kleur

npm v4.1.5 CI no status downloads 197.5M/month

install size

The fastest Node.js library for formatting terminal text with ANSI colors~!

Features

- No dependencies
- Super lightweight & performant
- Supports nested & chained colors
- No `String.prototype` modifications
- Conditional color support
- Fully treeshakable
- Familiar API

As of `v3.0` the Chalk-style syntax (magical getter) is no longer used.
Please visit History for migration paths supporting that syntax.

Install

```
$ npm install --save kleur
```

Usage

```
import kleur from 'kleur';

// basic usage
kleur.red('red text');

// chained methods
kleur.blue().bold().underline('howdy
        partner');

// nested methods
kleur.bold(`${
        white().bgRed('[ERROR]') } ${
        kleur.red().italic('Something
        happened')}`);
```

Chained Methods

```
const { bold, green } =
        require('kleur');

console.log(bold().red('this is a bold
        red message'));
console.log(bold().italic('this is a
        bold italicized message'));
console.log(bold().yellow().bgRed().italic('this
        is a bold yellow italicized message'));
console.log(green().bold().underline('this
        is a bold green underlined
        message'));
```

Nested Methods

```
const { yellow, red, cyan } =
        require('kleur');

console.log(yellow(`foo $
        {red().bold('red')} bar $
        {cyan('cyan')} baz`));
console.log(yellow('foo ' +
        red().bold('red') + ' bar ' +
        cyan('cyan') + ' baz'));
```

Conditional Support

Toggle color support as needed; `kleur` includes simple auto-detection which may not cover all cases.

> ***Note:*** *Both* `kleur` *and* `kleur/colors` *share the same detection logic.*

```
import kleur from 'kleur';

// manually disable
kleur.enabled = false;

// or use another library to detect
        support
kleur.enabled = require('color-
        support').level > 0;
```

```
console.log(kleur.red('I will only be
        colored red if the terminal
        supports colors'));
```

Important:

Colors will be disabled automatically in non [TTY contexts](). For example, spawning another process or piping output into another process will disable colorization automatically. To force colors in your piped output, you may do so with the `FORCE_COLOR=1` *environment variable:*

```
$ node app.js #=> COLORS
$ node app.js > log.txt  #=> NO COLORS
$ FORCE_COLOR=1 node app.js > log.txt
        #=> COLORS
$ FORCE_COLOR=0 node app.js > log.txt
        #=> NO COLORS
```

API

Any `kleur` method returns a `String` when invoked with input; otherwise chaining is expected.

It's up to the developer to pass the output to destinations like `console.log`, `process.stdout.write`, *etc.*

As I work more with Rust, the newer syntax feels so much better & more natural!

If you prefer the old syntax, you may migrate to `ansi-colors` or newer `chalk` releases.
Versions below `kleur@3.0` have been officially deprecated.

License

MIT © Luke Edwards

```
±0.19% (98 runs sampled)

# Stacked colors
  ansi-colors      x  23,331 ops/sec
±1.81% (94 runs sampled)
  chalk            x 337,178 ops/sec
±0.20% (98 runs sampled)
  kleur            x  78,299 ops/sec
±1.01% (97 runs sampled)
  kleur/colors     x 104,431 ops/sec
±0.22% (97 runs sampled)

# Nested colors
  ansi-colors      x  67,181 ops/sec
±1.15% (92 runs sampled)
  chalk            x 116,361 ops/sec
±0.63% (94 runs sampled)
  kleur            x 139,514 ops/sec
±0.76% (95 runs sampled)
  kleur/colors     x 145,716 ops/sec
±0.97% (97 runs sampled)
```

History

This project originally forked `ansi-colors`.

Beginning with `kleur@3.0`, the Chalk-style syntax (magical getter) has been replaced with function calls per key:

```
// Old:
c.red.bold.underline('old');

// New:
c.red().bold().underline('new');
```

The methods below are grouped by type for legibility purposes only. They each can be chained or nested with one another.

Colors: > black — red — green — yellow — blue — magenta — cyan — white — gray — grey

Backgrounds: > bgBlack — bgRed — bgGreen — bgYellow — bgBlue — bgMagenta — bgCyan — bgWhite

Modifiers: > reset — bold — dim — italic* — underline — inverse — hidden — strikethrough*

* *Not widely supported*

Individual Colors

When you only need a few colors, it doesn't make sense to import *all* of `kleur` because, as small as it is, `kleur` is not treeshakeable, and so most of its code will be doing nothing. In order to fix this, you can import from the `kleur/colors` submodule which *fully* supports tree-shaking.

The caveat with this approach is that color functions **are not** chainable~!
Each function receives and colorizes its input. You may combine colors, backgrounds, and modifiers by nesting function calls within other functions.

```
// or: import * as kleur from 'kleur/
        colors';
import { red, underline, bgWhite } from
        'kleur/colors';
```

```
red('red text');
//~> kleur.red('red text');

underline(red('red underlined text'));
//~> kleur.underline().red('red
        underlined text');

bgWhite(underline(red('red underlined
        text w/ white background')));
//~> kleur.bgWhite().underline().red('red
        underlined text w/ white
        background');
```

Note: *All the same colors, backgrounds, and modifiers are available.*

Conditional Support

The `kleur/colors` submodule also allows you to toggle color support, as needed.
It includes the same initial assumptions as `kleur`, in an attempt to have colors enabled by default.

Unlike `kleur`, this setting exists as `kleur.$.enabled` instead of `kleur.enabled`:

```
import * as kleur from 'kleur/colors';
// or: import { $, red } from 'kleur/
        colors';

// manually disabled
kleur.$.enabled = false;

// or use another library to detect
        support
```

```
kleur.$.enabled = require('color-
        support').level > 0;

console.log(red('I will only be colored
        red if the terminal supports
        colors'));
```

Benchmarks

Using Node v10.13.0

Load time

```
chalk        :: 5.303ms
kleur        :: 0.488ms
kleur/colors :: 0.369ms
ansi-colors  :: 1.504ms
```

Performance

```
# All Colors
  ansi-colors      x 177,625 ops/sec
±1.47% (92 runs sampled)
  chalk            x 611,907 ops/sec
±0.20% (92 runs sampled)
  kleur            x 742,509 ops/sec
±1.47% (93 runs sampled)
  kleur/colors     x 881,742 ops/sec
```