**Parametros y tipos de Funciones**

* **Tabla de contenido**
* **Parámetros**
* **Valores predeterminados**
* **Tipos de Funciones**
* **3 tipos de funciones**

En JavaScript, las funciones son una de las características fundamentales del lenguaje y son esenciales para crear programas complejos.

Ya hemos visto cómo crear funciones y cómo llamarlas, pero para sacar el máximo provecho de las funciones, es importante entender cómo funcionan los parámetros y argumentos.

En esta clase, vamos a explorar cómo funcionan los parámetros y argumentos en las funciones de JavaScript, cómo pasar valores a las funciones y cómo utilizar estos valores dentro de las funciones. También veremos ejemplos prácticos y explicaciones detalladas para que puedas entender cómo utilizar parámetros y argumentos para crear funciones más flexibles y reutilizables en tus programas.

**Parámetros**

Podemos pasar datos arbitrarios a funciones usando parámetros.

En el siguiente ejemplo, la función tiene dos parámetros: from y text.

1

2

3

4

5

function showMessage(from, text) { // parámetros: from, text

alert(from + ': ' + text);

}

showMessage('Ann', '¡Hola!'); // Ann: ¡Hola! (\*)

\*showMessage('Ann', "¿Cómo estás?"); // Ann: ¿Cómo estás? (\*\*)

Cuando la función se llama (\*) y (\*\*), los valores dados se copian en variables locales from y text. Y la función las utiliza.

Aquí hay un ejemplo más: tenemos una variable from y la pasamos a la función. Tenga en cuenta: la función cambia from, pero el cambio no se ve afuera, porque una función siempre obtiene una copia del valor:

1

2

3

4

5

6

7

8

function showMessage(from, text) {

from = '\*' + from + '\*'; // hace que "from" se vea mejor

alert( from + ': ' + text );

}

let from = "Ann";

showMessage(from, "Hola"); // \*Ann\*: Hola

// el valor de "from" es el mismo, la función modificó una copia local

alert( from ); // Ann

Cuando un valor es pasado como un parámetro de función, también se denomina argumento.

Para poner los términos claros:

Un parámetro es una variable listada dentro de los paréntesis en la declaración de función (es un término para el momento de la declaración)

Un argumento es el valor que es pasado a la función cuando esta es llamada (es el término para el momento en que se llama).

Declaramos funciones listando sus parámetros, luego las llamamos pasándoles argumentos.

En el ejemplo de arriba, se puede decir: “la función showMessage es declarada con dos parámetros, y luego llamada con dos argumentos: from y "Hola"”.

**Valores predeterminados**

Si una función es llamada, pero no se le proporciona un argumento, su valor correspondiente se convierte en undefined.

Por ejemplo, la función mencionada anteriormente showMessage(from, text) se puede llamar con un solo argumento:

showMessage("Ann");

Eso no es un error. La llamada mostraría "Ann: undefined". Como no se pasa un valor de text, este se vuelve undefined.

Podemos especificar un valor llamado “predeterminado” o “por defecto” (es el valor que se usa si el argumento fue omitido) en la declaración de función usando =:

1

2

3

4

function showMessage(from,text = "sin texto">) {

alert( from + ": " + text );

}

showMessage("Ann"); // Ann: sin texto

Ahora, si no se pasa el parámetro text, obtendrá el valor "sin texto"

El valor predeterminado también se asigna si el parámetro existe pero es estrictamente igual a undefined:

Copiar

showMessage("Ann", undefined); // Ann: sin texto

**Tipos de Funciones**

En JavaScript, existen tres tipos de funciones: funciones declaradas, funciones de expresión y funciones de flecha.

Funciones declaradas: Son funciones que se declaran con la palabra clave “function” seguida del nombre de la función. Ejemplo:

function sum(a, b) {

return a + b;

}

Estas funciones son hoisted, es decir, son elevadas al inicio del scope donde son declaradas, por lo tanto son accesibles antes de ser declaradas.

Funciones de expresión: son funciones que son asignadas a una variable y son definidas mediante una expresión. Ejemplo:

const sum = function(a, b) {

return a + b;

}

Funciones de flecha: son funciones cuyo sintaxis es abreviada y se definen mediante una flecha “=>” y no tiene un nombre. Ejemplo:

const sum = (a, b) => a + b;

**3 tipos de funciones**

Para entender a detalle como funciona cada uno de estas, mira el siguiente tutorial donde verás qué diferencia a estas 3 formas de escribir una función: