

[Skip to content](#)

[Sign up](#) [Sign in](#)

- [Explore](#)
- [Features](#)
- [Enterprise](#)
- [Blog](#)

- [Watch](#)
- [Star](#)
- [Fork](#)

[mbostock/d3](#)

- [Code](#)
- [Issues](#)
- [Pull requests](#)
- [Wiki](#)
- [Pulse](#)
- [Graphs](#)

Tutorials

Daekwon Kim edited this page Aug 3, 2015 · 137 revisions

Pages [74](#)

- [Home](#)
- [3.0](#)
- [3.1](#)
- [API 中文手册](#)
- [API Reference](#)
- [API Reference \(\)](#)
- [Api 参考](#)
- [Arrays](#)
- [Behaviors](#)
- [Bundle Layout](#)
- [Chord Layout](#)
- [Cluster Layout](#)
- [CN Home](#)
- [Colors](#)
- [Core](#)
- [Show 59 more pages...](#)

Clone this wiki locally

Wiki • Tutorials

Please feel free to add links to your work!

Introductions & Core Concepts

- [Introduction](#)
- [Let’s Make a Bar Chart, Parts I,](#)

II & III

- [Three Little Circles](#)
- [Thinking with Joins](#)
- [How Selections Work](#)
- [How Selections Work\(Korean\)](#)
- [General Update Pattern, Parts I, II & III](#)
- [Nested Selections](#)
- [Object Constancy](#)
- [Working with Transitions](#)
- [D3 Tutorials - Scott Murray](#)
- [Create A Bar Chart With D3 JavaScript - Vegibit](#)
- [Try D3 Now - Christophe Viau](#)
- [Getting Started with D3 Graphs - Patrick Mulder](#)
- [D3 for Mere Mortals - Luke Franci](#)
- [D3, Conceptually - Mikey Levine](#)
- [Introduction to D3 - Justin Palmer](#)
- [A tiny introduction to d3.js with Moon Phase Visualizer - palerdot](#)
- [D3.js experiments in the console - Sarah Allen](#)
- [Creating Basic Charts using d3.js - Ben Lorica](#)
- [Get dirty with data using d3.js - Clinton Montague](#)
- [For Protovis Users](#)
- [Manipulating data like a boss with d3 - Jerome Cukier](#)
- [Creating Animations and Transitions With D3 - Jerome Cukier](#)
- [Introduction to D3 and more tutorials - Andrew Davis](#)
- [d3 O'Clock: Building a Virtual Analog Clock with d3.js \(Part I\) - Eric S. Bullington](#)

- [How to Make an Interactive Network Visualization](#) - Jim Vallandingham
- [Learn how to make Data Visualizations with D3.js](#) - Dashingd3js
- [Introduction to d3.js and data-driven visualizations](#) - Kenny Peng
- <http://nowherenearithaca.blogspot.com/2012/06/annotating-d3-example-with-docco.html> - Brad Flyon
- [D3.js Tips and Tricks Blog](#) and pdf book version and read full text online - D3noob
- [Introduction to D3.js Geo](#) - Graham Jenson
- [learning D3.js\(2\)\(Chinese\)](#) - jtyjty99999
- [learning D3.js\(1\)\(Chinese\)](#) - jtyjty99999
- [First glance on D3.js \(codecademy course\)](#) - Jiecheng
- [Creating Interactive Charts with D3.js](#) - Anthony Ilukwe
- [How to handle dynamic JSON Data \(enter/exit\)](#) - Pier-Olivier Thibault
- [Understanding Selections](#) - Peter Cook
- [Visualize with d3js](#) - Pance Cavkovski
- [Drawing a many-to-many relationship with a simple blog-post-tag example](#) - Vijay Chakravarthy
- [Introduction to D3, with applications to big data](#) - Sam Selikoff

- [D3 Dynamic Tables with Nested Data](#) - Lee Mendelowitz
- [Getting started with D3.js](#) - Eyal Arubas
- [Visualizing Data with D3.js](#) - Tutorials for [Data Visualization Course](#) at University of Washington (compiled by Kanit "Ham" Wongsuphasawat)
- [On D3 Components](#) by Pedram Emrouznejad
- [Notes on my D3 Visualizations Development Workflow](#) by Patrick Altman
- [Make a Force-Directed China Map with D3.js \(Chinese\)](#) by Mantouhuahua
- [The Force-Directed Relationship Diagram with D3.js \(Chinese\)](#) by Mantouhuahua
- [Introduction to D3.js\(Traditional Chinese\)](#) by infographics.tw

Specific Techniques

- [Path and Transform Transitions](#)
- [Let's Make a Map](#)
- [Towards Reusable Charts](#)
- [Using Inkscape with d3](#) - Christophe Viau
- [Pie Chart Updating with Text](#) - Stephen Boak
- [How to Make Choropleth Maps in D3](#) - EJ Fox
- [Converting dynamic SVG to PNG with node.js, d3 and Imagemagick](#) - Wealthfront
- [Creating Animated Bubble Charts in D3](#) - Jim Vallandingham
- [Multiple area charts with d3.js](#)

- [Creating a Polar Area Diagram](#)
- Kristopher Reese
- [Smooth Transitioning of Polar Area Diagrams](#) - Kristopher Reese
- [Building a lightweight, flexible d3.js dashboard \(3-part series\)](#)
- Eric Seufert
- [Integrating D3 with a CouchDB database](#) - Reinhard Engel and Simon Metson
- [An interactive explanation of quadtrees](#) - Jim Kang
- [An A to Z of extra features for the d3 force layout](#) - Simon Raper
- [Stream data to create realtime, live-updating D3.js charts](#) - Ian Jennings
- [Graphing memory usage in realtime using D3.js and Rickshaw](#) - Ian Jennings

Blogs

- [Mike Bostock](#)
- [Jan Willem Tulp](#)
- [Jérôme Cukier](#)
- [Jim Vallandingham](#)
- [Bharat Bhole](#)
- [OUR D3.JS\(Chinese\)](#)
- [Peter Cook](#)
- [PubNub](#)
- [infographics.tw](#)

Books

- [Getting Started with D3](#)
Mike Dewar, O'Reilly Media,

June 2012

- [Interactive Data Visualization for the Web](#)
Scott Murray, O'Reilly Media, November 2012
- [Data Visualization with d3.js](#)
Swizec Teller, Packt Publishing, October 2013
- [Data Visualization with D3.js Cookbook](#)
Nick Qi Zhu, Packt Publishing, October 2013
- [Mastering D3.js](#)
Pablo Navarro Castillo, Packt Publishing, August 2014
- [D3.js in Action](#)
Elijah Meeks, Manning Publications, 2014
- [Learning D3.js Mapping](#)
Thomas Newton, Oscar Villarreal, Packt Publishing, 2014
- [Visual Storytelling with D3](#)
Ritchie King, Addison-Wesley, 2014
- [D3 on AngularJS](#)
Ari Lerner + Victor Powell, Leanpub, 2014
- [Data Visualization with d3.js Cookbook](#)

Courses

- [Data Visualization and D3.js](#)
Jonathan Dinu + Ryan Orban, Udacity, 2014
- [Data Visualization and Infographics with D3.js](#)
Alberto Cairo + Scott Murray, Knight Center, 2015

Talks and Videos

- [Introduction to D3](#)
Curran Kelleher, Bay Area D3 Meetup, April 2015
- [Free tagtree screencast - thinking with joins](#)
August 2014
- [For Example \(Write-up\)](#)
Eyeo Festival, June 2013.
- [Visualizing Data with Web Standards \(Slides\)](#)
W3Conf, November 2011.
- [SVG Open Keynote \(Slides\)](#)
Microsoft Research, October 2011.
- [Use the Force! \(Slides\)](#)
Trulia, September 2011.
- [D3 workshop \(Slides\)](#)
VIZBI, March 2012.
- [Intro to d3](#)
JavaScript User Group Munich, March 2012
- [Simple D3.js Bar Chart Webcast](#)
Ian Johnson.
- [Using Selections in D3 to Make Data-Driven Visualizations](#)
Ian Johnson.
- [Visual.ly Meetup Recap: Introductory D3 Workshop](#)
Aleksandra Todorova, Visual.ly February 16 2011.
- [First steps in data visualisation using d3.js](#)
Mike Dewar, New York Open Statistical Programming Meetup January 12 2012
- [Data Visualization Using D3.js](#)
Jim McCusker, TWed talk February 2012
- [An introduction to d3.js video](#)

[with synced visualisation](#)

Philip Roberts at TechMeetup
Edinburgh, November 2012

- [Slides and live code from the GAFFTA d3 intro workshop](#)
Ian Johnson 2012
- [Data Visualization with D3.js, slides and video](#)
Ben Clinkinbeard, NCDevCon 2012
- [Design process of The Electoral Map](#)
Shan Carter, Big Data Think Tank December 2012
- [D3.js - Data Visualisation in the Browser](#)
Peter Cook, Async Brighton, January 2013
- [An Intro to D3.js - Data-Driven Delight](#)
Anna Powell-Smith, Front-End London, January 2013
- [Building apps with D3.js](#)
Nathan Vander Wilt, CascadiaJS, November 2013
- [Data visualization for the web with D3.js \(English slides\)](#)
[Visualisation de données pour le web avec D3.js \(French video\) \(French slides\)](#)
Pablo Tamarit, Soft-Shake conference, October 2013
- [Ember and D3: Building a simple dashboard](#)
Sam Selikoff, Boston Ember Meetup, August 2013
- [D3 Layouts](#)
Peter Cook, Async Brighton, January 2015
- [Building Interactive Data Visualizations](#)
Jonathan Dinu, Strata San

Jose, February 2015

Meetups

- [Bay Area d3 User Group](#)
- [NYC D3.js](#)
- [London d3.js User Group](#)
- [Boston d3.js User Group](#)
- [Berlin Visualization Group](#)
- [Belo Horizonte d3.js User Group](#)
- [Twin Cities D3.js Meetup Group](#)
- [Boulder/Denver D3.js and Data Visualization](#)
- [Austin d3.js Meetup](#)
- [Auckland d3.js](#)
- [Bangalore d3.js User Group](#)

Research Papers

- [D3: Data-Driven Documents](#)
Michael Bostock, Vadim Ogievetsky, Jeffrey Heer
IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis), 2011

-
- [Status](#)
 - [API](#)
 - [Training](#)
 - [Shop](#)
 - [Blog](#)
 - [About](#)
 - [Help](#)
 - © 2015 GitHub, Inc.
 - [Terms](#)

- [Privacy](#)
- [Security](#)
- [Contact](#)

Something went wrong with that request. Please try again.

[Skip to content](#)





[Sign up](#) [Sign in](#)



This repository

- [Explore](#)
- [Features](#)
- [Enterprise](#)
- [Blog](#)

- [Watch 1,714](#)
- [Star 40,639](#)
- [Fork 10,588](#)

mbostock/d3 

- [Code](#) 
- [Issues](#) 
- [Pull requests](#) 
- [Wiki](#) 

- [Pulse](#) 
- [Graphs](#) 

HTTPS clone URL

Subversion checkout URL

You can clone with

HTTPS

or

Subversion

.

[Download ZIP](#)

API Reference

张天旭 edited this page Apr 26, 2015 · [212 revisions](#)

Pages 75

- [Home](#)
- [3.0](#)
- [3.1](#)
- [API 中文手册](#)
- [API Reference](#)
- [API Reference \(_____\)](#)
- [Api 参考](#)
- [Arrays](#)
- [Behaviors](#)
- [Bundle Layout](#)
- [Chord Layout](#)
- [Cluster Layout](#)
- [CN Home](#)
- [Colors](#)
- [Core](#)
- [CSV](#)
- [Drag Behavior](#)
- [Force Layout](#)
- [Formatting](#)
- [Gallery](#)
- [Geo](#)
- [Geo Paths](#)
- [Geo Projections](#)
- [Geo Streams](#)

- [Geometry](#)
- [Hierarchy Layout](#)
- [Histogram Layout](#)
- [Hull Geom](#)
- [ID Home](#)
- [Internals](#)
- [JP 3.0](#)
- [JP API Reference](#)
- [JP Home](#)
- [JP Tutorials](#)
- [Layouts](#)
- [Localization](#)
- [Math](#)
- [Namespaces](#)
- [Ordinal Scales](#)
- [Pack Layout](#)
- [Partition Layout](#)
- [Pie Layout](#)
- [Plugins](#)
- [Polygon Geom](#)
- [Quadtree Geom](#)
- [Quantitative Scales](#)
- [Release Notes](#)
- [Requests](#)
- [Scales](#)
- [Selections](#)
- [Stack Layout](#)
- [SVG](#)
- [SVG Axes](#)
- [SVG Controls](#)
- [SVG Events](#)
- [SVG Shapes](#)
- [Time](#)
- [Time Formatting](#)
- [Time Intervals](#)
- [Time Scales](#)
- [Transitions](#)
- [Tree Layout](#)

- [Treemap Layout](#)
- [Tutorials](#)
- [TW Home](#)
- [Upgrading to 3.0](#)
- [Voronoi Geom](#)
- [Zoom Behavior](#)
- [_____](#)
- [数学](#)
- [数组](#)
- [请求](#)
- [过渡](#)
- [选择器](#)
- [选择集](#)
- [Show 60 more pages...](#)

Clone this wiki locally

[Wiki](#) • **API Reference**

Everything in D3 is scoped under the `d3` namespace.

D3 uses [semantic versioning](#). You can find the current version of D3 as `d3.version`.

See one of:

- [Behaviors](#) - reusable interaction behaviors
- [Core](#) - selections, transitions, data, localization, colors, etc.
- [Geography](#) - project spherical coordinates, latitude & longitude math
- [Geometry](#) - utilities for 2D geometry, such as Voronoi diagrams and quadtrees
- [Layouts](#) - derive secondary data for positioning elements
- [Scales](#) - convert between data and visual encodings
- [SVG](#) - utilities for creating Scalable Vector Graphics
- [Time](#) - parse or format times, compute calendar intervals, etc.

[d3 \(core\)](#)

Selections

- [d3.event](#) - access the current user event for interaction.
- [d3.mouse](#) - gets the mouse position relative to a specified container.
- [d3.select](#) - select an element from the current document.
- [d3.selectAll](#) - select multiple elements from the current document.
- [d3.selection](#) - augment the selection prototype, or test instance types.
- [d3.touch](#) - gets a touch position relative to a specified container.
- [d3.touches](#) - gets the touch positions relative to a specified container.
- [selection.append](#) - create and append new elements.
- [selection.attr](#) - get or set attribute values.
- [selection.call](#) - call a function passing in the current selection.
- [selection.classed](#) - add or remove CSS classes.
- [selection.data](#) - get or set data for a group of elements, while computing a relational join.
- [selection.datum](#) - get or set data for individual elements, without computing a join.
- [selection.each](#) - call a function for each selected element.
- [selection.empty](#) - returns true if the selection is empty.
- [selection.enter](#) - returns placeholders for missing elements.
- [selection.exit](#) - returns elements that are no longer needed.
- [selection.filter](#) - filter a selection based on data.
- [selection.html](#) - get or set inner HTML content.
- [selection.insert](#) - create and insert new elements before existing elements.
- [selection.interrupt](#) - immediately interrupt the current transition, if any.
- [selection.node](#) - returns the first node in the selection.
- [selection.on](#) - add or remove event listeners for interaction.
- [selection.order](#) - reorders elements in the document to match the selection.
- [selection.property](#) - get or set raw properties.
- [selection.remove](#) - remove elements from the document.
- [selection.select](#) - subselect a descendant element for each selected element.
- [selection.selectAll](#) - subselect multiple descendants for each selected element.
- [selection.size](#) - returns the number of elements in the selection.
- [selection.sort](#) - sort elements in the document based on data.
- [selection.style](#) - get or set style properties.
- [selection.text](#) - get or set text content.
- [selection.transition](#) - start a transition on the selected elements.

Transitions

- [d3.ease](#) - customize transition timing.
- [d3.timer](#) - start a custom animation timer.
- [d3.interpolate](#) - interpolate two values.
- [d3.interpolateArray](#) - interpolate two arrays of values.
- [d3.interpolateHcl](#) - interpolate two HCL colors.
- [d3.interpolateHsl](#) - interpolate two HSL colors.
- [d3.interpolateLab](#) - interpolate two L*a*b* colors.
- [d3.interpolateNumber](#) - interpolate two numbers.
- [d3.interpolateObject](#) - interpolate two arbitrary objects.
- [d3.interpolateRgb](#) - interpolate two RGB colors.
- [d3.interpolateRound](#) - interpolate two integers.
- [d3.interpolateString](#) - interpolate two strings.
- [d3.interpolateTransform](#) - interpolate two 2D matrix transforms.
- [d3.interpolateZoom](#) - zoom and pan between two points smoothly.
- [d3.interpolators](#) - register a custom interpolator.
- [d3.timer.flush](#) - immediately execute any zero-delay timers.
- [d3.transition](#) - start an animated transition.
- [ease](#) - a parametric easing function.
- [interpolate](#) - a parametric interpolation function.
- [transition.attr](#) - smoothly transition to the new attribute value.
- [transition.attrTween](#) - smoothly transition between two attribute values
- [transition.call](#) - call a function passing in the current transition.
- [transition.delay](#) - specify per-element delay in milliseconds.
- [transition.duration](#) - specify per-element duration in milliseconds.
- [transition.each](#) - add a listener for transition end events.
- [transition.ease](#) - specify transition easing function.
- [transition.empty](#) - returns true if the transition is empty.
- [transition.filter](#) - filter a transition based on data.
- [transition.node](#) - returns the first node in the transition.
- [transition.remove](#) - remove selected elements at the end of a transition.
- [transition.select](#) - start a transition on a descendant element for each selected element.
- [transition.selectAll](#) - start a transition on multiple descendants for each selected element.
- [transition.size](#) - returns the number of elements in the selection.
- [transition.style](#) - smoothly transition to the new style property value.
- [transition.styleTween](#) - smoothly transition between two style property values.
- [transition.text](#) - set the text content when the transition starts.
- [transition.transition](#) - when this transition ends, start another one on the same elements.
- [transition.tween](#) - specify a custom tween operator to run as part of the transition.

Working with Arrays

- [d3.ascending](#) - compare two values for sorting.
- [d3.bisectLeft](#) - search for a value in a sorted array.
- [d3.bisector](#) - bisect using an accessor or comparator.
- [d3.bisectRight](#) - search for a value in a sorted array.
- [d3.bisect](#) - search for a value in a sorted array.
- [d3.descending](#) - compare two values for sorting.
- [d3.deviation](#) - compute the standard deviation of an array of numbers.
- [d3.entries](#) - list the key-value entries of an associative array.
- [d3.extent](#) - find the minimum and maximum value in an array.
- [d3.keys](#) - list the keys of an associative array.
- [d3.map](#) - a shim for ES6 maps, since objects are not hashes!
- [d3.max](#) - find the maximum value in an array.
- [d3.mean](#) - compute the arithmetic mean of an array of numbers.
- [d3.median](#) - compute the median of an array of numbers (the 0.5-quantile).
- [d3.merge](#) - merge multiple arrays into one array.
- [d3.min](#) - find the minimum value in an array.
- [d3.nest](#) - group array elements hierarchically.
- [d3.pairs](#) - returns an array of adjacent pairs of elements.
- [d3.permute](#) - reorder an array of elements according to an array of indexes.
- [d3.quantile](#) - compute a quantile for a sorted array of numbers.
- [d3.range](#) - generate a range of numeric values.
- [d3.set](#) - a shim for ES6 sets, since objects are not hashes!
- [d3.shuffle](#) - randomize the order of an array.
- [d3.sum](#) - compute the sum of an array of numbers.
- [d3.transpose](#) - transpose an array of arrays.
- [d3.values](#) - list the values of an associated array.
- [d3.variance](#) - compute the variance of an array of numbers.
- [d3.zip](#) - transpose a variable number of arrays.
- [map.empty](#) - returns false if the map has at least one entry.
- [map.entries](#) - returns the map's array of entries (key-values objects).
- [map.forEach](#) - calls the specified function for each entry in the map.
- [map.get](#) - returns the value for the specified key.
- [map.has](#) - returns true if the map contains the specified key.
- [map.keys](#) - returns the map's array of keys.
- [map.remove](#) - removes the entry for specified key.
- [map.set](#) - sets the value for the specified key.

- [map.size](#) - returns the number of entries in the map.
- [map.values](#) - returns the map's array of values.
- [nest.entries](#) - evaluate the nest operator, returning an array of key-values tuples.
- [nest.key](#) - add a level to the nest hierarchy.
- [nest.map](#) - evaluate the nest operator, returning an associative array.
- [nest.rollup](#) - specify a rollup function for leaf values.
- [nest.sortKeys](#) - sort the current nest level by key.
- [nest.sortValues](#) - sort the leaf nest level by value.
- [set.add](#) - adds the specified value.
- [set.empty](#) - returns true if the set has at least one value.
- [set.forEach](#) - calls the specified function for each value in the set.
- [set.has](#) - returns true if the set contains the specified value.
- [set.remove](#) - removes the specified value.
- [set.size](#) - returns the number of values in the set.
- [set.values](#) - returns the set's array of values.

Math

- [d3.random.bates](#) - generate a random number with a Bates distribution.
- [d3.random.irwinHall](#) - generate a random number with an Irwin–Hall distribution.
- [d3.random.logNormal](#) - generate a random number with a log-normal distribution.
- [d3.random.normal](#) - generate a random number with a normal distribution.
- [d3.transform](#) - compute the standard form of a 2D matrix transform.

Loading External Resources

- [d3.csv](#) - request a comma-separated values (CSV) file.
- [d3.html](#) - request an HTML document fragment.
- [d3.json](#) - request a JSON blob.
- [d3.text](#) - request a text file.
- [d3.tsv](#) - request a tab-separated values (TSV) file.
- [d3.xhr](#) - request a resource using XMLHttpRequest.
- [d3.xml](#) - request an XML document fragment.
- [xhr.abort](#) - abort an outstanding request.
- [xhr.get](#) - issue a GET request.
- [xhr.header](#) - set a request header.
- [xhr.mimeType](#) - set the Accept request header and override the response MIME type.
- [xhr.on](#) - add an event listener for "progress", "load" or "error" events.

- [xhr.post](#) - issue a POST request.
- [xhr.response](#) - set a response mapping function.
- [xhr.send](#) - issue a request with the specified method and data.

String Formatting

- [d3.format](#) - format a number as a string.
- [d3.formatPrefix](#) - returns the [SI prefix](#) for the specified value and precision.
- [d3.requote](#) - quote a string for use in a regular expression.
- [d3.round](#) - rounds a value to some digits after the decimal point.

CSV Formatting (d3.csv)

- [d3.csv.formatRows](#) - format an array of tuples into a CSV string.
- [d3.csv.format](#) - format an array of objects into a CSV string.
- [d3.csv.parseRows](#) - parse a CSV string into tuples, ignoring the header row.
- [d3.csv.parse](#) - parse a CSV string into objects using the header row.
- [d3.csv](#) - request a comma-separated values (CSV) file.
- [d3.dsv](#) - create a parser/formatter for the specified delimiter and mime type.
- [d3.tsv.formatRows](#) - format an array of tuples into a TSV string.
- [d3.tsv.format](#) - format an array of objects into a TSV string.
- [d3.tsv.parseRows](#) - parse a TSV string into tuples, ignoring the header row.
- [d3.tsv.parse](#) - parse a TSV string into objects using the header row.
- [d3.tsv](#) - request a tab-separated values (TSV) file.

Localization

- [d3.locale](#) - create a new locale using the specified strings.
- [locale.numberFormat](#) - create a new number formatter.
- [locale.timeFormat](#) - create a new time formatter / parser.

Colors

- [d3.hcl](#) - specify a color in HCL space.
- [d3.hsl](#) - specify a color in HSL space.
- [d3.lab](#) - specify a color in L*a*b* space.
- [d3.rgb](#) - specify a color in RGB space.
- [hcl.brighter](#) - increase lightness by some exponential factor (gamma).

- [hcl.darker](#) - decrease lightness by some exponential factor (gamma).
- [hcl.rgb](#) - convert from HCL to RGB.
- [hcl.toString](#) - convert an HCL color to a string.
- [hsl.brighter](#) - increase lightness by some exponential factor (gamma).
- [hsl.darker](#) - decrease lightness by some exponential factor (gamma).
- [hsl.rgb](#) - convert from HSL to RGB.
- [hsl.toString](#) - convert an HSL color to a string.
- [lab.brighter](#) - increase lightness by some exponential factor (gamma).
- [lab.darker](#) - decrease lightness by some exponential factor (gamma).
- [lab.rgb](#) - convert from L*a*b* to RGB.
- [lab.toString](#) - convert a L*a*b* color to a string.
- [rgb.brighter](#) - increase RGB channels by some exponential factor (gamma).
- [rgb.darker](#) - decrease RGB channels by some exponential factor (gamma).
- [rgb.hsl](#) - convert from RGB to HSL.
- [rgb.toString](#) - convert an RGB color to a string.

Namespaces

- [d3.ns.prefix](#) - access or extend known XML namespaces.
- [d3.ns.qualify](#) - qualify a prefixed name, such as "xlink:href".

Internals

- [d3.dispatch](#) - create a custom event dispatcher.
- [d3.functor](#) - create a function that returns a constant.
- [d3.rebind](#) - rebind an inherited getter/setter method to a subclass.
- [dispatch.on](#) - register or unregister an event listener.
- [dispatch.type](#) - dispatch an event to registered listeners.

d3.scale (Scales)

Quantitative

- [d3.scale.identity](#) - construct a linear identity scale.
- [d3.scale.linear](#) - construct a linear quantitative scale.
- [d3.scale.log](#) - construct a quantitative scale with an logarithmic transform.
- [d3.scale.pow](#) - construct a quantitative scale with an exponential transform.
- [d3.scale.quantile](#) - construct a quantitative scale mapping to quantiles.

- [d3.scale.quantize](#) - construct a linear quantitative scale with a discrete output range.
- [d3.scale.sqrt](#) - construct a quantitative scale with a square root transform.
- [d3.scale.threshold](#) - construct a threshold scale with a discrete output range.
- [identity.copy](#) - create a new scale from an existing scale.
- [identity.domain](#) - get or set the scale's domain and range.
- [identity.invert](#) - equivalent to identity; the identity function.
- [identity.range](#) - equivalent to identity.domain.
- [identity.tickFormat](#) - get a formatter for displaying tick values.
- [identity.ticks](#) - get representative values from the domain.
- [identity](#) - the identity function.
- [linear.clamp](#) - enable or disable clamping of the output range.
- [linear.copy](#) - create a new scale from an existing scale.
- [linear.domain](#) - get or set the scale's input domain.
- [linear.interpolate](#) - get or set the scale's output interpolator.
- [linear.invert](#) - get the domain value corresponding to a given range value.
- [linear.nice](#) - extend the scale domain to nice round numbers.
- [linear.rangeRound](#) - set the scale's output range, and enable rounding.
- [linear.range](#) - get or set the scale's output range.
- [linear.tickFormat](#) - get a formatter for displaying tick values.
- [linear.ticks](#) - get representative values from the input domain.
- [linear](#) - get the range value corresponding to a given domain value.
- [log.clamp](#) - enable or disable clamping of the output range.
- [log.copy](#) - create a new scale from an existing scale.
- [log.domain](#) - get or set the scale's input domain.
- [log.interpolate](#) - get or set the scale's output interpolator.
- [log.invert](#) - get the domain value corresponding to a given range value.
- [log.nice](#) - extend the scale domain to nice powers of ten.
- [log.rangeRound](#) - set the scale's output range, and enable rounding.
- [log.range](#) - get or set the scale's output range.
- [log.tickFormat](#) - get a formatter for displaying tick values.
- [log.ticks](#) - get representative values from the input domain.
- [log](#) - get the range value corresponding to a given domain value.
- [pow.clamp](#) - enable or disable clamping of the output range.
- [pow.copy](#) - create a new scale from an existing scale.
- [pow.domain](#) - get or set the scale's input domain.
- [pow.exponent](#) - get or set the exponent power.
- [pow.interpolate](#) - get or set the scale's output interpolator.
- [pow.invert](#) - get the domain value corresponding to a given range value.

- [pow.nice](#) - extend the scale domain to nice round numbers.
- [pow.rangeRound](#) - set the scale's output range, and enable rounding.
- [pow.range](#) - get or set the scale's output range.
- [pow.tickFormat](#) - get a formatter for displaying tick values.
- [pow.ticks](#) - get representative values from the input domain.
- [pow](#) - get the range value corresponding to a given domain value.
- [quantile.copy](#) - create a new scale from an existing scale.
- [quantile.domain](#) - get or set the scale's input domain (as discrete values).
- [quantile.invertExtent](#) - get the domain values for the specified range value.
- [quantile.quantiles](#) - get the scale's quantile bin thresholds.
- [quantile.range](#) - get or set the scale's output range (as discrete values).
- [quantile](#) - get the range value corresponding to a given domain value.
- [quantize.copy](#) - create a new scale from an existing scale.
- [quantize.domain](#) - get or set the scale's input domain.
- [quantize.invertExtent](#) - get the domain values for the specified range value.
- [quantize.range](#) - get or set the scale's output range (as discrete values).
- [quantize](#) - get the range value corresponding to a given domain value.
- [threshold.copy](#) - create a new scale from an existing scale.
- [threshold.domain](#) - get or set the scale's input domain.
- [threshold.invertExtent](#) - get the domain values for the specified range value.
- [threshold.range](#) - get or set the scale's output range (as discrete values).
- [threshold](#) - get the range value corresponding to a given domain value.

Ordinal

- [d3.scale.category10](#) - construct an ordinal scale with ten categorical colors.
- [d3.scale.category20b](#) - construct an ordinal scale with twenty categorical colors.
- [d3.scale.category20c](#) - construct an ordinal scale with twenty categorical colors.
- [d3.scale.category20](#) - construct an ordinal scale with twenty categorical colors.
- [d3.scale.ordinal](#) - construct an ordinal scale.
- [ordinal.copy](#) - create a new scale from an existing scale.
- [ordinal.domain](#) - get or set the scale's input domain.
- [ordinal.rangeBands](#) - divide a continuous output range for discrete bands.
- [ordinal.rangeBand](#) - get the discrete range band width.
- [ordinal.rangeExtent](#) - get the minimum and maximum values of the output range.
- [ordinal.rangePoints](#) - divide a continuous output range for discrete points.
- [ordinal.rangeRoundBands](#) - divide a continuous output range for discrete bands.
- [ordinal.rangeRoundPoints](#) - divide a continuous output range for discrete points.

- [ordinal.range](#) - get or set the scale's output range.
- [ordinal](#) - get the range value corresponding to a given domain value.

d3.svg (SVG)

Shapes

- [arc.centroid](#) - compute the arc centroid.
- [arc.cornerRadius](#) - get or set the corner radius accessor.
- [arc.endAngle](#) - get or set the end angle accessor.
- [arc.innerRadius](#) - get or set the inner radius accessor.
- [arc.outerRadius](#) - get or set the outer radius accessor.
- [arc.padAngle](#) - get or set the pad angle accessor.
- [arc.padRadius](#) - get or set the pad radius accessor.
- [arc.startAngle](#) - get or set the start angle accessor.
- [arc](#) - generate a solid arc, as in a pie or donut chart.
- [area.angle](#) - get or set the *angle* accessors.
- [area.defined](#) - control whether the area is defined at a given point.
- [area.defined](#) - control whether the area is defined at a given point.
- [area.endAngle](#) - get or set the *angle* (topline) accessor.
- [area.innerRadius](#) - get or set the inner *radius* (baseline) accessor.
- [area.interpolate](#) - get or set the interpolation mode.
- [area.outerRadius](#) - get or set the outer *radius* (topline) accessor.
- [area.radius](#) - get or set the *radius* accessors.
- [area.startAngle](#) - get or set the *angle* (baseline) accessor.
- [area.tension](#) - get or set the cardinal spline tension.
- [area.x0](#) - get or set the *x0*-coordinate (baseline) accessor.
- [area.x1](#) - get or set the *x1*-coordinate (topline) accessor.
- [area.x](#) - get or set the *x*-coordinate accessors.
- [area.y0](#) - get or set the *y0*-coordinate (baseline) accessor.
- [area.y1](#) - get or set the *y1*-coordinate (topline) accessor.
- [area.y](#) - get or set the *y*-coordinate accessors.
- [area](#) - generate a piecewise linear area, as in an area chart.
- [area](#) - generate a piecewise linear area, as in a polar area chart.
- [chord.endAngle](#) - get or set the arc end angle accessor.
- [chord.radius](#) - get or set the arc radius accessor.
- [chord.source](#) - get or set the source arc accessor.
- [chord.startAngle](#) - get or set the arc start angle accessor.

- [chord.target](#) - get or set the target arc accessor.
- [chord](#) - generate a quadratic Bézier connecting two arcs, as in a chord diagram.
- [d3.svg.arc](#) - create a new arc generator.
- [d3.svg.area.radial](#) - create a new area generator.
- [d3.svg.area](#) - create a new area generator.
- [d3.svg.chord](#) - create a new chord generator.
- [d3.svg.diagonal.radial](#) - create a new diagonal generator.
- [d3.svg.diagonal](#) - create a new diagonal generator.
- [d3.svg.line.radial](#) - create a new radial line generator.
- [d3.svg.line](#) - create a new line generator.
- [d3.svg.symbolTypes](#) - the array of supported symbol types.
- [d3.svg.symbol](#) - create a new symbol generator.
- [diagonal.projection](#) - get or set an optional point transform.
- [diagonal.source](#) - get or set the source point accessor.
- [diagonal.target](#) - get or set the target point accessor.
- [diagonal](#) - generate a two-dimensional Bézier connector, as in a node-link diagram.
- [diagonal](#) - generate a two-dimensional Bézier connector, as in a node-link diagram.
- [line.angle](#) - get or set the *angle* accessor.
- [line.defined](#) - control whether the line is defined at a given point.
- [line.defined](#) - control whether the line is defined at a given point.
- [line.interpolate](#) - get or set the interpolation mode.
- [line.interpolate](#) - get or set the interpolation mode.
- [line.radius](#) - get or set the *radius* accessor.
- [line.tension](#) - get or set the cardinal spline tension.
- [line.tension](#) - get or set the cardinal spline tension.
- [line.x](#) - get or set the *x*-coordinate accessor.
- [line.y](#) - get or set the *y*-coordinate accessor.
- [line](#) - generate a piecewise linear curve, as in a line chart.
- [line](#) - generate a piecewise linear curve, as in a polar line chart.
- [symbol.size](#) - get or set the symbol size (in square pixels) accessor.
- [symbol.type](#) - get or set the symbol type accessor.
- [symbol](#) - generate categorical symbols, as in a scatterplot.

Axes

- [axis.innerTickSize](#) - specify the size of inner ticks.
- [axis.orient](#) - get or set the axis orientation.
- [axis.outerTickSize](#) - specify the size of outer ticks.

- [axis.scale](#) - get or set the axis scale.
- [axis.tickFormat](#) - override the tick formatting for labels.
- [axis.tickPadding](#) - specify padding between ticks and tick labels.
- [axis.tickSize](#) - specify the size of major, minor and end ticks.
- [axis.ticks](#) - control how ticks are generated for the axis.
- [axis.tickValues](#) - specify tick values explicitly.
- [axis](#) - creates or updates an axis for the given selection or transition.
- [d3.svg.axis](#) - create a new axis generator.

Controls

- [brush.clear](#) - reset the brush extent.
- [brush.empty](#) - whether or not the brush extent is empty.
- [brush.event](#) - dispatch brush events after setting the extent.
- [brush.extent](#) - the brush's extent in zero, one or two dimensions.
- [brush.on](#) - listeners for when the brush is moved.
- [brush.x](#) - the brush's x -scale, for horizontal brushing.
- [brush.y](#) - the brush's y -scale, for vertical brushing.
- [brush](#) - apply a brush to the given selection or transition.
- [d3.svg.brush](#) - click and drag to select one- or two-dimensional regions.

d3.time (Time)

Time Formatting

- [d3.time.format.iso](#) - the ISO 8601 UTC time formatter.
- [d3.time.format.multi](#) - create a new local multi-resolution time formatter.
- [d3.time.format.utc](#) - create a new UTC time formatter for a given specifier.
- [d3.time.format](#) - create a new local time formatter for a given specifier.
- [format.parse](#) - parse a string into a date.
- [format](#) - format a date into a string.

Time Scales

- [d3.time.scale](#) - construct a linear time scale.
- [scale.clamp](#) - enable or disable clamping of the output range.
- [scale.copy](#) - create a new scale from an existing scale.
- [scale.domain](#) - get or set the scale's input domain.

- [scale.interpolate](#) - get or set the scale's output interpolator.
- [scale.invert](#) - get the domain value corresponding to a given range value.
- [scale.nice](#) - extend the scale domain to nice round numbers.
- [scale.rangeRound](#) - set the scale's output range, and enable rounding.
- [scale.range](#) - get or set the scale's output range.
- [scale.tickFormat](#) - get a formatter for displaying tick values.
- [scale.ticks](#) - get representative values from the input domain.
- [scale](#) - get the range value corresponding to a given domain value.

Time Intervals

- [d3.time.dayOfYear](#) - computes the day number.
- [d3.time.days](#) - alias for day.range.
- [d3.time.day](#) - every day (12:00 AM).
- [d3.time.fridayOfYear](#) - computes the friday-based week number.
- [d3.time.fridays](#) - alias for friday.range.
- [d3.time.friday](#) - every Friday (e.g., February 5, 12:00 AM).
- [d3.time.hours](#) - alias for hour.range.
- [d3.time.hour](#) - every hour (e.g., 1:00 AM).
- [d3.time.interval](#) - a time interval in local time.
- [d3.time.minutes](#) - alias for minute.range.
- [d3.time.minute](#) - every minute (e.g., 1:02 AM).
- [d3.time.mondayOfYear](#) - computes the monday-based week number.
- [d3.time.mondays](#) - alias for monday.range.
- [d3.time.monday](#) - every Monday (e.g., February 5, 12:00 AM).
- [d3.time.months](#) - alias for month.range.
- [d3.time.month](#) - every month (e.g., February 1, 12:00 AM).
- [d3.time.saturdayOfYear](#) - computes the saturday-based week number.
- [d3.time.saturdays](#) - alias for saturday.range.
- [d3.time.saturday](#) - every Saturday (e.g., February 5, 12:00 AM).
- [d3.time.seconds](#) - alias for second.range.
- [d3.time.second](#) - every second (e.g., 1:02:03 AM).
- [d3.time.sundayOfYear](#) - computes the sunday-based week number.
- [d3.time.sundays](#) - alias for sunday.range.
- [d3.time.sunday](#) - every Sunday (e.g., February 5, 12:00 AM).
- [d3.time.thursdayOfYear](#) - computes the thursday-based week number.
- [d3.time.thursdays](#) - alias for thursday.range.
- [d3.time.thursday](#) - every Thursday (e.g., February 5, 12:00 AM).

- [d3.time.tuesdayOfYear](#) - computes the tuesday-based week number.
- [d3.time.tuesdays](#) - alias for tuesday.range.
- [d3.time.tuesday](#) - every Tuesday (e.g., February 5, 12:00 AM).
- [d3.time.wednesdayOfYear](#) - computes the wednesday-based week number.
- [d3.time.wednesdays](#) - alias for wednesday.range.
- [d3.time.wednesday](#) - every Wednesday (e.g., February 5, 12:00 AM).
- [d3.time.weekOfYear](#) - alias for sundayOfYear.
- [d3.time.weeks](#) - alias for sunday.range.
- [d3.time.week](#) - alias for sunday.
- [d3.time.years](#) - alias for year.range.
- [d3.time.year](#) - every year (e.g., January 1, 12:00 AM).
- [interval.ceil](#) - rounds up to the nearest interval.
- [interval.floor](#) - rounds down to the nearest interval.
- [interval.offset](#) - returns a date offset by some interval.
- [interval.range](#) - returns dates within the specified range.
- [interval.round](#) - rounds up or down to the nearest interval.
- [interval.utc](#) - returns the UTC-equivalent time interval.
- [interval](#) - alias for interval.floor.

d3.layout (Layouts)

Bundle

- [bundle](#) - apply Holten's *hierarchical bundling* algorithm to edges.
- [d3.layout.bundle](#) - construct a new default bundle layout.

Chord

- [chord.chords](#) - retrieve the computed chord angles.
- [chord.groups](#) - retrieve the computed group angles.
- [chord.matrix](#) - get or set the matrix data backing the layout.
- [chord.padding](#) - get or set the angular padding between chord segments.
- [chord.sortChords](#) - get or set the comparator function for chords (z-order).
- [chord.sortGroups](#) - get or set the comparator function for groups.
- [chord.sortSubgroups](#) - get or set the comparator function for subgroups.
- [d3.layout.chord](#) - produce a chord diagram from a matrix of relationships.

Cluster

- [cluster.children](#) - get or set the accessor function for child nodes.
- [cluster.links](#) - compute the parent-child links between tree nodes.
- [cluster.nodeSize](#) - specify a fixed size for each node.
- [cluster.nodes](#) - compute the cluster layout and return the array of nodes.
- [cluster.separation](#) - get or set the spacing function between neighboring nodes.
- [cluster.size](#) - get or set the layout size in x and y .
- [cluster.sort](#) - get or set the comparator function for sibling nodes.
- [cluster](#) - alias for cluster.nodes.
- [d3.layout.cluster](#) - cluster entities into a dendrogram.

Force

- [d3.layout.force](#) - position linked nodes using physical simulation.
- [force.alpha](#) - get or set the layout's cooling parameter.
- [force.chargeDistance](#) - get or set the maximum charge distance.
- [force.charge](#) - get or set the charge strength.
- [force.drag](#) - bind a behavior to nodes to allow interactive dragging.
- [force.friction](#) - get or set the friction coefficient.
- [force.gravity](#) - get or set the gravity strength.
- [force.linkDistance](#) - get or set the link distance.
- [force.linkStrength](#) - get or set the link strength.
- [force.links](#) - get or set the array of links between nodes.
- [force.nodes](#) - get or set the array of nodes to layout.
- [force.on](#) - listen to updates in the computed layout positions.
- [force.resume](#) - reheat the cooling parameter and restart simulation.
- [force.size](#) - get or set the layout size in x and y .
- [force.start](#) - start or restart the simulation when the nodes change.
- [force.stop](#) - immediately terminate the simulation.
- [force.theta](#) - get or set the accuracy of the charge interaction.
- [force.tick](#) - run the layout simulation one step.

Hierarchy

- [d3.layout.hierarchy](#) - derive a custom hierarchical layout implementation.
- [hierarchy.children](#) - get or set the accessor function for child nodes.
- [hierarchy.links](#) - compute the parent-child links between tree nodes.
- [hierarchy.nodes](#) - compute the layout and return the array of nodes.

- [hierarchy.revalue](#) - recompute the hierarchy values.
- [hierarchy.sort](#) - get or set the comparator function for sibling nodes.
- [hierarchy.value](#) - get or set the value accessor function.
- [hierarchy](#) - alias for `hierarchy.nodes`.

Histogram

- [d3.layout.histogram](#) - construct a new default histogram layout.
- [histogram.bins](#) - specify how values are organized into bins.
- [histogram.frequency](#) - compute the distribution as counts or probabilities.
- [histogram.range](#) - get or set the considered value range.
- [histogram.value](#) - get or set the value accessor function.
- [histogram](#) - compute the distribution of data using quantized bins.

Pack

- [d3.layout.pack](#) - produce a hierarchical layout using recursive circle-packing.
- [pack.children](#) - get or set the children accessor function.
- [pack.links](#) - compute the parent-child links between tree nodes.
- [pack.nodes](#) - compute the pack layout and return the array of nodes.
- [pack.padding](#) - specify the layout padding in (approximate) pixels.
- [pack.radius](#) - specify the node radius, rather than deriving it from value.
- [pack.size](#) - specify the layout size in x and y .
- [pack.sort](#) - control the order in which sibling nodes are traversed.
- [pack.value](#) - get or set the value accessor used to size circles.
- [pack](#) - alias for `pack.nodes`.

Partition

- [d3.layout.partition](#) - recursively partition a node tree into a sunburst or icicle.
- [partition.children](#) - get or set the children accessor function.
- [partition.links](#) - compute the parent-child links between tree nodes.
- [partition.nodes](#) - compute the partition layout and return the array of nodes.
- [partition.size](#) - specify the layout size in x and y .
- [partition.sort](#) - control the order in which sibling nodes are traversed.
- [partition.value](#) - get or set the value accessor used to size circles.
- [partition](#) - alias for `partition.nodes`.

Pie

- [d3.layout.pie](#) - construct a new default pie layout.
- [pie.endAngle](#) - get or set the overall end angle of the pie.
- [pie.padAngle](#) - get or set the pad angle of the pie.
- [pie.sort](#) - control the clockwise order of pie slices.
- [pie.startAngle](#) - get or set the overall start angle of the pie.
- [pie.value](#) - get or set the value accessor function.
- [pie](#) - compute the start and end angles for arcs in a pie or donut chart.

Stack

- [d3.layout.stack](#) - construct a new default stack layout.
- [stack.offset](#) - specify the overall baseline algorithm.
- [stack.order](#) - control the order in which series are stacked.
- [stack.out](#) - get or set the output function for storing the baseline.
- [stack.values](#) - get or set the values accessor function per series.
- [stack.x](#) - get or set the *x*-dimension accessor function.
- [stack.y](#) - get or set the *y*-dimension accessor function.
- [stack](#) - compute the baseline for each series in a stacked bar or area chart.

Tree

- [d3.layout.tree](#) - position a tree of nodes tidily.
- [tree.children](#) - get or set the children accessor function.
- [tree.links](#) - compute the parent-child links between tree nodes.
- [tree.nodeSize](#) - specify a fixed size for each node.
- [tree.nodes](#) - compute the tree layout and return the array of nodes.
- [tree.separation](#) - get or set the spacing function between neighboring nodes.
- [tree.size](#) - specify the layout size in *x* and *y*.
- [tree.sort](#) - control the order in which sibling nodes are traversed.
- [tree](#) - alias for [tree.nodes](#).

Treemap

- [d3.layout.treemap](#) - use recursive spatial subdivision to display a tree of nodes.
- [treemap.children](#) - get or set the children accessor function.
- [treemap.links](#) - compute the parent-child links between tree nodes.

- [treemap.mode](#) - change the treemap layout algorithm.
- [treemap.nodes](#) - compute the treemap layout and return the array of nodes.
- [treemap.padding](#) - specify the padding between a parent and its children.
- [treemap.round](#) - enable or disable rounding to exact pixels.
- [treemap.size](#) - specify the layout size in x and y .
- [treemap.sort](#) - control the order in which sibling nodes are traversed.
- [treemap.sticky](#) - make the layout sticky for stable updates.
- [treemap.value](#) - get or set the value accessor used to size treemap cells.
- [treemap](#) - alias for [treemap.nodes](#).

[d3.geo \(Geography\)](#)

[Paths](#)

- [circle.angle](#) - specify the angular radius in degrees.
- [circle.origin](#) - specify the origin in latitude and longitude.
- [circle.precision](#) - specify the precision of the piecewise circle.
- [circle](#) - generate a piecewise circle as a Polygon.
- [d3.geo.area](#) - compute the spherical area of a given feature.
- [d3.geo.bounds](#) - compute the latitude-longitude bounding box for a given feature.
- [d3.geo.centroid](#) - compute the spherical centroid of a given feature.
- [d3.geo.circle](#) - create a circle generator.
- [d3.geo.distance](#) - compute the great-arc distance between two points.
- [d3.geo.graticule](#) - create a graticule generator.
- [d3.geo.interpolate](#) - interpolate between two points along a great arc.
- [d3.geo.length](#) - compute the length of a line string or the perimeter of a polygon.
- [d3.geo.path](#) - create a new geographic path generator.
- [d3.geo.rotation](#) - create a rotation function for the specified angles $[\lambda, \phi, \gamma]$.
- [graticule.extent](#) - get or set the major & minor extents.
- [graticule.lines](#) - generate an array of LineStrings of meridians and parallels.
- [graticule.majorExtent](#) - get or set the major extent.
- [graticule.majorStep](#) - get or set the major step intervals.
- [graticule.minorExtent](#) - get or set the minor extent.
- [graticule.minorStep](#) - get or set the minor step intervals.
- [graticule.outline](#) - generate a Polygon of the graticule's extent.
- [graticule.precision](#) - get or set the latitudinal precision.
- [graticule.step](#) - get or set the major & minor step intervals.
- [graticule](#) - generate a MultiLineString of meridians and parallels.

- [path.area](#) - compute the projected area of a given feature.
- [path.bounds](#) - compute the projected bounds of a given feature.
- [path.centroid](#) - compute the projected centroid of a given feature.
- [path.context](#) - get or set the render context.
- [path.pointRadius](#) - get or set the radius to display point features.
- [path.projection](#) - get or set the geographic projection.
- [path](#) - project the specified feature and render it to the context.
- [rotation.invert](#) - inverse-rotate the given location around the sphere.
- [rotation](#) - rotate the given location around the sphere.

Projections

- [albers.parallels](#) - get or set the projection's two standard parallels.
- [d3.geo.albersUsa](#) - a composite Albers projection for the United States.
- [d3.geo.albers](#) - the Albers equal-area conic projection.
- [d3.geo.azimuthalEqualArea.raw](#) - the raw azimuthal equal-area projection.
- [d3.geo.azimuthalEqualArea](#) - the azimuthal equal-area projection.
- [d3.geo.azimuthalEquidistant.raw](#) - the azimuthal equidistant projection.
- [d3.geo.azimuthalEquidistant](#) - the azimuthal equidistant projection.
- [d3.geo.conicConformal.raw](#) - the raw conic conformal projection.
- [d3.geo.conicConformal](#) - the conic conformal projection.
- [d3.geo.conicEqualArea.raw](#) the raw conic equal-area (a.k.a. Albers) projection.
- [d3.geo.conicEqualArea](#) the conic equal-area (a.k.a. Albers) projection.
- [d3.geo.conicEquidistant.raw](#) - the raw conic equidistant projection.
- [d3.geo.conicEquidistant](#) - the conic equidistant projection.
- [d3.geo.equirectangular.raw](#) - the raw equirectangular (plate carrée) projection.
- [d3.geo.equirectangular](#) - the equirectangular (plate carrée) projection.
- [d3.geo.gnomonic.raw](#) - the raw gnomonic projection.
- [d3.geo.gnomonic](#) - the gnomonic projection.
- [d3.geo.mercator.raw](#) - the raw Mercator projection.
- [d3.geo.mercator](#) - the spherical Mercator projection.
- [d3.geo.orthographic.raw](#) - the raw azimuthal orthographic projection.
- [d3.geo.orthographic](#) - the azimuthal orthographic projection.
- [d3.geo.projectionMutator](#) - create a standard projection from a mutable raw projection.
- [d3.geo.projection](#) - create a standard projection from a raw projection.
- [d3.geo.stereographic.raw](#) - the raw azimuthal stereographic projection.
- [d3.geo.stereographic](#) - the azimuthal stereographic projection.
- [d3.geo.transverseMercator.raw](#) - the raw transverse Mercator projection.

- [projection.center](#) - get or set the projection's center location.
- [projection.clipAngle](#) - get or set the radius of the projection's clip circle.
- [projection.clipExtent](#) - get or set the projection's viewport clip extent, in pixels.
- [projection.invert](#) - invert the projection for the specified point.
- [projection.precision](#) - get or set the precision threshold for adaptive resampling.
- [projection.rotate](#) - get or set the projection's three-axis rotation.
- [projection.scale](#) - get or set the projection's scale factor.
- [projection.stream](#) - wrap the specified stream listener, projecting input geometry.
- [projection.translate](#) - get or set the projection's translation position.
- [projection](#) - project the specified location.

Streams

- [clipExtent.extent](#) - sets the clip extent.
- [d3.geo.clipExtent](#) - a stream transform that clips geometries to a given axis-aligned rectangle.
- [d3.geo.stream](#) - convert a GeoJSON object to a geometry stream.
- [d3.geo.transform](#) - transform streaming geometries.
- [stream.lineEnd](#) - indicate the end of a line or ring.
- [stream.lineStart](#) - indicate the start of a line or ring.
- [stream.point](#) - indicate an x, y (and optionally z) coordinate.
- [stream.polygonEnd](#) - indicate the end of a polygon.
- [stream.polygonStart](#) - indicate the start of a polygon.
- [stream.sphere](#) - indicate a sphere.
- [transform.stream](#) - wraps a given stream.

d3.geom (Geometry)

Voronoi

- [d3.geom.voronoi](#) - create a Voronoi layout with default accessors.
- [voronoi.clipExtent](#) - get or set the clip extent for the tessellation.
- [voronoi.links](#) - compute the Delaunay mesh as a network of links.
- [voronoi.triangles](#) - compute the Delaunay mesh as a triangular tessellation.
- [voronoi.x](#) - get or set the x-coordinate accessor for each point.
- [voronoi.y](#) - get or set the y-coordinate accessor for each point.
- [voronoi](#) - compute the Voronoi tessellation for the specified points.

Quadtree

- [d3.geom.quadtree](#) - constructs a quadtree for an array of points.
- [quadtree.add](#) - add a point to the quadtree.
- [quadtree.find](#) - find the closest point in the quadtree.
- [quadtree.visit](#) - recursively visit nodes in the quadtree.

Polygon

- [d3.geom.polygon](#) - create a polygon from the specified array of points.
- [polygon.area](#) - compute the counterclockwise area of this polygon.
- [polygon.centroid](#) - compute the area centroid of this polygon.
- [polygon.clip](#) - clip the specified polygon to this polygon.

Hull

- [d3.geom.hull](#) - create a convex hull layout with default accessors.
- [hull](#) - compute the convex hull for the given array of points.
- [hull.x](#) - get or set the x -coordinate accessor.
- [hull.y](#) - get or set the y -coordinate accessor.

d3.behavior (Behaviors)

Drag

- [d3.behavior.drag](#)
- [drag.on](#)
- [drag.origin](#)

Zoom

- [d3.behavior.zoom](#) - create a zoom behavior.
- [zoom.center](#) - an optional focal point for mousewheel zooming.
- [zoom.duration](#) - get or set the dblclick transition duration.
- [zoom.event](#) - dispatch zoom events after setting the scale or translate.
- [zoom.on](#) - listeners for when the scale or translate changes.
- [zoom.scaleExtent](#) - optional limits on the scale factor.
- [zoom.scale](#) - the current scale factor.

- [zoom.size](#) - the dimensions of the viewport.
- [zoom.translate](#) - the current translate offset.
- [zoom.x](#) - an optional scale whose domain is bound to the x extent of the viewport.
- [zoom.y](#) - an optional scale whose domain is bound to the y extent of the viewport.
- [zoom](#) - apply the zoom behavior to the selected elements.

- [Status](#)
- [API](#)
- [Training](#)
- [Shop](#)
- [Blog](#)
- [About](#)
- [Help](#)

- © 2015 GitHub, Inc.
- [Terms](#)
- [Privacy](#)
- [Security](#)
- [Contact](#)

Something went wrong with that request. Please try again.

You signed in with another tab or window. [Reload](#) to refresh your session. You signed out in another tab or window. [Reload](#) to refresh your session.



to
content
Sign up Sign in

- **Explore**
- **Features**
- **Enterprise**
- **Blog**

- [Watch](#)
- [Star](#)
- [Fork](#)

mbostock/d3

- [Code](#)
- [Issues](#)
- [Pull requests](#)
- **Wiki**
- [Pulse](#)
- [Graphs](#)

Ordinal Scales

mbostock edited this page Dec 9, 2014 · 44 revisions

Pages 75

- [Home](#)
- [3.0](#)
- [3.1](#)
- [API 中文手册](#)
- [API Reference](#)
- [API Reference \(\)](#)
- [Api 参考](#)
- [Arrays](#)
- [Behaviors](#)
- [Bundle Layout](#)
- [Chord Layout](#)
- [Cluster Layout](#)
- [CN Home](#)
- [Colors](#)
- [Core](#)
- [Show 60 more pages...](#)

Clone this wiki locally

Wiki • [API Reference](#) • [Scales](#) •
Ordinal Scales

Scales are functions that map from an input domain to an output range. **Ordinal** scales have a discrete domain, such as a set of names or categories. There are also [quantitative scales](#), which have a continuous domain, such as the set of real numbers. Scales are an optional feature in D3; you don't have to use them, if you

prefer to do the math yourself. However, using scales can greatly simplify the code needed to map a dimension of data to a visual representation.

A scale object, such as that returned by [d3.scale.ordinal](#), is both an object and a function. That is: you can call the scale like any other function, and the scale has additional methods that change its behavior. Like other classes in D3, scales follow the method chaining pattern where setter methods return the scale itself, allowing multiple setters to be invoked in a concise statement.

[d3.scale.ordinal](#)()

Constructs a new ordinal scale with an empty domain and an empty range. The ordinal scale is invalid (always returning undefined) until an output range is specified.

[ordinal](#)(x)

Given a value *x* in the input domain, returns the corresponding value in the output range.

If the range was specified explicitly (as by [range](#), but not [rangeBands](#), [rangeRoundBands](#) or [rangePoints](#)), *and* the given value *x* is not in the scale's [domain](#), then *x* is implicitly added to the domain; subsequent invocations of the scale given the same value *x* will return the same value *y* from the range.

[ordinal.domain](#)([*values*])

If *values* is specified, sets the input domain of the ordinal scale to the specified array of values. The first element in *values* will be

mapped to the first element in the output range, the second domain value to the second range value, and so on. Domain values are stored internally in an associative array as a mapping from value to index; the resulting index is then used to retrieve a value from the output range. Thus, an ordinal scale's values must be coercible to a string, and the stringified version of the domain value uniquely identifies the corresponding range value. If *values* is not specified, this method returns the current domain.

Setting the domain on an ordinal scale is optional. If no domain is set, a [range](#) must be set explicitly. Then, each unique value that is passed to the scale function will be assigned a new value from the output range; in other words, the domain will be inferred implicitly from usage. Although domains may thus be constructed implicitly, it is still a good idea to assign the ordinal scale's domain explicitly to ensure deterministic behavior, as inferring the domain from usage will be dependent on ordering.

ordinal.**range**(*[values]*)

If *values* is specified, sets the output range of the ordinal scale to the specified array of values. The first element in the domain will be mapped to the first element in *values*, the second domain value to the second range value, and so on. If there are fewer elements in the range than in the domain, the scale will recycle values from the start of the range. If *values* is not specified, this method returns the current output range.

This method is intended for when the set of discrete output values is computed explicitly, such as a set of categorical colors. In other cases, such as determining

the layout of an ordinal scatterplot or bar chart, you may find the [rangePoints](#) or [rangeBands](#) operators more convenient.

ordinal.**rangePoints**(*interval*[, *padding*])

Sets the output range from the specified continuous *interval*. The array *interval* contains two elements representing the minimum and maximum numeric value. This interval is subdivided into *n* evenly-spaced **points**, where *n* is the number of (unique) values in the input domain. The first and last point may be offset from the edge of the interval according to the specified *padding*, which defaults to zero. The *padding* is expressed as a multiple of the spacing between points. A reasonable value is 1.0, such that the first and last point will be offset from the minimum and maximum value by half the distance between points.

rangepoints

```
var o = d3.scale.ordinal()
    .domain([1, 2, 3, 4])
    .rangePoints([0, 100]);

o.range(); // [0,
33.333333333333336,
66.66666666666667, 100]
```

ordinal.**rangeRoundPoints**(*interval*[, *padding*])

Like [rangePoints](#), except guarantees that the range values are integers so as to avoid antialiasing artifacts.

```
var o = d3.scale.ordinal()
    .domain([1, 2, 3, 4])
    .rangeRoundPoints([0, 100]);
```

```
o.range(); // [1, 34, 67, 100]
```

Note that rounding necessarily introduces additional outer padding which is, on average, proportional to the length of the domain. For example, for a domain of size 50, an additional 25px of outer padding on either side may be required. Modifying the range extent to be closer to a multiple of the domain length may reduce the additional padding.

```
var o = d3.scale.ordinal()
    .domain(d3.range(50))
    .rangeRoundPoints([0, 95]);

o.range(); // [23, 24, 25, ..., 70, 71, 72]
o.rangeRoundPoints([0, 100]);
o.range(); // [1, 3, 5, ..., 95, 97, 98]
```

(Alternatively, you could round the output of the scale manually or apply shape-rendering: crispEdges. However, this will result in irregularly spaced points.)

ordinal.rangeBands(interval[, padding[, outerPadding]])

Sets the output range from the specified continuous *interval*. The array *interval* contains two elements representing the minimum and maximum numeric value. This interval is subdivided into *n* evenly-spaced **bands**, where *n* is the number of (unique) values in the input domain. The bands may be offset from the edge of the interval and other bands according to the specified *padding*, which defaults to zero. The padding is typically in the range [0,1] and corresponds to the amount of space in the range interval to allocate to padding. A

value of 0.5 means that the band width will be equal to the padding width. The *outerPadding* argument is for the entire group of bands; a value of 0 means there will be padding only between rangeBands.

rangebands

```
var o = d3.scale.ordinal()
    .domain([1, 2, 3])
    .rangeBands([0, 100]);

o.rangeBand(); // 33.333333333333336
o.range(); // [0,
33.333333333333336,
66.66666666666667]
o.rangeExtent(); // [0, 100]
```

ordinal.**rangeRoundBands**(*interval*[,
padding[, *outerPadding*]])

Like [rangeBands](#), except guarantees that range values and band width are integers so as to avoid antialiasing artifacts.

```
var o = d3.scale.ordinal()
    .domain([1, 2, 3])
    .rangeRoundBands([0, 100]);

o.range(); // [1, 34, 67]
o.rangeBand(); // 33
o.rangeExtent(); // [0, 100]
```

Note that rounding necessarily introduces additional outer padding which is, on average, proportional to the length of the domain. For example, for a domain of size 50, an additional 25px of outer padding on either side may be required. Modifying the range extent to be closer to a multiple of the domain length may reduce the additional padding.

```
var o = d3.scale.ordinal()
    .domain(d3.range(50))
    .rangeRoundBands([0, 95]);

o.range(); // [23, 24, 25, ..., 70,
71, 72]

o.rangeRoundBands([0, 100]);
o.range(); // [0, 2, 4, ..., 94, 96,
98]
```

(Alternatively, you could round the output of the scale manually or apply shape-rendering: crispEdges. However, this will result in irregularly spaced and sized bands.)

ordinal.rangeBand()

Returns the band width. When the scale's range is configured with rangeBands or rangeRoundBands, the scale returns the lower value for the given input. The upper value can then be computed by offsetting by the band width. If the scale's range is set using range or rangePoints, the band width is zero.

ordinal.rangeExtent()

Returns a two-element array representing the extent of the scale's range, i.e., the smallest and largest values.

ordinal.copy()

Returns an exact copy of this ordinal scale. Changes to this scale will not affect the returned scale, and vice versa.

Categorical Colors

d3.scale.category10()

Constructs a new ordinal scale with a range of ten categorical colors:

1f77b4	#1f77b4
ff7f0e	#ff7f0e
2ca02c	#2ca02c
d62728	#d62728
9467bd	#9467bd
8c564b	#8c564b
e377c2	#e377c2
7f7f7f	#7f7f7f
bcbd22	#bcbd22
17becf	#17becf

d3.scale.**category20()**

Constructs a new ordinal scale with a range of twenty categorical colors:

1f77b4	#1f77b4
aec7e8	#aec7e8
ff7f0e	#ff7f0e
ffbb78	#ffbb78
2ca02c	#2ca02c
98df8a	#98df8a
d62728	#d62728
ff9896	#ff9896
9467bd	#9467bd
c5b0d5	#c5b0d5
8c564b	#8c564b
c49c94	#c49c94
e377c2	#e377c2
f7b6d2	#f7b6d2
7f7f7f	#7f7f7f

c7c7c7	#c7c7c7
bcbd22	#bcbd22
dbdb8d	#dbdb8d
17becf	#17becf
9edae5	#9edae5

d3.scale.**category20b()**





















Constructs a new ordinal scale with a range of twenty categorical colors:

393b79	#393b79
5254a3	#5254a3
6b6ecf	#6b6ecf
9c9ede	#9c9ede
637939	#637939
8ca252	#8ca252
b5cf6b	#b5cf6b
cedb9c	#cedb9c
8c6d31	#8c6d31
bd9e39	#bd9e39
e7ba52	#e7ba52

	#e7cb94
	#843c39
	#ad494a
	#d6616b
	#e7969c
	#7b4173
	#a55194
	#ce6dbd
	#de9ed6

d3.scale.**category20c()**

Constructs a new ordinal scale with a range of twenty categorical colors:

	#3182bd
	#6baed6
	#9ecae1
	#c6dbef
	#e6550d
	#fd8d3c
	#fdae6b
	#fdd0a2
	#31a354
	#74c476
	#a1d99b
	#c7e9c0
	#756bb1
	#9e9ac8
	#bcbddc
	#dadaeb
	#636363
	#969696
	#bdbdbd
	#d9d9d9

ColorBrewer

D3 also bundles some fantastic categorical color scales by [Cynthia Brewer](#). You can find those in either CSS or JavaScript form in [lib/colorbrewer](#).

For CSS, assign a class such as "q0-3", "q1-3" or "q2-3" to the element you wish it be filled. Then, set the class attribute on a

parent element (such as the SVG element) with the desired color scale name, such as "RdBu" or "Blues". For examples, see: [calendar heatmap](#), [choropleth](#).

For JavaScript, you can use `colorbrewer`. `RdBu[9]` or equivalent as the range of a `d3.scale.ordinal`. For example:

```
var o = d3.scale.ordinal()  
    .domain(["foo", "bar", "baz"])  
    .range(colorbrewer.RdBu[9]);
```

-
- [Status](#)
 - [API](#)
 - [Training](#)
 - [Shop](#)
 - [Blog](#)
 - [About](#)
 - [Help](#)
 - © 2015 GitHub, Inc.
 - [Terms](#)
 - [Privacy](#)
 - [Security](#)
 - [Contact](#)

Something went wrong with that request. Please try again.