Visualization Critique

yellkey.com/factor

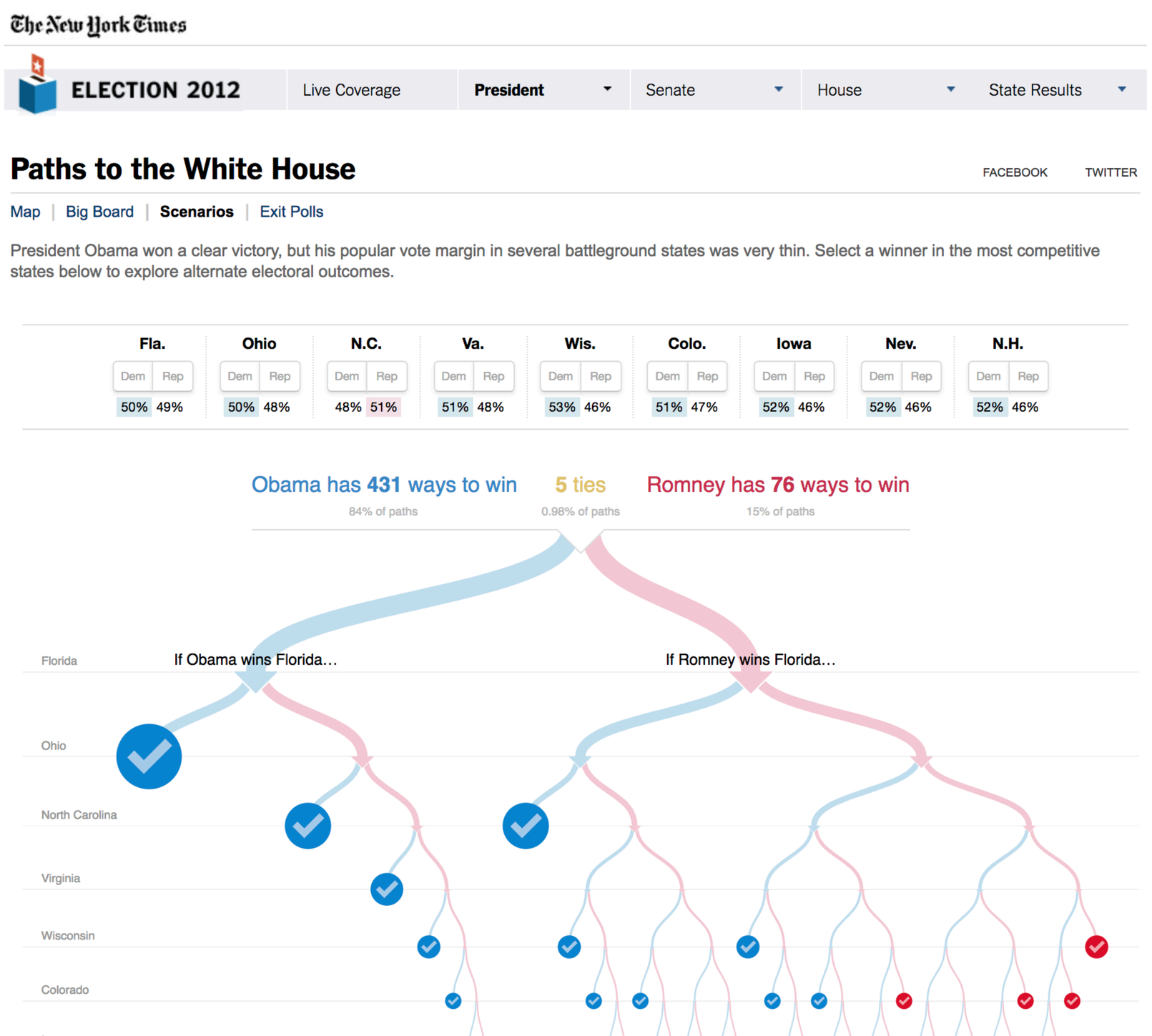
How are visual encodings used?

How is interactivity used?

To help structure your critique:

- *"I like..."*
- *"I wish..."*
- *"What if...?"*

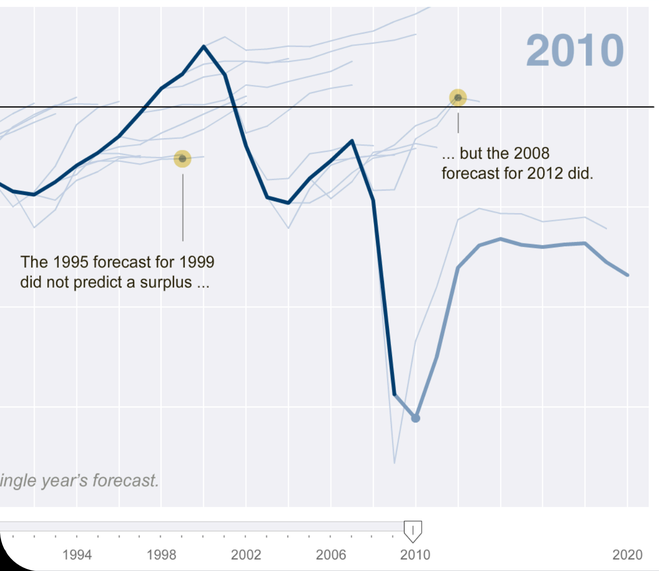
Think (~ 3 mins).
Pair (~7 mins).
Share.

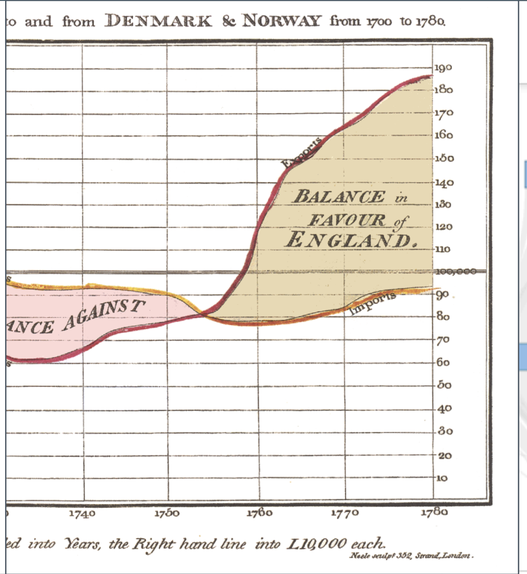


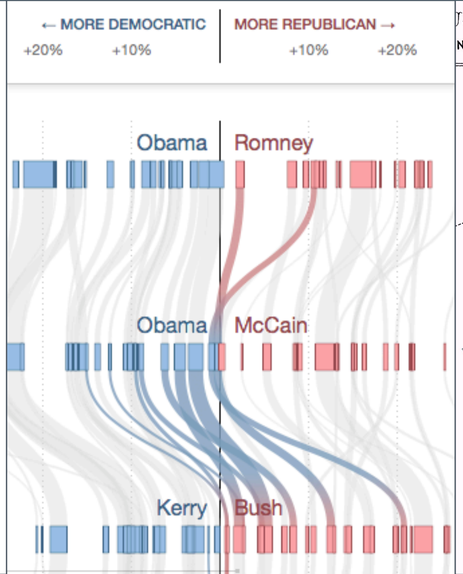
6.894: Interactive Data Visualization

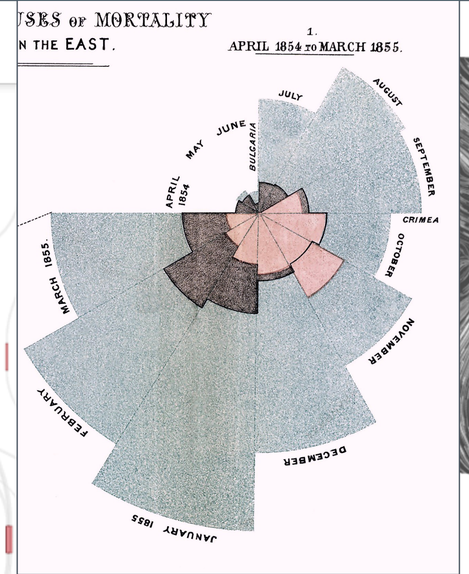
Selections, Signals, and Event Handlers

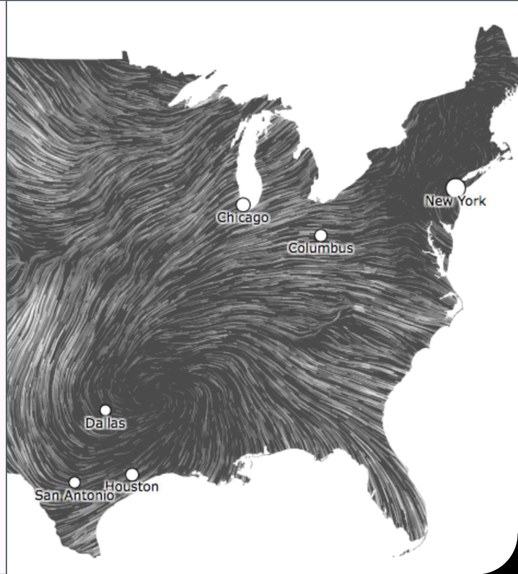
Arvind Satyanarayan



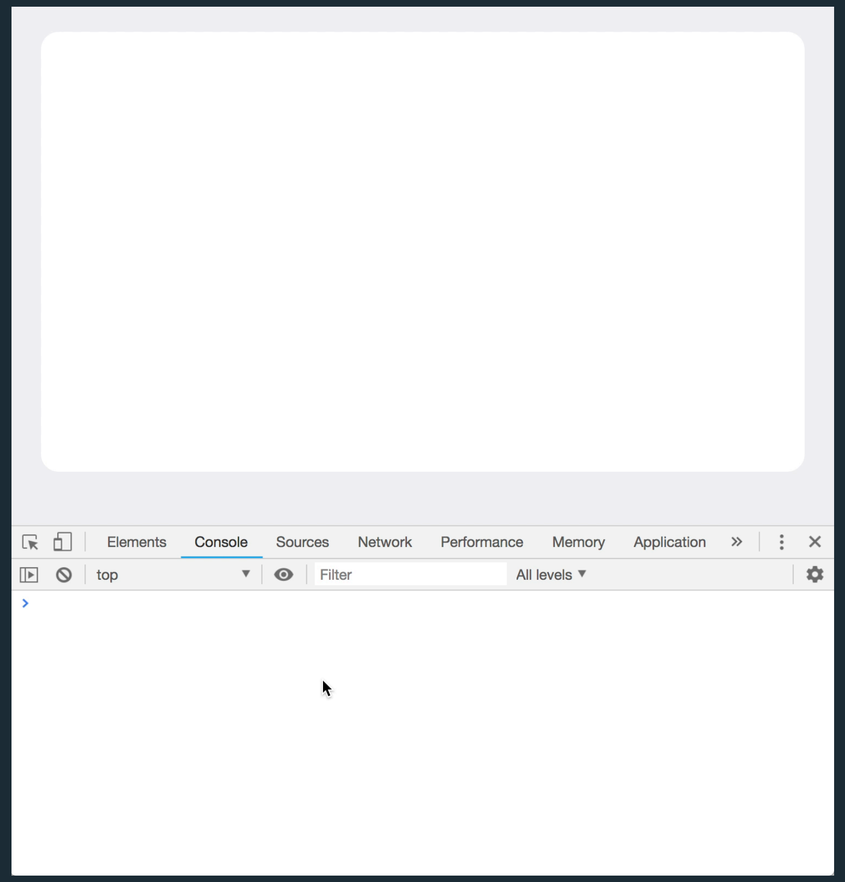


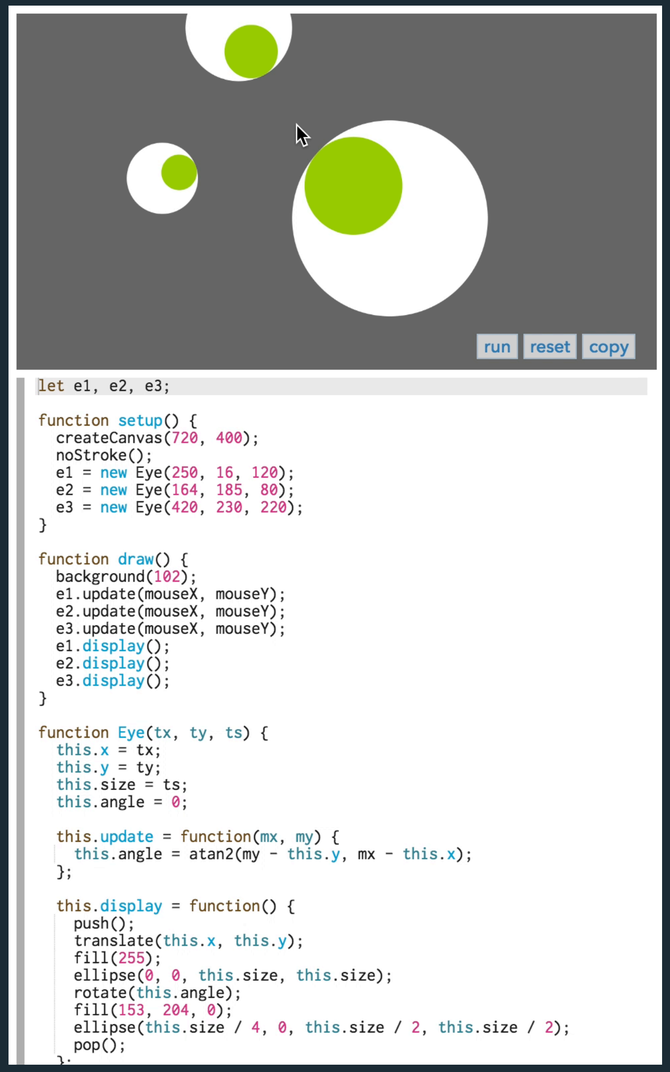






How do people create visualizations?





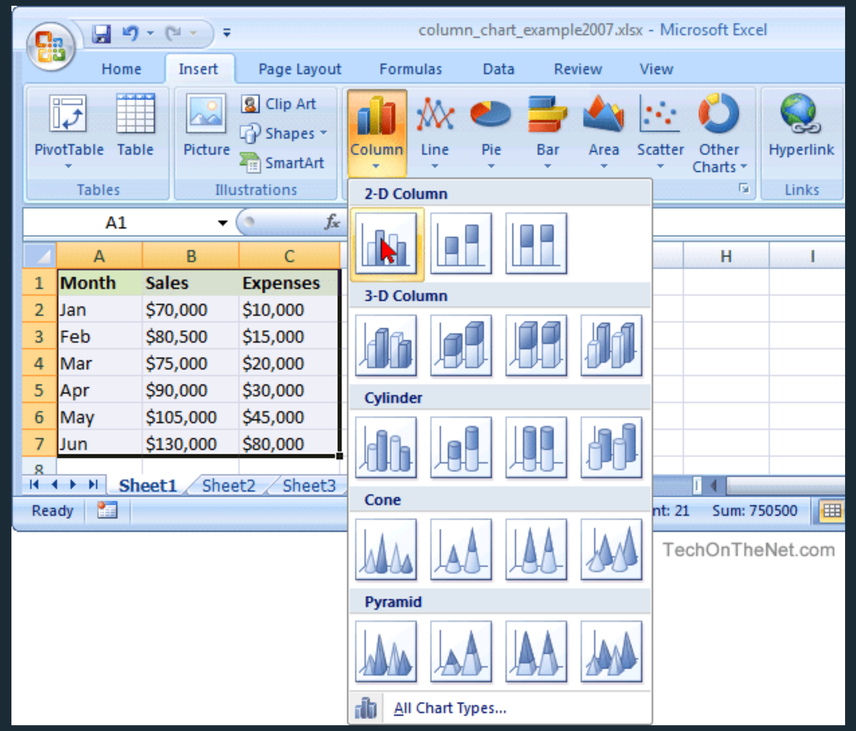
http://p5js.org

Graphics APIs

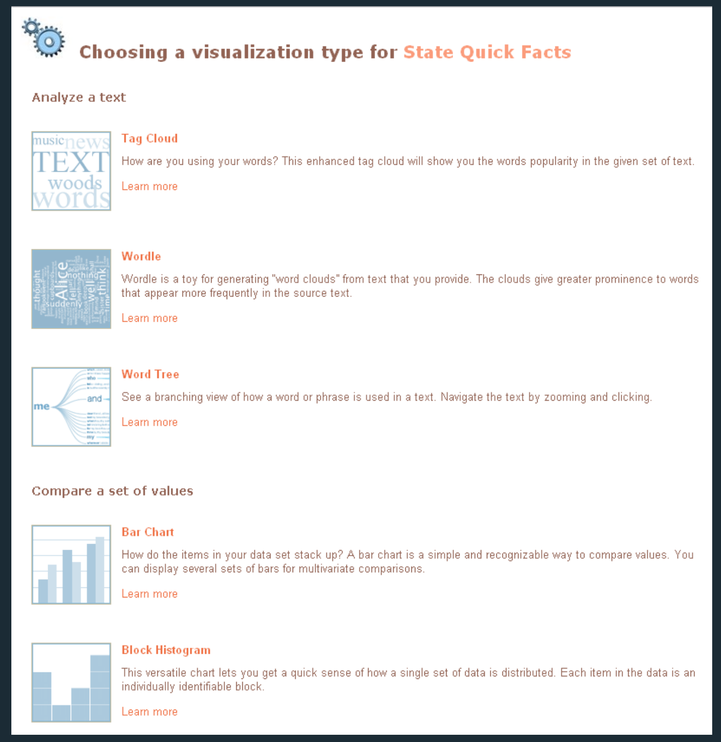
Canvas, Processing, OpenGL, Java2D

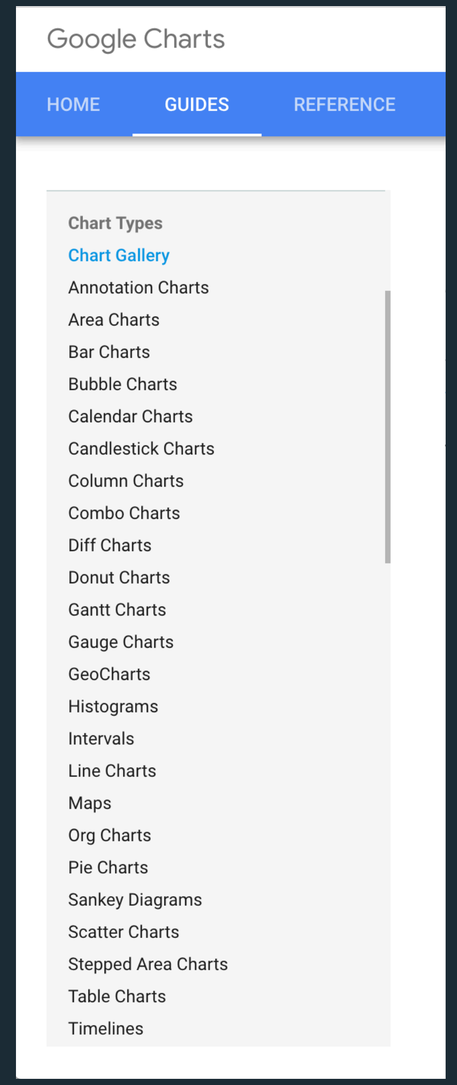
Chart Typologies

Excel, Many Eyes, Google Charts



https://www.techonthenet.com/excel/charts/column_chart2007.php



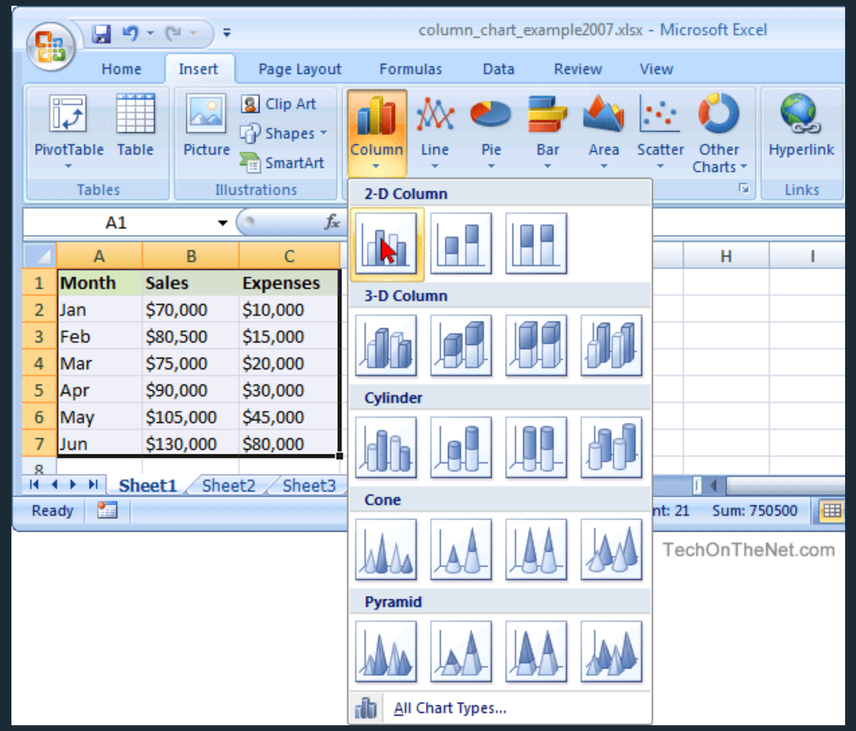


Graphics APIs

Canvas, Processing, OpenGL, Java2D

Chart Typologies

Excel, Many Eyes, Google Charts



https://www.techonthenet.com/excel/charts/column_chart2007.php

Graphics APIs

Canvas, Processing, OpenGL, Java2D

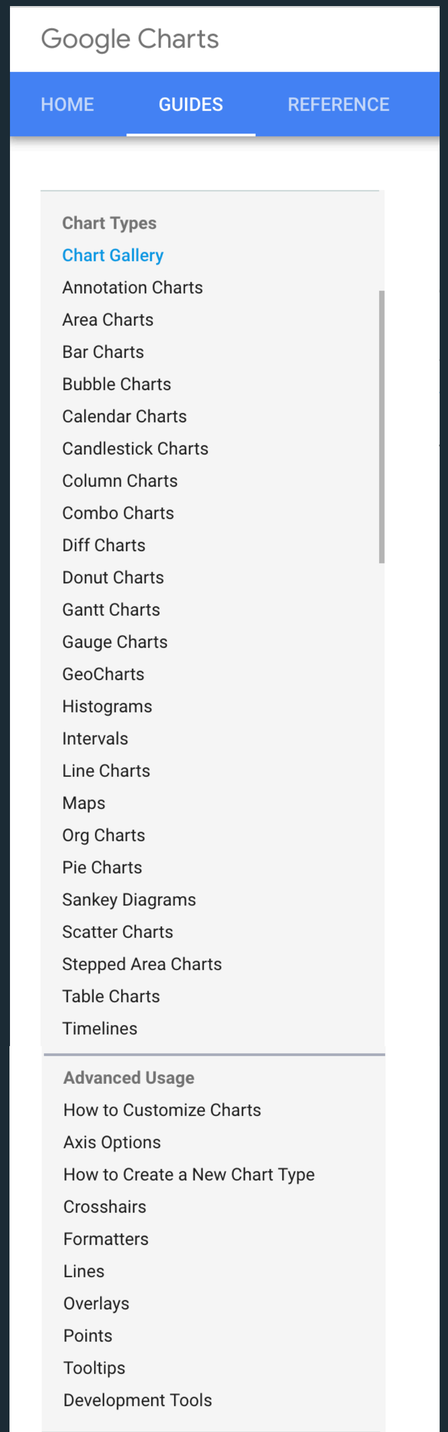
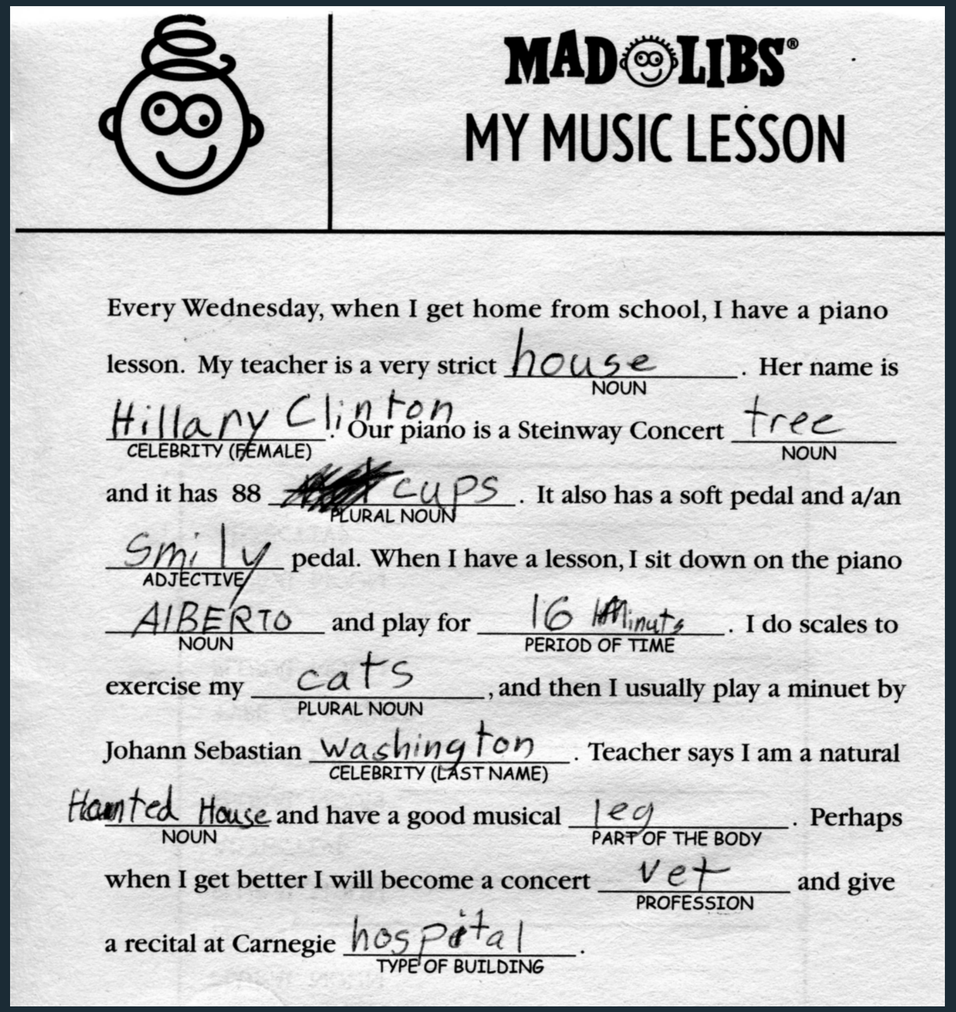


Chart Typologies

Excel, Many Eyes, Google Charts



MAD LIBS®

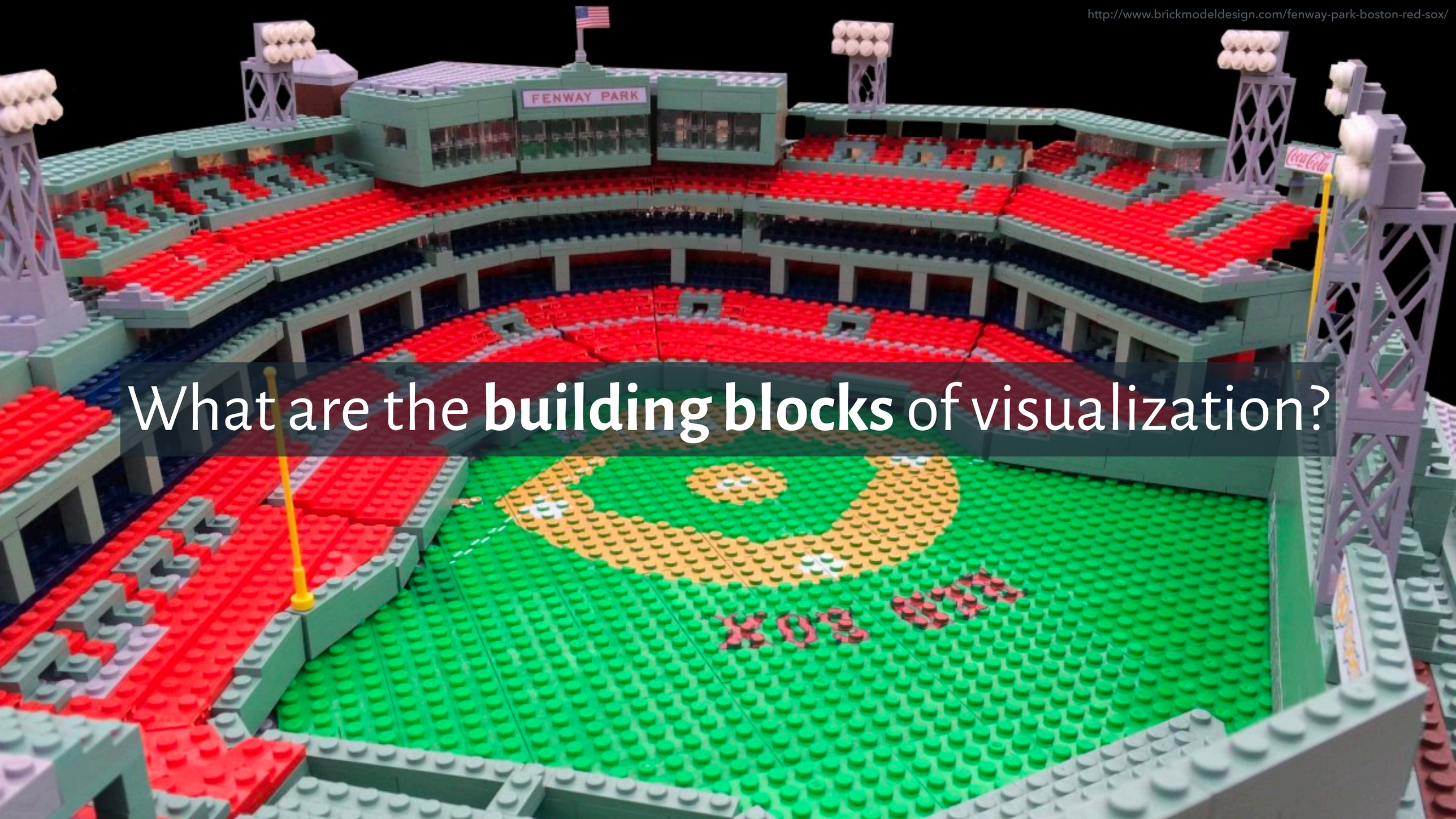
MY MUSIC LESSON

Every Wednesday, when I get home from school, I have a piano lesson. My teacher is a very strict house (NOUN). Her name is Hillary Clinton (CELEBRITY (FEMALE)). Our piano is a Steinway Concert tree (NOUN) and it has 88 ~~[illegible]~~ cups (PLURAL NOUN). It also has a soft pedal and a/an smily (ADJECTIVE) pedal. When I have a lesson, I sit down on the piano ALBERTO (NOUN) and play for 16 minuts (PERIOD OF TIME). I do scales to exercise my cats (PLURAL NOUN), and then I usually play a minuet by Johann Sebastian washington (CELEBRITY (LAST NAME)). Teacher says I am a natural Haunted House (NOUN) and have a good musical leg (PART OF THE BODY). Perhaps when I get better I will become a concert vet (PROFESSION) and give a recital at Carnegie hospital (TYPE OF BUILDING).

"[M]ost charting packages channel user requests into a **rigid array of chart types**. To atone for this lack of flexibility, they offer a kit of post-creation editing tools to return the image to what the user originally envisioned. **They give the user an impression of having explored data rather than the experience**."

Leland Wilkinson

The Grammar of Graphics, 1999



http://www.brickmodeldesign.com/fenway-park-boston-red-sox/

What are the **building blocks** of visualization?

Chart Typologies

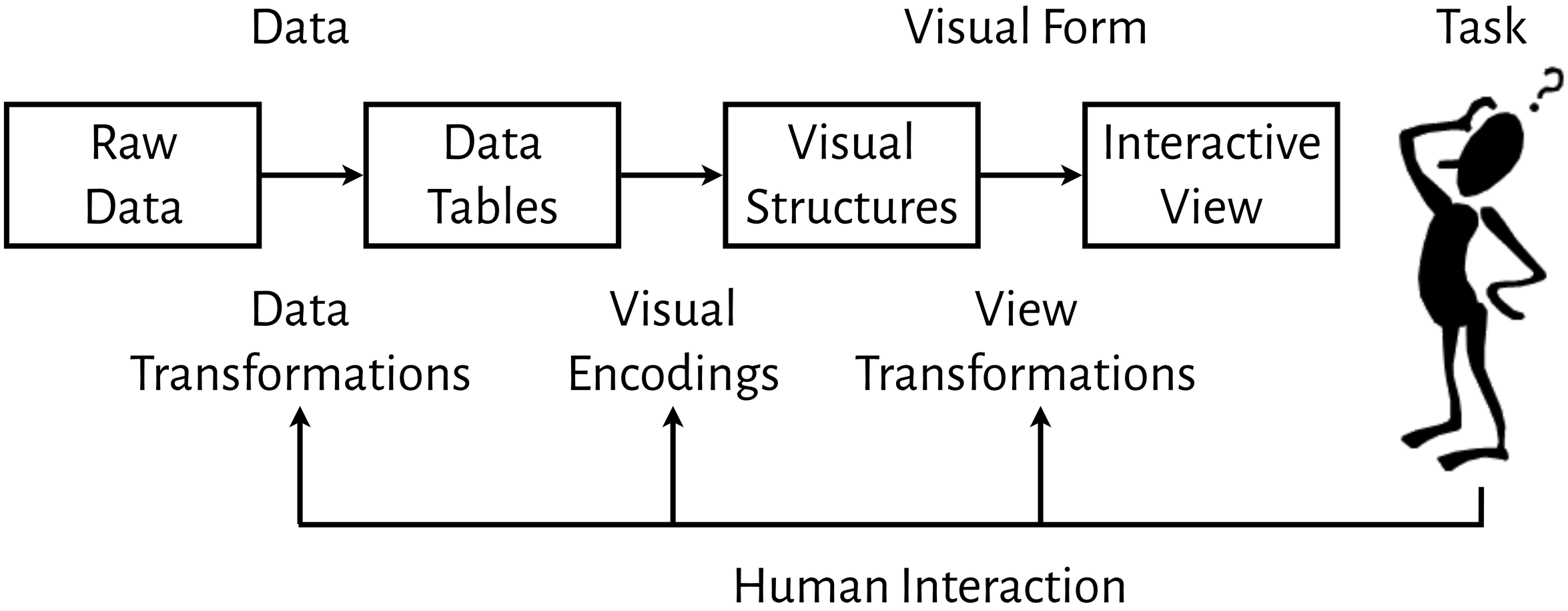
Excel, Many Eyes, Google Charts

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

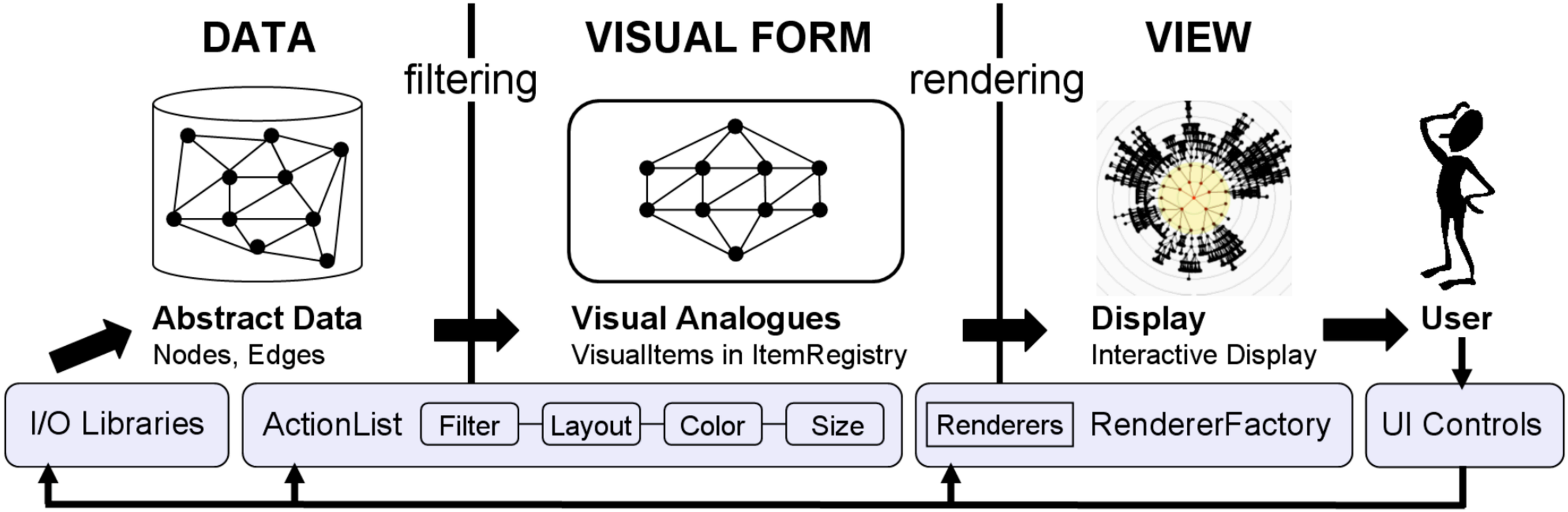
Canvas, Processing, OpenGL, Java2D



Component Architectures

Prefuse, Flare, Improvise, VTK

[Card, Mackinlay, Shneiderman. *Readings in Information Visualization*. 1999]



[Heer, Card, Landay. Prefuse. *CHI* 2005]

Component Architectures

Prefuse, Flare, Improvise, VTK

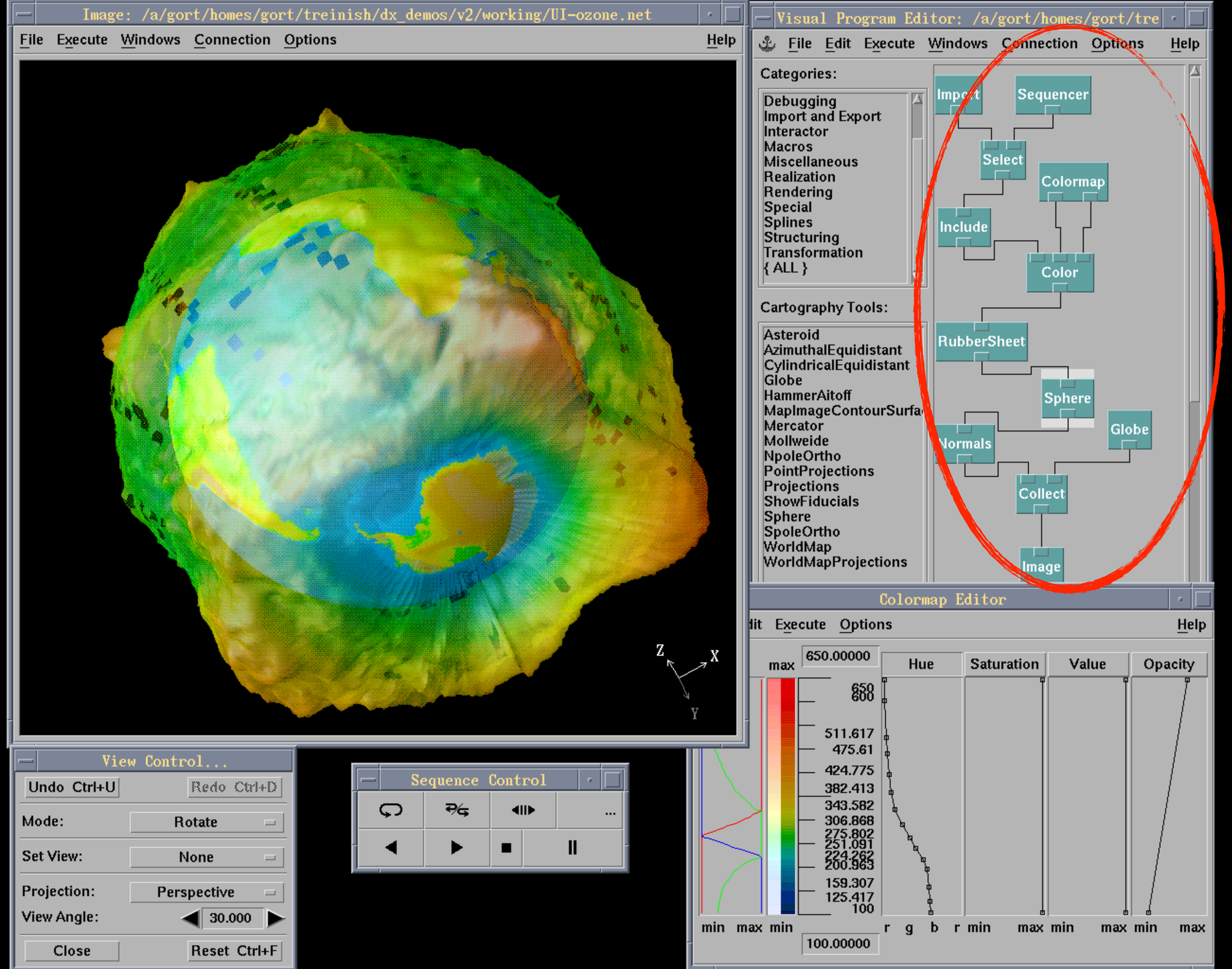


Image: /a/gort/homes/gort/treinish/dx_demos/v2/working/UI-ozone.net
File Execute Windows Connection Options Help
Visual Program Editor: /a/gort/homes/gort/tre
File Edit Execute Windows Connection Options Help
Categories:
Debugging
Import and Export
Interactor
Macros
Miscellaneous
Realization
Rendering
Special
Splines
Structuring
Transformation
{ ALL }
Cartography Tools:
Asteroid
AzimuthalEquidistant
CylindricalEquidistant
Globe
HammerAitoff
MapImageContourSurfa
Mercator
Mollweide
NpoleOrtho
PointProjections
Projections
ShowFiducials
Sphere
SpoleOrtho
WorldMap
WorldMapProjections
Import
Sequencer
Select
Colormap
Include
Color
RubberSheet
Sphere
Globe
Normals
Collect
Image
Z
X
Y
Colormap Editor
dit Execute Options Help
max
650.00000
650
600
511.617
475.61
424.775
382.413
343.582
306.868
275.802
251.091
224.262
200.963
159.307
125.417
100
100.00000
Hue
Saturation
Value
Opacity
min max min
r g b r min max min max min max
View Control...
Undo Ctrl+U
Redo Ctrl+D
Mode:
Rotate
Set View:
None
Projection:
Perspective
View Angle:
30.000
Close
Reset Ctrl+F
Sequence Control

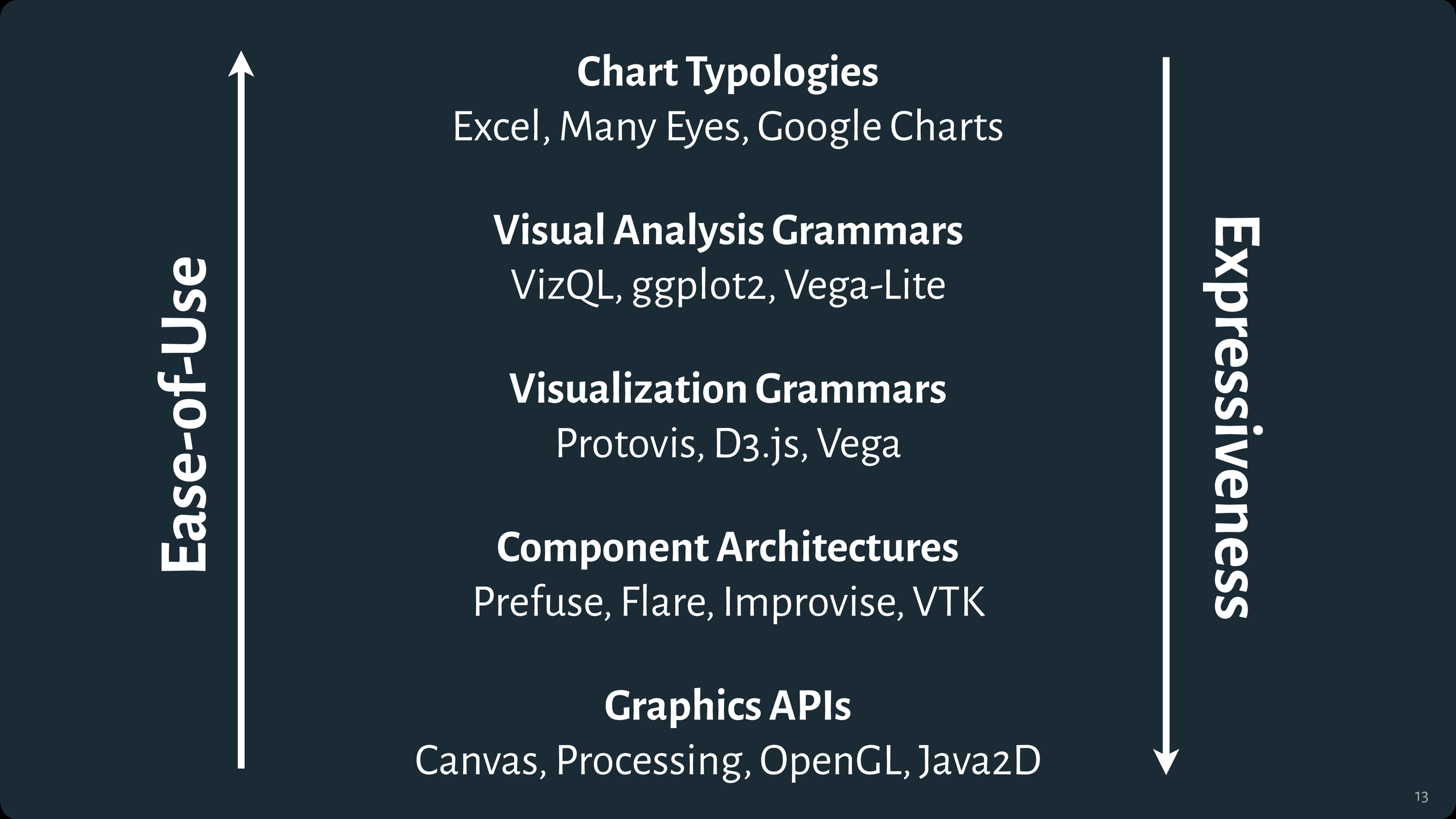

Ease-of-Use
Chart Typologies
Excel, Many Eyes, Google Charts
Visual Analysis Grammars
VizQL, ggplot2, Vega-Lite
Visualization Grammars
Protovis, D3.js, Vega
Component Architectures
Prefuse, Flare, Improvise, VTK
Graphics APIs
Canvas, Processing, OpenGL, Java2D
Expressiveness

Chart Typologies

Excel, Many Eyes, Google Charts

Charting Tools

Visual Analysis Grammars

VizQL, ggplot2, Vega-Lite

Visualization Grammars

Protovis, D3.js, Vega

Declarative Languages

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Canvas, Processing, OpenGL, Java2D

Programming Toolkits

Chart Typologies

Excel, Many Eyes, Google Charts

Charting Tools

Visual Analysis Grammars

VizQL, ggplot2, Vega-Lite

Visualization Grammars

Protovis, D3.js, Vega

Declarative Languages

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Canvas, Processing, OpenGL, Java2D

Programming Toolkits

Declarative Languages

Programming by describing **what**, not **how**.

Separate **specification** (what you want) from **execution** (how it should be computed).

```
d3.selectAll("rect")
   .data(stocks)
  .enter().append("rect")
   .attr("x", function(d) { return xscale(d.category); })
   .attr("y", function(d) { return yscale(d.price); })
```

In contrast, with imperative programming you must give explicit steps.

Declarative Languages

Better visualization? *Smart defaults.*

Reuse. *Write-once, then re-apply new data.*

Performance. *System optimizes processing + scalability.*

Portability. *Multiple devices, renderers, input modalities.*

Programmatic Generation. *Software that produces visualization, and does recommendation.*

Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

`iris.json`

Scales Map data values to visual values.

x : `petalWidth` → `x coordinate`

y : `sepalLength` → `y coordinate`

$color$: `species` → `["blue", ...]`

sepalLength	sepalWidth	petalLength	petalWidth	species
4.3	3	1.1	0.1	"setosa"
4.4	3	1.3	0.2	"setosa"
4.4	2.9	1.4	0.2	"setosa"
4.4	3.2	1.3	0.2	"setosa"
4.5	2.3	1.3	0.3	"setosa"
4.6	3.2	1.4	0.2	"setosa"
4.6	3.4	1.4	0.3	"setosa"
4.6	3.6	1	0.2	"setosa"
4.6	3.1	1.5	0.2	"setosa"
4.7	3.2	1.6	0.2	"setosa"
4.7	3.2	1.3	0.2	"setosa"
4.8	3	1.4	0.3	"setosa"
4.8	3.4	1.6	0.2	"setosa"
4.8	3	1.4	0.1	"setosa"
4.8	3.4	1.9	0.2	"setosa"
4.8	3.1	1.6	0.2	"setosa"
4.9	2.4	3.3	1	"versicolor"
4.9	3.1	1.5	0.2	"setosa"
4.9	3	1.4	0.2	"setosa"
4.9	3.6	1.4	0.1	"setosa"
4.9	3.1	1.5	0.1	"setosa"
4.9	2.5	4.5	1.7	"virginica"
5	3.3	1.4	0.2	"setosa"
5	3.4	1.6	0.4	"setosa"
5	3.2	1.2	0.2	"setosa"

Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

`iris.json`

Scales Map data values to visual values.

x : `petalWidth` → `x coordinate`

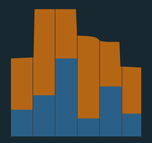
y : `sepalLength` → `y coordinate`

$color$: `species` → `["blue", ...]`

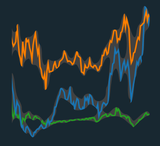
Guides Axes & legends to visualize scales.

Marks Data-representative graphics.


Area

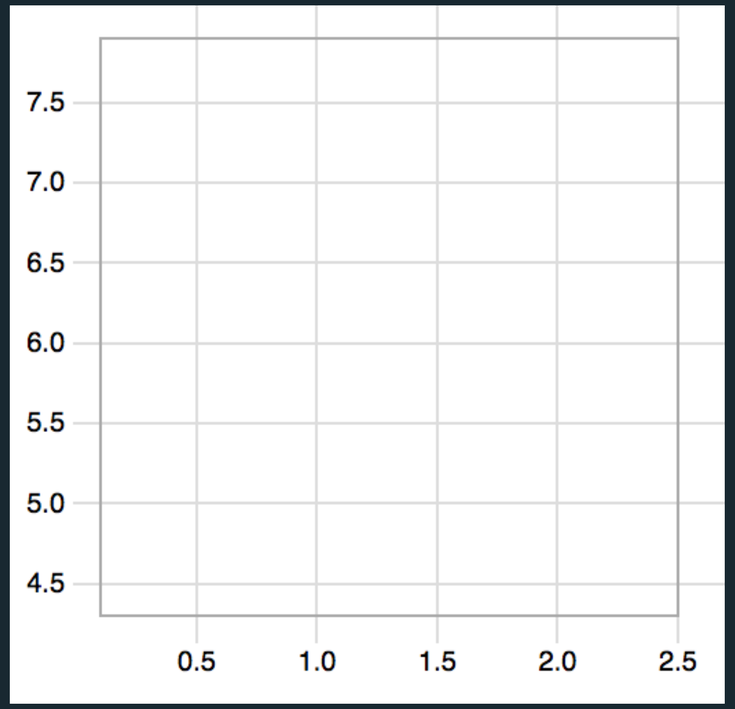

Rect


Symbol


Line


Arc

Text



Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

`iris.json`

Scales Map data values to visual values.

x : `petalWidth` → `x coordinate`

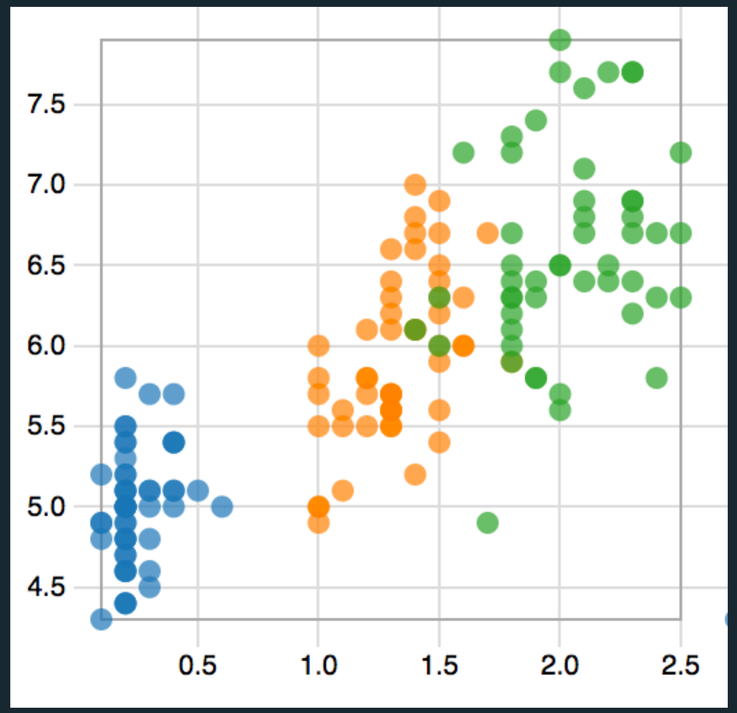
y : `sepalLength` → `y coordinate`

$color$: `species` → `["blue", ...]`

Guides Axes & legends to visualize scales.

Marks Data-representative graphics.

Area Rect Symbol Line Arc Text



Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

```
iris.json
fields: ["petal length", "petal width", ...]
```

Transforms Operators to filter, group, etc.

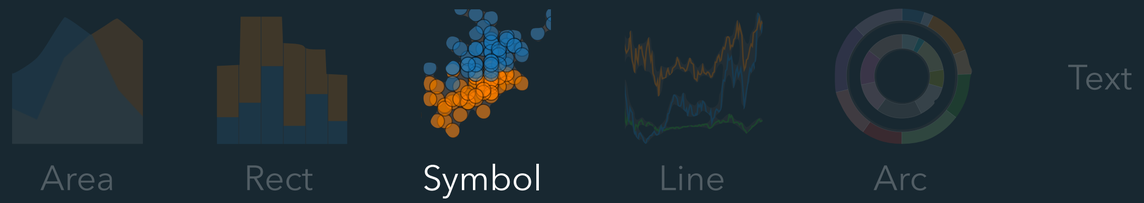
```
Cross product of fields
```

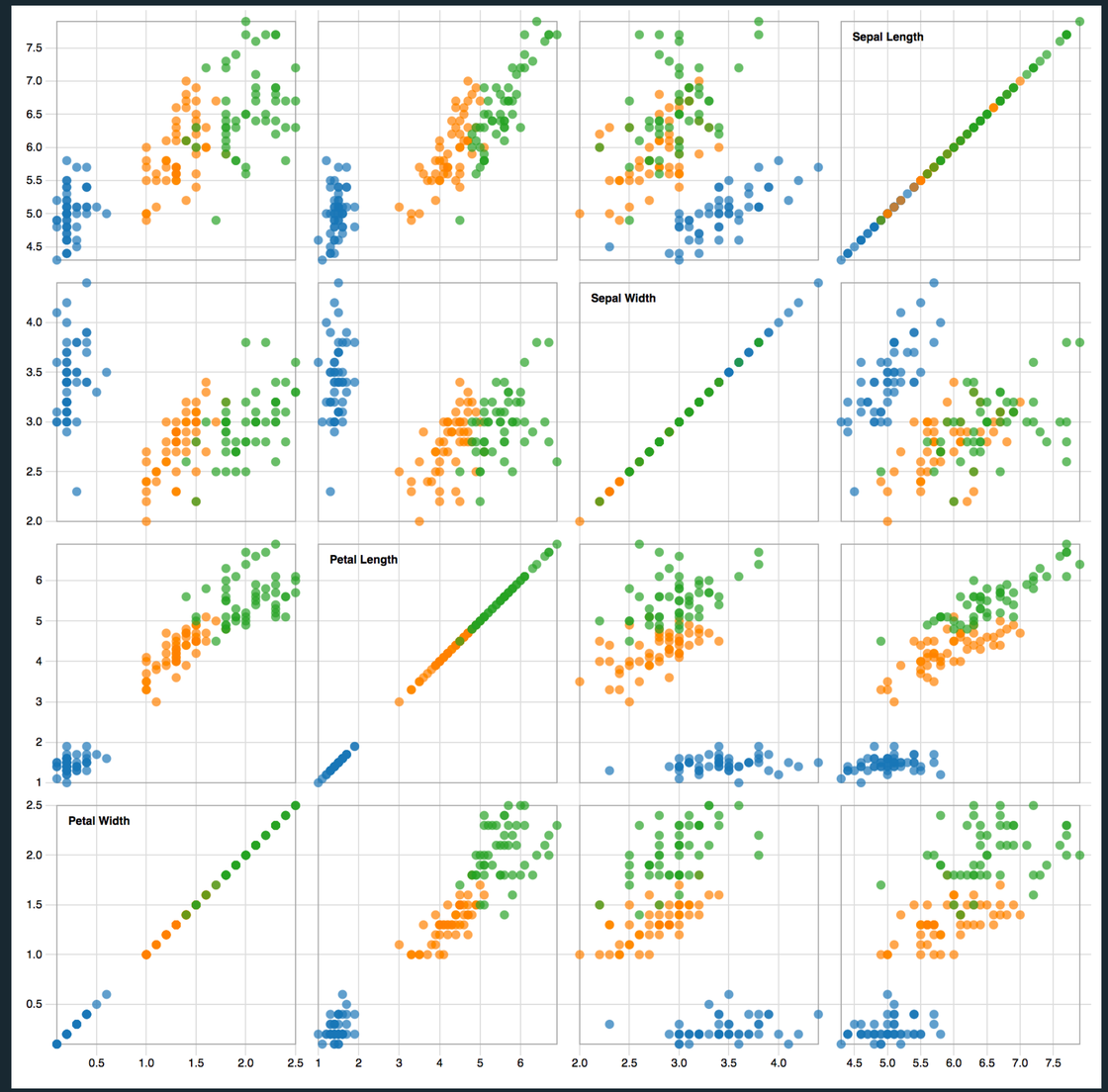
Scales Map data values to visual values.

```
x : petalWidth → x coordinate
y : sepalLength → y coordinate
color : species → ["blue", ...]
```

Guides Axes & legends to visualize scales.

Marks Data-representative graphics.





Declarative Visual Encodings

Data Input data to visualize.

`iris.json`

`fields: ["petal length", "petal width", ...]`

Transforms Operators to filter, group, etc.

`Cross product of fields`

Scales Map data values to visual values.

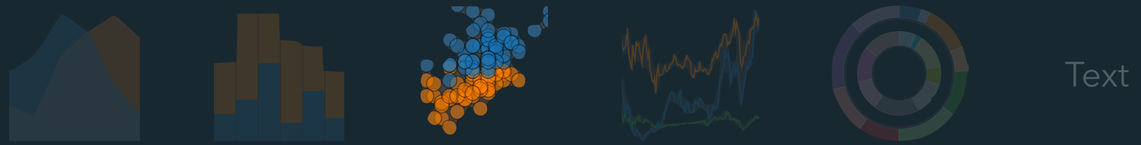
x : `petalWidth` → `x coordinate`

y : `sepalLength` → `y coordinate`

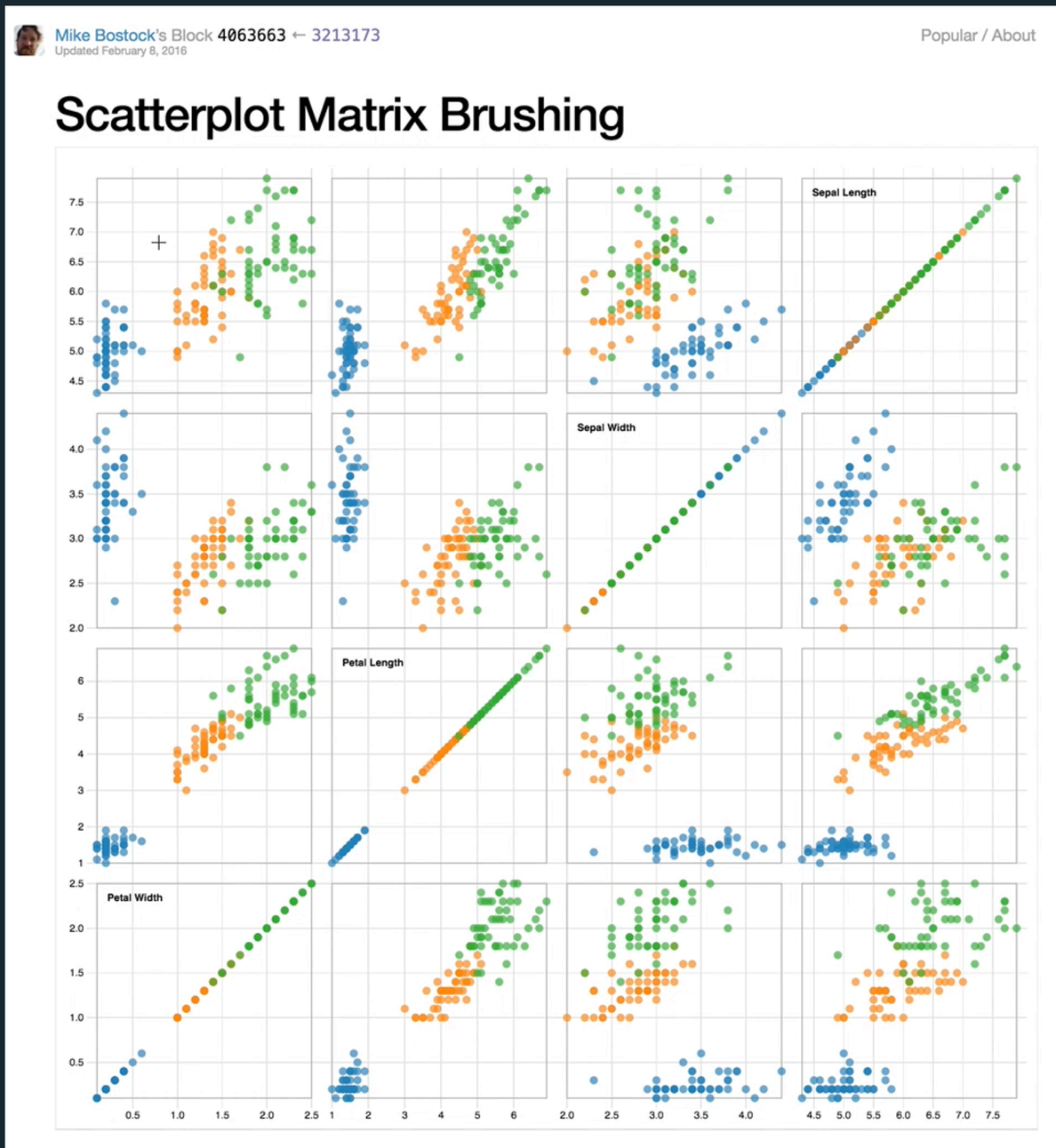
color : `species` → `["blue", ...]`

Guides Axes & legends to visualize scales.

Marks Data-representative graphics.



Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*



Imperative Interaction

Usability Issues

Execution exposed to users (e.g., low-level JavaScript idiosyncrasies).

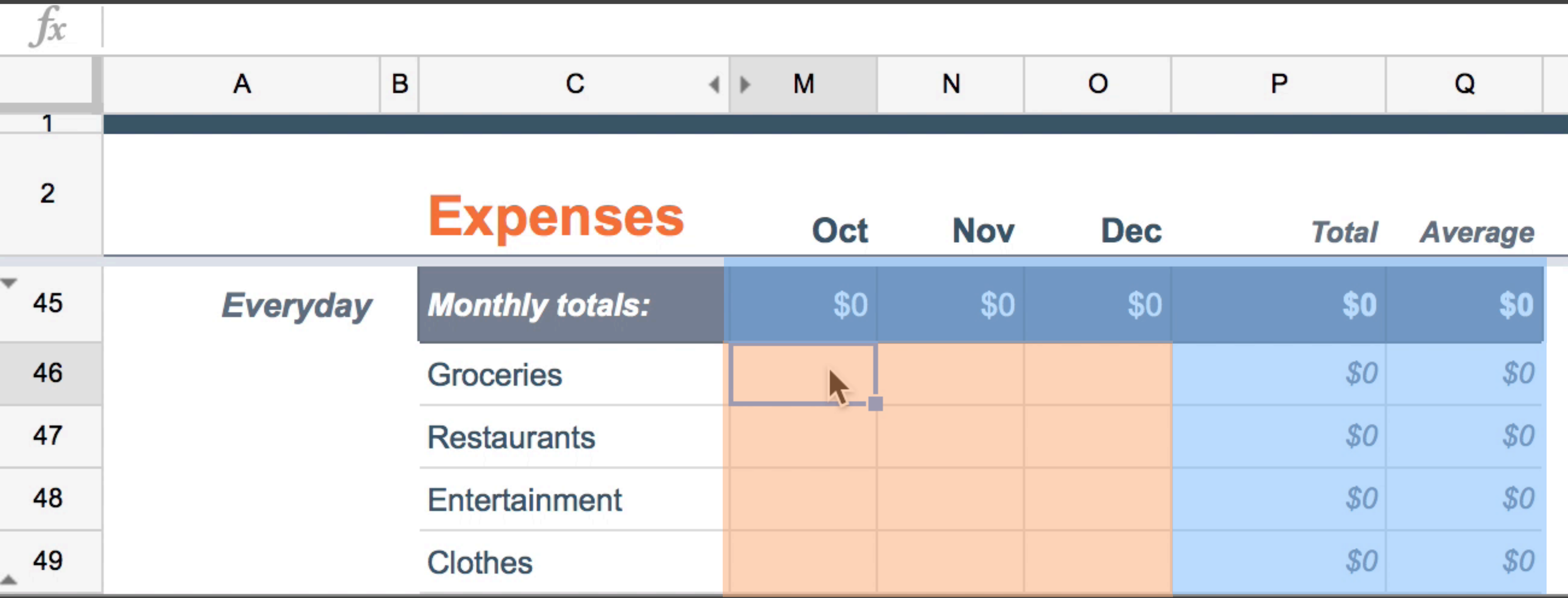
State: manually maintained and propagated.

Redefine visualization appearance in multiple locations (**“side-effects”)**

Callback Hell: unpredictable and interleaved execution order.

```
    .enter().append("circle")
      .attr("cx", function(d) { return x(d[p.x]); })
      .attr("cy", function(d) { return y(d[p.y]); })
      .attr("r", 4)
      .style("fill", function(d) { return color(d.species); });
}

var brushCell;

// Clear the previously-active brush, if any.
function brushstart(p) {
  if (brushCell !== this) {
    d3.select(brushCell).call(brush.clear());
    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);
    brushCell = this;
  }
}

// Highlight the selected circles.
function brushmove(p) {
  var e = brush.extent();
  svg.selectAll("circle").classed("hidden", function(d) {
    return e[0][0] > d[p.x] || d[p.x] > e[1][0]
        || e[0][1] > d[p.y] || d[p.y] > e[1][1];
  });
}

// If the brush is empty, select all circles.
function brushend() {
  if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
}

d3.select(self.frameElement).style("height", size * n + padding + 20 + "
);

unction cross(a, b) {
 var c = [], n = a.length, m = b.length, i, j;
 for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y:
 return c;
```

Reactive Programming



Events are streaming data.Dynamic variables (**signals**) automatically update.

Declarative Interaction Primitives

Data

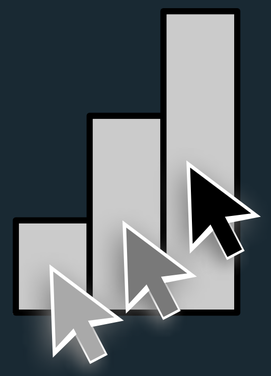
Transforms

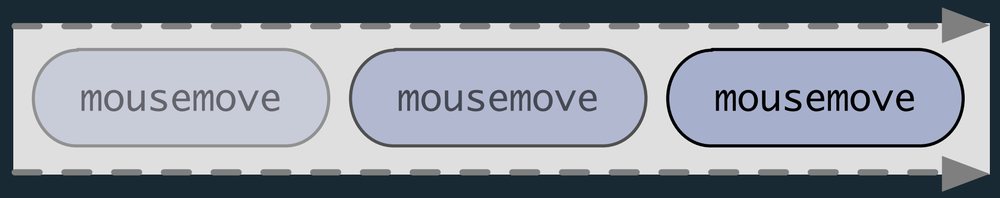
Scales

Guides

Marks

Event Streams





A stream of `mousemove` events that occur on `rect` marks .

`rect:mousemove`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

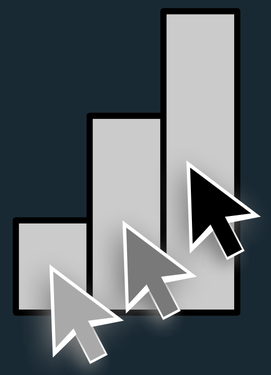
Transforms

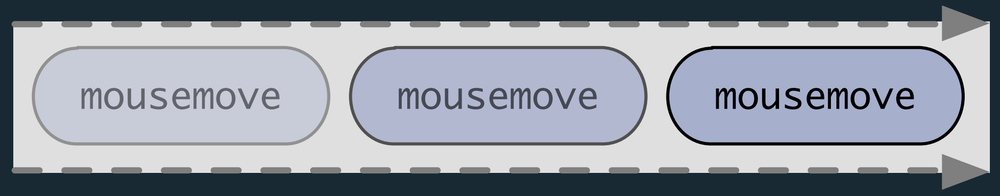
Scales

Guides

Marks

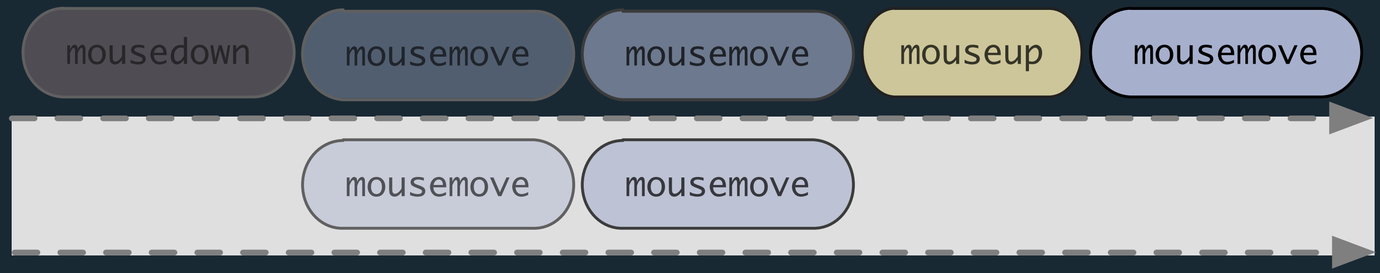
Event Streams





A stream of `mousemove` events that occur on `rect` marks .

`rect:mousemove`



`[mousedown, mouseup] > mousemove`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

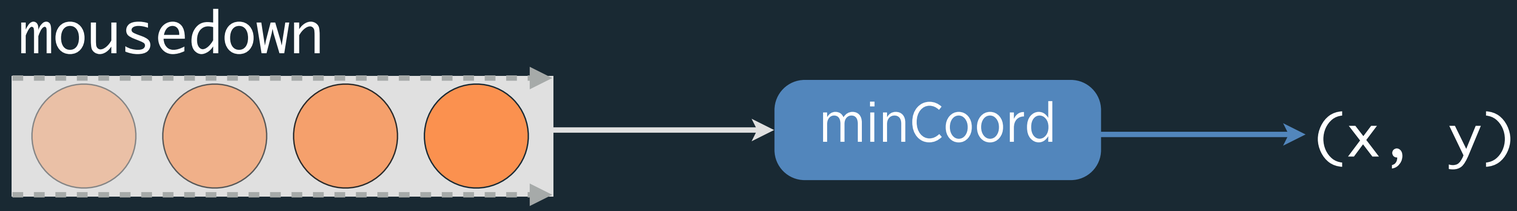
Scales

Guides

Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals Dynamic variables that are recalculated when events fire.



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals `minCoord := (mousedown.x, mousedown.y)`

Scale Inversions Lift visual/pixel values to the data domain.

```
minDatum.x := xScale.invert(minCoord.x)
```

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

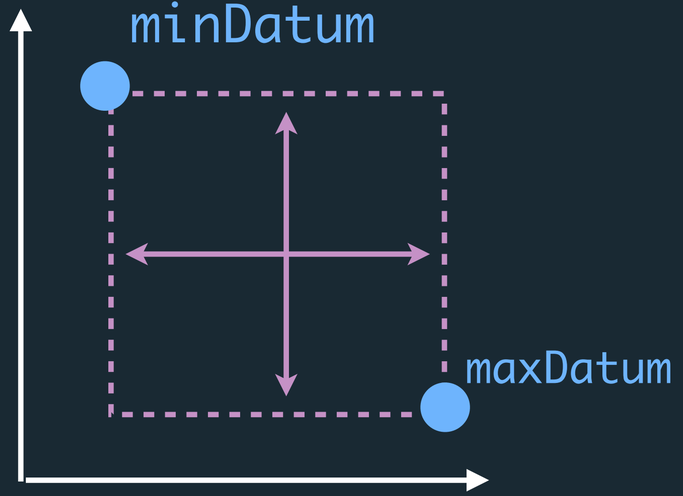
Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals `minCoord := (mousedown.x, mousedown.y)`

Scale Inversions `minDatum.x := xScale.invert(minCoord.x)`

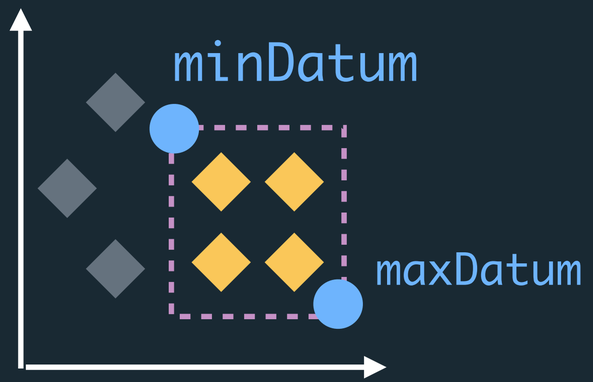
Predicates Functions that determine which points are selected.



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data	**Event Streams**	`[mousedown, mouseup] > mousemove`
Transforms	**Signals**	`minCoord := (mousedown.x, mousedown.y)`
Scales	**Scale Inversions**	`minDatum.x := xScale.invert(minCoord.x)`
Guides	**Predicates**	`p(d) := d in [minDatum, maxDatum]`
Marks	**Production Rules**	If-then-else rules to manipulate visual encodings.



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data	**Event Streams**	`[mousedown, mouseup] > mousemove`
Transforms	**Signals**	`minCoord := (mousedown.x, mousedown.y)`
Scales	**Scale Inversions**	`minDatum.x := xScale.invert(minCoord.x)`
Guides	**Predicates**	`p(d) := d in [minDatum, maxDatum]`
Marks	**Production Rules**	`fill := p(d) → colorScale(d.species)` `∅ → gray`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

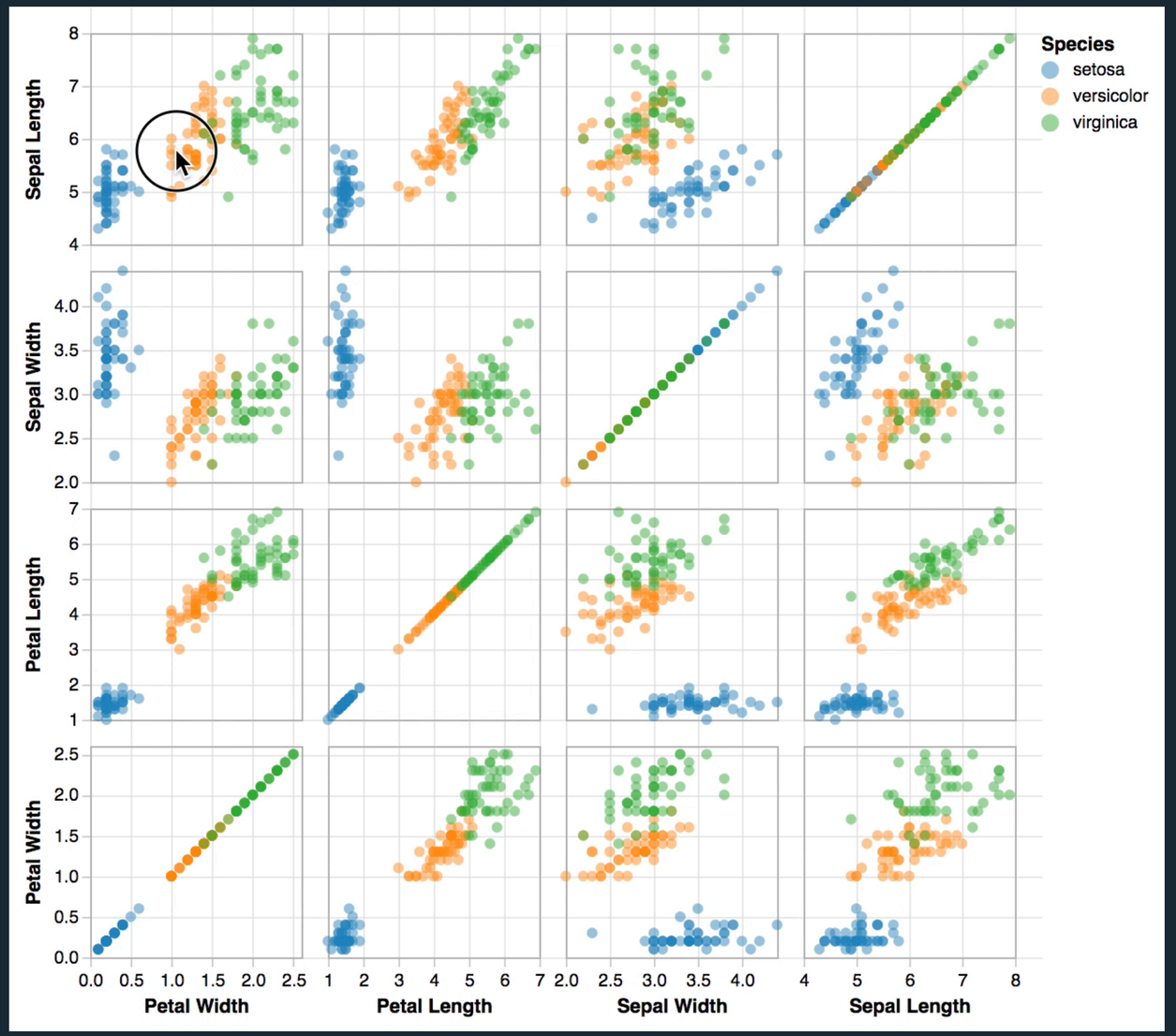
Event Streams

Signals

Scale Inversions

Predicates

Production Rules

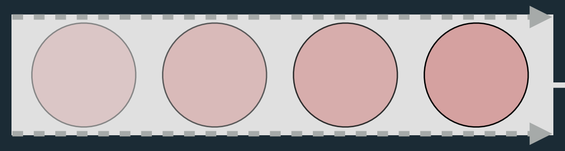


Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

mousedown

[mousedown, mouseup] >
mousemove

mousedown

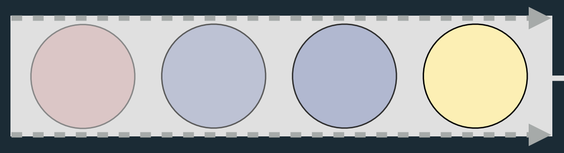


Start

(x, y)

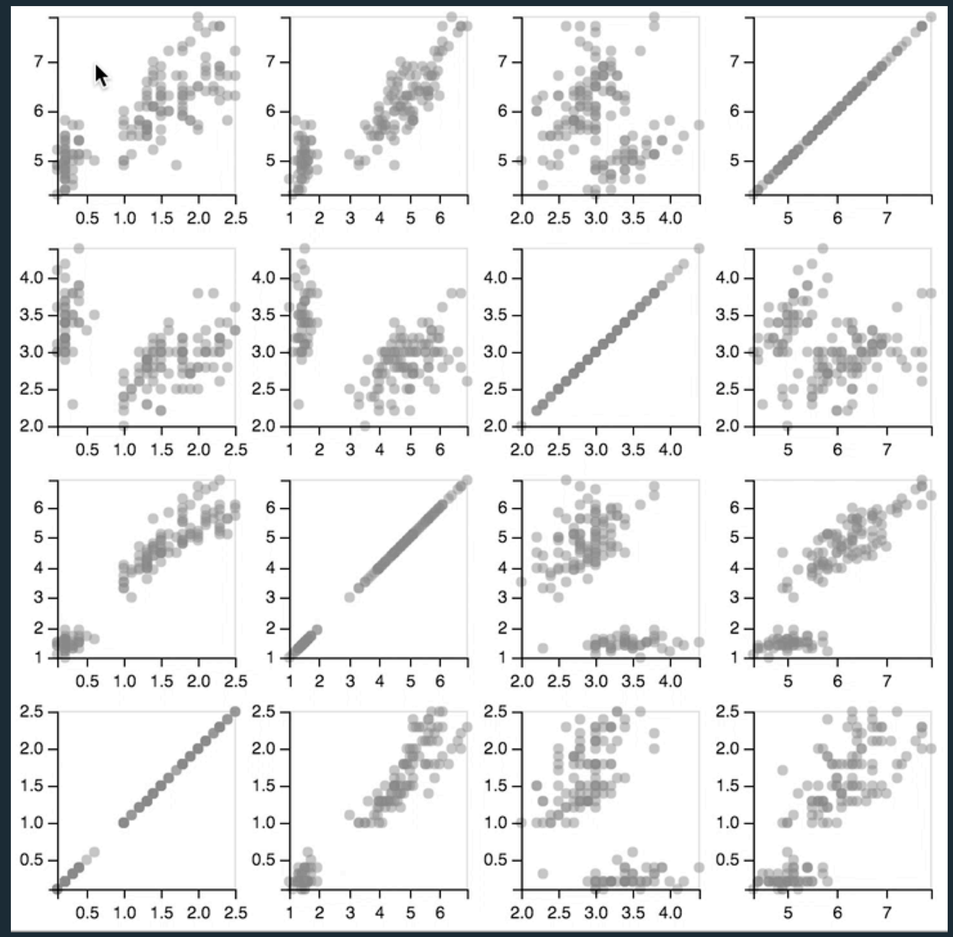
Rect
Mark

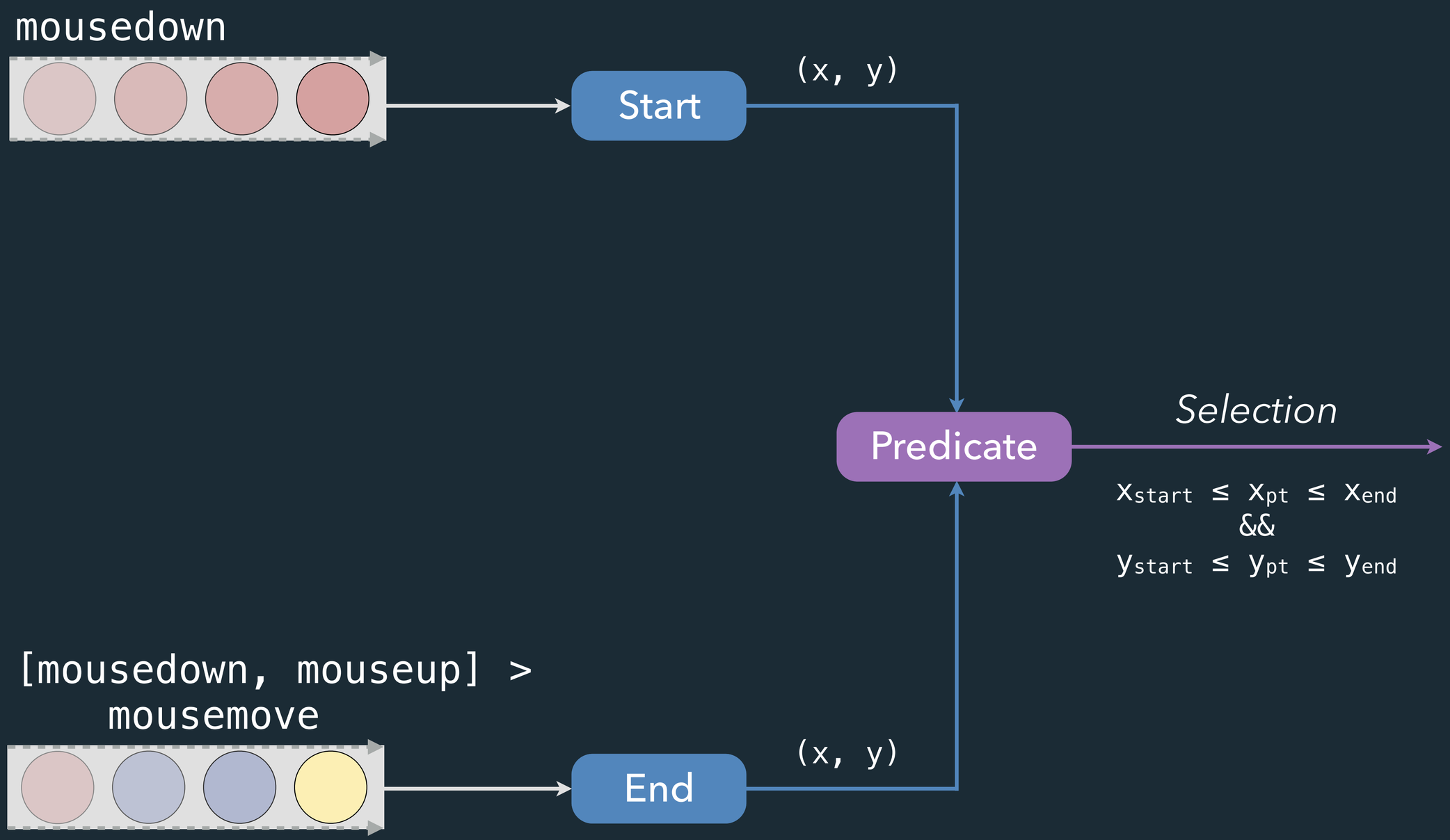
[mousedown, mouseup] >
mousemove

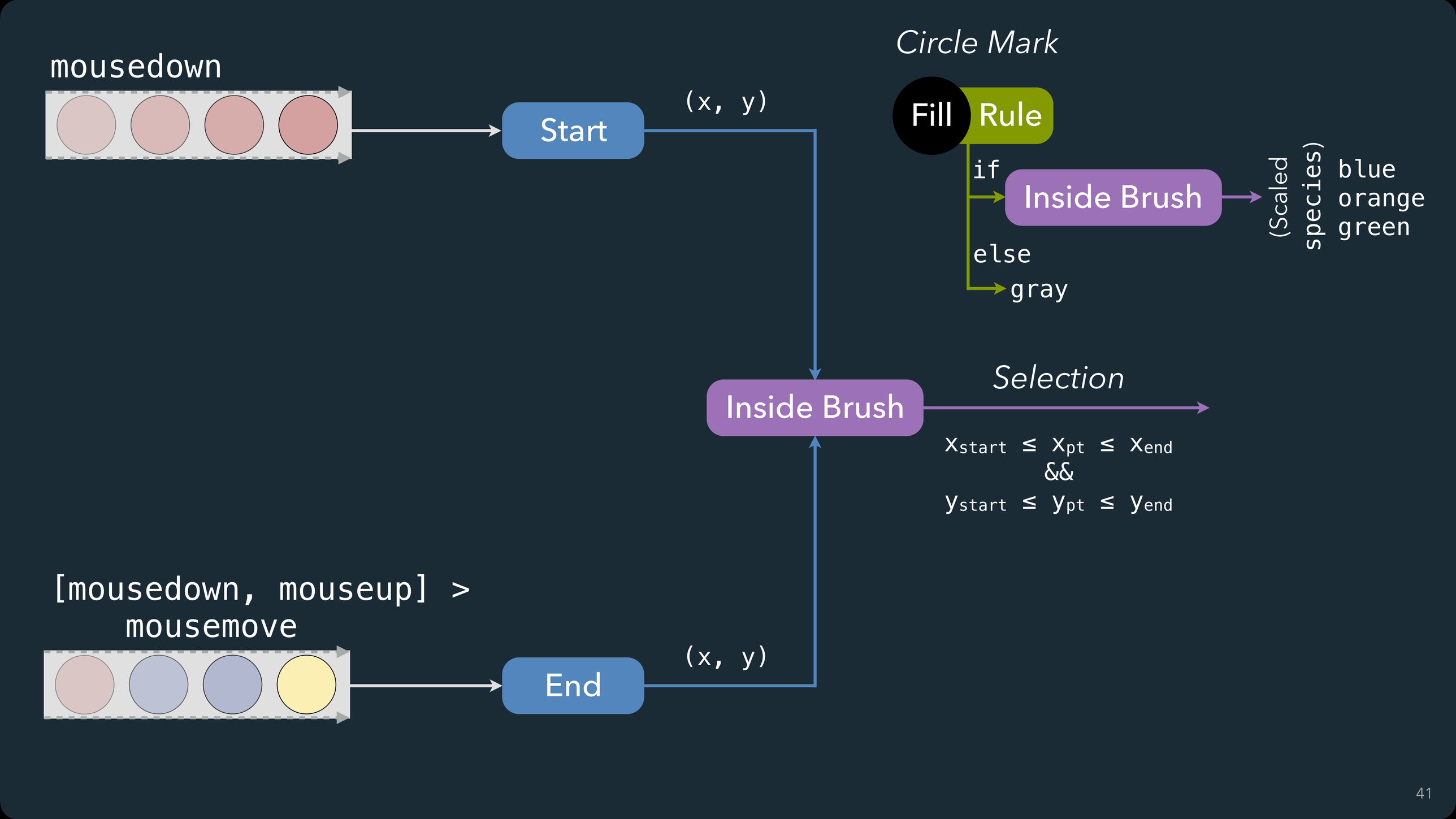


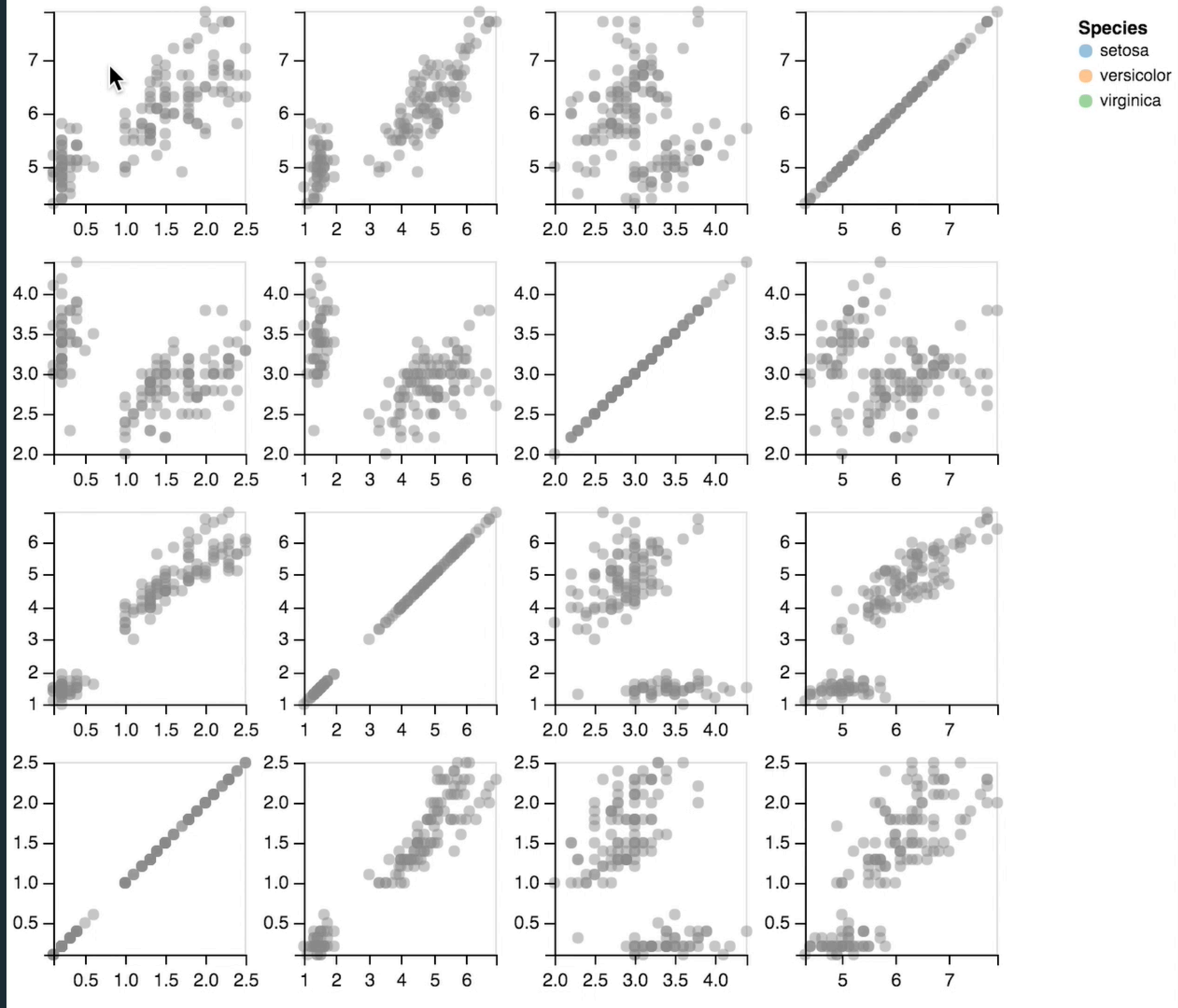
End

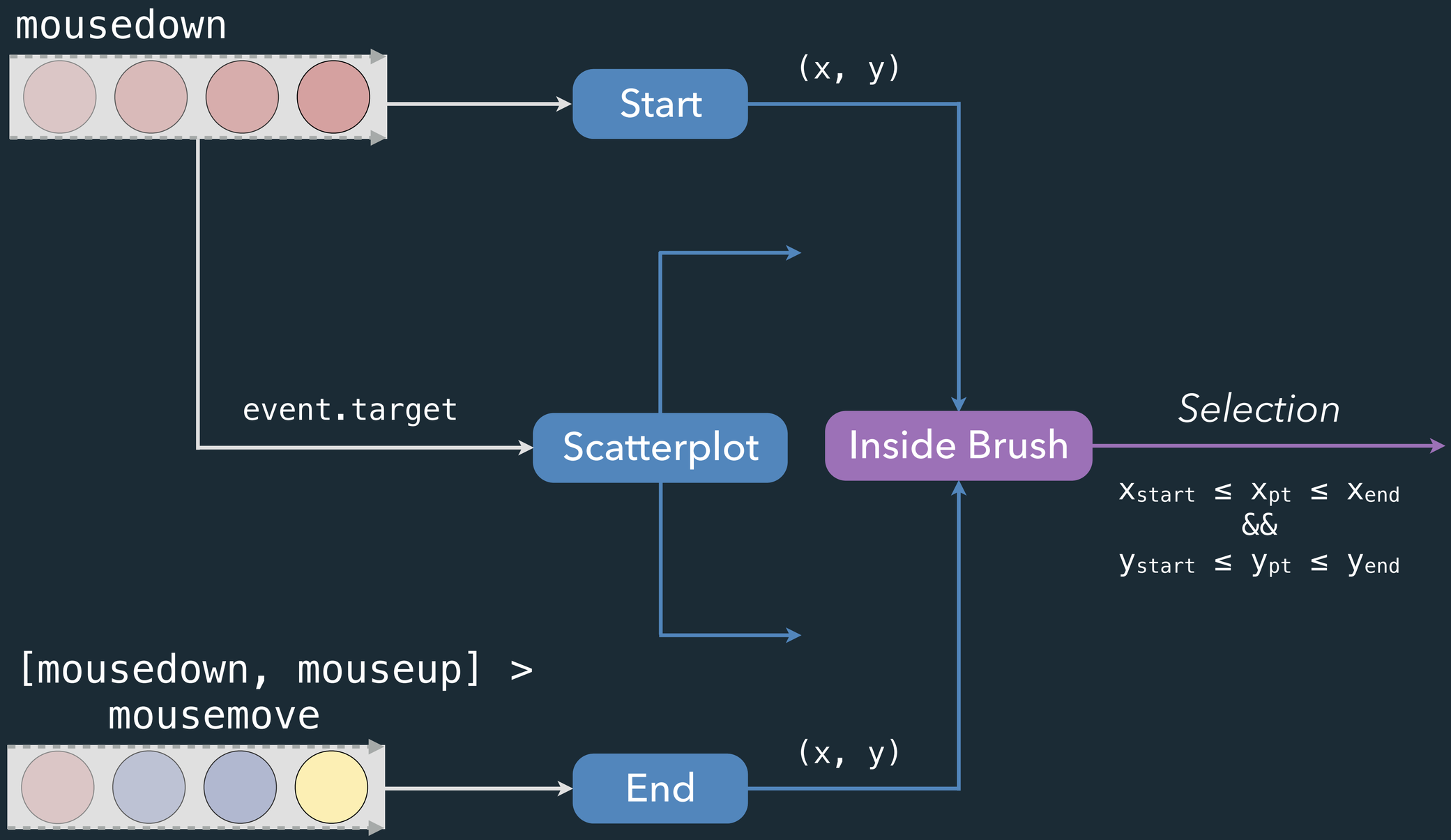
(x, y)

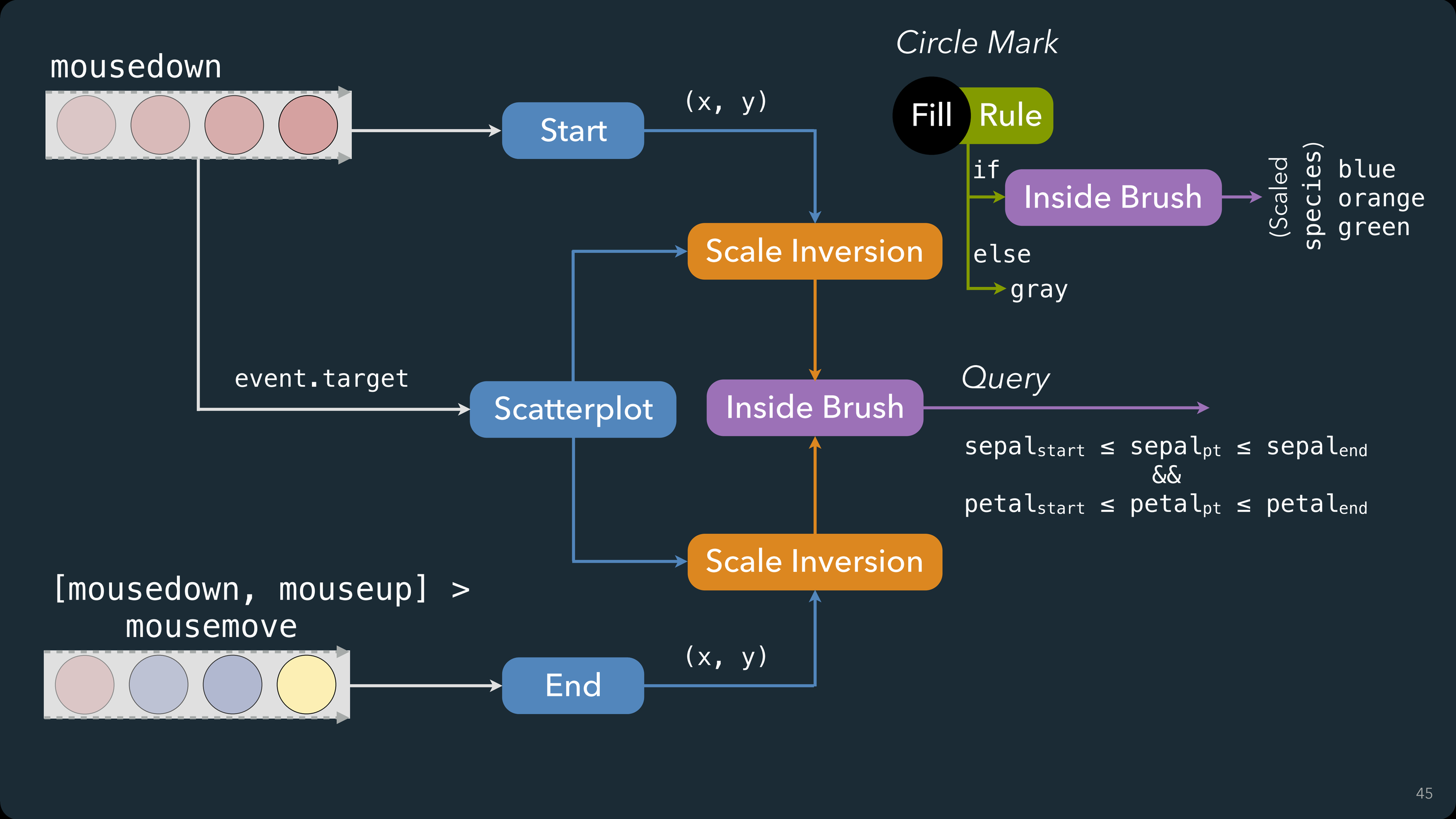


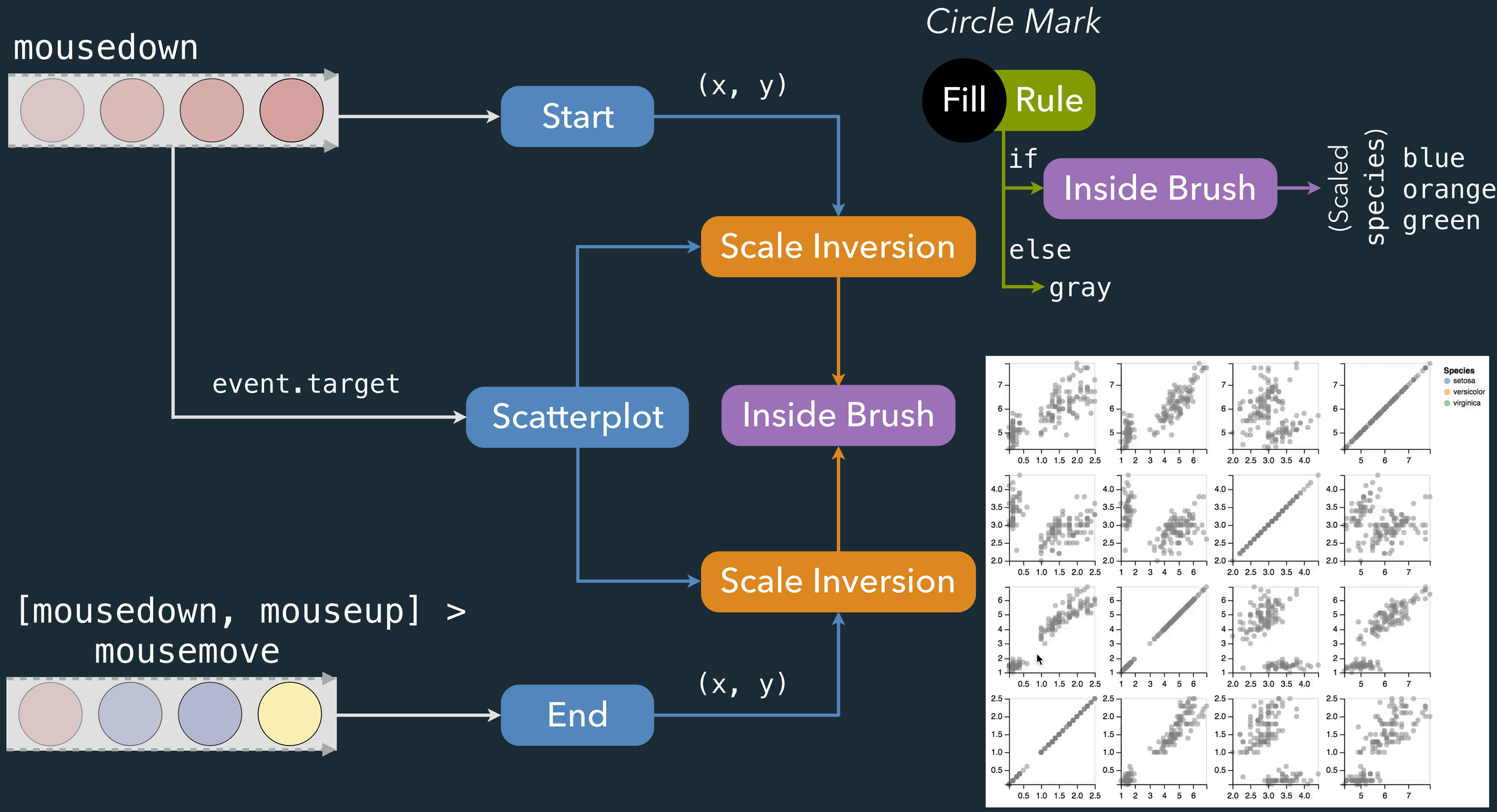

mousedown
Start
(x, y)
Predicate
Selection
$x_{start} \le x_{pt} \le x_{end}$
&&
$y_{start} \le y_{pt} \le y_{end}$
[mousedown, mouseup] >
mousemove
End
(x, y)

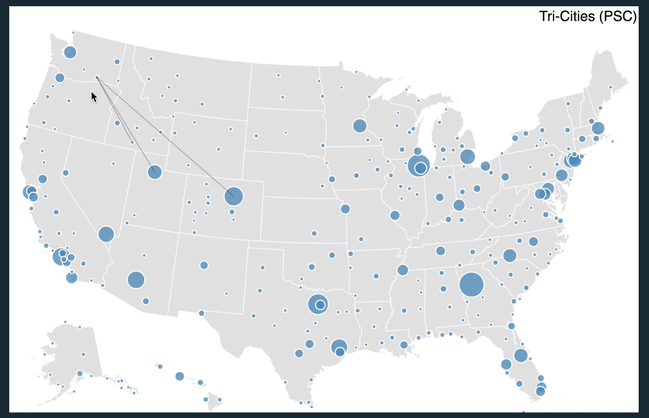

mousedown
Start
(x, y)
Circle Mark
Fill
Rule
if
Inside Brush
(Scaled species)
blue
orange
green
else
gray
Inside Brush
Selection
$x_{start} \le x_{pt} \le x_{end}$
&&
$y_{start} \le y_{pt} \le y_{end}$
[mousedown, mouseup] > mousemove
End
(x, y)

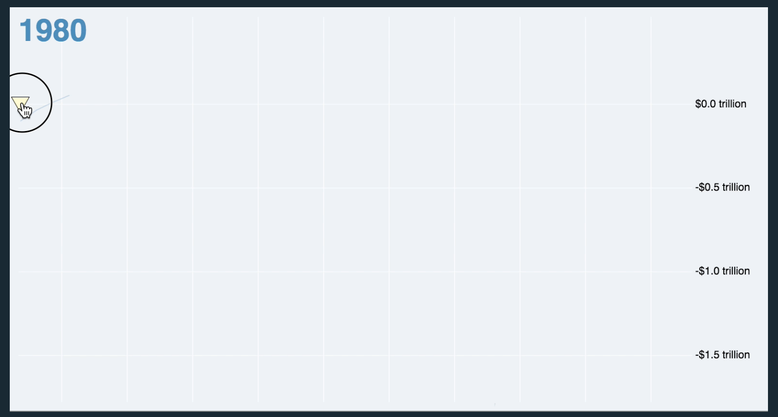

Species
setosa
versicolor
virginica

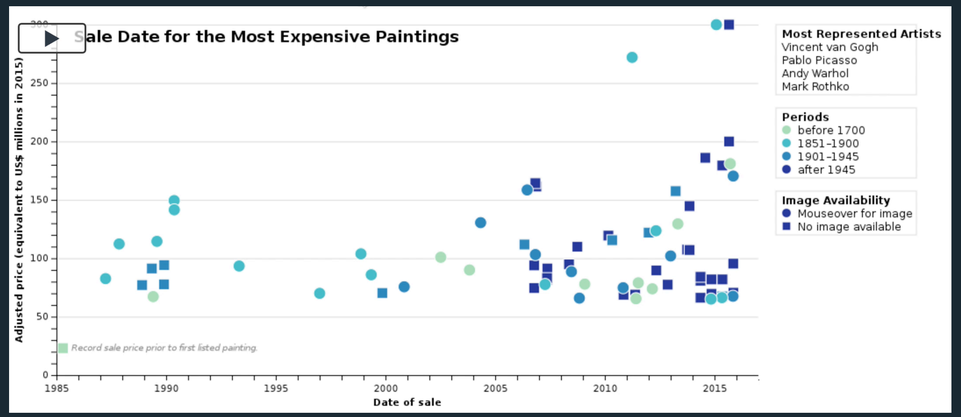

mousedown
Start
(x, y)
event.target
Scatterplot
Inside Brush
Selection
$x_{start} \le x_{pt} \le x_{end}$
&&
$y_{start} \le y_{pt} \le y_{end}$
[mousedown, mouseup] >
mousemove
End
(x, y)

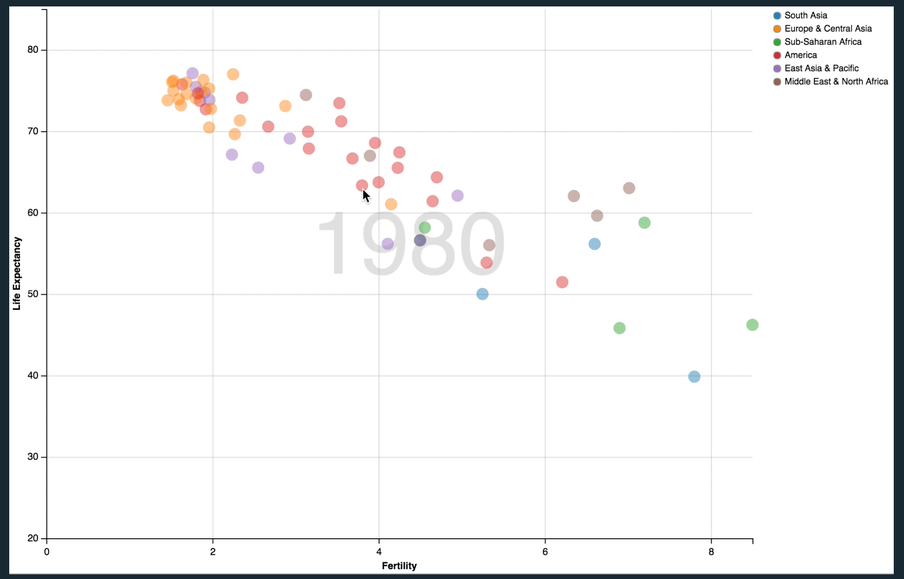

mousedown
Start
(x, y)
Circle Mark
Fill
Rule
if
Inside Brush
(Scaled species)
blue
orange
green
else
gray
Scale Inversion
event.target
Scatterplot
Inside Brush
Query
sepal_start ≤ sepal_pt ≤ sepal_end
&&
petal_start ≤ petal_pt ≤ petal_end
Scale Inversion
[mousedown, mouseup] > mousemove
(x, y)
End

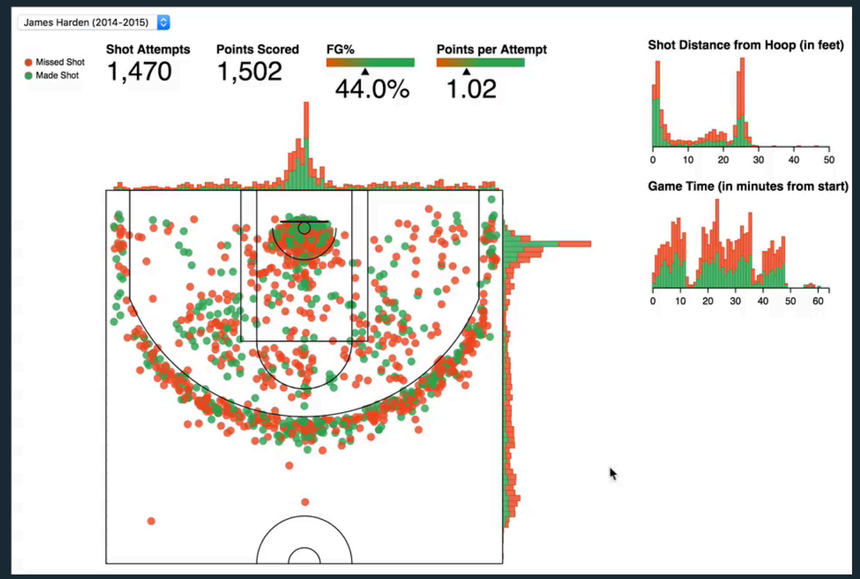

mousedown
Start
(x, y)
Scale Inversion
event.target
Scatterplot
Inside Brush
Scale Inversion
[mousedown, mouseup] >
mousemove
End
(x, y)
Circle Mark
Fill
Rule
if
Inside Brush
(Scaled species)
blue
orange
green
else
gray
Species
setosa
versicolor
virginica

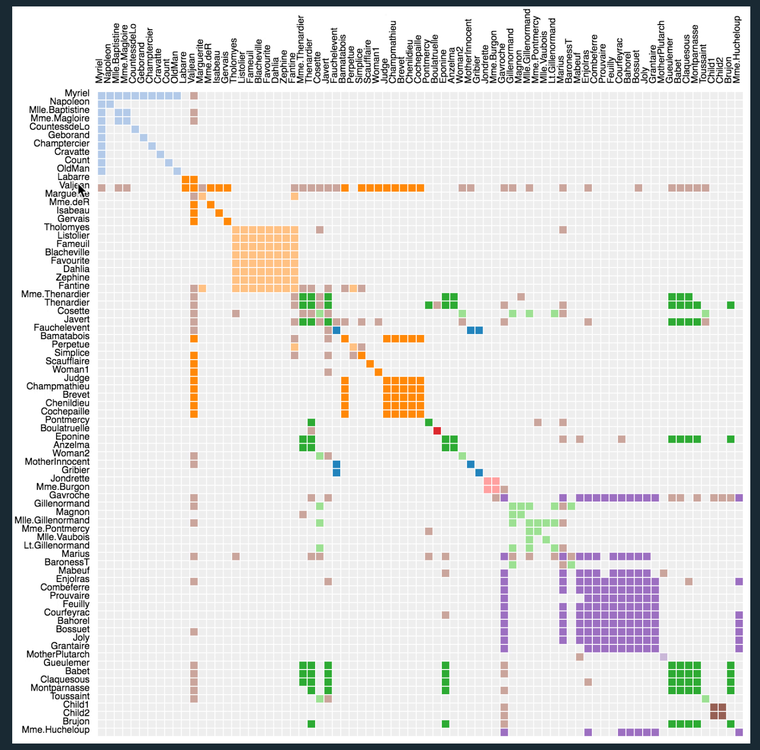


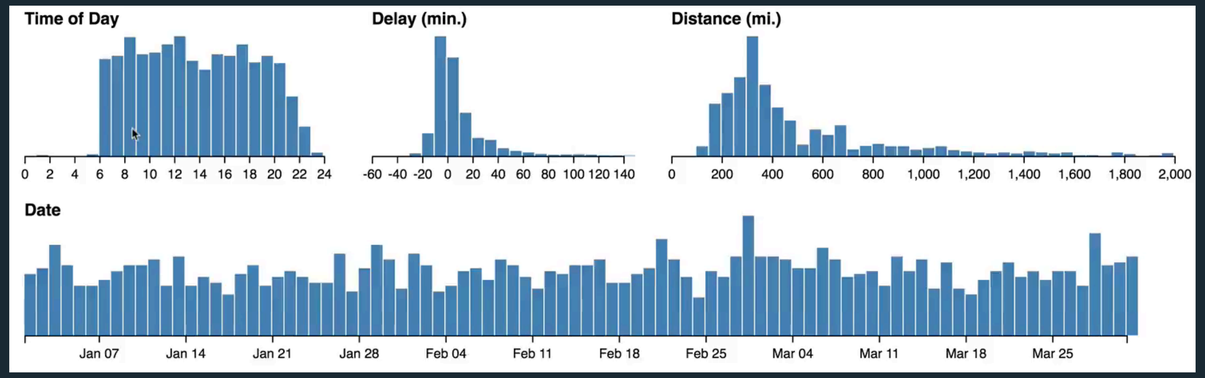












http://vega.github.io/vega-editor/

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Signals provide **abstraction barrier**. Re-target by providing alternate event streams without affecting downstream logic.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Signals provide **abstraction barrier**. Re-target by providing alternate event streams without affecting downstream logic.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

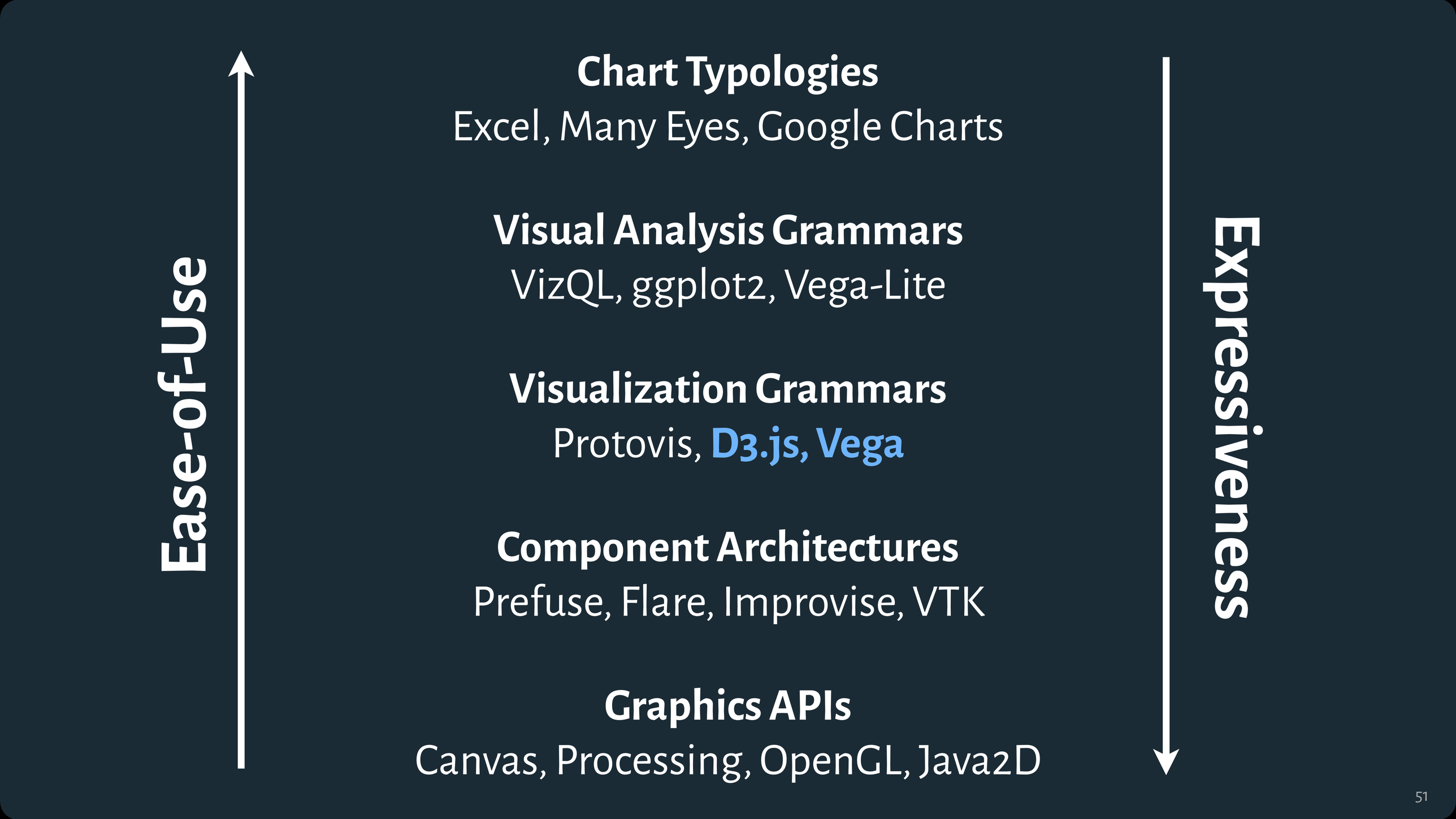
Vega's Interaction Primitives: Advantages

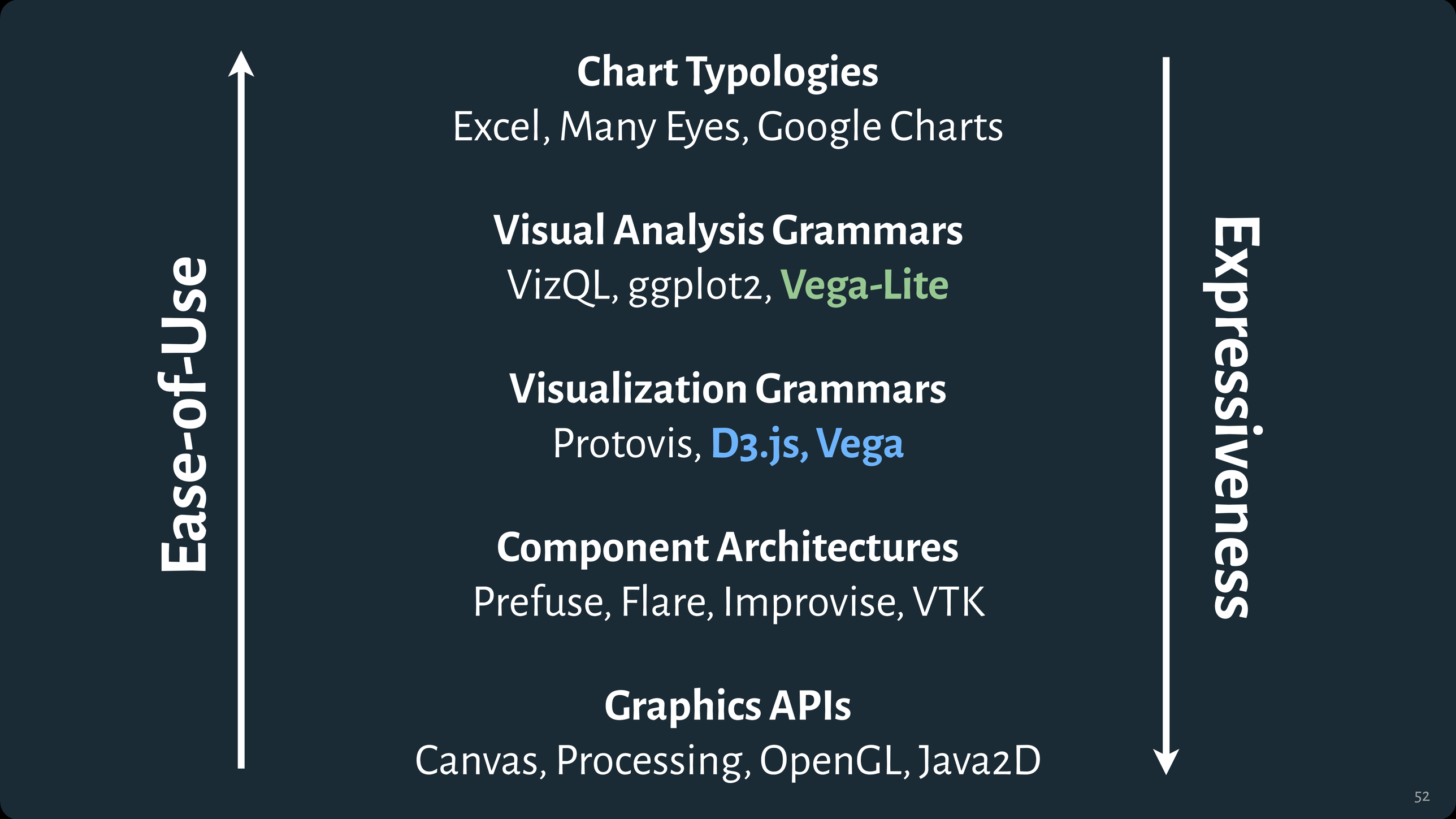
High **expressive ceiling**.

Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Signals provide **abstraction barrier**. Re-target by providing alternate event streams without affecting downstream logic.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*


Ease-of-Use
Chart Typologies
Excel, Many Eyes, Google Charts
Visual Analysis Grammars
VizQL, ggplot2, Vega-Lite
Visualization Grammars
Protovis, D3.js, Vega
Component Architectures
Prefuse, Flare, Improvise, VTK
Graphics APIs
Canvas, Processing, OpenGL, Java2D
Expressiveness


Ease-of-Use
Chart Typologies
Excel, Many Eyes, Google Charts
Visual Analysis Grammars
VizQL, ggplot2, Vega-Lite
Visualization Grammars
Protovis, D3.js, Vega
Component Architectures
Prefuse, Flare, Improvise, VTK
Graphics APIs
Canvas, Processing, OpenGL, Java2D
Expressiveness

Data

Transforms

Scales

Guides

Marks

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

Data

Transforms

Scales

Guides

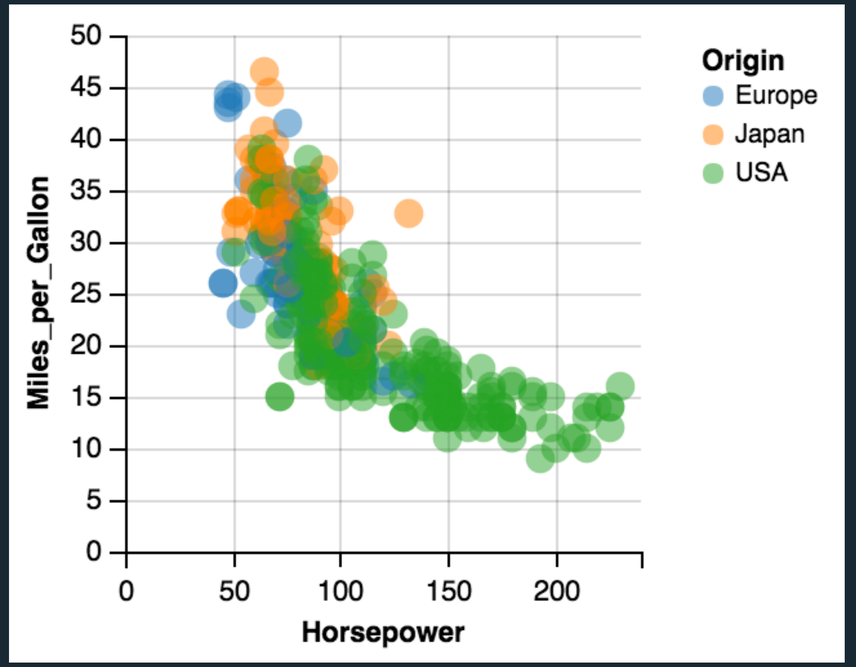
Marks

Encoding Channels

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

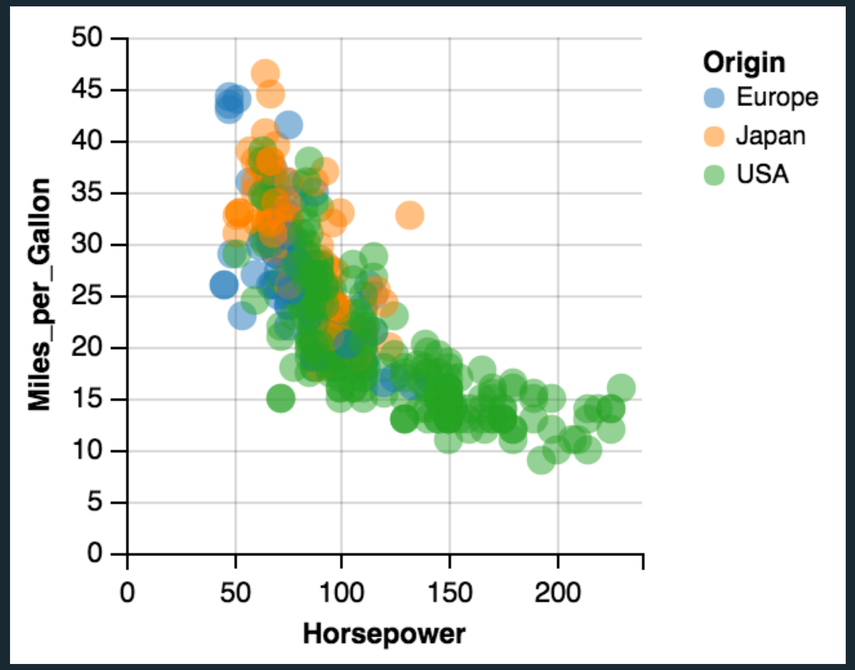
Vega-Lite Encoding Channels

```
{
  "data": {"url": "data/cars.json"},
  "mark": "circle",
  "encoding": {
    "x": {"field": "Horsepower", "type": "Q"},
    "y": {"field": "Miles_per_Gallon", "type": "Q"},
    "color": {"field": "Origin", "type": "N"}
  }
}
```



Vega-Lite compiles to Vega

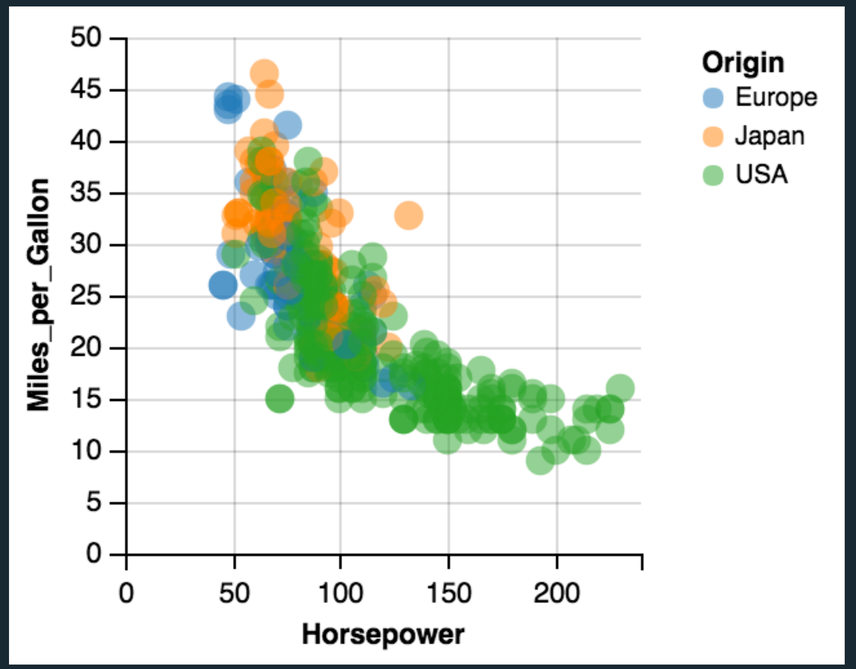
```
{
  "data": {"url": "data/cars.json"},
  "mark": "circle",
  "encoding": {
    "x": {"field": "Horsepower", "type": "Q"},
    "y": {"field": "Miles_per_Gallon", "type": "Q"},
    "color": {"field": "Origin", "type": "N"}
  }
}
```




```
{
  "width": 200, "height": 200,

  "data": [{"name": "cars", "url": "data/cars.json"}],

  "scales": [{
      "name": "x", "type": "linear",
      "domain": {"data": "cars", "field": "Horsepower"},
      "range": "width"
    },
    {"name": "y", "type": "linear", ...},
    {"name": "c", "type": "ordinal", ...}
  ],

  "axes": [{"type": "x", "scale": "x", ...}, ...],

  "legends": [{"fill": "c", ...}],

  "marks": [{
      "type": "symbol",
      "from": {"data": "cars"},
      "properties": {
        "enter": {
          "x": {"scale": "x", "field": "Horsepower"},
          "y": {"scale": "y", "field": "Miles_Per_Gallon"},
          "fill": {"scale": "c", "field": "Origin"},
          "fillOpacity": {"value": 0.5},
          "size": {"value": 100}
        }
      }
  }]
}
```

Vega-Lite Encoding Channels

```
{
  "data": {"url": "data/cars.json"},
  "mark": "circle",
  "encoding": {
    "x": {"field": "Horsepower", "type": "Q"},
    "y": {"field": "Miles_per_Gallon", "type": "Q"},
    "color": {"field": "Origin", "type": "N"}
  }
}
```

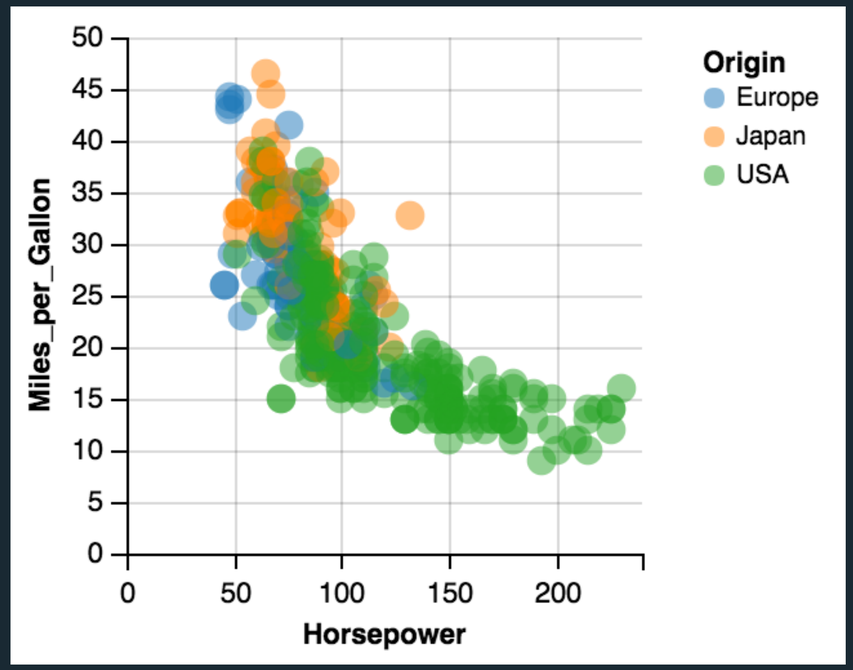


Concise by **omitting low-level details** (e.g., scale and guide definitions).

Compiler fills in **sensible defaults**. Users can **override them** for custom design.

Vega-Lite Encoding Channels

```
data:
  url: data/cars.json
mark: circle
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    field: Origin, type: nominal
```

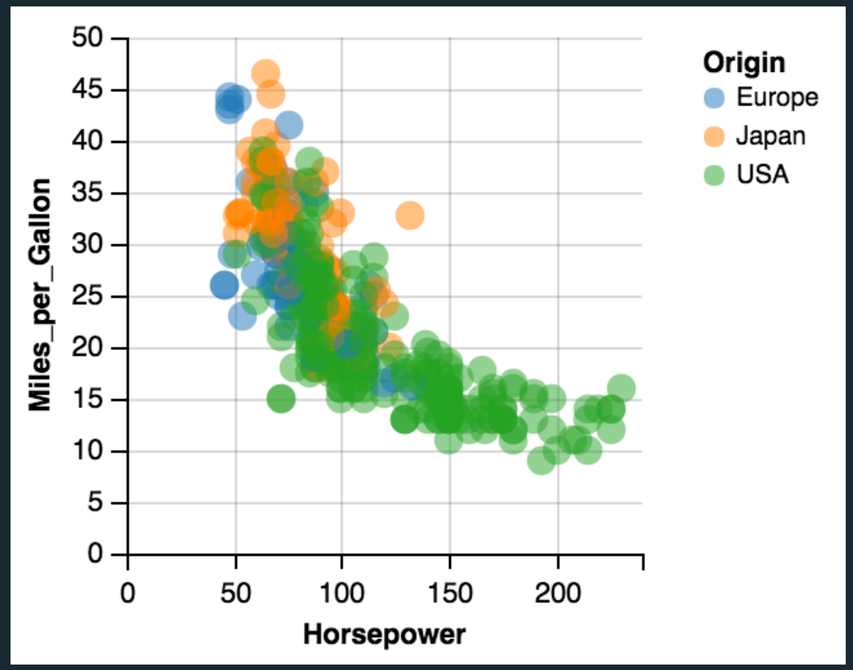


Concise by **omitting low-level details** (e.g., scale and guide definitions).

Compiler fills in **sensible defaults**. Users can **override them** for custom design.

Vega-Lite Encoding Channels

```yaml
data:
  url: data/cars.json
mark: circle
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    field: Origin, type: nominal
```

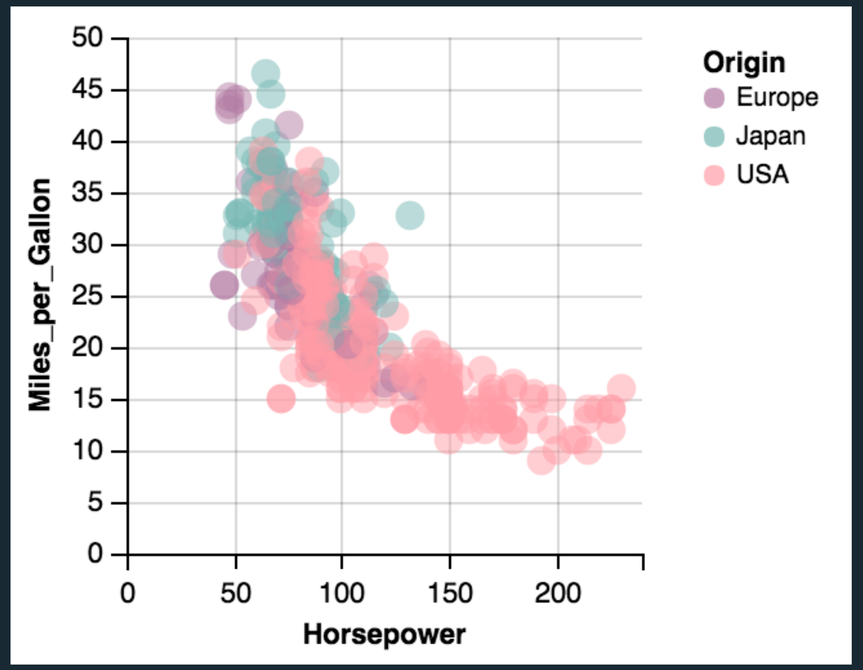


Concise by **omitting low-level details** (e.g., scale and guide definitions).

Compiler fills in **sensible defaults**. Users can **override them** for custom design.

Vega-Lite Encoding Channels

```yaml
data:
  url: data/cars.json
mark: circle
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    field: Origin, type: nominal
    scale:
      range: ["#b07aa1", "#76b7b2", "#ff9da7"]
```



Concise by **omitting low-level details** (e.g., scale and guide definitions).

Compiler fills in **sensible defaults**. Users can **override them** for custom design.

Data

Transforms

Scales

Guides

Marks

} Encoding Channels

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

Data

Transforms

Encoding Channels

Event Streams

Signals

Scale Inversions

Predicates

Production Rules

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

Data

Transforms

Encoding Channels

Event Streams

Signals

Scale Inversions

Predicates

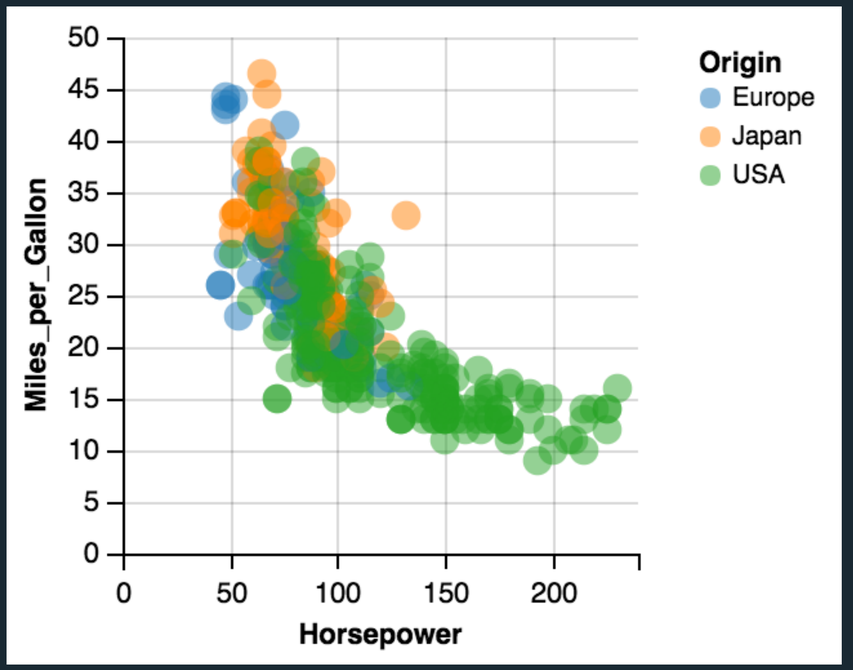
Production Rules

Selections

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

Vega-Lite Selections

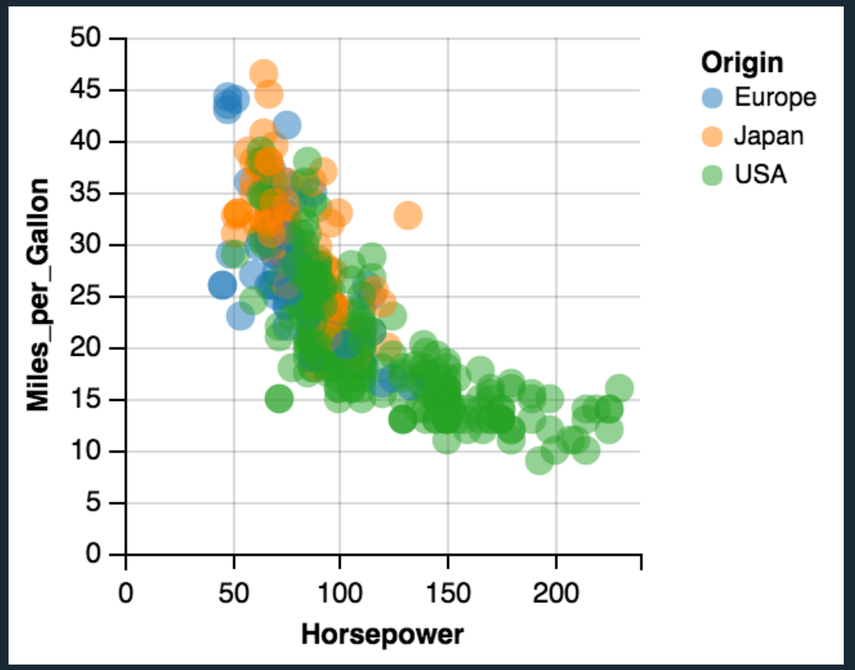
```
data:
  url: data/cars.json
mark: circle
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    field: Origin, type: nominal
```



Selections define event processing, signals, and a predicate function.

Vega-Lite Selections: A Single Point

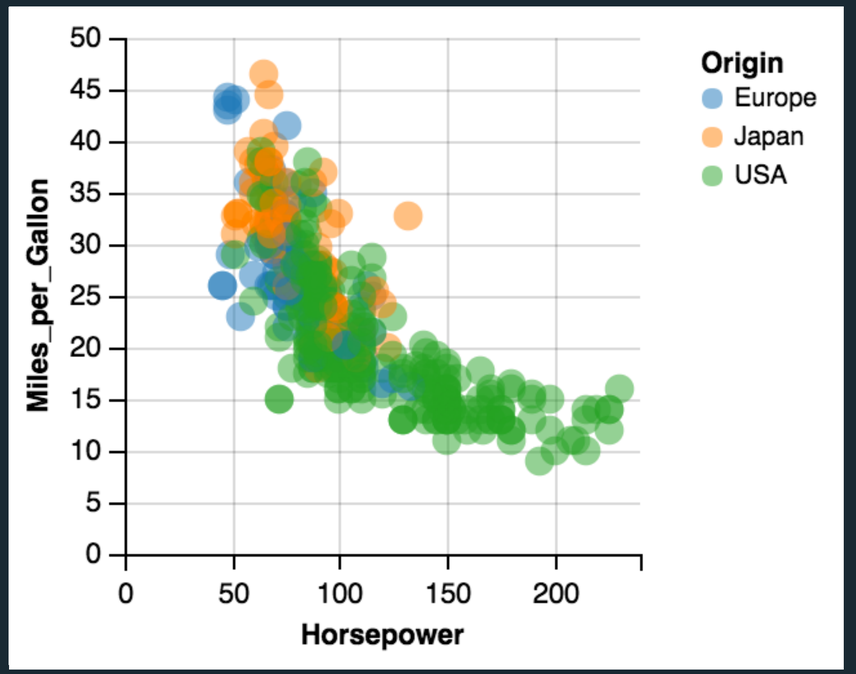
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    field: Origin, type: nominal
```



Selections define event processing, signals, and a predicate function.

Vega-Lite Selections: A Single Point

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```

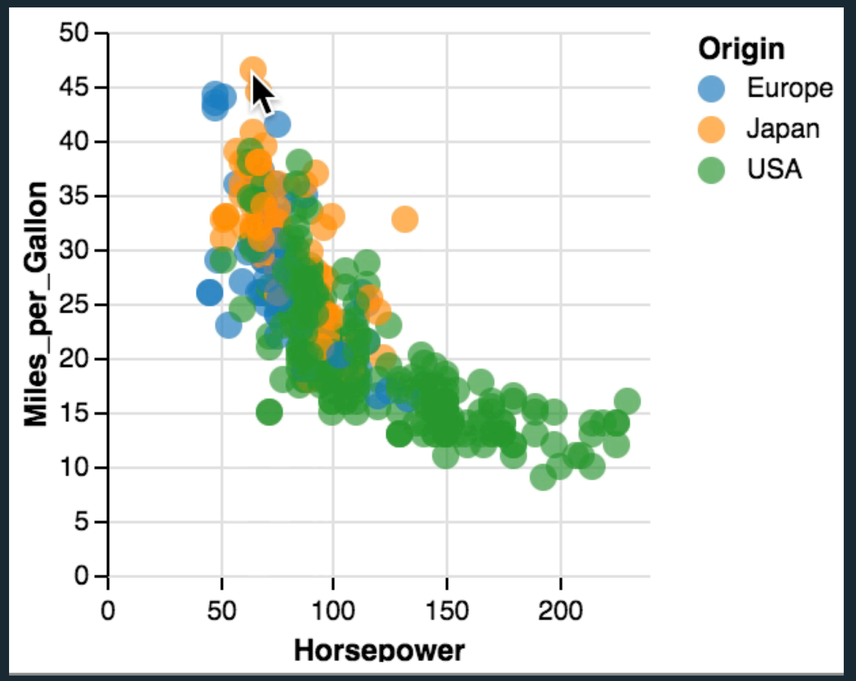


Selections define event processing, signals, and a predicate function.

Selection **types** provide defaults values for these components.

Vega-Lite Selections: Multiple Points

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: multi
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```

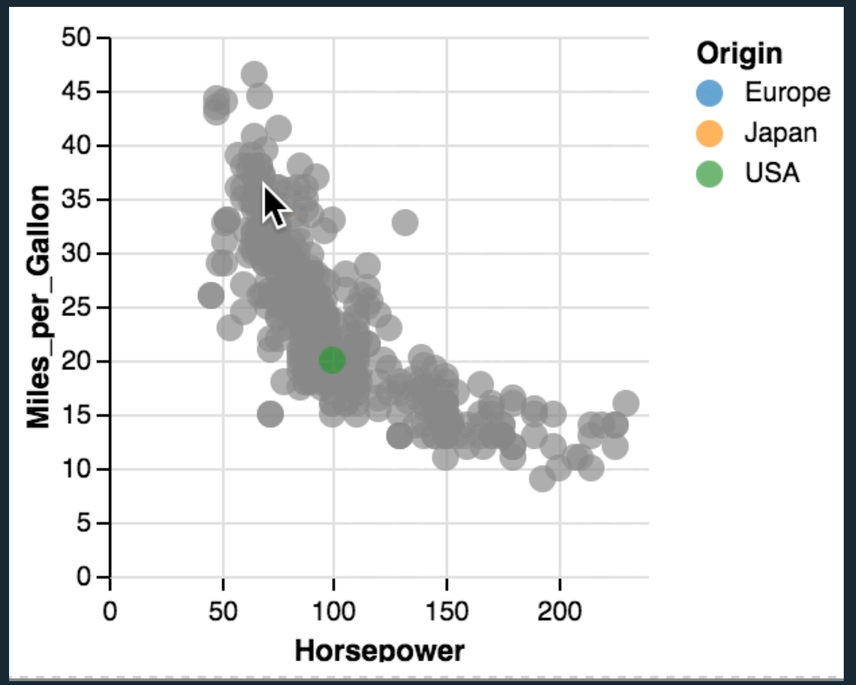


Selections define event processing, signals, and a predicate function.

Selection **types** provide defaults values for these components.

Vega-Lite Selections: Multiple Points on hover

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: multi, on: mouseover
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



Selections define event processing, signals, and a predicate function.

Selection **types** provide defaults values for these components.

Data

Transforms

Encoding Channels

Event Streams

Signals

Scale Inversions

Predicates

Production Rules

Selections

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

Data

Transforms

Encoding Channels

Event Streams

Signals

Scale Inversions

Predicates

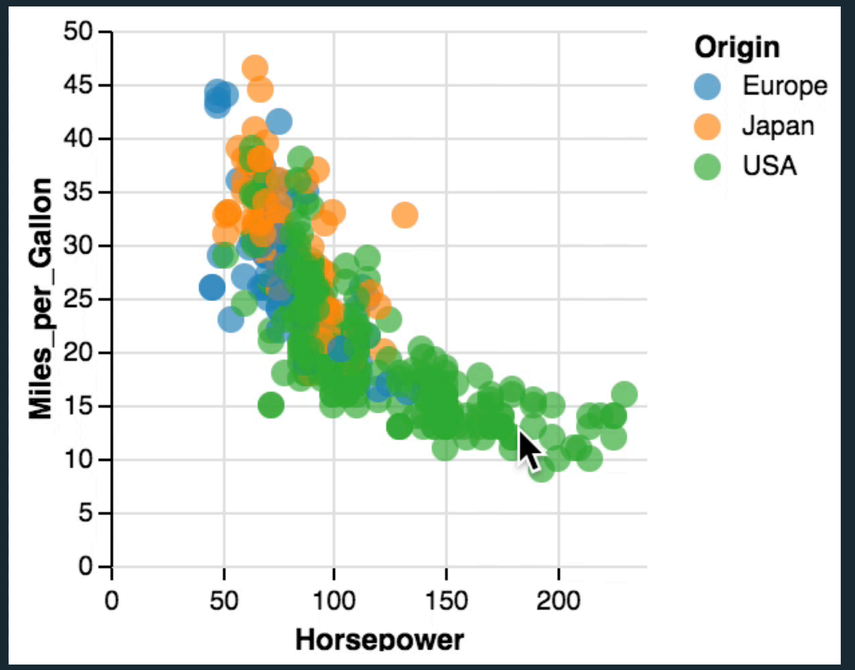
Production Rules

Selections +
Selection Transforms

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

Vega-Lite Selections: A Single Cylinder

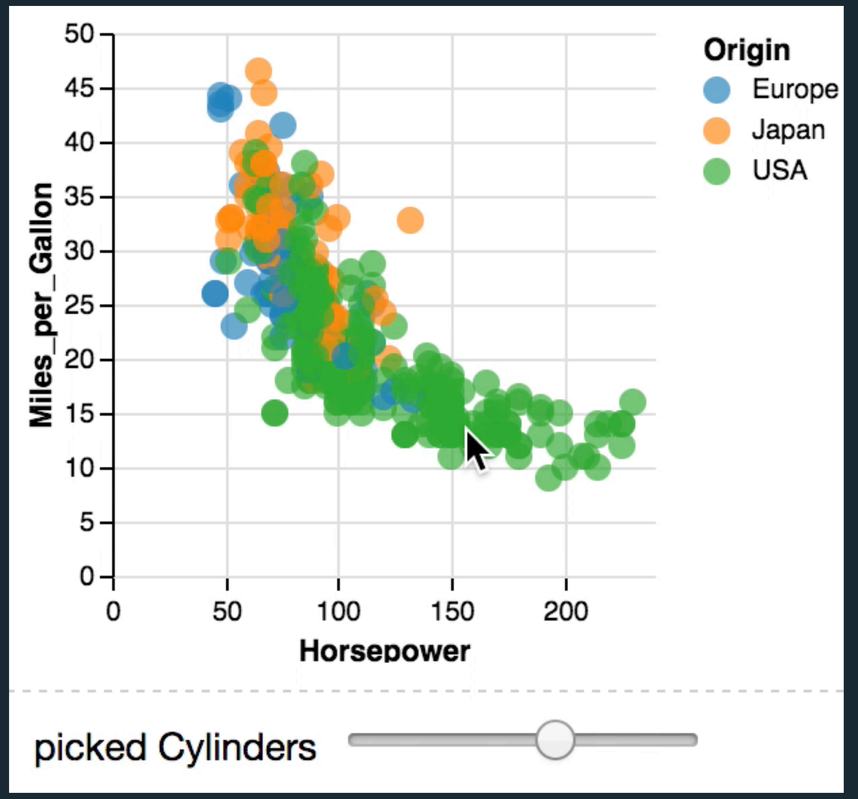
```yaml
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
    fields:
      - Cylinders
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



The **project transform** rewrites the predicate to match on `fields` or encodings.

Vega-Lite Selections: Bound Single Cylinder

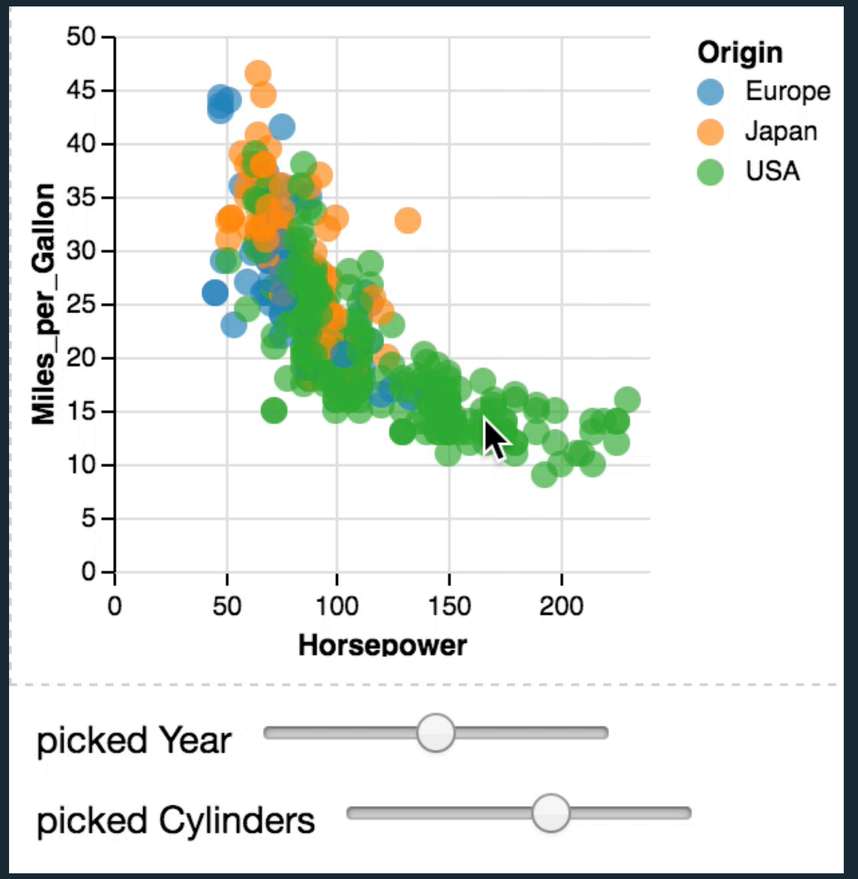
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
    fields:
      - Cylinders
    bind:
      input: range, min: 3, max: 8, step: 1
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
```



The **bind transform** drives selections via query widgets and scale functions.

Vega-Lite Selections: Bound Single Cylinder & Year

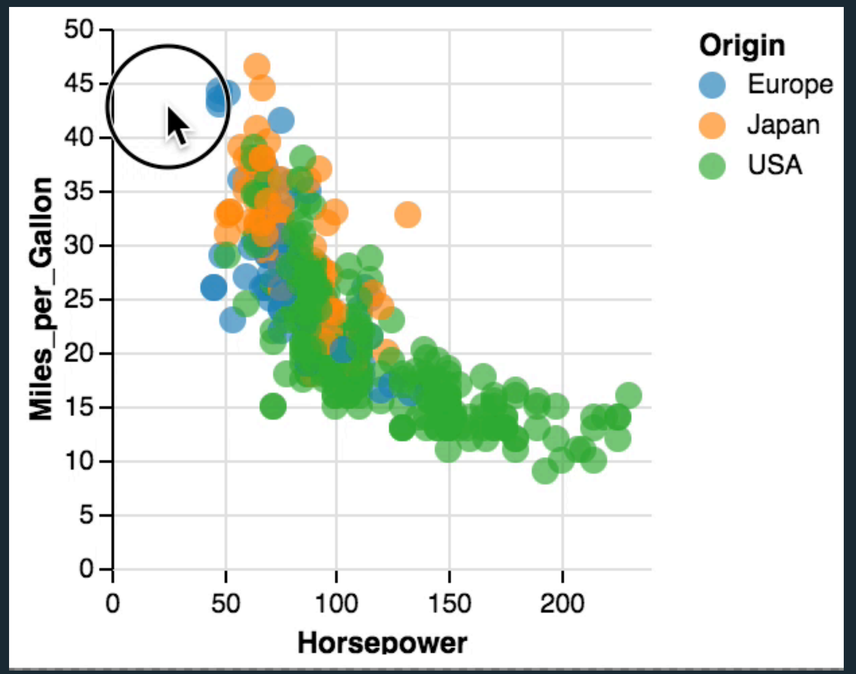
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
    fields:
      - Cylinders
      - Year
    bind:
      Cylinders:
        input: range, min: 3, max: 8, step: 1
      Year:
        input: range, min: 1969, max: 1981, step: 1
encoding:
  x:
    field: Horsepower, type: quantitative
```



The **bind transform** drives selections via query widgets and scale functions.

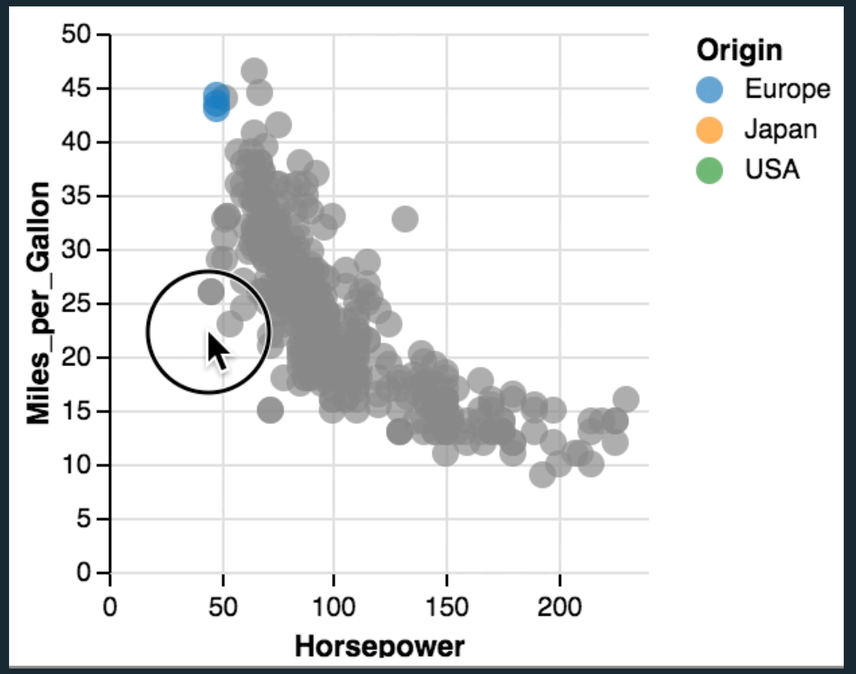
Vega-Lite Selections: Continuous Region

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: interval
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



Vega-Lite Selections: Single-Dimensional Region

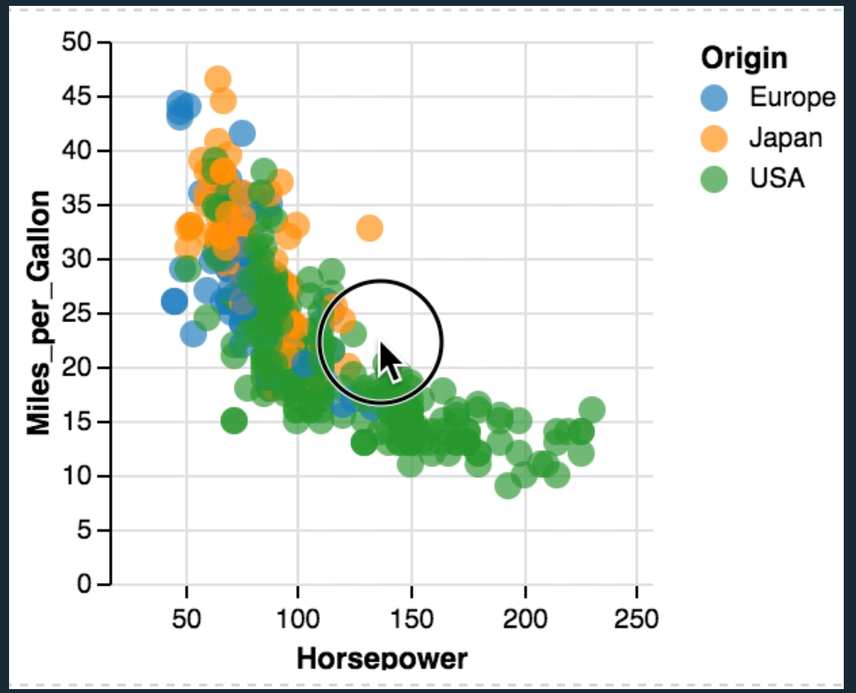
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: interval
    encodings:
      - X
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



The **project transform** rewrites the predicate to match on `fields` or encodings.

Vega-Lite Selections: Bound Single-Dimensional Region

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: interval
    encodings:
      - X
    bind: scales
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



The **bind transform** drives selections via query widgets and scale functions.

Data

Transforms

Encoding Channels

Event Streams

Signals

Scale Inversions

Predicates

Production Rules

Selections +
Selection Transforms

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

Data

Transforms

Encoding Channels

Selections

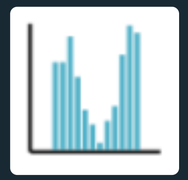
Selection Transforms

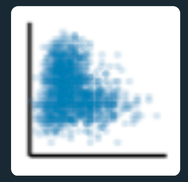
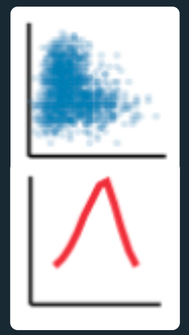
Production Rules

View Composition

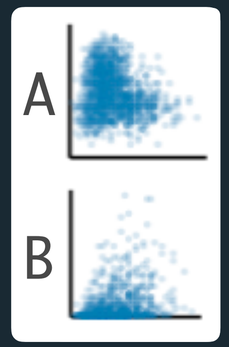
Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

Vega-Lite View Composition

layer: [ , ] = 

vconcat: [ , ] = 

repeat row: [A,B]

 = 

facet row: A

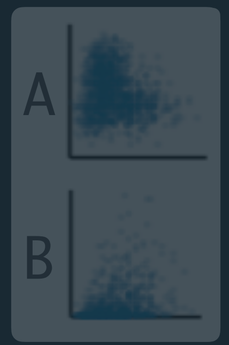
 = 

Vega-Lite View Composition Algebra

layer: [ , ] = 

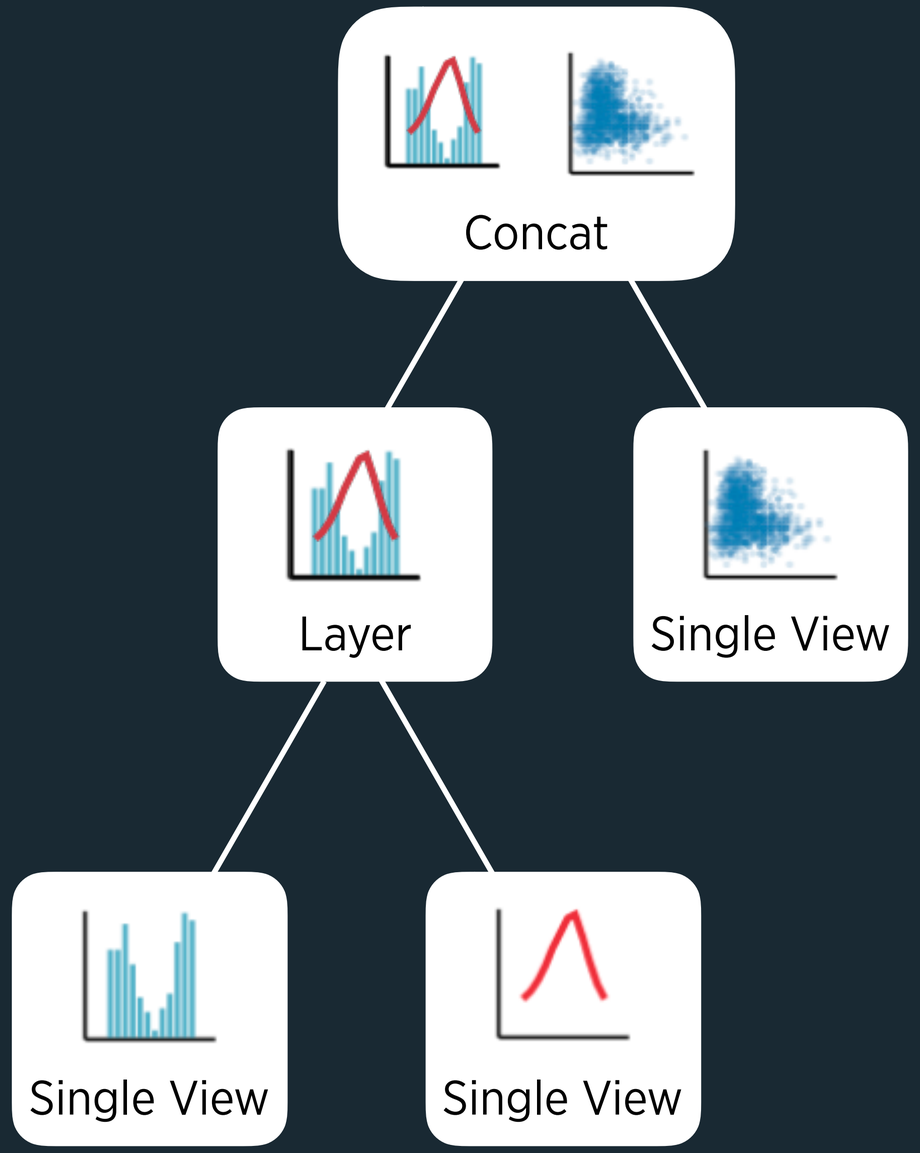
vconcat: [ , ] = 

repeat row: [A,B]

 = 

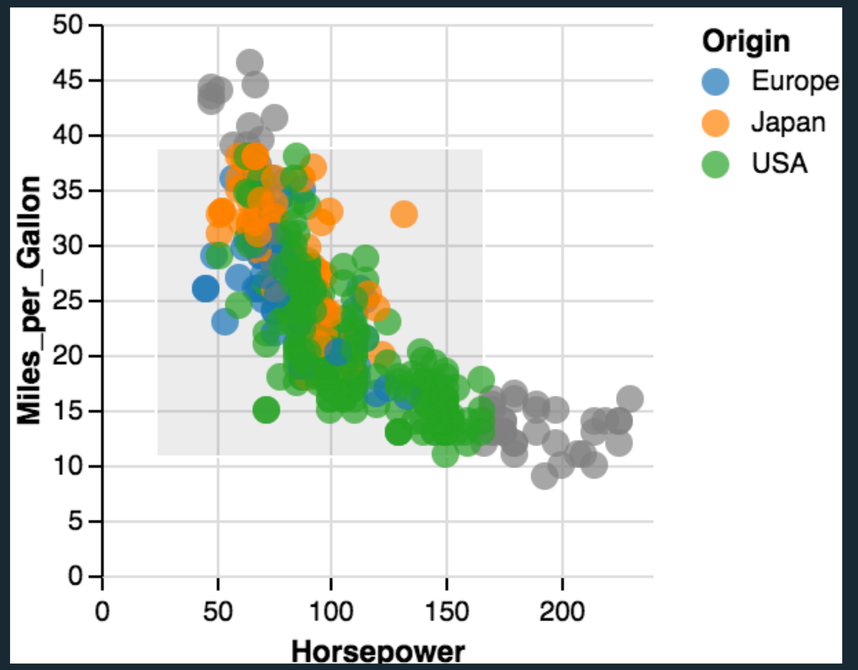
facet row: A

 = 



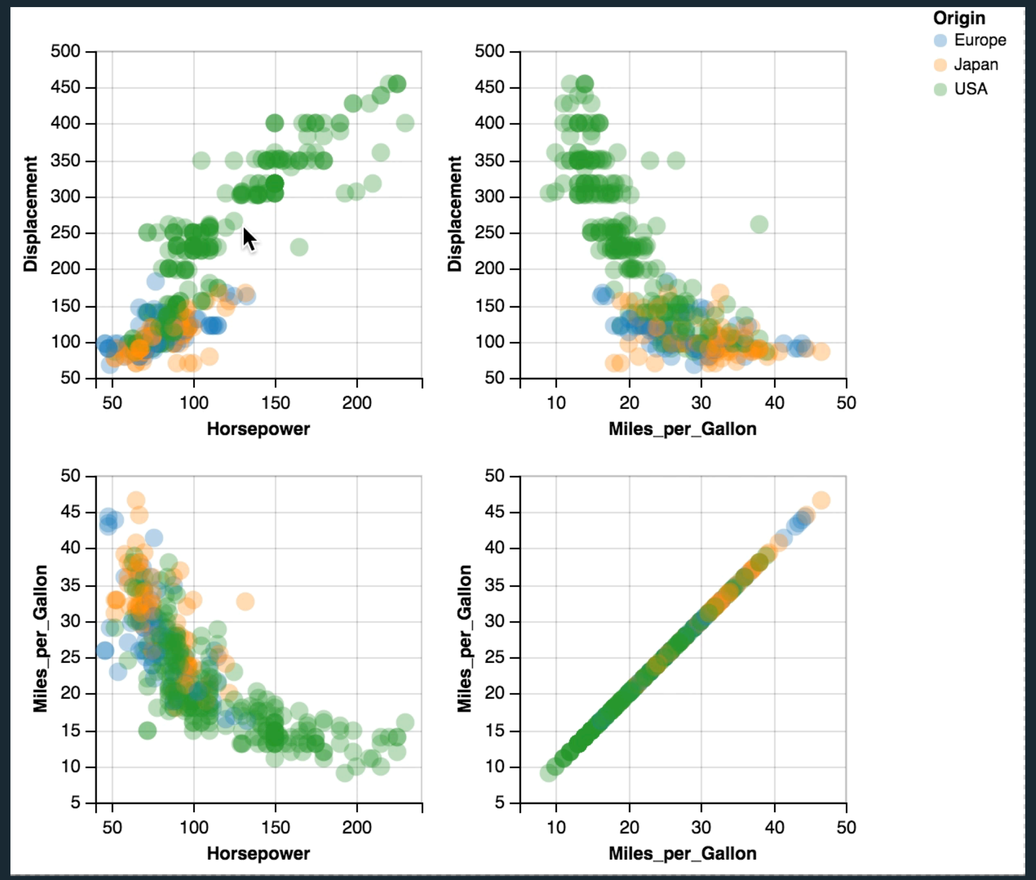
Vega-Lite Selections: Continuous Region

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: interval
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



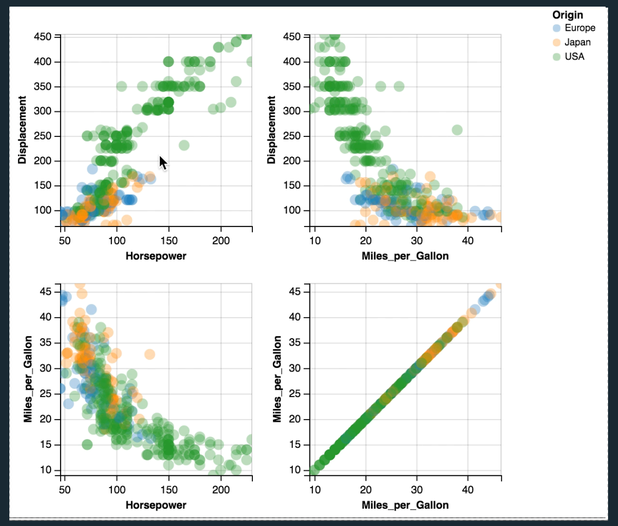
Vega-Lite Brushing & Linking

```
data:
  url: data/cars.json
repeat:
  row:
    - Displacement
    - Miles_per_Gallon
  column:
    - Horsepower
    - Miles_per_Gallon
spec:
  mark: circle
  selection:
    picked:
      type: interval
  encoding:
    x:
      field: Horsepower, type: quantitative
    y:
      field: Miles_per_Gallon, type: quantitative
    color:
      - if: picked, field: Origin, type: nominal
      - value: grey
```

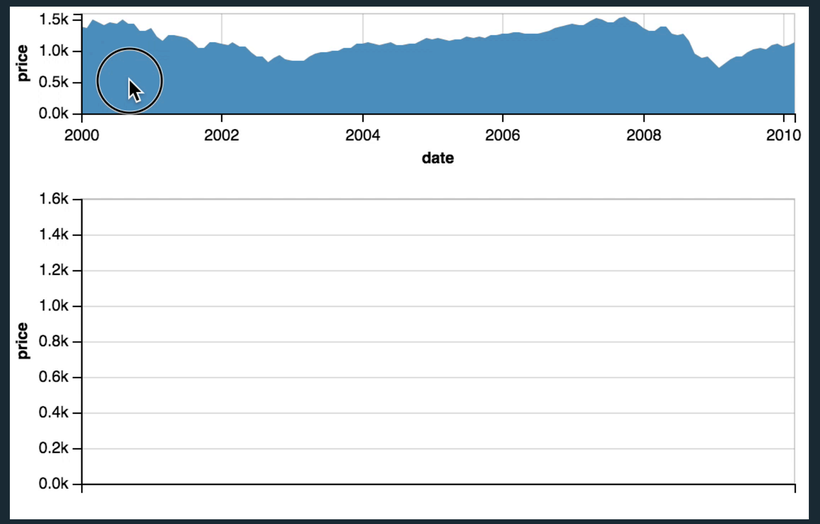


+ Panning & Zooming

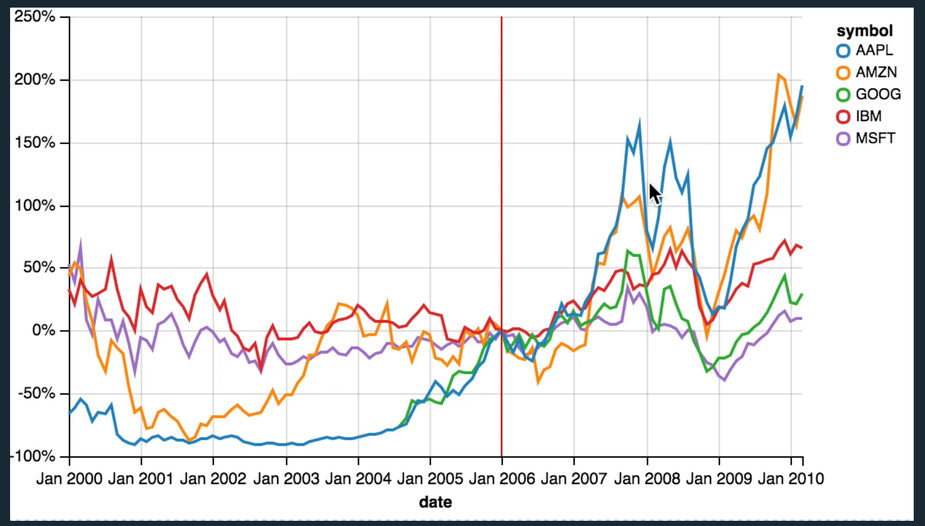
Brushing & Linking



Overview + Detail



Interactive Re-Normalization



Layered Cross Filtering