Resurse suplimentare
pe care recomand să le parcurgi

Tipuri de Date Primitive

JavaScript are șapte tipuri de date primitive:

1. Number

Numerele în JavaScript pot fi întregi sau cu virgulă mobilă (floating-point).

```
let integer = 42
let floatingPoint = 3.14
let negative = -10
let scientific = 2.5e6 // 2.5 * 10^6 = 2500000
```

Valori numerice speciale:

```
let infinity = Infinity
let minusInfinity = -Infinity
let notANumber = NaN // Not a Number - rezultat al operațiilor invalide
```

2. String

Șirurile de caractere pot fi definite folosind ghilimele simple, duble sau backticks (pentru template strings):

```
let simpleText = 'Acesta este un șir de caractere.'
let doubleText = "Și acesta este un șir de caractere."
let templateString = `Valoarea variabilei integer este: ${integer}`
```

Operații comune cu șiruri:

```
let firstName = 'Ana'
let lastName = 'Maria'
let fullName = firstName + ' ' + lastName // Concatenare
let length = fullName.length // Lungimea șirului
let uppercase = fullName.toUpperCase() // Convertire la majuscule
```

3. Boolean

Tipul Boolean are doar două valori: `true` și `false`.

```
let truth = true
let falsehood = false
let comparison = 5 > 3 // true
```

4. undefined

O variabilă care a fost declarată dar nu a fost inițializată are valoarea `undefined`.

```
let uninitializedVar
console.log(uninitializedVar) // undefined
```

5. null

Valoarea `null` reprezintă absența deliberată a oricărei valori sau a unui obiect.

```
let emptyValue = null
```

6. Symbol (ES6)

Symbols sunt valori unice și imutabile, folosite adesea ca identificatori pentru proprietăți de obiecte.

```
let uniqueSymbol = Symbol('descriere')
```

7. BigInt (ES2020)

Pentru numere întregi mai mari decât limita standard de 2^53 - 1.

```
let veryLargeNumber = 9007199254740991n // 'n' la sfârșitul numărului indică BigInt
```

Verificarea Tipului de Date

Poți afla tipul unei valori folosind operatorul `typeof`:

```
typeof 42 // "number"
typeof 'text' // "string"
typeof true // "boolean"
typeof undefined // "undefined"
typeof null // "object" (aceasta este o anomalie în JavaScript)
typeof Symbol() // "symbol"
typeof 10n // "bigint"
```

Conversii între Tipuri de Date

Conversie la String

```
let number = 42
let numberText = String(number) // "42"
// sau
numberText = number.toString() // "42"
// sau
numberText = number + '' // "42"
```

Conversie la Number

```
let text = '42'
let number = Number(text) // 42
// sau
number = +text // 42
// sau
number = parseInt(text, 10) // 42 (baza 10)
number = parseFloat('3.14') // 3.14
```

Conversie la Boolean

```
let anyValue = 'text'
let bool = Boolean(anyValue) // true

// Valorile "falsy" (care devin false): 0, "", NaN, null, undefined, false
Boolean(0) // false
Boolean('') // false
Boolean(NaN) // false
Boolean(null) // false
Boolean(undefined) // false
```

Operatori în JavaScript

Operatori Aritmetici

```
let a = 10
let b = 3

let sum = a + b // 13
let difference = a - b // 7
let product = a * b // 30
let quotient = a / b // 3.3333...
let remainder = a % b // 1 (restul împărțirii)
let power = a ** b // 1000 (10 la puterea 3)

// Incrementare și decrementare
let c = 5
c++ // c devine 6 (post-incrementare)
++c // c devine 7 (pre-incrementare)
c-- // c devine 6 (post-decrementare)
--c // c devine 5 (pre-decrementare)
```

Operatori de Atribuire

```
let x = 10
x += 5 // x = x + 5 (x devine 15)
x -= 3 // x = x - 3 (x devine 12)
x *= 2 // x = x * 2 (x devine 24)
x /= 4 // x = x / 4 (x devine 6)
x %= 4 // x = x % 4 (x devine 2)
x **= 3 // x = x ** 3 (x devine 8)
```

Operatori de Comparație

```
let a = 5
let b = '5'

// Comparații de valoare
a == b // true (compară doar valoarea, nu și tipul)
a != b // false

// Comparații stricte (verifică și tipul)
a === b // false (valori egale, dar tipuri diferite)
a !== b // true

// Comparații relaționale
a > 3 // true
a < 10 // true
a >= 5 // true
a <= 4 // false
```

Operatori Logici

```
let x = 5
let y = 10

// AND logic (&&) - returnează true doar dacă ambele condiții sunt true
x > 0 && y < 20 // true
x > 10 && y < 20 // false

// OR logic (||) - returnează true dacă cel puțin o condiție este true
x > 0 || y < 5 // true
x > 10 || y < 5 // false

// NOT logic (!) - inversează valoarea de adevăr
!(x > 0) // false
!(x > 10) // true

// Evaluare în scurt-circuit
let z = x > 0 && 'positive' // "positive"
let w = x < 0 || 'non-negative' // "non-negative"
```

Operatori de Șir (String)

```
let greeting = 'Bună'
let name = 'Ana'
let message = greeting + ' ' + name // "Bună Ana" (concatenare)

// Template literals (ES6)
let templateMessage = `${greeting} ${name}!` // "Bună Ana!"
```

Operatorul Conditional (Ternar)

```
let age = 20
let status = age >= 18 ? 'adult' : 'minor'
// Echivalent cu:
// if (age >= 18) {
//   status = "adult";
// } else {
//   status = "minor";
// }
```

Operatorul de Coalescență Nulă (Nullish Coalescing)

Introdus în ES2020, returnează primul operand dacă nu este `null` sau `undefined`, altfel returnează al doilea operand.

```
let user = null
let userName = user ?? 'Anonim' // "Anonim"

let number = 0
let value = number ?? 10 // 0 (pentru că 0 nu este null sau undefined)
```

Precedența Operatorilor

Operatorii sunt evaluați într-o anumită ordine de precedență. Poți folosi paranteze pentru a schimba această ordine:

```
let result = 2 + 3 * 4 // 14 (înmulțirea are precedență)
let resultWithParentheses = (2 + 3) * 4 // 20 (parantezele schimbă ordinea de evaluare)
```