

Paràmetres en les funcions

Els paràmetres a les funcions

El pas de paràmetres a una funció en JavaScript es fa de manera flexible. El llenguatge gestiona els paràmetres utilitzant la llista definida en la funció, però també ofereix mecanismes per definir valors per defecte i per capturar un nombre indefinit de paràmetres.

Paràmetres Obligatoris

En JavaScript, tècnicament, **tots els paràmetres són opcionals** per naturalesa del llenguatge, ja que no hi ha una sintaxi que obligui la seva presència.

- Si un paràmetre es defineix a la funció però **no es passa** en la crida, el seu valor serà automàticament `undefined`.

Mètode per Simular un Paràmetre Obligatori

Per assegurar que un paràmetre important no s'omet, s'utilitza una comprovació explícita (sovint llançant un error) al principi de la funció.

JavaScript

```
function crearUsuari(nom, email) {  
  // Comprovació manual per simular l'obligatorietat  
  if (nom === undefined) {  
    throw new Error("El paràmetre 'nom' és obligatori.");  
  }  
  if (email === undefined) {  
    console.warn("L'email no s'ha proporcionat. Es pot considerar un error o un valor per defecte.");  
  }  
  
  // Lògica de la funció...  
  console.log(`Usuari creat: ${nom}, Email: ${email}`);  
}  
  
// Ús:
```

```
crearUsuari("Anna");           // Sortida: Advertència (WARN), Usuari creat: Anna, Email:
undefined
// crearUsuari();              // Llança error!
```

Paràmetres Opcionals (Valors per Defecte)

Els paràmetres opcionals reben un valor per defecte quan el cridador de la funció els omet o els passa explícitament com a `undefined`. Aquesta funcionalitat es va introduir a **ES6 (ECMAScript 2015)**.

Sintaxi

Simplement assignes un valor al paràmetre a la declaració de la funció.

JavaScript

```
function obtenirPreu(producte, descompte = 0, moneda = 'EUR') {
  const preuFinal = producte.preu * (1 - descompte);
  console.log(`Preu de ${producte.nom}: ${preuFinal.toFixed(2)} ${moneda}`);
}

const llibre = { nom: "La Roda del Temps", preu: 25 };

// Cas 1: Només producte (usa valors per defecte)
obtenirPreu(llibre);
// Sortida: Preu de La Roda del Temps: 25.00 EUR

// Cas 2: Amb descompte (usa moneda per defecte)
obtenirPreu(llibre, 0.15);
// Sortida: Preu de La Roda del Temps: 21.25 EUR

// Cas 3: Tots els paràmetres
obtenirPreu(llibre, 0.20, 'USD');
// Sortida: Preu de La Roda del Temps: 20.00 USD
```

Paràmetres Indefinits (Paràmetres Rest)

Per gestionar un nombre variable (indefinit) de paràmetres, s'utilitzen els **Paràmetres Rest** (`...`), una altra característica d'ES6.

La sintaxi `...nomVariable` recull tots els arguments restants que no han estat assignats a paràmetres individuals i els empaqueta en un sol **array**.

Sintaxi

El paràmetre rest s'ha de col·locar **al final** de la llista de paràmetres de la funció.

JavaScript

```
function afegirElements(elementPrincipal, ...elementsAdicionals) {
  console.log(`Element principal: ${elementPrincipal}`);

  if (elementsAdicionals.length > 0) {
    console.log("Elements afegits com a array:");
    console.log(elementsAdicionals); // És un Array!
    return elementPrincipal + ' + ' + elementsAdicionals.join(' + ');
  } else {
    return elementPrincipal;
  }
}

// Cas 1: Dos paràmetres
afegirElements('Poma', 'Plàtan');
// elementPrincipal = 'Poma'
// elementsAdicionals = ['Plàtan']

// Cas 2: Diversos paràmetres
const resultat = afegirElements('Cafè', 'Llet', 'Sucre', 'Xocolata');
// elementPrincipal = 'Cafè'
// elementsAdicionals = ['Llet', 'Sucre', 'Xocolata']
console.log(`Resultat final: ${resultat}`);
// Sortida: Resultat final: Cafè + Llet + Sucre + Xocolata
```

L'Objecte arguments

Abans d'ES6, els desenvolupadors utilitzaven l'objecte especial `arguments` (accessible dins de qualsevol funció) per accedir a tots els paràmetres passats. No obstant això, `arguments` **no és un array real** i és menys flexible que els Paràmetres Rest. Els **Paràmetres Rest (...)** són la **pràctica moderna i preferida** per a gestionar un nombre indefinit d'arguments.

```
function mostrarArguments(a, b) {
  console.log("--- Objecte arguments ---");
  console.log(arguments);
  // Sortida: Objecte similar a un Array que conté [10, 20, 30, 40]

  console.log(`Primer argument definit (a): ${a}`);
  // Sortida: Primer argument definit (a): 10

  console.log(`Tots els arguments passats:`);
```

```
// Per recórrer arguments s'ha d'utilitzar un bucle for tradicional
for (let i = 0; i < arguments.length; i++) {
  console.log(`Argument ${i}: ${arguments[i]}`);
}

mostrarArguments(10, 20, 30, 40);

// Si passem menys arguments, el comportament és el mateix:
// mostrarArguments('hola');
// Sortida: Argument 0: hola. Arguments[1] seria undefined, però no falla.
```

```
x = findMax(1, 123, 500, 115, 44, 88);

function findMax() {
  var i;
  var max = -Infinity;
  for (i = 0; i < arguments.length; i++) {
    if (arguments[i] > max) {
      max = arguments[i];
    }
  }
  return max;
}
```

Tot i que aquesta funció és vàlida i funcional, el codi modern de JavaScript utilitzaria mètodes més nets i concisos, com els Paràmetres Rest (...) o el mètode `Math.max()`. El mètode `Math.max()` pot acceptar una llista d'arguments.

```
function findMaxModern(...nombres) {
  // Els Paràmetres Rest recullen els arguments en un Array real
  return Math.max(...nombres);
}

let y = findMaxModern(1, 123, 500, 115, 44, 88);
// y serà 500
```

🕒 Revisió núm. 2

★ Admin l'ha creat 2025-09-25 11:35:38 UTC

✎ Admin l'ha actualitzat 2025-09-25 11:36:11 UTC