SODA

Release 1.0

Jack Roddy

Feb 16, 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

3.1 The basics

3.1.1 Creating a Chart

Charts are instantiated with a *ChartConfig* argument. Every property in a ChartConfig is optional, but you'll probably want to at least provide a selector. The selector is a string that is used as a CSS selector to locate the DOM element that the Chart will be inserted into during instantiation.

Here, we'll assume our page has a div with the id "soda-chart," and we'll instantiate a Chart that will be inserted into that div. We'll call the Chart constructor with an anonymously defined ChartConfig that has the selector property set to the string "#soda-chart."

```
let chart = new soda.Chart({
  selector: "#soda-chart",
});
```

Note: Running this code will indeed create a Chart in the selected div, but it will be completely blank. We'll see how to start to add visual content to a Chart in the next section.

3.1.2 Rendering Annotations

Once a Chart has been instantiated, we can render *Annotation objects* in it by calling the render() method. Render() takes a *RenderParams* object as an argument.

To render glyphs in a Chart, we'll first need to instantiate a list of Annotation objects. We'll cover the creation of meaningful Annotation objects later in the tutorial, but for now we'll use the *generateAnnotations utility function* to easily get some Annotation objects to play around with. Once we have the Annotations, we'll call render with them in an anonymously defined RenderParams object. By default, a Chart will render the Annotations its given as rectangles. Again, we'll go into more detail on defining customized render behaviors later in the tutorial.

```
let chart = new soda.Chart({
  selector: "#soda-chart"
});

let ann = soda.generateAnnotations({
  n: 10
})
```

(continues on next page)

(continued from previous page)

```
chart.render({
  annotations: ann
})
```

3.1.3 Adding an axis

When instantiated with a ChartConfig that has the axis property set to true, a Chart will render a horizontal axis above the Chart. The axis will be placed in the Chart's overflow viewport. This allows it to be rendered in the pad above the Chart's first row. If you want to place a non-default axis in your chart, check out the tutorial on axis glyphs.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.1.4 Changing the row height

The height of the rows in a Chart can be controlled by setting the rowHeight property on the ChartConfig.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true
  rowHeight: 20,
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.1.5 Changing the pad size

Charts have a configurable white space padding around their viewports that can be controlled by setting the padSize property in the ChartConfig.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  padSize: 100
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.1.6 Enabling zooming

Charts can optionally be configured to enable realtime zooming and panning. To do this, set the zoomable property to true on the ChartConfig. A Chart can be panned by clicking and dragging or zoomed with ctrl + scrolling.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.1.7 Constraining the zoom level

The zoom level on a Chart that has zooming enabled can be constrained using the scaleExtent property on the ChartConfig. The scaleExtent property is a tuple of numbers of the form [min $_k$, max $_k$], where min $_k$ controls how far a Chart can be zoomed out and max $_k$ controls how far a Chart can be zoomed in.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
```

(continues on next page)

(continued from previous page)

```
  scaleExtent: [0.5, 10]
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.1.8 Enabling resizing

Charts can be configured to automatically resize themselves to fit into their DOM container. To enable resizing, set the resizable property to true in the ChartConfig.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  resizable: true
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

Note: To see resizing in action here, you'll want to view this example in CodePen or press the 0.5x or 0.25x button in the embed.

3.1.9 Enabling row stripes

The rows in a Chart can optionally be striped with alternating colors. To enable this, set the rowStripes property to true on the ChartConfig.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  resizable: true,
  rowStripes: true
});
```

(continues on next page)

(continued from previous page)

```
let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.1.10 Setting an explicit render range

When a Chart's render() method is called, the initial render range can be specified in the RenderParams using the start and end properties.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  resizable: true,
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann,
  start: 0,
  end: 2000,
})
```

This section of the tutorial covers the basics of what makes up a SODA visualization and some higher level customizations that can be made.

Contents:

3.2 Custom rendering

3.2.1 Overriding the default Chart rendering routine - colored rectangles

To change the way a Chart renders annotations, we need to override the default rendering routine by providing a callback function that will be assigned to the Chart's inRender property. The callback function has two parameters:

- this: a reference to the Chart itself
- params: a reference to the *RenderParams* argument when render() is called

In the canonical SODA pattern, the inRender callback should be responsible for making calls to SODA's glyph rendering API.

For example, the default Chart inRender callback is defined as:

```
let defaultInRender = function (
  this: soda.Chart<any>,
  params: soda.RenderParams
): void {
  soda.rectangle({
    chart: this,
    annotations: params.annotations || [],
  })
}
```

If we wanted to change the colors of our rectangles, we might define something like:

```
let customInRender = function (this, params): void {
  soda.rectangle({
    chart: this,
    annotations: params.annotations || [],
    fillColor: "cadetblue",
    strokeColor: "cadetblue",
  })
}
```

We can supply the callback on the *ChartConfig*:

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  inRender: customInRender
});
```

Alternatively, we can set the Chart's inRender property after instantiation:

```
let chart = new soda.Chart({
  selector: "#soda-chart"
});

chart.inRender = customInRender
```

Finally, here's a full example that anonymously defines the inRender callback in the ChartConfig:

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  inRender: function (this, params): void {
    soda.rectangle({
      chart: this,
      annotations: params.annotations || [],
      fillColor: "cadetblue",
      strokeColor: "cadetblue",
    })
  }
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.2.2 Coloring rectangles using callbacks

In the previous example, we saw how to override the Chart's default rendering routine to draw colored rectangles. Instead, you may want to dynamically color your glyphs according to some property of the data they represent. We can achieve this by providing a *GlyphCallback* as an argument to the glyph rendering API in the place of a static value.

GlyphCallbacks are callback functions that are supplied with an *AnnotationDatum* argument while returning a value that is appropriate for setting a glyph styling property. An AnnotationDatum is a simple object that contains references to an individual Annotation object and the Chart that it has (or will be) rendered in. The rendering API will evaluate GlyphCallbacks for each Annotation object supplied, resulting in dynamic styling.

In this example, we'll define some callbacks that set the fill and stroke colors of our rectangle glyphs depending onn an Annotation's row property: Annotations with even numbered row properties will be colored blue, while Annotations with odd numbered rows will be colored purple.

A GlyphCallback definition with explicit type parameters looks something like:

```
let colors = ['cadetblue', '#9581a9']

let colorCallback = (
  //type parameters for clarity
  d: AnnotationDatum<
    soda.Annotation,
    soda.Chart,
    string>
) => colors[d.a.row % 2]
```

With our coloring GlyphCallback defined, we can do something like:

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  inRender: function (this, params): void {
    soda.rectangle({
      chart: this,
      annotations: params.annotations || [],
      fillColor: colorCallback,
      strokeColor: colorCallback,
    })
  }
});
```

Finally, here's a full example that sets the fill and stroke colors differently:

```
let colors = ['cadetblue', '#9581a9']

let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  inRender: function (this, params): void {
    soda.rectangle({
      chart: this,
      annotations: params.annotations || [],
      fillColor: (d) => colors[d.a.row % 2],
      strokeColor: (d) => colors[d.a.row % 1],
    })
  }
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.2.3 Lines

We can also render Annotations as lines.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  inRender: function (this, params): void {
    soda.line({
```

(continues on next page)

(continued from previous page)

```
      chart: this,
      annotations: params.annotations || [],
      strokeColor: 'cadetblue'
    })
  }
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.2.4 Arcs

We can also render Annotations as arcs.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  inRender: function (this, params): void {
    soda.arc({
      chart: this,
      annotations: params.annotations || [],
      strokeColor: 'cadetblue'
    })
  }
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.2.5 Chevron lines

We can also render Annotations as lines with chevron arrows on them.

```
let orientations = [soda.Orientation.Forward, soda.Orientation.Reverse]

let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  inRender: function (this, params): void {
    soda.chevronLine({
      chart: this,
      annotations: params.annotations || [],
      chevronSpacing: 10,
      strokeColor: 'cadetblue',
      chevronStrokeColor: 'cadetblue',
      orientation: (d) => orientations[d.a.y % 2]
    })
  }
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.2.6 Chevron rectangles

We can also render Annotations as rectangles with chevron arrows inside them.

```
let orientations = [soda.Orientation.Forward, soda.Orientation.Reverse]

let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  inRender: function (this, params): void {
    soda.chevronRectangle({
      chart: this,
      annotations: params.annotations || [],
      chevronSpacing: 5,
      strokeColor: 'cadetblue',
      chevronStrokeColor: 'cadetblue',
      orientation: (d) => orientations[d.a.y % 2]
    })
  }
});
```

(continues on next page)

(continued from previous page)

```
let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.2.7 Text

We can also render Annotations as text. This is a little more complicated because we have to define a callback function that returns a list of text.

```
let colors = ['cadetblue', '#9581a9']

let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  rowHeight: 18,
  inRender: function (this, params): void {
    soda.text({
      chart: this,
      annotations: params.annotations || [],
      textFn: (a) => {
        return [
          `${a.id}: ${a.start} - ${a.end} | ${a.row}`,
          `${a.id}: ${a.start} - ${a.end}`,
          `${a.id}`,
          "...",
        ];
      },
      fillColor: (d) => colors[d.a.row % 2],
    })
  }
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

This section showcases some of SODA's lower level customization features.

Contents:

- *Overriding the default Chart rendering routine - colored rectangles*
- *Coloring rectangles using callbacks*
- *Lines*
- *Arcs*
- *Chevron lines*
- *Chevron rectangles*
- *Text*

3.3 Vertical layout

3.3.1 Automatic layout

In all of the examples in the tutorial, we've been getting some toy annotation data using the *generateAnnotations* function, which by default returns a list of Annotations uniformly distributed in a preconfigured layout. Here, we'll provide a *GenerationPattern*.Random argument, which will give us randomly distributed Annotations with a row property of 0.

```
let ann: Soda.Annotation = soda.generateAnnotations({
  n: 100,
  generationPattern: soda.GenerationPattern.Random
})
```

If we were to render these annotations, they would all be stacked on top of each other in the first row:

If we don't have a preconfigured layout, a *Chart* can be configured to automatically calculate one for us. By setting the autoLayout property to true on our *RenderParams*, the annotations provided will be laid out in a condense, non-overlapping pattern.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 100,
  generationPattern: soda.GenerationPattern.Random
})

chart.render({
  annotations: ann,
  autoLayout: true
})
```

3.3.2 Explicitly calling a SODA layout function

We can also explicitly call one of SODA's layout functions on a list of Annotations.

```
let ann: Soda.Annotation = soda.generateAnnotations({
  n: 100,
  generationPattern: soda.GenerationPattern.Random
})

// this applies a layout directly to the Annotation
// objects by modifying their row properties
soda.intervalGraphLayout(ann);
```

3.3.3 Comparing SODA layout functions

Currently, SODA has three layout functions, each of which will produce a layout with slightly different properties:

- *intervalGraphLayout*
- *greedyGraphLayout*
- *heuristicGraphLayout*

The following CodePen examples show the application of each layout function on some random Annotations.

Interval layout

This uses an optimal interval scheduling algorithm, and it tends to look neat and orderly.

Greedy layout

This uses a greedy graph coloring algorithm, and it will tend to place larger annotations towards the top.

Heuristic layout

This uses a heuristic graph coloring algorithm, and it tends to look somewhat random.

3.3.4 Externally defining a layout

If you don't want to use a SODA layout function, you're free to define a layout by adjusting the row properties of your *Annotation* objects.

For example, if we wanted to force every glyph into its own row, we might do something like this:

```
let ann: Soda.Annotation = soda.generateAnnotations({
  n: 100,
})
```

(continues on next page)

(continued from previous page)

```
let row = 0;
for (const a of ann) {
  a.row = row++;
}
```

This section explains how to use SODA's builtin layout optimization functions to efficiently visualize your data in a compact, non-overlaping way.

Contents:

3.4 Using common Bioinformatics data formats

3.4.1 Rendering BED data

The BED format is one of the annotation data formats that has become a de facto standard in Bioinformatics. SODA supports parsing BED records as strings into extensions of the *Annotation object*.

Note: There are a handful of BED specifications, and SODA has several corresponding parsing functions and Annotation objects. Simply put, the number after the word "Bed" describes the number of fields in the corresponding BED records/objects. The final pair listed here is for generic BED parsing, in which the first 3 fields are required and the last 9 are optional.

SODA's BED parsing functions and their return types:

- *parseBed3Record* -> *Bed3Annotation*
- *parseBed6Record* -> *Bed6Annotation*
- *parseBed9Record* -> *Bed9Annotation*
- *parseBed12Record* -> *Bed12Annotation*
- *parseBedRecord* -> *BedAnnotation*

In this example, we'll render some toy BED9 data.

To make our lives a little easier and keep the TypeScript compiler happy, we'll first define an extension of the *RenderParams* interface. The key here is that we are overriding the type of the annotations property with Bed9Annotation[], which will propagate through the inner workings of SODA up to where we might define an override to a rendering function.

```
interface BedExampleRenderParams extends soda.RenderParams {
  annotations: soda.Bed9Annotation[]
}
```

Now, let's instantiate a Chart, but this time we'll explicitly supply our extended BedExampleRenderParams as a type parameter.

```
// Chart has a type parameter defined as P extends RenderParams
let chart = new soda.Chart<BedExampleRenderParams>({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  inRender: function (
    // explicit types for clarity here, but the compiler should actually
    // be able to infer these types automatically
    this: Chart<BedExampleRenderParams>,
    params: BedExampleRenderParams
  ): void {
    soda.rectangle({
      chart: this,
      annotations: params.annotations,
      // the TypeScript compiler won't complain about us using the itemRgb
      // property because of the type argument we supplied above
      fillColor: (d) => `rgb(${d.a.itemRgb})`,
      strokeColor: "black"
    })
  }
});
```

Now that our Chart is configured to render our BED data, let's get some data for it to render. Here, we'll just define a string inline that holds some BED records, parse it into Annotation objects, and render it.

```
// these BED data are from a toy example at https://genome.ucsc.edu/FAQ/FAQformat.html
↪#format1
let bedData = `
chr7    127471196  127472363  Pos1  0  +  127471196  127472363  255,0,0
chr7    127472363  127473530  Pos2  0  +  127472363  127473530  255,0,0
chr7    127473530  127474697  Pos3  0  +  127473530  127474697  255,0,0
chr7    127474697  127475864  Pos4  0  +  127474697  127475864  255,0,0
chr7    127475864  127477031  Neg1  0  -  127475864  127477031  0,0,255
chr7    127477031  127478198  Neg2  0  -  127477031  127478198  0,0,255
chr7    127478198  127479365  Neg3  0  -  127478198  127479365  0,0,255
chr7    127479365  127480532  Pos5  0  +  127479365  127480532  255,0,0
chr7    127480532  127481699  Neg4  0  -  127480532  127481699  0,0,255
`

let ann: soda.Bed9Annotation[] = soda.parseRecordsFromString(
  soda.parseBed9Record,
  bedData
);

chart.render({
  annotations: ann
})
```

Finally, here's a full example that omits explicit type annotations to be a bit more concise:

```
interface BedExampleRenderParams extends soda.RenderParams {
  annotations: soda.Bed9Annotation[]
}

let chart = new soda.Chart<BedExampleRenderParams>({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  inRender: function (this, params): void {
    soda.rectangle({
      chart: this,
      annotations: params.annotations,
      fillColor: (d) => `rgb(${d.a.itemRgb})`,
      strokeColor: "black"
    })
  }
});

// these BED data are from a toy example at https://genome.ucsc.edu/FAQ/FAQformat.html
↪#format1
let bedData = `
chr7    127471196  127472363  Pos1  0  +  127471196  127472363  255,0,0
chr7    127472363  127473530  Pos2  0  +  127472363  127473530  255,0,0
chr7    127473530  127474697  Pos3  0  +  127473530  127474697  255,0,0
chr7    127474697  127475864  Pos4  0  +  127474697  127475864  255,0,0
chr7    127475864  127477031  Neg1  0  -  127475864  127477031  0,0,255
chr7    127477031  127478198  Neg2  0  -  127477031  127478198  0,0,255
chr7    127478198  127479365  Neg3  0  -  127478198  127479365  0,0,255
chr7    127479365  127480532  Pos5  0  +  127479365  127480532  255,0,0
chr7    127480532  127481699  Neg4  0  -  127480532  127481699  0,0,255
`

let ann = soda.parseRecordsFromString(
  soda.parseBed9Record,
  bedData
);

chart.render({
  annotations: ann
})
```

3.4.2 Rendering GFF3 data

The GFF3 format is another annotation data format that is perhaps a bit less common than the BED format. SODA supports parsing GFF3 records as strings into extensions of the *Annotation object*.

SODA's GFF3 parsing function and its return type:

- parseGff33Record -> *Gff3Annotation*

Note: The GFF3 format is quite complicated and, in some sense, open-ended. Your mileage may vary. If you have an

issue with SODA's GFF3 parsing, we're definitely interested in hearing about it.

In this example we'll render some real GFF3 data that comes from Gencode.

We'll start off by creating an extension to *RenderParams*, and a custom inRender() implementation.

```
import * as d3 from 'https://cdn.skypack.dev/d3';

interface Gff3ExampleRenderParams extends soda.RenderParams {
  annotations: soda.Gff3Annotation[];
}

let chart = new soda.Chart<Gff3ExampleRenderParams>({
  selector: "#soda-chart",
  rowHeight: 14,
  axis: true,
  zoomable: true,
  resizable: true,
  inRender: function (this, params: Gff3ExampleRenderParams): void {
    let colorScale = d3.scaleOrdinal(d3.schemeTableau10);

    soda.chevronRectangle({
      chart: this,
      annotations: params.annotations,
      orientation: (d) => d.a.strand!,
      // this would be kind of dangerous in production code, but we're using
      // the ! operator here since we know we can get this attribute
      fillColor: (d) => colorScale(d.a.attributes!.get("Parent")!),
      strokeColor: "black",
    })
  }
});
```

Now that our Chart is configured to render our GFF3 data, let's get some data for it to render. Here, we'll just define a string inline that holds some GFF3 records and parse it into Annotation objects. These data describe genes, transcripts and exons.

```
// these data come from Gencode
// compare roughly to the gene track at https://uswest.ensembl.org/Homo_sapiens/Location/
↪View?r=1:11000-36000;db=core
let gff3Data = `
chr1        HAVANA  gene    11869   14409   .       +       .       ID=ENSG00000223972.5;
↪gene_id=ENSG00000223972.5;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;level=2;hgnc_id=HGNC:37102;havana_gene=OTTHUMG00000000961.2
chr1        HAVANA  transcript      11869   14409   .       +       .
↪ID=ENST00000456328.2;Parent=ENSG00000223972.5;gene_id=ENSG00000223972.5;transcript_
↪id=ENST00000456328.2;gene_type=transcribed_unprocessed_pseudogene;gene_name=DDX11L1;
↪transcript_type=processed_transcript;transcript_name=DDX11L1-202;level=2;transcript_
↪support_level=1;hgnc_id=HGNC:37102;tag=basic;havana_gene=OTTHUMG00000000961.2;havana_
↪transcript=OTTHUMT00000362751.1
chr1        HAVANA  exon    11869   12227   .       +       .
↪ID=exon:ENST00000456328.2:1;Parent=ENST00000456328.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000456328.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=processed_transcript;transcript_name=DDX11L1-202;exon_
↪number=1;exon_id=ENSE00002234944.1;level=2;transcript_support_level=1;hgnc_
↪id=HGNC:37102;tag=basic;havana_gene=OTTHUMG00000000961.2;havana_
↪transcript=OTTHUMT00000362751.1
```

(continues on next page)

(continued from previous page)

```
chr1	HAVANA	exon	12613	12721	.	+	.	␣
↪ID=exon:ENST00000456328.2:2;Parent=ENST00000456328.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000456328.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=processed_transcript;transcript_name=DDX11L1-202;exon_
↪number=2;exon_id=ENSE00003582793.1;level=2;transcript_support_level=1;hgnc_
↪id=HGNC:37102;tag=basic;havana_gene=OTTHUMG00000000961.2;havana_
↪transcript=OTTHUMT00000362751.1
chr1	HAVANA	exon	13221	14409	.	+	.	␣
↪ID=exon:ENST00000456328.2:3;Parent=ENST00000456328.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000456328.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=processed_transcript;transcript_name=DDX11L1-202;exon_
↪number=3;exon_id=ENSE00002312635.1;level=2;transcript_support_level=1;hgnc_
↪id=HGNC:37102;tag=basic;havana_gene=OTTHUMG00000000961.2;havana_
↪transcript=OTTHUMT00000362751.1
chr1	HAVANA	transcript	12010	13670	.	+	.	␣
↪ID=ENST00000450305.2;Parent=ENSG00000223972.5;gene_id=ENSG00000223972.5;transcript_
↪id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_name=DDX11L1;
↪transcript_type=transcribed_unprocessed_pseudogene;transcript_name=DDX11L1-201;level=2;
↪transcript_support_level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;
↪havana_gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	exon	12010	12057	.	+	.	␣
↪ID=exon:ENST00000450305.2:1;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=1;exon_id=ENSE00001948541.1;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	exon	12179	12227	.	+	.	␣
↪ID=exon:ENST00000450305.2:2;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=2;exon_id=ENSE00001671638.2;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	exon	12613	12697	.	+	.	␣
↪ID=exon:ENST00000450305.2:3;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=3;exon_id=ENSE00001758273.2;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	exon	12975	13052	.	+	.	␣
↪ID=exon:ENST00000450305.2:4;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=4;exon_id=ENSE00001799933.2;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	exon	13221	13374	.	+	.	␣
↪ID=exon:ENST00000450305.2:5;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=5;exon_id=ENSE00001746346.2;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
```

(continues on next page)

(continued from previous page)

```
chr1	HAVANA	exon	13453	13670	.	+	.	␣
↪ID=exon:ENST00000450305.2:6;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=6;exon_id=ENSE00001863096.1;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	gene	14404	29570	.	-	.	ID=ENSG00000227232.5;
↪gene_id=ENSG00000227232.5;gene_type=unprocessed_pseudogene;gene_name=WASH7P;level=2;
↪hgnc_id=HGNC:38034;havana_gene=OTTHUMG00000000958.1
chr1	HAVANA	transcript	14404	29570	.	-	.	␣
↪ID=ENST00000488147.1;Parent=ENSG00000227232.5;gene_id=ENSG00000227232.5;transcript_
↪id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;transcript_
↪type=unprocessed_pseudogene;transcript_name=WASH7P-201;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:38034;ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;
↪havana_transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	29534	29570	.	-	.	␣
↪ID=exon:ENST00000488147.1:1;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=1;exon_
↪id=ENSE00001890219.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	24738	24891	.	-	.	␣
↪ID=exon:ENST00000488147.1:2;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=2;exon_
↪id=ENSE00003507205.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	18268	18366	.	-	.	␣
↪ID=exon:ENST00000488147.1:3;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=3;exon_
↪id=ENSE00003477500.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	17915	18061	.	-	.	␣
↪ID=exon:ENST00000488147.1:4;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=4;exon_
↪id=ENSE00003565697.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	17606	17742	.	-	.	␣
↪ID=exon:ENST00000488147.1:5;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=5;exon_
↪id=ENSE00003475637.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	17233	17368	.	-	.	␣
↪ID=exon:ENST00000488147.1:6;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=6;exon_
↪id=ENSE00003502542.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
```

(continues on next page)

(continued from previous page)

```
chr1	HAVANA	exon	16858	17055	.	-	.	␣
↪ID=exon:ENST00000488147.1:7;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=7;exon_
↪id=ENSE00003553898.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	16607	16765	.	-	.	␣
↪ID=exon:ENST00000488147.1:8;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=8;exon_
↪id=ENSE00003621279.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	15796	15947	.	-	.	␣
↪ID=exon:ENST00000488147.1:9;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=9;exon_
↪id=ENSE00002030414.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	15005	15038	.	-	.	␣
↪ID=exon:ENST00000488147.1:10;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=10;exon_
↪id=ENSE00001935574.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	14404	14501	.	-	.	␣
↪ID=exon:ENST00000488147.1:11;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=11;exon_
↪id=ENSE00001843071.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	ENSEMBL	gene	17369	17436	.	-	.	ID=ENSG00000278267.1;
↪gene_id=ENSG00000278267.1;gene_type=miRNA;gene_name=MIR6859-1;level=3;hgnc_
↪id=HGNC:50039
chr1	ENSEMBL	transcript	17369	17436	.	-	.	␣
↪ID=ENST00000619216.1;Parent=ENSG00000278267.1;gene_id=ENSG00000278267.1;transcript_
↪id=ENST00000619216.1;gene_type=miRNA;gene_name=MIR6859-1;transcript_type=miRNA;
↪transcript_name=MIR6859-1-201;level=3;transcript_support_level=NA;hgnc_id=HGNC:50039;
↪tag=basic
chr1	ENSEMBL	exon	17369	17436	.	-	.	␣
↪ID=exon:ENST00000619216.1:1;Parent=ENST00000619216.1;gene_id=ENSG00000278267.1;
↪transcript_id=ENST00000619216.1;gene_type=miRNA;gene_name=MIR6859-1;transcript_
↪type=miRNA;transcript_name=MIR6859-1-201;exon_number=1;exon_id=ENSE00003746039.1;
↪level=3;transcript_support_level=NA;hgnc_id=HGNC:50039;tag=basic
chr1	HAVANA	gene	29554	31109	.	+	.	ID=ENSG00000243485.5;
↪gene_id=ENSG00000243485.5;gene_type=lncRNA;gene_name=MIR1302-2HG;level=2;hgnc_
↪id=HGNC:52482;tag=ncRNA_host;havana_gene=OTTHUMG00000000959.2
chr1	HAVANA	transcript	29554	31097	.	+	.	␣
↪ID=ENST00000473358.1;Parent=ENSG00000243485.5;gene_id=ENSG00000243485.5;transcript_
↪id=ENST00000473358.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_type=lncRNA;
↪transcript_name=MIR1302-2HG-202;level=2;transcript_support_level=5;hgnc_id=HGNC:52482;
↪tag=not_best_in_genome_evidence,dotter_confirmed,basic;havana_gene=OTTHUMG00000000959.
↪2;havana_transcript=OTTHUMT00000002840.1
```

(continues on next page)

(continued from previous page)

```
chr1	HAVANA	exon	29554	30039	.	+	.	␣
↪ID=exon:ENST00000473358.1:1;Parent=ENST00000473358.1;gene_id=ENSG00000243485.5;
↪transcript_id=ENST00000473358.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_
↪type=lncRNA;transcript_name=MIR1302-2HG-202;exon_number=1;exon_id=ENSE00001947070.1;
↪level=2;transcript_support_level=5;hgnc_id=HGNC:52482;tag=not_best_in_genome_evidence,
↪dotter_confirmed,basic;havana_gene=OTTHUMG00000000959.2;havana_
↪transcript=OTTHUMT00000002840.1
chr1	HAVANA	exon	30564	30667	.	+	.	␣
↪ID=exon:ENST00000473358.1:2;Parent=ENST00000473358.1;gene_id=ENSG00000243485.5;
↪transcript_id=ENST00000473358.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_
↪type=lncRNA;transcript_name=MIR1302-2HG-202;exon_number=2;exon_id=ENSE00001922571.1;
↪level=2;transcript_support_level=5;hgnc_id=HGNC:52482;tag=not_best_in_genome_evidence,
↪dotter_confirmed,basic;havana_gene=OTTHUMG00000000959.2;havana_
↪transcript=OTTHUMT00000002840.1
chr1	HAVANA	exon	30976	31097	.	+	.	␣
↪ID=exon:ENST00000473358.1:3;Parent=ENST00000473358.1;gene_id=ENSG00000243485.5;
↪transcript_id=ENST00000473358.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_
↪type=lncRNA;transcript_name=MIR1302-2HG-202;exon_number=3;exon_id=ENSE00001827679.1;
↪level=2;transcript_support_level=5;hgnc_id=HGNC:52482;tag=not_best_in_genome_evidence,
↪dotter_confirmed,basic;havana_gene=OTTHUMG00000000959.2;havana_
↪transcript=OTTHUMT00000002840.1
chr1	HAVANA	transcript	30267	31109	.	+	.	␣
↪ID=ENST00000469289.1;Parent=ENSG00000243485.5;gene_id=ENSG00000243485.5;transcript_
↪id=ENST00000469289.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_type=lncRNA;
↪transcript_name=MIR1302-2HG-201;level=2;transcript_support_level=5;hgnc_id=HGNC:52482;
↪tag=not_best_in_genome_evidence,basic;havana_gene=OTTHUMG00000000959.2;havana_
↪transcript=OTTHUMT00000002841.2
chr1	HAVANA	exon	30267	30667	.	+	.	␣
↪ID=exon:ENST00000469289.1:1;Parent=ENST00000469289.1;gene_id=ENSG00000243485.5;
↪transcript_id=ENST00000469289.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_
↪type=lncRNA;transcript_name=MIR1302-2HG-201;exon_number=1;exon_id=ENSE00001841699.1;
↪level=2;transcript_support_level=5;hgnc_id=HGNC:52482;tag=not_best_in_genome_evidence,
↪basic;havana_gene=OTTHUMG00000000959.2;havana_transcript=OTTHUMT00000002841.2
chr1	HAVANA	exon	30976	31109	.	+	.	␣
↪ID=exon:ENST00000469289.1:2;Parent=ENST00000469289.1;gene_id=ENSG00000243485.5;
↪transcript_id=ENST00000469289.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_
↪type=lncRNA;transcript_name=MIR1302-2HG-201;exon_number=2;exon_id=ENSE00001890064.1;
↪level=2;transcript_support_level=5;hgnc_id=HGNC:52482;tag=not_best_in_genome_evidence,
↪basic;havana_gene=OTTHUMG00000000959.2;havana_transcript=OTTHUMT00000002841.2
chr1	ENSEMBL	gene	30366	30503	.	+	.	ID=ENSG00000284332.1;
↪gene_id=ENSG00000284332.1;gene_type=miRNA;gene_name=MIR1302-2;level=3;hgnc_
↪id=HGNC:35294
chr1	ENSEMBL	transcript	30366	30503	.	+	.	␣
↪ID=ENST00000607096.1;Parent=ENSG00000284332.1;gene_id=ENSG00000284332.1;transcript_
↪id=ENST00000607096.1;gene_type=miRNA;gene_name=MIR1302-2;transcript_type=miRNA;
↪transcript_name=MIR1302-2-201;level=3;transcript_support_level=NA;hgnc_id=HGNC:35294;
↪tag=basic
chr1	ENSEMBL	exon	30366	30503	.	+	.	␣
↪ID=exon:ENST00000607096.1:1;Parent=ENST00000607096.1;gene_id=ENSG00000284332.1;
↪transcript_id=ENST00000607096.1;gene_type=miRNA;gene_name=MIR1302-2;transcript_
↪type=miRNA;transcript_name=MIR1302-2-201;exon_number=1;exon_id=ENSE00003695741.1;
↪level=3;transcript_support_level=NA;hgnc_id=HGNC:35294;tag=basic
```

(continues on next page)

(continued from previous page)

```
chr1	HAVANA	gene	34554	36081	.	-	.	ID=ENSG00000237613.2;
↪gene_id=ENSG00000237613.2;gene_type=lncRNA;gene_name=FAM138A;level=2;hgnc_
↪id=HGNC:32334;havana_gene=OTTHUMG00000000960.1
chr1	HAVANA	transcript	34554	36081	.	-	.	␣
↪ID=ENST00000417324.1;Parent=ENSG00000237613.2;gene_id=ENSG00000237613.2;transcript_
↪id=ENST00000417324.1;gene_type=lncRNA;gene_name=FAM138A;transcript_type=lncRNA;
↪transcript_name=FAM138A-201;level=2;transcript_support_level=1;hgnc_id=HGNC:32334;
↪tag=basic;havana_gene=OTTHUMG00000000960.1;havana_transcript=OTTHUMT00000002842.1
chr1	HAVANA	exon	35721	36081	.	-	.	␣
↪ID=exon:ENST00000417324.1:1;Parent=ENST00000417324.1;gene_id=ENSG00000237613.2;
↪transcript_id=ENST00000417324.1;gene_type=lncRNA;gene_name=FAM138A;transcript_
↪type=lncRNA;transcript_name=FAM138A-201;exon_number=1;exon_id=ENSE00001656588.1;
↪level=2;transcript_support_level=1;hgnc_id=HGNC:32334;tag=basic;havana_
↪gene=OTTHUMG00000000960.1;havana_transcript=OTTHUMT00000002842.1
chr1	HAVANA	exon	35277	35481	.	-	.	␣
↪ID=exon:ENST00000417324.1:2;Parent=ENST00000417324.1;gene_id=ENSG00000237613.2;
↪transcript_id=ENST00000417324.1;gene_type=lncRNA;gene_name=FAM138A;transcript_
↪type=lncRNA;transcript_name=FAM138A-201;exon_number=2;exon_id=ENSE00001669267.1;
↪level=2;transcript_support_level=1;hgnc_id=HGNC:32334;tag=basic;havana_
↪gene=OTTHUMG00000000960.1;havana_transcript=OTTHUMT00000002842.1
`;

let ann: soda.Gff3Annotation[] = soda.parseRecordsFromString(
  soda.parseGff3Record,
  gff3Data
);
```

Now that we have some Gff3Annotation objects, we're going to filter out everything but the exons.

```
let exonAnn = ann.filter((a) => a.type == "exon");
```

With our exon annotations in hand, say we want to logically group them by the transcript they belong to. To accomplish that, we can use *aggregateTransitive*.

In these data, each record has a Parent field in the attributes property. In a call to the aggregation function, we'll supply a criterion callback that checks if two Annotations have the same parent. The function will return a list of AnnotationGroups, each oone containing a group of exons that belongs to a particular transcript.

```
let groups: soda.AnnotationGroup<soda.Gff3Annotation>[] =
  soda.aggregateTransitive({
    annotations: exonAnn,
    criterion: (a: soda.Gff3Annotation, b: soda.Gff3Annotation) =>
      // this would be kind of dangerous in production code, but we're using
      // the ! operator here since we know we can get these attributes
      a.attributes!.get("Parent")! == b.attributes!.get("Parent"!),
  });
```

Now, say we want to make sure our exon groups are drawn in the same row in our Chart. If we pass AnnotationGroups into one of SODA's layout functions, it will ensure that every member of a group is assigned to the same row.

After running the AnnotationGroups through the layout function, we can render the Annotation objects they contained directly.

```
chart.render({
  annotations: exonAnn,
});
```

Finally, here's a full example that omits explicit type annotations to be a bit more concise:

```
import * as d3 from 'https://cdn.skypack.dev/d3';

interface Gff3ExampleRenderParams extends soda.RenderParams {
  annotations: soda.Gff3Annotation[];
}

let chart = new soda.Chart<Gff3ExampleRenderParams>({
  selector: "#soda-chart",
  rowHeight: 14,
  axis: true,
  zoomable: true,
  resizable: true,
  inRender: function (this, params: Gff3ExampleRenderParams): void {
    let colorScale = d3.scaleOrdinal(d3.schemeTableau10);

    soda.chevronRectangle({
      chart: this,
      annotations: params.annotations,
      orientation: (d) => d.a.strand!,
      fillColor: (d) => colorScale(d.a.attributes!.get("Parent")!),
      strokeColor: "black",
    })
  }
});

// these data come from Gencode
// compare roughly to the gene track at https://uswest.ensembl.org/Homo_sapiens/Location/
↪View?r=1:11000-36000;db=core
let gff3Data = `
chr1        HAVANA  gene    11869   14409   .       +       .       ID=ENSG00000223972.5;
↪gene_id=ENSG00000223972.5;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;level=2;hgnc_id=HGNC:37102;havana_gene=OTTHUMG00000000961.2
chr1        HAVANA  transcript      11869   14409   .       +       .       ␣
↪ID=ENST00000456328.2;Parent=ENSG00000223972.5;gene_id=ENSG00000223972.5;transcript_
↪id=ENST00000456328.2;gene_type=transcribed_unprocessed_pseudogene;gene_name=DDX11L1;
↪transcript_type=processed_transcript;transcript_name=DDX11L1-202;level=2;transcript_
↪support_level=1;hgnc_id=HGNC:37102;tag=basic;havana_gene=OTTHUMG00000000961.2;havana_
↪transcript=OTTHUMT00000362751.1
chr1        HAVANA  exon    11869   12227   .       +       .       ␣
↪ID=exon:ENST00000456328.2:1;Parent=ENST00000456328.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000456328.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=processed_transcript;transcript_name=DDX11L1-202;exon_
↪number=1;exon_id=ENSE00002234944.1;level=2;transcript_support_level=1;hgnc_
↪id=HGNC:37102;tag=basic;havana_gene=OTTHUMG00000000961.2;havana_
↪transcript=OTTHUMT00000362751.1
chr1        HAVANA  exon    12613   12721   .       +       .       ␣
↪ID=exon:ENST00000456328.2:2;Parent=ENST00000456328.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000456328.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=processed_transcript;transcript_name=DDX11L1-202;exon_
↪number=2;exon_id=ENSE00003582793.1;level=2;transcript_support_level=1;hgnc_
↪id=HGNC:37102;tag=basic;havana_gene=OTTHUMG00000000961.2;havana_
↪transcript=OTTHUMT00000362751.1
```

(continues on next page)

(continued from previous page)

```
chr1	HAVANA	exon	13221	14409	.	+	.	␣
↪ID=exon:ENST00000456328.2:3;Parent=ENST00000456328.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000456328.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=processed_transcript;transcript_name=DDX11L1-202;exon_
↪number=3;exon_id=ENSE00002312635.1;level=2;transcript_support_level=1;hgnc_
↪id=HGNC:37102;tag=basic;havana_gene=OTTHUMG00000000961.2;havana_
↪transcript=OTTHUMT00000362751.1
chr1	HAVANA	transcript	12010	13670	.	+	.	␣
↪ID=ENST00000450305.2;Parent=ENSG00000223972.5;gene_id=ENSG00000223972.5;transcript_
↪id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_name=DDX11L1;
↪transcript_type=transcribed_unprocessed_pseudogene;transcript_name=DDX11L1-201;level=2;
↪transcript_support_level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;
↪havana_gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	exon	12010	12057	.	+	.	␣
↪ID=exon:ENST00000450305.2:1;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=1;exon_id=ENSE00001948541.1;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	exon	12179	12227	.	+	.	␣
↪ID=exon:ENST00000450305.2:2;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=2;exon_id=ENSE00001671638.2;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	exon	12613	12697	.	+	.	␣
↪ID=exon:ENST00000450305.2:3;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=3;exon_id=ENSE00001758273.2;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	exon	12975	13052	.	+	.	␣
↪ID=exon:ENST00000450305.2:4;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=4;exon_id=ENSE00001799933.2;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	exon	13221	13374	.	+	.	␣
↪ID=exon:ENST00000450305.2:5;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=5;exon_id=ENSE00001746346.2;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
chr1	HAVANA	exon	13453	13670	.	+	.	␣
↪ID=exon:ENST00000450305.2:6;Parent=ENST00000450305.2;gene_id=ENSG00000223972.5;
↪transcript_id=ENST00000450305.2;gene_type=transcribed_unprocessed_pseudogene;gene_
↪name=DDX11L1;transcript_type=transcribed_unprocessed_pseudogene;transcript_
↪name=DDX11L1-201;exon_number=6;exon_id=ENSE00001863096.1;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:37102;ont=PGO:0000005,PGO:0000019;tag=basic;havana_
↪gene=OTTHUMG00000000961.2;havana_transcript=OTTHUMT00000002844.2
```

(continues on next page)

(continued from previous page)

```
chr1	HAVANA	gene	14404	29570	.	-	.	ID=ENSG00000227232.5;
↪gene_id=ENSG00000227232.5;gene_type=unprocessed_pseudogene;gene_name=WASH7P;level=2;
↪hgnc_id=HGNC:38034;havana_gene=OTTHUMG00000000958.1
chr1	HAVANA	transcript	14404	29570	.	-	.	␣
↪ID=ENST00000488147.1;Parent=ENSG00000227232.5;gene_id=ENSG00000227232.5;transcript_
↪id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;transcript_
↪type=unprocessed_pseudogene;transcript_name=WASH7P-201;level=2;transcript_support_
↪level=NA;hgnc_id=HGNC:38034;ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;
↪havana_transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	29534	29570	.	-	.	␣
↪ID=exon:ENST00000488147.1:1;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=1;exon_
↪id=ENSE00001890219.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	24738	24891	.	-	.	␣
↪ID=exon:ENST00000488147.1:2;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=2;exon_
↪id=ENSE00003507205.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	18268	18366	.	-	.	␣
↪ID=exon:ENST00000488147.1:3;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=3;exon_
↪id=ENSE00003477500.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	17915	18061	.	-	.	␣
↪ID=exon:ENST00000488147.1:4;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=4;exon_
↪id=ENSE00003565697.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	17606	17742	.	-	.	␣
↪ID=exon:ENST00000488147.1:5;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=5;exon_
↪id=ENSE00003475637.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	17233	17368	.	-	.	␣
↪ID=exon:ENST00000488147.1:6;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=6;exon_
↪id=ENSE00003502542.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	16858	17055	.	-	.	␣
↪ID=exon:ENST00000488147.1:7;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=7;exon_
↪id=ENSE00003533898.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
```

(continues on next page)

(continued from previous page)

```
chr1	HAVANA	exon	16607	16765	.	-	.	␣
↪ID=exon:ENST00000488147.1:8;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=8;exon_
↪id=ENSE00003621279.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	15796	15947	.	-	.	␣
↪ID=exon:ENST00000488147.1:9;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=9;exon_
↪id=ENSE00002030414.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	15005	15038	.	-	.	␣
↪ID=exon:ENST00000488147.1:10;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=10;exon_
↪id=ENSE00001935574.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	HAVANA	exon	14404	14501	.	-	.	␣
↪ID=exon:ENST00000488147.1:11;Parent=ENST00000488147.1;gene_id=ENSG00000227232.5;
↪transcript_id=ENST00000488147.1;gene_type=unprocessed_pseudogene;gene_name=WASH7P;
↪transcript_type=unprocessed_pseudogene;transcript_name=WASH7P-201;exon_number=11;exon_
↪id=ENSE00001843071.1;level=2;transcript_support_level=NA;hgnc_id=HGNC:38034;
↪ont=PGO:0000005;tag=basic;havana_gene=OTTHUMG00000000958.1;havana_
↪transcript=OTTHUMT00000002839.1
chr1	ENSEMBL	gene	17369	17436	.	-	.	ID=ENSG00000278267.1;
↪gene_id=ENSG00000278267.1;gene_type=miRNA;gene_name=MIR6859-1;level=3;hgnc_
↪id=HGNC:50039
chr1	ENSEMBL	transcript	17369	17436	.	-	.	␣
↪ID=ENST00000619216.1;Parent=ENSG00000278267.1;gene_id=ENSG00000278267.1;transcript_
↪id=ENST00000619216.1;gene_type=miRNA;gene_name=MIR6859-1;transcript_type=miRNA;
↪transcript_name=MIR6859-1-201;level=3;transcript_support_level=NA;hgnc_id=HGNC:50039;
↪tag=basic
chr1	ENSEMBL	exon	17369	17436	.	-	.	␣
↪ID=exon:ENST00000619216.1:1;Parent=ENST00000619216.1;gene_id=ENSG00000278267.1;
↪transcript_id=ENST00000619216.1;gene_type=miRNA;gene_name=MIR6859-1;transcript_
↪type=miRNA;transcript_name=MIR6859-1-201;exon_number=1;exon_id=ENSE00003746039.1;
↪level=3;transcript_support_level=NA;hgnc_id=HGNC:50039;tag=basic
chr1	HAVANA	gene	29554	31109	.	+	.	ID=ENSG00000243485.5;
↪gene_id=ENSG00000243485.5;gene_type=lncRNA;gene_name=MIR1302-2HG;level=2;hgnc_
↪id=HGNC:52482;tag=ncRNA_host;havana_gene=OTTHUMG00000000959.2
chr1	HAVANA	transcript	29554	31097	.	+	.	␣
↪ID=ENST00000473358.1;Parent=ENSG00000243485.5;gene_id=ENSG00000243485.5;transcript_
↪id=ENST00000473358.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_type=lncRNA;
↪transcript_name=MIR1302-2HG-202;level=2;transcript_support_level=5;hgnc_id=HGNC:52482;
↪tag=not_best_in_genome_evidence,dotter_confirmed,basic;havana_gene=OTTHUMG00000000959.
↪2;havana_transcript=OTTHUMT00000002840.1
chr1	HAVANA	exon	29554	30039	.	+	.	␣
↪ID=exon:ENST00000473358.1:1;Parent=ENST00000473358.1;gene_id=ENSG00000243485.5;
↪transcript_id=ENST00000473358.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_
↪type=lncRNA;transcript_name=MIR1302-2HG-202;exon_number=1;exon_id=ENSE00001947070.1;
↪level=2;transcript_support_level=5;hgnc_id=HGNC:52482;tag=not_best_in_genome_evidence,
↪dotter_confirmed,basic;havana_gene=OTTHUMG00000000959.2;havana_
↪transcript=OTTHUMT00000002840.1
```

(continues on next page)

(continued from previous page)

```
chr1	HAVANA	exon	30564	30667	.	+	.	␣
↪ID=exon:ENST00000473358.1:2;Parent=ENST00000473358.1;gene_id=ENSG00000243485.5;
↪transcript_id=ENST00000473358.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_
↪type=lncRNA;transcript_name=MIR1302-2HG-202;exon_number=2;exon_id=ENSE00001922571.1;
↪level=2;transcript_support_level=5;hgnc_id=HGNC:52482;tag=not_best_in_genome_evidence,
↪dotter_confirmed,basic;havana_gene=OTTHUMG00000000959.2;havana_
↪transcript=OTTHUMT00000002840.1
chr1	HAVANA	exon	30976	31097	.	+	.	␣
↪ID=exon:ENST00000473358.1:3;Parent=ENST00000473358.1;gene_id=ENSG00000243485.5;
↪transcript_id=ENST00000473358.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_
↪type=lncRNA;transcript_name=MIR1302-2HG-202;exon_number=3;exon_id=ENSE00001827679.1;
↪level=2;transcript_support_level=5;hgnc_id=HGNC:52482;tag=not_best_in_genome_evidence,
↪dotter_confirmed,basic;havana_gene=OTTHUMG00000000959.2;havana_
↪transcript=OTTHUMT00000002840.1
chr1	HAVANA	transcript	30267	31109	.	+	.	␣
↪ID=ENST00000469289.1;Parent=ENSG00000243485.5;gene_id=ENSG00000243485.5;transcript_
↪id=ENST00000469289.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_type=lncRNA;
↪transcript_name=MIR1302-2HG-201;level=2;transcript_support_level=5;hgnc_id=HGNC:52482;
↪tag=not_best_in_genome_evidence,basic;havana_gene=OTTHUMG00000000959.2;havana_
↪transcript=OTTHUMT00000002841.2
chr1	HAVANA	exon	30267	30667	.	+	.	␣
↪ID=exon:ENST00000469289.1:1;Parent=ENST00000469289.1;gene_id=ENSG00000243485.5;
↪transcript_id=ENST00000469289.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_
↪type=lncRNA;transcript_name=MIR1302-2HG-201;exon_number=1;exon_id=ENSE00001841699.1;
↪level=2;transcript_support_level=5;hgnc_id=HGNC:52482;tag=not_best_in_genome_evidence,
↪basic;havana_gene=OTTHUMG00000000959.2;havana_transcript=OTTHUMT00000002841.2
chr1	HAVANA	exon	30976	31109	.	+	.	␣
↪ID=exon:ENST00000469289.1:2;Parent=ENST00000469289.1;gene_id=ENSG00000243485.5;
↪transcript_id=ENST00000469289.1;gene_type=lncRNA;gene_name=MIR1302-2HG;transcript_
↪type=lncRNA;transcript_name=MIR1302-2HG-201;exon_number=2;exon_id=ENSE00001890064.1;
↪level=2;transcript_support_level=5;hgnc_id=HGNC:52482;tag=not_best_in_genome_evidence,
↪basic;havana_gene=OTTHUMG00000000959.2;havana_transcript=OTTHUMT00000002841.2
chr1	ENSEMBL	gene	30366	30503	.	+	.	ID=ENSG00000284332.1;
↪gene_id=ENSG00000284332.1;gene_type=miRNA;gene_name=MIR1302-2;level=3;hgnc_
↪id=HGNC:35294
chr1	ENSEMBL	transcript	30366	30503	.	+	.	␣
↪ID=ENST00000607096.1;Parent=ENSG00000284332.1;gene_id=ENSG00000284332.1;transcript_
↪id=ENST00000607096.1;gene_type=miRNA;gene_name=MIR1302-2;transcript_type=miRNA;
↪transcript_name=MIR1302-2-201;level=3;transcript_support_level=NA;hgnc_id=HGNC:35294;
↪tag=basic
chr1	ENSEMBL	exon	30366	30503	.	+	.	␣
↪ID=exon:ENST00000607096.1:1;Parent=ENST00000607096.1;gene_id=ENSG00000284332.1;
↪transcript_id=ENST00000607096.1;gene_type=miRNA;gene_name=MIR1302-2;transcript_
↪type=miRNA;transcript_name=MIR1302-2-201;exon_number=1;exon_id=ENSE00003695741.1;
↪level=3;transcript_support_level=NA;hgnc_id=HGNC:35294;tag=basic
chr1	HAVANA	gene	34554	36081	.	-	.	ID=ENSG00000237613.2;
↪gene_id=ENSG00000237613.2;gene_type=lncRNA;gene_name=FAM138A;level=2;hgnc_
↪id=HGNC:32334;havana_gene=OTTHUMG00000000960.1
chr1	HAVANA	transcript	34554	36081	.	-	.	␣
↪ID=ENST00000417324.1;Parent=ENSG00000237613.2;gene_id=ENSG00000237613.2;transcript_
↪id=ENST00000417324.1;gene_type=lncRNA;gene_name=FAM138A;transcript_type=lncRNA;
↪transcript_name=FAM138A-201;level=2;transcript_support_level=1;hgnc_id=HGNC:32334;
↪tag=basic;havana_gene=OTTHUMG00000000960.1;havana_transcript=OTTHUMT00000002843.1
```

(continues on next page)

(continued from previous page)

```
chr1        HAVANA  exon    35721   36081   .       -       .       ␣
↪ID=exon:ENST00000417324.1:1;Parent=ENST00000417324.1;gene_id=ENSG00000237613.2;
↪transcript_id=ENST00000417324.1;gene_type=lncRNA;gene_name=FAM138A;transcript_
↪type=lncRNA;transcript_name=FAM138A-201;exon_number=1;exon_id=ENSE00001656588.1;
↪level=2;transcript_support_level=1;hgnc_id=HGNC:32334;tag=basic;havana_
↪gene=OTTHUMG00000000960.1;havana_transcript=OTTHUMT00000002842.1
chr1        HAVANA  exon    35277   35481   .       -       .       ␣
↪ID=exon:ENST00000417324.1:2;Parent=ENST00000417324.1;gene_id=ENSG00000237613.2;
↪transcript_id=ENST00000417324.1;gene_type=lncRNA;gene_name=FAM138A;transcript_
↪type=lncRNA;transcript_name=FAM138A-201;exon_number=2;exon_id=ENSE00001669267.1;
↪level=2;transcript_support_level=1;hgnc_id=HGNC:32334;tag=basic;havana_
↪gene=OTTHUMG00000000960.1;havana_transcript=OTTHUMT00000002842.1
`;

let ann: soda.Gff3Annotation[] = soda.parseRecordsFromString(
  soda.parseGff3Record,
  gff3Data
);

let exonAnn = ann.filter((a) => a.type == "exon");

let groups: soda.AnnotationGroup<soda.Gff3Annotation>[] =
  soda.aggregateTransitive({
    annotations: exonAnn,
    criterion: (a, b) =>
      a.attributes!.get("Parent")! == b.attributes!.get("Parent"!),
  });

soda.intervalGraphLayout(groups);

chart.render({
  annotations: exonAnn,
});
```

While SODA was developed specifically to help support visualizing *nonstandard* annotation data, we still offer some support for some of the standard data formats. Here, you'll find some examples that show how to use some of those standard formats in a SODA visualization.

Contents:

3.5 Interactivity

3.5.1 Click behaviors

Once glyphs have been rendered in a *Chart*, you can bind arbitrary click behaviors to them using the *clickBehavior* function. In this example, we'll set up a click behavior that calls the browser's alert() function when a glyph is clicked.

To start off, let's first instantiate a Chart and render some glyphs:

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

Now, we'll make a call to the clickBehavior function, which takes a *ClickConfig* object as an argument. All we need to do is supply a list of Annotations and a callback function that will be executed when a glyph is clicked. The callback function receives as arguments a reference to a D3 Selection to the glyph that was clicked and the *AnnotationDatum* bound to that glyph.

```
// explicit type parameters added for clarity
soda.clickBehavior<soda.Annotation, soda.Chart>({
  annotations: ann
  click: (
    // we're usually not too concerned about how the D3 Selection is typed
    s: d3.Selection<any, any, any, any>,
    // explicit type parameters added for clarity
    d: AnnotationDatum<soda.Annotation, soda.Chart<any>>
   ) => {
    alert(`${d.a.id} clicked`)
  }
})
```

Finally, here's a full example that omits explicit type parameters and performs the call to clickBehavior in an inRender override:

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  inRender: function (this, params): void {
    soda.rectangle({
      chart: this,
      annotations: params.annotations || []
    })
```

(continues on next page)

(continued from previous page)

```
    soda.clickBehavior({
      annotations: params.annotations,
      click: (s, d) => {
        alert(`${d.a.id} clicked`)
      }
    })
  }
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

3.5.2 Hover behaviors

Hover behaviors can be bound to glyphs using the *hoverBehavior* function. The hoverBehavior function takes a *HoverConfig* as an argument.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
  inRender: function (this, params): void {
    soda.rectangle({
      chart: this,
      annotations: params.annotations || []
    })

    soda.hoverBehavior({
      annotations: params.annotations,
      mouseover: (s, d) => {
        s.style("fill", "cadetblue")
      },
      mouseout: (s, d) => {
        s.style("fill", "black")
      }
    })
  }
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})
```

(continues on next page)

(continued from previous page)

```
chart.render({
  annotations: ann
})
```

3.5.3 Tooltips

Toolips can be bound to glyphs using the *tooltip* function. The tooltip function takes a *TooltipConfig* as an argument.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true,
    inRender: function (this, params): void {
    soda.rectangle({
      chart: this,
      annotations: params.annotations || []
    })

    soda.tooltip({
      annotations: params.annotations,
      text: (d) => d.a.id
    })
  }
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

This section teaches you how to how to add some interactive features to your SODA visualization.

Contents:

- *Click behaviors*
- *Hover behaviors*
- *Tooltips*

3.6 Exporting images

In this example, we'll see how to export the current visual content of a Chart as a PNG image.

To start off, let's first instantiate a Chart and render some glyphs:

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})
```

Once we have a Chart rendered the way we want it, we simply have to call the *exportPng* function, which takes an *ExportConfig* object as an argument.

```
// when this function is called, the PNG
// will start to download in your browser
soda.exportPng({
  chart: chart,
  filename: "soda-chart.png"
})
```

Instead, you may want to bind the download function to a button on your page. To accomplish that, you might write something like:

```
// this assumes the DOM has a button with the id "save-image"
// we'll need to wrap the call to exportPng() in an anonymous
// function so we can pass the appropriate arguments
document.getElementById("save-image").onclick = () => soda.exportPng({
  chart: chart,
  filename: "soda-chart.png"
});
```

3.7 Multi-chart visualizations

3.7.1 Multiple charts rendering the same Annotations

You may find yourself wanting to build a visualization with multiple Chart components.

In this simple example, we will create two Charts that will render the same data slightly differently.

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true
});

let chart2 = new soda.Chart({
  selector: "#soda-chart2",
  axis: true,
  zoomable: true,
  inRender: function(this, params): void {
    soda.rectangle({
      chart: this,
      annotations: params.annotations || [],
      fillColor: "cadetblue"
    })
  }
});

let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})

chart2.render({
  annotations: ann
})
```

3.7.2 Syncing the zoom level across multiple charts

In this example, we'll build a multi-chart visualization that acts more like a set of "tracks" in a genome browser, where each Chart has a synchronized zoom level.

We'll start off by instantiating a couple of Charts:

```
let chart = new soda.Chart({
  selector: "#soda-chart",
  axis: true,
  zoomable: true
});

// we'll leave the axis off of this one
let chart2 = new soda.Chart({
  zoomable: true,
  inRender: function(this, params): void {
    soda.rectangle({
      chart: this,
```

(continues on next page)

(continued from previous page)

```
      annotations: params.annotations || [],
      fillColor: "cadetblue"
    })
  }
});
```

Now we'll create a *ZoomSyncer* object and add the Charts to it. After we've done this, the Charts will have synchronized zooming and panning.

```
let zoomSyncer = new soda.ZoomSyncer();
zoomSyncer.add([chart, chart2]);
```

Finally we can render some glyphs as usual:

```
let ann: Soda.Annotation = soda.generateAnnotations({
  n: 10
})

chart.render({
  annotations: ann
})

chart2.render({
  annotations: ann
})
```

This section explains how to configure multi-chart visualizations in SODA.

Contents:

- *Multiple charts rendering the same Annotations*
- *Syncing the zoom level across multiple charts*

This tutorial explains SODA concepts by example. Throughout the tutorial, it is generally assumed you have read everything up to the point you are currently at. If you get confused, try backing up a bit. If you find anything that seems missing or poorly explained, we're interested in hearing about it.

Contents:

- *The basics* - learn how to make basic SODA visualizations
- *Custom rendering* - learn how to customize the way SODA visualizations look
- *Vertical layout* - learn how vertical layouts work in SODA
- *Using common Bioinformatics data formats* - learn how to visualize common Bioinformatics data formats with SODA
- *Interactivity* - learn how to add interactive behaviors to a SODA visualization
- *Exporting images* - export a SODA visualization as a static image
- *Multi-chart visualizations* - learn how to configure multi-chart SODA visualizations

CHAPTER

FOUR

API

4.1 Classes

4.1.1 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.

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).

suppressWarnings

```
suppressWarnings: boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

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.

4.1.2 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
```

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): void
```

Add an Annotation to the group.

Parameters

- ann: A

Returns: void

4.1.3 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.

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.

suppressWarnings

```
suppressWarnings: boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

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.

4.1.4 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.

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).

suppressWarnings

```
suppressWarnings: boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

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.

4.1.5 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.

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.

suppressWarnings

```
suppressWarnings: boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

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.

4.1.6 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.

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.

suppressWarnings

```
suppressWarnings: boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

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.

4.1.7 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.

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.

suppressWarnings

```
suppressWarnings: boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

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.

4.1.8 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 to the Chart's DOM container. This is usually 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.

_renderEnd

```
_renderEnd: number
```

The semantic end coordinate of what is currently rendered.

_renderParams

```
_renderParams: undefined | P
```

The last used render parameters.

_renderStart

```
_renderStart: number
```

The semantic start coordinate of what is currently rendered.

_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.

axis

```
axis: boolean
```

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

glyphModifiers

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

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

id

```
id: string
```

A unique identifier for the Chart.

inRender

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

The second rendering callback function.

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.

preRender

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

The first rendering callback function.

resizable

```
resizable: boolean
```

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

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.

scaleExtent

```
scaleExtent: None
```

A list of two numbers that define the extent to which a zoom event is allowed to transform the TrackChart's underlying scale. Simply put, this controls how far in and out a user will be able to zoom. The first number is the maximum zoom-out factor, and the second is the maximum zoom-in factor. For example, setting this to [1, 10] will prevent a user from zooming out past the point at which the chart is initially rendered, and allow them to zoom in by a factor of 10. For more info, see https://github.com/d3/d3-zoom/blob/master/README.md#zoom_scaleExtent

translateExtent

```
translateExtent: (chart: Chart <any>): None
```

This is a callback function that is used to set the translate extent (left/right panning) allowed when a zoom event is applied to the TrackChart. It needs to be a callback, because it needs the absolute width of the TrackChart's SVG viewport, which is allowed to change throughout the TrackChart's lifetime. For example, setting this to: (chart) => [[0, 0], [chart.width, chart.height]] will restrict the panning in the TrackChart to exactly the range that was initially rendered. For more info, see https://github.com/d3/d3-zoom/blob/master/README.md#zoom_translateExtent

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.

xScaleBase

```
xScaleBase: ScaleLinear <number, number>
```

The base D3 scale that will be used to rescale the Chart's xScale.

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 height attribute on the viewport DOM element.

renderEnd

```
get renderEnd(): number
```

Getter for the renderEnd property

renderParams

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

Getter for the Chart's most recently used RenderParams.

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

Setter for the renderParms property.

renderStart

```
get renderStart(): number
```

Getter for the renderStart 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

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

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

disableZoom

```
disableZoom(): void
```

This disables zooming on the Chart.

Returns: void

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

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

rescaleXScale

```
rescaleXScale(transformArg: Transform): void
```

This rescales the Chart's x translation scale. If a transform argument is provided, it will use that. Otherwise, it will use the Chart's internal transform object.

Parameters

- transformArg: Transform

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

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

zoom

```
zoom(): void
```

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

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

4.1.9 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

4.1.10 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.

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.

suppressWarnings

```
suppressWarnings: boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

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.

4.1.11 PlotAnnotation

```
class PlotAnnotation
```

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

Constructors

```
(config: PlotAnnotationConfig): PlotAnnotation
```

Parameters

- config: PlotAnnotationConfig

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.

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).

suppressWarnings

```
suppressWarnings: boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

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.

4.1.12 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

characters

```
characters: None []
```

An array of [position, character] from the sequence.

columnTypes

```
columnTypes: ColumnType []
```

An array that describes the type of each position

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.

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).

suppressWarnings

```
suppressWarnings: boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

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.

4.1.13 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

4.2 Interfaces

4.2.1 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.

4.2.2 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.

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).

suppressWarnings

```
suppressWarnings: undefined | boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

width

```
width: undefined | number
```

The width of the annotation in semantic coordinates.

4.2.3 AnnotationConfigWithGroup

```
interface AnnotationConfigWithGroup<A extends Annotation>
```

An interface that extends AnnotationConfig for initializing AnnotationGroups.

Type parameters

- A: Annotation

Properties

end

```
end: undefined | number
```

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

group

```
group: A []
```

A list of Annotations to initially fill an AnnotationGroup with.

id

```
id: undefined | string
```

A unique identifier for an Annotation object.

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).

suppressWarnings

```
suppressWarnings: undefined | boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

width

```
width: undefined | number
```

The width of the annotation in semantic coordinates.

4.2.4 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.

4.2.5 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
```

4.2.6 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.

bindTarget

```
bindTarget: 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.

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.

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.

4.2.7 BarPlotConfig

```
interface BarPlotConfig<P extends PlotAnnotation, C extends Chart>
```

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

Type parameters

- P: PlotAnnotation
- C: Chart

Properties

annotations

```
annotations: P []
```

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

barHeightFn

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

binSpan

```
binSpan: 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.

bindTarget

```
bindTarget: 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.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

fillColor

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

A callback to define the fill color of the glyph.

fillOpacity

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

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <P, 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 <P, C, string>
```

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

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <P, 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 <P, 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 <P, 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 <P, 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 <P, C, number>
```

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

strokeWidth

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

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

width

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

A callback to define the pixel width of the glyph.

x

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

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

y

```
y: undefined | number | GlyphCallback <P, 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.

4.2.8 ChartConfig

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

This describes the parameters for configuring and initializing a Chart.

Type parameters

- P: RenderParams

Properties

axis

```
axis: undefined | boolean
```

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

height

```
height: undefined | number
```

The height in pixels of the Chart's viewport.

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.

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.

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.

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.

scaleExtent

```
scaleExtent: undefined | None
```

A range of floats that constrains the zoom level.

selector

```
selector: undefined | string
```

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

translateExtent

```
translateExtent: undefined | (c: Chart <P>): None
```

A callback function that provides a set of ranges that constrains the horizontal translation of the Chart.

width

```
width: undefined | number
```

The height in pixels of the Chart's viewport.

zoomable

```
zoomable: undefined | boolean
```

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

4.2.9 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.

bindTarget

```
bindTarget: 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.

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.

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.

4.2.10 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.

bindTarget

```
bindTarget: 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.

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.

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.

4.2.11 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.

bindTarget

```
bindTarget: 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.

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.

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.

4.2.12 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.

4.2.13 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.

4.2.14 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.

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.

suppressWarnings

```
suppressWarnings: undefined | boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

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.

4.2.15 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.

4.2.16 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.

bindTarget

```
bindTarget: 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.

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.

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.

4.2.17 GlyphMapping

```
interface GlyphMapping
```

An interface that contains a D3 selection to a glyph and the Chart it's rendered in.

Properties

chart

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

A reference to the Chart that the glyph is rendered in.

selection

```
selection: Selection <any, any, any, any>
```

The D3 selection to the glyph.

4.2.18 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.

bindTarget

```
bindTarget: 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.

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.

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.

4.2.19 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.

4.2.20 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.

bindTarget

```
bindTarget: 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.

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.

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.

4.2.21 LinePlotConfig

```
interface LinePlotConfig<P extends PlotAnnotation, C extends Chart>
```

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

Type parameters

- P: PlotAnnotation
- C: Chart

Properties

annotations

```
annotations: P []
```

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

binSpan

```
binSpan: 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.

bindTarget

```
bindTarget: 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.

chart

```
chart: C
```

The Chart object in which the glyphs will be rendered.

fillColor

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

A callback to define the fill color of the glyph.

fillOpacity

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

A callback to define the fill opacity of the glyph.

height

```
height: undefined | number | GlyphCallback <P, 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.

pathData

```
pathData: undefined | string | GlyphCallback <P, C, string>
```

A callback that returns a string that defines the line's SVG path

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 <P, C, string>
```

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

strokeDashArray

```
strokeDashArray: undefined | string | GlyphCallback <P, 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 <P, 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 <P, 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 <P, 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 <P, C, number>
```

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

strokeWidth

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

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

width

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

A callback to define the pixel width of the glyph.

x

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

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

y

```
y: undefined | number | GlyphCallback <P, 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.

4.2.22 PlotAnnotationConfig

```
interface PlotAnnotationConfig
```

An interface that defines the parameters for initializing a PlotAnnotation.

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.

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).

suppressWarnings

```
suppressWarnings: undefined | boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

width

```
width: undefined | number
```

The width of the annotation in semantic coordinates.

xValues

```
xValues: undefined | number []
```

The x values of the plot data.

yValues

```
yValues: number []
```

The y values of the plot data.

4.2.23 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.

bindTarget

```
bindTarget: 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.

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.

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.

4.2.24 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.

4.2.25 SequenceAnnotationConfig

```
interface SequenceAnnotationConfig
```

An interface that defines the parameters for initializing a SequenceAnnotation.

Properties

columnTypes

```
columnTypes: undefined | ColumnType []
```

An array of ColumnTypes, which should indicate the type of each position in the sequence. This array should be the same length as the sequence string.

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.

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).

suppressWarnings

```
suppressWarnings: undefined | boolean
```

This flag suppresses Annotation initialization warnings. Unless you really know what you're doing, you'll probably want to leave this alone.

width

```
width: undefined | number
```

The width of the annotation in semantic coordinates.

4.2.26 SequenceConfig

```
interface SequenceConfig<S extends SequenceAnnotation, C extends Chart>
```

An interface that defines the parameters for a call to sequenc 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.

bindTarget

```
bindTarget: 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.

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.

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.

4.2.27 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>
```

annotations

```
annotations: A []
```

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

bindTarget

```
bindTarget: 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.

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.

textAnchor

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

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.

4.2.28 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.

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.

4.2.29 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.

4.2.30 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.

binSpan

```
binSpan: 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.

bindTarget

```
bindTarget: 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.

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.

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.

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.

4.2.31 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.

4.3 Functions

4.3.1 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> []

4.3.2 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> []

4.3.3 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>

4.3.4 barPlot

```
function barPlot<P extends PlotAnnotation, C extends Chart>(config: BarPlotConfig <P, C>
↪): d3.Selection
```

This renders PlotAnnotations as bar plots in a Chart.

Type parameters

- P: PlotAnnotation
- C: Chart

Parameters

- config: BarPlotConfig <P, C>

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

4.3.5 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>

4.3.6 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>

4.3.7 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

4.3.8 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

4.3.9 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 []

4.3.10 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

4.3.11 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: PlotAnnotation []

4.3.12 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 []

4.3.13 getAllAnnotationIds

```
function getAllAnnotationIds(): None
```

This returns a list of all of the Annotation IDs that have been used to render glyphs.

Returns: string []

4.3.14 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

4.3.15 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>

4.3.16 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

4.3.17 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

4.3.18 heatmap

```
function heatmap<P extends PlotAnnotation, C extends Chart>(config: HeatmapConfig <P, C>
↪): d3.Selection
```

This renders PlotAnnotations as heatmaps in a Chart.

Type parameters

- P: PlotAnnotation
- C: Chart

Parameters

- config: HeatmapConfig <P, C>

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

4.3.19 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

4.3.20 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>

4.3.21 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

4.3.22 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

4.3.23 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>

4.3.24 linePlot

```
function linePlot<P extends PlotAnnotation, C extends Chart>(config: LinePlotConfig <P,␣
↪C>): d3.Selection
```

This renders PlotAnnotations as line plots in a Chart.

Type parameters

- P: PlotAnnotation
- C: Chart

Parameters

- config: LinePlotConfig <P, C>

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

4.3.25 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

4.3.26 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

4.3.27 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

4.3.28 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

4.3.29 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

4.3.30 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

4.3.31 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

4.3.32 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 []

4.3.33 queryGlyphMap

```
function queryGlyphMap<C extends Chart>(config: GlyphMapQueryConfig <C>): None
```

This function returns the GlyphMappings for the target Annotation IDs.

Type parameters

- C: Chart

Parameters

- config: GlyphMapQueryConfig <C>

Returns: GlyphMapping | GlyphMapping [] | undefined

4.3.34 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>

4.3.35 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

4.3.36 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>

4.3.37 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>

4.3.38 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

4.3.39 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. The vertical axes can be fixed in place, but they are configured to move during zoom events by default.

Type parameters

- A: Annotation
- C: Chart

Parameters

- config: VerticalAxisConfig <A, C>

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

4.4 Enumerations

4.4.1 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
```

4.4.2 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.

4.4.3 ColumnType

```
enum ColumnType
```

An enum to represent the type of a column in a sequence alignment.

Members

Deletion

```
Deletion: = "2"
```

This represents a gap in a sequence alignment.

Insertion

```
Insertion: = "3"
```

This represents an insertion in a sequence alignment.

Match

```
Match: = "0"
```

This represents a match in the sequence alignment.

Substitution

```
Substitution: = "1"
```

This represents a substitution in a sequence alignment.

4.4.4 GenerationPattern

```
enum GenerationPattern
```

Members

Random

```
Random: = "random"
```

Sequential

```
Sequential: = "sequential"
```

4.4.5 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.

4.5 Type aliases

4.5.1 AnnotationGroupConfig

```
type AnnotationGroupConfig = AnnotationConfigWithGroup | AnnotationConfig
```

A type that is simply the union of AnnotationConfig and AnnotationConfigWithGroup.

Type parameters

- A: Annotation

4.5.2 Bed12AnnotationConfig

```
type Bed12AnnotationConfig = Bed12Record & AnnotationConfig
```

A type that defines the parameters to initialize a Bed12Annotation.

4.5.3 Bed3AnnotationConfig

```
type Bed3AnnotationConfig = Bed3Record & AnnotationConfig
```

A type that defines the parameters to initialize a Bed3Annotation.

4.5.4 Bed6AnnotationConfig

```
type Bed6AnnotationConfig = Bed6Record & AnnotationConfig
```

A type that defines the parameters to initialize a Bed6Annotation.

4.5.5 Bed9AnnotationConfig

```
type Bed9AnnotationConfig = Bed9Record & AnnotationConfig
```

A type that defines the parameters to initialize a Bed9Annotation.

4.5.6 BedAnnotationConfig

```
type BedAnnotationConfig = AnnotationConfig & BedRecord
```

A type that defines the parameters to initialize a BedAnnotation.

4.5.7 GlyphCallback

```
type GlyphCallback = (d: AnnotationDatum <A, C>): V
```

A type that describes the callback functions used across SODA to define glyph properties dynamically.

Type parameters

- A: Annotation
- C: Chart
- V: generic

4.5.8 GlyphProperty

```
type GlyphProperty = GlyphCallback | V
```

A type that is simply the union of GlyphCallback<A, C, V> and the value V that it returns.

Type parameters

- A: Annotation
- C: Chart
- V: generic

This is the documentation for SODA, a TypeScript/Javascript library for creating genomic annotation visualizations.

Contents