**throw**

The **throw** statement throws a user-defined exception. Execution of the current function will stop (the statements after throw won't be executed), and control will be passed to the first [catch](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch) block in the call stack. If no catch block exists among caller functions, the program will terminate.

[**Try it**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/throw#try_it)

[**Syntax**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/throw#syntax)

throw expression

Copy to Clipboard

expression

The expression to throw.

[**Description**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/throw#description)

Use the throw statement to throw an exception. When you throw an exception, expression specifies the value of the exception. Each of the following throws an exception:

throw 'Error2'; // generates an exception with a string value

throw 42; // generates an exception with the value 42

throw true; // generates an exception with the value true

throw new Error('Required'); // generates an error object with the message of Required

Copy to Clipboard

Also note that the throw statement is affected by [automatic semicolon insertion (ASI)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar#automatic_semicolon_insertion) as no line terminator between the throw keyword and the expression is allowed.

[**Examples**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/throw#examples)

[**Throw an object**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/throw#throw_an_object)

You can specify an object when you throw an exception. You can then reference the object's properties in the catch block. The following example creates an object of type UserException and uses it in a throw statement.

function UserException(message) {

this.message = message;

this.name = 'UserException';

}

function getMonthName(mo) {

mo--; // Adjust month number for array index (1 = Jan, 12 = Dec)

const months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul',

'Aug', 'Sep', 'Oct', 'Nov', 'Dec'];

if (months[mo] !== undefined) {

return months[mo];

} else {

throw new UserException('InvalidMonthNo');

}

}

let monthName;

try {

// statements to try

const myMonth = 15; // 15 is out of bound to raise the exception

monthName = getMonthName(myMonth);

} catch (e) {

monthName = 'unknown';

console.error(e.message, e.name); // pass exception object to err handler

}

Copy to Clipboard

[**Another example of throwing an object**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/throw#another_example_of_throwing_an_object)

The following example tests an input string for a U.S. zip code. If the zip code uses an invalid format, the throw statement throws an exception by creating an object of type ZipCodeFormatException.

/\*

\* Creates a ZipCode object.

\*

\* Accepted formats for a zip code are:

\* 12345

\* 12345-6789

\* 123456789

\* 12345 6789

\*

\* If the argument passed to the ZipCode constructor does not

\* conform to one of these patterns, an exception is thrown.

\*/

class ZipCode {

static pattern = /[0-9]{5}([- ]?[0-9]{4})?/;

constructor(zip) {

zip = String(zip);

const match = zip.match(ZipCode.pattern);

if (!match) {

throw new ZipCodeFormatException(zip);

}

// zip code value will be the first match in the string

this.value = match[0];

}

valueOf() {

return this.value;

}

toString() {

return this.value;

}

}

class ZipCodeFormatException extends Error {

constructor(zip) {

super(`${zip} does not conform to the expected format for a zip code`);

}

}

/\*

\* This could be in a script that validates address data

\* for US addresses.

\*/

const ZIPCODE\_INVALID = -1;

const ZIPCODE\_UNKNOWN\_ERROR = -2;

function verifyZipCode(z) {

try {

z = new ZipCode(z);

} catch (e) {

const isInvalidCode = e instanceof ZipCodeFormatException;

return isInvalidCode ? ZIPCODE\_INVALID : ZIPCODE\_UNKNOWN\_ERROR;

}

return z;

}

a = verifyZipCode(95060); // returns 95060

b = verifyZipCode(9560); // returns -1

c = verifyZipCode('a'); // returns -1

d = verifyZipCode('95060'); // returns 95060

e = verifyZipCode('95060 1234'); // returns 95060 1234

Copy to Clipboard

[**Rethrow an exception**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/throw#rethrow_an_exception)

You can use throw to rethrow an exception after you catch it. The following example catches an exception with a numeric value and rethrows it if the value is over 50. The rethrown exception propagates up to the enclosing function or to the top level so that the user sees it.

try {

throw n; // throws an exception with a numeric value

} catch (e) {

if (e <= 50) {

// statements to handle exceptions 1-50

} else {

// cannot handle this exception, so rethrow

throw e;

}

}

Copy to Clipboard