



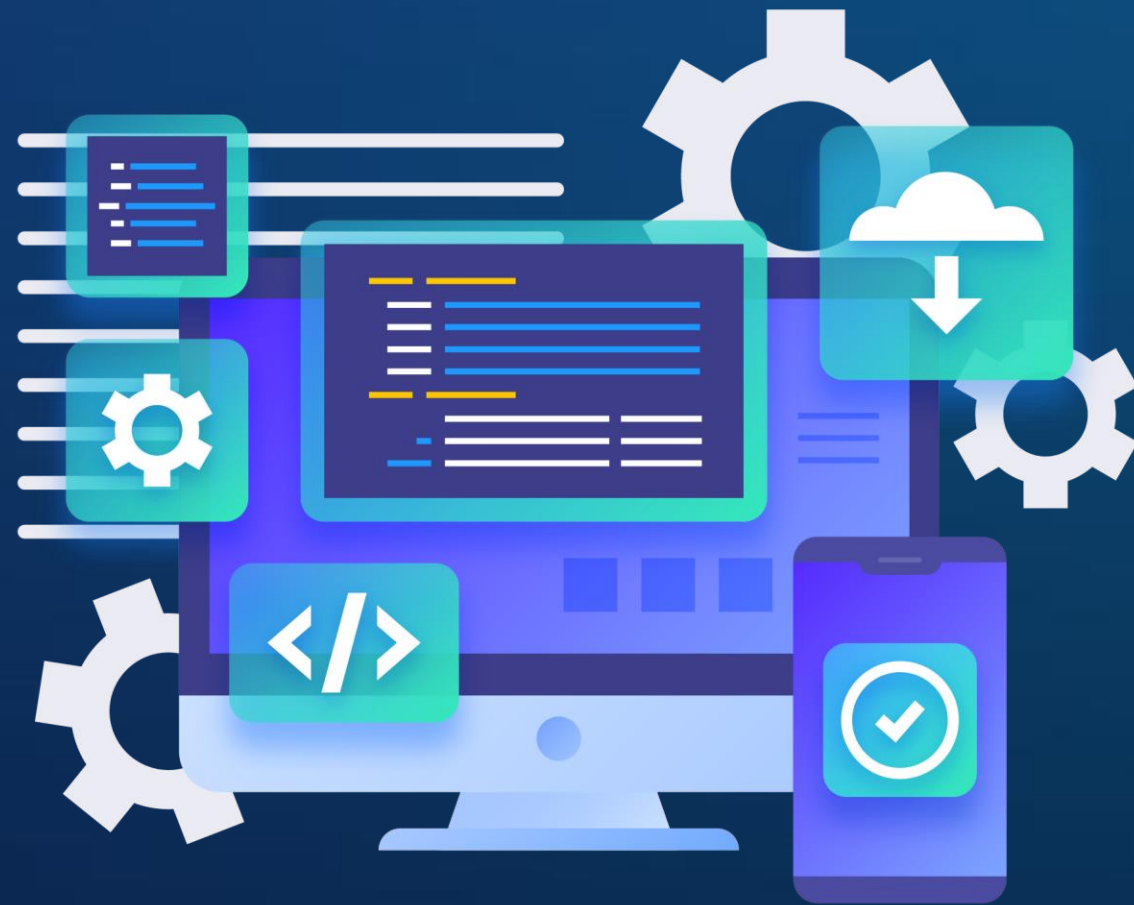
Sesión 13

Programación Nivel Básico



UNIVERSIDAD
LIBRE®

eTraining®



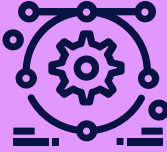
Sesión 13:

Introducción a Javascript


Declaración de funciones

Objetivos de la sesión

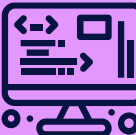
Al finalizar esta sesión estarás en capacidad de:



Aprender cómo se ejecuta JavaScript en un navegador.



Conocer las reglas sintácticas y semánticas que definen la estructura y significado de los elementos y expresiones del lenguaje de programación JavaScript.



Ampliar su funcionalidad y facilitar el desarrollo mediante el uso de librerías.

Introducción a las funciones en JavaScript

Las funciones son bloques de código reutilizables. Juegan un papel crucial en el desarrollo en JavaScript.

Permiten organizar y simplificar el código, mejorando su legibilidad y mantenimiento.



¿Qué es una función?



Concepto de Función

Una función es un bloque de código reutilizable que realiza una acción específica en programación.



Colaboración en Funciones

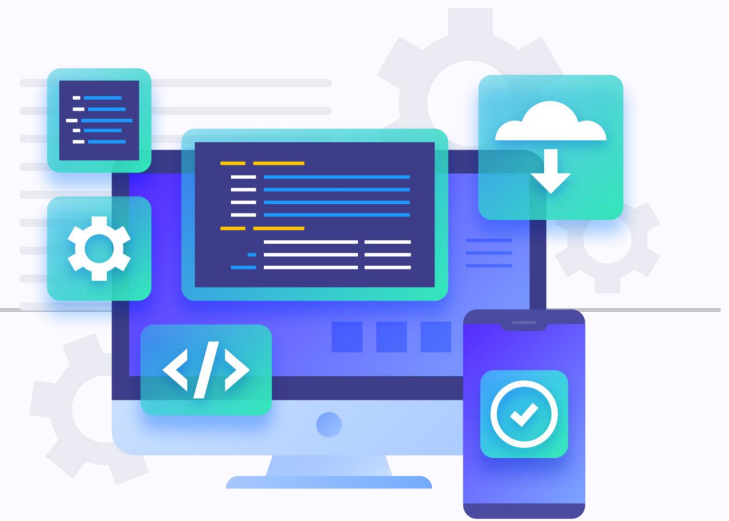
Las funciones permiten que varios desarrolladores trabajen en tareas específicas de manera eficiente.



Ejecución de Funciones

Las funciones toman entradas, realizan cálculos y devuelven resultados configurables.

Sintaxis básica de una función



1

1. Palabra clave

Utiliza "function" para declarar una función.

2

2. Nombre de la función

Elige un nombre descriptivo para mejorar la legibilidad.

3

3. Paréntesis

Incluye paréntesis para los parámetros, si son necesarios.

4

4. Llaves

Envuelve el código de la función entre llaves.

La sintaxis de una función es clave para su efectiva implementación. Cada parte tiene un propósito: palabra clave, nombre, parámetros y el bloque de código. Comprender estas partes es esencial para escribir funciones eficientes.



Declaración de una función

Las funciones en JavaScript se declaran para estructurar el código. Esto permite reutilizar lógicas en diferentes partes de una aplicación. A continuación, se presenta una estructura jerárquica sobre la declaración de funciones.





Función con parámetros

1

Definición de Parámetros

Los parámetros son variables que se pasan a una función. Permiten que la función reciba información externa.

2

Uso en Funciones

Los parámetros se declaran en la definición de la función. Se utilizan dentro de la función para realizar cálculos o procesos.

3

Ejemplo de Parámetros

Por ejemplo, una función puede aceptar dos números y devolver su suma. Así, puedes realizar operaciones con diferentes valores.





Función con Valores de Retorno

1

Definición

Una función con valor de retorno es aquella que devuelve un resultado después de ejecutarse.

2

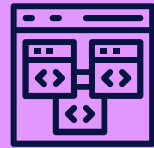
Sintaxis

Utiliza la palabra clave **return** seguida del valor que deseas devolver.

3

Ejemplo

Por ejemplo, una función puede calcular el área de un círculo y devolver el resultado.



Ámbito de las variables en funciones



Ámbito Global

Las variables globales son accesibles desde cualquier parte del código. Deben usarse con precaución para evitar conflictos.



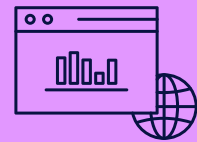
Ámbito Local

Las variables locales solo existen dentro de la función. Esto ayuda a mantener el código organizado y libre de errores.



Closures

Las funciones pueden acceder a variables de su entorno. Esto crea poderosas técnicas de programación.



CSS

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.



Funciones Anónimas

Definición

Las funciones anónimas son funciones sin nombre. Se utilizan cuando no se necesita reutilizar el código.

Ventajas

Facilitan la escritura de código más limpio y conciso. Permiten encapsular lógica sin crear nombres innecesarios.

Uso Común

Son frecuentemente usadas en callbacks y expresiones. Ayudan en la programación funcional.

Ejemplo Rápido

Ejemplo simple: `var suma = function(a, b) { return a + b; };`



```
owlcon } from "../assets/icons/arrow.svg";  
tcon } from "../assets/icons/bolt.svg";  
ightArrowlcon } from "../assets/icons/right-arrow.svg";
```

```
ffect, useRef } from "react";  
"react-transition-group";
```

```
"eslintConfig": {  
  "extends": [  
    "react-app",  
    "react-app" ]  
}
```



Funciones de flecha (arrow functions)

Definición

Las funciones de flecha son una forma concisa de escribir funciones en JavaScript. Su sintaxis es más simple que la de las funciones tradicionales.

Características

Las funciones de flecha no tienen su propio *this*, lo que facilita su uso en ciertas situaciones. Son ideales para funciones cortas y callbacks.

Ejemplo

Un ejemplo de función de flecha es: `() => { console.log('Hola'); }`. Esta estructura permite una escritura más limpia y entendible.



Funciones como objetos de primera clase

Definición

Las funciones en JavaScript son tratadas como objetos. Esto permite que se asignen a variables.

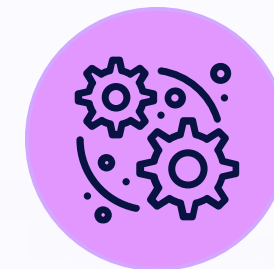
También pueden ser pasadas como argumentos a otras funciones.



Implicaciones

Este enfoque proporciona flexibilidad y potencia en el desarrollo.

Facilita el uso de funciones callback y promesas.





Funciones Callback

Definición

Las funciones callback son funciones pasadas como argumento a otra función. Se ejecutan en respuesta a eventos específicos.

Uso Común

Se utilizan en programación asíncrona, especialmente con llamadas a APIs. Estas funciones ayudan a manejar los resultados una vez que están disponibles.

Ventajas

Mejoran la legibilidad del código y permiten la ejecución de tareas de manera eficiente. Facilitando el manejo de operaciones complejas.

Funciones Recursivas



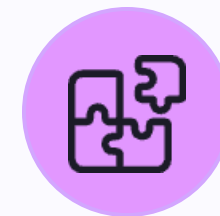
Definición

Las funciones recursivas son funciones que se llaman a sí mismas. Tienen una condición de parada para evitar bucles infinitos.



Ejemplo: Factorial

Un ejemplo clásico es la función factorial. Se calcula multiplicando el número por su anterior.



Uso

Se utilizan en problemas que pueden dividirse en subproblemas similares. Son útiles en programación de algoritmos complejos.

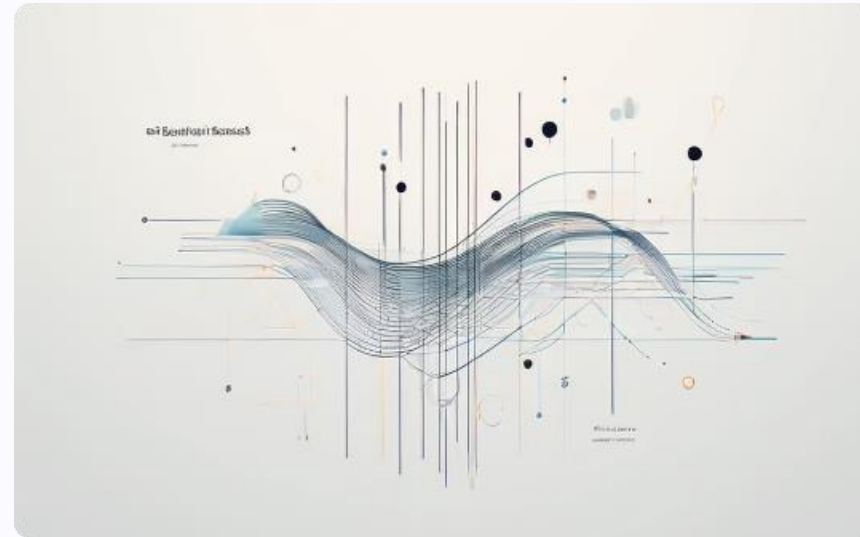


Funciones Predefinidas en JavaScript



Funciones Matemáticas

JavaScript ofrece funciones matemáticas integradas para cálculos complejos. Estas hacen que las operaciones sean eficientes y fáciles de implementar.



Funciones de Cadenas

Las funciones de cadenas permiten manipular y consultar texto. Son esenciales para la gestión de datos en aplicaciones.



Funciones de Arreglos

JavaScript proporciona métodos para manipular arreglos de forma efectiva. Facilitan operaciones como búsqueda y transformación de datos.



Ejercicio 1: Función que calcula el área de un círculo

1

Paso 1: Definir la función

Inicia escribiendo la palabra clave **function** seguida del nombre de la función, por ejemplo, **calcularArea**.

2

Paso 2: Usar parámetros

La función debe recibir un parámetro, **radio**, que será el radio del círculo.

3

Paso 3: Calcular el área

Utiliza la fórmula **Math.PI * radio * radio** para calcular el área.

4

Paso 4: Retornar el resultado

Finaliza la función usando **return** para devolver el área calculada.



Solución del Ejercicio 1

1

Código de la función

La función calcula el área de un círculo. Utiliza la fórmula $A = \pi * r^2$, donde "r" es el radio.

2

Uso de la función

Llama a la función con un valor para el radio. Esto devolverá el área calculada.

3

Ejemplo práctico

Por ejemplo, para un radio de 5, el área es aproximadamente 78.54. Esta solución es útil en geometría.

Ejercicio 2: Función que convierte grados Celsius a Fahrenheit

1

Punto de partida

Usar la fórmula para conversión.

2

Calcular

Apliquemos la fórmula matemática.

3

Resultado

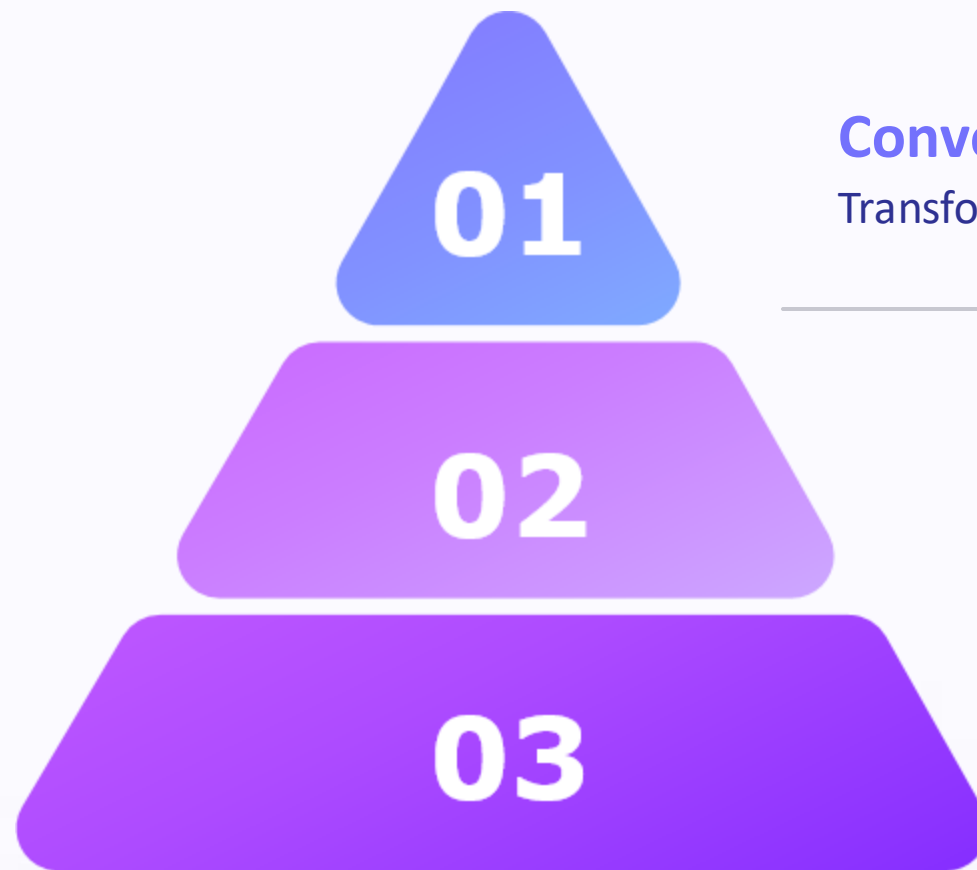
Obtén el valor en Fahrenheit.

Para convertir de Celsius a Fahrenheit, la fórmula es: $(^{\circ}\text{C} \times 9/5) + 32 = ^{\circ}\text{F}$. Este ejercicio ayuda a entender la manipulación básica de funciones en JavaScript al aplicar cálculos matemáticos.



Solución del Ejercicio 2

La función que convierte grados Celsius a Fahrenheit toma un parámetro y devuelve el resultado. A continuación, se presenta una representación de la solución utilizando una estructura piramidal.



Conversión

Transforma Celsius a Fahrenheit.

Fórmula

$$F = (C * 9/5) + 32$$

Ejemplo

Input: 25°C, Output: 77°F

Esta función es útil en aplicaciones que requieren conversión de temperatura.



Solución en Código del Ejercicio 2

```
1 <script>
2   let centi, fare;
3
4   centi = parseFloat(prompt("Ingresa grados centígrados"));
5   fare = (9/5*centi)+32;
6
7   console.log(centi+" centígrados equivalen a "+fare+" fahrenheit");
8 </script>
```

Ejercicio 3: Función que verifica si un número es par o impar

En este ejercicio, vamos a escribir una función en JavaScript que determina si un número es par o impar. Esta es una habilidad fundamental en programación, dado que las condiciones están presentes en muchos algoritmos.



La función se puede definir de la siguiente manera:

```
function esPar(num) { return num % 2 === 0; }
```

Cuando se llama a la función, simplemente pasamos un número y obtendremos el resultado adecuado.



Solución del Ejercicio 3

1

1. Función definida

Creamos una función que recibe un número.

2

2. Verificación

Comprobamos si el número es par o impar.

3

3. Retorno del resultado

Retornamos un mensaje con el resultado.

La función verifica si un número es par o impar. Si el número es par, retorna "El número es par". Si es impar, retorna "El número es impar".



Solución en Código del Ejercicio 3

```
1 function esPar(numero) {  
2   return numero % 2 === 0;  
3 }  
4  
5 function esImpar(numero) {  
6   return numero % 2 !== 0;  
7 }  
8  
9 console.log(esPar(4)); // true  
10 console.log(esPar(5)); // false  
11 console.log(esImpar(3)); // true  
12 console.log(esImpar(10)); // false
```




Aplicaciones de las funciones en JavaScript

1 Modularidad

Las funciones permiten dividir el código en partes más pequeñas y manejables.

2 Reutilización

Las funciones se pueden llamar múltiples veces, evitando la duplicación de código.

3 Abstracción

Las funciones ocultan la complejidad, permitiendo un uso más fácil de las operaciones.

4 Callbacks y Asincronía

Las funciones pueden ser usadas como callbacks, facilitando la programación asíncrona.



Funciones en la gestión de datos



Estructuras de Datos

Las funciones permiten manejar estructuras de datos. Esto incluye listas, objetos y mapas. Facilitan la organización y acceso a la información.



Gestión de Bases de Datos

Con funciones, se pueden realizar consultas en bases de datos. Estas ayudan a extraer, modificar o eliminar información eficientemente.



Interacción con APIs

Las funciones son esenciales para consumir APIs. Facilitan la comunicación y el manejo de datos en aplicaciones web.



Analítica de Datos

Mediante funciones, se pueden analizar grandes volúmenes de datos. Esto ayuda a obtener insights valiosos para la toma de decisiones.



Buenas prácticas al declarar funciones



Claridad en la Nomenclatura

Usa nombres descriptivos para las funciones. Esto ayuda a entender su propósito fácilmente.



Comentarios Útiles

Incluir comentarios claros mejora la mantenibilidad del código. Explica razonamientos complejos.



Evitar Funciones Demasiado Largas

Mantén las funciones cortas y específicas. Facilita la lectura y el testeado del código.



Consistencia en la Estructura

Sigue un estilo de codificación consistente. Esto promueve la cohesión en el equipo de trabajo.

Resumen y conclusiones

Importancia de las funciones

Las funciones son fundamentales en JavaScript. Permiten la reutilización del código y simplifican la programación.

Flexibilidad y eficiencia

Utilizar funciones optimiza el tiempo de desarrollo. Facilitan la organización del código y la implementación de tareas complejas.

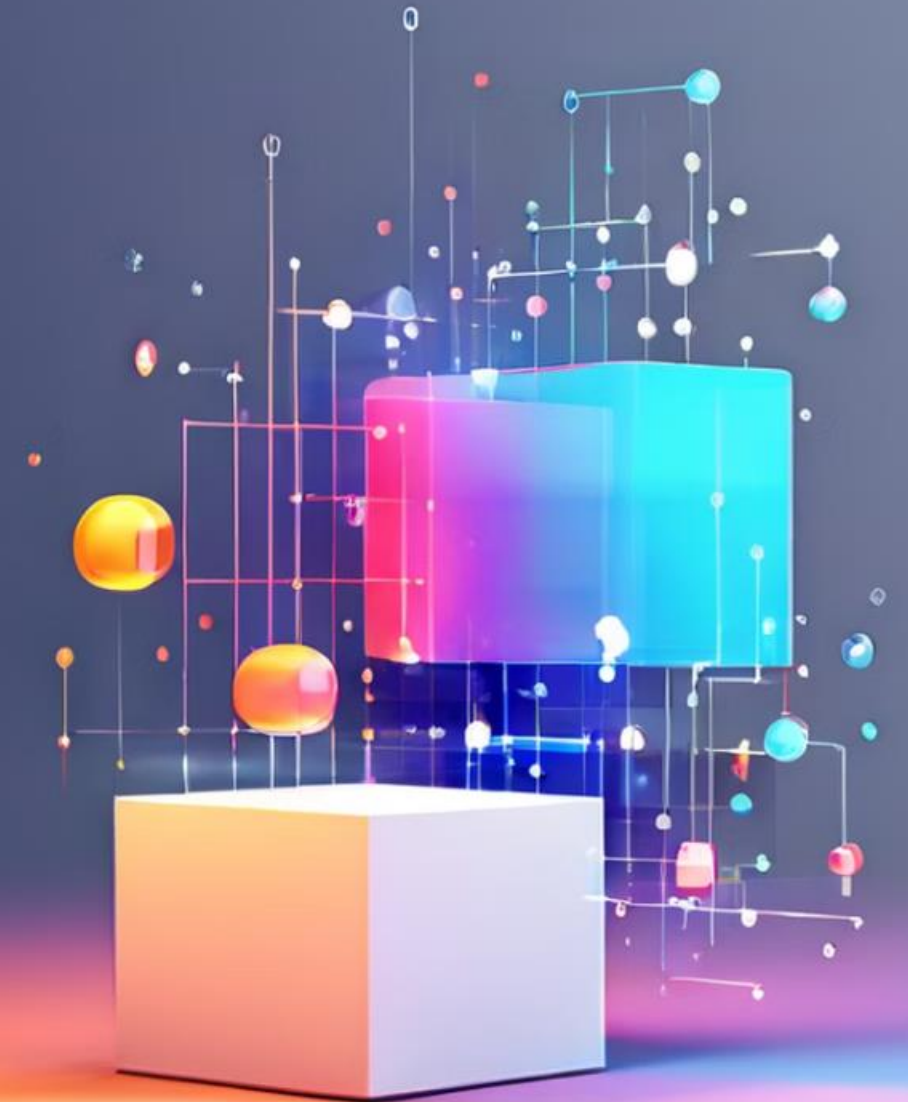


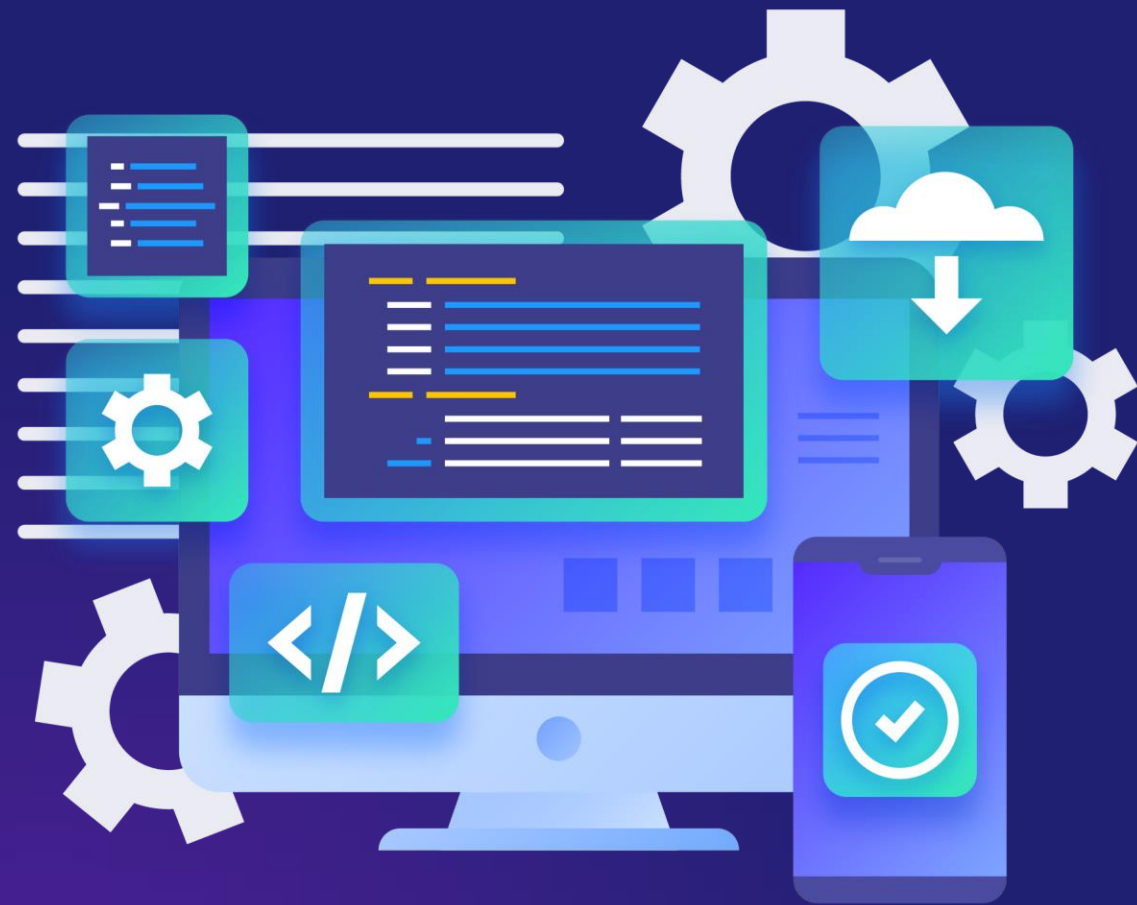
Buenas prácticas

Siempre declare funciones con nombres descriptivos. Mantenga un estilo de codificación limpio y consistente.

Conclusión final

Dominar funciones en JavaScript mejora las habilidades del desarrollador. Se recomienda practicar regularmente para afianzar los conceptos.





Ejercicios de Práctica



¡Gracias

**Por ser parte de esta
Experiencia de aprendizaje!**