

Search...

Getting started

Components

- M4Q
- About
- Population
- Constructor
- Ajax
- Animation
- Loops
- Visibility
- Effects
- Subtree functions
- Attributes
- Html, text and value
- Css and classes
- Position and size
- Manipulation
- DataSet
- Events
- Utils
- Base
- Containers
- Grid system
- Typography
- Tables
- Forms
- Buttons
- Images
- Figures
- Lists
- Form components
- Checkbox
- File input
- Input
- Input material
- Keypad
- Rating
- Radio
- Select
- Slider
- Double Slider
- Spinner
- Switch
- Tag input
- Textarea
- Menus
- App bar
- Bottom navigation
- Bottom sheet
- Menu
- Ribbon Menu
- Side bar
- Side navigation
- Controls
- Accordion
- Badge
- Carousel
- Cards
- Cube
- Counter
- Charms
- Chat
- Donut
- Image compare
- Image magnifier
- Gravatar
- List
- ListView
- Master
- NavView
- Panels
- Progress & Activity
- Streamer
- Stepper
- Splitter
- Tabs
- Tabs material
- Table
- Tiles
- TreeView
- Wizard
- Information
- Dialogs
- Info box
- Hints
- Notify system
- Popovers
- Toasts
- Windows
- Date & time
- Calendar
- Calendar picker
- Date picker
- Time picker
- Countdown
- Formatting date

Color module

Metro 4 has a built-in module for working with colors. You can manipulate and transform colors and can create any color schemes.

About

All routines for works with colors are in the object `Metro.colors`. Module works with formats: `rgb`, `rgba`, `hex`, `hsv`, `hsl` and `cmky`. You can use color conversion functions, as well as functions for generating common color schemes `monochromatic`, `complementary`, `split-complementary`, `double-complementary`, `analogous`, `triadic`, `tetradic` and `square`. Also you can use special functions `grayscale`, `darken`, `lighten`.

Formats

Module works with any formats: `rgb`, `rgba`, `hex`, `hsv`, `hsl` and `cmky`. Each format is used as an `object`, exclude `hex` format. It stored as a `string`.

Color	Desc
hex	A <code>hex</code> color is specified with: <code>#RRGGBB</code> . RR (red), GG (green) and BB (blue) are hexadecimal integers between 00 and FF specifying the intensity of the color. For example, #0000FF is displayed as blue, because the blue component is set to its highest value (FF) and the others are set to 00.
rgb	A <code>rgb</code> color is specified with: <code>{r, g, b}</code> object. An RGB color value is specified with: <code>rgb(red, green, blue)</code> . Each parameter (red, green, and blue) defines the intensity of the color as an integer between 0 and 255. For example, <code>rgb(0, 0, 255)</code> is rendered as blue, because the blue parameter is set to its highest value (255) and the others are set to 0.
rgba	A <code>rgba</code> color is specified with: <code>{r, g, b, a}</code> object. An RGBA color value is specified with: <code>rgb(red, green, blue, alpha)</code> . Each parameter (red, green, and blue) defines the intensity of the color as an integer between 0 and 255 and alpha specified opacity from 0 to 1.
hsv	HSV stands for hue, saturation, and value. A <code>hsv</code> color is specified with: <code>{h, s, v}</code> object.
hsl	HSL stands for hue, saturation, and lightness. A <code>hsl</code> color is specified with: <code>{h, s, l}</code> object.
cmky	CMYK colors is a combination of <code>CYAN</code> , <code>MAGENTA</code> , <code>YELLOW</code> , and <code>BLACK</code> . A <code>cmky</code> color is specified with: <code>{c, m, y, k}</code> object.

Palettes

Module contains three color palettes: `metro` - 21 colors, `standard` - 140 colors and mix `metro` and `standard` with metro in priority. You can get color names with function `palette(name)`.

```
var metro = Metro.colors.palette(Metro.colors.PALETTES.METRO);
var standard = Metro.colors.palette(Metro.colors.PALETTES.STANDARD);
var all = Metro.colors.palette(Metro.colors.PALETTES.ALL);

...result for metro
[
  "lime", "green", "emerald", "blue", "teal", "cyan", "cobalt",
  "indigo", "violet", "pink", "magenta", "crimson", "red", "orange",
  "amber", "yellow", "brown", "olive", "steel", "mauve", "taupe"
]
```

You can get color `hex` value with function `color(name, palette)`.

```
var col = Metro.colors.color("cyan", Metro.colors.PALETTES.METRO);

...value for col is #1ba1e2
```

Check & convert

Metro 4 contains functions for check color values and convert colors from onw to others.

Check functions

The following functions are used to check the color value:

Function	Desc
isColor(val)	Check if value is a metro 4 color value
isHEX(val)	Check if value is a hex color value
isRGB(val)	Check if value is a rgb color value
isRGBA(val)	Check if value is a rgba color value
isHSV(val)	Check if value is a hsv color value
isHSL(val)	Check if value is a hsl color value
isHSLA(val)	Check if value is a hsla color value
isCMYK(val)	Check if value is a hsl color value
is(val)	Return type of color value
isLight(color)	Return true if color is light
isDark(color)	Return true if color is dark

Table of contents

- Color module
- About
- Formats
- Palettes
- Check & convert
- Tone functions
- Schemes
- Monochromatic
- Complementary
- Split-complementary
- Double-complementary
- Analogous
- Triadic
- Tetradic
- Square
- Websafe colors

- Media
- Video player
- Audio player
- Tools
- Collapse
- Color module
- Draggable
- Dropdown
- Form validator
- Hotkeys
- Micro templates
- Ripple
- Storage
- Session storage
- Sorter
- Touch and swipe

- Utilities
- Animations
- Additional

Convert functions

The following functions are used to convert a color value from one to others in Metro 4 color formats:

Function	Desc
<code>toHEX(val)</code>	Any type color value to hex: <code>#RRGGBB</code>
<code>toRGB(val)</code>	Any type color value to rgb: <code>{r, g, b}</code>
<code>toRGBA(val, alpha)</code>	Any type color value to rgba: <code>{r, g, b, a}</code>
<code>toHSV(val)</code>	Any type color value to hsv: <code>{h, s, v}</code>
<code>toHSL(val)</code>	Any type color value to hsl: <code>{h, s, l}</code>
<code>toHSLA(val)</code>	Any type color value to hsla: <code>{h, s, l, a}</code>
<code>toCMYK(val)</code>	Any type color value to hsl: <code>{c, m, y, k}</code>

```
var cm = Metro.colors;
var c = "#cc0000";

cm.toHEX(c); // #cc0000
cm.toRGB(c); // {r: 204, g: 0, b: 0}
cm.toRGBA(c, .5); // {r: 204, g: 0, b: 0, a: 0.5}
cm.toHSV(c); // {h: 0, s: 1, v: 0.8}
cm.toHSL(c); // {h: 0, s: 1, l: 0.4}
cm.toCMYK(c); // {c: 0, m: 100, y: 100, k: 20}
```



The following functions are used to convert a color value to string value:

Function	Desc
<code>toHexString(val)</code>	Return hexadecimal string for color: <code>#RRGGBB</code>
<code>toRgbString(val)</code>	Return rgb string for color: <code>rgb(r, g, b)</code>
<code>toRgbaString(val)</code>	Return rgba string for color: <code>rgba(r, g, b, a)</code>
<code>toHsvString(val)</code>	Return hsv string for color: <code>hsv(h, s, v)</code>
<code>toHslString(val)</code>	Return hsl string for color: <code>hsl(h, s, l)</code>
<code>toHslaString(val)</code>	Return hsl string for color: <code>hsl(h, s, l)</code>
<code>toCmykString(val)</code>	Return cmyk string for color: <code>cmyk(c, m, y, k)</code>
<code>toString(val)</code>	Return string for color value

```
var cm = Metro.colors;
var c = "#cc0000";

cm.toHexString(c); // #cc0000
cm.toRgbString(c); // rgb(204,0,0)
cm.toRgbaString(c); // rgba(204,0,0,1)
cm.toHsvString(c); // hsv(0,1,0.8)
cm.toHslString(c); // hsl(0,1,0.4)
cm.toCmykString(c); // cmyk(0,100,100,20)
```



Tone functions

Metro 4 color module contains functions to change color tone: `grayscale`, `lighten` and `darken`.

grayscale	lighten	normal	darken
-----------	---------	--------	--------

Function	Desc
<code>grayscale(color)</code>	Calculate grayscale color for color value
<code>lighten(color, percent)</code>	Lighten color. Percent value must be from 0 to 100.
<code>darken(color, percent)</code>	Darken color. Percent value must be from 0 to 100.

```
var cm = Metro.colors;
var c = "#cc0000";

cm.grayscale(c); // #2b2b2b
cm.lighten(c, 10); // #d60a0a
cm.darken(c, 10); // #c20000
```



Schemes

Metro 4 color module contains function `getScheme()`. This function generate any color schemes from base color: `monochromatic`, `complementary`, `split-complementary`, `double-complementary`, `analogous`, `triadic`, `tetradic` and `square`.

Function	Desc
<code>getScheme(color, name, format, options)</code>	Generate color scheme from base color. <code>color</code> - color in Metro 4 color module format, <code>name</code> - scheme name, <code>format</code> - format for return values (hex, rgb, rgba, hsv, hsl, cmyk), <code>options</code> - additional options.

```
var cm = Metro.colors;
var colors = cm.getScheme("#cc0000", "mono", "hex", {algorithm: 1});
```

Options

Options	Desc
angle	Angle for analogous and split-complementary schemes
algorithm	Algorithm for monochromatic schemes (1, 2, 3, 4)
tint1, tint2	This option use for monochromatic schemes 1 for create two left colors
shadel, shade2	This option use for monochromatic schemes 1 for create two right colors
distance	This option use for monochromatic schemes 2, 3 and determines how many colors will be generated
step	This option use for monochromatic schemes 2, 3, 4 and determines with what step the color will be generated. Default is 0.1 . Value must be in range from 0 to 1.
alpha	This option use for default alpha channel value for rgba colors.

Also you can set options with function `setup({...})` .


```
var cm = Metro.colors, colors;

cm.setup({
  alpha: .5,
  algorithm: 1
})

colors = cm.getScheme("#cc0000", "mono", "rgba");
```

Monochromatic

Monochromatic colors are all the colors (tones, tints and shades) of a single hue. Monochromatic color schemes are derived from a single base hue and extended using its shades, tones and tints. Tints are achieved by adding white and shades and tones are achieved by adding a darker color, grey or black. Monochromatic color schemes provide opportunities in art and visual communications design as they allow for a greater range of contrasting tones that can be used to attract attention, create focus and support legibility.



#ffffcc

#ffff66

#ffff00

#999900


#4d4d00

```
var cm = Metro.colors,
    colors, color = "#ffff00";

colors = cm.getScheme(color, "mono", "hex");
```

Analogous

Analogous (or adjacent colors) is a color scheme using one base color and two secondary colors placed symetrically around it on the color wheel. The base color is main, while the secondary colors should be used only for highlights and accents. It always looks very elegant and clear. There is no tension in the palette, and typically it's uniformly all warm, or all cold. If a base color on the warm-cold border is chosen, the color with opposite "temperature" may be used for accenting the other two colors.



#80ff00

#ffff00


#ff8000

```
var cm = Metro.colors,
    colors, color = "#ffff00";

colors = cm.getScheme(color, "analog", "hex");
```

Complementary

Complementary is a color scheme using one base color and its complement, the color on the exact opposite side of the color wheel. The base color is main and dominant, while the complementary color is used only as an accent.



#ffff00

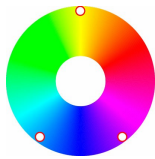
#0000ff

```
var cm = Metro.colors,
    colors, color = "#ffff00";

colors = cm.getScheme(color, "complement", "hex");
```

Split-complementary

Split-complementary is a color scheme using one base color and two secondary colors. Instead of using a complementary color, two colors placed symmetrically around it on the color wheel are used. The base color is main, while the secondary colors should be used only for highlights and accents.



#0080ff

#ffff00

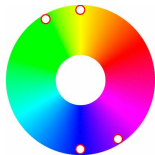
#8000ff

```
var cm = Metro.colors,
    colors, color = "#ffff00";

colors = cm.getScheme(color, "split", "hex");
```

Double-complementary

Double-complementary is a color scheme using two sets of base color and their complements with next rules on wheel: 180 °, -30 °, 180 °. Both base colors are equivalent, cannot be decided which one should be the main color (though a designer could choose one). Less distance between two base colors causes less tension in the result. However, this scheme is always more "nervous" and "action" than other schemes. While working with it, we have to take care especially of relations between one color and the complement of its adjacent color.



#ffff00

#0000ff

#8000ff

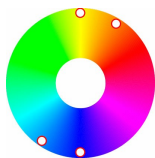
#80ff00

```
var cm = Metro.colors,
    colors, color = "#ffff00";

colors = cm.getScheme(color, "double", "hex");
```

Tetradic

Tetradic is a color scheme using two sets of base color and their complements with next rules on wheel: 180 °, 30 °, 180 °. Both base colors are equivalent, cannot be decided which one should be the main color (though a designer could choose one). Less distance between two base colors causes less tension in the result. However, this scheme is always more "nervous" and "action" than other schemes. While working with it, we have to take care especially of relations between one color and the complement of its adjacent color.



#ffff00

#0000ff

#ff8000

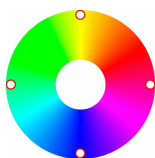
#0080ff

```
var cm = Metro.colors,
    colors, color = "#ffff00";

colors = cm.getScheme(color, "tetra", "hex");
```

Square

A square is a color scheme, a special variant of the dual color scheme, with the equal distance between all colors. All four colors are distributed evenly around the color wheel, causing there is no clear dominance of one color. The scheme is always vibrant, nervous and colorful, there is equal tension between all colors. Square is very aggressive color scheme, requiring very good planning and very sensitive approach to relations of these colors.



#ffff00

#00ff80

#0000ff

#ff0080

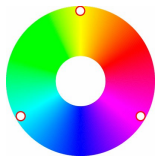
```
var cm = Metro.colors,
    colors, color = "#ffff00";

colors = cm.getScheme(color, "square", "hex");
```



Triadic

A triad is a color scheme, a special variant of the split-complementary color scheme, with the equal distance between all colors. All three colors are distributed evenly around the color wheel, causing there is no clear dominance of one color. The scheme is always vibrant and colorful, designers should use it and balance very carefully to maintain the desired effects and color meaning.



#ffff00

#00ffff

#ff00ff

```
var cm = Metro.colors,
    colors, color = "#ffff00";

colors = cm.getScheme(color, "triadic", "hex");
```



Additional examples: [schemes 1](#), [schemes 2](#), [grayscale](#).

Websafe colors

You can get [websafe](#) color from any. To get it, use function [websafe\(color\)](#) or specific for color format [hex2websafe](#) , [rgb2websafe](#) , [rgba2websafe](#) , [hsv2websafe](#) , [hsl2websafe](#) and [cmyk2websafe](#) .

#9d4232 - not safe

#993333 - websafe

```
var cm = Metro.colors,
    color = "#9d4232";

cm.websafe(color); // #993333
```

