NODE JS

Clase 10: Estructuras de control / While/ Do-while

adn POTENCIÁ TU FUTURO

Nación Servicios

IT Patagonia

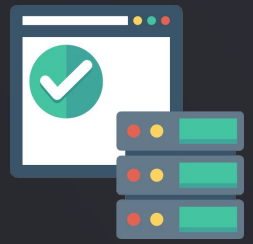
Codeki™


Estructuras de Control

01

Estructuras de Control

JavaScript ofrece varias estructuras de control para ejecutar lógica condicional y bucles.



01

Estructuras de Control

If-Else

La estructura if-else permite ejecutar un bloque de código si se cumple una condición y otro bloque de código si no se cumple.

01

Nación Servicios

Estructuras de Control

Switch

La estructura switch se utiliza para seleccionar una opción de entre varias opciones posibles.

01

Estructuras de Control

For loop

El bucle for se utiliza para repetir un bloque de código un número específico de veces.

01

Estructuras de Control

While loop

El bucle while se utiliza para repetir un bloque de código mientras se cumpla una condición.

01

Estructuras de Control

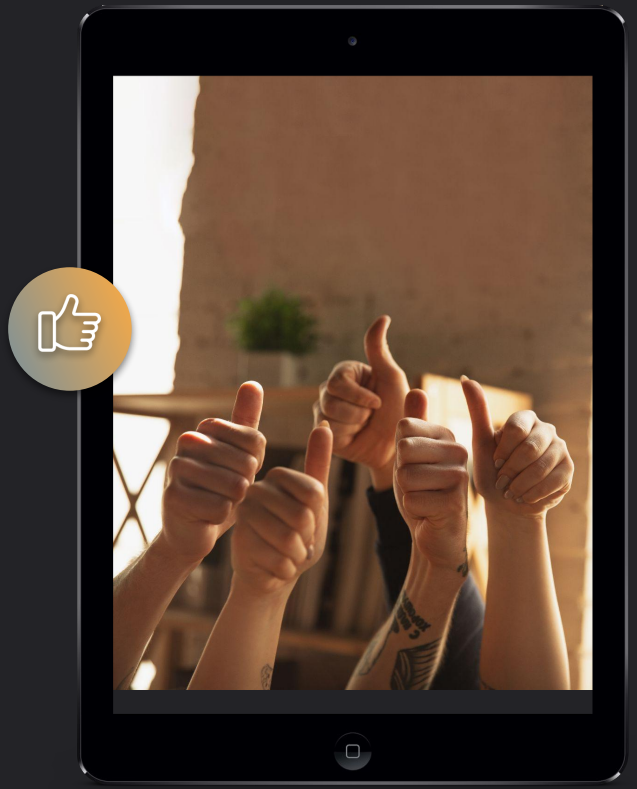
Do-While loop

El bucle do-while es similar al bucle while, pero se ejecuta al menos una vez antes de verificar la condición.

02

Estructuras de control

A continuación vamos a ver los conceptos mencionados anteriormente de manera más extensa y con ejemplos de código para que puedan practicar mientras estudian.



02

Estructuras de control

IF – ELSE

- Las declaraciones condicionales se utilizan para realizar diferentes acciones basadas en diferentes condiciones
- Muy a menudo, cuando escribe código, desea realizar diferentes acciones para diferentes decisiones

Nación Servicios

Estructuras de control

IF – ELSE

- Se utiliza if para especificar un bloque de código que se ejecutará, si una condición especificada es verdadera
- Se utiliza else para especificar un bloque de código a ejecutar, si la misma condición es falsa
- Se utiliza else if para especificar una nueva condición para probar, si la primera condición es falsa.
- Se utiliza switch para especificar muchos bloques alternativos de código para ejecutar.

IF – ELSE

Nación Servicios

```
if ( condicion ){
    // Bloque de código
}
```

IF – ELSE

Nación Servicios

```
if( condicion ){
    // Bloque de código
} else {
    // Bloque de Código cuando la condición es falsa
}
```

IF – ELSE

```
if( condicion1 ) {
    // Bloque de código
} else if ( condicion2 ) {
    // Bloque de código cuando la condicion 1 es falsa y la 2 es verdadera
} else {
  // Bloque de Código cuando la condición 2 es falsa
}
```

IF – ELSE

Ejemplo, comparación entre variables.

```
var numeroA = 30;
var numeroB = 30;

    if( numeroA >= numeroB ){
        console.log('Numero A es mayor a numero B ');
    } else {
        console.log('Numero B es mayor que numero A');
    }
```

IF – ELSE

Ejemplo, comparación entre variables con valores buleanos.

```
var boolean1 = false;
var boolean2 = true;
    // Compuerta Or
    if ( boolean1 || boolean2 ) {
        console.log('|| Condición es True');

    } else {
        console.log('|| Condición es False');
    }
    // Compuerta AND
    if( boolean1 && boolean2 ){
        console.log('&& Condición es True');
    } else {
        console.log('&& Condicón es False');
    }
    // Compuerta Not
    console.log('Boolean Negado :'+(!boolean1));
```


While

03

WHILE

Los bucles pueden ejecutar un bloque de código mientras se cumpla una condición específica.

03

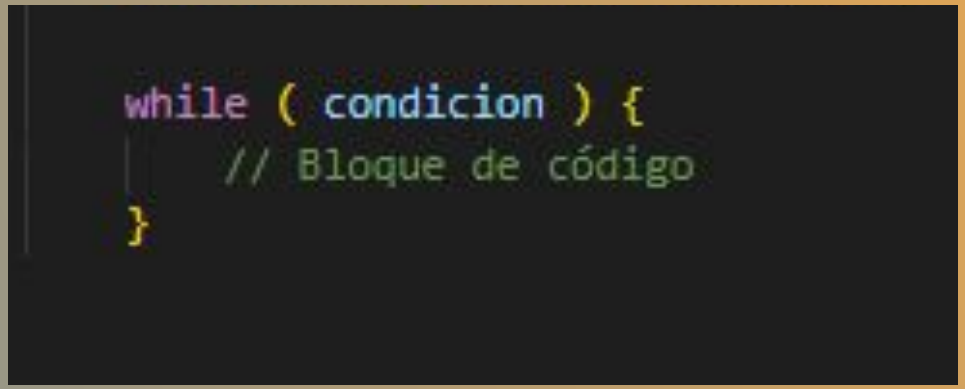
Nación Servicios

WHILE

El ciclo while

El ciclo while recorre un bloque de código siempre que una condición específica sea verdadera.

WHILE



WHILE

```
while ( condicion ) {
    // Bloque de código
    /* Se ejecutará el bloque de código
       mientras condicion sea verdadero ,
       en caso contrario saldrá del ciclo
    */
}
```

03

WHILE

Es recomendable que además de la condición también se tenga en cuenta un número máximo de bucles, dado que si la condición nunca llega a ser falsa , el programa caerá en ciclo infinito , y esto dará problemas.

WHILE

```
var contadorCiclos = 0;
var condicion = true;
    while ( contadorCiclos<=1000 && condicion ) {
        /*
        Bucle de código
                Cada ciclo en que condición sea 'true' y contadorCiclo sea<= 1000
                se incrementará en 1 contadorCiclo
        */
       contadorCiclos++;
    }
```

03

WHILE

Escribir un algoritmo que permita leer datos numéricos enteros hasta que aparezca un valor igual a cero, que calcule y muestre la suma de los datos leídos.

WHILE

```
// 1) Leer datos
// 2) Calcular suma de todos
// 3) Mostrar datos

    // 1.1) Leer numero
    // 1.2) Mientras numero sea distinto de cero
    // 1.3) Leer numero

    // 2.1) for numero 1 hasta N
    // 2.2) sumar números

    // 3.1) Mostrar suma de números
```

WHILE

```
var numeros = [];
var suma = 0;
var contador = 0;
    numeros[contador] = parseInt(prompt('Ingrese Número:'+(contador+1)));

    while(contador <= 30 && numeros[contador]!== 0){
        contador++;
        numeros[contador] = parseInt(prompt('Ingrese Número:'+(contador+1)));
    }

    for( let i=0; i<=contador ; i++){
        suma = suma + numeros[i];
    }

    console.log('La suma total es -> '+suma);
```

WHILE

Nación Servicios

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script src="./app10.js"></script>
</body>
</html>
```


Do - While

04

DO-WHILE

El bucle do-while es una estructura de control en JavaScript que ejecuta un bloque de código al menos una vez y luego repite la ejecución mientras se cumpla una condición.

DO-WHILE

Nación Servicios

```
do {
    // Código a ejecutar al menos una vez
} while (condición);
```

04

DO-WHILE

La condición se evalúa después de ejecutar el bloque de código. Si la condición es verdadera, se repite la ejecución del bloque de código. Si la condición es falsa, el bucle se detiene y la ejecución continúa con la siguiente línea de código después del bucle.

DO-WHILE

Nación Servicios

```
let contador = 1;

do {
    // Bloque de código
  console.log(contador);
  contador++;
} while (i <= 5);
```

DO-WHILE

Nación Servicios

```
let password;
let attempts = 0;

do {
  password = prompt('Introduce la contraseña:');
  attempts++;
} while (password !== 'secreto' && attempts < 3);

if (password === 'secreto') {
  console.log('Contraseña correcta. Acceso permitido.');
} else {
  console.log('Has excedido el número de intentos permitidos. Acceso denegado.');
}
```

04

DO-WHILE

En este ejemplo, el bucle do-while se utiliza para solicitar al usuario una contraseña. El bucle se repetirá mientras la contraseña ingresada no sea igual a "secreto" y el número de intentos sea menor a 3.


Parte Asincrónica

05

Ejercicio 1

Se desea analizar cuántos autos circulan con patente que tiene numeración par y cuántos con numeración impar en un día.
Le solicitan escribir un algoritmo que permita ingresar la terminación de la patente (entre 0 y 9) hasta ingresar -1 e informar cuántas se ingresaron con numeración par y cuántas con numeración impar

05

Ejercicio 2

Ingresar un número n, validar que sea positivo.

Luego:

a) Mostrar los primeros n números impares hasta el número ingresado.

Ejemplo:

- Si se ingresa 5, se debe mostrar 1 3 5
- Si se ingresa 8, se debe mostrar 1 3 5 7
- Si se ingresa -5, se debe pedir otro número.

b) Informar la suma de esos números impares.

¡Muchas gracias!