EXPRESIONES DE FUNCION

Existe varias sintaxis para crear una función, por ejemplo la funcion llamada "Expresión de Función".
Esto nos permite crear una nueva función en el medio de cualquier expresión

```
let saludar = function() {
    alert( "Hola, ¿como estas?" );
};
```

Aquí podemos ver una variable "saludar" obteniendo un valor (la nueva función) creada como function() { alert("Hola,¿como estas?"); }.
Como la creación de una función ocurre en el contexto de una expresión de asignación, (el lado derecho de =), esto es una Expresión de función.

La funcion es un valor

No importa cómo es creada la función, una función es un valor.

La última línea no ejecuta la función, porque no hay paréntesis después de "name". Existen lenguajes de programación en los que cualquier mención del nombre de una función causa su ejecución, pero no en JavaScript

```
function name() {
    alert( "Fernando" );
}

alert( name );
```

```
function ask(question, yes, no) {
    if (confirm(question)) yes()
    else no();
}

ask(
    "Sacaste 10 en el examen?",
    function() { alert("felicidades, eres muy listo"); },
    function() { alert("Si estudias mas, mejoraras"); }
);
```

Funciones callback

Los argumentos de ask se llaman funciones callback
La idea es que pasamos una función y esperamos que se "devuelva la llamada" más tarde si es necesario. En nuestro caso, showOk se convierte en la callback para la respuesta "Yes", y showCancel para la respuesta "No".

Expresion de declaracion vs declaracion de funcion

-Expresión de Función: una función, creada dentro de una expresión o dentro de otra construcción sintáctica. Aquí, la función es creada en el lado derecho de la "expresión de asignación" =:

```
let sum = function(a, b) {
  return a + b;
};
```

Las expresiones de Función son creadas cuando la ejecución las alcance.

-Declaración de Función: una función, declarada como una instrucción separada, en el flujo de código principal.

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

Una declaración de Función puede ser llamada antes de ser definida.

Cuando necesitamos declarar una función, lo primera ha considerar es la sintaxis de la Declaración de Función. Da más libertad en cómo organizar nuestro código, porque podemos llamar a tales funciones antes de que sean declaradas.
También es un poco más fácil de buscar function f(...) {...} en el código comparado con let f = function(...) {...}. La Declaración de Función es más llamativa.
...Pero si una Declaración de Función no nos conviene por alguna razón, o necesitamos declaración condicional (hemos visto un ejemplo), entonces se debe usar la Expresión de función.

FUNCIONES FLECHA

Hay otra sintaxis muy simple y concisa para crear funciones, que a menudo es mejor que las Expresiones de funciones.

```
let func = (arg1, arg2, ..., argN) => expression;
```

```
let Multiply = (a, b) => a * b;

alert( sum(4, 8) ); //12
```

Si no hay parámetros, los paréntesis estarán vacíos; pero deben estar presentes.

```
let saludo = () => alert("Hola");

saludo();
```

Son muy convenientes para acciones simples de una línea, cuando somos demasiado flojos para escribir muchas palabras.

Funciones de flecha multilinea

En ese caso debemos encerrarlos entre llaves. La diferencia principal es que las llaves necesitan usar un “return” para devolver un valor (tal como lo hacen las funciones comunes).

```
let multiply = (a, b) => {
  let result = a * b;
  return result;
};

alert( multiply(4, 8) ); // 32
```