**Aplicación de Tecnologías Web**

Tema n.° 5

Estructuras en JavaScript

**Índice**

**Pág.**

[Introducción 3](#_Toc161303348)

[1.5. Estructuras de control en JavaScript 4](#_Toc161303354)

[1.5.1. La cláusula If 4](#_Toc161303357)

[1.5.2. La cláusula switch 5](#_Toc161303358)

[1.5.3. Estructuras iterativas en JavaScript 6](#_Toc161303359)

[Recursos complementarios 12](#_Toc161303360)

[Referencias 12](#_Toc161303361)

[Autoevaluación 13](#_Toc161303362)

# Introducción

En el mundo de la programación, la capacidad de dirigir el flujo de ejecución de un programa es esencial para construir aplicaciones dinámicas y eficientes. JavaScript, como lenguaje de programación versátil y ampliamente utilizado en el desarrollo web, ofrece poderosas herramientas conocidas como Estructuras de Control e Iterativas, que permiten a los desarrolladores tomar decisiones y repetir tareas de manera efectiva.

**Estructuras de Control**

Las Estructuras de Control en JavaScript son mecanismos que determinan el orden en que se ejecutan las instrucciones en un programa. La cláusula if y la cláusula else son fundamentales para realizar decisiones condicionales, permitiendo que ciertas acciones se ejecuten solo cuando se cumple una condición específica. La cláusula switch proporciona una manera eficiente de manejar múltiples casos, simplificando el código en comparación con múltiples instrucciones if-else anidadas.

**Estructuras Iterativas**

Por otro lado, las Estructuras Iterativas, o bucles, permiten la repetición de bloques de código, facilitando la manipulación de datos y la ejecución de tareas similares de manera eficiente. El bucle for es ideal para iteraciones con un número conocido de repeticiones, mientras que el bucle while es útil cuando la condición de repetición se evalúa antes de la ejecución. El bucle do-while garantiza que el bloque de código se ejecute al menos una vez, independientemente de la condición inicial.

La combinación efectiva de estas estructuras proporciona a los desarrolladores el control necesario para adaptar el comportamiento de sus programas a diversas situaciones. Al comprender y dominar estas herramientas, los programadores pueden mejorar la legibilidad del código, optimizar la eficiencia y resolver problemas de manera más efectiva.

En esta exploración de las Estructuras de Control e Iterativas en JavaScript, descubriremos cómo estas herramientas fundamentales se entrelazan para formar la columna vertebral de la lógica de programación en este lenguaje



# Estructuras de control en JavaScript



Muy a menudo, es necesario que una parte del código JavaScript escrito se ejecute cuando una o varias condiciones se cumplan y que no se ejecute en caso contrario. Es decir, tendremos código que se ejecutará de forma condicional. Para poder realizar este tipo de sentencias, JavaScript dispone de diferentes estructuras de control del flujo del código.

## La cláusula If

La sentencia más sencilla y conocida es la que comienza por la cláusula if, que seguida de una condición lógica (que se evaluará como true o false) define un bloque de código que se ejecutará únicamente si la condición es satisfecha. Un ejemplo sería:

**Figura 1.**

Estructura if



La consola solo mostrará la frase si la variable edad tiene un valor estrictamente menor que 78.

Junto a la cláusula if tenemos otras dos que suelen acompañarla casi siempre. Son else if y else. Si estamos barajando varias opciones en nuestro programa, podemos hacer uso de else if para tratar los diferentes escenarios. Por último, si ninguna de las condiciones anteriores se cumplió, podemos hacer uso de else para indicar las acciones a realizar en el caso de que ninguna de las condiciones anteriores se diese. Observemos un ejemplo:

**Figura 2.**

Estructura if-else



Al igual que en otros lenguajes de programación, en JavaScript también podemos hacer uso del **operador ternario** para comprobar una condición y actuar en consecuencia de su cumplimiento (o no) en una única línea. Su sintaxis, al igual que en otros lenguajes, es la siguiente:

**Figura 3.**

Operador ternario



Un ejemplo sería:

**Figura 4.**

Código del operador ternario



## La cláusula switch

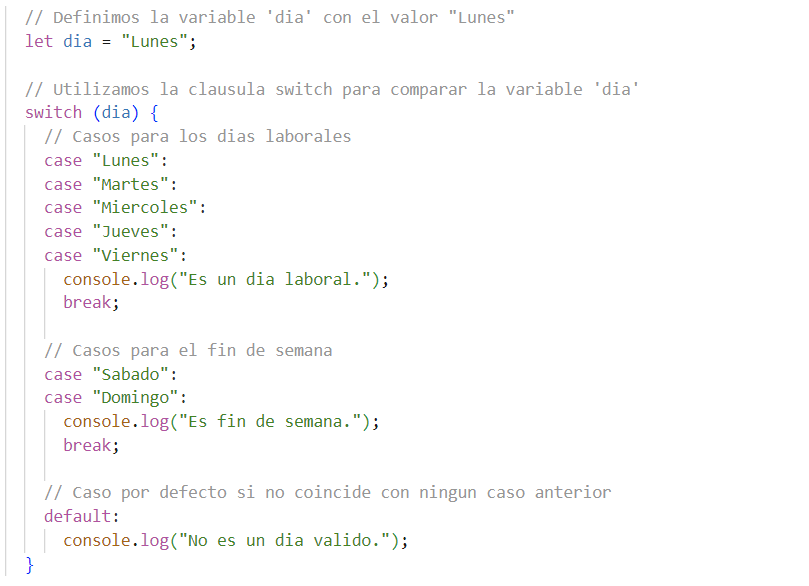
La cláusula switch nos permite construir disyuntivas más complejas. Es una estructura de control que se utiliza para realizar múltiples comparaciones de una expresión y ejecutar bloques de código según el valor coincidente. Es una alternativa más limpia y legible a múltiples declaraciones if-else anidadas.

El funcionamiento de la cláusula switch es el siguiente:

* La expresión dentro del switch se evalúa una vez.
* Se compara el valor resultante con cada caso.
* Si se encuentra una coincidencia, se ejecuta el bloque de código correspondiente y se sale del switch usando la palabra clave break.
* Si no se encuentra una coincidencia, se ejecuta el bloque de código asociado al caso default (si está presente).

**Figura 5**

Días de la Semana.



## Estructuras iterativas en JavaScript

Una estructura iterativa nos permite construir bucles en los cuáles es posible repetir una o varias líneas de código. Estudiaremos por tanto los bucles for y while.

**Bucles while**

Los bucles while ejecutan las instrucciones que haya dentro de un bloque mientras la condición enunciada se cumpla. Un ejemplo:

**Figura 6.**

Bucle while.



Si ejecutamos este código en la consola de JavaScript, veremos que el resultado que se imprime por pantalla es el siguiente:

**Figura 7**

Ejecución del bucle while



Nótese que se ha hecho uso de template literals para intercalar cadenas de texto con el valor de las variables con las que estamos trabajando. Es muy importante que el código que se encuentre dentro del bloque del while pueda dar lugar a un cambio en la condición que se está evaluando. De no ser así, se podría formar un bucle infinito, algo que casi con toda seguridad será un comportamiento no deseado.

Existe una variación del bucle while llamada do while. La diferencia entre ambas es que la versión do while se ejecutará siempre una vez como mínimo, ya que la condición del while se comprobará al final del bloque, mientras que esto no ocurre con la versión que acabamos de ver. Un ejemplo sería el siguiente:

**Figura 8**

Bucle do-while



Si ejecutamos este código en la consola de JavaScript, veremos que el resultado que se imprime por pantalla es el siguiente:

**Figura 9**

Ejecución del bucle do-while.



**Bucles for**

Se trata de otra estructura diferente para poder realizar bucles. Su sintaxis es bastante distinta a la de la opción anterior:

**Figura 10**

Bucle for.



Un bucle for tiene cuatro partes muy diferenciadas:

**Figura 11**

Partes del bucle for.



Al igual que ocurría con los bucles while, es necesario definir bien la condición, el valor inicial de la variable de estado y cómo va a modificarse su valor en cada iteración para evitar dar lugar a bucles infinitos no deseados. Veamos un ejemplo de este tipo de bucles:

**Figura 12**

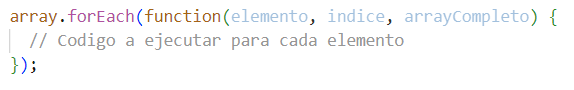
Bucle for.



**Bucles forEach**

La cláusula forEach es una función integrada en JavaScript que proporciona una forma elegante y funcional de recorrer cada elemento de un array. Facilita la ejecución de una función específica en cada elemento del array, simplificando tareas como la manipulación de datos, filtrado y más.

La sintaxis básica de la cláusula forEach es la siguiente:

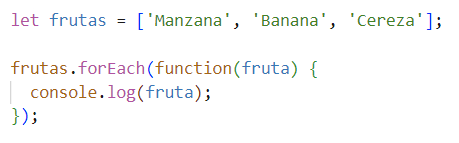


* elemento: Representa el valor del elemento actual durante la iteración.
* indice: Representa la posición del elemento actual en el array.
* arrayCompleto: Es una referencia al array sobre el cual se está iterando.

Veamos algunos ejemplos prácticos para ilustrar el uso de la cláusula forEach:

**Figura 13**

Imprimir cada elemento de un array



# Recursos complementarios

* Documentación de JavaScript

<https://developer.mozilla.org/es/docs/Web/JavaScript>

# Referencias

Brown, A. R., & Wilson, S. (2013). *JavaScript Testing with Jasmine: JavaScript Behavior-Driven Development.* O'Reilly Media.

Crockford, D. (2008)*. JavaScript: The Good Parts*. O'Reilly Media.

Duckett, J. (2011). *JavaScript & jQuery*. John Wiley & Sons Books.

Flanagan, D. (2011). *JavaScript: The Definitive Guide: Activate Your Web Pages (6th ed.).* O'Reilly Media.

Freeman, A., & Robson, S. (2012). *Head First JavaScript Programming: A Brain-Friendly Guide.* O'Reilly Media.

McPeak, J., Kuksenok, K., & Robinson, C. (2012). *Learning Node: Moving to the Server-Side*. O'Reilly Media.

Osmani, A. (2013). *Developing Backbone.js Applications*. O'Reilly Media.

Resig, J., Bibeault, B., & Marconcini, J. (2013). *Secrets of the JavaScript Ninja*. Manning Publications.

# Autoevaluación

1. **¿Cuál de las siguientes palabras clave se utiliza para iniciar una estructura de control if en JavaScript?**

start

begin

if

initiate

1. **¿Qué declaración se utiliza para manejar múltiples casos en una estructura de control en JavaScript?**

for

switch

while

case

1. **En una estructura if-else, ¿qué bloque de código se ejecuta si la condición es falsa?**

Bloque if

Bloque else

Ninguno

Ambos bloques

1. **¿Cuál es la finalidad de la estructura else if en JavaScript?**

Para manejar errores

Para agregar condiciones adicionales

Para finalizar la ejecución del programa

Para iniciar un bucle

1. **¿Cuál de las siguientes opciones representa una combinación de una estructura de control y una estructura iterativa?**

if

while

for

switch

1. **¿Qué estructura de control es adecuada para ejecutar un bloque de código mientras se cumpla una condición?**

for

if

while

switch

1. **¿Cuál de los siguientes bucles se ejecuta al menos una vez, incluso si la condición es falsa desde el principio?**

for

while

do-while

forEach

1. **¿Cómo se incrementa una variable i en un bucle for en JavaScript?**

i++

++i

i+=1

Todas las anteriores

1. **¿Cuál es el propósito principal del bucle for...in en JavaScript?**

Iterar sobre propiedades de un objeto

Iterar sobre elementos de un array

Crear funciones

Ninguna de las anteriores

1. **¿Cuál de las siguientes opciones representa una estructura de control de múltiples casos en JavaScript?**

for

while

switch

if-else