Error Reference

Error object, and has a **name**, **message**, and **stack** property.

1. **Error.name**

The name data property of Error.prototype is shared by all Error instances. It represents the name for the type of error. For Error.prototype.name, the initial value is "Error". Subclasses like TypeError and SyntaxError provide their own name properties.

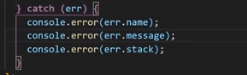
1. **Error: message**

The message data property of an Error instance is a human-readable description of the error.

1. **Error.stack**

this.stack = <call stack>; // non-standard, but most environments support it

the stack property have both name and message properties.



Error objects are thrown when runtime errors occur. The Error object can also be used as a base object for user-defined exceptions. See below for standard built-in error types.

**Description**

Runtime errors result in new Error objects being created and thrown.

Error is a serializable object, so it can be cloned with structuredClone() or copied between Workers using postMessage().

Error types

Besides the generic Error constructor, there are other core error constructors in JavaScript. For client-side exceptions

List of errors

1. [Error: Permission denied to access property "x"](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Property_access_denied)
2. [InternalError: too much recursion](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Too_much_recursion)
3. [RangeError: BigInt division by zero](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/BigInt_division_by_zero)
4. [RangeError: BigInt negative exponent](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/BigInt_negative_exponent)
5. [RangeError: argument is not a valid code point](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Not_a_valid_code_point)
6. [RangeError: invalid array length](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Invalid_array_length)
7. [RangeError: invalid date](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Invalid_date)
8. [RangeError: precision is out of range](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Precision_range)
9. [RangeError: radix must be an integer](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Bad_radix)
10. [RangeError: repeat count must be less than infinity](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Resulting_string_too_large)
11. [RangeError: repeat count must be non-negative](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Negative_repetition_count)
12. [RangeError: x can't be converted to BigInt because it isn't an integer](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Cant_be_converted_to_BigInt_because_it_isnt_an_integer)
13. [ReferenceError: "x" is not defined](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Not_defined)
14. [ReferenceError: assignment to undeclared variable "x"](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Undeclared_var)
15. [ReferenceError: can't access lexical declaration 'X' before initialization](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Cant_access_lexical_declaration_before_init)
16. [ReferenceError: deprecated caller or arguments usage](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Deprecated_caller_or_arguments_usage)
17. [ReferenceError: reference to undefined property "x"](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Undefined_prop)
18. [SyntaxError: "0"-prefixed octal literals and octal escape seq. are deprecated](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Deprecated_octal)
19. [SyntaxError: "use strict" not allowed in function with non-simple parameters](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Strict_non_simple_params)
20. [SyntaxError: "x" is a reserved identifier](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Reserved_identifier)
21. [SyntaxError: JSON.parse: bad parsing](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/JSON_bad_parse)
22. [SyntaxError: Unexpected '#' used outside of class body](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Hash_outside_class)
23. [SyntaxError: Unexpected token](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Unexpected_token)
24. [SyntaxError: Using //@ to indicate sourceURL pragmas is deprecated. Use //# instead](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Deprecated_source_map_pragma)
25. [SyntaxError: a declaration in the head of a for-of loop can't have an initializer](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Invalid_for-of_initializer)
26. [SyntaxError: applying the 'delete' operator to an unqualified name is deprecated](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Delete_in_strict_mode)
27. [SyntaxError: await is only valid in async functions, async generators and modules](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Bad_await)
28. [SyntaxError: cannot use `??` unparenthesized within `||` and `&&` expressions](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Cant_use_nullish_coalescing_unparenthesized)
29. [SyntaxError: continue must be inside loop](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Bad_continue)
30. [SyntaxError: for-in loop head declarations may not have initializers](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Invalid_for-in_initializer)
31. [SyntaxError: function statement requires a name](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Unnamed_function_statement)
32. [SyntaxError: identifier starts immediately after numeric literal](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Identifier_after_number)
33. [SyntaxError: illegal character](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Illegal_character)
34. [SyntaxError: invalid BigInt syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Invalid_BigInt_syntax)
35. [SyntaxError: invalid assignment left-hand side](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Invalid_assignment_left-hand_side)
36. [SyntaxError: invalid regular expression flag "x"](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Bad_regexp_flag)
37. [SyntaxError: label not found](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Label_not_found)
38. [SyntaxError: missing ) after argument list](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Missing_parenthesis_after_argument_list)
39. [SyntaxError: missing ) after condition](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Missing_parenthesis_after_condition)
40. [SyntaxError: missing : after property id](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Missing_colon_after_property_id)
41. [SyntaxError: missing ; before statement](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Missing_semicolon_before_statement)
42. [SyntaxError: missing = in const declaration](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Missing_initializer_in_const)
43. [SyntaxError: missing ] after element list](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Missing_bracket_after_list)
44. [SyntaxError: missing formal parameter](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Missing_formal_parameter)
45. [SyntaxError: missing name after . operator](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Missing_name_after_dot_operator)
46. [SyntaxError: missing variable name](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/No_variable_name)
47. [SyntaxError: missing } after function body](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Missing_curly_after_function_body)
48. [SyntaxError: missing } after property list](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Missing_curly_after_property_list)
49. [SyntaxError: redeclaration of formal parameter "x"](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Redeclared_parameter)
50. [SyntaxError: return not in function](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Bad_return)
51. [SyntaxError: test for equality (==) mistyped as assignment (=)?](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Equal_as_assign)
52. [SyntaxError: unlabeled break must be inside loop or switch](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Bad_break)
53. [SyntaxError: unparenthesized unary expression can't appear on the left-hand side of '\*\*'](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Unparenthesized_unary_expr_lhs_exponentiation)
54. [SyntaxError: unterminated string literal](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Unterminated_string_literal)
55. [TypeError: "x" has no properties](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/No_properties)
56. [TypeError: "x" is (not) "y"](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Unexpected_type)
57. [TypeError: "x" is not a constructor](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Not_a_constructor)
58. [TypeError: "x" is not a function](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Not_a_function)
59. [TypeError: "x" is not a non-null object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/No_non-null_object)
60. [TypeError: "x" is read-only](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Read-only)
61. [TypeError: 'x' is not iterable](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/is_not_iterable)
62. [TypeError: More arguments needed](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/More_arguments_needed)
63. [TypeError: Reduce of empty array with no initial value](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Reduce_of_empty_array_with_no_initial_value)
64. [TypeError: X.prototype.y called on incompatible type](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Called_on_incompatible_type)
65. [TypeError: can't assign to property "x" on "y": not an object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Cant_assign_to_property)
66. [TypeError: can't convert BigInt to number](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Cant_convert_BigInt_to_number)
67. [TypeError: can't convert x to BigInt](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Cant_convert_x_to_BigInt)
68. [TypeError: can't define property "x": "obj" is not extensible](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Cant_define_property_object_not_extensible)
69. [TypeError: can't delete non-configurable array element](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Non_configurable_array_element)
70. [TypeError: can't redefine non-configurable property "x"](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Cant_redefine_property)
71. [TypeError: cannot use 'in' operator to search for 'x' in 'y'](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/in_operator_no_object)
72. [TypeError: cyclic object value](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Cyclic_object_value)
73. [TypeError: invalid 'instanceof' operand 'x'](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/invalid_right_hand_side_instanceof_operand)
74. [TypeError: invalid Array.prototype.sort argument](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Array_sort_argument)
75. [TypeError: invalid assignment to const "x"](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Invalid_const_assignment)
76. [TypeError: property "x" is non-configurable and can't be deleted](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Cant_delete)
77. [TypeError: setting getter-only property "x"](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Getter_only)
78. [URIError: malformed URI sequence](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Malformed_URI)
79. [Warning: -file- is being assigned a //# sourceMappingURL, but already has one](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Already_has_pragma)
80. [Warning: unreachable code after return statement](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Stmt_after_return)

**Error() constructor**

The Error() constructor creates Error objects.

Syntax

JS

new Error()

new Error(message)

new Error(message, options)

new Error(message, fileName)

new Error(message, fileName, lineNumber)

Error()

Error(message)

Error(message, options)

Error(message, fileName)

Error(message, fileName, lineNumber)

**Examples**

Function call or new construction

When Error is used like a function, that is without new, it will return an Error object. Therefore, a mere call to Error will produce the same output that constructing an Error object via the new keyword would.

<html>

<body>

<h2>JavaScript Error Handling</h2>

<p>How to use <b>catch</b> to display an error.</p>

<p id="demo"></p>

<script>

function error(message){

this.message = message;

this.name = "ReferenceError";

}

try {

//throw new error("Abbas"); // function call

// throw Error("error"); // call without new keyword

throw new Error("error2") // call with new keyword the result will same

altwert("Welcome guest!");

}

catch(err) {

alert(err.name);

alert(err.message);

}

</script>

</body>

</html>

JS

const x = Error("I was created using a function call!");

// above has the same functionality as following

const y = new Error('I was constructed via the "new" keyword!');

**Rethrowing** an error with a cause

It is sometimes useful to catch an error and re-throw it with a new message. In this case you should pass the original error into the constructor for the new Error, as shown.

JS

try {

frameworkThatCanThrow();

} catch (err) {

throw new Error("New error message", { cause: err });

}

**Important and common errors**

**1. RangeError**

This is thrown when a number is outside an allowable range of values.

For example,

const l = console.logconst arr = [90,88]  
arr.length=90\*\*99

We have an array, arr with two elements. Next, we try to grow the array to contain 90\*\*99 == 2.9512665430652753e+193 elements.

This number is way past the size arrays can be grown to. Running it will throw a RangeError:

$ node errors  
errors.js:4  
arr.length=90\*\*99  
 ^RangeError: Invalid array length

because the number we want to increase the arr array to is out of the range specified by JS.

**2. ReferenceError**

This error is thrown when a reference made to a variable/item is broken. That is the variable/item doesn’t exist.

For example,

const l=console.logconst cat = "cat"  
cat  
dog

We have a variable cat initialized to “cat”. Next, we referred to the cat variable and dog variable. cat variable exists but dog variable doesn’t.

cat will return “cat”, while dog will throw a reference error because the name dog can’t be found on the environment record.

$ node errors  
errors.js:3  
dog  
^ReferenceError: dog is not defined

Whenever we create or define a variable, the variable name will be written to an environment record. This environment record is like key-value storage,

+-------------+  
| Key | Value |  
---------------  
| cat | "cat" |  
+-------------+

that stores variables defined in our program, whenever we reference a variable. The environment record is searched with the name of the variable as key when it is found on the record the value is extracted and returned. calling a function that hasn’t been defined.

Now, when we create or define a variable without assignment. The variable is written to the environment record, with the key as the variable name but the value will hold undefined.

var catenv record  
+-----------------+  
| Key | Value |  
-------------------  
| cat | undefined |  
+-----------------+

When the variable is later assigned a value, the variable is searched in the env record and when found the initial undefined valued is overwritten with the assigned value.

var cat  
cat = "cat"env record  
+-------------+  
| Key | Value |  
---------------  
| cat | "cat" |  
+-------------+

So, a RefernceError is thrown by the JS engine when a variable name can’t be found in the env record.

+-------------+  
| Key | Value |  
---------------  
| cat | "cat" |  
+-------------+cat // "cat", yes, :) it's there  
dog // :( what's this? can't find it

Note: an undefined variable won’t throw ReferenceError because it exists in the env record just that its value hasn’t been set.

**3. SyntaxError**

This is the most common error we encounter. This error occurs when we type code that the JS engine can understand.

This error is caught by the JS engine during parsing. There are different stages in the JS engine our code is put through before we see those results on the terminal.

* tokenization
* parsing
* interpreting

tokenization breaks the source of the code into individual units. At this stage, numbers, keywords, literals, operators are sorted out and individually marked.

Next, the token stream generated will be passed to the parsing stage, which is handled by a parser. This is where an AST is generated from the token stream. AST is an abstract representation of the structure of our code.

During these two stages, tokenization and parsing, if the syntax/source of our codes doesn’t conform to the syntax rules of JS makes the stages fail and throw SyntaxError. For example,

const l = console.loglet cat h = "cat"

What is with the lone “h”? The “h” being there breaks the code.

$ node errors  
errors.js:3  
let cat h = "cat"  
 ^SyntaxError: Unexpected identifier

See, Node.js points out the problem. It says that the “h” was unexpected. IT being there breaks the declaration of the cat variable.

So we can say syntax error occurs during parsing/compile time.

**4. TypeError**

TypeError is used to indicate an unsuccessful operation when none of the other NativeError objects are an appropriate indication of the failure cause.

TypeError occurs when an operation is performed on a wrong data type. Maybe a boolean is expected but a sting is found.

For example,

if we try to convert a number to uppercase like this:

const num = 123  
num.toUpperCase()

This will throw a TypeError

$ node errors  
errors.js:4  
num.toUpperCase()  
 ^TypeError: num.toUpperCase is not a function

because the toUpperCase function expects a string data type. The toUpperCase function is intentionally generic; it does not require that its this value be a String object. Therefore, it can be transferred to other kinds of objects for use as a method.

Only strings are converted to uppercase or lowercase if we call the toUpperCase function on Objects, Boolean, Symbol, null, undefined data types we will get the TypeError because it is the wrong data type it operates.

**5. URIError**

This indicates that one of the global URI handling functions was used in a way that is incompatible with its definition.

URI (Uniform Resource Indicator) in JS has the functions: decodeURI, decodeURIComponent, etc.

If we call any of them with the wrong parameter we will get a URIError

decodeURI("%")  
^URIError: URI malformed

decodeURI, gets the unencoded version of a URI. “%” is not the right URI, so a URIError was thrown.

URIError is thrown when there’s a problem encoding or decoding the URI.

**6. EvalError**

This is used to identify errors when using the global eval() function.

According to EcmaSpec 2018 edition:

This exception is not currently used within this specification. This object remains for compatibility with previous editions of this specification.

**7. InternalError**

This error occurs internally in the JS engine, especially when it has too much data to handle and the stack grows way over its critical limit.

This occurs when the JS engine is overwhelmed by too many recursions, too many switch cases, etc

switch(num) {  
 case 1:  
 ...  
 break  
 case 2:  
 ...  
 break  
 case 3:  
 ...  
 break  
 case 4:  
 ...  
 break  
 case 5:  
 ...  
 break  
 case 6:  
 ...  
 break  
 case 7:  
 ...  
 break  
 ... up to 1000 cases  
 }

Too much recursion, a simple example is this:

function foo() {  
 foo()  
}  
foo()