

## Introducere în JavaScript



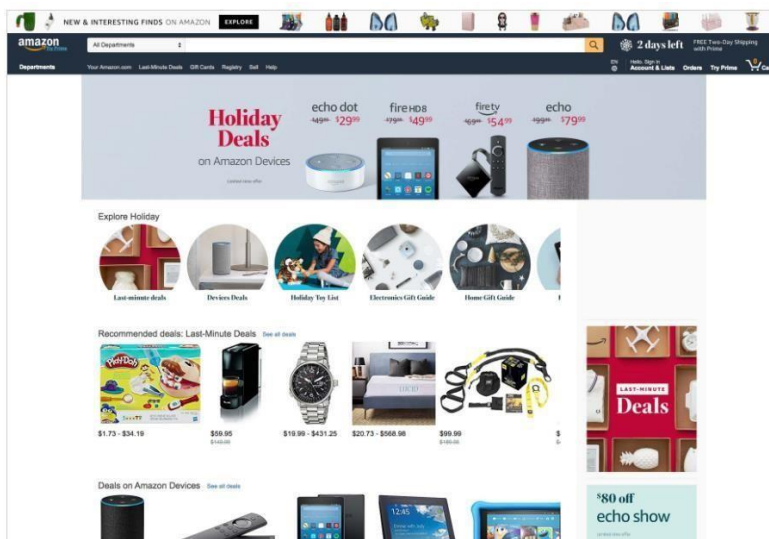
**JavaScript** este un limbaj de scriptare, apărut din nevoia ca logica și inteligența să fie și pe partea de client, nu doar pe partea de server.

*Limbajul HTML* oferă autorilor de pagini Web o anumită flexibilitate, dar statică. Documentele HTML nu pot interacționa cu utilizatorul în alt mod mai dinamic, decât prin a pune la dispoziția acestuia legături la alte resurse (URL-uri). Astfel, un pas important spre **interactivizare** a fost realizat cu JavaScript, care **permite inserarea în paginile web a script-urilor care se execută în cadrul paginii web**, mai exact în cadrul browser-ului utilizatorului, ușurând astfel și traficul dintre server și client. [Cum arată o pagină web fără JavaScript:](#)



YouTube Homepage

[Site-uri care funcționează bine fără limbajul JavaScript:](#)



## Inserarea limbajului JavaScript într-un fișier HTML

Codul JavaScript trebuie inserat între tag-urile `<script>` `</script>`.

⇒ **Extern** – ca fișier `.js`

```
1 <html>
2   <head>
3     <title>Titlul meu</title>
4     <script src="fisier.js"></script>
5   </head>
6
7   <body>
8
9   </body>
10 </html>
```

**Obs:** un fișier `.js` se poate scrie/edita în orice editor de text (Notepad, Notepad++, Sublime, Visual Studio Code etc.)

⇒ **Intern** – în interiorul tag-ului `<script>`

```
1 <html>
2   <head>
3     <title>Titlul meu</title>
4     <script type="text/javascript">
5       var a = 3;
6       console.log(a);
7     </script>
8   </head>
9
10  <body>
11
12  </body>
13 </html>
```

Se poate plasa o referință de script extern în `<head>` sau `<body>`. Scriptul se va comporta ca și când ar fi localizat exact unde e localizat tagul `<script>`.

⇒ **Inline** – în tag-ul în care se dorește acțiunea

```
1 <html>
2   <head>
3     <title>Titlul meu</title>
4   </head>
5
6   <body>
7     <button type="button" onclick="document.getElementById('demo').innerHTML = Date()">
8   </body>
9 </html>
```

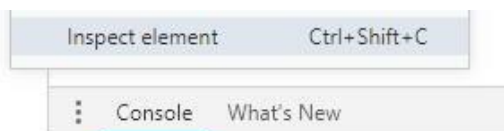
## Console

Consola este un panou care afișează mesaje importante, cum ar fi erorile, pentru dezvoltatori. O mare parte din munca pe care o face computerul cu codul scris este invizibilă în mod implicit.

În JavaScript, cuvântul cheie de **consolă** se referă la un obiect, o colecție de date și de acțiuni care se pot folosi în cod. O acțiune (o metodă) care este incorporată în obiectul consolă este metoda **log()**.

```
<script>
  var a = 3;
  console.log(a);
</script>
```

Pentru a vizualiza rezultatele codului JavaScript în consolă, se deschide pagina HTML într-un browser, se accesează „Inspect element/Inspectează” și apoi butonul „Console”.



## Comentarii

Comentariile pot explica ceea ce face codul, lasă instrucțiuni pentru dezvoltatori sau adaugă alte adnotări utile.

### Există două tipuri de comentarii de cod în JavaScript:

1. Un comentariu **pe o singură linie** - va comenta pe o singură linie și este notat cu două slash-uri înainte „//”.

```
3 // nume
4 console.log('Mihai');
5 // prenume
6 console.log('Roxana');
```

2. Un comentariu **pe mai multe linii** - va comenta mai multe linii și este notat cu „/\*” pentru a începe comentariul și cu „\*/” pentru a termina comentariul.

```
8 /* Nume si prenume
9 Oras
10 Facultate */
11 console.log('Mihai Roxana');
12 console.log('Bucuresti');
13 console.log('FIIR');
```

## Tipuri de date

Tipurile de date sunt *clasificările oferite diferitelor variabile* folosite în programare. În JavaScript, există **șapte tipuri de date fundamentale**:

1. **Number**: Orice număr, inclusiv cifre cu zecimale: 4, 1516, 23.42.
2. **String**: Orice grupare de caractere pe tastatură (litere, numere, spații, simboluri etc.) înconjurată de apostrof '...' sau ghilimele "...".
3. **Boolean**: Acest tip de date are doar două valori posibile: **true** sau **false**. Este util să vă gândiți la acest tip de date ca răspunsuri la o întrebare: "Da" sau "Nu".

4. **Null:** Acest tip de date reprezintă absența intenționată a unei valori și este reprezentat de cuvântul cheie **null**.
5. **Undefined:** Acest tip de date este notat cu cuvântul cheie **undefined** (nedefinit). De asemenea, reprezintă absența unei valori, deși are o utilizare diferită de cea nulă.
6. **Simbol:** O caracteristică mai nouă a limbajului - simbolurile sunt identificatori unici, utili în codificarea mai complexă.
7. **Obiect:** Colecții de date conexe.

Primele 6 dintre aceste tipuri sunt considerate tipuri de date primitive. Acestea sunt cele mai de bază tipuri de date în JavaScript. Obiectele sunt mai complexe.

## Operatori aritmetici

Un **operator** este un caracter care execută o sarcină în cod. JavaScript are mai multe funcționalități în operatori aritmetici, care permit efectuarea de calcule matematice cu numere. Acestea includ următorii operatori și simbolurile corespunzătoare:

1. **Adunare:** +
2. **Scădere:** -
3. **Înmulțire:** \*
4. **Împărțire:** /
5. **Rest:** %

**Retineți** că atunci când folosim **console.log()** calculatorul va evalua expresia din paranteze și va trimite acel rezultat la consolă. Dacă vrem să tipărim caracterele 3 + 4, punem expresia între ghilimele și le tipărim ca un șir.

```
3 console.log(3 + 4); // Rezultat: 7
4 console.log("3 + 4"); // Rezultat: 3 + 4
```

## Concatenarea șirurilor

**Operatorii nu sunt doar pentru numere!** Atunci când un operator „+” este utilizat pe două șiruri de caractere, acesta îmbină cele 2 șiruri. Acest proces de îmbinare a unui șir la altul se numește **concatenare**.

```
7 console.log('Unu' + ',' + 'doi' + ',' + 'trei' + ' ' + 'si patru');
8 // Rezultat: Unu,doi,trei si patru
```

## Proprietăți

Fiecare instanță șir are o proprietate numită lungime (**length**) care stochează numărul de caractere din acel șir. *Se pot prelua informații despre proprietăți adăugând șirul și numele proprietății.*

```
console.log('Hello'.length);
// Rezultat: 5
```

Un alt operator este `."`, îl numim, evident, **operatorul punct**. În exemplul de mai sus valoarea salvată la lungimea proprietății este extrasă din instanța șirului.

## Variabilele

În programare, o **variabilă** este un container pentru o valoare. Vă puteți gândi la variabile ca niște containere mici pentru informații care trăiesc în memoria calculatorului. Variabilele oferă, de asemenea, o *modalitate de etichetare a datelor* cu un nume descriptiv, astfel încât programele noastre pot fi înțelese mai clar de către cititor și de noi înșine.

Pe scurt, *variabilele etichetează și stochează datele în memorie*. Este important să se facă distincția că **variabilele nu sunt valori**.

```
10 var nume = 'Roxana';
11 var varsta = 23;
12 console.log(nume);    // Rezultat: Roxana
13 console.log(varsta);  // Rezultat: 23
```

### Există 3 tipuri de variabile în JavaScript:

1. **var** – cea mai utilizată
2. **let** – permite declararea de variabile care sunt limitate în expresia în care sunt utilizate
3. **const** – nu poate fi realocată, deoarece este o variabilă constantă. Dacă încercați să realocați o variabilă *const*, veți primi o eroare *TypeError*.

Modul în care se pot folosi variabilele și operatorii matematici pentru a calcula noi valori și a le atribui unei variabile:

```
16 var numar = 6;
17 console.log("Valoarea pentru numar este: " + numar);
18 numar = numar + 1;
19 console.log("Noul numar este: " + numar);
```

Un alt mod în care s-ar fi putut reasigna **numar** după ce se efectuează o operație matematică asupra lui este folosirea **operatorilor de atribuire matematică încorporați**. Astfel, linia de mai sus s-ar rescrie:

```
18 numar += 1; // aceeași valoare ca și numar = numar + 1
```

De asemenea, există și alți operatori de atribuire matematică: **-=**, **\*=**, și **/=** care funcționează într-un mod similar.

Operatorul **+** poate fi utilizat pentru a combina două valori de șir, chiar dacă aceste valori sunt stocate în variabile:

```
3 var animalFavorit = "Catel";
4 console.log("Animalul meu favorit este: " + animalFavorit);
```

În versiunea ES6 a JavaScript-ului, se pot insera (interpola) variabile în șiruri utilizând **placeholder**. Pentru exemplul de mai sus, o altă formă pentru cod este:

```
4 console.log(`Animalul meu favorit este: ${animalFavorit}`);
```

Se observă faptul că structura din **console.log()** este între ``...``, denumit **accent grav** sau mai poate fi găsit sub denumiri cum ar fi *acute*, *backtick*, *grave* etc. De obicei acest caracter este localizat în partea de sus a tastaturii, la stânga tastei 1. Unul dintre cele mai mari beneficii pentru utilizarea literalului de șabloane este ușurința în citirea codului.

## Operatorul typeof

În timp ce un cod este scris, poate fi utilă urmărirea tipurilor de date ale variabilelor din programul. Acest lucru se poate face prin intermediul operatorului **typeof**.

```
9 var universitate = "Universitatea POLITEHNICA Bucuresti";
10 var etaje = 5;
11 var deschis = true;
12 console.log(typeof universitate); // Rezultat: string
13 console.log(typeof etaje); // Rezultat: number
14 console.log(typeof deschis); // Rezultat: boolean
```

## Instrucțiunea IF (dacă – atunci)

Deseori se efectuează sarcini bazate pe o condiții. De exemplu:

- × **dacă** vremea este frumoasă astăzi, **atunci** vom ieși afară
- × **dacă** suntem obosiți, **atunci** vom merge la culcare

În programare, se poate efectua o sarcină bazată pe o condiție folosind o instrucțiune **if**.

```
16 if (true) {
17     console.log("Uraa");
18 }
```

## Instrucțiunea IF ... ELSE (dacă – atunci – altfel)

Condițiile, totuși, pot deveni mai complexe:

- × **dacă** vremea este frumoasă astăzi, **atunci** vom ieși afară, **altfel** rămânem în casă
- × **dacă** suntem obosiți, **atunci** vom merge la culcare, **altfel** ne uităm la film

Adăugarea unui comportament implicit la instrucțiunea **if**, se face prin adăugarea unei declarații **else**, pentru a rula un bloc de cod atunci când condiția evaluează *false*.

```
16 if (false) {
17     console.log("Codul din aceasta instructiune nu va functiona!");
18 }
19 else {
20     console.log("Dar condul din aceasta instructiune va functiona!")
21 }
22 // Rezultat: Dar condul din aceasta instructiune va functiona!
```

O instrucțiune **else** trebuie să fie asociată cu o instrucțiune **if** și împreună sunt denumite o declarație **if ... else**.

## Operatorii de comparare

În cadrul declarațiilor condiționale, sunt folosite diferite tipuri de operatori pentru a compara valori. Acești operatori sunt numiți **operatori de comparare**.

**Cei mai cunoscuți operatori de comparare și sintaxa lor:**

- Mai mic decât: <
- Mai mare decât: >
- Mai mic sau egal cu: < =
- Mai mare sau egal cu: > =
- Este egal ca valoare cu: = =
- Este egal ca valoare **ȘI** ca tip de dată cu: = = =
- NU este egal cu: !=
- NU este egal ca valoare **SAU** ca tip cu: != =

De asemenea, se pot utiliza operatori de comparare pe diferite tipuri de date, cum ar fi șiruri de caractere:

```

8  if('mere' === 'pere') {
9      console.log("Sunt aceleasi fructe");
10 }
11 else {
12     console.log("NU sunt aceleasi fructe");
13 }
14 // Rezultat: NU sunt aceleasi fructe

```

**Operatori logici**

În JavaScript, există operatori care lucrează cu valori booleene, cunoscuți ca **operatori logici**. Se folosesc operatorii logici pentru a adăuga o logică mai sofisticată condițiilor.

**Există 3 operatori logici:**

- operatorul **ȘI**: &&
- operatorul **SAU**: ||
- operatorul **NU**, negare: !

**Cum funcționează acești operatori?**

A	B	A && B	A    B	!A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Când se folosește operatorul **&&**, toate condițiile trebuie să fie adevărate pentru ca declarația globală să fie adevărată.

Când se folosește operatorul **||**, este suficient ca numai una dintre condiții să fie adevărată pentru ca declarația globală să fie adevărată.

Operatorul negare ! inversează sau neagă valoarea unui boolean:

```
8 var ninge = true;
9 console.log(!ninge); // Rezultat: false
```



[JavaScript Tutorial \(w3schools.com\)](https://www.w3schools.com)

[The Modern JavaScript Tutorial](#)

[Limbajul JavaScript \(ceiti.md\)](#)

## Exerciții laborator:

1. Realizează o pagină *Exercitiul\_1.html* în care:

- Creează o variabilă numită myName și setează valoarea acesteia cu prenumele tău.
- Creează o variabilă numită mySurname și setează valoarea acesteia cu numele tău.
- Creează o variabilă numită myAge și setează valoarea acesteia cu vârsta ta.
- Creează o variabilă numită studyYear și setează valoarea acesteia la 3.
- Creează o variabilă numită myHobby și valoarea acesteia cu un hobby de-al tău.
- Folosind operatorul += modifică valoarea variabilei myAge pentru a avea valoarea 25.
- Afișează în consolă următorul text, folosind atât operatorul de concatenare +, cât și placeholders:  
/\* Buna! Ma numesc NUME PRENUME și am VARSTA ani. Sunt la Universitatea Politehnica Bucuresti, în anul AN.  
Unul dintre hobby-urile mele este HOBBY. \*/
- Afișează în consolă tipul variabilei myName.
- Afișează în consolă tipul variabilei myAge.
- Afișează în consolă tipul variabilei myHobby.

2. Realizează o pagină *Exercitiul\_2.html* în care:

- Declară o variabilă numită tranzactie1 cu valoarea true.
  - Declară o variabilă numită tranzactie2 cu valoarea false.
  - Creează o instrucțiune dacă-atunci în care condiția este tranzactie1, iar în interiorul acesteia afișează textul "Tranzactie acceptata!"
  - Creează o instrucțiune dacă-atunci-altfel în care condiția este tranzactie2, iar:
    - a) în interiorul lui IF afișează textul "Tranzactie acceptată!"
    - b) în interiorul lui ELSE afișează textul "Tranzactie refuzată!"
- Describe comportamentul observat și explică de ce se întâmplă astfel. Descrierea se face aici, ca un comentariu.

3. Realizează o pagină *Exercitiul\_3.html* care să afișeze în consolă, în ordine descrescătoare, 3 numere introduse în cadrul codului.

*Exemplu:* se dau numerele: 15, 4, 12 ⇒ se afișează în consolă: 15, 12, 4



4. Realizează o pagină *Exercitiul\_4.html* care să rezolve următorul exercițiu:

- a. Se dau patru numere. Să se determine dacă ele ar putea reprezenta laturile unui dreptunghi, iar în caz afirmativ, să se determine aria și perimetrul.

*Exemplu1:* se dau numerele 25, 4, 25, 16  $\Rightarrow$  se afișează în consolă: Cele 4 numere nu pot reprezenta laturi ale unui dreptunghi!

*Exemplu2:* se dau numerele 13, 17, 13, 17  $\Rightarrow$  se afișează în consolă: Cele 4 numere pot reprezenta laturi ale unui dreptunghi! Perimetrul este: 60. Aria este: 221.

5. Realizează o pagină *Exercitiul\_5.html* care:

- a) Primește trei numere în cod (lungimile a 3 laturi)
- b) Verifică dacă cele 3 laturi pot forma un triunghi
- c) Dacă da, să afișeze ce tip de triunghi este:
  - **Echilateral** (toate laturile egale)
  - **Isoscel** (două laturi egale)
  - **Oarecare** (toate diferite)

**Obs:** numerele pentru exercițiile 3,4,5 nu se dau de la tastatură, ele vor fi scrise în cadrul codului.

**Instrucțiuni laborator:** Rezolvările exercițiilor se vor încărca, sub formă de arhivă (rar/zip) pe Moodle, până la finalul laboratorului.