# Jonas Schmedtmann „The Complete JavaScript Course 2022 - From Zero to Expert” [Udemy] (2021)

Toate mesajele de eroare sunt afişate în consolă automat, fără a fi nevoie de funcţia „console.log/()” – se recomandă urmărirea permanentă a consolei pentru depistarea lor.

Este afişat numai primul mesaj, pentru că el blochează tot – nu se mai execută nimic după el.

Convenţii:

* Notare standard în JavaScript pentru numele de variabile: camelCase.
* Cuvântul „name” este un fel de cuvânt rezervat şi nu se recomandă folosirea lui la numele unei variabile, deşi nu este sintactic greşit, dar, în unele cazuri, poate crea probleme.
* Numele unei variabile să nu înceapă cu o majusculă, deoarece acest tip de notare este folosit în JavaScript la programarea orientată pe obiecte.
* Cuvintele scrise toate cu majuscule sunt rezervate pentru constante, nu se folosesc pentru variabile.
* Numele de variabile trebuie să fie descriptive.

Tipuri de date:

* Simple (/primitive):
  1. Number: numere în virgulă mobilă.
  2. String: secvenţe de caractere între ghilimele simple sau duble, altfel JavaScript le confundă cu nume de variabile
  3. Boolean: tip logic,m care poate fi „true” sau „false”.
  4. Undefined: valoarea luată de o variabilă care nu a fost definită /(‘valoate goală’) – a fost declarată, dar nu i s-a atribuit o valoare.
  5. Null: înseamnă tot ‘valoate goală’), dar este folosit în circumstanţe diferite.
  6. Symbol: introdus în ES2015 ca o valoare care este unică şi nu poate fi modificată.
  7. BigInt: introdus în ES2020 pentru numere întregi prea mari pe care tipul „Number” nu le poate reprezenta.
* Complexe /(obiecte)

Caracteristică JavaScript: tipărirea dinamică se referă la faptul că tipurile de date sunt determinate automat la atribuirea unei valori, tipul de date stocat într-o variabilă nu trebuie definit manual.

În JavaScript, valoarea are un tip, nu variabila.

Variabilele stochează pur şi simplu valori care au un tip.

Se poate schimba uşor tipul unei valori conţinute de o variabilă, fără a mai folosi cuvântul „let” în faţă, prin o nouă atribuire, altfel sp .

„undefined” este şi valoarea şi tipul valorii.

Bug JavaScript: comanda „console.log(typeof null);” returnează „Object”, nu „null”.

Declarare variabile în JavaScript:

* var: modul vechi de declarare a variabilelor.
* let: introdus în ES2015 – este JavaScript modern pentru declararea de variabile care se pot schimba mai târziu în timpul execuţiei (mutate the variable) .
* const: introdus în ES2015 – este JavaScript modern pentru declararea de variabile care se nu pot schimba mai târziu în timpul execuţiei (immutable variable); nu se pot declara constante goale, fără o valoare iniţială.

Practică recomandată: cât mai puţine mutaţii de variabile sau schimbări de variabile.

Schimbarea de variabile introduc un potenţial de a crea bug-uri, erori în cod.

Se recomandă evitarea folosirii cuvântului cheie „var”.

let: block scoped.

var: function scoped.

Variabilele se pot inţializa şi fără cuvintele cheie „let”, „const” sau „var”, dar acest lucru determină crearea unei proprietăţi în obiectul global.

Variabilele ar trebui declarate întotdeauna.

Consola are acces la toate variabilele care rulează în tab-ul browser-ului curent.

Tabel cu precedenţa operatorilor:

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence>

Literalele şablon au fost introduse de ES2015.

Literalele şablon: între ghilimele simple înclinate la stânga /(backticks) - ` `.

Literalele şablon pot conţine şi nume de variabile între acolade şi precedate de caracterul „$”:

`${nume\_variabilă}’

La fel, ele pot conţine şi şiruri de caractere pe mai multe linii (multilinie), fără a fi nevoie de caracterul „linie nouă” – \n.

Între acolade se pot scrie şi expresii.

Scrierea unui emoji în Windows: tasta Windows + .

Conversie tipuri de date: explicită, manual de către utilizator, la vedere în cod – cu Number(), String(), Boolean ().

Tipul lui „NaN” este „number”, dar este un număr invalid – în urma „Number(‘papa’).

Valori logice (boolean) de fals nu sunt exact false iniţial, dar ele vor deveni „false” atunci când sunt convertite la „boolean”:

1. 0
2. ‘’
3. undefined
4. null
5. NaN

Celelalte valori vor fi convertite la valoarea logică de adevăr, „true”, dacă se încearcă conversia lor la „boolean”.

Un obiect gol are valoarea logică de „true”.

Nu se poate face conversia la „undefined” sau la „null”.

Coerciţie tipuri de date: implicită, automat de către JavaScript, în fundal.

În practică, conversia la „boolean” este întotdeauna implicită, nu explicită.

Tipuri de egalităţi:

* ==: operator de egalitate „loose”: efectuează coerciţia tipurilor
* ===: operator de egalitate strictă: nu efectuează coerciţia tipurilor

Regulă generală pentru un cod curat: evitarea folosirii operatorului de egalitate „loose”, ==.

Preluare date de la utilizator prin intermediul browser-ului: funcţia „prompt()” – returnează un şir de caractere.

Expresia: o bucată de cod care produce o valoare.

Instrucţiunea /(statement): o porţiune mai mare de cod care se execută şi care nu produce o valoare asociată ei.

Instrucţiunea este ca o propoziţie şi expresiile sunt cuvintele care alcătuiesc propoziţia.

De obicei, orice se termină cu „;” (punct şi virgulă) este o instrucţiunea.

Într-un literal şablon se pot insera numai expresii, dar nu instrucţiuni.

Operatorul ternar „(…) ? … : …” produce o valoare, deci este o expresie.

Operatorul ternar permite declararea condițională de variabile direct, fără declararea iniţială în afara blocului „if”.

Deoarece este o expresie, operatorul ternar se poate folosi şi în literali şablon.

Modul strict în JavaScript permite scrierea unui cod mai sigur şi se activează prin introducerea pe primul rând al scriptului a următoarei secvenţe – ea trebuie să fie prima instrucţiune din script:

'use strict';

Permite dezvoltatorilor să evite erorile accidentale:

* ne interzice să facem anumite lucruri;
* creează erori vizibile în consola dezvoltatorului, unde, în alte situaţii, JavaScript ar cădea doar pe neobservate;
* introduce o listă scurtă de nume de variabile care sunt rezervate pentru caracteristici care ar putea fi adăugate la limbaj în viitor.

Funcţii:

* declarare funcţii: se pot apela înaintea definirii în cod
* expresii de funcţii: asigură o structură frumoasă în care întâi trebuie definite toate funcţiile la începutul codului şi de abia apoi ele pot fi invocate – recomandat => codul este mai frumos şi mai structurat
* funcţii săgeată: introduse de ES2015, sunt o formă de expresii de funcții, în forma cea mai simplă, cu un parametru şi o singură linie de cod

Funcţiile sunt valori, nu un tip de date, deci se pot stoca în variabile.

O funcție este doar o valoare.

Se recomandă stocarea tuturor în variabile, valori şi funcţii.

Variabilele declarate cu „var” creează o proprietate în obiectul „window” global.