

# JavaScript

A JavaScript egy magas szintű, dinamikusan típusos programozási nyelv, amelyet gyakran kliensoldali webfejlesztésre használnak. Lehetővé teszi az interaktív és dinamikus weboldalak készítését, például űrlapellenőrzést, animációkat vagy aszinkron kommunikációt (HTTP). A JavaScript a böngészőben fut, és a DOM (Document Object Model) manipulálásával képes módosítani a weboldal tartalmát és viselkedését a felhasználói interakciók alapján. Modern fejlesztés során gyakran kombinálják HTML-lel és CSS-sel, valamint különböző keretrendszerekkel (pl. Angular, React, Vue.js, Svelte) a hatékonyabb fejlesztés érdekében.

## Beágyazás HTML dokumentumba

JavaScript kódok beágyazhatóak HTML dokumentumba, a `<script>` elem segítségével:

```
<sxh html> <!DOCTYPE html> <html lang="en"> <head>
```

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>JavaScript Demo</title>
```

```
</head> <body>
```

```
<h4>Hello World!</h4>
```

```
<script>
  // Ide írhatod a JavaScript kódod
  console.log("Hello, World!");
</script>
```

```
</body> </html> </sxh>
```

Ezen kívül külső fájlból is importálhatók:

```
<sxh html> <!DOCTYPE html> <html lang="en"> <head>
```

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>JavaScript Demo</title>
```

```
</head> <body>
```

```
<h4>Hello World!</h4>
```

```
<!-- JavaScript importálás külső fájlból -->
<script src="./app.js"></script>
```

```
</body> </html> </sxh>
```

## Fejlesztői konzol

A fejlesztői konzol (Developer Console) lehetővé teszi JavaScript kódok futtatását és hibakeresését a böngészőben. A leggyakrabban használt funkciók közé tartozik a `console.log()` függvény, amely segítségével üzeneteket (pl. változók tartalmát) tudunk kiírni a konzolra.

A legtöbb böngészőben az F12 gomb lenyomásával érhetjük el.

## Változók

A változók lehetőséget adnak az értékek tárolására és későbbi felhasználásra. A `let` kulcsszó használata esetén a változó a későbbiekben kaphat más értéket, míg `const` kulcsszó esetén az érték később nem változtatható.

```
<sxh javascript> let x = 10;
```

```
const pi = 3.14; </sxh>
```

## Primitív adattípusok

JavaScriptben különböző primitív adattípusok állnak rendelkezésre:

- `string`: szöveges adat
- `number`: szám (egész vagy lebegőpontos)
- `boolean`: logikai érték (igaz/hamis)
- `null`: üres érték
- `undefined`: a változó deklarált, de nem kapott még értéket

```
<sxh javascript> let name = 'Béla'; string let age = 15; number let registered = true; boolean let email = null; let phoneNumber = undefined; </sxh> ===== Elágazások ===== Az elágazások segítségével a program különböző utakon haladhat előre, a megjelölt feltételek teljesülése esetén. <sxh javascript> let age = 20; if (age < 18) { console.log("Kiskorú"); } else if (age >= 18 && age < 65) { console.log("Felnőtt"); } else { console.log("Nyugdíjas"); } </sxh> ===== Ciklusok ===== A ciklusok lehetővé teszik a kód ismételt végrehajtását, amíg egy adott feltétel teljesül. <sxh javascript> for ciklus: számok kiírása 0-4-ig for (let i = 0; i < 5; i++) {
```

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

```
}
```

```
while ciklus: 5! kiszámítása let factorial = 1; let i = 1; while (i <= 5) { factorial *= i; i++; } console.log(factorial); </sxh> ===== Tömbök ===== A tömbök olyan adatszerkezetek, amelyek több elemet képesek tárolni egyetlen változóban. Ezek típusa nem kötött, egy tömb akár többféle adattípussal rendelkező értéket is tárolhat. <sxh javascript> const values = [ "red", "green", "blue", 0, 5, true ]; </sxh> ===== Tömb bejárása ciklusokkal ===== for-in ciklus == A for-in ciklus a tömb indexeit járja be: <sxh javascript> for (const index in values) { console.log(`Index: ${index}, Érték: ${values[index]} `); } </sxh> == for-of ciklus == A for-of ciklus közvetlenül a tömb elemeit járja be: <sxh javascript> for (const value of values) { console.log(`Érték: ${value}`); } </sxh> =====
```

Transzformációs metódusok === map metódus == A map metódus egy új tömböt hoz létre az eredeti tömb elemeinek módosításával: `<sxh javascript> const values = [ 'red', 'green', 'blue' ]; const uppercasedValues = values.map(value => value.toUpperCase()); console.log(uppercasedValues); </sxh>` == filter metódus == A filter metódus egy új tömböt hoz létre az eredeti tömb azon elemeiből, amelyek megfelelnek egy feltételnek: `<sxh javascript> const numbers = [1, 156, 1499, 142, 111, 42]; const evenNumbers = numbers.filter(value => value % 2 == 0); console.log(evenNumbers); </sxh>` ===== Függvények ===== A függvények valamilyen újrafelhasználható logikát tartalmaznak. Lehetnek olyan eljárások, amik csak végrehajtanak egy feladatot, vagy akár olyanok is, amik visszatérési értékkel is rendelkezhetnek. `<sxh javascript> function greet(name) { console.log("Hello, " + name + "!"); } greet("World");` Hello, World!

```
function square(number) {
```

```
  return number * number;
```

```
}
```

`console.log(square(2));` 4 `</sxh>` ===== Objektumok ===== Az objektumok kulcs-érték párokat tárolnak. A kulcsok szövegek (*name*, *age*, *male*, *greet*), míg az értékek valamely primitív (pl. *number*, *string*, *boolean*, *null*, *undefined*) vagy összetett (pl. *Object*, *Array*, *Function*) adattípusba tartozhatnak. `<sxh javascript> let person = { name: "John", age: 30, male: true, greet: function() { console.log(`Hi, my name is ${this.name}!`); } }; </sxh>` ===== Feladat ===== Készíts egy JavaScript programot, amely egy **számkitaláló játékot** valósít meg! A program véletlenszerűen generál egy számot 0 és 1 000 000 között, majd a játékos próbálja kitalálni azt. A tippek maximális száma 20. A program minden próbálkozás után visszajelzést ad: \* „**XY. tipp nem talált: A megoldás kisebb.**”, ha a tipp nagyobb a keresett számnál. \* „**XY. tipp nem talált: A megoldás nagyobb.**”, ha a tipp kisebb a keresett számnál. \* „**Gratulálok, XY lépésből eltaláltad!**”, ha a tipp helyes. \* „**Sajnos ez most nem sikerült! A megoldás: XYZ**”, ha a játékos elérte a maximális próbálkozások számát, de nem találta el a számot. A játék akkor ér véget, amikor a játékos eltalálja a számot vagy eléri a tippek maximális számát. \* Véletlenszám generáláshoz használd a [Math.random\(\)](#) függvényt! \* Az üzenetek kiírásához használd az [alert\(\)](#), a tipp bekéréséhez a [prompt\(\)](#) és [parseInt\(\)](#) függvényeket. \* Ellenőrizd, hogy a felhasználó által megadott érték valóban szám-e! Amennyiben nem, a [parseInt\(\)](#) függvény NaN (Not a Number) értékkel tér vissza, melyet az [isNaN\(\)](#) függvénnyel szűrhetsz ki. Elkészült megoldásodat töltsd fel a GitHub repository-dba!

From:

<https://edu.iit.uni-miskolc.hu/> - Institute of Information Science - University of Miskolc

Permanent link:

[https://edu.iit.uni-miskolc.hu/tanszek:oktatas:web\\_technologia\\_alapjai:js](https://edu.iit.uni-miskolc.hu/tanszek:oktatas:web_technologia_alapjai:js)

Last update: **2026/03/09 08:53**

