DigitalHouse >
Coding School

ARROW FUNCTIONS

“

Las funciones son de lo que más vas a usar a la hora de programar en Javascript.

Las **arrow functions** nos permiten escribirlas con una **sintaxis** más **compacta**.

ESTRUCTURA BÁSICA

Pensemos en una función simple que podríamos programar de la manera habitual, una suma de dos números.

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

Ahora veamos la versión reducida de esa misma función, al transformarla en una arrow function.

```
let sumar = (a, b) => a + b;
```

ESTRUCTURA BÁSICA

Nombre

Las arrow functions, son **siempre anónimas**, es decir que no tienen nombre como las funciones normales.

```
(a, b) => a + b;
```

Si queremos nombrarlas, es necesario escribirlas como una **función expresada**, es decir, asignarla como valor de una variable.

```
let sumar = (a, b) => a + b;
```

ESTRUCTURA BÁSICA

Parámetros

Usamos paréntesis para indicar los parámetros. Si nuestra función no recibe parámetros, debemos escribirlos igual.

```
let sumar = (a, b) => a + b;
```

Una particularidad de este tipo de funciones es que si recibe un **único parámetro,** podemos prescindir de los paréntesis.

```
let doble = a => a * 2;
```

ESTRUCTURA BÁSICA

Operador flecha

Lo usamos para indicarle a Javascript que vamos a escribir una función (*reemplaza a la palabra reservada* `function`).

Lo que está a la izquierda de la fecha será la entrada de la función (los parámetros) y lo que está a la derecha, la salida (el retorno)

```
let sumar = (a, b) => a + b;
```

ESTRUCTURA BÁSICA

Cuerpo

Escribimos la lógica de la función. Si la función tiene una sola línea de código y ésta misma es la que hay que retornar, no hacen falta las llaves ni la palabra reservada `return`.

```
let sumar = (a, b) => a + b;
```

De lo contrario, vamos a necesitar utilizar ambas.

```
let sumar = (a, b) => {
    return a + b;
};
```

“

Las **arrow functions** reciben su nombre por el operador **=>**

Si lo miramos con un poco de imaginación, se parece a una flecha.

En inglés suele llamarse **fat arrow** (flecha gorda) para diferenciarlo de otra combinación parecida ->

{ código }

```
let saludo = () => 'Hola Mundo!';

let dobleDe = numero => numero * 2;

let suma = (a, b) => a + b;

let horaActual = () => {
    let fecha = new Date();
    return fecha.getHours() + ':' + fecha.getMinutes();
}
```

{ código }

```
let saludo = () => 'Hola Mundo!';

let dobleDe = numero => numero * 2;

let suma = (a, b) => a + b;

let horaActual = () => {
    let fecha = new Date();
    return fecha.getHours() + ':' + fecha.getMinutes();
}
```

Arrow function **sin parámetros**.

Requiere de los paréntesis para iniciarse.

Al tener una sola línea de código, y que esta misma sea la que quiero retornar, el **return** queda **implícito**.

{ código }

```
let saludo = () => 'Hola Mundo!';

let dobleDe = numero => numero * 2;

let suma = (a, b) => a + b;

let horaActual = () => {
    let fecha = new Date();
    return fecha.getHours() + ':' + fecha.getMinutes();
}
```

Arrow function **con un único parámetro** *(no necesitamos los paréntesis para indicarlo)* y con un return **implícito**.

{ código }

```
let saludo = () => 'Hola Mundo!';

let dobleDe = numero => numero * 2;

let suma = (a, b) => a + b;

let horaActual = () => {
    let fecha = new Date();
    return fecha.getHours() + ':' + fecha.getMinutes();
}
```

Arrow function **con dos parámetros**.

Necesita de los paréntesis y con un return **implícito**.

{ código }

```
let saludo = () => 'Hola Mundo!';

let dobleDe = numero => numero * 2;

let suma = (a, b) => a + b;

let horaActual = () => {
    let fecha = new Date();
    return fecha.getHours() + ':' + fecha.getMinutes();
}
```

Arrow function sin parámetros y con un return **explícito**.

En este caso hacemos uso de las llaves y el return ya que la lógica de esta función se desarrolla en más de una línea de código.