sustantiva

CON•SENTIDO

VARIABLES, EXPRESIONES Y SENTENCIAS CONDICIONALES

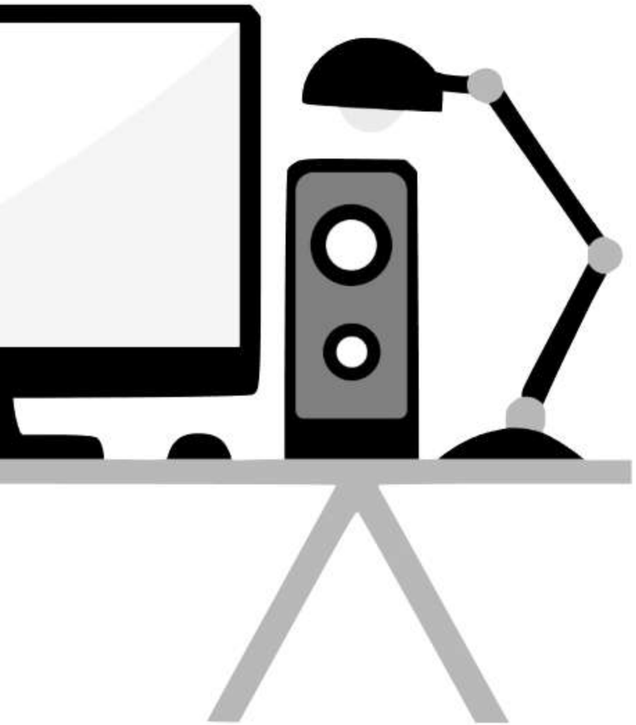
Lección 02

Competencia General

Utilizar variables simples y sentencias condicionales para el control del flujo de un algoritmo que resuelve un problema simple acorde al lenguaje JavaScript.

sustantiva
CON•SENTIDO

Contenidos



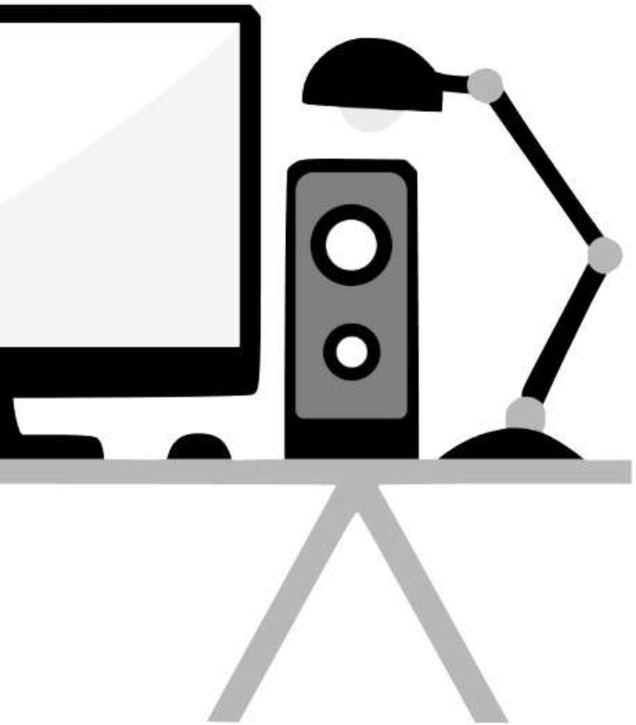
Variables

1. ¿Qué es una variable?
2. Consejos para nombrar una variable
3. Declaración de una variable
4. Inicialización de una variable
5. Variables constantes
6. Tipos de variables
7. Scope de una variable

Operadores y expresiones

1. Operadores aritméticos
2. Procedencia de operadores
3. Operadores de incremento y decremento
4. Operadores de comparación

Contenidos



Cadenas de caracteres

1. Creación de una cadena de caracteres y uso de comillas
2. Concatenación

Sentencias Condicionales

1. Expresiones y operaciones lógicas
2. Manejo de flujo if y else
3. Implementando código a partir de un diagrama de flujo
4. Manejando condiciones de borde

sustantiva
CON•SENTIDO

¿Qué es una variable?

Una variable es un “contenedor” que se le asigna un nombre y sirve para almacenar algún valor. Este valor puede ser cualquier tipo de datos que soporte JavaScript.

```
// Declarando varios tipos de variable
var name = "Pedro" // String
var students = 40 // Number
var countries = ["Venezuela", "Colombia", “Chile"] // Array
var grades = { Carlos: "B", Paula: "A" } // Object
var success = true // Boolean
var nothing = null // null
```

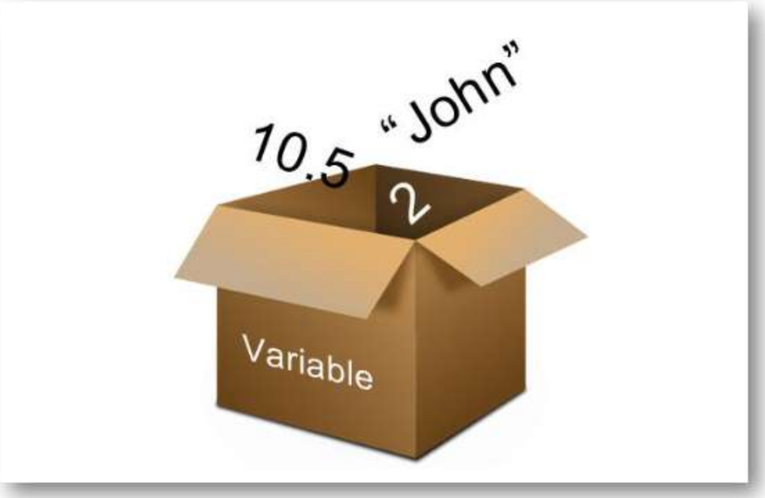
Consejos para nombrar variables

Una variable debe tener un nombre simbólico que represente el contenido de dicha variable, estos nombres en JavaScript deben empezar por una letra, un guión bajo o un símbolo de $.

Nota: Los nombres de variables en JavaScript son “case sensitive” esto quiere decir que se distingue mayúsculas de minúsculas

```
var name = "Pedro" // String
var Name = "Pedro" // String
```

Estas dos variables son distintas a pesar de que sus valores son iguales.



Declaración de una variable

Aunque no es obligatorio, siempre se recomienda declarar variables antes de usarlas en JavaScript. Para realizar una declaración de una variable se escribe una palabra clave que identifica el tipo de declaración seguida de su nombre simbólico.

```
// Declarando variables
var nombre;
let autos;
```

Existen tres tipos de declaraciones en JavaScript; var, let y const.

Declaración de una variable

var
Una variable tipo var puede cambiar su valor pero su scope es local

let
Una variable tipo let puede cambiar su valor, pero es una variable que solo existe en el bloque donde fue declarada.

const
Una variable tipo const NO puede cambiar su valor en ningún momento del ciclo de la aplicación.

Característica	const	let	var
Alcance global	NO	NO	SI
Alcance en una función	SI	SI	SI
Alcance en un bloque	SI	SI	NO
¿Se puede modificar?	NO	SI	SI

Inicialización de una variable

Una vez que una variable es declarada, es posible inicializarla en la misma línea o después de declararla.

La recomendación es siempre declarar primero la variable y luego inicializarla, esto para evitar asignar valores a variables inexistentes.

```
// Declarando variables
var nombre;
let autos;

// Inicializando variables
nombre = "Pedro“;
autos = 3;
```

Variables constantes

Una variable constante es de solo lectura en un bloque establecido, por tanto, no será posible cambiar su valor mediante asignación.

Para utilizar una variable constante debemos declararla e inicializarla en la misma línea.

```
// Declarando variables constantes
const pi = 3.14159;
const e = 2.71828;
```

Tipos de variables

```
// declarando una nueva variable
let autos;

// inicializando la variable tipo String
autos = “Mercedes Benz”;

// mutando variable a tipo Number
autos = 3;
```

JavaScript es un lenguaje dinámicamente tipado, es decir, cuando declaramos variables y estas no incluyen tipado, JavaScript por detrás le asigna el tipado de acuerdo con el tipo de dato que se ingresa. Además, una variable puede ser de un tipo en un instante y más tarde puede mutar en otro tipo según el valor de entrada.

sustantiva
CON•SENTIDO

Tipos de variables

Las variables en JavaScript se dividen en tres categorías:

- Primitivas (String, Number, Boolean): hace referencia a un tipo de dato que **NO posee métodos ni propiedades**, además, los valores asignados a estos tipos **son inmutables**.

- Compuestas y referenciales (Object, Array, Function): son variables que poseen métodos y propiedades, hacen referencia a un objeto o una función.

- Especiales (Undefined, Null): este tipo de variables tienen un valor no definido, por ende son variables declaradas pero no inicializadas.

Tipos de variables

Las variables tipo String

Se usan para representar datos textuales, estas variables se inicializan usando **comillas simples o dobles**, pero ambas deben coincidir al inicio y cierre.

Nota: Se pueden incluir comillas dentro de los String, siempre y cuando estas NO coincidan con el tipo de comillas externas.

```
// Declarando variables
let pregunta;
let respuesta;

// Inicializando variables
pregunta = “¿Es la variable tipo ‘String’?”;
respuesta = ‘Si es tipo “String”’;
```

Tipos de variables

Las variables tipo Number

Esta variable engloba todos los números (enteros, racionales, reales, etc) en una sola variable. Además, se incluyen tipos especiales para casos concretos como **infinity, -infinity y NaN** (not a Number)

```
// Declarando variables
let a;
let b;
let c;
let d;

// Inicializando variables
a = 15;
b = 20.6;
c = 15 / 0; // c = infinity
d = “texto” / 4; // d = NaN
```

Tipos de variables

Las variables tipo Boolean

Este tipo de variable solo puede almacenar un valor lógico (Verdadero o Falso).

```
// Declarando variables
let a;
let b;

// Inicializando variables
a = true;
b = false;
```

Tipos de variables

Las variables tipo Object

Este tipo de variable puede contener un conjunto de variables en su interior, recordemos que las variables almacenadas en los objetos cumplen las mismas reglas de nombres de variables.

Cada variable debe estar separada por una coma (,) al interior del objeto. Estas variables usualmente se le llaman propiedades del objeto

```
// Declarando variables
let auto;

// Inicializando variables
auto = {
  modelo: “BMW X3”,
  color: “Rojo”,
  patente: “XL TR 96”
};
```

Tipos de variables

Las variables tipo Array

Este tipo de variables se utiliza para almacenar una colección de valores agrupadas en una sola variable, a diferencia de un objeto esta no tiene propiedades, solo valores agrupados.

```
// Declarando variables
let colores;

// Inicializando variables
colores = [ “rojo”, “amarillo”, “azul” ];
```

Tipos de variables

La función typeof

Esta es una función utilizada para obtener el tipo de dato que contiene una variable.

```
// Declarando variables
let variable;

// Inicializando variables
variable = “Este es un String”;

// Usando la función typeof
Typeof variable;
  |“String”
```

Scope de una variable

Cuando declaras una variable fuera de una función, esta se denomina **variable global**, porque está disponible para cualquier código en el documento.

Por el contrario si una variable es declarada dentro de una función, se le denomina **variable local**, porque está disponible sólo dentro de esa función donde fue creada.

```
// Variable global
let colores;

function (){
  let variable; // Variable local
}
```

Operadores aritméticos

JavaScript cuenta con los operadores aritméticos estándar:

- El operador suma (+), produce la suma de los operandos numéricos
- El operador de resta (-), resta los dos operandos.
- El operador división (/) produce el cociente donde el operando izquierdo es el dividendo y el derecho es el divisor
- El operador multiplicación (*) produce el producto de los dos operandos.
- El operador resto (%) devuelve el módulo de una operación de división.

```
// Declarando variables
let a;
let b;
// Inicializando variables
a = 2;
b = 2;
// suma
a + b; // resultado 4
// resta
a - b; // resultado 0
// división
a / b; // resultado 1
// multiplicación
a / b; // resultado 1
// resto
a % b; // resultado 0
```

Procedencia de operadores

Si una expresión tiene más de un operador, el orden de ejecución se define por su precedencia, en otras palabras, existe un orden implícito entre los operadores.

El orden establecido es:
multiplicación > división > resto > suma > resta

```
// Declarando variables
let a;
let b;
let c;

// Inicializando variables
a = 2;
b = 3;
c = 4;

// Operaciones
a * b / c; // retorna 1.5
c * b / a; // retorna 6
```

Operadores de incremento y decremento

En JavaScript tenemos unos operadores aritméticos dedicados a sumar o restar un uno a una variable numérica.
La asignación de la operación de suma o resta dependerá de la posición del incremento o decremento.

- Si usamos el incremento o decremento como prefijo a la variable, entonces la asignación será previa al retorno.

- Si usamos el incremento o decremento como sufijo, entonces, primero retorna el valor y luego lo asigna.

```
// Declarando variables
let a;
let b;

// Inicializando variables
a = 3;
b = ++a; // a y b valen 4

a = 3;
b = a++; // a vale 3 y b vale 4
```

Operadores de comparación

Los operadores de comparación se utilizan para determinar la igualdad o diferencia entre las variables o valores, estas retornan un boolean (verdadero o falso). Los operadores de comparación en JavaScript son los siguientes:

- Igual (==)
- Distinto (!=)
- Mayor que (>)
- Mayor o igual que (>=)
- Menor que (<)
- Menor o igual que (<=)

```
// Declarando variables
let a;
// Inicializando variables
a = 3;
// Operadores de comparación
a == 3; // devuelve true
a != 3; // devuelve false
a > 1; // devuelve true
a >= 3; // devuelve true
a < 3; // devuelve false
a <= 1; // devuelve false
```

Cadena de caracteres

Las cadenas de caracteres se pueden crear como primitivas tipo String a partir de un texto literal.
Las cadenas son útiles para almacenar datos que representan texto, además, estas incluyen operaciones como *length* que retorna la cantidad de caracteres o *chartAt()* que accede al carácter en la posición indicada.

```
// uso de comillas en cadena de caracteres
const string1 = "Una cadena primitiva";
const string2 = 'También una cadena primitiva';
const string3 = `Otra cadena primitiva más`;

“cadena”.charAt(1) // devuelve “a”
“cadena”.length // devuelve 6
```

Concatenación

En JavaScript es posible unir dos cadenas de caracteres mediante el operador de suma (+), al unir dos cadenas el retorno de esta será una sola cadena que contiene las ambos textos.

```
// Declarando variables
let nombre;
let apellido;
// Inicializando variables
nombre = 'Ignacio';
apellido = 'Sepúlveda';

// Operación de concatenación
console.log(nombre + " " + apellido);
//retorna "Ignacio Sepulveda"
```

Expresiones y operaciones lógicas

En JavaScript se dispone de operadores lógicos que comparan valores y retornan booleanos.

Un operador lógico compara dos expresiones, las evalúa, y retorna dependiendo del operador.

Existe tres operadores:

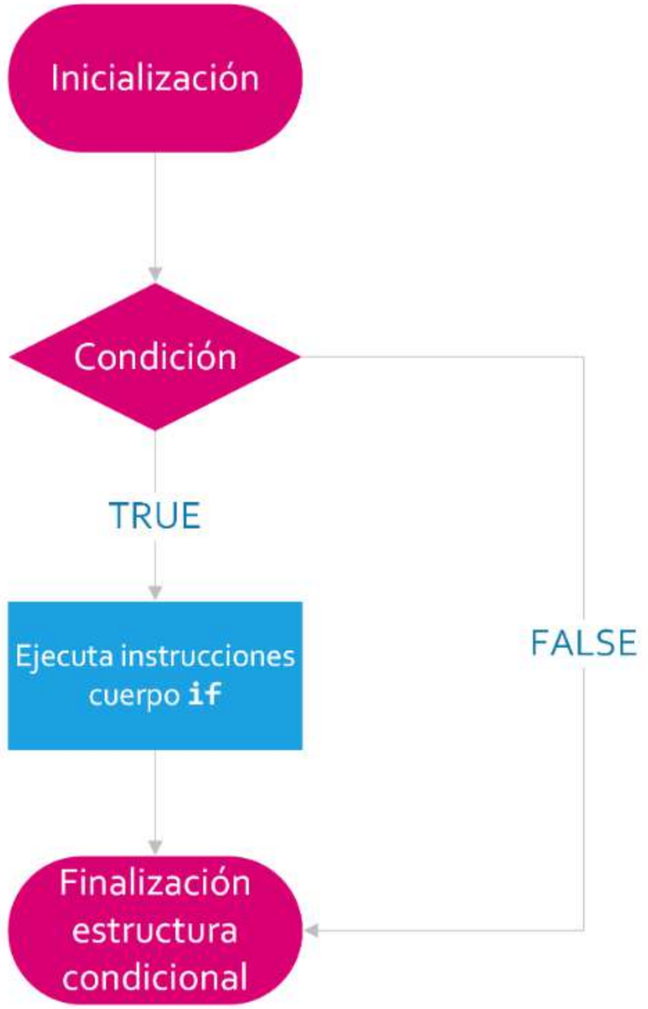
- AND (&&)
- OR (||)
- NOT (!)

```
true && true //devuelve el segundo valor, verdadero
true && false //devuelve el segundo valor, falso
false && false //devuelve el primer valor, falso

true || true //devuelve el primer valor, verdadero
true || false //devuelve el primer valor, verdadero
false || false //devuelve el segundo valor, falso

!true  // devuelve false
!false // devuelve true
```

Flujo if else

La instrucción *if* que significa “si” permite evaluar una expresión y sobre la base del resultado (verdadero o falso) ejecutar una instrucción (bloque de código o otra.

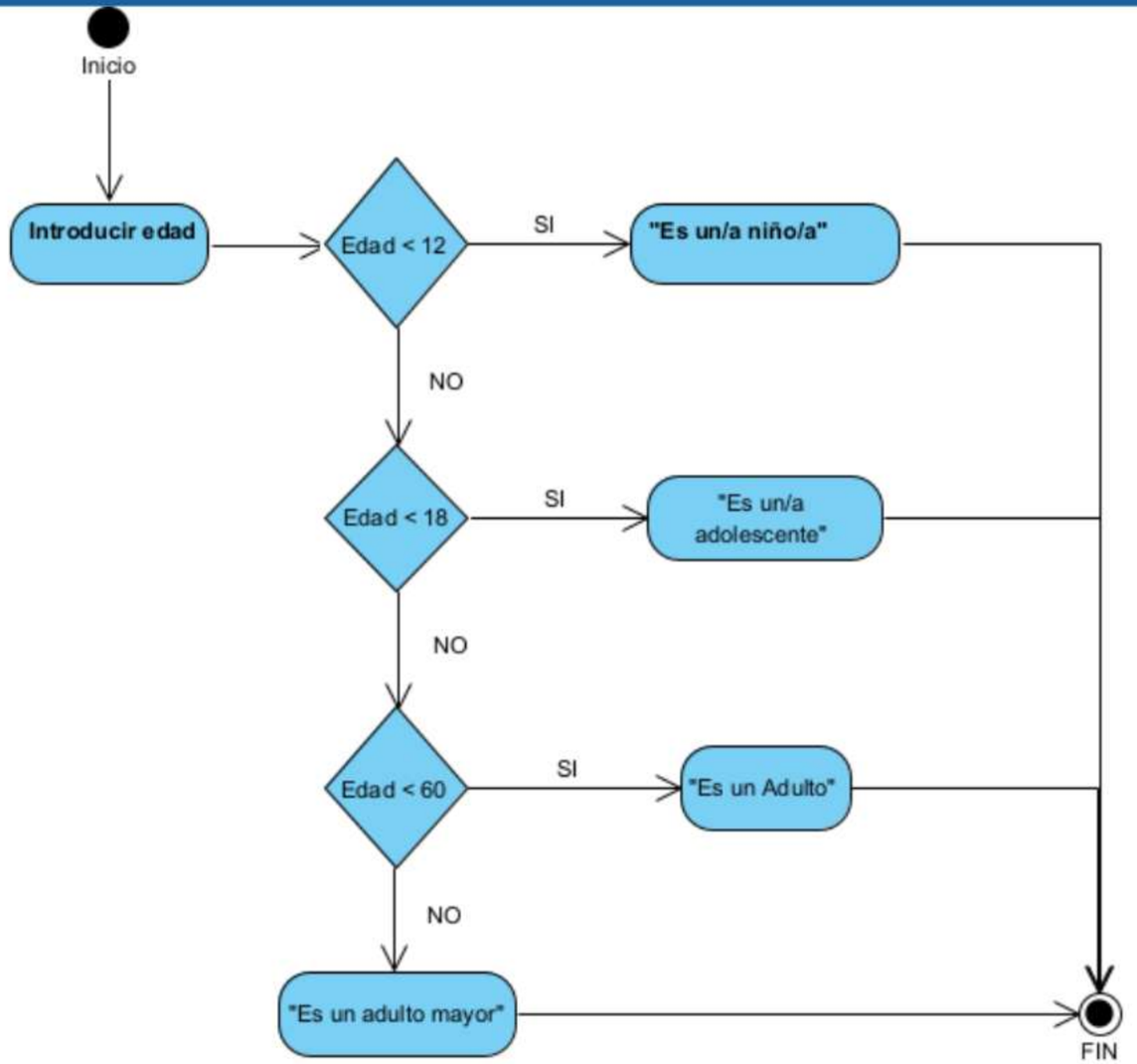
```
if (condición){
  //instrucción condición verdadera
}else{
  //instrucción condición falsa
}
```



Código a partir de un diagrama de flujo

Práctica de código

A partir del siguiente diagrama, determinar el código en JavaScript.

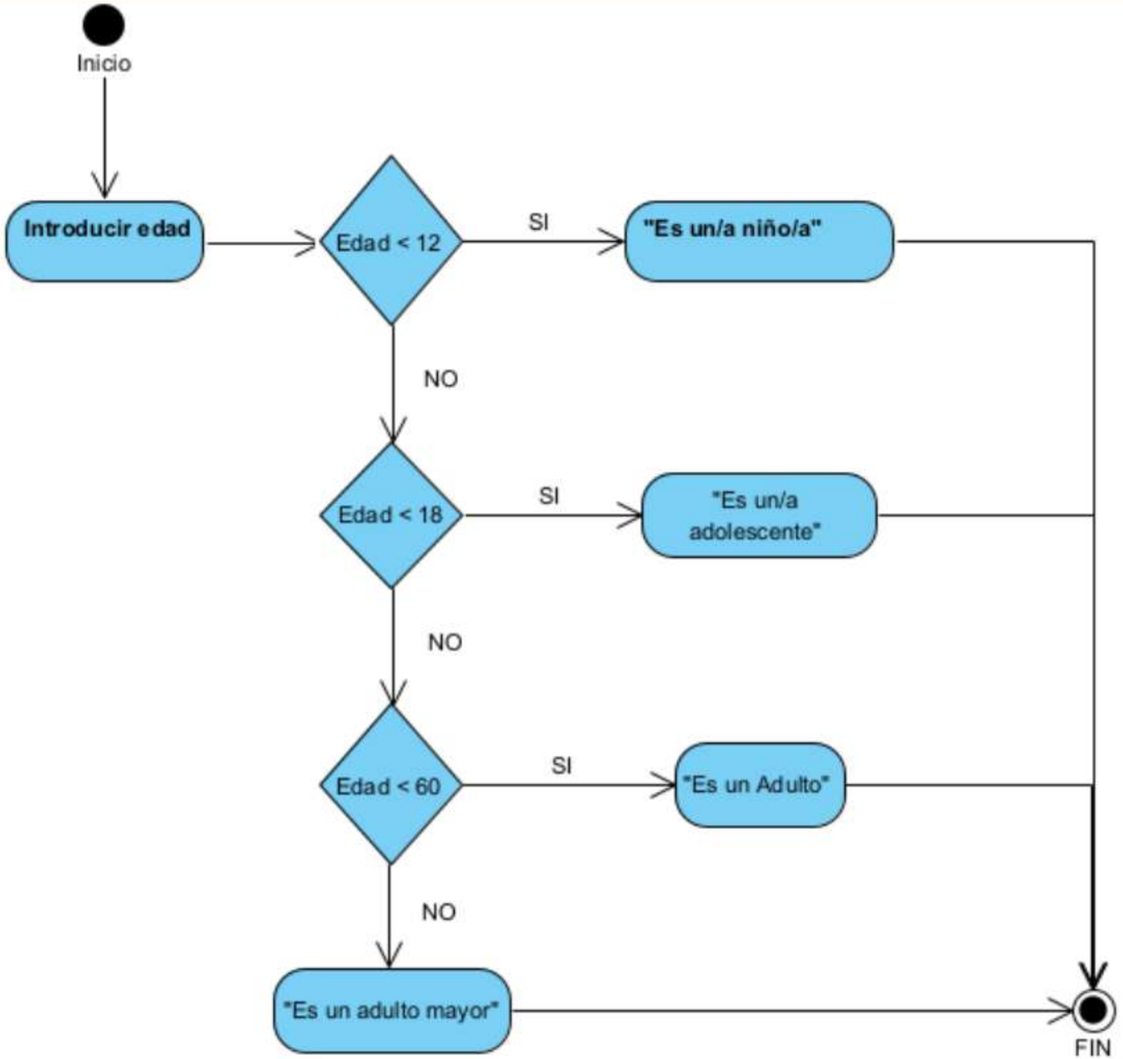


Código a partir de un diagrama de flujo

Práctica de código

A partir del siguiente diagrama, determinar el código en JavaScript.
SOLUCIÓN:

```
if (edad < 12){
  return “es un/a niño/niña”;
}else if (edad < 18){
  return “es un/a adolecente”;
}else if (edad < 60){
  return “es un adulto”;
}else{
  return “es un adulto mayor”;
}
```



Manejando condiciones de borde

Analiza el ejercicio anterior, ¿Qué sucede si la variable edad está vacía o nula?

Cada vez que se codifique un algoritmo, siempre se debe considerar el dominio total de las entradas, para manejar el dominio total de las salidas.

Intenta identificar todas las posibles entradas que puede tener la variable edad en el ejercicio anterior.

Actividad

Usando todo lo visto en esta lección, realiza en código JavaScript un juego de piedra papel o tijera de dos jugadores.

Al finalizar cada jugada deberá imprimir por pantalla las estadísticas de cuantas rondas ganadas para el jugador 1, jugador 2 y empates.


sustantiva
CON•SENTIDO