
iWantHue

Colors for data scientists. *Generate and refine palettes of optimally distinct colors.*

iWantHue allows you to generate palettes of colors. It is about mastering the properties of a palette by setting a range of Hue, Chroma (unbiased saturation) and Lightness. You can generate palettes of any size or just get the generator for a javascript project. The algorithm optimizes the perceptive distance in the color subspace, ensuring an optimal readability.

How it works

1. **K-means** or **force vector repulsion** algorithms ensure an even distribution of colors
2. The **CIE Lab** color space is used for computation, since it fits human perception
3. The **Hue/Chroma/Lightness** color space is used to set constraints, since it is user-friendly

Examples and a tutorial

Idea

The idea behind *iWantHue* is to distribute colors **evenly**, in a **perceptively coherent** space, constrained by **user-friendly** settings, to generate **high quality** custom palettes.

Explanations and an experiment on color theory

How to use it in your own code

You can install `iwanthue` for node.js and the browser using npm:

```
1 npm install iwanthue
```

Usage

```
1 var iwanthue = require('iwanthue');
2
3 // Generate a simple palette with 5 colors
4 var palette = iwanthue(5);
5
6 // With some options
7 var palette = iwanthue(5, {
8   clustering: 'force-vector',
9   seed: 'cool-palette',
```

```
10   quality: 100
11 });
```

Arguments

- **count** *number*: number of colors in the generated palette.
- **settings** *?object*: Settings:
 - **attempts** *?number* [1]: Number of times should the function try to generate a palette in order to only keep the one maximizing the minimum distance between two of the palette colors. Will only make a single attempt by default.
 - **colorSpace** *?string|object|array* [**default**]: Color space preset. Check this file for the full list of preset names, or go to the website to try them. Alternatively you can pass the **colorSpace** either as a [hmin, hmax, cmin, cmax, lmin, lmax] array or a {hmin, hmax, cmin, cmax, lmin, lmax} object where keys can be omitted.
 - **colorFilter** *?function*: Function used to filter suitable colors. Takes a [r, g, b] color and the same [l, a, b] color as arguments.
 - **clustering** *?string* [k-means]: Clustering method to use. Can either be k-means or force-vector.
 - **quality** *?number* [50]: Quality of the clustering: iterations factor for force-vector, colorspace sampling for k-means.
 - **ultraPrecision** *?boolean* [**false**]: Ultra precision for k-means colorspace sampling?
 - **distance** *?string* [euclidean]: Distance function to use. Can be euclidean, cmc, compromise (colorblind), protanope, deutanope or tritanope.
 - **seed** *?string|number*: Random number generator seed. Useful to produce the same palette every time based on some data attribute.

Palette

A helper class representing a categorical color palette over a set of given values.

```
1 // To build your palette from a unique list of values
2 var Palette = require('iwanthue/palette');
3
4 var palette = Palette.generateFromValues(
5   'cities',
6   ['one', 'two', 'three'],
7   {defaultColor: '#000'}
8 );
9
10 palette.get('one');
11 >>> '#19d3a2'
12
```

```
13 palette.get('unknown');
14 >>> '#000'
15
16 // From entries
17 var palette = Palette.fromEntries(
18   'cities',
19   [['one', 'red'], ['two', 'blue']],
20   '#000'
21 );
22
23 // From mapping
24 var palette = Palette.fromMapping(
25   'cities',
26   {one: 'red', two: 'blue'},
27   'red'
28 );
```

Palette.generateFromValues arguments

- **name** *string*: palette or category name used as [seed](#) for iwanthue.
- **values** *iter*: unique values for the represented category.
- **settings** *?object*:
 - **defaultColor** *?string* [[#ccc](#)]: default color to return in case desired value is not known.
 - ... any setting that can be passed to iwanthue.

Palette.fromEntries arguments

- **name** *string*: palette or category name used as [seed](#) for iwanthue.
- **entries** *iter*: key, value entries mapping values to colors.
- **defaultColor** *?string* [[#ccc](#)]: default color to return in case desired value is not known.

Palette.fromMapping arguments

- **name** *string*: palette or category name used as [seed](#) for iwanthue.
- **mapping** *object|Map*: mapping from values to colors.
- **defaultColor** *?string* [[#ccc](#)]: default color to return in case desired value is not known.

Palette.fromJSON arguments

- **data** *object*: serialized palette data ([{name, defaultColor, entries}](#)).

Palette members

- **name** *string*: name of the palette.
- **size** *number*: number of colors.
- **defaultColor** *string*: default color.

-
- **map** *Map*: map from values to colors.

Palette methods

- **get**: return the color for the given value or the default color if value is unknown.
- **has**: return whether the value is known to the palette.
- **forEach**: callback iteration over (color, value).
- **colors**: return the palette's colors as an array.
- **toJSON**: return the palette in a serialized format fit for JSON.

Precomputed palettes

If you don't want to load iwanthue's whole code into your client app or if you just want to prototype things quickly, the npm module also packs some precomputed palettes that you can import.

All of the presets export palettes ranging from 2 to 15 colors.

Here is how to load them:

```
1 // Default palettes
2 var palettes = require('iwanthue/precomputed');
3
4 // Need a palette for 6 colors:
5 var sixColorsPalette = palettes[6];
6
7 // K-means palettes
8 var palettes = require('iwanthue/precomputed/k-means');
9
10 // Force vector palettes
11 var palettes = require('iwanthue/precomputed/force-vector');
12
13 // Colorblind alternatives
14 var palettes = require('iwanthue/precomputed/k-means-colorblind');
15 var palettes = require('iwanthue/precomputed/force-vector-colorblind');
```

Here is the full list of precomputed palettes:

```
1 iwanthue/precomputed/force-vector
2 iwanthue/precomputed/index
3 iwanthue/precomputed/k-means
4 iwanthue/precomputed/ultra-k-means
5 iwanthue/precomputed/force-vector-colorblind
6 iwanthue/precomputed/k-means-colorblind
7 iwanthue/precomputed/k-means-all
8 iwanthue/precomputed/force-vector-all
9 iwanthue/precomputed/k-means-fancy-light
10 iwanthue/precomputed/force-vector-fancy-light
11 iwanthue/precomputed/k-means-fancy-dark
```

```
12 iwanthue/precomputed/force-vector-fancy-dark
13 iwanthue/precomputed/k-means-shades
14 iwanthue/precomputed/force-vector-shades
15 iwanthue/precomputed/k-means-tarnish
16 iwanthue/precomputed/force-vector-tarnish
17 iwanthue/precomputed/k-means-pastel
18 iwanthue/precomputed/force-vector-pastel
19 iwanthue/precomputed/k-means-pimp
20 iwanthue/precomputed/force-vector-pimp
21 iwanthue/precomputed/k-means-intense
22 iwanthue/precomputed/force-vector-intense
23 iwanthue/precomputed/k-means-fluo
24 iwanthue/precomputed/force-vector-fluo
25 iwanthue/precomputed/k-means-red-roses
26 iwanthue/precomputed/force-vector-red-roses
27 iwanthue/precomputed/k-means-ochre-sand
28 iwanthue/precomputed/force-vector-ochre-sand
29 iwanthue/precomputed/k-means-yellow-lime
30 iwanthue/precomputed/force-vector-yellow-lime
31 iwanthue/precomputed/k-means-green-mint
32 iwanthue/precomputed/force-vector-green-mint
33 iwanthue/precomputed/k-means-ice-cube
34 iwanthue/precomputed/force-vector-ice-cube
35 iwanthue/precomputed/k-means-blue-ocean
36 iwanthue/precomputed/force-vector-blue-ocean
37 iwanthue/precomputed/k-means-indigo-night
38 iwanthue/precomputed/force-vector-indigo-night
39 iwanthue/precomputed/k-means-purple-wine
40 iwanthue/precomputed/force-vector-purple-wine
```

More info

- The tool itself is available online
- Its source code is available on GitHub
- If you have issues or requests, tell us about them
- You will find more tools on Médialab Tools