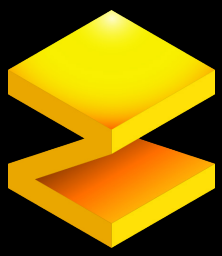

3Data

zeus.gl

Developer Documentation

Table of Contents

zues.gl Developer Documentation

Plot Container

The plot container holds all other plot elements as children, and handles some high level management and defaults.

plot

Place the plot component on the parent entity containing all layers, guides, et cetera.

This component manages plot elements, ensuring updates are performed in the correct order for accuracy and positioning any number of legends automatically.

In the future, the plot component should also provide heritable defaults for themes and aesthetic mappings.

Layers

A plot can have one or more layers which add the visual aspects to a plot by mapping data to aesthetic properties to create marks such as points or lines. Aesthetic properties are always arrays. Required aesthetics for a layer must have a value for each mark and have the same length. Optional aesthetics can either have one value for each mark or a single default value that will be used for all marks. If not specified, optional aesthetic default values will be determined by the theme. Unless a related scale is added to the plot, layers expect aesthetic data to already be scaled, e.g. x, y, and z in a range of 0 to 1.

layer-point

layer-point creates a 3D scatterplot by placing primitive geometries as marks at x, y, and z coordinates.

Aesthetic Property	Descrition	Default Value
x	x coordinate	Required
y	y coordinate	Required
z	z coordinate	Required
Shape	Geometry primitive for each mark	[ ]
Size	Size of each mark (scaling percentage)	[ ]
Color	Color of each mark	[ ]
Spin	Rotation speed of each mark (radians per second)	[ ]

layer-text

A layer using mappable text as the mark, e.g. for data labels

Aesthetic Property	Description	Default Value
x	x coordinates	Required
y	y coordinates	Required
z	z coordinates	Required
label	text to display	Required
size	Size of each label (scaling percentage)	[ ]
color	Color of each label	[ ]
angle	y-axis rotation of each label	[ ]

Static Property	Description	Default Value
mixin	Id of a mixin to assign to every label, e.g. for specifying text attribute alignment	‘’

Layer development

plot-data API

The plot-data component serves as an intermediary between a layer and its input data to allow for blending of data from multiple sources, applying transforms to the data in a manner that is opaque to layers, and inheritance of data specified at the plot level. Every layer should include plot-data as a dependency component and use the following methods:

registerLayer(component): call in init to connect the layer.
```
this.plotData = this.el.components[‘plot-data’]
this.plotData.registerLayer(this, this.aesthetics)
```

prepareData(): call in prepare before training scales (generally before utilizing the aesthetic data in any way). This allows transforms a first pass at the data.

get(aestheticName): use this method whenever accessing aesthetic data to retreive the properly merged and/or transformed data.

finishData(dataScaled): call in draw after scaling with the object containing all scaled aesthetic data. This allows transforms a second pass for changes in plot-space (as opposed to raw data space).
wrapUp(dataScaled, instanceGroups): call at the end of draw after all 3D object creation is complete and pass in the optional instanceGroups argument for instanced layers. Allows transforms to make final adjustments such as animations.

Transforms

Transforms have the opportunity to modify the data seen by a layer before and/or after it is scaled., e.g. counting all the observations that fall in a single x,z bin for a histogram and replacing all of the individual observations with just the bin totals.

API

The plot-data component acts as the intermediary for this and should be listed as a dependency in every transform component.

plot-data.registerTransform(component): call in transform init to link with layer (note that the specific layer in use need not be known as it will also register itself with plot-data)
  this.el.components['plot-data'].registerTransform(this)

Transform components expose the following methods to hook into the plotting process. All methods are optional.

Setup hooks

plotBound(plotComponent): Called as soon as the parent plot component is available (may be called during registerTransform handler).

layerBound(layerComponent): Called as soon as the layer component is available (may be called during registerTransform handler).

Layer-level hooks

These hooks are called from layer methods. They should not depend on the state of other layers in the plot as the calling order is indeterminable.

getTransformed(aestheticName, inputArray): A chance to replace the pre-scaling input data whenever a layer accesses it. Do not modify the inputData array in place, as it may be linked via data binding to other layers and plots. Return the replacement array or return undefined to leave it unchanged.

preScalingTransform(inputData, layerComponent): A chance to perform operations based on the entire data object before it is seen by layers and scaled. Do not modify the arrays in place as they may be linked via data binding to other layers and plots. Often used to create the transformed data and store it for later use in getTransformed.

postScalingTransform(scaledData, layerComponent): A chance to modify the data after it has been scaled into plot-space but before the layer has performed any drawing. Modify the scaledData object and its arrays directly; it is private to the current layer and is accessed directly by the layer during drawing (getTransformed is not used after scaling)

postDrawingHook(scaledData, layerComponent, instanceGroups): A final hook after the layer has finished drawing, enabling access to the created 3D objects. The optional instanceGroups argument will contain information about instancing indices for instanced layers.

Plot level hooks

These hooks are called for all transforms from the main plot update handler in between plot phases. Here you can be sure all layers have finished the previous phase.

mediScalingTransform(inputData, layerComponent): Called after all layers have finished scale training but before any data has been scaled. A safe time to change scale parameters.

plotWrapUpHook(layerComponent): Called after all layers and guides have completed drawing just before the 'plot-update-complete'event. plot.isUpdating is still true at this stage.

Layer v. Plot transforms: Transforms are generally applied to layers, but may be registered on the plot container as well. Transforms on the plot have a limited API: preScalingTransform (called before any layers prepare), mediScalingTransform (called before other layer-level medi-scaling transforms) and plotWrapUpHook (called before other layer-level medi-scaling transforms).

transform-bin-count

Transforms data into a 2D x/z histogram with hexagonal binning and a y value equal to the count in each bin.

schema

Property	Description
outputTo	Array. Which aesthetics will be overwritten with the bin count.
stretchToFit	boolean. Whether to modify the size of layer marks to fill the bin area. Useful for histogram layers like layer-honeycomb, but generally not desired when applied to others like layer-text

transform-force-directed-simulation

Plot-level transform to manage a 3D force-directed simulation running on worker threads.

Property	Description
tickSpeed	How frequently the simulation is updated (ms)

transform-force-directed

Layer-level transform to apply the results of transform-force-directed-simulation to a layer in real time.

Property	Description
output	‘edges’ or ‘nodes’. Whether to expose the node positions (x,y,z) or edge positions (x,x2,y,y2,z,z2) to the layer

transform-reference-grid

Layer-level transform to draw a static grid across the full range of x/z values at a fixed height. Grid intervals are defined by the segments properties. Optionally, transforms the UVs of the attached layer’s mesh such that each pixel of the texture consistently aligns with the same data value. This requires the specification of textureDomain properties to define the extent of the full texture in data space. Designed to transform a layer-surface into a reference map for geospatial data visualizations in both linear and spherical coordinate systems. For a whole-each equirectangular projection, textureDomainX is [-90, 90] and textureDomainZ is [-180, 180]. The horizontal aspect of the texture maps to the Z dimension. Compatible with mod-animate-transition.

transform-geodetic

Column layer-level transform to convert positions (x,y,z) to latitude, height, and longitude.

Modifiers

Mods use the same API as transforms, and the difference is purely a naming convention to clarify intent. While transforms change what information is presented (e.g. substituting individual observations with a summary statistic), mods change how the information is presented (e.g. animating transition states)

mod-animate-transition

Intercedes in the layer update process to animate from one state to the next.

Property	Descrition
duration	Length of the animation in ms

mod-nasa-map

Requests map imagery for geospatial visualizations from NASA's Global Imagery Browse Services (GIBS). The GIBS imagery archive includes approximately 900 imagery products representing visualized science data from the NASA Earth Observing System Data and Information System (EOSDIS). More information about available map layer names can be found in NASA's GIBS documentation: https://wiki.earthdata.nasa.gov/display/GIBS/GIBS+Available+Imagery+Products Detailed developer documentation: https://wiki.earthdata.nasa.gov/display/GIBS/GIBS+API+for+Developers#GIBSAPIfor-Developers-OGCWebMapService(WMS)

Property	Description	Default
mapLayerName	Name of map layer or comma separated list of map layer names.	'BlueMarble_ShadedRelief_Bathymetry'
mapTime	Requested time of NASA GIBS map imagery. If omitted, most recent image will be used. Uses ISO 8601 date format. For example: 2018-10-01	``

mod-arcgis-map

Experimental. Requests map imagery for geospatial visualizations from an ESRI ArcGIS Server Feature Service.

Property	Description	Default
mapServerUrl	Specifies the public URL of the ArcGIS Feature Service.	'https://services.arcgisonline.com/arcgis/rest/services/World_Street_Map/MapServer'

Guides

Guides explain the scaling and mapping in a plot.

guide-axis

Add ticks, labels, title, and interaction zones for a positional aesthetic.

Property	Description	Default Value
axis	x, y, or z axis	‘x’
title	Array of length 1. Name of axis mapping.	[‘’]
breaks	Array. Positions along the axis for ticks.	[ ]
labels	Array. Text to display at each break.	[ ]

title, breaks, and labels are arrays and can be bound with data-binding.

guide-legend

Add a legend for a non-positional aesthetic

Property	Description	Default Value
aesthetic	color, shape, size, or spin	‘color’
title	Array of length 1. Name of axis mapping.	[‘’]
breaks	Array. Aesthetic values to display	[ ]
labels	Array. Text to display at each break	[ ]

title, breaks, and labels are arrays and can be bound with data-binding.

Scales

Scales map from raw data into aesthetic values. Scales are added to the plot entity and will perform data mapping for all layers and guides which utilize the aesthetic. Importantly, every scale can be 'trained' from multiple sources (e.g. layers) in order to create a consistent scaling for the entire plot.

A scale must expose the following methods for the plot manager:
- reset() (clear training)
- train(values) (determine input domain from successive calls each with an array of input data)
- scale(values) (take an array of input data and return a new array of mapped data)

If a scale implements schema properties, its update method should also perform the necessary resetting/reconstruction.

Positional scales

scale-position-x, scale-position-y, and scale-position-z map raw input data into plot space. Handles both continuous and discrete data with automatic detection.

API

The single component property can be used to specify a transform.

Value	Description
linear	Default
log	Base 10 log transform
logNatural	Natural log transform
square	Order 2 power transform

scale-shape

Map data into geometric shapes for the shape aesthetic. Will convert continuous data to a discrete scale via binning.

API

scale-shape takes a single property naming the set of shapes to use from the list below.

Value	Description	Max Levels
identity	Does no mapping. Use when the data in the shape variable has already been mapped to geometry names.	

primitives	Default. Map up to 10 levels using all available primitive geometries	10: sphere, box, cone, torus, torusKnot, cylinder, tetrahedron, octahedron, dodecahedron, and icosahedron
spinnable	Shapes which can clearly portray rotation for the spin aesthetic	6: tetrahedron, box, octahedron, dodecahedron, icosahedron, torus
sizable	Shapes which are scaled to have equivalent volume in order to accurately portray variations in the size aesthetic	6: sphere, box, icosahedron, dodecahedron, octahedron, tetrahedron
platonic	The platonic solids in increasing order of complexity. Suitable for ranked/ordered scales and best graphical performance	5: tetrahedron, box, octahedron, dodecahedron, icosahedron

scale-color

Map either continuous or discrete data to a color scale. At present, the output scale is always discrete and continuous data is binned into a discrete sequential or divergent color scale. This is because layer-point instancing is more efficient with a discrete set of colors, but future updates to the instancing library may enable continuous color.

API

Takes a d3-scale-chromatic color scheme name as its single property. The scale should be a “Categorical” scale for discrete data. For continuous data, the scale should be a “Diverging” or “Sequential” scale that provides a “scheme” and not just an interpolator.

Defaults to “Category10” for discrete data and “Spectral” for continuous.

scale-size

Scales the size of points with a multiplicative scaling factor. If used alongside a shape scale, the “sizable” or “platonic” scales are most appropriate to ensure accurate proportional representation across shapes.

API

Takes a single property input that is an array of length 2 specifying the minimum and maximum scaling factors to use.

scale-spin

Scales the y-axis rotation speed of points.

API

Takes a single property input that is an array of length 2 specifying the minimum and maximum rotation speeds in radians / second.

Themes

Themes determine the values to use for properties that aren't mapped to data, such as label font size, as well as default values to use for optional aesthetics.

theme

Layers, guides, and the plot container all have the theme component attached.

Property	Description	Default Value
size	Size of geometric marks, approximately equal to radius	0.01
shape	Primitive shape for geometric marks	'sphere'
color	Color of geometric marks	'black'
spin	Rotation speed of geometric marks (radians per second)	0
font	A-Frame text component font specification	'roboto'
fontScale	Text scale factor	0.75
fontColor	Text color	'#000'
labelDirection	Option for axis guides. 'perpendicular' makes longer labels more readable	'parallel'
titleScale	Guide title scale factor	1
titleColor	Guide title text color	'#000'
guideWidth	Size of legends	'0.3'
guideHeight	Size of legends	'0.3'
guideMargin	Padding within legends	'0.1'