

# JavaScript Fundamentals - Q&A

## 1. Variabile și Tipuri de Date

 Care este diferența dintre var, let și const?

 A:


- var → **Are hoisting**, dar **nu are block scope**. Evită folosirea sa.
- let → Are **block scope**, poate fi reassignată, dar nu redeclarată în același scope.
- const → Are **block scope**, **nu poate fi reassignată** și trebuie inițializată.

 Exemplu:

```
if (true) {  
  var x = 10;  
  let y = 20;  
  const z = 30;  
}
```

console.log(x); //  10 (var nu respectă block scope)

console.log(y); //  ReferenceError (y are block scope)

console.log(z); //  ReferenceError (z are block scope)

---

 Ce tipuri de date primitive există în JavaScript?

 A:

1. String
2. Number
3. Boolean
4. Undefined
5. Null
6. BigInt
7. Symbol

 Exemplu:

```
let a = "Hello"; // String
let b = 42;      // Number
let c = true;    // Boolean
let d;           // Undefined
let e = null;    // Null
let f = 9007199254740991n; // BigInt
let g = Symbol("unique"); // Symbol
```

---

### ? Cum verific tipul unei variabile în JavaScript?

✓ A: Folosind `typeof`

```
console.log(typeof 42); // "number"
console.log(typeof "text"); // "string"
console.log(typeof undefined); // "undefined"
console.log(typeof null); // "object" (bug în JavaScript)
```

---

### ? Ce se întâmplă dacă adun un număr cu un string? (5 + "5")

✓ A: Se face **concatenare**!

```
console.log(5 + "5"); // "55" (numărul 5 devine string)
console.log("5" - 2); // 3 (string-ul "5" este convertit în număr)
```

---

## 2. Operatori și Expresii

### ? Care este diferența dintre `==` și `===`?

✓ A:

- `==` → Compară **valoarea** (face conversie de tip implicit).
- `===` → Compară **valoarea + tipul** (mai sigur).

✦ Exemplu:

```
console.log(5 == "5"); // ✓ true (convertire automată)
console.log(5 === "5"); // ✗ false (tipuri diferite)
```

---

? Ce face operatorul && și || în JavaScript?

✓ A:

- && → Returnează **primul fals** sau **ultimul adevărat**.
- || → Returnează **primul adevărat**.

✚ Exemplu:

```
console.log(true && false); // false
```

```
console.log(true || false); // true
```

```
console.log(0 || "Hello"); // "Hello" (0 e fals)
```

```
console.log(0 && "Hello"); // 0 (se oprește la primul fals)
```

---

? Cum funcționează operatorul ternar?

✓ A:

condiție ? valoare\_dacă\_true : valoare\_dacă\_false

✚ Exemplu:

```
let mesaj = (10 > 5) ? "Corect!" : "Greșit!";
```

```
console.log(mesaj); // "Corect!"
```

---

? Ce se întâmplă când fac console.log(undefined + 1)?

✓ A:

```
console.log(undefined + 1); // NaN (Not a Number)
```

undefined nu poate fi convertit într-un număr valid.

---

### 3. Funcții

? Ce este hoisting-ul și cum afectează funcțiile?

✓ A: JavaScript mută **declarațiile** de variabile și funcții în partea de sus a scope-ului.

✚ Exemplu:

```
hello(); // ✓ Funcționează!
```

```
function hello() {  
  console.log("Salut!");  
}
```

💡 **Dar cu var sau let, hoisting-ul diferă:**

```
console.log(x); // ❌ Undefined  
  
var x = 5;
```

---

❓ **Care este diferența dintre o funcție declarată și o funcție anonimă?**

✅ **A:**

- **Funcție declarată:** Poate fi apelată înainte de definiție.
- **Funcție anonimă:** Este salvată într-o variabilă.

📌 **Exemplu:**

```
function declarata() {  
  console.log("Funcție declarată");  
}
```

```
const anonima = function() {  
  console.log("Funcție anonimă");  
};
```

---

❓ **Ce sunt arrow functions și când le folosim?**

✅ **A:** Arrow functions sunt o sintaxă scurtă pentru funcții. **Nu au this propriu.**

📌 **Exemplu:**

```
const suma = (a, b) => a + b;  
  
console.log(suma(2, 3)); // 5
```

---

**? Ce este un callback function și de ce este util?**

**✓ A:**

- O funcție **pasată ca argument** la altă funcție.
- Folosită în cod asincron.

**📌 Exemplu:**

```
function salut(nume, callback) {  
  console.log("Salut, " + nume);  
  callback();  
}  
  
salut("Ana", () => console.log("Callback apelat!"));
```

---

#### 4. Obiecte și Array-uri

**? Cum creez un obiect în JavaScript și cum accesez valorile din el?**

**✓ A:**

```
let persoana = { nume: "Ion", varsta: 30 };
```

```
console.log(persoana.nume); // "Ion"
```

```
console.log(persoana["varsta"]); // 30
```

---

**? Cum pot șterge o proprietate dintr-un obiect?**

**✓ A:**

```
delete persoana.varsta;
```

---

**? Cum funcționează forEach, map, filter și reduce?**

**✓ A:**

```
let numere = [1, 2, 3, 4, 5];
```

```
// forEach → Execută o acțiune pe fiecare element
```

```
numere.forEach(n => console.log(n));
```

// map → Creează un nou array transformând fiecare element

```
let dublate = numere.map(n => n * 2);
```

// filter → Returnează doar elementele care respectă condiția

```
let pare = numere.filter(n => n % 2 === 0);
```

// reduce → Reduce array-ul la o singură valoare

```
let suma = numere.reduce((total, n) => total + n, 0);
```

---

 **Ce este localStorage și cum îl folosesc?**

 **A:**

```
localStorage.setItem("user", "Alice");
```

```
console.log(localStorage.getItem("user")); // "Alice"
```

```
localStorage.removeItem("user");
```

```
localStorage.clear();
```

## 5. Controlul Fluxului

 **Cum funcționează un for loop comparativ cu un while loop?**

 **A:**

- for → Se folosește când **știm de câte ori vrem să repetăm** codul.
- while → Se folosește când **nu știm exact de câte ori se va executa**.

 **Exemplu:**

// for loop (execută exact 5 iterații)

```
for (let i = 0; i < 5; i++) {
```

```
  console.log(i);
```

```
}
```

// while loop (execută până când condiția devine falsă)

```
let x = 0;
while (x < 5) {
  console.log(x);
  x++;
}
```

---

**? Ce face break și continue într-un loop?**

**✓ A:**

- break → **iese complet** din loop.
- continue → **Sari peste iterația curentă** și mergi la următoarea.

**✦ Exemplu:**

```
for (let i = 0; i < 5; i++) {
  if (i === 2) continue; // Sari peste i = 2
  if (i === 4) break; // Oprește complet la i = 4
  console.log(i);
}
```

**Output:** 0, 1, 3

---

**? Care este diferența dintre try...catch și if...else pentru tratarea erorilor?**

**✓ A:**

- if...else este folosit pentru **verificări logice**.
- try...catch este folosit pentru **prinderea și tratarea erorilor neașteptate**.

**✦ Exemplu:**

```
try {
  let rezultat = numar * 10; // ✗ `numar` nu este definit!
} catch (error) {
```

```
console.log("A apărut o eroare:", error.message);  
}
```

---

## 6. DOM & Evenimente

**? Cum selectez un element HTML în JavaScript?**

**✓ A:**

```
document.getElementById("id");  
document.querySelector(".clasa");  
document.querySelectorAll("div");
```

---

**? Care este diferența dintre .innerHTML și .textContent?**

**✓ A:**

- .innerHTML → **Poate interpreta HTML.**
- .textContent → Returnează doar **text simplu.**

**✚ Exemplu:**

```
let div = document.querySelector("#test");
```

```
div.innerHTML = "<b>Salut!</b>"; // Apare îngroșat
```

```
div.textContent = "<b>Salut!</b>"; // Afișează exact "<b>Salut!</b>"
```

---


**? Cum adaug și elimin evenimente folosind addEventListener?**


**✓ A:**

```
let buton = document.querySelector("button");
```

```
function alerta() {  
    alert("Ai apăsat pe buton!");  
}
```



buton.addEventListener("click", alerta); //  Adaugă eveniment

buton.removeEventListener("click", alerta); //  Elimină eveniment

---

### ? Ce este propagarea evenimentelor (event bubbling & capturing)?

 A:

- **Bubbling** → Evenimentul **se propagă de la copil la părinte** (default).
- **Capturing** → Evenimentul **se propagă de la părinte la copil**.

 Exemplu:

```
document.querySelector("#parinte").addEventListener("click", () => {  
  console.log("Părinte");  
}, true); // `true` = capturing
```

```
document.querySelector("#copil").addEventListener("click", () => {  
  console.log("Copil");  
}, false); // `false` = bubbling
```

---

## 7. Asincronism și Promisiuni

### ? Ce este event loop-ul în JavaScript?

 A:

- **Mecanismul care gestionează execuția asincronă** în JavaScript.
- Mută codul din **callback queue** în **call stack**.

 Exemplu:

```
console.log("1");  
setTimeout(() => console.log("2"), 0);  
console.log("3");
```

**Output:**

CopyEdit

3

2

(Callback-ul setTimeout rulează **după** codul sincron!)

---

 **Care este diferența dintre setTimeout și setInterval?**

 **A:**


- setTimeout(func, timp) → Execută **o singură dată** după timp ms.
- setInterval(func, timp) → Execută **repetitiv** la fiecare timp ms.

 **Exemplu:**

```
setTimeout(() => console.log("O dată"), 1000);
```

```
setInterval(() => console.log("Repetitiv"), 2000);
```

---

 **Ce este o promisiune (Promise) și cum funcționează .then() și .catch()?**

 **A:**

- O promisiune reprezintă **o operație asincronă care poate avea succes sau eșua**.

 **Exemplu:**

```
let promisiune = new Promise((resolve, reject) => {  
  setTimeout(() => resolve("Succes!"), 2000);  
});
```

promisiune

```
.then(rezultat => console.log(rezultat)) // "Succes!"  
.catch(eroare => console.log(eroare));
```

---

 **Cum funcționează async/await și care sunt avantajele față de promisiuni?**

 **A:**

- **Face codul asincron să arate mai curat** (fără .then()).
- await **așteaptă** finalizarea unei promisiuni înainte de a continua.

📌 **Exemplu:**

```
async function getData() {  
  let rezultat = await promisiune;  
  console.log(rezultat);  
}
```

```
getData(); // "Succes!" după 2 secunde
```

---

## 8. Miscellaneous (Diverse)

❓ **Ce este strict mode în JavaScript și cum îl activez?**

✅ **A:**

- Previne erori comune și îmbunătățește performanța.
- Se activează cu "use strict"; la începutul fișierului.

📌 **Exemplu:**

```
"use strict";  
  
x = 10; // ❌ ReferenceError: x is not defined
```

---

❓ **Cum pot preveni comportamentul implicit al unui formular în JavaScript?**

✅ **A:**

```
document.querySelector("form").addEventListener("submit", (event) => {  
  event.preventDefault();  
});
```

---

❓ **Cum funcționează destructurarea obiectelor și array-urilor?**

✅ **A:**

```
let { nume, varsta } = { nume: "Ana", varsta: 25 };  
console.log(nume, varsta); // "Ana", 25
```

```
let [primul, alDoilea] = [10, 20];  
console.log(primul, alDoilea); // 10, 20
```

---

 **Ce este closure și cum funcționează în JavaScript?**

 **A:**

- Un closure apare când o funcție **își amintește variabilele din scope-ul părinte**.

 **Exemplu:**

```
function salut(ume) {  
  return function() {  
    console.log("Salut, " + ume);  
  };  
}
```

```
let mesaj = salut("Alex");  
mesaj(); // "Salut, Alex"
```

---

 **Ce este localStorage și cum îl folosesc?**

 **A:**

```
localStorage.setItem("user", "John");  
console.log(localStorage.getItem("user")); // "John"  
localStorage.removeItem("user");  
localStorage.clear();
```