JavaScript reference

The JavaScript reference serves as a repository of facts about the JavaScript language. The entire language is described here in detail. As you write JavaScript code, you'll refer to these pages often (thus the title "JavaScript reference").

The JavaScript language is intended to be used within some larger environment, be it a browser, server-side scripts, or similar. For the most part, this reference attempts to be environment-agnostic and does not target a web browser environment.

If you are new to JavaScript, start with the guide. Once you have a firm grasp of the fundamentals, you can use the reference to get more details on individual objects and language constructs.

Built-ins

JavaScript standard built-in objects, along with their methods and properties.

Value properties

- `globalThis`
- `Infinity`
- `NaN`
- `undefined`

Function properties

- `eval()`
- `isFinite()`
- `isNaN()`
- `parseFloat()`
- `parseInt()`
- `decodeURI()`
- `decodeURIComponent()`
- `encodeURI()`
- `encodeURIComponent()`
- `escape()` 🗑
- `unescape()` 🗑

Fundamental objects

- `Object`
- `Function`

- `Boolean`
- `Symbol`

Error objects

- `Error`
- `AggregateError`
- `EvalError`
- `RangeError`
- `ReferenceError`
- `SuppressedError`
- `SyntaxError`
- `TypeError`
- `URIError`
- `InternalError` ⚠

Numbers and dates

- `Number`
- `BigInt`
- `Math`
- `Date`
- `Temporal`

Text processing

- `String`
- `RegExp`

Indexed collections

- `Array`
- `Int8Array`
- `Uint8Array`
- `Uint8ClampedArray`
- `Int16Array`
- `Uint16Array`
- `Int32Array`
- `Uint32Array`
- `BigInt64Array`
- `BigUint64Array`
- `Float16Array`
- `Float32Array`
- `Float64Array`

Keyed collections

- `Map`

- `Set`
- `WeakMap`
- `WeakSet`

Structured data

- `ArrayBuffer`
- `SharedArrayBuffer`
- `DataView`
- `Atomics`
- `JSON`

Managing memory

- `WeakRef`
- `FinalizationRegistry`

Control abstraction objects

- `Iterator`
- `AsyncIterator`
- `Promise`
- `GeneratorFunction`
- `AsyncGeneratorFunction`
- `Generator`
- `AsyncGenerator`
- `AsyncFunction`
- `DisposableStack`
- `AsyncDisposableStack`

Reflection

- `Reflect`
- `Proxy`

Internationalization

- `Intl`
- `Intl.Collator`
- `Intl.DateTimeFormat`
- `Intl.DisplayNames`
- `Intl.DurationFormat`
- `Intl.ListFormat`
- `Intl.Locale`
- `Intl.NumberFormat`
- `Intl.PluralRules`
- `Intl.RelativeTimeFormat`
- `Intl.Segmenter`

Statements

JavaScript statements and declarations

Control flow

- `return`
- `break`
- `continue`
- `throw`
- `if...else`
- `switch`
- `try...catch`

Declaring variables

- `var`
- `let`
- `const`
- `using`
- `await using`

Functions and classes

- `function`
- `function*`
- `async function`
- `async function*`
- `class`

Iterations

- `do...while`
- `for`
- `for...in`
- `for...of`
- `for await...of`
- `while`

Others

- Empty
- Block
- Expression statement
- `debugger`
- `export`
- `import`

- label
- `with` 🗑

Expressions and operators

JavaScript expressions and operators.

Primary expressions

- `this`
- Literals
- `[]`
- `{}`
- `function`
- `class`
- `function*`
- `async function`
- `async function*`
- `/ab+c/i`
- `` `string` ``
- `( )`

Left-hand-side expressions

- Property accessors
- `?.`
- `new`
- `new.target`
- `import.meta`
- `super`
- `import()`

Increment and decrement

- `A++`
- `A--`
- `++A`
- `--A`

Unary operators

- `delete`
- `void`
- `typeof`
- `+`
- `-`
- `~`

- `!`
- `await`

Arithmetic operators

- `**`
- `*`
- `/`
- `%`
- `+` (Plus)
- `-`

Relational operators

- `<` (Less than)
- `>` (Greater than)
- `<=`
- `>=`
- `instanceof`
- `in`

Equality operators

- `==`
- `!=`
- `===`
- `!==`

Bitwise shift operators

- `<<`
- `>>`
- `>>>`

Binary bitwise operators

- `&`
- `|`
- `^`

Binary logical operators

- `&&`
- `||`
- `??`

Conditional (ternary) operator

- `(condition ? ifTrue : ifFalse)`

Assignment operators

- `=`
- `*=`
- `/=`
- `%=`
- `+=`
- `-=`
- `<<=`
- `>>=`
- `>>>=`
- `&=`
- `^=`
- `|=`
- `**=`
- `&&=`
- `||=`
- `??=`
- `[a, b] = arr`, `{ a, b } = obj`

Yield operators

- `yield`
- `yield*`

Spread syntax

- `...obj`

Comma operator

- `,`

Functions

JavaScript functions.

- Arrow Functions
- Default parameters
- Rest parameters
- `arguments`
- Method definitions
- getter
- setter

Classes

JavaScript classes.

- `constructor`
- `extends`
- Private elements
- Public class fields
- `static`
- Static initialization blocks

Regular expressions

JavaScript regular expressions.

- Backreference: `\1`, `\2`
- Capturing group: `(...)`
- Character class: `[...]`, `[^...]`
- Character class escape: `\d`, `\D`, `\w`, `\W`, `\s`, `\S`
- Character escape: `\n`, `\u{...}`
- Disjunction: `|`
- Input boundary assertion: `^`, `$`
- Literal character: `a`, `b`
- Lookahead assertion: `(?=...)`, `(?!...)`
- Lookbehind assertion: `(?<=...)`, `(?<!...)`
- Modifier: `(?ims-ims:...)`
- Named backreference: `\k<name>`
- Named capturing group: `(?<name>...)`
- Non-capturing group: `(?:...)`
- Quantifier: `*`, `+`, `?`, `{n}`, `{n,}`, `{n,m}`
- Unicode character class escape: `\p{...}`, `\P{...}`
- Wildcard: `.`
- Word boundary assertion: `\b`, `\B`

Additional reference pages

- JavaScript technologies overview
- Execution model
- Lexical grammar
- Data types and data structures
- Iteration protocols
- Trailing commas
- Errors
- Strict mode

- Deprecated features