

> Terminal string styling done right [![Build Status](https://travis-ci.org/chalk/chalk.svg?branch=master)](https://travis-ci.org/chalk/chalk) [![Coverage Status](https://coveralls.io/repos/github/chalk/chalk/badge.svg?branch=master)](https://coveralls.io/github/chalk/chalk?branch=master) [![npm dependents](https://badgen.net/npm/dependents/chalk)](https://www.npmjs.com/package/chalk?activeTab=dependents) [![Downloads](https://badgen.net/npm/dt/chalk)](https://www.npmjs.com/package/chalk) [![](https://img.shields.io/badge/unicorn-approved-ff69b4.svg)](https://www.youtube.com/watch?v=9auOCbH5Ns4) [![XO code style](https://img.shields.io/badge/code\_style-XO-5ed9c7.svg)](https://github.com/xojs/xo) ![TypeScript-ready](https://img.shields.io/npm/types/chalk.svg)  ## Highlights - Expressive API - Highly performant - Ability to nest styles - [256/Truecolor color support](#256-and-truecolor-color-support) - Auto-detects color support - Doesn't extend `String.prototype` - Clean and focused - Actively maintained - [Used by ~46,000 packages](https://www.npmjs.com/browse/depended/chalk) as of October 1, 2019 ## Install ```console $ npm install chalk ``` ## Usage ```js const chalk = require('chalk'); console.log(chalk.blue('Hello world!')); ``` Chalk comes with an easy to use composable API where you just chain and nest the styles you want. ```js const chalk = require('chalk'); const log = console.log; // Combine styled and normal strings log(chalk.blue('Hello') + ' World' + chalk.red('!')); // Compose multiple styles using the chainable API log(chalk.blue.bgRed.bold('Hello world!')); // Pass in multiple arguments log(chalk.blue('Hello', 'World!', 'Foo', 'bar', 'biz', 'baz')); // Nest styles log(chalk.red('Hello', chalk.underline.bgBlue('world') + '!')); // Nest styles of the same type even (color, underline, background) log(chalk.green( 'I am a green line ' + chalk.blue.underline.bold('with a blue substring') + ' that becomes green again!' )); // ES2015 template literal log(` CPU: ${chalk.red('90%')} RAM: ${chalk.green('40%')} DISK: ${chalk.yellow('70%')} `); // ES2015 tagged template literal log(chalk` CPU: {red ${cpu.totalPercent}%} RAM: {green ${ram.used / ram.total \* 100}%} DISK: {rgb(255,131,0) ${disk.used / disk.total \* 100}%} `); // Use RGB colors in terminal emulators that support it. log(chalk.keyword('orange')('Yay for orange colored text!')); log(chalk.rgb(123, 45, 67).underline('Underlined reddish color')); log(chalk.hex('#DEADED').bold('Bold gray!')); ``` Easily define your own themes: ```js const chalk = require('chalk'); const error = chalk.bold.red; const warning = chalk.keyword('orange'); console.log(error('Error!')); console.log(warning('Warning!')); ``` Take advantage of console.log [string substitution](https://nodejs.org/docs/latest/api/console.html#console\_console\_log\_data\_args): ```js const name = 'Sindre'; console.log(chalk.green('Hello %s'), name); //=> 'Hello Sindre' ``` ## API ### chalk.`