ECMAScript versions (ES5, ES6/ES2015, ES2016+)

🧠 What is ECMAScript?

ECMAScript (ES) is the **standard specification** that defines the core features of the **JavaScript** language. It is maintained by **TC39** (Technical Committee 39) under **ECMA International**.

> JavaScript is an implementation of the ECMAScript standard.

📅 ECMAScript Version History and Features

🔷 ES5 (ECMAScript 5) – Released in 2009

The first major standard widely supported by all modern browsers.

✅ Key Features:

- `strict mode`: Enforces stricter parsing and error handling.

```
"use strict";
```

- Array methods: `forEach()`, `map()`, `filter()`, `reduce()`, `some()`, `every()`
- `Object.keys()`, `Object.create()`
- `JSON.parse()` and `JSON.stringify()`
- `Array.isArray()`
- Property descriptors with `Object.defineProperty()`
- Better error handling

⚙️ Example:

```
var arr = [1, 2, 3];
var doubled = arr.map(function(num) {
  return num * 2;
});
```

🔷 ES6 / ES2015 – Released in 2015

A **huge upgrade** and turning point for JavaScript. Introduced many new features.

✅ Key Features:

- `let` **and** `const`: Block-scoped variable declarations
- **Arrow functions**:

  ```
  const add = (a, b) ⇒ a + b;
  ```

- **Classes**:

  ```
  class Person {
    constructor(name) {
      this.name = name;
    }
  }
  ```

- **Template literals**:

  ```
  const greeting = `Hello, ${name}`;
  ```

- **Default parameters**
- **Destructuring assignment**

  ```
  const [a, b] = [1, 2];
  ```

- **Rest and Spread operators**

- **Promises**
- **Modules**:

```
import { add } from './math.js';
```

- **for...of loop**
- **Symbol type**
- **Map and Set collections**

🔷 ES2016 (ES7) – Released in 2016

A smaller update focused on a couple of features.

✅ Key Features:

- `Array.prototype.includes()`:

```
[1, 2, 3].includes(2); // true
```

- Exponentiation operator `*`:

```
2 ** 3 === 8;
```

🔷 ES2017 (ES8) – Released in 2017

✅ Key Features:

- `async/await` for asynchronous programming

```
async function fetchData() {
  const res = await fetch(url);
  return await res.json();
}
```

- `Object.values()`, `Object.entries()`

- `String.prototype.padStart()` and `padEnd()`
- `Trailing commas` in function parameters
- Shared memory and Atomics

🔷 ES2018 (ES9) – Released in 2018

✅ Key Features:

- Rest/Spread properties in objects
- `Promise.prototype.finally()`
- Asynchronous iteration: `for await...of`
- RegExp improvements (named groups, lookbehind, etc.)

🔷 ES2019 (ES10) – Released in 2019

✅ Key Features:

- `Array.prototype.flat()` and `flatMap()`
- `Object.fromEntries()`
- `String.prototype.trimStart()` and `trimEnd()`
- Optional `catch` binding:

```
try {
  // ...
} catch {
  // no error variable needed
}
```

🔷 ES2020 (ES11) – Released in 2020

✅ Key Features:

- **Nullish Coalescing Operator** `??`

```
let name = userInput ?? "Default";
```

- **Optional Chaining** `?.`

```
let city = user?.address?.city;
```

- `Promise.allSettled()`
- `globalThis` – universal global object
- `BigInt` type for large integers

🔷 ES2021 (ES12) – Released in 2021

✅ Key Features:

- Logical assignment operators: `&&=`, `||=`, `??=`
- `String.prototype.replaceAll()`
- WeakRefs and FinalizationRegistry
- Numeric separators for readability: `1_000_000`

🔷 ES2022 (ES13) – Released in 2022

✅ Key Features:

- Class fields and static initialization blocks
- Top-level `await` in ES modules
- `.at()` method for arrays

```
[1, 2, 3].at(-1); // 3
```

🔷 ES2023 (ES14) – Released in 2023

✅ Key Features:

- `Array.prototype.findLast()` and `findLastIndex()`
- Symbol disposal and resource management
- Import attributes for JSON modules

🧩 Summary Table

Version	Year	Major Features
ES5	2009	Strict mode, array methods
ES6/ES2015	2015	Classes, `let/const`, modules, arrow functions, promises
ES2016	2016	`includes()`, `**`
ES2017	2017	`async/await`, `Object.values()`, padding
ES2018	2018	Rest/spread, async iterators, `finally()`
ES2019	2019	`flat()`, `fromEntries()`, trim methods
ES2020	2020	`??`, `?.`, `BigInt`, `Promise.allSettled()`
ES2021	2021	Logical assignment, `replaceAll()`, WeakRefs
ES2022	2022	Top-level `await`, class fields, `.at()`
ES2023	2023	`findLast()`, import attributes, Symbol.dispose

📘 Conclusion

- ECMAScript evolves **annually** to improve JavaScript's performance, readability, and developer experience.
- Modern JavaScript projects mainly use **ES6+** features with **transpilers like Babel** and bundlers like Webpack or Vite to ensure backward compatibility.