educación IT
CODE YOUR FUTURE

# JavaScript desde cero

Módulo 5

# *Arrow functions*

# *Arrow functions*

Las ***funciones flecha***, o *arrow functions*, ofrecen una estructura moderna y ligeramente diferente, respecto a las funciones convencionales JavaScript.

Además de aportar una **estructura distinta a las funciones convencionales, integra algunas características de JavaScript moderno que simplifican su elaboración**.

Veremos a continuación cómo estructurarlas.

Este es un **ejemplo de una función convencional** que utiliza, como siempre: la palabra reservada `function`, seguido del nombre de la función, sus paréntesis y las llaves de bloque que encierran su lógica.

```
function cargarProductos() {
    for (producto of productosElectronicos) {
        divContenedor.innerHTML += retornarHTMLDeProducto(producto)
    }
}
```

Aquí se observa, **la misma función, pero convertida a *arrow function***. Como vemos, parte de su nombre proviene de la flecha que se forma al combinar el carácter **igual** seguido de **mayor a**.

```
const cargarProductos = ()=> {
    for (producto of productosElectronicos) {
        divContenedor.innerHTML += retornarHTMLDeProducto(producto)
    }
}
```

## Declaración

Las funciones flecha requieren ser declaradas con la palabra reservada **`const`**, para cumplimentar el mismo cuidado que debemos tener al declarar: objetos, enlaces a elementos HTML, constantes, y toda aquella declaración de funcionalidades JS que necesitamos que no puedan ser sobreescritas, con otros valores.

# *Arrow functions* con parámetros

Cuando se deben utilizar **parámetros**, se definen tal como en una función convencional: **dentro de sus paréntesis**.

Internamente los utilizamos de igual forma que con las funciones convencionales.

En el ejemplo de la derecha, se observa también el uso de ***Template String + Literals*** sobre el valor resultante de la operación aritmética.

```
const multiplicarNumeros = (nroA, nroB)=> {
    let resultado = nroA * nroB
    console.log(`Resultado: ${resultado}`)
}
```

educación IT
CODE YOUR FUTURE

# *Arrow functions* con retorno

# *Arrow functions* con retorno

Y si debemos retornar valores o resultados con una *arrow function*, podemos aplicar también el **mismo mecanismo que utilizamos con las funciones convencionales**.

Aunque también, estas nos ofrecen **simplificar aún más su estructura**.

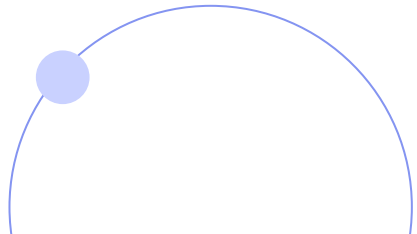
```
const multiplicarNumeros = (nroA, nroB)=> {
    let resultado = nroA * nroB
    return resultado
}
```

En el ejemplo del slide anterior, se puede **simplificar el retorno** del resultado de la operación aritmética a una sola línea de código, y suprimir la creación innecesaria de una variable.

Esto resume nuestra función, a una sola línea de código:

```
const multiplicarNumeros = (nroA, nroB)=> {
    return nroA * nroB
}

console.log("Resultado:", multiplicaNumeros(21, 75))
```



## Retorno implícito

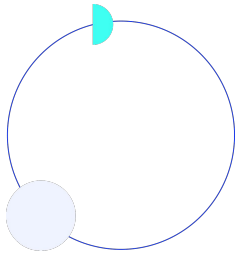
Cuando una *arrow function* puede resolver una tarea con una sola línea de código, nos da la capacidad de **suprimir las llaves de bloque que encierran su lógica, y de evitar utilizar la palabra reservada return**. En el ejemplo de la derecha, vemos y obtendremos el mismo resultado que lo que lograríamos con el primero de los ejemplos. Esto se denomina: *retorno implícito*.

```
const multiplicarNumeros = (nroA, nroB)=> {
    return nroA * nroB
}

// Arrow function simplificada

const multiplicarNumeros = (nroA, nroB)=> nroA * nroB
```

No es necesario aplicar *arrow functions* con retorno implícito que utilicen parámetros. También lo podemos hacer con **funcionalidades que no requieren parámetros**, tal como muestra el ejemplo de la siguiente diapositiva.



```
const carritoProductos = document.querySelector("img.carrito-productos span")

const carrito = [{id: 1, nombre: "Teclado Bluetooth", importe: 11950},
                 {id: 2, nombre: "Mouse Bluetooth", importe: 6990},
                 {id: 3, nombre: "Parlantes Wifi", importe: 22450},]



const obtenerTotalCarrito = ()=> carrito.length


carritoProductos.textContent = obtenerTotalCarrito()
```

Incluso las funciones flecha con retorno implícito son aplicables también en los ***Callbacks*** de los eventos, en lugar de tener que definir una función anónima.

```
const carritoProductos = document.querySelector("img.carrito-productos span")

carritoProductos.onclick = function() {
                                        navigator.href = "carrito.html"
                                       }

carritoProductos.onclick = ()=> navigator.href = "carrito.html"
```

# ¡Sigamos trabajando!