color-support

A module which will endeavor to guess your terminal's level of color support.

This is similar to `supports-color`, but it does not read `process.argv`.

1. If not in a node environment, not supported.
2. If stdout is not a TTY, not supported, unless the `ignoreTTY` option is set.
3. If the `TERM` environ is `dumb`, not supported, unless the `ignoreDumb` option is set.
4. If on Windows, then support 16 colors.
5. If using Tmux, then support 256 colors.
6. Handle continuous-integration servers. If `CI` or `TEAMCITY_VERSION` are set in the environment, and `TRAVIS` is not set, then color is not supported, unless `ignoreCI` option is set.
7. Guess based on the `TERM_PROGRAM` environ. These terminals support 16m colors:
   - `iTerm.app` version 3.x supports 16m colors, below support 256
   - `MacTerm` supports 16m colors
   - `Apple_Terminal` supports 256 colors
   - Have more things that belong on this list? Send a PR!
8. Make a guess based on the `TERM` environment variable. Any `xterm-256color` will get 256 colors. Any screen, xterm, vt100, color, ansi, cygwin, or linux `TERM` will get 16 colors.
9. If `COLORTERM` environment variable is set, then support 16 colors.
10. At this point, we assume that color is not supported.

USAGE

```
var testColorSupport = require('color-support')
var colorSupport = testColorSupport(/* options object */)

if (!colorSupport) {
  console.log('color is not supported')
} else if (colorSupport.has16m) {
  console.log('\x1b[38;2;102;194;255m16m colors\x1b[0m')
} else if (colorSupport.has256) {
  console.log('\x1b[38;5;119m256 colors\x1b[0m')
} else if (colorSupport.hasBasic) {
  console.log('\x1b[31mbasic colors\x1b[0m')
} else {
  console.log('this is impossible, but colors are not supported')
}
```

If you don't have any options to set, you can also just look at the flags which will all be set on the test function itself. (Of course, this doesn't return a falsey value when colors aren't supported, and doesn't allow you to set options.)

```
var colorSupport = require('color-support')

if (colorSupport.has16m) {
  console.log('\x1b[38;2;102;194;255m16m colors\x1b[0m')
} else if (colorSupport.has256) {
  console.log('\x1b[38;5;119m256 colors\x1b[0m')
} else if (colorSupport.hasBasic) {
  console.log('\x1b[31mbasic colors\x1b[0m')
} else {
  console.log('colors are not supported')
}
```

Options

You can pass in the following options.

- ignoreTTY - default false. Ignore the `isTTY` check.
- ignoreDumb - default false. Ignore `TERM=dumb` environ check.
- ignoreCI - default false. Ignore `CI` environ check.
- env - Object for environment vars. Defaults to `process.env`.
- stream - Stream for `isTTY` check. Defaults to `process.stdout`.
- term - String for `TERM` checking. Defaults to `env.TERM`.
- alwaysReturn - default false. Return an object when colors aren't supported (instead of returning `false`).
- level - A number from 0 to 3. This will return a result for the specified level. This is useful if you want to be able to set the color support level explicitly as a number in an environment variable or config, but then use the object flags in your program. Except for `alwaysReturn` to return an object for level 0, all other options are ignored, since no checking is done if a level is explicitly set.

Return Value

If no color support is available, then `false` is returned by default, unless the `alwaysReturn` flag is set to `true`. This is so that the simple question of "can I use colors or not" can treat any truthy return as "yes".

Otherwise, the return object has the following fields:

- `level` - A number from 0 to 3
  - `0` - No color support
  - `1` - Basic (16) color support
  - `2` - 256 color support
  - `3` - 16 million (true) color support
- `hasBasic` - Boolean
- `has256` - Boolean
- `has16m` - Boolean

CLI

You can run the `color-support` bin from the command line which will just dump the values as this module calculates them in whatever env it's run. It takes no command line arguments.

Credits

This is a spiritual, if not actual, fork of supports-color by the ever prolific Sindre Sorhus.