidir si algo se mostrará o no** según determinadas circunstancias o, en todo caso, **repetir una instrucción varias veces**.

Para realizar estas secuencias más complejas y con múltiples posibilidades se debe trabajar con **estructuras de control de flujo**.

Antes de avanzar, necesitamos recordar algunos tipos de variables y operadores que usaremos en estas **nuevas estructuras**.



Tipo booleano

Los tipos de variables de datos **booleanos** son aquellos conformados únicamente por 2 posibles valores: **`true/false`**.

Un booleano puede ser **definido en forma explícita,** por ejemplo, al asignar un "`true`" a una variable. Recordemos que, por ejemplo, si se genera una ventana de confirmación, se espera que el resultado, según el usuario acepte o no, sea **verdadero/falso.**

```
index.html  ×  codigo.js  ×
var respuesta = confirm('Usted está seguro que quiere continuar?');
```

El resultado será el siguiente:

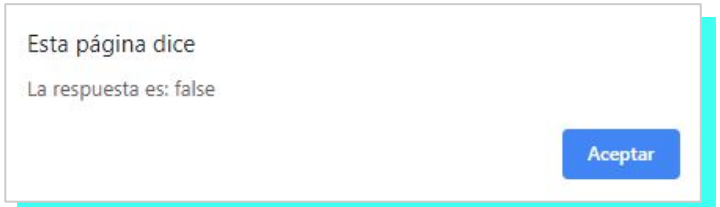
Vamos a mostrar el resultado en una alerta:

index.html × codigo.js ×

```
var respuesta = confirm('Usted está seguro que quiere continuar?');

alert('La respuesta es: ' + respuesta);
```

Si el usuario cancela, el resultado será el siguiente:



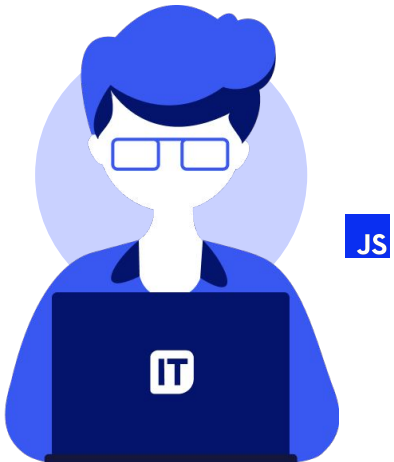
Operadores relacionales

Vamos a recordar que tenemos la posibilidad de trabajar con operadores relacionales, por ejemplo:

>	Mayor a
<	Menor a
>=	Mayor o igual a
<=	Menor o igual a
==	“Igual a” → Utilizado para verificar si 2 valores coinciden
!=	“Distinto de” → Utilizado para verificar si 2 valores NO coinciden
===	“Idéntico a” → Utilizado para verificar si 2 valores coinciden y son del mismo tipo de dato

Si utilizamos **operadores relacionales**, se podría generar el siguiente código, donde se compara si **num1** es menor a **num2**. Es decir, si 10 es menor a 20...

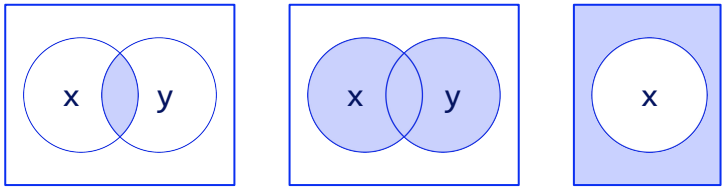
```
1   var num1 = 10;
2   var num2 = 30;
3   var res = num1 < num2;
4   console.log(res);
5
```



Operadores lógicos

También podemos utilizar **operadores lógicos** para generar procesos más complejos donde decimos, por ejemplo, que un **número es igual o mayor a otro** pero también que **debe cumplir o no con otra condición.**

Siempre es interesante **graficar estas situaciones** para entender el proceso de descarte, donde algo termina dando **`true` o `false`**. Veamos los gráficos a la derecha.



x **and** y | x **or** y | **not** x

educación IT
CODE YOUR FUTURE

Estructuras condicionales

Condicionales

Permiten tomar decisiones acerca de **qué sentencias (órdenes) debe ejecutar el programa,** en base a una condición / expresión booleana, es decir si algo es **`true/false`.**

Se puede decir que a diferencia de los programas que veníamos generando, en este caso según el dato de entrada se seguirá una serie de pasos u otra. Por lo tanto, **el resultado o dato de salida dependerá del dato de entrada.**

Condicionales: if

El condicional `if` permite tomar una decisión en base a una condición.

Cuando la condición sea evaluada como verdadera, se ejecutará un bloque con un conjunto de sentencias asociado al **`if`.**

Se comenzará a trabajar en **pseudocódigo.** Esto es esencial para no cometer errores y saber siempre hacia dónde apunta nuestro programa.

```
Inicio

Entero: Edad

Tomar a través de ventana de usuario la Edad

Si(Edad >=20) entonces

Mostrar en alerta “Tu tienes 20 años o más”

Fin Si

Fin Algoritmo
```

A partir del **pseudocódigo** anterior, la sintaxis será similar a la siguiente:

index.html codigo.js

```
if(condicion){

resultado

                    }
```

Ahora, generemos algo real para probar en el código:

```
var edad = 30;

if (edad >= 20) {
    console.log('Tienes 20 años o más');
}
```

El dato de entrada se pedirá y asignará a través de una **ventana de usuario**:

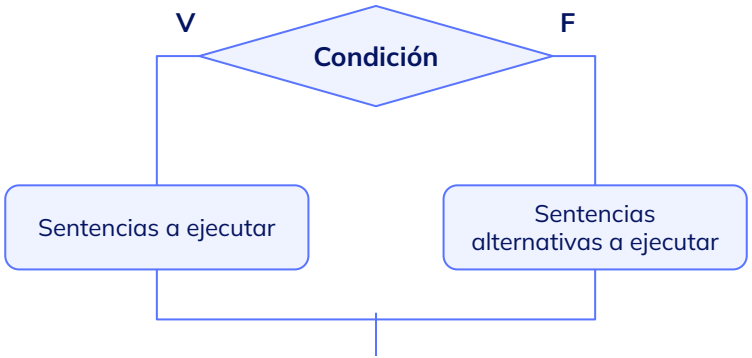
```
var edad = prompt('Ingrese su edad');

if (edad >= 20) {
    console.log('Tienes 20 años o más');
}
```

JS

Condicionales: if, else

En este caso, **se fija una alternativa.** Un algoritmo y un pseudocódigo que representen esta situación sería:




```
Inicio

Entero: Edad

Tomar a través de ventana de usuario la Edad

Si(Edad >=20) entonces

Mostrar en alerta “Tu tienes 20 años o más”

Sino

Mostrar en alerta “Usted tiene menos de 20 años”

Fin Si

Fin Algoritmo
```

Generamos el código del caso anterior en nuestro **codigo.js:**

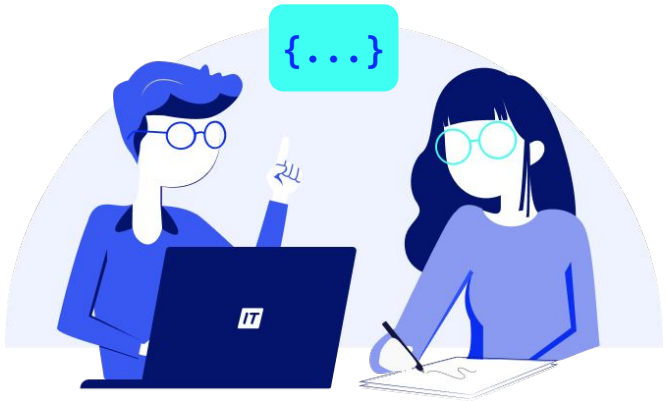
```
var edad = prompt('Ingrese su edad');

if (edad >= 18) {
    console.log('Tienes 18 años o más');
} else {
    console.log('Eres menor de edad');
}
```

Condicionales: if, else, elseif

La estructura **`elseif`** brinda la posibilidad de que **se ejecuten una serie de pasos si una opción (no la primera) es verdadera**.

Veamos en el próximo slide un ejemplo de cómo podría ser nuestro **pseudocódigo**.



```
Inicio
Entero: Edad
Tomar a través de ventana de usuario la Edad
Si(Edad >20) entonces
      Mostrar en alerta “Usted tiene más de 20 años”
Sino
      Si(Edad =20 o Edad > 15)
      Mostrar en una alerta “Usted tiene entre 20 y 15 años”
Sino
      Mostrar en una alerta “Usted tiene menos de 15 años, ¡adiós!”
Fin Si
Fin Algoritmo
```

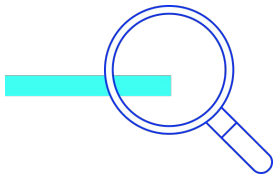
El código se vería de la siguiente forma:

```
1   var edad = prompt('Ingrese su edad');
2
3   if (edad >= 18) {
4       console.log('Tienes 18 años o más');
5   } else if (edad > 13 && edad < 18) {
6       console.log('Eres adolescente');
7   } else {
8       console.log('Eres menor de edad');
9   }
10
```

JS

Revisión

- Repasar el concepto de **estructura de control de flujo**.
- Generar diferentes opciones de **pseudocódigo** para trabajar con **condicionales**.
- Implementar **`if`, `else`, `elseif`**.
- Combinar los **diferentes tipos de output vistos**.
- Implementar los conocimientos en el **Proyecto Integrador**.



educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!