# cssesc [Build status](https://travis-ci.org/mathiasbynens/cssesc) [Code coverage status](https://codecov.io/gh/mathiasbynens/cssesc)

A JavaScript library for escaping CSS strings and identifiers while generating the shortest possible ASCII-only output.

This is a JavaScript library for [escaping text for use in CSS strings or identifiers](https://mathiasbynens.be/notes/css-escapes) while generating the shortest possible valid ASCII-only output. [Here’s an online demo.](https://mothereff.in/css-escapes)

[A polyfill for the CSSOM CSS.escape() method is available in a separate repository.](https://mths.be/cssescape) (In comparison, *cssesc* is much more powerful.)

Feel free to fork if you see possible improvements!

## Installation

Via [npm](https://www.npmjs.com/):

npm install cssesc

In a browser:

<script src="cssesc.js"></script>

In [Node.js](https://nodejs.org/):

const cssesc = require('cssesc');

In Ruby using [the ruby-cssesc wrapper gem](https://github.com/borodean/ruby-cssesc):

gem install ruby-cssesc

require 'ruby-cssesc'

CSSEsc.escape('I ♥ Ruby', is\_identifier: true)

In Sass using [sassy-escape](https://github.com/borodean/sassy-escape):

gem install sassy-escape

body {

content: escape('I ♥ Sass', $is-identifier: true);

}

## API

### cssesc(value, options)

This function takes a value and returns an escaped version of the value where any characters that are not printable ASCII symbols are escaped using the shortest possible (but valid) [escape sequences for use in CSS strings or identifiers](https://mathiasbynens.be/notes/css-escapes).

cssesc('Ich ♥ Bücher');

// → 'Ich \\2665 B\\FC cher'

cssesc('foo 𝌆 bar');

// → 'foo \\1D306 bar'

By default, cssesc returns a string that can be used as part of a CSS string. If the target is a CSS identifier rather than a CSS string, use the isIdentifier: true setting (see below).

The optional options argument accepts an object with the following options:

#### isIdentifier

The default value for the isIdentifier option is false. This means that the input text will be escaped for use in a CSS string literal. If you want to use the result as a CSS identifier instead (in a selector, for example), set this option to true.

cssesc('123a2b');

// → '123a2b'

cssesc('123a2b', {

'isIdentifier': true

});

// → '\\31 23a2b'

#### quotes

The default value for the quotes option is 'single'. This means that any occurences of ' in the input text will be escaped as \', so that the output can be used in a CSS string literal wrapped in single quotes.

cssesc('Lorem ipsum "dolor" sit \'amet\' etc.');

// → 'Lorem ipsum "dolor" sit \\\'amet\\\' etc.'

// → "Lorem ipsum \"dolor\" sit \\'amet\\' etc."

cssesc('Lorem ipsum "dolor" sit \'amet\' etc.', {

'quotes': 'single'

});

// → 'Lorem ipsum "dolor" sit \\\'amet\\\' etc.'

// → "Lorem ipsum \"dolor\" sit \\'amet\\' etc."

If you want to use the output as part of a CSS string literal wrapped in double quotes, set the quotes option to 'double'.

cssesc('Lorem ipsum "dolor" sit \'amet\' etc.', {

'quotes': 'double'

});

// → 'Lorem ipsum \\"dolor\\" sit \'amet\' etc.'

// → "Lorem ipsum \\\"dolor\\\" sit 'amet' etc."

#### wrap

The wrap option takes a boolean value (true or false), and defaults to false (disabled). When enabled, the output will be a valid CSS string literal wrapped in quotes. The type of quotes can be specified through the quotes setting.

cssesc('Lorem ipsum "dolor" sit \'amet\' etc.', {

'quotes': 'single',

'wrap': true

});

// → '\'Lorem ipsum "dolor" sit \\\'amet\\\' etc.\''

// → "\'Lorem ipsum \"dolor\" sit \\\'amet\\\' etc.\'"

cssesc('Lorem ipsum "dolor" sit \'amet\' etc.', {

'quotes': 'double',

'wrap': true

});

// → '"Lorem ipsum \\"dolor\\" sit \'amet\' etc."'

// → "\"Lorem ipsum \\\"dolor\\\" sit \'amet\' etc.\""

#### escapeEverything

The escapeEverything option takes a boolean value (true or false), and defaults to false (disabled). When enabled, all the symbols in the output will be escaped, even printable ASCII symbols.

cssesc('lolwat"foo\'bar', {

'escapeEverything': true

});

// → '\\6C\\6F\\6C\\77\\61\\74\\"\\66\\6F\\6F\\\'\\62\\61\\72'

// → "\\6C\\6F\\6C\\77\\61\\74\\\"\\66\\6F\\6F\\'\\62\\61\\72"

#### Overriding the default options globally

The global default settings can be overridden by modifying the css.options object. This saves you from passing in an options object for every call to encode if you want to use the non-default setting.

// Read the global default setting for `escapeEverything`:

cssesc.options.escapeEverything;

// → `false` by default

// Override the global default setting for `escapeEverything`:

cssesc.options.escapeEverything = true;

// Using the global default setting for `escapeEverything`, which is now `true`:

cssesc('foo © bar ≠ baz 𝌆 qux');

// → '\\66\\6F\\6F\\ \\A9\\ \\62\\61\\72\\ \\2260\\ \\62\\61\\7A\\ \\1D306\\ \\71\\75\\78'

### cssesc.version

A string representing the semantic version number.

### Using the cssesc binary

To use the cssesc binary in your shell, simply install cssesc globally using npm:

npm install -g cssesc

After that you will be able to escape text for use in CSS strings or identifiers from the command line:

$ cssesc 'föo ♥ bår 𝌆 baz'

f\F6o \2665 b\E5r \1D306 baz

If the output needs to be a CSS identifier rather than part of a string literal, use the -i/--identifier option:

$ cssesc --identifier 'föo ♥ bår 𝌆 baz'

f\F6o\ \2665\ b\E5r\ \1D306\ baz

See cssesc --help for the full list of options.

## Support

This library supports the Node.js and browser versions mentioned in [.babelrc](https://github.com/mathiasbynens/cssesc/blob/master/.babelrc). For a version that supports a wider variety of legacy browsers and environments out-of-the-box, [see v0.1.0](https://github.com/mathiasbynens/cssesc/releases/tag/v0.1.0).

## Author

| [**twitter/mathias**](https://twitter.com/mathias) |
| --- |
| [Mathias Bynens](https://mathiasbynens.be/) |

## License

This library is available under the [MIT](https://mths.be/mit) license.