<https://262.ecma-international.org/13.0/>

ECMA-262, 13th edition, June 2022  
ECMAScript® 2022 Language Specification



**About this Specification(规范)**

The document at <https://tc39.es/ecma262/> is the most accurate and up-to-date ECMAScript specification. It contains the content of the most recent yearly snapshot plus any [finished](https://github.com/tc39/proposals/blob/HEAD/finished-proposals.md)

[proposals](https://github.com/tc39/proposals/blob/HEAD/finished-proposals.md) (建议书；提案)(those that have reached Stage 4 in the [proposal process](https://tc39.es/process-document/) and thus are implemented in several implementations and will be in the next practical revision) since that snapshot was taken.

Since publication of the first edition in 1997, ECMAScript has grown to be one of the world's most widely used general-purpose programming languages. It is best known as the language embedded in web browsers but has also been widely adopted for server and embedded applications(嵌入式应用).

1 Scope

This Standard defines the ECMAScript 2022 general-purpose programming language.

2 Conformance 规范

A conforming implementation of ECMAScript must provide and support all the types, values, objects, properties, functions, and program syntax and semantics described in this specification.

4 Overview

This section contains a non-normative overview of the ECMAScript language.

ECMAScript is an object-oriented programming language for performing computations(计算指令) and manipulating(操作) computational(计算的) objects within a [host environment](https://262.ecma-international.org/13.0/" \l "host-environment). ECMAScript as defined here is not intended to be computationally self-sufficient(自给自足); indeed, there are no provisions(供应补给品) in this specification for input of external data or output of computed results. Instead, it is expected that the computational environment of an ECMAScript program will provide not only the objects and other facilities described in this specification but also certain environment-specific objects, whose description and behaviour are beyond the scope of this specification except to indicate that they may provide certain properties that can be accessed and certain functions that can be called from an ECMAScript program.

# 18 ECMAScript Standard Built-in Objects

There are certain built-in objects available **whenever**(每当) an ECMAScript [Script](https://262.ecma-international.org/13.0/" \l "prod-Script) or [Module](https://262.ecma-international.org/13.0/" \l "prod-Module) begins execution. One, the [global object](https://262.ecma-international.org/13.0/" \l "sec-global-object), is part of the global environment of the executing program. Others are accessible as initial properties of the [global object](https://262.ecma-international.org/13.0/" \l "sec-global-object) or indirectly as properties of accessible built-in objects.

Unless specified otherwise, a built-in object that is callable as a function is a built-in [function object](https://262.ecma-international.org/13.0/" \l "function-object) with the characteristics described in [10.3](https://262.ecma-international.org/13.0/" \l "sec-built-in-function-objects). Unless specified otherwise, the [[Extensible]] internal slot of a built-in object initially has the value true. Every built-in [function object](https://262.ecma-international.org/13.0/" \l "function-object) has a [[Realm]] **internal(内在的)** slot whose value is the [Realm Record](https://262.ecma-international.org/13.0/" \l "realm-record) of the [realm](https://262.ecma-international.org/13.0/" \l "realm) for which the object was initially created.

Many built-in objects are functions: they can be invoked with arguments. Some of them furthermore are [constructors](https://262.ecma-international.org/13.0/" \l "constructor): they are functions intended for use with the **new** operator. For each built-in function, this specification describes the arguments required by that function and the properties of that [function object](https://262.ecma-international.org/13.0/" \l "function-object). For each built-in [constructor](https://262.ecma-international.org/13.0/" \l "constructor), this specification furthermore describes properties of the prototype object of that [constructor](https://262.ecma-international.org/13.0/" \l "constructor) and properties of specific object instances returned by a **new** expression that invokes that [constructor](https://262.ecma-international.org/13.0/" \l "constructor).

Unless otherwise specified in the description of a particular function, if a built-in function or [constructor](https://262.ecma-international.org/13.0/" \l "constructor) is given fewer arguments than the function is specified to require, the function or [constructor](https://262.ecma-international.org/13.0/" \l "constructor) shall behave exactly as if it had been given sufficient additional arguments, each such argument being the undefined value. Such missing arguments are considered to be “not present” and may be identified in that manner by specification algorithms. In the description of a particular function, the terms “this value” and “NewTarget” have the meanings given in [10.3](https://262.ecma-international.org/13.0/" \l "sec-built-in-function-objects).

Unless otherwise specified in the description of a particular function, if a built-in function or [constructor](https://262.ecma-international.org/13.0/" \l "constructor) described is given more arguments than the function is specified to allow, the extra arguments are evaluated by the call and then ignored by the function. However, an implementation may define implementation specific behaviour relating to such arguments as long as the behaviour is not the throwing of a TypeError exception that is predicated simply on the presence of an extra argument.

NOTE 1

Implementations that add additional capabilities to the set of built-in functions are encouraged to do so by adding new functions rather than adding new parameters to existing functions.

Unless otherwise specified every built-in function and every built-in [constructor](https://262.ecma-international.org/13.0/" \l "constructor) has the [Function prototype object](https://262.ecma-international.org/13.0/" \l "sec-properties-of-the-function-prototype-object), which is the initial value of the expression **Function.prototype** ([20.2.3](https://262.ecma-international.org/13.0/" \l "sec-properties-of-the-function-prototype-object)), as the value of its [[Prototype]] internal slot.

Unless otherwise specified every built-in prototype object has the [Object prototype object](https://262.ecma-international.org/13.0/" \l "sec-properties-of-the-object-prototype-object), which is the initial value of the expression **Object.prototype** ([20.1.3](https://262.ecma-international.org/13.0/" \l "sec-properties-of-the-object-prototype-object)), as the value of its [[Prototype]] internal slot, except the [Object prototype object](https://262.ecma-international.org/13.0/" \l "sec-properties-of-the-object-prototype-object) itself.

Built-in [function objects](https://262.ecma-international.org/13.0/" \l "function-object) that are not identified as [constructors](https://262.ecma-international.org/13.0/" \l "constructor) do not implement the [[Construct]] internal method unless otherwise specified in the description of a particular function.

Each built-in function defined in this specification is created by calling the [CreateBuiltinFunction](https://262.ecma-international.org/13.0/" \l "sec-createbuiltinfunction) abstract operation ([10.3.3](https://262.ecma-international.org/13.0/" \l "sec-createbuiltinfunction)). The values of the length and name parameters are the initial values of the "length" and "name" properties as discussed below. The values of the prefix parameter are similarly discussed below.

Every built-in [function object](https://262.ecma-international.org/13.0/" \l "function-object), including [constructors](https://262.ecma-international.org/13.0/" \l "constructor), has a "length" property whose value is a non-negative [integral Number](https://262.ecma-international.org/13.0/" \l "integral-number). Unless otherwise specified, this value is equal to the number of required parameters shown in the subclause heading for the function description. Optional parameters and rest parameters are not included in the parameter count.

NOTE 2

For example, the [function object](https://262.ecma-international.org/13.0/" \l "function-object) that is the initial value of the "map" property of the [Array prototype object](https://262.ecma-international.org/13.0/" \l "sec-properties-of-the-array-prototype-object) is described under the subclause heading «Array.prototype.map (callbackFn [ , thisArg])» which shows the two named arguments callbackFn and thisArg, the latter being optional; therefore the value of the "length" property of that [function object](https://262.ecma-international.org/13.0/" \l "function-object) is 1.

Unless otherwise specified, the "length" property of a built-in [function object](https://262.ecma-international.org/13.0/" \l "function-object) has the attributes { [[Writable]]: false, [[Enumerable]]: false, [[Configurable]]: true }.

Every built-in [function object](https://262.ecma-international.org/13.0/" \l "function-object), including [constructors](https://262.ecma-international.org/13.0/" \l "constructor), has a "name" property whose value is a String. Unless otherwise specified, this value is the name that is given to the function in this specification. Functions that are identified as anonymous functions use the empty String as the value of the "name" property. For functions that are specified as properties of objects, the name value is the [property name](https://262.ecma-international.org/13.0/" \l "property-name) string used to access the function. Functions that are specified as get or set accessor functions of built-in properties have "get" or "set" (respectively) passed to the prefix parameter when calling [CreateBuiltinFunction](https://262.ecma-international.org/13.0/" \l "sec-createbuiltinfunction).

The value of the "name" property is explicitly specified for each built-in functions whose [property key](https://262.ecma-international.org/13.0/" \l "sec-object-type) is a Symbol value. If such an explicitly specified value starts with the prefix "get " or "set " and the function for which it is specified is a get or set accessor function of a built-in property, the value without the prefix is passed to the name parameter, and the value "get" or "set" (respectively) is passed to the prefix parameter when calling [CreateBuiltinFunction](https://262.ecma-international.org/13.0/" \l "sec-createbuiltinfunction).

Unless otherwise specified, the "name" property of a built-in [function object](https://262.ecma-international.org/13.0/" \l "function-object) has the attributes { [[Writable]]: false, [[Enumerable]]: false, [[Configurable]]: true }.

Every other [data property](https://262.ecma-international.org/13.0/" \l "sec-object-type) described in clauses [19](https://262.ecma-international.org/13.0/" \l "sec-global-object) through [28](https://262.ecma-international.org/13.0/" \l "sec-reflection) and in Annex [B.2](https://262.ecma-international.org/13.0/" \l "sec-additional-built-in-properties) has the attributes { [[Writable]]: true, [[Enumerable]]: false, [[Configurable]]: true } unless otherwise specified.

Every [accessor property](https://262.ecma-international.org/13.0/" \l "sec-object-type) described in clauses [19](https://262.ecma-international.org/13.0/" \l "sec-global-object) through [28](https://262.ecma-international.org/13.0/" \l "sec-reflection) and in Annex [B.2](https://262.ecma-international.org/13.0/" \l "sec-additional-built-in-properties) has the attributes { [[Enumerable]]: false, [[Configurable]]: true } unless otherwise specified. If only a get accessor function is described, the set accessor function is the default value, undefined. If only a set accessor is described the get accessor is the default value, undefined.