

A yellow square containing the letters 'JS' in a large, bold, black sans-serif font.

JS

JAVASCRIPT

Tema 3 – Programación avanzada

JAVASCRIPT

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

■ Funciones

- Palabra reservada ***function***
- Nombre de la función
- Retorno de la función: ***return***
- Argumentos de la función
 - **Argumentos con valor por defecto**
- Invocación de una función
- Asignación de una función a una variable
- Uso de la partícula ***this***.
- Funciones anónimas
- Variable argumentes → Recoge todos los argumentos de una función.

JAVASCRIPT

JS

■ Funciones 'flecha'

- Funciones sin nombre.
- Aplicables a map, filter,

Sintaxis:

```
param => expression;
```

```
(param) => expression;
```

```
(param1, paramN) => expression;
```

```
(param) => {  
  let a = 1;  
  return a + b;  
};
```

JAVASCRIPT

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

■ Funciones 'flecha'

Sintaxis (continuación):

```
(param1, paramN) => {  
  let a = 1;  
  return a + b;  
};
```

```
(params) => ({ foo: "a" });
```

```
(a = 400, b = 20, c) => expression;
```

JAVASCRIPT

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

- Parámetros ***rest***.
 - Permiten que una función reciba un número indeterminado de argumentos.
 - Deben ir al final.
 - `function nombreFuncion(parametro1, ...parametroRest)`
 - Se recorren con for-of.
 - Se acceden por posición con [índice] (como un array).

JAVASCRIPT

JS

- Sintaxis extendida **spread**.

- Convierte un array en una relación de argumentos.

```
function suma(x,y) {  
    console.log(x+y);  
}
```

```
let datos = [5,2]
```

```
suma(...datos)
```

- Se pueden usar para crear arrays:

```
let datos = [5,2]
```

```
let resultado = [1, 5, ...datos, 8]
```

JAVASCRIPT

■ Funciones generadoras. **function***

- Se ejecutan en cada llamada.
- Devuelve un objeto generator.
- Operador **yield**.

```
function* generador(index) {  
    while (index < 5) {  
        yield index;  
        index++;  
    }  
}  
  
const iterator = generador(0);  
let valor;  
while ((valor = iterator.next().value) != null) {  
    console.log(valor);  
}
```

JAVASCRIPT

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

- Ámbito de las variables: local vs global.
- Ámbito de dentro de las funciones: variables locales y argumentos.
- Variables vs constantes.
- Bloques de código.
- Elevación o *hoisting*.

JAVASCRIPT



- Sentencias break y continue:
 - (Ver estructuras de control en UD02).

JAVASCRIPT

JS

▪ Excepciones:

- try
- catch
 - Uso cambiando con instanceof
- finally
- Anidación de try-catch.
- Sentencia throw.

JAVASCRIPT

JS

- Enlaces:
- <https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Functions>
- <https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions>
- [https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow functions](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow_functions)
- [https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/rest parameters](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/rest_parameters)
- [https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Spread syntax](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Spread_syntax)
- https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Statements/function*
- [https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Grammar and types](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Grammar_and_types)
- [https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Control flow and error handling](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Control_flow_and_error_handling)
- [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global Objects/Error](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Error)