How Selenized improves on Solarized

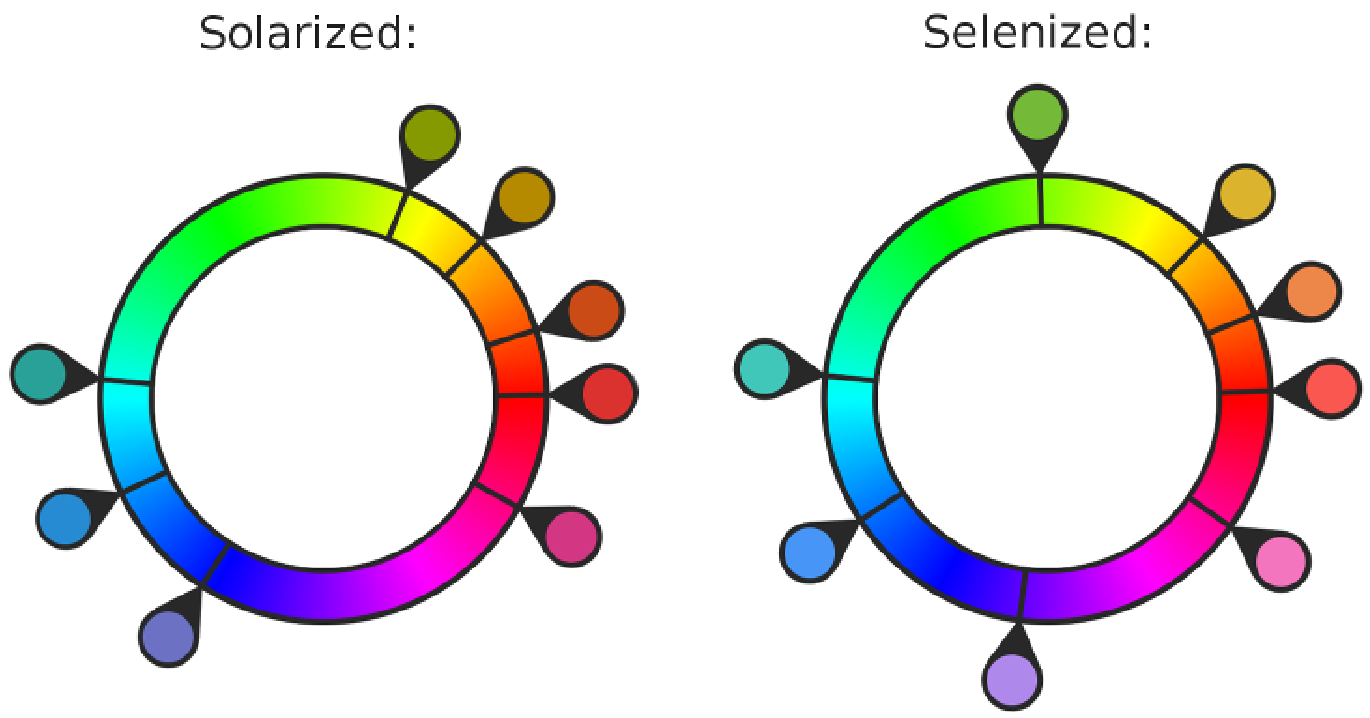
I really liked the design principles behind Solarized - it has some great features. However, it has a few problems as well, which Selenized solves:

- Confusing accent colors
- Not enough contrast
- Dark version is too dark
- Hacky implementation in terminals

Better accent colors

Some accent colors in Solarized are confusing or too similar to each other:

- green is too much like yellow (looks like olive)
- orange looks almost like red
- magenta too close to red as well
- violet is easy to confuse with blue



This is less of a problem on high-end, color-calibrated displays, but on a regular screen (like the one on my ThinkPad from a couple years ago) it can be really difficult to tell some colors apart. Selenized makes the differences clearer.

Slightly higher contrast

Ethan says on Solarized website that he had tested it in a variety of lighting conditions. However, in practice I have encountered numerous situations when Solarized readability was not satisfactory. This is confirmed by Web Content Accessibility Guidelines, which requires contrast of 4.5 for AA grade and 7.0 for AAA grade:

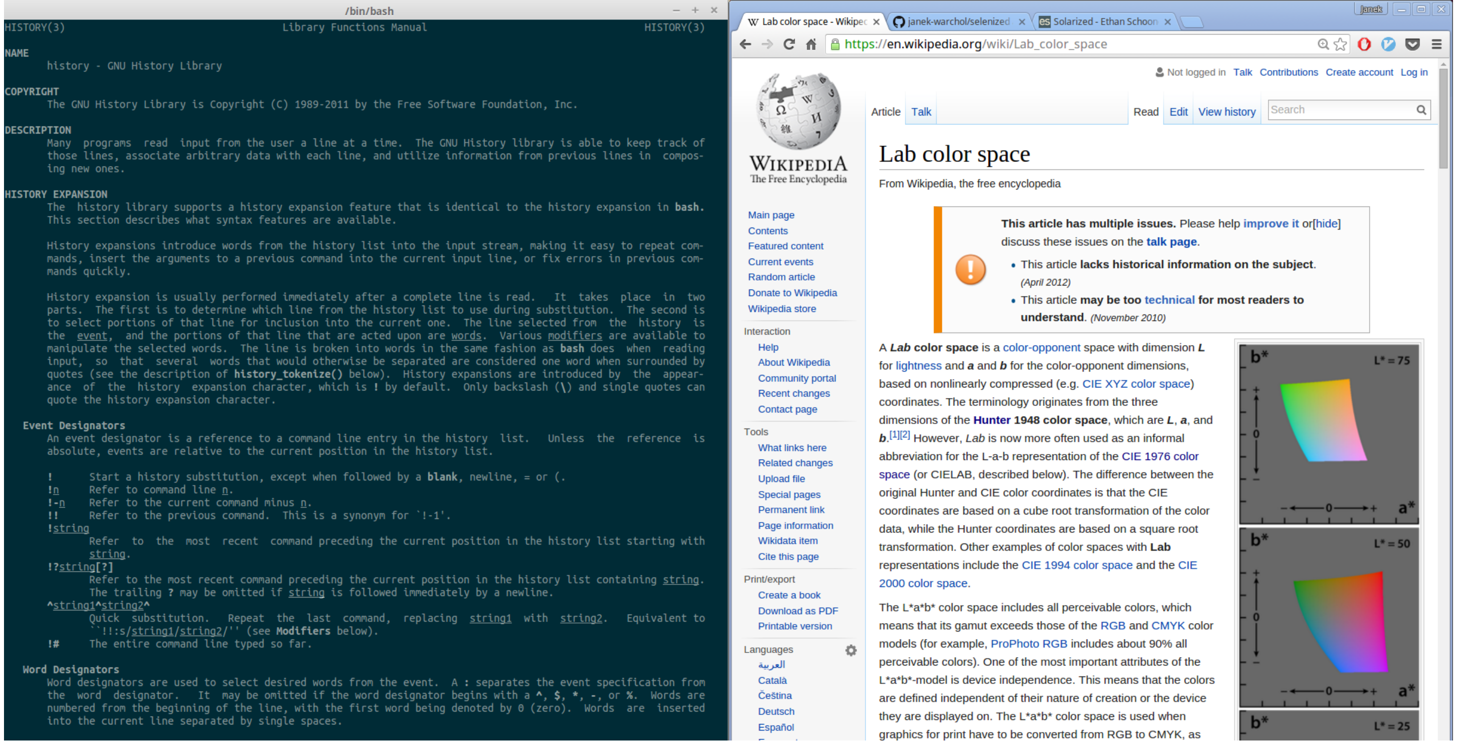
color/color combination	Solarized	Selenized dark	Selenized black
main content (foreground)	4.75 - AA ✔️	6.07 - AA ✔️	9.05 - AAA ✔️✔️
fg on bg highlight	4.11 - 🤔*	5.04 - AA ✔️	7.81 - AAA ✔️✔️
secondary content (comments)	2.79 - AA ❌	3.23 - 🤔*	3.97 - 🤔*
red (darkest accent color)	3.25 - 🤔*	3.71 - 🤔*	4.79 - AA ✔️

🤔* *WCAG says that contrast of 3.0 is acceptable if the font has sufficient size and weight. 3.0 is also the minimum contrast for body text required by ISO-9241-3.*

Contrast in Selenized remains moderately low, but it is significantly more readable in poor lighting than Solarized.

Better lightness

Solarized dark may work well when used all by itself, but it's *too dark* when placed next to a window with high-contrast content:



```
/bin/bash
HISTORY(3)                        Library Functions Manual                        HISTORY(3)

NAME
       history - GNU History Library

COPYRIGHT
       The GNU History Library is Copyright (C) 1989-2011 by the Free Software Foundation, Inc.

DESCRIPTION
       Many  programs  read  input from the user a line at a time.  The GNU History library is able to keep track of
       those lines, associate arbitrary data with each line, and utilize information from previous lines in  compos-
       ing new ones.

HISTORY EXPANSION
       The  history library supports a history expansion feature that is identical to the history expansion in bash.
       This section describes what syntax features are available.

       History expansions introduce words from the history list into the input stream, making it easy to repeat com-
       mands, insert the arguments to a previous command into the current input line, or fix errors in previous com-
       mands quickly.

       History expansion is usually performed immediately after a complete line is read.   It  takes  place  in  two
       parts.  The first is to determine which line from the history list to use during substitution.  The second is
       to select portions of that line for inclusion into the current one.  The line selected from  the  history  is
       the  event,  and the portions of that line that are acted upon are words.  Various modifiers are available to
       manipulate the selected words.  The line is broken into words in the same fashion as bash does  when  reading
       input,  so  that  several  words that would otherwise be separated are considered one word when surrounded by
       quotes (see the description of history_tokenize() below).  History expansions are introduced by  the  appear-
       ance  of  the  history  expansion character, which is ! by default.  Only backslash (\) and single quotes can
       quote the history expansion character.

   Event Designators
       An event designator is a reference to a command line entry in the history  list.   Unless  the  reference  is
       absolute, events are relative to the current position in the history list.

       !      Start a history substitution, except when followed by a blank, newline, = or (.
       !n     Refer to command line n.
       !-n    Refer to the current command minus n.
       !!     Refer to the previous command.  This is a synonym for `!-1'.
       !string
              Refer  to  the  most  recent  command preceding the current position in the history list starting with
              string.
       !?string[?]
              Refer to the most recent command preceding the current position in the history list containing string.
              The trailing ? may be omitted if string is followed immediately by a newline.
       ^string1^string2^
              Quick  substitution.   Repeat  the  last  command,  replacing  string1  with  string2.   Equivalent to
              ``!!:s/string1/string2/'' (see Modifiers below).
       !#     The entire command line typed so far.

   Word Designators
       Word designators are used to select desired words from the event.  A : separates the event specification from
       the  word  designator.   It  may be omitted if the word designator begins with a ^, $, *, -, or %.  Words are
       numbered from the beginning of the line, with the first word being denoted by 0 (zero).  Words  are  inserted
       into the current line separated by single spaces.
```

W Lab color space - Wikipe× janek-warchol/selenized× Solarized - Ethan Schoon×

https://en.wikipedia.org/wiki/Lab_color_space

Not logged in Talk Contributions Create account Log in

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
العربية
Català
Čeština
Deutsch
Español

Article Talk Read Edit View history Search

Lab color space

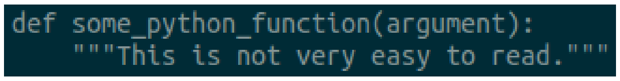
From Wikipedia, the free encyclopedia

This article has multiple issues. Please help **improve it** or[hide]
discuss these issues on the **talk page**.
- This article **lacks historical information on the subject**. *(April 2012)*
- This article **may be too technical for most readers to understand**. *(November 2010)*

A ***Lab* color space** is a color-opponent space with dimension ***L*** for lightness and ***a*** and ***b*** for the color-opponent dimensions, based on nonlinearly compressed (e.g. CIE XYZ color space) coordinates. The terminology originates from the three dimensions of the **Hunter 1948 color space**, which are ***L***, ***a***, and ***b***.[1][2] However, *Lab* is now more often used as an informal abbreviation for the L-a-b representation of the CIE 1976 color space (or CIELAB, described below). The difference between the original Hunter and CIE color coordinates is that the CIE coordinates are based on a cube root transformation of the color data, while the Hunter coordinates are based on a square root transformation. Other examples of color spaces with **Lab** representations include the CIE 1994 color space and the CIE 2000 color space.

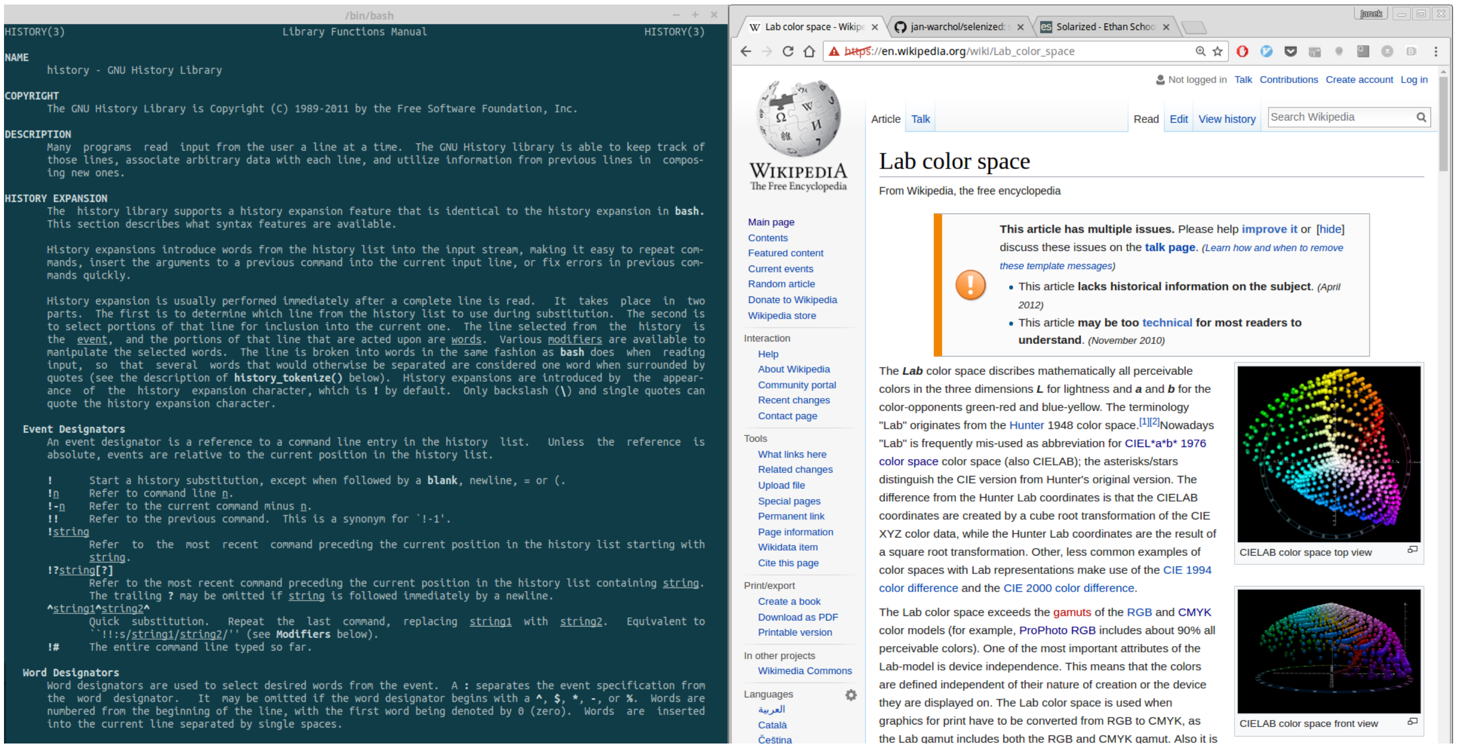
The L*a*b* color space includes all perceivable colors, which means that its gamut exceeds those of the RGB and CMYK color models (for example, ProPhoto RGB includes about 90% all perceivable colors). One of the most important attributes of the L*a*b*-model is device independence. This means that the colors are defined independent of their nature of creation or the device they are displayed on. The L*a*b* color space is used when graphics for print have to be converted from RGB to CMYK, as

This is even more visible when Solarized dark is used for code snippets on a website with white background:



```
def some_python_function(argument):
    """This is not very easy to read."""
```

Selenized dark, being slightly lighter and having more contrast, doesn't have this problem:



```
/bin/bash
HISTORY(3)                        Library Functions Manual                        HISTORY(3)

NAME
       history - GNU History Library

COPYRIGHT
       The GNU History Library is Copyright (C) 1989-2011 by the Free Software Foundation, Inc.

DESCRIPTION
       Many  programs  read  input from the user a line at a time.  The GNU History library is able to keep track of
       those lines, associate arbitrary data with each line, and utilize information from previous lines in  compos-
       ing new ones.

HISTORY EXPANSION
       The  history library supports a history expansion feature that is identical to the history expansion in bash.
       This section describes what syntax features are available.

       History expansions introduce words from the history list into the input stream, making it easy to repeat com-
       mands, insert the arguments to a previous command into the current input line, or fix errors in previous com-
       mands quickly.

       History expansion is usually performed immediately after a complete line is read.   It  takes  place  in  two
       parts.  The first is to determine which line from the history list to use during substitution.  The second is
       to select portions of that line for inclusion into the current one.  The line selected from  the  history  is
       the  event,  and the portions of that line that are acted upon are words.  Various modifiers are available to
       manipulate the selected words.  The line is broken into words in the same fashion as bash does  when  reading
       input,  so  that  several  words that would otherwise be separated are considered one word when surrounded by
       quotes (see the description of history_tokenize() below).  History expansions are introduced by  the  appear-
       ance  of  the  history  expansion character, which is ! by default.  Only backslash (\) and single quotes can
       quote the history expansion character.

   Event Designators
       An event designator is a reference to a command line entry in the history  list.   Unless  the  reference  is
       absolute, events are relative to the current position in the history list.

       !      Start a history substitution, except when followed by a blank, newline, = or (.
       !n     Refer to command line n.
       !-n    Refer to the current command minus n.
       !!     Refer to the previous command.  This is a synonym for `!-1'.
       !string
              Refer  to  the  most  recent  command preceding the current position in the history list starting with
              string.
       !?string[?]
              Refer to the most recent command preceding the current position in the history list containing string.
              The trailing ? may be omitted if string is followed immediately by a newline.
       ^string1^string2^
              Quick  substitution.   Repeat  the  last  command,  replacing  string1  with  string2.   Equivalent to
              ``!!:s/string1/string2/'' (see Modifiers below).
       !#     The entire command line typed so far.

   Word Designators
       Word designators are used to select desired words from the event.  A : separates the event specification from
       the  word  designator.   It  may be omitted if the word designator begins with a ^, $, *, -, or %.  Words are
       numbered from the beginning of the line, with the first word being denoted by 0 (zero).  Words  are  inserted
       into the current line separated by single spaces.
```

W Lab color space - Wikipe× jan-warchol/selenized× Solarized - Ethan Schoo×

~~https~~://en.wikipedia.org/wiki/Lab_color_space

Not logged in Talk Contributions Create account Log in

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

In other projects
Wikimedia Commons

Languages
العربية
Català
Čeština

Article Talk Read Edit View history Search Wikipedia

Lab color space

From Wikipedia, the free encyclopedia

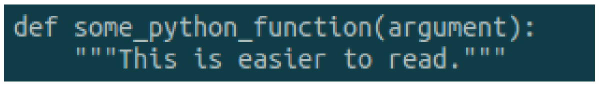
This article has multiple issues. Please help **improve it** or [hide]
discuss these issues on the **talk page**. *(Learn how and when to remove these template messages)*
- This article **lacks historical information on the subject**. *(April 2012)*
- This article **may be too technical for most readers to understand**. *(November 2010)*

The ***Lab*** color space discribes mathematically all perceivable colors in the three dimensions ***L*** for lightness and ***a*** and ***b*** for the color-opponents green-red and blue-yellow. The terminology "Lab" originates from the Hunter 1948 color space.[1][2]Nowadays "Lab" is frequently mis-used as abbreviation for CIEL*a*b* 1976 color space color space (also CIELAB); the asterisks/stars distinguish the CIE version from Hunter's original version. The difference from the Hunter Lab coordinates is that the CIELAB coordinates are created by a cube root transformation of the CIE XYZ color data, while the Hunter Lab coordinates are the result of a square root transformation. Other, less common examples of color spaces with Lab representations make use of the CIE 1994 color difference and the CIE 2000 color difference.

The Lab color space exceeds the gamuts of the RGB and CMYK color models (for example, ProPhoto RGB includes about 90% all perceivable colors). One of the most important attributes of the Lab-model is device independence. This means that the colors are defined independent of their nature of creation or the device they are displayed on. The Lab color space is used when graphics for print have to be converted from RGB to CMYK, as the Lab gamut includes both the RGB and CMYK gamut. Also it is

CIELAB color space top view

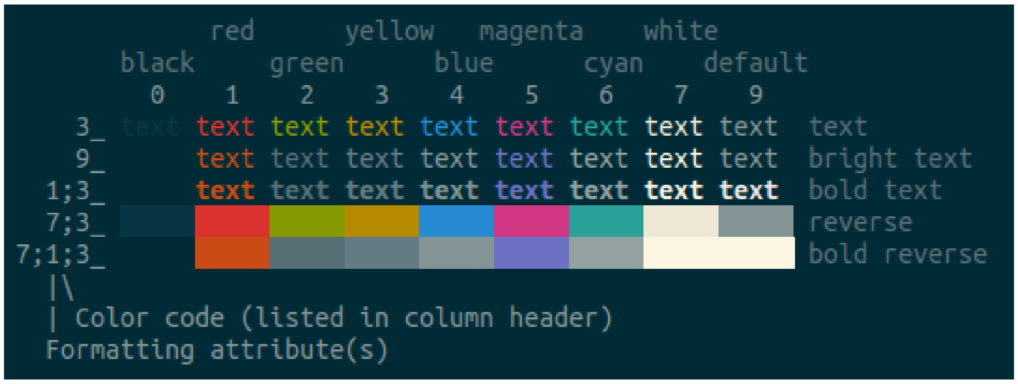
CIELAB color space front view



```
def some_python_function(argument):
    """This is easier to read."""
```

Better terminal compatibility

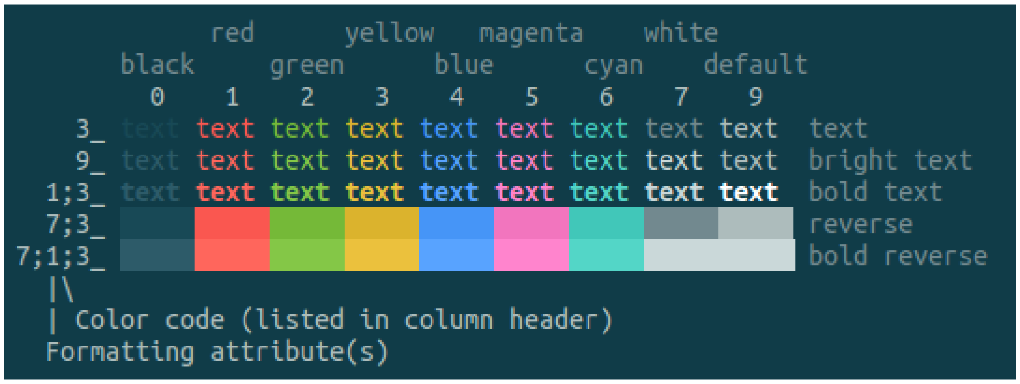
Solarized puts both light and dark variants in one color palette, resulting in a weird mapping of ANSI color codes to actual colors. For example, Solarized maps color code meant for bright/bold green to "base 01" (greyish shade used for comments):



```
            red      yellow   magenta    white
       black     green      blue      cyan     default
         0    1    2    3    4    5    6    7    9
     3_ text text text text text text text text text  text
     9_      text text text text text text text text  bright text
   1;3_      text text text text text text text text  bold text
   7;3_                                               reverse
 7;1;3_                                               bold reverse
   |\
   | Color code (listed in column header)
   Formatting attribute(s)
```

Because of that many command line programs will produce strange or unreadable output (see examples here and here).

Selenized keeps standard meaning of terminal color codes:



```
            red      yellow   magenta    white
       black     green      blue      cyan     default
         0    1    2    3    4    5    6    7    9
     3_ text text text text text text text text text  text
     9_ text text text text text text text text text  bright text
   1;3_ text text text text text text text text text  bold text
   7;3_                                               reverse
 7;1;3_                                               bold reverse
   |\
   | Color code (listed in column header)
   Formatting attribute(s)
```