SODA

Release 1.0

Jack Roddy

May 04, 2022

CONTENTS

CHAPTER

ONE

INTRODUCTION

SODA is a lightweight TypeScript/Javascript library for building dynamic and interactive visualizations of biological sequence annotation. Visualizations produced by SODA can be easily integrated with web pages, and it is easy to define interactions between SODA components and other page features.

1.1 Before you start

SODA is still in the early stages of its development and is currently maintained by one person. If you encounter any bugs, find anything in the library or documentation confusing, or think there are gaps in the feature set, *please* consider submitting an issue.

SODA adheres to the semantic versioning guidelines, so any (intentional) breaking changes to the API will be accompanied by a bump in the major version number.

1.2 Design philosophies

The development of SODA is guided by a handful of design philosophies:

SODA is developed in TypeScript

Types make code safer, easier to understand, and less painful to maintain.

TypeScript does a fantastic job of adding static typing to JavaScript. If you're not familiar with TypeScript, check out the TypeScript handbook.

Of course, you are still free to use SODA as a JavaScript library, but you'll miss out on a bit of safety.

SODA features use callback functions

If you've spent time writing JavaScript, it's a safe bet that you're familiar with the concept of a callback function. However, if you've never used callback functions before, it's probably worth taking a quick moment to check out the MDN Web Docs section on callback functions.

Callback functions are used throughout SODA for interactivity and dynamic styling.

SODA is not a visualization tool–it is a library with which visualization tools can be built

There are countless tools that provide out of the box solutions for visualizing sequence annotation; SODA is not one of those tools. Although there are many common visualization patterns for annotation data, there will always be edge case scenarios with visualization needs that don't quite fit into one of those patterns. For developers who find themselves in one of those scenarios, SODA aims to provide an option that they might find a bit more palatable than turning to a low-level visualization library like D3.

SODA makes few assumptions about your data and the way it should be visualized

SODA never tries to make stylistic decisions for you. Instead, you are in control of deciding how data is visually represented and how that representation changes in response to interactions with the visualization. The only assumption that SODA makes about your data is that it describes annotations along one dimension (e.g. a genome).

CHAPTER

TWO

INSTALLATION AND SETUP

SODA is implemented in TypeScript, which means it can be used in both TypeScript and JavaScript.

2.1 SODA as a TypeScript library

To get the full benefit of TypeScript when using SODA, you'll probably want to use it in an npm project.

If you have never used npm before, you'll first need to install Node. Depending on your operating system, there may be several ways to do that. Regardless of which platform you are on, you should be able to install it from the Node homepage

Alternatively, you could install node with a package manager:

Homebrew:

```
brew install node
```

Apt (Ubuntu, Debian):

```
sudo apt install nodejs
```

After installing node, you can initialize a directory as an npm project:

```
mkdir my-project/
cd my-project/
npm init
```

Once you have an npm project, brand new or otherwise, you can install SODA:

```
npm install @sodaviz/soda
```

If you want to, you can instead download the SODA source code from the GitHub repository and compile it with the TypeScript compiler, tsc.

2.2 SODA as a JavaScript library

If you'd rather just use SODA as a JavaScript library, the easiest way is probably to grab the bundle from skypack.

You could also download the source code from the GitHub repository, compile it, and bundle it yourself with something like webpack.

CHAPTER

THREE

TUTORIAL

Note: The tutorial is currently being reworked for the current version of SODA. If you want something to look at for now, please navigate to the tutorial from the previous version of soda. While there are a few changes to the current version that don't quite align with the previous version, it's still mostly accurate.

CHAPTER

FOUR

API

4.1 Contrib

4.1.1 Classes

RadialChart

```
class RadialChart<P extends RenderParams>
```

This Chart class is designed for rendering features in a circular context, e.g. bacterial genomes.

Type parameters

- P: RenderParams

Constructors

```
(config: RadialChartConfig <P>): RadialChart
```

Type parameters

- P: RenderParams

Parameters

- config: RadialChartConfig

Properties

_axisAnn

```
_axisAnn: undefined | Annotation
```

The Annotation object that is used to render the horizontal axis (if enabled).

_containerSelection

```
_containerSelection: undefined | Selection <any, any, any, any>
```

A d3 selection of the Chart's DOM container. This is a pre-existing DOM element (probably a div).

_padHeight

```
_padHeight: number
```

The height in pixels of the Chart's SVG pad.

_padWidth

```
_padWidth: number
```

The width in pixels of the Chart's SVG pad.

_renderParams

```
_renderParams: undefined | P
```

The last used render parameters.

_rowStripePatternSelection

```
_rowStripePatternSelection: undefined | Selection <SVGPatternElement, any, any, any>
```

A D3 selection of the SVG pattern that is used for row striping.

_rowStripeRectSelection

```
_rowStripeRectSelection: undefined | Selection <SVGRectElement, any, any, any>
```

A D3 Selection of the SVG rectangle that is used for row striping.

_selector

```
_selector: undefined | string
```

A string that can be used to uniquely select the target DOM container.

_transform

```
_transform: Transform
```

The Transform object that describes the current zoom transformation.

_viewportHeight

```
_viewportHeight: number
```

The height in pixels of the Chart's SVG viewport.

_viewportWidth

```
_viewportWidth: number
```

The width in pixels of the Chart's SVG viewport.

axisRadius

```
axisRadius: undefined | number
```

The radius of the circle that defines the axis placement.

axisType

```
axisType: undefined | Bottom | Top
```

This indicates whether or not the Chart has a horizontal axis.

defSelection

```
defSelection: Selection <any, any, any, any>
```

A d3 selection of the Chart's defs element. See: https://developer.mozilla.org/en-US/docs/Web/SVG/Element/defs

divHeight

```
divHeight: undefined | string | number
```

The CSS height property of the Chart's div.

divMargin

```
divMargin: undefined | number
```

The CSS margin property of the Chart's div.

divOutline

```
divOutline: undefined | string
```

The CSS outline property of the Chart's div.

divOverflowX

```
divOverflowX: undefined | string
```

The CSS overflow-x property of the Chart's div.

divOverflowY

```
divOverflowY: undefined | string
```

The CSS overflow-y property of the Chart's div.

divSelection

```
divSelection: Selection <any, any, any, any>
```

A d3 selection of the Chart's inner div. This is created when the Chart is instantiated and placed inside of the selected container in the DOM.

divWidth

```
divWidth: undefined | string | number
```

The CSS width property of the Chart's div.

domainConstraint

```
domainConstraint: (chart: Chart <P>): None
```

This constrains the Chart's domain, which in turn constrains both zoom level and panning. The parameter is a callback function that is evaluated after each zoom event to produce an interval that constrains the domain.

glyphModifiers

```
glyphModifiers: GlyphModifier <any, any> []
```

A list of GlyphModifiers that control the glyphs rendered in the Chart.

highlightSelection

```
highlightSelection: Selection <any, any, any, any>
```

A d3 selection of the Chart's highlight.

id

```
id: string
```

A unique identifier for the Chart.

inRender

```
inRender: (params: P): void
```

The second rendering callback function.

initialDomain

```
initialDomain: None
```

The initialized domain of the Chart when render() is called with the initializeXScale flag.

innerRadius

```
innerRadius: number
```

The inner radius of the conceptual annulus that defines the Chart annotation track.

leftPadSize

```
leftPadSize: number
```

The number of pixels of padding on the left side of the Chart.

lowerPadSize

```
lowerPadSize: number
```

The number of pixels of padding on the bottom of the Chart.

observers

```
observers: ChartObserver []
```

A list of observers attached to the Chart.

outerRadius

```
outerRadius: number
```

The outer radius of the conceptual annulus that defines the Chart annotation track.

overflowViewportSelection

```
overflowViewportSelection: Selection <any, any, any, any>
```

A d3 selection of the Chart's viewport that allows rendering overflow.

padSelection

```
padSelection: Selection <any, any, any, any>
```

A d3 selection of the viewport's padding container.

padSize

```
padSize: number
```

The number of pixels of padding around each edge of the Chart.

postRender

```
postRender: (params: P): void
```

The final rendering callback function.

postResize

```
postResize: (): void
```

The callback function that the Chart executes after resize() is called.

postZoom

```
postZoom: (): void
```

The callback function that the Chart executes after zoom() is called.

preRender

```
preRender: (params: P): void
```

The first rendering callback function.

resizable

```
resizable: boolean
```

This controls whether or not the Chart has automatic resizing enabled.

rightPadSize

```
rightPadSize: number
```

The number of pixels of padding on the right side of the Chart.

rowCount

```
rowCount: number
```

The number of rows in the Chart.

rowHeight

```
rowHeight: number
```

The height in pixels of a horizontal row in the Chart. This defaults to a value of 10.

rowStripes

```
rowStripes: boolean
```

This controls whether or not the rows will be colored in an alternating pattern.

trackHeight

```
trackHeight: number
```

The "height" of the radial track on which annotations will be rendered. Conceptually, this is equal to to the difference of the radii of two concentric circles that define an annulus.

trackOutlineSelection

```
trackOutlineSelection: undefined | Selection <any, any, any, any>
```

A d3 selection to the track outline.

upperPadSize

```
upperPadSize: number
```

The number of pixels of padding on the top of the Chart.

viewportSelection

```
viewportSelection: Selection <any, any, any, any>
```

A d3 selection of the Chart's viewport.

xScale

```
xScale: ScaleLinear <number, number>
```

A D3 scale that the Chart will use to translate between semantic and viewport coordinates. This scale will be periodically re-scaled after zoom events.

zoomConstraint

```
zoomConstraint: None
```

A Chart's contents are scaled by a scaling factor k. If a zoomConstraint of the form [min_k, max_k] is provided, the scaling factor will be constrained to that range. This will not constrain panning.

zoomable

```
zoomable: boolean
```

This controls whether or not the Chart has zooming enabled.

Accessors

containerSelection

```
get containerSelection(): Selection <any, any, any, any>
```

Get a D3 selection of the Chart's DOM Container. This throws an exception if the value is undefined, which probably means the entire chart is detached from the DOM.

padHeight

```
get padHeight(): number
```

Getter for the padHeight property.

```
set padHeight(height: number): void
```

Setter for the padHeight property. This actually adjusts the height attribute on the viewport DOM element.

padWidth

```
get padWidth(): number
```

Getter for the padWidth property.

```
set padWidth(width: number): void
```

Setter for the padWidth property. This actually adjusts the width attribute on the viewport DOM element.

renderParams

```
get renderParams(): P
```

Getter for the Chart's most recently used RenderParams.

```
set renderParams(params: P): void
```

Setter for the renderParms property.

rowStripePatternSelection

```
get rowStripePatternSelection(): Selection <SVGPatternElement, any, any, any>
```

A getter for the rowStripePatternSelection property. This serves as a null guard.

rowStripeRectSelection

```
get rowStripeRectSelection(): Selection <SVGRectElement, any, any, any>
```

A getter for the rowStripeSelection property. This serves as a null guard.

selector

```
get selector(): string
```

A getter for the Chart's selector property. The selector should be able to uniquely select the Chart's DOM container.

transform

```
get transform(): Transform
```

Getter for the transform property. This also updates the internal transform on the Chart's pad DOM element.

```
set transform(transform: Transform): void
```

Setter for the transform property.

viewportHeight

```
get viewportHeight(): number
```

Getter for the viewportHeight property.

```
set viewportHeight(height: number): void
```

Setter for the viewportHeight property. This actually adjusts the height property on the viewport DOM element.

viewportWidth

```
get viewportWidth(): number
```

Getter for the viewportWidth property.

```
set viewportWidth(width: number): void
```

Setter for the viewportWidth property. This actually adjusts the width property on the viewport DOM element.

Methods

addAxis

```
addAxis(): void
```

Returns: void

addGlyphModifier

```
addGlyphModifier(modifier: GlyphModifier <A, C>, initialize: boolean): void
```

This adds a GlyphModifier to the Chart.

Type parameters

- A: Annotation
- C: Chart

Parameters

- modifier: GlyphModifier <A, C>
- initialize: boolean

Returns: void

addTrackOutline

```
addTrackOutline(): void
```

Returns: void

alertObservers

```
alertObservers(): void
```

This calls each of this Chart's attached observer's alert() method.

Returns: void

applyGlyphModifiers

```
applyGlyphModifiers(): void
```

This applies each of the Chart's GlyphModifier.zoom() methods, resulting in each of the glyphs in the Chart being appropriately redrawn for the current zoom level.

Returns: void

applyLayoutAndSetRowCount

```
applyLayoutAndSetRowCount(params: P): void
```

Parameters

- params: P

Returns: void

calculateContainerDimensions

```
calculateContainerDimensions(): DOMRect
```

This uses d3 to select the Chart's DOM container and returns a DOMRect that describes that containers dimensions.

Returns: DOMRect

calculateDivDimensions

```
calculateDivDimensions(): DOMRect
```

Returns: DOMRect

calculatePadDimensions

```
calculatePadDimensions(): DOMRect
```

This returns a DOMRect that describes the pad dimensions.

Returns: DOMRect

calculatePadHeight

```
calculatePadHeight(): number
```

This calculates and returns the width of the SVG viewport in pixels.

Returns: number

calculatePadWidth

```
calculatePadWidth(): number
```

This calculates and returns the width of the SVG viewport in pixels.

Returns: number

calculateViewportDimensions

```
calculateViewportDimensions(): DOMRect
```

This returns a DOMRect that describes the viewport's dimensions.

Returns: DOMRect

calculateViewportHeight

```
calculateViewportHeight(): number
```

This checks the current height of the viewport in the DOM and returns it.

Returns: number

calculateViewportWidth

```
calculateViewportWidth(): number
```

This calculates the current width of the viewport in the DOM and returns it.

Returns: number

clear

```
clear(): void
```

This method clears all glyphs that have been rendered in the Chart.

Returns: void

clearHighlight

```
clearHighlight(selector: string): void
```

Parameters

- selector: string

Returns: void

configureResize

```
configureResize(): void
```

This configures the Chart to respond to browser resize events. The default resize behavior is for the Chart to maintain the current semantic view range, either stretching or shrinking the current view.

Returns: void

configureZoom

```
configureZoom(): void
```

Returns: void

defaultInRender

```
defaultInRender(params: P): void
```

Type parameters

- P: RenderParams

Parameters

- params: P

Returns: void

defaultPostRender

```
defaultPostRender(): void
```

Type parameters

- P: RenderParams

Returns: void

defaultPreRender

```
defaultPreRender(params: P): void
```

Parameters

- params: P

Returns: void

disableZoom

```
disableZoom(): void
```

This disables zooming on the Chart.

Returns: void

domainFromMousemoveEvent

```
domainFromMousemoveEvent(transform: Transform, sourceEvent: WheelEvent): None
```

Parameters

- transform: Transform
- sourceEvent: WheelEvent

Returns: None

domainFromWheelEvent

```
domainFromWheelEvent(transform: Transform, sourceEvent: WheelEvent): None
```

Parameters

- transform: Transform
- sourceEvent: WheelEvent

Returns: None

fitPadHeight

```
fitPadHeight(): void
```

This fits the Chart's SVG padding based off of the rowCount, rowHeight and padSize properties.

Returns: void

fitRadialDimensions

```
fitRadialDimensions(): void
```

Returns: void

fitRowStripes

```
fitRowStripes(): void
```

This automatically sets the dimensions of the row stripe DOM elements.

Returns: void

fitViewport

```
fitViewport(): void
```

This fits the Chart's SVG viewport based off of the Chart's pad size.

Returns: void

getContainerHeight

```
getContainerHeight(): number
```

This calculates and returns the Chart's DOM container's height in pixels.

Returns: number

getContainerWidth

```
getContainerWidth(): number
```

This calculates and returns the Chart's DOM container's width in pixels.

Returns: number

getSemanticViewRange

```
getSemanticViewRange(): ViewRange
```

Returns: ViewRange

highlight

```
highlight(config: HighlightConfig): string
```

Parameters

- config: HighlightConfig

Returns: string

initializeXScale

```
initializeXScale(start: number, end: number): void
```

This initializes an x translation scale with the provided coordinates and the dimensions of the Chart.

Parameters

- start: number
- end: number

Returns: void

initializeXScaleFromRenderParams

```
initializeXScaleFromRenderParams(params: P): void
```

This initializes an x translation scale with the provided RenderParams and the dimensions of the Chart.

Parameters

- params: P

Returns: void

render

```
render(params: P): void
```

This method stores the render parameters on the Chart and calls preRender(), inRender(), and postRender().

Parameters

- params: P

Returns: void

renderAxis

```
renderAxis(): void
```

Returns: void

renderTrackOutline

```
renderTrackOutline(): void
```

Returns: void

resetTransform

```
resetTransform(): void
```

Reset the Chart's transform to the zoom identity (no translation, no zoom).

Returns: void

resize

```
resize(): void
```

Returns: void

setDomain

```
setDomain(domain: None): void
```

Set the domain of the Chart's x scale.

Parameters

- domain: None

Returns: void

setRange

```
setRange(range: None): void
```

Set the range of the Chart's x scale.

Parameters

- range: None

Returns: void

setRowStripes

```
setRowStripes(): void
```

This initializes the DOM elements that form the row stripes in the Chart, if enabled.

Returns: void

setToContainerDimensions

```
setToContainerDimensions(): void
```

This calculates the Chart's DOM container's dimensions and sets the Chart's SVG pad to fill those dimensions.

Returns: void

squareToContainerHeight

```
squareToContainerHeight(): void
```

This calculates the height of the Chart's DOM container and sets the Chart's SVG pad to a square with that height.

Returns: void

squareToContainerWidth

```
squareToContainerWidth(): void
```

This calculates the width of the Chart's DOM container and sets the Chart's SVG pad to a square with that width.

Returns: void

squareToDivWidth

```
squareToDivWidth(): void
```

Returns: void

updateDivProperties

```
updateDivProperties(): void
```

Returns: void

updateRange

```
updateRange(): void
```

Returns: void

zoom

```
zoom(): void
```

Returns: void

zoomHighlight

```
zoomHighlight(): void
```

Returns: void

inferRenderRange

```
inferRenderRange(params: P): None
```

A utility function to attempt to infer a semantic range on RenderParams when no range is explicitly supplied.

Type parameters

- P: RenderParams

Parameters

- params: P

Returns: None

4.1.2 Interfaces

AlignmentAnnotations

```
interface AlignmentAnnotations
```

The return type for the getAlignmentAnnotations() function.

Properties

all

```
all: SequenceAnnotation []
```

gaps

```
gaps: SequenceAnnotation
```

insertions

```
insertions: SequenceAnnotation []
```

matches

```
matches: SequenceAnnotation
```

substitutions

```
substitutions: SequenceAnnotation
```

AlignmentConfig

```
interface AlignmentConfig
```

This defines the parameters for a call to the getAlignmentAnnotations() function.

Properties

end

```
end: undefined | number
```

id

```
id: string
```

query

```
query: string
```

row

```
row: number
```

start

```
start: number
```

target

```
target: string
```

RadialChartConfig

```
interface RadialChartConfig<P extends RenderParams>
```

A simple interface that defines the parameters that initialize a RadialChart

Type parameters

- P: RenderParams

Properties

axisType

```
axisType: undefined | Bottom | Top
```

This controls whether or not the Chart will render a horizontal axis.

debugShading

```
debugShading: undefined | boolean
```

If this is set to true, the pad and viewport will be shaded so that they are visible in the browser.

divHeight

```
divHeight: undefined | string | number
```

The height in pixels of the Chart's containing div.

divMargin

```
divMargin: undefined | number
```

The CSS margin property for the Chart's div.

divOutline

```
divOutline: undefined | string
```

The CSS outline property for the Chart's div.

divOverflowX

```
divOverflowX: undefined | string
```

The CSS overflow-x setting of the Chart's containing div.

divOverflowY

```
divOverflowY: undefined | string
```

The CSS overflow-y setting of the Chart's containing div.

divWidth

```
divWidth: undefined | string | number
```

The width in pixels of the Chart's containing div.

domainConstraint

```
domainConstraint: undefined | (chart: Chart <P>): None
```

This constrains the Chart's domain, which in turn constrains both zoom level and panning. The parameter is a callback function that is evaluated after each zoom event to produce an interval that constrains the domain.

id

```
id: undefined | string
```

A unique identifier for the Chart. This will be generated automatically if one isn't provided.

inRender

```
inRender: undefined | (params: P): void
```

The second rendering callback function.

leftPadSize

```
leftPadSize: undefined | number
```

The number of pixels of padding on the left side of the Chart.

lowerPadSize

```
lowerPadSize: undefined | number
```

The number of pixels of padding on the bottom of the Chart.

padSize

```
padSize: undefined | number
```

The number of pixels of padding around each edge of the Chart.

postRender

```
postRender: undefined | (params: P): void
```

The final rendering callback function.

postResize

```
postResize: undefined | (): void
```

The callback function that the Chart executes after resize() is called.

postZoom

```
postZoom: undefined | (): void
```

The callback function that the Chart executes after zoom() is called.

preRender

```
preRender: undefined | (params: P): void
```

The first rendering callback function.

resizable

```
resizable: undefined | boolean
```

This controls whether or not the Chart will automatically resize itself as it's container changes size. This will cause the Chart to ignore explicit height/width arguments in the config.

rightPadSize

```
rightPadSize: undefined | number
```

The number of pixels of padding on the right side of the Chart.

rowCount

```
rowCount: undefined | number
```

The number of rows that will be rendered.

rowHeight

```
rowHeight: undefined | number
```

The height in pixels of a horizontal row in the Chart. This defaults to a value of 10.

rowStripes

```
rowStripes: undefined | boolean
```

This controls whether or not the rows will be colored in an alternating pattern.

selector

```
selector: undefined | string
```

A string that can be used to uniquely select the target DOM container.

trackHeight

```
trackHeight: undefined | number
```

The "height" of the radial track on which annotations will be rendered. Conceptually, this is equal to to the difference of the radii of two concentric circles that define an annulus.

upperPadSize

```
upperPadSize: undefined | number
```

The number of pixels of padding on the top of the Chart.

zoomConstraint

```
zoomConstraint: undefined | None
```

A Chart's contents are scaled by a scaling factor k. If a zoomConstraint of the form [min_k, max_k] is provided, the scaling factor will be constrained to that interval. This will not constrain panning.

zoomable

```
zoomable: undefined | boolean
```

This controls whether or not the Chart will be configured to allow zooming and panning.

4.1.3 Functions

getAlignmentAnnotations

```
function getAlignmentAnnotations(config: AlignmentConfig): None
```

This returns a set of SequenceAnnotations defined such that the provided query sequence can be rendered in a Chart as if it were aligned to a chromosome. The matches, substitutions, gaps, and insertions are returned as separate objects. The idea here is that they can be rendered individually with different style parameters.

Parameters

- config: AlignmentConfig

Returns: None

radialRectangle

```
function radialRectangle<A extends Annotation, C extends RadialChart>(config:␣
↪RectangleConfig <A, C>): d3.Selection
```

This renders a list of Annotation objects as rectangles in a RadialChart.

Type parameters

- A: Annotation
- C: RadialChart

Parameters

- config: RectangleConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

sliceContinuousAnnotation

```
function sliceContinuousAnnotation(annotation: ContinuousAnnotation, start: number, end:␣
↪number): None
```

Parameters

- annotation: ContinuousAnnotation
- start: number
- end: number

Returns: ContinuousAnnotation | undefined

sliceSequenceAnnotation

```
function sliceSequenceAnnotation(annotation: SequenceAnnotation, start: number, end:␣
↪number): None
```

Parameters

- annotation: SequenceAnnotation
- start: number
- end: number

Returns: SequenceAnnotation | undefined

4.2 Classes

4.2.1 Annotation

```
class Annotation
```

Annotation objects are the main data structure used by SODA to store annotation data.

Constructors

```
(config: AnnotationConfig): Annotation
```

Parameters

- config: AnnotationConfig

Properties

end

```
end: number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

row

```
row: number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

start

```
start: number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

width

```
width: number
```

The width of the annotation in semantic coordinates.

Accessors

w

```
get w(): number
```

A convenience getter that returns the width property.

```
set w(w: number): void
```

A convenience setter that sets the width property.

x

```
get x(): number
```

A convenience getter that returns the start property.

```
set x(x: number): void
```

A convenience setter that sets the start property.

x2

```
get x2(): number
```

A convenience getter that returns the end property.

```
set x2(x: number): void
```

A convenience setter that sets the end property.

y

```
get y(): number
```

A convenience getter that returns the row property.

```
set y(y: number): void
```

A convenience setter that sets the row property.

4.2.2 AnnotationGroup

```
class AnnotationGroup<A extends Annotation>
```

An Annotation class that contains a group of Annotations.

Type parameters

- A: Annotation

Constructors

```
(config: AnnotationGroupConfig <A>): AnnotationGroup
```

Type parameters

- A: Annotation

Parameters

- config: AnnotationGroupConfig

Properties

end

```
end: number
```

group

```
group: A []
```

The group of Annotations that live in this object.

id

```
id: string
```

row

```
row: number
```

start

```
start: number
```

suppressWarnings

```
suppressWarnings: boolean
```

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

width

```
width: number
```

Accessors

w

```
get w(): number
```

A convenience getter that returns the width property.

```
set w(w: number): void
```

A convenience setter that sets the width property.

x

```
get x(): number
```

A convenience getter that returns the start property.

```
set x(x: number): void
```

A convenience setter that sets the start property.

x2

```
get x2(): number
```

A convenience getter that returns the end property.

```
set x2(x: number): void
```

A convenience setter that sets the end property.

y

```
get y(): number
```

A convenience getter that returns the row property.

```
set y(y: number): void
```

A convenience setter that sets the row property. It also sets the row property on every member of the group property.

Methods

add

```
add(ann: A | A []): void
```

Add an Annotation or list of Annotations to the group.

Parameters

- ann: A | A []

Returns: void

addAnnotation

```
addAnnotation(ann: A): void
```

Add an Annotation to the group.

Parameters

- ann: A

Returns: void

4.2.3 Bed12Annotation

```
class Bed12Annotation
```

An annotation object to represent BED annotations explicitly constrained in the BED12 format.

Constructors

```
(config: Bed12AnnotationConfig): Bed12Annotation
```

Parameters

- config: Bed12AnnotationConfig

Properties

blockCount

```
blockCount: number
```

A BED12 field for records that should be drawn as discontiguous/fragmented glyphs. This describes the number of fragments.

blockSizes

```
blockSizes: number []
```

A BED12 field for records that should be drawn as discontiguous/fragmented glyphs. This describes the size of each fragment.

blockStarts

```
blockStarts: number []
```

A BED12 field for records that should be drawn as discontiguous/fragmented glyphs. This describes the offset of each fragment.

chrom

```
chrom: string
```

A BED3 field that describes the chromosome of the record.

end

```
end: number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

itemRgb

```
itemRgb: string
```

A BED9 field BED field that defines the color of the feature. It is an RGB string, e.g. (0, 1, 256).

name

```
name: string
```

A BED6 field that describes the name of the record.

row

```
row: number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

score

```
score: number
```

A BED6 field that describes the "score" of the record.

start

```
start: number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

strand

```
strand: Orientation
```

A BED6 field that describes the orientation/strand of the record.

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

thickEnd

```
thickEnd: number
```

A BED9 field that describes at which coordinate the feature should stop being drawn "thickly."

thickStart

```
thickStart: number
```

A BED9 field that describes at which coordinate the feature should start being drawn "thickly."

width

```
width: number
```

The width of the annotation in semantic coordinates.

Accessors

w

```
get w(): number
```

A convenience getter that returns the width property.

```
set w(w: number): void
```

A convenience setter that sets the width property.

x

```
get x(): number
```

A convenience getter that returns the start property.

```
set x(x: number): void
```

A convenience setter that sets the start property.

x2

```
get x2(): number
```

A convenience getter that returns the end property.

```
set x2(x: number): void
```

A convenience setter that sets the end property.

y

```
get y(): number
```

A convenience getter that returns the row property.

```
set y(y: number): void
```

A convenience setter that sets the row property.

4.2.4 Bed3Annotation

```
class Bed3Annotation
```

An annotation object to represent BED annotations explicitly constrained in the BED3 format.

Constructors

```
(config: Bed3AnnotationConfig): Bed3Annotation
```

Parameters

- config: Bed3AnnotationConfig

Properties

chrom

```
chrom: string
```

A BED3 field that describes the chromosome of the record.

end

```
end: number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

row

```
row: number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

start

```
start: number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

width

```
width: number
```

The width of the annotation in semantic coordinates.

Accessors

w

```
get w(): number
```

A convenience getter that returns the width property.

```
set w(w: number): void
```

A convenience setter that sets the width property.

x

```
get x(): number
```

A convenience getter that returns the start property.

```
set x(x: number): void
```

A convenience setter that sets the start property.

x2

```
get x2(): number
```

A convenience getter that returns the end property.

```
set x2(x: number): void
```

A convenience setter that sets the end property.

y

```
get y(): number
```

A convenience getter that returns the row property.

```
set y(y: number): void
```

A convenience setter that sets the row property.

4.2.5 Bed6Annotation

```
class Bed6Annotation
```

An annotation object to represent BED annotations explicitly constrained in the BED6 format.

Constructors

```
(config: Bed6AnnotationConfig): Bed6Annotation
```

Parameters

- config: Bed6AnnotationConfig

Properties

chrom

```
chrom: string
```

A BED3 field that describes the chromosome of the record.

end

```
end: number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

name

```
name: string
```

A BED6 field that describes the name of the record.

row

```
row: number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

score

```
score: number
```

A BED6 field that describes the "score" of the record.

start

```
start: number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

strand

```
strand: Orientation
```

A BED6 field that describes the orientation/strand of the record.

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

width

```
width: number
```

The width of the annotation in semantic coordinates.

Accessors

w

```
get w(): number
```

A convenience getter that returns the width property.

```
set w(w: number): void
```

A convenience setter that sets the width property.

x

```
get x(): number
```

A convenience getter that returns the start property.

```
set x(x: number): void
```

A convenience setter that sets the start property.

x2

```
get x2(): number
```

A convenience getter that returns the end property.

```
set x2(x: number): void
```

A convenience setter that sets the end property.

y

```
get y(): number
```

A convenience getter that returns the row property.

```
set y(y: number): void
```

A convenience setter that sets the row property.

4.2.6 Bed9Annotation

```
class Bed9Annotation
```

An annotation object to represent BED annotations explicitly constrained in the BED9 format.

Constructors

```
(config: Bed9AnnotationConfig): Bed9Annotation
```

Parameters

- config: Bed9AnnotationConfig

Properties

chrom

```
chrom: string
```

A BED3 field that describes the chromosome of the record.

end

```
end: number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

itemRgb

```
itemRgb: string
```

A BED9 field BED field that defines the color of the feature. It is an RGB string, e.g. (0, 1, 256).

name

```
name: string
```

A BED6 field that describes the name of the record.

row

```
row: number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

score

```
score: number
```

A BED6 field that describes the "score" of the record.

start

```
start: number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

strand

```
strand: Orientation
```

A BED6 field that describes the orientation/strand of the record.

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

thickEnd

```
thickEnd: number
```

A BED9 field that describes at which coordinate the feature should stop being drawn "thickly."

thickStart

```
thickStart: number
```

A BED9 field that describes at which coordinate the feature should start being drawn "thickly."

width

```
width: number
```

The width of the annotation in semantic coordinates.

Accessors

w

```
get w(): number
```

A convenience getter that returns the width property.

```
set w(w: number): void
```

A convenience setter that sets the width property.

x

```
get x(): number
```

A convenience getter that returns the start property.

```
set x(x: number): void
```

A convenience setter that sets the start property.

x2

```
get x2(): number
```

A convenience getter that returns the end property.

```
set x2(x: number): void
```

A convenience setter that sets the end property.

y

```
get y(): number
```

A convenience getter that returns the row property.

```
set y(y: number): void
```

A convenience setter that sets the row property.

4.2.7 BedAnnotation

```
class BedAnnotation
```

An Annotation definition for any BED records. Any fields up through BED12 are supported by this class, but nothing beyond the BED3 fields are guaranteed to be defined. For more information on BED records, see https://genome.ucsc.edu/FAQ/FAQformat.html#format1.

Constructors

```
(config: BedAnnotationConfig): BedAnnotation
```

Parameters

- config: BedAnnotationConfig

Properties

blockCount

```
blockCount: undefined | number
```

A BED12 field for records that should be drawn as discontiguous/fragmented glyphs. This describes the number of fragments.

blockSizes

```
blockSizes: undefined | number []
```

A BED12 field for records that should be drawn as discontiguous/fragmented glyphs. This describes the size of each fragment.

blockStarts

```
blockStarts: undefined | number []
```

A BED12 field for records that should be drawn as discontiguous/fragmented glyphs. This describes the offset of each fragment.

chrom

```
chrom: string
```

A BED3 field that describes the chromosome of the record.

end

```
end: number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

itemRgb

```
itemRgb: undefined | string
```

A BED9 field BED field that defines the color of the feature. It is an RGB string, e.g. (0, 1, 256).

name

```
name: undefined | string
```

A BED6 field that describes the name of the record.

row

```
row: number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

score

```
score: undefined | number
```

A BED6 field that describes the "score" of the record.

start

```
start: number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

strand

```
strand: undefined | Forward | Reverse | Unknown | Unoriented
```

A BED6 field that describes the orientation/strand of the record.

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

thickEnd

```
thickEnd: undefined | number
```

A BED9 field that describes at which coordinate the feature should stop being drawn "thickly."

thickStart

```
thickStart: undefined | number
```

A BED9 field that describes at which coordinate the feature should start being drawn "thickly."

width

```
width: number
```

The width of the annotation in semantic coordinates.

Accessors

w

```
get w(): number
```

A convenience getter that returns the width property.

```
set w(w: number): void
```

A convenience setter that sets the width property.

x

```
get x(): number
```

A convenience getter that returns the start property.

```
set x(x: number): void
```

A convenience setter that sets the start property.

x2

```
get x2(): number
```

A convenience getter that returns the end property.

```
set x2(x: number): void
```

A convenience setter that sets the end property.

y

```
get y(): number
```

A convenience getter that returns the row property.

```
set y(y: number): void
```

A convenience setter that sets the row property.

4.2.8 Chart

```
class Chart<P extends RenderParams>
```

This is used to render Annotation objects as glyphs in the browser.

Type parameters

- P: RenderParams

Constructors

```
(config: ChartConfig <P>): Chart
```

Type parameters

- P: RenderParams

Parameters

- config: ChartConfig

Properties

_axisAnn

```
_axisAnn: undefined | Annotation
```

The Annotation object that is used to render the horizontal axis (if enabled).

_containerSelection

```
_containerSelection: undefined | Selection <any, any, any, any>
```

A d3 selection of the Chart's DOM container. This is a pre-existing DOM element (probably a div).

_padHeight

```
_padHeight: number
```

The height in pixels of the Chart's SVG pad.

_padWidth

```
_padWidth: number
```

The width in pixels of the Chart's SVG pad.

_renderParams

```
_renderParams: undefined | P
```

The last used render parameters.

_rowStripePatternSelection

```
_rowStripePatternSelection: undefined | Selection <SVGPatternElement, any, any, any>
```

A D3 selection of the SVG pattern that is used for row striping.

_rowStripeRectSelection

```
_rowStripeRectSelection: undefined | Selection <SVGRectElement, any, any, any>
```

A D3 Selection of the SVG rectangle that is used for row striping.

_selector

```
_selector: undefined | string
```

A string that can be used to uniquely select the target DOM container.

_transform

```
_transform: Transform
```

The Transform object that describes the current zoom transformation.

_viewportHeight

```
_viewportHeight: number
```

The height in pixels of the Chart's SVG viewport.

_viewportWidth

```
_viewportWidth: number
```

The width in pixels of the Chart's SVG viewport.

axisType

```
axisType: undefined | Bottom | Top
```

This indicates whether or not the Chart has a horizontal axis.

defSelection

```
defSelection: Selection <any, any, any, any>
```

A d3 selection of the Chart's defs element. See: https://developer.mozilla.org/en-US/docs/Web/SVG/Element/defs

divHeight

```
divHeight: undefined | string | number
```

The CSS height property of the Chart's div.

divMargin

```
divMargin: undefined | number
```

The CSS margin property of the Chart's div.

divOutline

```
divOutline: undefined | string
```

The CSS outline property of the Chart's div.

divOverflowX

```
divOverflowX: undefined | string
```

The CSS overflow-x property of the Chart's div.

divOverflowY

```
divOverflowY: undefined | string
```

The CSS overflow-y property of the Chart's div.

divSelection

```
divSelection: Selection <any, any, any, any>
```

A d3 selection of the Chart's inner div. This is created when the Chart is instantiated and placed inside of the selected container in the DOM.

divWidth

```
divWidth: undefined | string | number
```

The CSS width property of the Chart's div.

domainConstraint

```
domainConstraint: (chart: Chart <P>): None
```

This constrains the Chart's domain, which in turn constrains both zoom level and panning. The parameter is a callback function that is evaluated after each zoom event to produce an interval that constrains the domain.

glyphModifiers

```
glyphModifiers: GlyphModifier <any, any> []
```

A list of GlyphModifiers that control the glyphs rendered in the Chart.

highlightSelection

```
highlightSelection: Selection <any, any, any, any>
```

A d3 selection of the Chart's highlight.

id

```
id: string
```

A unique identifier for the Chart.

inRender

```
inRender: (params: P): void
```

The second rendering callback function.

initialDomain

```
initialDomain: None
```

The initialized domain of the Chart when render() is called with the initializeXScale flag.

leftPadSize

```
leftPadSize: number
```

The number of pixels of padding on the left side of the Chart.

lowerPadSize

```
lowerPadSize: number
```

The number of pixels of padding on the bottom of the Chart.

observers

```
observers: ChartObserver []
```

A list of observers attached to the Chart.

overflowViewportSelection

```
overflowViewportSelection: Selection <any, any, any, any>
```

A d3 selection of the Chart's viewport that allows rendering overflow.

padSelection

```
padSelection: Selection <any, any, any, any>
```

A d3 selection of the viewport's padding container.

padSize

```
padSize: number
```

The number of pixels of padding around each edge of the Chart.

postRender

```
postRender: (params: P): void
```

The final rendering callback function.

postResize

```
postResize: (): void
```

The callback function that the Chart executes after resize() is called.

postZoom

```
postZoom: (): void
```

The callback function that the Chart executes after zoom() is called.

preRender

```
preRender: (params: P): void
```

The first rendering callback function.

resizable

```
resizable: boolean
```

This controls whether or not the Chart has automatic resizing enabled.

rightPadSize

```
rightPadSize: number
```

The number of pixels of padding on the right side of the Chart.

rowCount

```
rowCount: number
```

The number of rows in the Chart.

rowHeight

```
rowHeight: number
```

The height in pixels of a horizontal row in the Chart. This defaults to a value of 10.

rowStripes

```
rowStripes: boolean
```

This controls whether or not the rows will be colored in an alternating pattern.

upperPadSize

```
upperPadSize: number
```

The number of pixels of padding on the top of the Chart.

viewportSelection

```
viewportSelection: Selection <any, any, any, any>
```

A d3 selection of the Chart's viewport.

xScale

```
xScale: ScaleLinear <number, number>
```

A D3 scale that the Chart will use to translate between semantic and viewport coordinates. This scale will be periodically re-scaled after zoom events.

zoomConstraint

```
zoomConstraint: None
```

A Chart's contents are scaled by a scaling factor k. If a zoomConstraint of the form [min_k, max_k] is provided, the scaling factor will be constrained to that range. This will not constrain panning.

zoomable

```
zoomable: boolean
```

This controls whether or not the Chart has zooming enabled.

Accessors

containerSelection

```
get containerSelection(): Selection <any, any, any, any>
```

Get a D3 selection of the Chart's DOM Container. This throws an exception if the value is undefined, which probably means the entire chart is detached from the DOM.

padHeight

```
get padHeight(): number
```

Getter for the padHeight property.

```
set padHeight(height: number): void
```

Setter for the padHeight property. This actually adjusts the height attribute on the viewport DOM element.

padWidth

```
get padWidth(): number
```

Getter for the padWidth property.

```
set padWidth(width: number): void
```

Setter for the padWidth property. This actually adjusts the width attribute on the viewport DOM element.

renderParams

```
get renderParams(): P
```

Getter for the Chart's most recently used RenderParams.

```
set renderParams(params: P): void
```

Setter for the renderParms property.

rowStripePatternSelection

```
get rowStripePatternSelection(): Selection <SVGPatternElement, any, any, any>
```

A getter for the rowStripePatternSelection property. This serves as a null guard.

rowStripeRectSelection

```
get rowStripeRectSelection(): Selection <SVGRectElement, any, any, any>
```

A getter for the rowStripeSelection property. This serves as a null guard.

selector

```
get selector(): string
```

A getter for the Chart's selector property. The selector should be able to uniquely select the Chart's DOM container.

transform

```
get transform(): Transform
```

Getter for the transform property. This also updates the internal transform on the Chart's pad DOM element.

```
set transform(transform: Transform): void
```

Setter for the transform property.

viewportHeight

```
get viewportHeight(): number
```

Getter for the viewportHeight property.

```
set viewportHeight(height: number): void
```

Setter for the viewportHeight property. This actually adjusts the height property on the viewport DOM element.

viewportWidth

```
get viewportWidth(): number
```

Getter for the viewportWidth property.

```
set viewportWidth(width: number): void
```

Setter for the viewportWidth property. This actually adjusts the width property on the viewport DOM element.

Methods

addAxis

```
addAxis(force: boolean): void
```

If the Chart.axis property is set to true, this adds a horizontal axis to the Chart above the top row. Alternatively, if the force=true is supplied it will ignore the Chart.axis setting and add an axis anyway.

Parameters

- force: boolean

Returns: void

addGlyphModifier

```
addGlyphModifier(modifier: GlyphModifier <A, C>, initialize: boolean): void
```

This adds a GlyphModifier to the Chart.

Type parameters

- A: Annotation
- C: Chart

Parameters

- modifier: GlyphModifier <A, C>
- initialize: boolean

Returns: void

alertObservers

```
alertObservers(): void
```

This calls each of this Chart's attached observer's alert() method.

Returns: void

applyGlyphModifiers

```
applyGlyphModifiers(): void
```

This applies each of the Chart's GlyphModifier.zoom() methods, resulting in each of the glyphs in the Chart being appropriately redrawn for the current zoom level.

Returns: void

applyLayoutAndSetRowCount

```
applyLayoutAndSetRowCount(params: P): void
```

Selectively apply the layout as defined in the RenderParams argument and set the rowCount property to an appropriate value. If a rowCount is defined in the RenderParams, it will not be overwritten. If the RenderParams are configured such that no layout is applied, rowCount will be set to the max row property of the Annotations in the RenderParams.

Parameters

- params: P

Returns: void

calculateContainerDimensions

```
calculateContainerDimensions(): DOMRect
```

This uses d3 to select the Chart's DOM container and returns a DOMRect that describes that containers dimensions.

Returns: DOMRect

calculateDivDimensions

```
calculateDivDimensions(): DOMRect
```

Returns: DOMRect

calculatePadDimensions

```
calculatePadDimensions(): DOMRect
```

This returns a DOMRect that describes the pad dimensions.

Returns: DOMRect

calculatePadHeight

```
calculatePadHeight(): number
```

This calculates and returns the width of the SVG viewport in pixels.

Returns: number

calculatePadWidth

```
calculatePadWidth(): number
```

This calculates and returns the width of the SVG viewport in pixels.

Returns: number

calculateViewportDimensions

```
calculateViewportDimensions(): DOMRect
```

This returns a DOMRect that describes the viewport's dimensions.

Returns: DOMRect

calculateViewportHeight

```
calculateViewportHeight(): number
```

This checks the current height of the viewport in the DOM and returns it.

Returns: number

calculateViewportWidth

```
calculateViewportWidth(): number
```

This calculates the current width of the viewport in the DOM and returns it.

Returns: number

clear

```
clear(): void
```

This method clears all glyphs that have been rendered in the Chart.

Returns: void

clearHighlight

```
clearHighlight(selector: string): void
```

Clear highlights from the Chart. If a selector is supplied, only the highlight that matches that selector will be removed. Otherwise, all highlights will be removed.

Parameters

- selector: string

Returns: void

configureResize

```
configureResize(): void
```

This configures the Chart to respond to browser resize events. The default resize behavior is for the Chart to maintain the current semantic view range, either stretching or shrinking the current view.

Returns: void

configureZoom

```
configureZoom(): void
```

This configures the chart's viewport to appropriately handle browser zoom events.

Returns: void

defaultInRender

```
defaultInRender(params: P): void
```

Type parameters

- P: RenderParams

Parameters

- params: P

Returns: void

defaultPostRender

```
defaultPostRender(): void
```

Type parameters

- P: RenderParams

Returns: void

defaultPreRender

```
defaultPreRender(params: P): void
```

Parameters

- params: P

Returns: void

disableZoom

```
disableZoom(): void
```

This disables zooming on the Chart.

Returns: void

domainFromMousemoveEvent

```
domainFromMousemoveEvent(transform: Transform, sourceEvent: WheelEvent,␣
↪domainConstraint: None): None
```

This method produces a new domain from a browser mousemove event.

Parameters

- transform: Transform
- sourceEvent: WheelEvent
- domainConstraint: None

Returns: None

domainFromWheelEvent

```
domainFromWheelEvent(transform: Transform, sourceEvent: WheelEvent, domainConstraint:␣
↪None): None
```

This method produces a new domain from a browser wheel event.

Parameters

- transform: Transform
- sourceEvent: WheelEvent
- domainConstraint: None

Returns: None

fitPadHeight

```
fitPadHeight(): void
```

This fits the Chart's SVG padding based off of the rowCount, rowHeight and padSize properties.

Returns: void

fitRowStripes

```
fitRowStripes(): void
```

This automatically sets the dimensions of the row stripe DOM elements.

Returns: void

fitViewport

```
fitViewport(): void
```

This fits the Chart's SVG viewport based off of the Chart's pad size.

Returns: void

getContainerHeight

```
getContainerHeight(): number
```

This calculates and returns the Chart's DOM container's height in pixels.

Returns: number

getContainerWidth

```
getContainerWidth(): number
```

This calculates and returns the Chart's DOM container's width in pixels.

Returns: number

getSemanticViewRange

```
getSemanticViewRange(): ViewRange
```

Get the semantic coordinate range of what is currently shown in the Chart's viewport.

Returns: ViewRange

highlight

```
highlight(config: HighlightConfig): string
```

This method highlights a region in the Chart. If no selector is provided, one will be auto generated and returned by the function.

Parameters

- config: HighlightConfig

Returns: string

initializeXScale

```
initializeXScale(start: number, end: number): void
```

This initializes an x translation scale with the provided coordinates and the dimensions of the Chart.

Parameters

- start: number
- end: number

Returns: void

initializeXScaleFromRenderParams

```
initializeXScaleFromRenderParams(params: P): void
```

This initializes an x translation scale with the provided RenderParams and the dimensions of the Chart.

Parameters

- params: P

Returns: void

render

```
render(params: P): void
```

This method stores the render parameters on the Chart and calls preRender(), inRender(), and postRender().

Parameters

- params: P

Returns: void

resetTransform

```
resetTransform(): void
```

Reset the Chart's transform to the zoom identity (no translation, no zoom).

Returns: void

resize

```
resize(): void
```

This resizes the Chart. If the Chart has resizing enabled, this is called automatically when a browser zoom event occurs.

Returns: void

setDomain

```
setDomain(domain: None): void
```

Set the domain of the Chart's x scale.

Parameters

- domain: None

Returns: void

setRange

```
setRange(range: None): void
```

Set the range of the Chart's x scale.

Parameters

- range: None

Returns: void

setRowStripes

```
setRowStripes(): void
```

This initializes the DOM elements that form the row stripes in the Chart, if enabled.

Returns: void

setToContainerDimensions

```
setToContainerDimensions(): void
```

This calculates the Chart's DOM container's dimensions and sets the Chart's SVG pad to fill those dimensions.

Returns: void

squareToContainerHeight

```
squareToContainerHeight(): void
```

This calculates the height of the Chart's DOM container and sets the Chart's SVG pad to a square with that height.

Returns: void

squareToContainerWidth

```
squareToContainerWidth(): void
```

This calculates the width of the Chart's DOM container and sets the Chart's SVG pad to a square with that width.

Returns: void

squareToDivWidth

```
squareToDivWidth(): void
```

Returns: void

updateDivProperties

```
updateDivProperties(): void
```

Returns: void

updateRange

```
updateRange(): void
```

Set the range of the Chart's x scale to the viewport dimensions.

Returns: void

zoom

```
zoom(): void
```

This is the handler method that will be called when the Chart's viewport receives a browser zoom event.

Returns: void

zoomHighlight

```
zoomHighlight(): void
```

Returns: void

inferRenderRange

```
inferRenderRange(params: P): None
```

A utility function to attempt to infer a semantic range on RenderParams when no range is explicitly supplied.

Type parameters

- P: RenderParams

Parameters

- params: P

Returns: None

4.2.9 ChartObserver

```
class ChartObserver
```

An abstract class for objects that "observe" Charts.

Constructors

```
(): ChartObserver
```

Properties

charts

```
charts: Chart <any> []
```

A list of Charts that the Plugin will pay attention to.

Methods

add

```
add(chart: Chart | Chart <any> []): void
```

This method registers a Chart or list of Charts with the Plugin.

Parameters

- chart: Chart | Chart <any> []

Returns: void

addChart

```
addChart(chart: Chart <any>): void
```

Add a Chart to the observer.

Parameters

- chart: Chart <any>

Returns: void

alert

```
alert(chart: Chart <any>): void
```

The method that will be called when the observer is alerted by a Chart.

Parameters

- chart: Chart <any>

Returns: void

4.2.10 ContinuousAnnotation

```
class ContinuousAnnotation
```

An Annotation object that can be used to represent data that should be visualized as a plot.

Constructors

```
(config: ContinuousAnnotationConfig): ContinuousAnnotation
```

Parameters

- config: ContinuousAnnotationConfig

Properties

end

```
end: number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

maxValue

```
maxValue: number
```

The maximum y value in the data points.

minValue

```
minValue: number
```

The minimum y value in the data points.

pointWidth

```
pointWidth: number
```

The distance between two consecutive data points.

points

```
points: None []
```

The individual data points for the plot.

row

```
row: number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

start

```
start: number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

width

```
width: number
```

The width of the annotation in semantic coordinates.

Accessors

w

```
get w(): number
```

A convenience getter that returns the width property.

```
set w(w: number): void
```

A convenience setter that sets the width property.

x

```
get x(): number
```

A convenience getter that returns the start property.

```
set x(x: number): void
```

A convenience setter that sets the start property.

x2

```
get x2(): number
```

A convenience getter that returns the end property.

```
set x2(x: number): void
```

A convenience setter that sets the end property.

y

```
get y(): number
```

A convenience getter that returns the row property.

```
set y(y: number): void
```

A convenience setter that sets the row property.

4.2.11 Gff3Annotation

```
class Gff3Annotation
```

An Annotation class for storing GFF3 records. For more information see http://gmod.org/wiki/GFF3/

Constructors

```
(config: Gff3AnnotationConfig): Gff3Annotation
```

Parameters

- config: Gff3AnnotationConfig

Properties

attributes

```
attributes: undefined | Map <string, string>
```

A horrifying GFF3 field that is essentially an anything goes set of key value pairs describing anything anybody every wants to add to a GFF3 record.

end

```
end: number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

phase

```
phase: undefined | None | None | None
```

A GFF3 field that describes the phase for CDS (coding sequence) annotations.

row

```
row: number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

score

```
score: undefined | number
```

A GFF3 field that should describe the score of the annotation.

seqid

```
seqid: undefined | string
```

A GFF3 field: "The ID of the landmark used to establish the coordinate system for the current feature…"

source

```
source: undefined | string
```

A GFF3 field: "The source is a free text qualifier intended to describe the algorithm or operating procedure that generated this feature…"

start

```
start: number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

strand

```
strand: undefined | Forward | Reverse | Unknown | Unoriented
```

A GFF3 field that describes the strand of the annotation.

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

type

```
type: undefined | string
```

A GFF3 field that is supposed to be "constrained to be either: (a) a term from the "lite" sequence ontology, SOFA; or (b) a SOFA accession number." However, this is currently not enforced by SODA.

width

```
width: number
```

The width of the annotation in semantic coordinates.

Accessors

w

```
get w(): number
```

A convenience getter that returns the width property.

```
set w(w: number): void
```

A convenience setter that sets the width property.

x

```
get x(): number
```

A convenience getter that returns the start property.

```
set x(x: number): void
```

A convenience setter that sets the start property.

x2

```
get x2(): number
```

A convenience getter that returns the end property.

```
set x2(x: number): void
```

A convenience setter that sets the end property.

y

```
get y(): number
```

A convenience getter that returns the row property.

```
set y(y: number): void
```

A convenience setter that sets the row property.

4.2.12 SequenceAnnotation

```
class SequenceAnnotation
```

An Annotation class that holds position specific sequence data. For instance, this can be used to render each character in the query of a sequence alignment at the chromosome position that it was aligned to. This is pretty expensive performance-wise.

Constructors

```
(conf: SequenceAnnotationConfig): SequenceAnnotation
```

Parameters

- conf: SequenceAnnotationConfig

Properties

end

```
end: number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

row

```
row: number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

sequence

```
sequence: string
```

The sequence string to be rendered in the visualization.

start

```
start: number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

width

```
width: number
```

The width of the annotation in semantic coordinates.

Accessors

w

```
get w(): number
```

A convenience getter that returns the width property.

```
set w(w: number): void
```

A convenience setter that sets the width property.

x

```
get x(): number
```

A convenience getter that returns the start property.

```
set x(x: number): void
```

A convenience setter that sets the start property.

x2

```
get x2(): number
```

A convenience getter that returns the end property.

```
set x2(x: number): void
```

A convenience setter that sets the end property.

y

```
get y(): number
```

A convenience getter that returns the row property.

```
set y(y: number): void
```

A convenience setter that sets the row property.

4.2.13 ZoomSyncer

```
class ZoomSyncer
```

This class can be used to synchronize the zoom level across different Charts.

Constructors

```
(): ZoomSyncer
```

Properties

charts

```
charts: Chart <any> []
```

A list of Charts that the Plugin will pay attention to.

Methods

add

```
add(chart: Chart | Chart <any> []): void
```

This method registers a Chart or list of Charts with the Plugin.

Parameters

- chart: Chart | Chart <any> []

Returns: void

addChart

```
addChart(chart: Chart <any>): void
```

Add a Chart to the observer.

Parameters

- chart: Chart <any>

Returns: void

alert

```
alert(caller: Chart <any>): void
```

The ZoomZyncer alert method synchronizes all of the Transforms on each of the Charts is is observing and fires the zooming functionality.

Parameters

- caller: Chart <any>

Returns: void

4.3 Interfaces

4.3.1 AggregationConfig

```
interface AggregationConfig<A extends Annotation>
```

This defines the parameters for a call to an Annotation aggregration function.

Type parameters

- A: Annotation

Properties

annotations

```
annotations: A []
```

The list of Annotations to be aggregated.

criterion

```
criterion: (a: A, b: A): boolean
```

The comparison function that serves as the criterion for aggregation.

idPrefix

```
idPrefix: undefined | string
```

The ID prefix for each resulting AnnotationGroup. E.g. if the idPrefix "group" is supplied, the resulting groups will have IDs of the form: "group-1," "group-2," etc.

4.3.2 AnnotationConfig

```
interface AnnotationConfig
```

An interface that defines the initialization parameters for an Annotation object. For everything to work as expected, you should supply the start property and one of either end or width.

Properties

end

```
end: undefined | number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: undefined | string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

row

```
row: undefined | number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

start

```
start: undefined | number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

width

```
width: undefined | number
```

The width of the annotation in semantic coordinates.

4.3.3 AnnotationDatum

```
interface AnnotationDatum<A extends Annotation, C extends Chart>
```

An interface that simply joins an Annotation object and a Chart is has been rendered in.

Type parameters

- A: Annotation
- C: Chart

Properties

a

```
a: A
```

The Annotation object.

c

```
c: C
```

The Chart object.

4.3.4 AnnotationGenerationConfig

```
interface AnnotationGenerationConfig
```

An interface that defines the parameters for a call to the generateAnnotations function.

Properties

generationPattern

```
generationPattern: undefined | Sequential | Random
```

maxX

```
maxX: undefined | number
```

maxY

```
maxY: undefined | number
```

n

```
n: number
```

pad

```
pad: undefined | number
```

startY

```
startY: undefined | number
```

width

```
width: undefined | number
```

4.3.5 ArcConfig

```
interface ArcConfig<A extends Annotation, C extends Chart>
```

An interface that defines the parameters for a call to the arc rendering function.

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.6 BarPlotConfig

```
interface BarPlotConfig<A extends ContinuousAnnotation, C extends Chart>
```

An interface that defines the parameters for a call to the barPlot rendering function.

Type parameters

- A: ContinuousAnnotation
- C: Chart

Properties

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

barHeightFn

```
barHeightFn: undefined | (ann: A, point: None): number
```

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

domain

```
domain: undefined | None | GlyphCallback <A, C, None>
```

This defines the domain of the plot.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

range

```
range: undefined | None | GlyphCallback <A, C, None>
```

This defines the range of the plot.

rowSpan

```
rowSpan: undefined | number
```

The number of bins that the plot will span. This defaults to 1, which forces the plot to fit into one row. If an argument is supplied, it will cause the plot to grow downward. It will have no effect if a custom lineFunc is supplied.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.7 ChartConfig

```
interface ChartConfig<P extends RenderParams>
```

This describes the parameters for configuring and initializing a Chart.

Type parameters

- P: RenderParams

Properties

axisType

```
axisType: undefined | Bottom | Top
```

This controls whether or not the Chart will render a horizontal axis.

debugShading

```
debugShading: undefined | boolean
```

If this is set to true, the pad and viewport will be shaded so that they are visible in the browser.

divHeight

```
divHeight: undefined | string | number
```

The height in pixels of the Chart's containing div.

divMargin

```
divMargin: undefined | number
```

The CSS margin property for the Chart's div.

divOutline

```
divOutline: undefined | string
```

The CSS outline property for the Chart's div.

divOverflowX

```
divOverflowX: undefined | string
```

The CSS overflow-x setting of the Chart's containing div.

divOverflowY

```
divOverflowY: undefined | string
```

The CSS overflow-y setting of the Chart's containing div.

divWidth

```
divWidth: undefined | string | number
```

The width in pixels of the Chart's containing div.

domainConstraint

```
domainConstraint: undefined | (chart: Chart <P>): None
```

This constrains the Chart's domain, which in turn constrains both zoom level and panning. The parameter is a callback function that is evaluated after each zoom event to produce an interval that constrains the domain.

id

```
id: undefined | string
```

A unique identifier for the Chart. This will be generated automatically if one isn't provided.

inRender

```
inRender: undefined | (params: P): void
```

The second rendering callback function.

leftPadSize

```
leftPadSize: undefined | number
```

The number of pixels of padding on the left side of the Chart.

lowerPadSize

```
lowerPadSize: undefined | number
```

The number of pixels of padding on the bottom of the Chart.

padSize

```
padSize: undefined | number
```

The number of pixels of padding around each edge of the Chart.

postRender

```
postRender: undefined | (params: P): void
```

The final rendering callback function.

postResize

```
postResize: undefined | (): void
```

The callback function that the Chart executes after resize() is called.

postZoom

```
postZoom: undefined | (): void
```

The callback function that the Chart executes after zoom() is called.

preRender

```
preRender: undefined | (params: P): void
```

The first rendering callback function.

resizable

```
resizable: undefined | boolean
```

This controls whether or not the Chart will automatically resize itself as it's container changes size. This will cause the Chart to ignore explicit height/width arguments in the config.

rightPadSize

```
rightPadSize: undefined | number
```

The number of pixels of padding on the right side of the Chart.

rowCount

```
rowCount: undefined | number
```

The number of rows that will be rendered.

rowHeight

```
rowHeight: undefined | number
```

The height in pixels of a horizontal row in the Chart. This defaults to a value of 10.

rowStripes

```
rowStripes: undefined | boolean
```

This controls whether or not the rows will be colored in an alternating pattern.

selector

```
selector: undefined | string
```

A string that can be used to uniquely select the target DOM container.

upperPadSize

```
upperPadSize: undefined | number
```

The number of pixels of padding on the top of the Chart.

zoomConstraint

```
zoomConstraint: undefined | None
```

A Chart's contents are scaled by a scaling factor k. If a zoomConstraint of the form [min_k, max_k] is provided, the scaling factor will be constrained to that interval. This will not constrain panning.

zoomable

```
zoomable: undefined | boolean
```

This controls whether or not the Chart will be configured to allow zooming and panning.

4.3.8 ChevronGlyphConfig

```
interface ChevronGlyphConfig<A extends Annotation, C extends Chart>
```

An interface that defines the common parameters for calls to chevron glyph rendering functions.

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

chevronFillColor

```
chevronFillColor: undefined | string | GlyphCallback <A, C, string>
```

This defines the fill color of the chevron arrows.

chevronFillOpacity

```
chevronFillOpacity: undefined | number | GlyphCallback <A, C, number>
```

This defines the fill opacity of the chevron arrows.

chevronHeight

```
chevronHeight: undefined | number | GlyphCallback <A, C, number>
```

This defines the height of the chevron arrows.

chevronSpacing

```
chevronSpacing: undefined | number | GlyphCallback <A, C, number>
```

This defines the spacing between each chevron arrow.

chevronStrokeColor

```
chevronStrokeColor: undefined | string | GlyphCallback <A, C, string>
```

This defines the stroke color of the chevron arrows.

chevronStrokeOpacity

```
chevronStrokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

This defines the stroke opacity of the chevron arrows.

chevronWidth

```
chevronWidth: undefined | number | GlyphCallback <A, C, number>
```

This defines the width of the chevron arrows.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

orientation

```
orientation: undefined | Forward | Reverse | Unknown | Unoriented | GlyphCallback <A, C,␣
↪Orientation>
```

This defines the direction that the chevron arrows will point.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.9 ChevronLineConfig

```
interface ChevronLineConfig<A extends Annotation, C extends Chart>
```

An interface that defines the parameters for a call to the chevronLine rendering function.

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

chevronFillColor

```
chevronFillColor: undefined | string | GlyphCallback <A, C, string>
```

This defines the fill color of the chevron arrows.

chevronFillOpacity

```
chevronFillOpacity: undefined | number | GlyphCallback <A, C, number>
```

This defines the fill opacity of the chevron arrows.

chevronHeight

```
chevronHeight: undefined | number | GlyphCallback <A, C, number>
```

This defines the height of the chevron arrows.

chevronSpacing

```
chevronSpacing: undefined | number | GlyphCallback <A, C, number>
```

This defines the spacing between each chevron arrow.

chevronStrokeColor

```
chevronStrokeColor: undefined | string | GlyphCallback <A, C, string>
```

This defines the stroke color of the chevron arrows.

chevronStrokeOpacity

```
chevronStrokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

This defines the stroke opacity of the chevron arrows.

chevronWidth

```
chevronWidth: undefined | number | GlyphCallback <A, C, number>
```

This defines the width of the chevron arrows.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

orientation

```
orientation: undefined | Forward | Reverse | Unknown | Unoriented | GlyphCallback <A, C,␣
↪Orientation>
```

This defines the direction that the chevron arrows will point.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.10 ChevronRectangleConfig

```
interface ChevronRectangleConfig<A extends Annotation, C extends Chart>
```

An interface that defines the parameters for a call to the chevronRectangle rendering function.

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

chevronFillColor

```
chevronFillColor: undefined | string | GlyphCallback <A, C, string>
```

This defines the fill color of the chevron arrows.

chevronFillOpacity

```
chevronFillOpacity: undefined | number | GlyphCallback <A, C, number>
```

This defines the fill opacity of the chevron arrows.

chevronHeight

```
chevronHeight: undefined | number | GlyphCallback <A, C, number>
```

This defines the height of the chevron arrows.

chevronSpacing

```
chevronSpacing: undefined | number | GlyphCallback <A, C, number>
```

This defines the spacing between each chevron arrow.

chevronStrokeColor

```
chevronStrokeColor: undefined | string | GlyphCallback <A, C, string>
```

This defines the stroke color of the chevron arrows.

chevronStrokeOpacity

```
chevronStrokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

This defines the stroke opacity of the chevron arrows.

chevronWidth

```
chevronWidth: undefined | number | GlyphCallback <A, C, number>
```

This defines the width of the chevron arrows.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

orientation

```
orientation: undefined | Forward | Reverse | Unknown | Unoriented | GlyphCallback <A, C,␣
↪Orientation>
```

This defines the direction that the chevron arrows will point.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.11 ClickConfig

```
interface ClickConfig<A extends Annotation, C extends Chart>
```

An interface that defines the parameters for a call to the clickBehavior function.

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

The Annotations to which the interaction is applied.

chart

```
chart: undefined | C
```

The Chart to which the interaction is applied.

click

```
click: InteractionCallback <A, C>
```

A callback function that will be responsible for executing the click behavior. It will implicitly receive references to both a D3 Selection to the Annotation's representative glyph and the Annotation object itself.

selector

```
selector: undefined | string
```

The selector of the glyphs to which the interaction is applied.

4.3.12 ContinuousAnnotationConfig

```
interface ContinuousAnnotationConfig
```

An interface that defines the parameters for initializing a ContinuousAnnotation.

Properties

end

```
end: undefined | number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: undefined | string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

row

```
row: undefined | number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

start

```
start: undefined | number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

values

```
values: number []
```

The list of values that describe the continuous data.

width

```
width: undefined | number
```

The width of the annotation in semantic coordinates.

4.3.13 ExportConfig

```
interface ExportConfig<C extends Chart>
```

An interface that defines the parameters for a call to the exportPng function.

Type parameters

- C: Chart

Properties

chart

```
chart: C
```

The Chart to export.

filename

```
filename: undefined | string
```

The filename for the exported PNG.

pixelRatio

```
pixelRatio: undefined | number
```

The pixel ratio of the rendered PNG. Using a number larger than 1 will over-render the PNG, making it larger. Using smaller numbers currently has strange behavior, and it's not recommended.

4.3.14 FullGlyphQueryConfig

```
interface FullGlyphQueryConfig
```

Properties

chart

```
chart: Chart <any>
```

Constrain the query to Annotations rendered in this Chart.

id

```
id: string
```

Constrain the query to Annotations with this ID.

selector

```
selector: string
```

Constrain the query to Annotations with this selector.

4.3.15 Gff3AnnotationConfig

```
interface Gff3AnnotationConfig
```

An interface that defines the initialization parameters for a Gff3Annotation.

Properties

attributes

```
attributes: undefined | Map <string, string>
```

A horrifying GFF3 field that is essentially an anything goes set of key value pairs describing anything anybody every wants to add to a GFF3 record.

end

```
end: undefined | number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: undefined | string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

phase

```
phase: undefined | None | None | None
```

A GFF3 field that describes the phase for CDS (coding sequence) annotations.

row

```
row: undefined | number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

score

```
score: undefined | number
```

A GFF3 field that should describe the score of the annotation.

seqid

```
seqid: undefined | string
```

A GFF3 field: "The ID of the landmark used to establish the coordinate system for the current feature…"

source

```
source: undefined | string
```

A GFF3 field: "The source is a free text qualifier intended to describe the algorithm or operating procedure that generated this feature…"

start

```
start: undefined | number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

strand

```
strand: undefined | Forward | Reverse | Unknown | Unoriented
```

A GFF3 field that describes the strand of the annotation.

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

type

```
type: undefined | string
```

A GFF3 field that is supposed to be "constrained to be either: (a) a term from the "lite" sequence ontology, SOFA; or (b) a SOFA accession number." However, this is currently not enforced by SODA.

width

```
width: undefined | number
```

The width of the annotation in semantic coordinates.

4.3.16 Gff3Record

```
interface Gff3Record
```

An interface that describes the fields in a GFF3 record. For more information see http://gmod.org/wiki/GFF3/

Properties

attributes

```
attributes: undefined | Map <string, string>
```

A horrifying GFF3 field that is essentially an anything goes set of key value pairs describing anything anybody every wants to add to a GFF3 record.

phase

```
phase: undefined | None | None | None
```

A GFF3 field that describes the phase for CDS (coding sequence) annotations.

score

```
score: undefined | number
```

A GFF3 field that should describe the score of the annotation.

seqid

```
seqid: undefined | string
```

A GFF3 field: "The ID of the landmark used to establish the coordinate system for the current feature…"

source

```
source: undefined | string
```

A GFF3 field: "The source is a free text qualifier intended to describe the algorithm or operating procedure that generated this feature…"

strand

```
strand: undefined | Forward | Reverse | Unknown | Unoriented
```

A GFF3 field that describes the strand of the annotation.

type

```
type: undefined | string
```

A GFF3 field that is supposed to be "constrained to be either: (a) a term from the "lite" sequence ontology, SOFA; or (b) a SOFA accession number." However, this is currently not enforced by SODA.

4.3.17 GlyphConfig

```
interface GlyphConfig<A extends Annotation, C extends Chart>
```

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.18 GlyphQueryConfig

```
interface GlyphQueryConfig
```

An interface that defines the parameters for a call to the queryGlyphMap() function.

Properties

chart

```
chart: undefined | Chart <any>
```

Constrain the query to Annotations rendered in this Chart.

id

```
id: undefined | string
```

Constrain the query to Annotations with this ID.

selector

```
selector: undefined | string
```

Constrain the query to Annotations with this selector.

4.3.19 HighlightConfig

```
interface HighlightConfig
```

This describes the parameters for a call to the Chart.highlight() function.

Properties

color

```
color: undefined | string
```

The color of the highlight. This defaults to black.

end

```
end: number
```

The end of the region to be highlighted in semantic coordinates.

opacity

```
opacity: undefined | number
```

The opacity of the highlight. This defaults to 0.1.

selector

```
selector: undefined | string
```

The selector that will be applied to the highlight object in the DOM. This will be auto generated if not supplied.

start

```
start: number
```

The start of the region to be highlighted in semantic coordinates.

4.3.20 HorizontalAxisConfig

```
interface HorizontalAxisConfig<A extends Annotation, C extends Chart>
```

An interface that defines the parameters for a call to the horizontalAxis rendering function.

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

axisType

```
axisType: undefined | Bottom | Top
```

This determines whether the ticks and labels with be placed on the top or the bottom of the axis.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

domain

```
domain: undefined | None | GlyphCallback <A, C, None>
```

This defines the domain of the D3 scale used to create the axis glyph.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

fixed

```
fixed: undefined | boolean
```

If this is set to true, the axis glyph will not translate or scale during zoom events.

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

range

```
range: undefined | None | GlyphCallback <A, C, None>
```

This defines the range of the D3 scale used to create the axis glyph.

scaleToBinHeight

```
scaleToBinHeight: undefined | boolean
```

If this is set to true, the axis glyph will be forced (by stretching) into the height of a row in the Chart.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

tickSizeOuter

```
tickSizeOuter: undefined | number | GlyphCallback <A, C, number>
```

This defines the tick property that will be passed to D3's axis.tickSizeOuter function. For more information, see https://github.com/d3/d3-axis#axis_tickSizeOuter

ticks

```
ticks: undefined | number | GlyphCallback <A, C, number>
```

This defines the tick property that will be passed to D3's axis.ticks function. For more information, see https://github.com/d3/d3-axis#axis_ticks

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.21 HoverConfig

```
interface HoverConfig<A extends Annotation, C extends Chart>
```

An interface that defines the parameters for a call to the hoverBehavior function.

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

The Annotations to which the interaction is applied.

chart

```
chart: undefined | C
```

The Chart to which the interaction is applied.

mouseout

```
mouseout: InteractionCallback <A, C>
```

A callback function that will be responsible for executing the mouseout behavior. It receives a d3 selection of the glyph and the Annotation object it represents as arguments.

mouseover

```
mouseover: InteractionCallback <A, C>
```

A callback function that will be responsible for executing the mouseover behavior. It receives a d3 selection of the glyph and the Annotation object it represents as arguments.

selector

```
selector: undefined | string
```

The selector of the glyphs to which the interaction is applied.

4.3.22 LineConfig

```
interface LineConfig<A extends Annotation, C extends Chart>
```

An interface that defines the parameters for a call to the line rendering function.

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

x1

```
x1: undefined | number | GlyphCallback <A, C, number>
```

x2

```
x2: undefined | number | GlyphCallback <A, C, number>
```

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

y1

```
y1: undefined | number | GlyphCallback <A, C, number>
```

y2

```
y2: undefined | number | GlyphCallback <A, C, number>
```

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.23 LinePlotConfig

```
interface LinePlotConfig<A extends ContinuousAnnotation, C extends Chart>
```

An interface that defines the parameters for a call to the linePlot rendering function.

Type parameters

- A: ContinuousAnnotation
- C: Chart

Properties

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

domain

```
domain: undefined | None | GlyphCallback <A, C, None>
```

This defines the domain of the plot.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

lowerFillColor

```
lowerFillColor: undefined | string | GlyphCallback <A, C, string>
```

Setting this will fill the area below the plot with the color.

lowerFillOpacity

```
lowerFillOpacity: undefined | number | GlyphCallback <A, C, number>
```

This controls the opacity of the area below the plot. If no upperFillColor is supplied, setting this will trigger the lower fill in black with the supplied opacity.

pathData

```
pathData: undefined | string | GlyphCallback <A, C, string>
```

A callback that returns a string that defines the line's SVG path

range

```
range: undefined | None | GlyphCallback <A, C, None>
```

This defines the range of the plot.

rowSpan

```
rowSpan: undefined | number
```

The number of bins that the plot will span. This defaults to 1, which forces the plot to fit into one row. If an argument is supplied, it will cause the plot to grow downward. It will have no effect if a custom lineFunc is supplied.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

upperFillColor

```
upperFillColor: undefined | string | GlyphCallback <A, C, string>
```

Setting this will fill the area above the plot with the color.

upperFillOpacity

```
upperFillOpacity: undefined | number | GlyphCallback <A, C, number>
```

This controls the opacity of the area above the plot. If no upperFillColor is supplied, setting this will trigger the upper fill in black with the supplied opacity.

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.24 RectangleConfig

```
interface RectangleConfig<A extends Annotation, C extends Chart>
```

An interface that defines the parameters for a call to the rectangle rendering function.

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.25 RenderParams

```
interface RenderParams
```

This defines the parameters for a call to a Chart's rendering method.

Properties

annotations

```
annotations: undefined | Annotation []
```

The Annotation objects to be rendered.

autoLayout

```
autoLayout: undefined | boolean
```

This determines whether or not the Chart will use an automatic layout function.

end

```
end: undefined | number
```

The end coordinate of the region that will be rendered.

initializeXScale

```
initializeXScale: undefined | boolean
```

Whether or not to initialize the Chart's xScale with the range of the query.

layoutFn

```
layoutFn: undefined | (ann: Annotation []): number
```

If this is provided, the Chart will use it to define a layout for the provided annotations.

rowCount

```
rowCount: undefined | number
```

The number of rows that will be rendered.

start

```
start: undefined | number
```

The start coordinate of the region that will be rendered.

4.3.26 SequenceAnnotationConfig

```
interface SequenceAnnotationConfig
```

An interface that defines the parameters for initializing a SequenceAnnotation.

Properties

end

```
end: undefined | number
```

The end position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

id

```
id: undefined | string
```

A unique identifier for an Annotation object. If an ID is not provided, one will be automatically generated by SODA when the Annotation is instantiated.

row

```
row: undefined | number
```

This describes which horizontal row the Annotation will be rendered in a Chart, assuming that the y-positioning is not overwritten during a call to the glyph rendering API.

sequence

```
sequence: string
```

The sequence string to be rendered.

start

```
start: undefined | number
```

The start position of the annotation in semantic coordinates (generally a position on a chromosome in base pairs).

tag

```
tag: undefined | string
```

An optional tag for the annotation. This can be thought of as a something of a secondary ID.

width

```
width: undefined | number
```

The width of the annotation in semantic coordinates.

4.3.27 SequenceConfig

```
interface SequenceConfig<S extends SequenceAnnotation, C extends Chart>
```

An interface that defines the parameters for a call to the sequence rendering function.

Type parameters

- S: SequenceAnnotation
- C: Chart

Properties

annotations

```
annotations: S []
```

A list of Annotation objects that will be used to render the glyphs.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

fillColor

```
fillColor: undefined | string | GlyphCallback <S, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <S, C, number>
```

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <S, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <S, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <S, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <S, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <S, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <S, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <S, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <S, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

width

```
width: undefined | number | GlyphCallback <S, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <S, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <S, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.28 TextConfig

```
interface TextConfig<A extends Annotation, C extends Chart>
```

An interface that defines the parameters for a call to the text rendering function.

Type parameters

- A: Annotation
- C: Chart

Properties

alignmentBaseline

```
alignmentBaseline: undefined | string | GlyphCallback <A, C, string>
```

How the text glyph is aligned with it's parent. See: https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/alignment-baseline

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

fontFamily

```
fontFamily: undefined | string | GlyphCallback <A, C, string>
```

The font family that will be used. See: https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/font-family

fontSize

```
fontSize: undefined | number | GlyphCallback <A, C, number>
```

The font size of the text.

fontStyle

```
fontStyle: undefined | string | GlyphCallback <A, C, string>
```

The font style: normal, italic, or oblique. See: https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/font-style

fontWeight

```
fontWeight: undefined | string | GlyphCallback <A, C, string>
```

The weight of the font: normal, bold, bolder, lighter. See: https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/font-weight

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

textAnchor

```
textAnchor: undefined | string | GlyphCallback <A, C, string>
```

Where the text is aligned to: start, middle, or end. See: https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/text-anchor

textFn

```
textFn: (a: A, c: C): None
```

A callback to extract a list of text to display from the represented Annotation object. It is a list of text because TextGlyphs can display varying length text depending on how much room is available at the Chart's current zoom level.

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.29 TooltipConfig

```
interface TooltipConfig<A extends Annotation, C extends Chart>
```

An interface that defines the parameters for a call to the tooltip function.

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

The Annotations to which the interaction is applied.

backgroundColor

```
backgroundColor: undefined | string | GlyphCallback <A, C, string>
```

This defines the background color of the tooltip.

borderRadius

```
borderRadius: undefined | number | GlyphCallback <A, C, number>
```

This defines the border radius of the tooltip.

chart

```
chart: undefined | C
```

The Chart to which the interaction is applied.

opacity

```
opacity: undefined | number | GlyphCallback <A, C, number>
```

This defines the opacity of the tooltip.

padding

```
padding: undefined | number | GlyphCallback <A, C, number>
```

This defines the CSS padding of the tooltip.

selector

```
selector: undefined | string
```

The selector of the glyphs to which the interaction is applied.

text

```
text: GlyphProperty <A, C, string>
```

This defines the text for the tooltip.

textColor

```
textColor: undefined | string | GlyphCallback <A, C, string>
```

This defines the tooltip text color.

4.3.30 Transform

```
interface Transform
```

A re-export of d3.ZoomTransform, with the x, y, and k properties overwritten as public variables. D3 strongly advises against messing with its transform objects directly, but we actually want to do that in SODA sometimes.

Properties

k

```
k: number
```

The scaling factor described by the Transform.

x

```
x: number
```

The x translation described by the Transform.

y

```
y: number
```

The y translation described by the Transform.

4.3.31 VerticalAxisConfig

```
interface VerticalAxisConfig<A extends Annotation, C extends Chart>
```

An interface that defines the parameters for a call to the verticalAxis rendering function.

Type parameters

- A: Annotation
- C: Chart

Properties

annotations

```
annotations: A []
```

A list of Annotation objects that will be used to render the glyphs.

axisType

```
axisType: undefined | Left | Right
```

This determines whether the ticks and labels will be placed on the left or the right of the axis.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

domain

```
domain: undefined | None | GlyphCallback <A, C, None>
```

This defines the domain of the axis.

fillColor

```
fillColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the fill color of the glyph.

fillOpacity

```
fillOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel height of the glyph.

initializeFn

```
initializeFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's initialization method, which typically sets most of the style related properties from the GlyphConfig. Don't use this unless you know what you're doing.

range

```
range: undefined | None | GlyphCallback <A, C, None>
```

This defines the range of the axis.

rowSpan

```
rowSpan: undefined | number
```

The number of bins that the axis will span. This defaults to 1, which forces the axis to fit into one row. If an argument is supplied, it will cause the axis to grow downward. It will have no effect if a custom domain function is supplied.

selector

```
selector: undefined | string
```

The string that will be used to uniquely identify the call to the glyph rendering function. In the DOM, the glyphs' elements will have this assigned as an ID. If the same selector is supplied to two distinct calls to the same glyph function, the rendering results of the first call will be cleared and replaced with the results of the second.

strokeColor

```
strokeColor: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the color of the border around the glyph.

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke dash array of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray

strokeDashOffset

```
strokeDashOffset: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke dash array (if supplied) of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dashoffset

strokeLineCap

```
strokeLineCap: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the stroke linecap of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap

strokeLineJoin

```
strokeLineJoin: undefined | string | GlyphCallback <A, C, string>
```

A callback to define the offset for the stroke linejoin of the glyph. See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin

strokeOpacity

```
strokeOpacity: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the opacity of the border around the glyph.

strokeWidth

```
strokeWidth: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the width of the border around the glyph.

target

```
target: undefined | Selection <any, any, any, any> | Viewport | Overflow | Defs
```

This determines the parent DOM element in which the glyphs will be rendered. When supplying a BindTarget, the rendering function will find the appropriate parent in the supplied Chart. When supplying a D3 selection, the rendering function will explicitly use the selected element.

tickSizeOuter

```
tickSizeOuter: undefined | number | GlyphCallback <A, C, number>
```

This defines the tick property that will be passed to D3's axis.tickSizeOuter function. For more information, see https://github.com/d3/d3-axis#axis_tickSizeOuter

ticks

```
ticks: undefined | number | GlyphCallback <A, C, number>
```

This defines the tick property that will be passed to D3's axis.ticks function. For more information, see https://github.com/d3/d3-axis#axis_ticks

width

```
width: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel width of the glyph.

x

```
x: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel x coordinate of the glyph.

y

```
y: undefined | number | GlyphCallback <A, C, number>
```

A callback to define the pixel y coordinate of the glyph

zoomFn

```
zoomFn: undefined | (): void
```

A callback function that will be passed to the GlyphModifier that will manage the glyphs created with this config. If provided, this callback function will override the GlyphModifier's zoom method, which typically sets most of the positioning related properties from the GlyphConfig. Don't use this unless you know what you're doing.

4.3.32 ViewRange

```
interface ViewRange
```

This describes a range in semantic coordinates (e.g. base pairs). This will typically describe the current rendered view in a Chart.

Properties

end

```
end: number
```

The semantic end of the view.

start

```
start: number
```

The semantic start of the view.

width

```
width: number
```

The semantic width of the view.

4.4 Functions

4.4.1 aggregateIntransitive

```
function aggregateIntransitive<A extends Annotation>(config: AggregationConfig <A>): None
```

A utility function that aggregates Annotation objects into Annotation groups based off of the supplied criterion. This function assumes that your aggregation criterion is not transitive, i.e. if criterion(a, b) and criterion(b, c) evaluate to true, then criterion(a, c) doesn't necessarily evaluate to true.

Type parameters

- A: Annotation

Parameters

- config: AggregationConfig <A>

Returns: AnnotationGroup <A> []

4.4.2 aggregateTransitive

```
function aggregateTransitive<A extends Annotation>(config: AggregationConfig <A>): None
```

A utility function that aggregates Annotation objects into Annotation groups based off of the supplied criterion. This function assumes that your aggregation criterion is transitive, i.e. if criterion(a, b) and criterion(b, c) evaluate to true, then criterion(a, c) must evaluate to true.

Type parameters

- A: Annotation

Parameters

- config: AggregationConfig <A>

Returns: AnnotationGroup <A> []

4.4.3 arc

```
function arc<A extends Annotation, C extends Chart>(config: ArcConfig <A, C>): d3.
↪Selection
```

This renders a list of Annotation objects as arcs in a Chart.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: ArcConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.4.4 barPlot

```
function barPlot<A extends ContinuousAnnotation, C extends Chart>(config: BarPlotConfig
↪<A, C>): d3.Selection
```

This renders PlotAnnotations as bar plots in a Chart.

Type parameters

- A: ContinuousAnnotation
- C: Chart

Parameters

- config: BarPlotConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.4.5 chevronLine

```
function chevronLine<A extends Annotation, C extends Chart>(config: ChevronLineConfig <A,
↪ C>): d3.Selection
```

This renders Annotations as lines with chevron arrows in a Chart.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: ChevronLineConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.4.6 chevronRectangle

```
function chevronRectangle<A extends Annotation, C extends Chart>(config:␣
↪ChevronRectangleConfig <A, C>): d3.Selection
```

This renders Annotations as rectangles with chevron arrows in a Chart.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: ChevronRectangleConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.4.7 clickBehavior

```
function clickBehavior<A extends Annotation, C extends Chart>(config: ClickConfig <A, C>
↪): void
```

This applies click interactions to a list of Annotations.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: ClickConfig <A, C>

Returns: void

4.4.8 exportPng

```
function exportPng<C extends Chart>(config: ExportConfig <C>): void
```

Save the current view in a chart as a PNG image.

Type parameters

- C: Chart

Parameters

- config: ExportConfig <C>

Returns: void

4.4.9 generateAnnotations

```
function generateAnnotations(conf: AnnotationGenerationConfig): None
```

A utility function to generate some uniformly distributed Annotation objects. This is intended for testing/prototyping/playing around.

Parameters

- conf: AnnotationGenerationConfig

Returns: Annotation []

4.4.10 generateId

```
function generateId(prefix: string): string
```

Get an auto-generated string identifier of the form "<prefix>-<count>," where prefix defaults to "soda-id" and count is incremented for every call to this function. A unique count is maintained for each prefix.

Parameters

- prefix: string

Returns: string

4.4.11 generatePlotAnnotations

```
function generatePlotAnnotations(conf: AnnotationGenerationConfig): None
```

A utility function to generate some PlotAnnotation objects. This is intended for testing/prototyping/playing around.

Parameters

- conf: AnnotationGenerationConfig

Returns: ContinuousAnnotation []

4.4.12 generateSequenceAnnotations

```
function generateSequenceAnnotations(conf: AnnotationGenerationConfig): None
```

A utility function to generate some SequenceAnnotation objects. This is intended for testing/prototyping/playing around.

Parameters

- conf: AnnotationGenerationConfig

Returns: SequenceAnnotation []

4.4.13 getAllAnnotationIds

```
function getAllAnnotationIds(): None
```

This returns a list of all of the Annotation IDs that have been used to render glyphs.

Returns: string []

4.4.14 getAnnotationById

```
function getAnnotationById(id: string): Annotation
```

This function produces a reference to Annotation object that is mapped with the provided string id. It will throw an exception if the id is not in the internal map.

Parameters

- id: string

Returns: Annotation

4.4.15 getAxis

```
function getAxis(scale: d3.ScaleLinear <number, number>, axisType: AxisType): d3.Axis
```

A utility function that returns the results of the various d3 axis functions.

Parameters

- scale: d3.ScaleLinear <number, number>
- axisType: AxisType

Returns: d3.Axis <number | None>

4.4.16 getHorizontalAxisAnnotation

```
function getHorizontalAxisAnnotation(chart: Chart <any>, row: number): Annotation
```

A utility function that returns an Annotation object that is convenient to use for rendering a horizontal axis that spans a Chart's viewport.

Parameters

- chart: Chart <any>
- row: number

Returns: Annotation

4.4.17 greedyGraphLayout

```
function greedyGraphLayout<A extends Annotation>(ann: A [], tolerance: number,␣
↪vertSortFunction: (verts: string [], graph: AnnotationGraph <A>): void): number
```

This function takes a list of Annotation objects and uses a deterministic greedy graph coloring algorithm to assign each of them a y coordinate in terms of horizontal bins that will prevent any horizontal overlap when they are rendered in a Chart.

Type parameters

- A: Annotation

Parameters

- ann: A []
- tolerance: number
- vertSortFunction: (verts: string [], graph: AnnotationGraph <A>): void

Returns: number

4.4.18 heatmap

```
function heatmap<A extends ContinuousAnnotation, C extends Chart>(config: HeatmapConfig
↪<A, C>): d3.Selection
```

This renders PlotAnnotations as heatmaps in a Chart.

Type parameters

- A: ContinuousAnnotation
- C: Chart

Parameters

- config: HeatmapConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.4.19 heuristicGraphLayout

```
function heuristicGraphLayout(ann: Annotation [], nIters: number, tolerance: number):␣
↪number
```

This function takes a list of Annotation objects and uses a non-deterministic greedy graph coloring heuristic to assign each of them a y coordinate in terms of horizontal bins that will prevent any horizontal overlap when they are rendered in a Chart.

Parameters

- ann: Annotation []
- nIters: number
- tolerance: number

Returns: number

4.4.20 horizontalAxis

```
function horizontalAxis<A extends Annotation, C extends Chart>(config:␣
↪HorizontalAxisConfig <A, C>): d3.Selection
```

This renders Annotations as horizontal axes in a Chart.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: HorizontalAxisConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.4.21 hoverBehavior

```
function hoverBehavior<A extends Annotation, C extends Chart>(config: HoverConfig <A, C>
↪): void
```

This applies hover interactions to a list of Annotations.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: HoverConfig <A, C>

Returns: void

4.4.22 intervalGraphLayout

```
function intervalGraphLayout(ann: Annotation [], tolerance: number): number
```

This function takes a list of Annotation objects and uses a greedy interval scheduling algorithm to assign each of them a y coordinate in terms of horizontal bins that will prevent any horizontal overlap when they are rendered in a Chart.

Parameters

- ann: Annotation []
- tolerance: number

Returns: number

4.4.23 keyFromQueryConfig

```
function keyFromQueryConfig(config: FullGlyphQueryConfig): string
```

Parameters

- config: FullGlyphQueryConfig

Returns: string

4.4.24 line

```
function line<A extends Annotation, C extends Chart>(config: LineConfig <A, C>): d3.
↪Selection
```

This renders a list of Annotation objects as lines in a Chart.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: LineConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.4.25 linePlot

```
function linePlot<A extends ContinuousAnnotation, C extends Chart>(config:␣
↪LinePlotConfig <A, C>): d3.Selection
```

This renders PlotAnnotations as line plots in a Chart.

Type parameters

- A: ContinuousAnnotation
- C: Chart

Parameters

- config: LinePlotConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.4.26 parseBed12Record

```
function parseBed12Record(record: string): Bed12Annotation
```

A utility function to explicitly parse BED12 records. The resulting objects will have all twelve fields of the BED format.

Parameters

- record: string

Returns: Bed12Annotation

4.4.27 parseBed3Record

```
function parseBed3Record(record: string): Bed3Annotation
```

A utility function to explicitly parse BED3 records. The resulting objects will only have the first three fields of the BED format.

Parameters

- record: string

Returns: Bed3Annotation

4.4.28 parseBed6Record

```
function parseBed6Record(record: string): Bed6Annotation
```

A utility function to explicitly parse BED6 records. The resulting objects will only have the first six fields of the BED format.

Parameters

- record: string

Returns: Bed6Annotation

4.4.29 parseBed9Record

```
function parseBed9Record(record: string): Bed9Annotation
```

A utility function to explicitly parse BED9 records. The resulting objects will only have the first nine fields of the BED format.

Parameters

- record: string

Returns: Bed9Annotation

4.4.30 parseBedRecord

```
function parseBedRecord(record: string): BedAnnotation
```

A utility function to parse a general BED record. There are no guarantees about which fields end up being present in the resulting BED objects.

Parameters

- record: string

Returns: BedAnnotation

4.4.31 parseGff3Record

```
function parseGff3Record(record: string): Gff3Annotation
```

A utility function to parse a GFF3 record string. This should work in most cases, but probably does not exactly meet the GFF3 parsing standards. This function will be hardened and tested much more thoroughly in the future.

Parameters

- record: string

Returns: Gff3Annotation

4.4.32 parseOrientation

```
function parseOrientation(str: string): Orientation
```

A utility function to parse an Orientation enum from a string. For now, this is pretty basic and far from robust.

Parameters

- str: string

Returns: Orientation

4.4.33 parseRecordsFromString

```
function parseRecordsFromString<A extends Annotation>(parseFn: (record: string): A,␣
↪recordString: string, recordSeparator: RegExp): None
```

A generalized utility function to parse multiple data records from a single string into multiple Annotation objects.

Type parameters

- A: Annotation

Parameters

- parseFn: (record: string): A
- recordString: string
- recordSeparator: RegExp

Returns: A []

4.4.34 queryGlyphMap

```
function queryGlyphMap(config: GlyphQueryConfig): None
```

This function returns GlyphMappings. If all three parameters (id, selector, chart) are supplied in the config, the function will return a single D3 selection. Otherwise, the function will return a list of D3 selections.

Parameters

- config: GlyphQueryConfig

Returns: d3.Selection | d3.Selection <any, any, any, any> [] | undefined

4.4.35 rectangle

```
function rectangle<A extends Annotation, C extends Chart>(config: RectangleConfig <A, C>
↪): d3.Selection
```

This renders a list of Annotation objects as rectangles in a Chart.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: RectangleConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.4.36 removeGlyphsByQuery

```
function removeGlyphsByQuery(config: GlyphQueryConfig): void
```

Parameters

- config: GlyphQueryConfig

Returns: void

4.4.37 resolveValue

```
function resolveValue<A extends Annotation, C extends Chart, V>(property: GlyphProperty
↪<A, C, V>, d: AnnotationDatum <A, C>): V
```

A utility function that resolves the value from a GlyphProperty. If the property is a callback function, it will be called to retrieve the value. Otherwise, it will just return the value.

Type parameters

- A: Annotation
- C: Chart
- V: generic

Parameters

- property: GlyphProperty <A, C, V>
- d: AnnotationDatum <A, C>

Returns: V

4.4.38 sequence

```
function sequence<S extends SequenceAnnotation, C extends Chart>(config: SequenceConfig
↪<S, C>): d3.Selection
```

This renders a list of SequenceAnnotation objects as sequence glyphs in a Chart.

Type parameters

- S: SequenceAnnotation
- C: Chart

Parameters

- config: SequenceConfig <S, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.4.39 setKeySeparator

```
function setKeySeparator(separator: string): void
```

Set the separator that SODA uses to build map keys. The keys are of the form: <annotation ID><separator><glyph selector><separator><chart ID>.

Parameters

- separator: string

Returns: void

4.4.40 text

```
function text<A extends Annotation, C extends Chart>(config: TextConfig <A, C>): d3.
↪Selection
```

This renders a list of Annotation objects as text in a Chart.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: TextConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.4.41 tooltip

```
function tooltip<A extends Annotation, C extends Chart>(config: TooltipConfig <A, C>):␣
↪void
```

This applies tooltip interactions to a list of Annotations.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: TooltipConfig <A, C>

Returns: void

4.4.42 unmapAnnotationById

```
function unmapAnnotationById(id: string): void
```

Parameters

- id: string

Returns: void

4.4.43 verticalAxis

```
function verticalAxis<A extends Annotation, C extends Chart>(config: VerticalAxisConfig
↪<A, C>): d3.Selection
```

This renders Annotations as vertical axes in a chart. This is intended to be used in conjunction with one of the plotting glyph modules.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: VerticalAxisConfig <A, C>

Returns: d3.Selection <SVGGElement, string, any, any>

4.5 Enumerations

4.5.1 AxisType

```
enum AxisType
```

A simple enum to serve as an argument for selecting which D3 Axis function to call.

Members

Bottom

```
Bottom: = 0
```

Left

```
Left: = 2
```

Right

```
Right: = 3
```

Top

```
Top: = 1
```

4.5.2 BindTarget

```
enum BindTarget
```

An enumeration of the targets in a Chart that an Annotation can be bound to.

Members

Defs

```
Defs: = "defs"
```

The defs section, where things like patterns are supposed to go.

Overflow

```
Overflow: = "overflow"
```

The secondary viewport of a Chart in which a glyph is allowed to render outside the explicit bounds.

Viewport

```
Viewport: = "viewport"
```

The default viewport of a Chart.

4.5.3 GenerationPattern

```
enum GenerationPattern
```

Members

Random

```
Random: = "random"
```

Sequential

```
Sequential: = "sequential"
```

4.5.4 Orientation

```
enum Orientation
```

A simple enum to define strand orientation.

Members

Forward

```
Forward: = "+"
```

Represents the forward strand.

Reverse

```
Reverse: = "-"
```

Represents the reverse strand.

Unknown

```
Unknown: = "?"
```

Represents an unknown strand where strand information would be relevant (if it were known).

Unoriented

```
Unoriented: = "."
```

Represents no strand.

This is the documentation for SODA, a TypeScript/Javascript library for creating genomic annotation visualizations.

Contents