readme.md for @push.rocks/consolecolor

□□ **Beautiful terminal colors for Node.js** - Make your console output pop with vibrant ANSI colors!

Install

```
pnpm install @push.rocks/consolecolor
# or
npm install @push.rocks/consolecolor
```

What it does

`@push.rocks/consolecolor` transforms your boring console output into a vibrant, colorful experience. Perfect for CLI tools, logging systems, or any Node.js application that needs to make terminal output more readable and visually appealing.

Usage

□□ Quick Start

```
import * as consolecolor from '@push.rocks/consolecolor';

// Simple colored text
const blueText = consolecolor.coloredString('Hello World!', 'blue');
console.log(blueText);

// Text with background color
```

```
const alertMessage = consolecolor.coloredString('WARNING', 'red', 'white');
console.log(alertMessage);
```

□□ Available Colors

The module supports these vibrant colors:

- `black` - Deep black ●
- `blue` - Bright blue □□
- `brown` - Warm brown □□
- `cyan` - Cool cyan □□
- `green` - Fresh green □□
- `orange` - Vibrant orange □□
- `pink` - Soft pink □□
- `red` - Bold red □□
- `white` - Pure white ○

□□ API Reference

`coloredString(text, fontColor, backgroundColor?)`

Creates a colored string for terminal output.

Parameters:

- `text` (string): The text to colorize
- `fontColor` (TColorName): The color for the text
- `backgroundColor` (TColorName, optional): The background color

Returns: A string with ANSI color codes

```
// Basic usage
const simpleColored = consolecolor.coloredString('Status: OK', 'green');

// With background
const highlighted = consolecolor.coloredString('CRITICAL', 'white', 'red');
```

□□ Real-World Examples

Creating a Logger with Color-Coded Severity

```
import * as consolecolor from '@push.rocks/consolecolor';

class ColorLogger {
  info(message: string) {
    console.log(consolecolor.coloredString(`□ INFO: ${message}`, 'cyan'));
  }

  success(message: string) {
    console.log(consolecolor.coloredString(`✔ SUCCESS: ${message}`, 'green'));
  }

  warning(message: string) {
    console.log(consolecolor.coloredString(`⚠ WARNING: ${message}`, 'orange', 'black'));
  }

  error(message: string) {
    console.log(consolecolor.coloredString(`✖ ERROR: ${message}`, 'red', 'white'));
  }
}

const logger = new ColorLogger();
logger.info('Application started');
logger.success('Database connected');
logger.warning('Memory usage above 80%');
logger.error('Failed to write to disk');
```

Progress Indicator with Colors

```
import * as consolecolor from '@push.rocks/consolecolor';

function showProgress(step: number, total: number, status: 'pending' | 'running' | 'done') {
  const percentage = Math.round((step / total) * 100);
  const progressBar = '▓'.repeat(percentage / 5) + '░'.repeat(20 - percentage / 5);

  let statusColor: 'orange' | 'blue' | 'green' = 'orange';
  if (status === 'running') statusColor = 'blue';
  if (status === 'done') statusColor = 'green';

  const output = consolecolor.coloredString(
    `[${progressBar}] ${percentage}% - ${status}`,
```

```
    statusColor
  );

  process.stdout.write('\r' + output);
}

// Usage
showProgress(5, 10, 'running');
```

Colorful Data Table

```
import * as consolecolor from '@push.rocks/consolecolor';

function printColorfulTable(data: Array<{name: string, status: string, value: number}>) {
  // Header
  console.log(consolecolor.coloredString('━'.repeat(50), 'cyan'));
  console.log(
    consolecolor.coloredString('Name', 'white') + '\t\t' +
    consolecolor.coloredString('Status', 'white') + '\t\t' +
    consolecolor.coloredString('Value', 'white')
  );
  console.log(consolecolor.coloredString('━'.repeat(50), 'cyan'));

  // Data rows
  data.forEach(row => {
    const statusColor = row.status === 'active' ? 'green' :
                        row.status === 'pending' ? 'orange' : 'red';
    const valueColor = row.value > 80 ? 'green' :
                       row.value > 40 ? 'orange' : 'red';

    console.log(
      row.name + '\t\t' +
      consolecolor.coloredString(row.status, statusColor) + '\t\t' +
      consolecolor.coloredString(row.value.toString(), valueColor)
    );
  });

  console.log(consolecolor.coloredString('━'.repeat(50), 'cyan'));
}
```

```
// Example usage
printColorfulTable([
  { name: 'Service A', status: 'active', value: 95 },
  { name: 'Service B', status: 'pending', value: 60 },
  { name: 'Service C', status: 'failed', value: 15 }
]);
```

□□ TypeScript Support

This module is written in TypeScript and provides full type definitions out of the box:

```
import type { TColorName, IRGB } from '@push.rocks/consolecolor';

// TColorName type for color validation
const myColor: TColorName = 'blue'; // □ Valid
// const invalid: TColorName = 'purple'; // □ TypeScript error

// IRGB interface for RGB values
const customRGB: IRGB = { r: 5, g: 3, b: 1 }; // Used internally
```

□□ Use Cases

- **CLI Tools**: Make your command-line tools more user-friendly with color-coded output
- **Logging Systems**: Differentiate log levels with distinct colors
- **Build Scripts**: Highlight errors, warnings, and success messages
- **Development Tools**: Color-code test results, linting output, or deployment statuses
- **Data Visualization**: Create simple colored charts and graphs in the terminal
- **Interactive CLIs**: Guide users with colored prompts and responses

□□ Best Practices

1. **Use colors consistently** - Establish a color scheme (e.g., red for errors, green for success)
2. **Consider accessibility** - Not everyone can distinguish all colors equally
3. **Provide non-color alternatives** - Use symbols alongside colors (✔, ✖, ⚠)
4. **Test in different terminals** - Colors may appear differently across terminal emulators
5. **Keep it readable** - Avoid color combinations that are hard to read

License and Legal Information

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the license file within this repository.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH and are not included within the scope of the MIT license granted herein. Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines, and any usage must be approved in writing by Task Venture Capital GmbH.

Company Information

Task Venture Capital GmbH
Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

Revision #3
Created 2026-03-28 11:09:51 UTC by foss.global Team
Updated 2026-03-28 12:16:27 UTC by foss.global Team