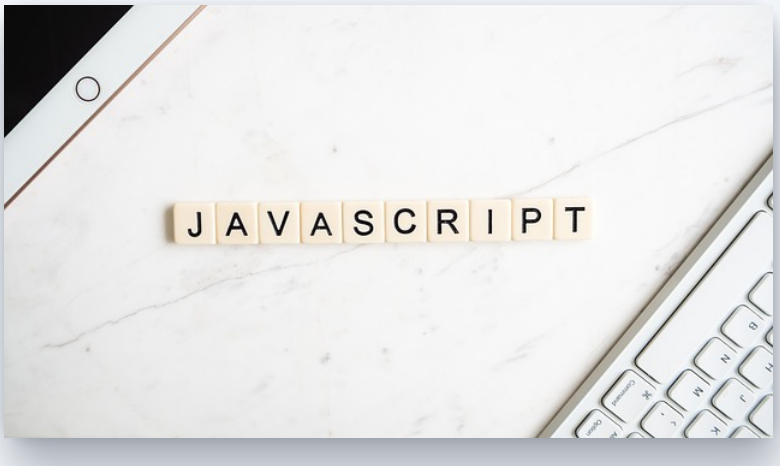

Curso de JavaScript

Aprende JavaScript desde cero

Descripción general

Este curso te proporcionará los fundamentos necesarios para empezar a programar en JavaScript. Aprenderás sobre variables, tipos de datos, operadores y expresiones,

estructuras de control, funciones, arreglos y objetos, eventos y manipulación del DOM, y funciones de orden superior y callbacks.

01 Variables

01 Variables y Tipos de Datos

En JavaScript, las variables son utilizadas para almacenar y manipular datos. Los tipos de datos son las diferentes categorías de valores que pueden ser asignados a una variable. En esta sección del curso, exploraremos la declaración de variables y los diferentes tipos de datos en JavaScript.

Declaración de variables

La declaración de variables en JavaScript se realiza utilizando las palabras clave `let` o `const`, seguidas del nombre de la variable. Por ejemplo:

```
 let nombre = "Juan";
const edad = 25;
```

En el ejemplo anterior, se declara una variable llamada `nombre` y se le asigna el valor "Juan". La palabra clave `let` se utiliza para declarar variables que pueden ser modificadas posteriormente, mientras que `const` se utiliza para variables cuyo valor no puede ser cambiado una vez asignado.

Tipos de datos

JavaScript cuenta con varios tipos de datos, entre los más comunes se encuentran:

Números

Los números en JavaScript pueden ser enteros o decimales. Por ejemplo:

```
 let numero = 10; // entero
let numeroDecimal = 3.14; // decimal
```

Cadenas de texto

Las cadenas de texto se utilizan para representar texto en JavaScript. Se pueden definir utilizando comillas simples o dobles. Por ejemplo:

```
 let texto = "Hola, mundo!";
let otraCadena = 'Ejemplo de cadena';
```

Booleanos

Los valores booleanos representan la verdaderidad o falsedad de una condición. En JavaScript, los valores booleanos pueden ser `true` o `false`. Por ejemplo:

```
 let esVerdadero = true;
let esFalso = false;
```

Conversión entre tipos

En JavaScript, es posible convertir un tipo de dato en otro utilizando diferentes métodos. Algunas conversiones comunes incluyen:

- Convertir un número a una cadena de texto utilizando el método `toString()`.
- Convertir una cadena de texto a un número utilizando las funciones `parseInt()` o `parseFloat()`.
- Convertir un valor a un booleano utilizando el operador de negación `!`.

En resumen, en esta sección del curso "Curso de JavaScript" hemos aprendido sobre la declaración de variables y los tipos de datos en JavaScript. Hemos explorado cómo declarar variables utilizando las palabras clave `let` y `const`, y los diferentes tipos de datos como números, cadenas de texto y booleanos. También hemos mencionado las conversiones entre tipos de datos en JavaScript.

Continúa aprendiendo y practicando estos conceptos a medida que avances en el curso.

Conclusión - Variables y Tipos de Datos

El tema de Variables y Tipos de Datos en el curso de JavaScript te ha proporcionado una base sólida para comprender cómo trabajar con variables, declararlas correctamente y manipular diferentes tipos de datos como números, cadenas y booleanos. También has aprendido a realizar conversiones entre tipos de datos, lo que es fundamental en el desarrollo de aplicaciones web.

02 Operadores y Expresiones

Operadores y Expresiones

En JavaScript, los operadores y expresiones nos permiten realizar diferentes tipos de manipulaciones y cálculos con valores y variables. Estos elementos son

fundamentales para la programación y nos ayudan a realizar tareas como enrutamiento, cálculos matemáticos, comparaciones y más.

Operadores Aritméticos

JavaScript proporciona una serie de operadores aritméticos que nos permiten realizar operaciones matemáticas básicas. Estos operadores incluyen:

- Suma (+): permite sumar dos valores.
- Resta (-): permite restar un valor de otro.
- Multiplicación (*): permite multiplicar dos valores.
- División (/): permite dividir un valor entre otro.
- Modulo (%): devuelve el resto de una división entre dos valores.

Operadores de Comparación

Los operadores de comparación se utilizan para comparar valores y devolver un valor booleano (true o false) en función del resultado de la comparación. Algunos de los operadores de comparación más comunes son:

- Igual (===): compara si dos valores son iguales.
- No igual (!==): compara si dos valores son diferentes.
- Mayor que (>): compara si un valor es mayor que otro.
- Mayor o igual que (>=): compara si un valor es mayor o igual que otro.
- Menor que (<): compara si un valor es menor que otro.
- Menor o igual que (<=): compara si un valor es menor o igual que otro.

Operadores Lógicos

Los operadores lógicos se utilizan para combinar o invertir valores booleanos. Los operadores lógicos más comunes son:

- Y lógico (&&): devuelve true si ambos valores son true.
- O lógico (||): devuelve true si al menos uno de los valores es true.
- No lógico (!): invierte el valor booleano.

Expresiones

En JavaScript, una expresión es una combinación de variables, valores y operadores que se evalúa para producir un nuevo valor. Por ejemplo, la expresión `a + b` suma dos variables y produce un nuevo valor.

Las expresiones pueden ser tan simples como una única variable o tan complejas como se requiera. Se pueden combinar diferentes operadores y utilizar paréntesis para establecer un orden de evaluación específico.

Es importante tener en cuenta que las expresiones pueden tener efectos secundarios, como la modificación de variables o la llamada a funciones.

En resumen, los operadores y expresiones en JavaScript nos permiten realizar cálculos, comparaciones y manipulaciones de valores de manera eficiente. Estos elementos son esenciales en la programación y son clave para comprender y utilizar JavaScript de manera efectiva.

Conclusión - Operadores y Expresiones

En el tema de Operadores y Expresiones, has descubierto cómo utilizar los diferentes operadores aritméticos, de comparación y lógicos para realizar cálculos y evaluaciones en tus programas. También has aprendido sobre la concatenación de cadenas y la forma en que se evalúan las expresiones en JavaScript.

03 Estructuras de Control

Las estructuras de control en JavaScript permiten controlar el flujo de ejecución de un programa. Esto significa que podemos tomar decisiones en base a ciertas condiciones y repetir una determinada acción varias veces. En esta sección del curso aprenderemos sobre las siguientes estructuras de control:

Sentencias condicionales (if, else if, else)

Las sentencias condicionales nos permiten ejecutar diferentes bloques de código dependiendo de si una condición es verdadera o falsa. La sintaxis básica de una sentencia if es la siguiente:

```
 if (condicion) {
  // bloque de codigo si la condicion es verdadera
} else if (otraCondicion) {
  // bloque de codigo si la otra condicion es verdadera
} else {
  // bloque de codigo si ninguna de las condiciones anteriores es verdadera
}
```

En el ejemplo anterior, si la condición es verdadera, se ejecutará el bloque de código correspondiente. Si la condición es falsa pero la otraCondicion es verdadera, se ejecutará el bloque de código correspondiente a ela estructura else if. Si ninguna de las condiciones anteriores es verdadera, se ejecutará el bloque de código correspondiente a la estructura else.

Bucles (for, while)

Los bucles nos permiten ejecutar un bloque de código varias veces. Hay diferentes tipos de bucles en JavaScript, pero los dos más comunes son el bucle for y el bucle while.

El bucle for se utiliza cuando conocemos la cantidad de veces que se debe ejecutar el bloque de código. La sintaxis básica de un bucle for es la siguiente:

```
for (let i = 0; i < limite; i++) {
 // bloque de codigo
```

```
}
```

En el ejemplo anterior, el bucle se ejecutará mientras la variable `i` sea menor que el límite establecido. En cada iteración, la variable `i` se incrementa en uno.

El bucle while se utiliza cuando la cantidad de iteraciones no está determinada de antemano. La sintaxis básica de un bucle while es la siguiente:

```
 while (condicion) {
  // bloque de codigo
}
```

En el ejemplo anterior, el bucle se ejecutará mientras la condición sea verdadera. Es importante asegurarse de que la condición se vuelva falsa en algún momento para evitar bucles infinitos.

Sentencias de control (break, continue)

Las sentencias de control nos permiten controlar el flujo de ejecución dentro de un bucle o estructura condicional.

La sentencia `break` se utiliza para salir de un bucle o interruptor. Cuando se ejecuta la sentencia `break`, el programa se salta el resto del bucle o interruptor y continúa con la ejecución del código que sigue después.

La sentencia `continue` se utiliza para saltar la iteración actual de un bucle y continuar con la siguiente iteración. Cuando se ejecuta la sentencia `continue`, el programa se salta el resto del bloque de código en esa iteración y pasa a la siguiente iteración.

En resumen, las estructuras de control en JavaScript son herramientas fundamentales para controlar el flujo de ejecución de un programa. Podemos tomar decisiones utilizando sentencias condicionales, repetir acciones utilizando bucles y controlar el flujo dentro de bucles y estructuras condicionales utilizando sentencias de control. Estas estructuras nos permiten escribir programas más complejos y funcionales.

Conclusión - Estructuras de Control

Las Estructuras de Control son fundamentales en cualquier lenguaje de programación, y en el curso de JavaScript has aprendido sobre las sentencias condicionales como if, else if y else, así como los bucles for y while. También has explorado las sentencias de control break y continue, que te permiten controlar el flujo de ejecución de tu código de manera más precisa.

04 Funciones

Las funciones son bloques de código reutilizables que realizan una tarea específica cuando son invocadas. En JavaScript, las funciones son un componente fundamental y poderoso del lenguaje. A continuación, exploraremos los diferentes aspectos de las funciones en JavaScript.

Declaración de funciones

En JavaScript, se pueden declarar funciones utilizando la palabra clave `function`, seguida de un nombre y un par de paréntesis. Estos paréntesis pueden contener los parámetros que la función espera recibir. Luego, el código de la función se coloca entre llaves (`{}`) para delimitar su inicio y fin.

Por ejemplo, consideremos la siguiente función que saluda a una persona:

```
function saludar(nombre) {
  console.log("Hola, " + nombre + "!");
}
```

En este caso, la función `saludar` acepta un parámetro `nombre` y muestra un mensaje de saludo en la consola.

Llamada a funciones

Después de declarar una función, podemos invocarla o llamarla utilizando su nombre seguido de un par de paréntesis. Si la función espera parámetros, estos se pueden proporcionar dentro de los paréntesis.

Siguiendo el ejemplo anterior, podemos llamar a la función `saludar` de la siguiente manera:

```
saludar("María");
```

Al llamar a esta función con el argumento `"María"`, se mostrará el mensaje "Hola, María!" en la consola.

Retorno de valores

Además de ejecutar un bloque de código, las funciones en JavaScript también pueden retornar un valor utilizando la palabra clave `return`. Esto nos permite obtener un resultado específico de una función y utilizarlo en otras partes de nuestro programa.

Veamos un ejemplo de una función que realiza una suma y devuelve el resultado:

```
 function suma(a, b) {
  return a + b;
}
```

En este caso, la función `suma` acepta dos parámetros `a` y `b`, realiza la operación de suma y devuelve el resultado utilizando la palabra clave `return`.

Podemos utilizar esta función y almacenar su resultado en una variable, como se muestra a continuación:

```
 let resultado = suma(3, 7);
console.log("Resultado: " + resultado);
```

Al ejecutar este código, se muestra en la consola el mensaje "Resultado: 10", que es el valor retornado por la función `suma`.

Ámbito de variables

Las funciones en JavaScript también tienen su propio ámbito de variables, lo que significa que las variables declaradas dentro de una función solo son visibles y accesibles dentro de esa función, a menos que sean declaradas como variables globales.

Esto es importante tenerlo en cuenta, ya que el ámbito de una variable determina su alcance y cómo se puede acceder a ella desde diferentes partes del código.

Conclusión - Funciones

El tema de Funciones te ha introducido al mundo de la modularidad y la reutilización de código. Has aprendido a declarar funciones, utilizar parámetros y argumentos para hacerlas más flexibles y a retornar valores para obtener

resultados. También has entendido el ámbito de las variables y cómo afecta el alcance de las mismas dentro de las funciones.

05 Arreglos y Objetos

En JavaScript, los **arreglos** y los **objetos** son estructuras de datos muy utilizadas para almacenar y manipular información de manera organizada. A continuación, veremos cómo se crean y se manipulan estas estructuras.

Arreglos

Un arreglo es una colección ordenada de elementos. Cada elemento del arreglo se identifica por su posición, que se conoce como **índice**. Para crear un arreglo en JavaScript, utilizamos corchetes `[]` y separamos los elementos con comas `,`. Por ejemplo:

```
let numeros = [1, 2, 3, 4, 5];
```

En el arreglo `numeros`, tenemos cinco elementos: 1, 2, 3, 4, y 5. Podemos acceder a los elementos del arreglo utilizando su índice, que comienza desde cero. Por ejemplo:

```
console.log(numeros[2]); // Acceso al tercer elemento: 3
```

También podemos modificar los elementos de un arreglo utilizando su índice:

```
numeros[0] = 10;
console.log(numeros); // [10, 2, 3, 4, 5]
```

Además de acceder y modificar los elementos, los arreglos en JavaScript tienen métodos que nos permiten realizar diferentes operaciones. Por ejemplo, podemos agregar elementos al final del arreglo utilizando el método `push()`:

```
numeros.push(6);
console.log(numeros); // [10, 2, 3, 4, 5, 6]
```

Podemos eliminar el último elemento del arreglo utilizando el método `pop()`:

```
numeros.pop();
console.log(numeros); // [10, 2, 3, 4, 5]
```

Objetos

Un objeto es una colección de propiedades, donde cada propiedad tiene un nombre y un valor asociado. Para crear un objeto en JavaScript, utilizamos llaves `{}` y

separamos las propiedades con comas `,`. Cada propiedad se define con un nombre seguido de dos puntos `:` y su valor correspondiente. Por ejemplo:

```
 let persona = {
  nombre: "Ana",
  edad: 30,
  profesion: "Ingeniera"
};
```

En el objeto `persona`, tenemos tres propiedades: `nombre`, `edad` y `profesion`. Podemos acceder a las propiedades del objeto utilizando la notación de punto. Por ejemplo:

```
 console.log(persona.nombre); // "Ana"
```

También podemos modificar el valor de una propiedad de un objeto:

```
 persona.edad = 31;
console.log(persona); // { nombre: "Ana", edad: 31, profesion: "Ingeniera" }
```

Al igual que los arreglos, los objetos en JavaScript tienen métodos que nos permiten realizar diferentes operaciones. Por ejemplo, podemos agregar una nueva propiedad a un objeto utilizando la notación de punto:

```
 persona.ciudad = "Madrid";
console.log(persona); // { nombre: "Ana", edad: 31, profesion: "Ingeniera", ci
```

Podemos eliminar una propiedad de un objeto utilizando el operador `delete`:

```
 delete persona.profesion;
console.log(persona); // { nombre: "Ana", edad: 31, ciudad: "Madrid" }
```

Los arreglos y los objetos son estructuras de datos fundamentales en JavaScript. Al dominar su uso, tendrás una base sólida para trabajar con datos de manera eficiente y organizada en tus programas JavaScript.

Recuerda practicar y explorar más sobre arreglos y objetos para profundizar tus conocimientos y habilidades en JavaScript.

Conclusión - Arreglos y Objetos

En el tema de Arreglos y Objetos, has aprendido a trabajar con estructuras de datos más complejas en JavaScript. Has descubierto cómo crear y manipular arreglos, acceder a sus elementos y utilizar propiedades y métodos específicos. Además, has explorado la creación y manipulación de objetos, que te permiten organizar y manipular datos de manera eficiente.

06 Eventos y Manipulación del DOM

En JavaScript, los eventos y la manipulación del DOM son dos aspectos fundamentales para interactuar con el usuario y cambiar dinámicamente el contenido y comportamiento de una página web. En esta sección del curso, aprenderemos cómo asociar eventos a elementos HTML y cómo manipular el DOM utilizando JavaScript.

Asociación de eventos a elementos HTML

Para realizar acciones en respuesta a las interacciones de los usuarios, es necesario asociar eventos a los elementos HTML. Un evento puede ser cualquier acción que suceda en la página, como hacer clic en un botón, mover el cursor o pulsar una tecla. JavaScript proporciona métodos para asociar eventos a elementos específicos y ejecutar una función en respuesta a ese evento.

Un ejemplo de asociación de eventos se encuentra en el siguiente código:

```
 document.getElementById("boton").addEventListener("click", function() {
      console.log("Botón clickeado");
});
```

En este caso, estamos asociando un evento de clic al elemento con el identificador "boton". Cuando el botón sea clickeado, se ejecutará la función anónima y se imprimirá en la consola el mensaje "Botón clickeado". Podemos asociar diferentes eventos a diferentes elementos y realizar acciones específicas en respuesta a cada uno de ellos.

Manipulación del DOM (Document Object Model)

El DOM representa la estructura de un documento HTML como un árbol de elementos. Con JavaScript, podemos acceder a los elementos del DOM y modificar su contenido, atributos y estilos. Esto nos permite cambiar dinámicamente la apariencia y el comportamiento de una página web.

Cambio de contenido

```
document.getElementById("parrafo").textContent = "Nuevo texto";
```

En este ejemplo, estamos cambiando el contenido del elemento con el identificador "parrafo". Utilizando la propiedad `textContent`, asignamos el valor "Nuevo texto" al contenido de ese elemento. Podemos modificar el contenido de cualquier tipo de elemento, ya sea un párrafo, una etiqueta de encabezado, un enlace, etc.

Cambio de estilos

```
document.getElementById("elemento").style.color = "red";
```

En este caso, estamos cambiando el estilo del elemento con el identificador "elemento". Utilizando la propiedad `style`, podemos acceder a los estilos del elemento y cambiar cualquier propiedad CSS. En este ejemplo, estamos cambiando el color del texto a rojo. Podemos modificar cualquier propiedad CSS, como tamaño de fuente, fondo, margen, etc.

La manipulación del DOM nos permite crear páginas web interactivas y dinámicas, donde podemos responder a las acciones de los usuarios y modificar el contenido y los estilos en tiempo real.

En resumen, en esta sección del curso de JavaScript hemos aprendido sobre los eventos y la manipulación del DOM. Hemos visto cómo asociar eventos a elementos HTML y cómo modificar el contenido y los estilos de los elementos utilizando JavaScript. Estos conocimientos nos permitirán crear páginas web más interactivas y personalizadas. Es importante practicar y explorar más para profundizar en estos conceptos en el desarrollo de aplicaciones web.

Conclusión - Eventos y Manipulación del DOM

La manipulación del DOM (Document Object Model) y los eventos son aspectos clave en el desarrollo web con JavaScript.

En este tema del curso, has aprendido a asociar eventos a elementos HTML, a manipular el contenido y estilos del DOM y a reaccionar ante las interacciones del usuario. Estas habilidades te permiten crear aplicaciones web dinámicas e interactivas.

07 Funciones de Orden Superior y Callbacks

Funciones de Orden Superior y Callbacks

Las funciones de orden superior son una característica avanzada de JavaScript que te permite tratar a las funciones como cualquier otro valor. Esto significa que las

funciones pueden pasarse como parámetros a otras funciones, pueden ser devueltas como resultados de otras funciones y también pueden ser asignadas a variables.

Una de las principales aplicaciones de las funciones de orden superior es el uso de callbacks. Un callback es simplemente una función que se pasa como argumento a otra función y que se ejecuta después de que cierto evento ocurra o después de completar alguna operación.

En JavaScript, los callbacks son muy utilizados en eventos de manejo del DOM, en operaciones asincrónicas como peticiones AJAX y en la ejecución de funciones asincrónicas como setInterval() y setTimeout().

Un ejemplo de uso de callbacks es la función de orden superior `operar` que recibe dos números y una función de operación como argumentos. Esta función ejecuta la operación deseada utilizando los números de entrada y retorna el resultado.

```
 function operar(a, b, operacion) {
    return operacion(a, b);
}
```

Supongamos que tenemos dos funciones de operación, `suma` y `resta`, que toman dos números como parámetros y retornan el resultado de sumar y restar esos números, respectivamente.

```
 function suma(a, b) {
    return a + b;
}

function resta(a, b) {
    return a - b;
}
```

Podemos utilizar estas funciones como callbacks en la función `operar` para realizar la operación deseada.

```
 let resultadoSuma = operar(5, 3, suma);
let resultadoResta = operar(10, 4, resta);

console.log("Suma: " + resultadoSuma);
console.log("Resta: " + resultadoResta);
```

En este ejemplo, pasamos las funciones `suma` y `resta` como callbacks a la función `operar` y obtuvimos los resultados deseados.

El uso de funciones de orden superior y callbacks es una técnica muy poderosa en JavaScript que te permite flexibilidad y modularidad en tu código, ya que puedes reutilizar funciones existentes y personalizar su comportamiento según tus necesidades.

Recuerda practicar con ejemplos adicionales y explorar más acerca de las funciones de orden superior y los callbacks para aprovechar todo su potencial en tus proyectos de JavaScript.

Conclusión - Funciones de Orden Superior y Callbacks

Finalmente, en el tema de Funciones de Orden Superior y Callbacks, has explorado conceptos más avanzados de JavaScript. Has aprendido a trabajar con funciones como valores, a utilizar funciones de orden superior para resolver

problemas más complejos y a emplear callbacks para ejecutar código de manera asíncrona. Estas habilidades te permiten escribir un código más eficiente y expresivo.

Pongamos en práctica tus conocimientos

08 Ejercicios Practicos

En esta lección, pondremos la teoría en práctica a través de actividades prácticas. Haga clic en los elementos a continuación para verificar cada ejercicio y desarrollar habilidades prácticas que lo ayudarán a tener éxito en el tema.

Declaración de variables

Crea una variable llamada 'nombre' y asígnale tu nombre. Luego, crea una constante llamada 'edad' y asígnale tu edad actual.

Tipos de datos

Crea una variable llamada 'numero' y asígnale un número cualquiera. Luego, crea una variable llamada 'texto' y asígnale un mensaje de saludo. Por último, crea una variable llamada 'esVerdadero' y asígnale un valor booleano.

Operaciones aritméticas

Crea dos variables: 'a' y 'b', y asígnales valores numéricos. Luego, realiza las siguientes operaciones: suma, resta, multiplicación y división entre 'a' y 'b'.

Operadores de comparación

Crea dos variables: 'a' y 'b', y asígnales valores numéricos. Luego, utiliza los operadores de comparación para verificar si 'a' es igual a 'b', si 'a' es mayor que 'b' y si 'a' es menor que 'b'.

Sentencias condicionales

Crea una variable llamada 'edad' y asígnale un valor numérico. Utiliza una sentencia condicional para determinar si la persona es mayor de edad o menor de edad, y muestra un mensaje en cada caso.

Bucles

Utiliza un bucle 'for' para repetir una acción 5 veces. Dentro del bucle, muestra en consola el número de iteración. Luego, utiliza un bucle 'while' para contar desde 0 hasta 2, mostrando en consola el valor del contador en cada iteración.

Declaración y uso de funciones

Crea una función llamada 'saludar' que reciba como parámetro un nombre y muestre en consola un mensaje de saludo. Luego, llama a la función pasando tu nombre como argumento. Por último, crea una función llamada 'suma' que reciba dos números como parámetros y retorne la suma de los mismos. Llama a la función pasando dos números y muestra el resultado en consola.

Creación y manipulación de arreglos

Crea un arreglo llamado 'numeros' con varios números. Accede al tercer elemento del arreglo y muestra su valor en consola.

Creación y manipulación de objetos

Crea un objeto llamado 'persona' con varias propiedades: 'nombre', 'edad' y 'profesion'. Accede a la propiedad 'nombre' del objeto y muestra su valor en consola.

Asociación de eventos

Asocia un evento de clic a un botón HTML y muestra un mensaje en consola cuando el botón sea clickeado.

Cambio de contenido y estilos

Selecciona un párrafo HTML y cambia su contenido por otro texto. Luego, selecciona un elemento HTML y cambia su color de texto a rojo.

Funciones de orden superior

Crea una función de orden superior llamada 'operar' que reciba tres parámetros: 'a', 'b' y 'operacion'. La función 'operar' debe llamar a la operación pasando 'a' y 'b' como argumentos y retornar el resultado. Luego, crea dos funciones 'suma' y 'resta' que reciban dos números como parámetros y retornen la suma y la resta respectivamente. Utiliza la función de orden superior 'operar' para calcular la suma y la resta de dos números y muestra los resultados en consola.

Uso de callbacks

Crea dos funciones 'suma' y 'resta' que reciban dos números como parámetros y retornen la suma y la resta respectivamente. Luego, utiliza estas funciones como callbacks en una función de orden superior llamada 'operar' para calcular la suma y la resta de dos números y muestra los resultados en consola.

Repasemos lo que acabamos de ver hasta ahora

09 Resumen

- ✓ El tema de Variables y Tipos de Datos en el curso de JavaScript te ha proporcionado una base sólida para comprender cómo trabajar con variables, declararlas correctamente y manipular diferentes tipos de datos como números, cadenas y booleanos. También has aprendido a realizar conversiones entre tipos de datos, lo que es fundamental en el desarrollo de aplicaciones web.
- ✓ En el tema de Operadores y Expresiones, has descubierto cómo utilizar los diferentes operadores aritméticos, de comparación y lógicos para realizar cálculos y evaluaciones en tus programas. También has aprendido sobre la concatenación de cadenas y la forma en que se evalúan las expresiones en JavaScript.
- ✓ Las Estructuras de Control son fundamentales en cualquier lenguaje de programación, y en el curso de JavaScript has aprendido sobre las sentencias condicionales como if, else if y else, así como los bucles for y while. También

has explorado las sentencias de control break y continue, que te permiten controlar el flujo de ejecución de tu código de manera más precisa.

- ✓ El tema de Funciones te ha introducido al mundo de la modularidad y la reutilización de código. Has aprendido a declarar funciones, utilizar parámetros y argumentos para hacerlas más flexibles y a retornar valores para obtener resultados. También has entendido el ámbito de las variables y cómo afecta el alcance de las mismas dentro de las funciones.
- ✓ En el tema de Arreglos y Objetos, has aprendido a trabajar con estructuras de datos más complejas en JavaScript. Has descubierto cómo crear y manipular arreglos, acceder a sus elementos y utilizar propiedades y métodos específicos. Además, has explorado la creación y manipulación de objetos, que te permiten organizar y manipular datos de manera eficiente.
- ✓ La manipulación del DOM (Document Object Model) y los eventos son aspectos clave en el desarrollo web con JavaScript. En este tema del curso, has aprendido a asociar eventos a elementos HTML, a manipular el contenido y estilos del DOM y a reaccionar ante las interacciones del usuario. Estas habilidades te permiten crear aplicaciones web dinámicas e interactivas.
- ✓ Finalmente, en el tema de Funciones de Orden Superior y Callbacks, has explorado conceptos más avanzados de JavaScript. Has aprendido a trabajar con funciones como valores, a utilizar funciones de orden superior para resolver problemas más complejos y a emplear callbacks para ejecutar código de manera asíncrona. Estas habilidades te permiten escribir un código más eficiente y expresivo.

Comprueba tus conocimientos respondiendo unas preguntas

10 Prueba

1. ¿Qué son las variables en JavaScript?

- ◯ Sentencias que controlan el flujo del programa
- ◯ Declaraciones que guardan valores
- ◯ Funciones que realizan operaciones

2. ¿Cuáles son los tipos de datos primitivos en JavaScript?

- ◯ Funciones y eventos
- ◯ Números, cadenas y booleanos
- ◯ Arreglos y objetos

3. ¿Cuál de los siguientes es un operador aritmético en JavaScript?

- ◯ +
- ◯ ==
- ◯ &&

---

4. ¿Qué estructura de control se utiliza para tomar decisiones?

- ◯ if
- ◯ while
- ◯ for

---

5. ¿Qué se utiliza para crear y manipular arreglos en JavaScript?

- ◯ Operadores aritméticos
- ◯ Funciones de orden superior
- ◯ Sintaxis de corchetes []

---

6. ¿Qué se utiliza para asociar eventos a elementos HTML?

- ◯ Funciones de orden superior
- ◯ addEventListener
- ◯ Manipulación del DOM

---

7. ¿Qué se utiliza para pasar una función como argumento a otra función en JavaScript?

- ◯ Funciones de orden superior

- ◯ Operadores aritméticos
- ◯ Operadores lógicos

8. ¿Qué se utiliza para acceder a una propiedad de un objeto en JavaScript?

- ◯ Operadores de comparación
- ◯ Sintaxis de punto .
- ◯ Sintaxis de corchetes []

9. ¿Cuál es el operador de concatenación de cadenas en JavaScript?

- ◯ ==
- ◯ +
- ◯ &&

10. ¿Cuál de los siguientes es un bucle en JavaScript?

- ◯ if
- ◯ while
- ◯ for

11. ¿Cuál de los siguientes no es un tipo de dato en JavaScript?

- ◯ Arreglos
- ◯ Números

- ◯ Eventos

12. ¿Cuál de los siguientes es un operador de comparación en JavaScript?

- ◯ ==
- ◯ +
- ◯ &&

13. ¿Qué se utiliza para cambiar el contenido de un elemento HTML en JavaScript?

- ◯ addEventListener
- ◯ Operadores lógicos
- ◯ Manipulación del DOM

14. ¿Cuál de los siguientes es un tipo de dato en JavaScript?

- ◯ Booleanos
- ◯ Eventos
- ◯ Funciones

15. ¿Qué se utiliza para retornar un valor en una función en JavaScript?

- ◯ return
- ◯ if
- ◯ for

Conclusión

Felicidades!

¡Felicitaciones por completar este curso! Has dado un paso importante para desbloquear todo tu potencial. Completar este curso no se trata solo de adquirir conocimientos; se trata de poner ese conocimiento en práctica y tener un impacto positivo en el mundo que te rodea.

Comparte este curso