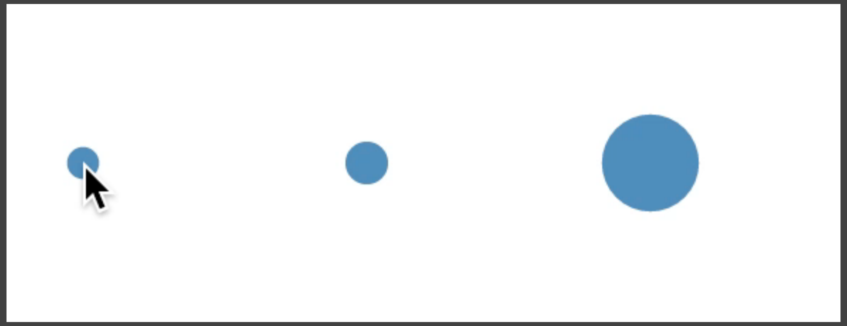
Reactive Building Blocks

Interactive Visualizations with Vega

Arvind Satyanarayan @arvindsatya1
Stanford University

Jane Hoffswell
Dominik Moritz @domoritz
Kanit "Ham" Wongsuphasawat @kanitw
Jeffrey Heer @jeffrey_heer
University of Washington

Three Little Circles*



Visual Design

```
var circle = svg.selectAll('circle')
      .data([32, 57, 293]);

  circle.enter().append('circle')
      .attr('fill', 'steelblue')
      .attr('cy', 60)
      .attr('cx',
        function(d, i) { return i * 100 + 30; })
      .attr('r',
        function(d) { return Math.sqrt(d); });
```

Map data values to visual properties.

Declarative design: specify *what* we want, rather than *how* it should be computed.

Interaction Design

```
var dragging = null;
circle.on('mousedown', function() {
  dragging = d3.select(this)
    .attr('fill', 'goldenrod');
});

d3.select(window).on('mouseup', function() {
  dragging.attr('fill', 'steelblue');
  dragging = null;
}).on('mousemove', function() {
  if (!dragging) return;
  dragging.attr('cy', d3.event.pageY);
  d3.event.stopPropagation();
});
```

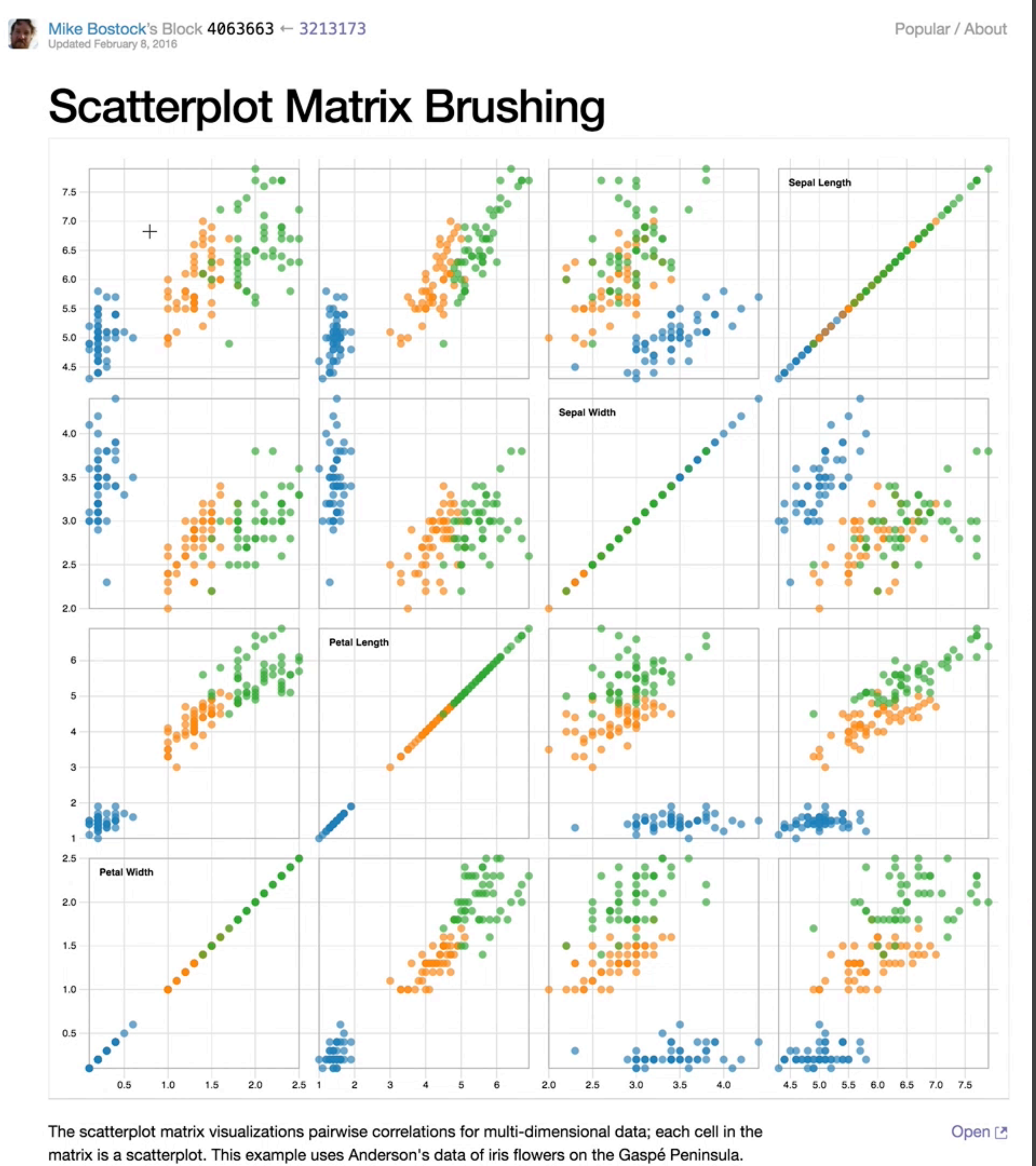
Imperative design: define *explicit steps* of *how* it should be computed.

* Mike Bostock, 2014. https://bost.ocks.org/mike/circles/

The Trouble with Imperative Interaction

```
var dragging = null;
circle.on('mousedown', function() {
  dragging = d3.select(this)
    .attr('fill', 'goldenrod');
});

d3.select(window).on('mouseup', function() {
  dragging.attr('fill', 'steelblue');
  dragging = null;
}).on('mousemove', function() {
  if (!dragging) return;
  dragging.attr('cy', d3.event.pageY);
  d3.event.stopPropagation();
});
```

1. Manually maintain state.
2. Re-define visual appearance in multiple locations.
3. Low-level idiosyncrasies.
4. Callback Hell: unpredictable and interleaved execution.


Mike Bostock's Block 4063663 ← 3213173
Updated February 8, 2016

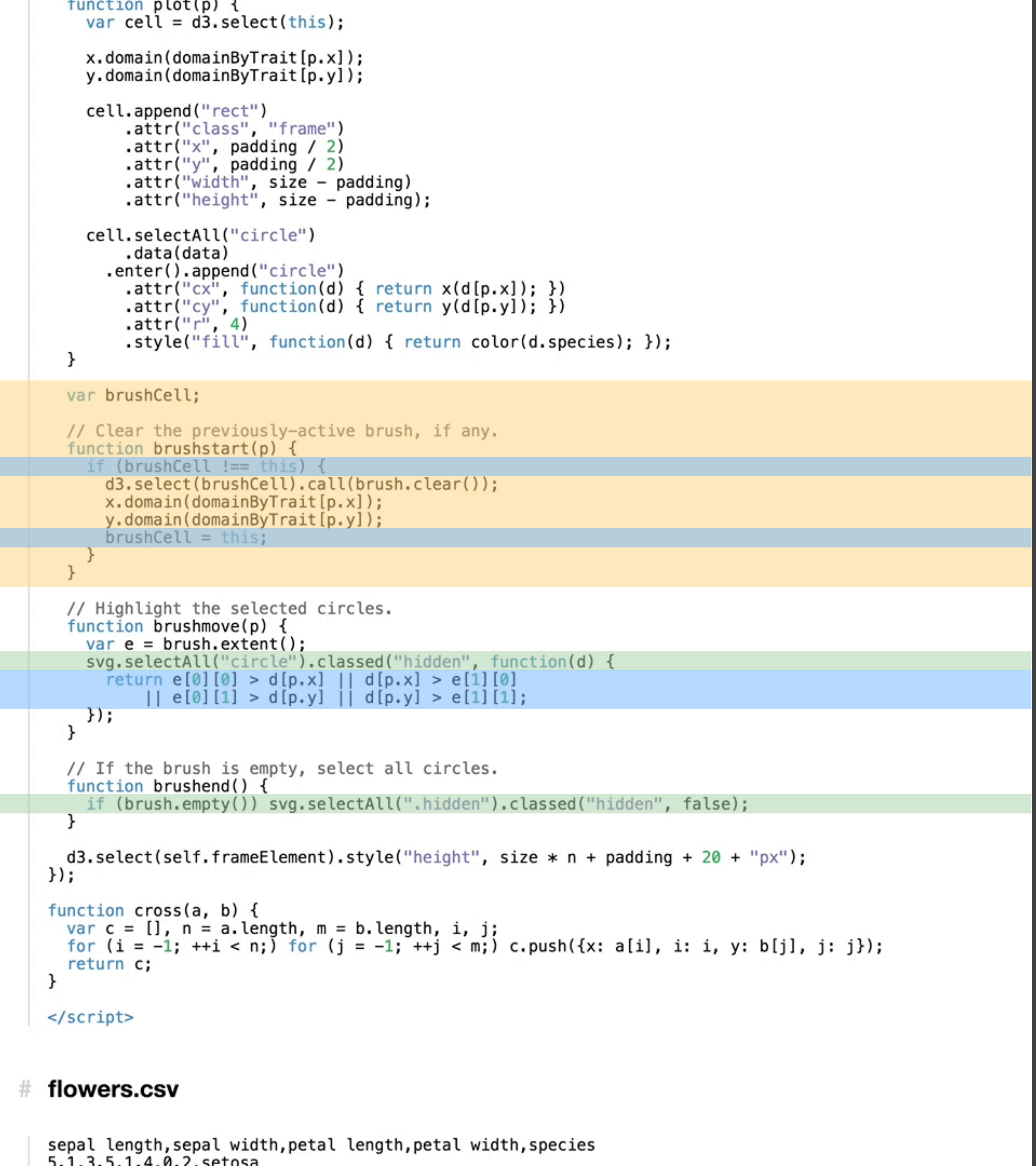
Popular / About

Scatterplot Matrix Brushing

The scatterplot matrix visualizations pairwise correlations for multi-dimensional data; each cell in the matrix is a scatterplot. This example uses Anderson's data of iris flowers on the Gaspé Peninsula.

Open

1. Manually maintain state.
2. Re-define visual appearance in multiple locations.
3. Low-level idiosyncrasies.
4. Callback Hell: unpredictable and interleaved execution.



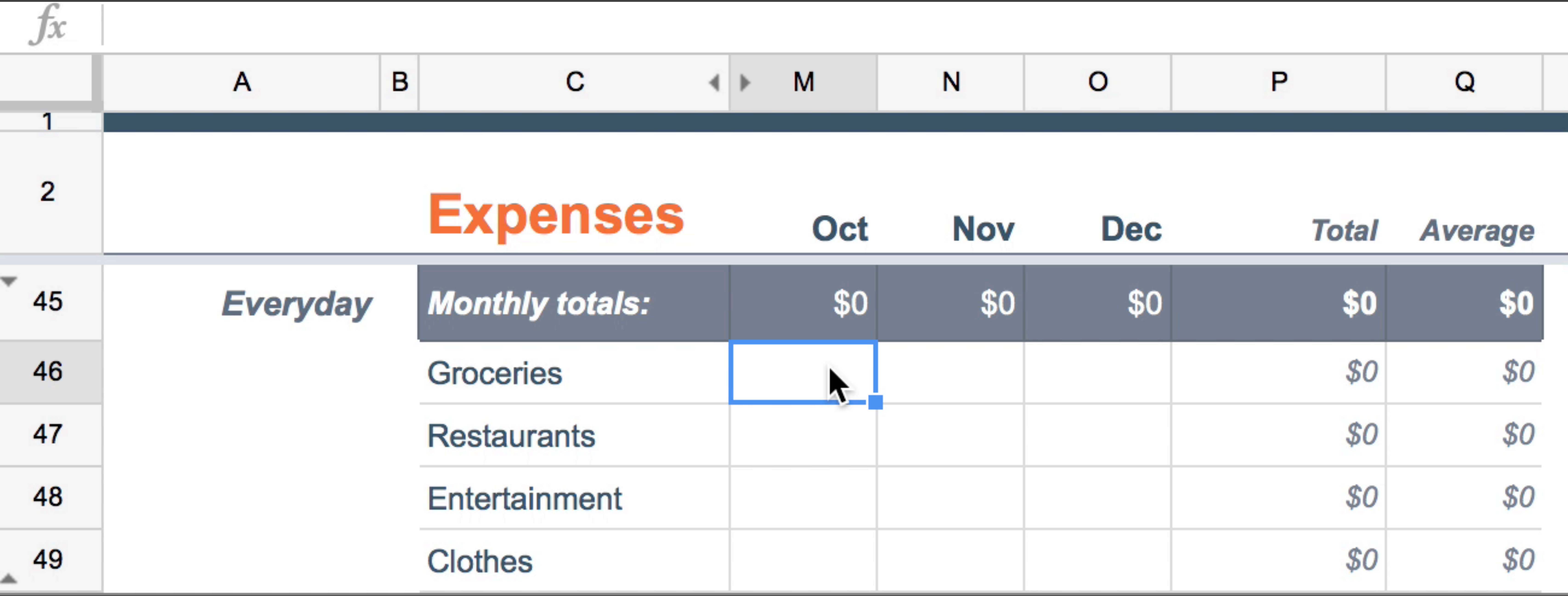
```
  function plot(p) {
    var cell = d3.select(this);

    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);

    cell.append("rect")
        .attr("class", "frame")
        .attr("x", padding / 2)
        .attr("y", padding / 2)
        .attr("width", size - padding)
        .attr("height", size - padding);

    cell.selectAll("circle")
        .data(data)
      .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 + "px");
});

function 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: b[j], j: j});
  return c;
}

</script>
```

flowers.csv

```
sepal length,sepal width,petal length,petal width,species
5.1,3.5,1.4,0.2,setosa
```

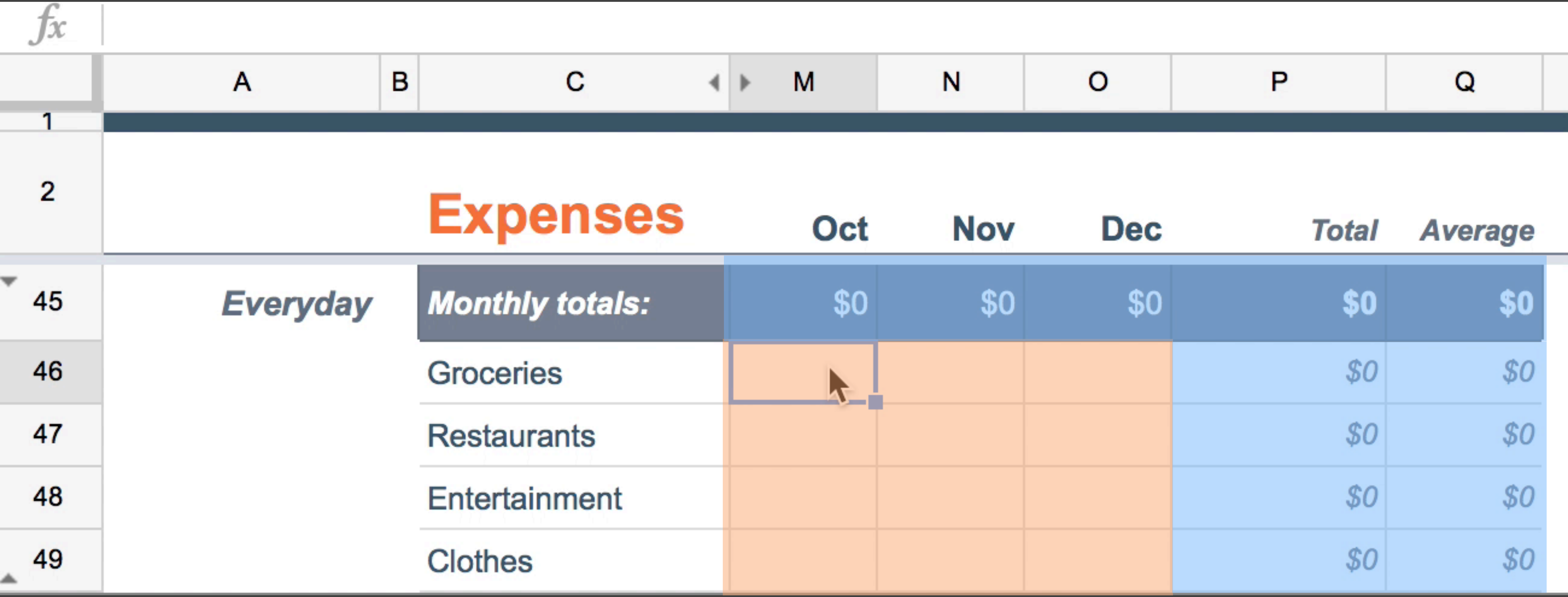
1. Manually maintain state.
2. Re-define visual appearance in multiple locations.
3. Low-level idiosyncrasies.
4. Callback Hell: unpredictable and interleaved execution.

Reactive Programming



	A	B	C	M	N	O	P	Q
1								
2			**Expenses**	**Oct**	**Nov**	**Dec**	***Total***	***Average***
45	***Everyday***		***Monthly totals:***	$0	$0	$0	**$0**	**$0**
46			Groceries				*$0*	*$0*
47			Restaurants				*$0*	*$0*
48			Entertainment				*$0*	*$0*
49			Clothes				*$0*	*$0*

Reactive Programming



	A	B	C	M	N	O	P	Q
1								
2			Expenses	Oct	Nov	Dec	*Total*	*Average*
45	*Everyday*		***Monthly totals:***	$0	$0	$0	**$0**	**$0**
46			Groceries				*$0*	*$0*
47			Restaurants				*$0*	*$0*
48			Entertainment				*$0*	*$0*
49			Clothes				*$0*	*$0*

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

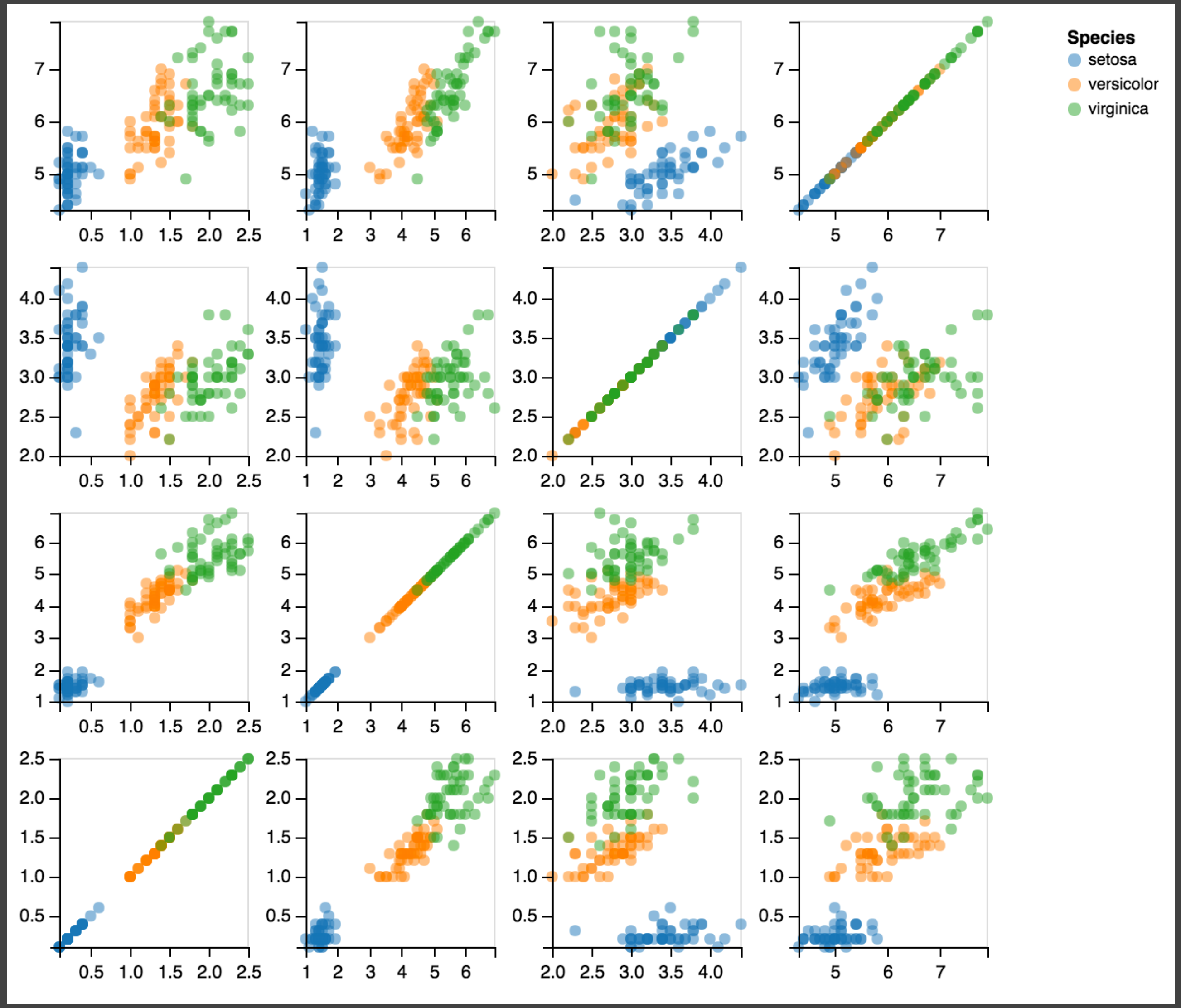
Data + Transforms

Scales

Guides

Marks

```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "iris", "url": "data/iris.json"},
    {"name": "fields", "values": ["sepalWidth", ...]
  ],
  "scales": [
    {
      "name": "color", "type": "ordinal",
      "domain": {"data": "iris", "field": "species"}
      "range": "category20"
    }, ...
  ],
  "legends": [
    {"fill": "x", "scale": "color"}
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "fields",
      "transform": [{"type": "cross"}]
    },
    "marks": [{
      "type": "symbol",
      "from": {"data": "iris"},
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }],
    "scales": [{
      "name": "sx", "type": "linear",
      "domain": {"data": "iris", "field": {"parent": "a.data"},
```



Data

Transforms

Scales

Guides

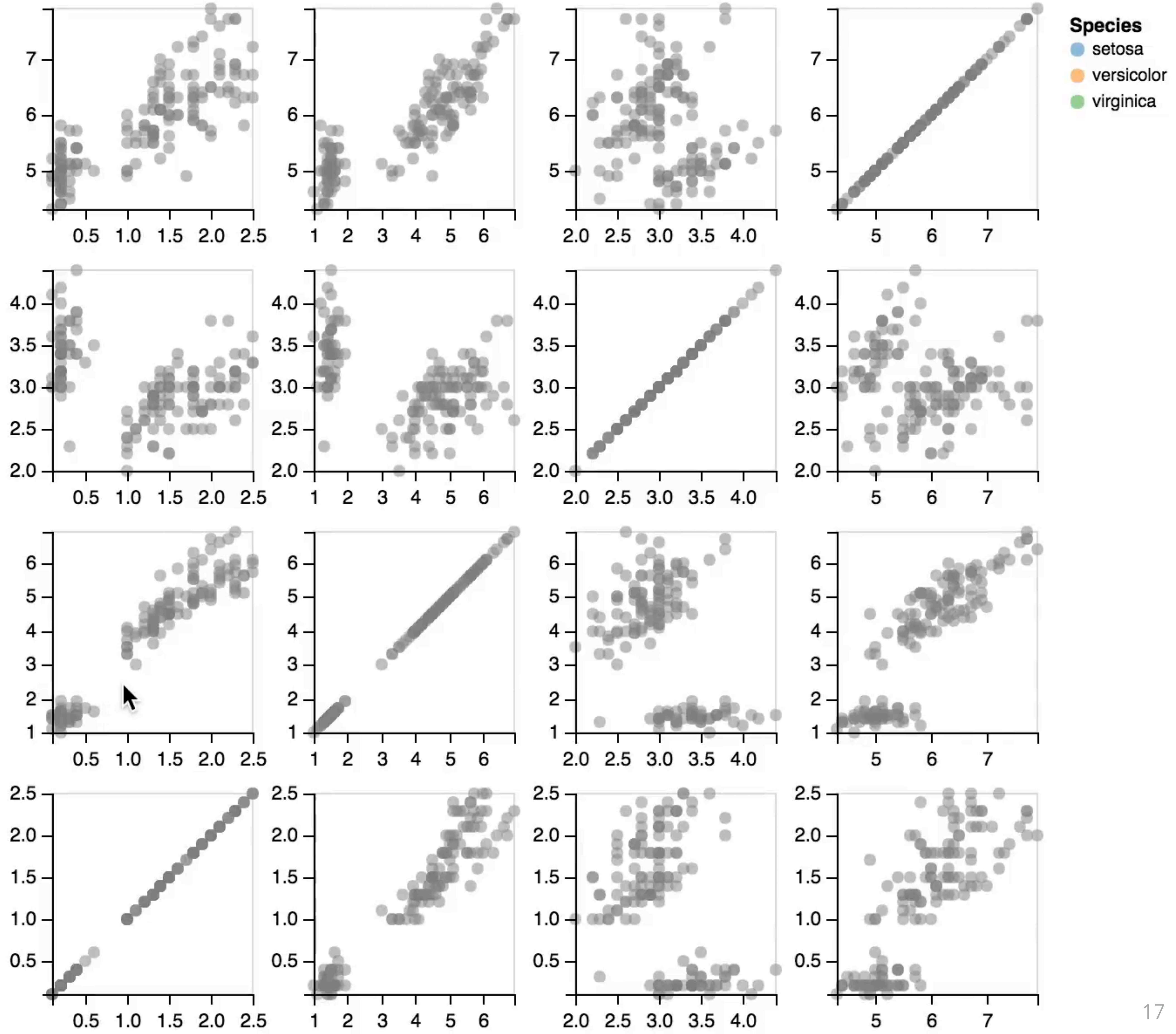
Marks

Event Streams	`[mousedown, mouseup] > mousemove`
Signals	`minX = min(width, event.x)`
Scale Inversions	`minVal = xScale.invert(minX)`
Predicates	`p(t) = minVal ≤ t.value ≤ maxVal`

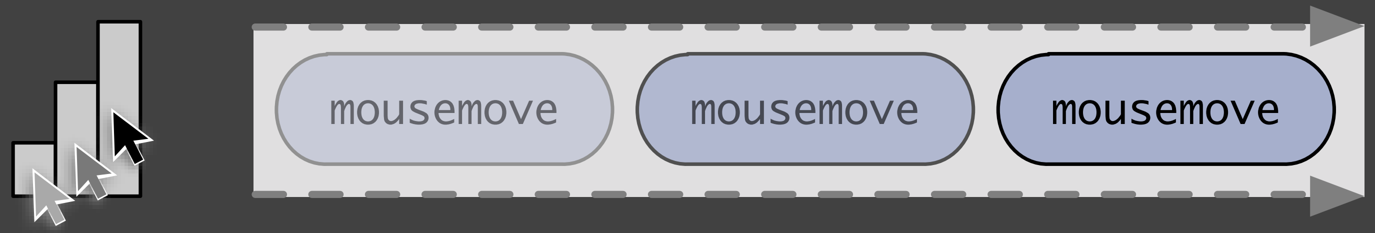
Production Rules	`fill = p(t) → colorScale(t.category)`
	`         ∅   → gray`

Example

Brushing & Linking



Events are a form of **streaming data**.

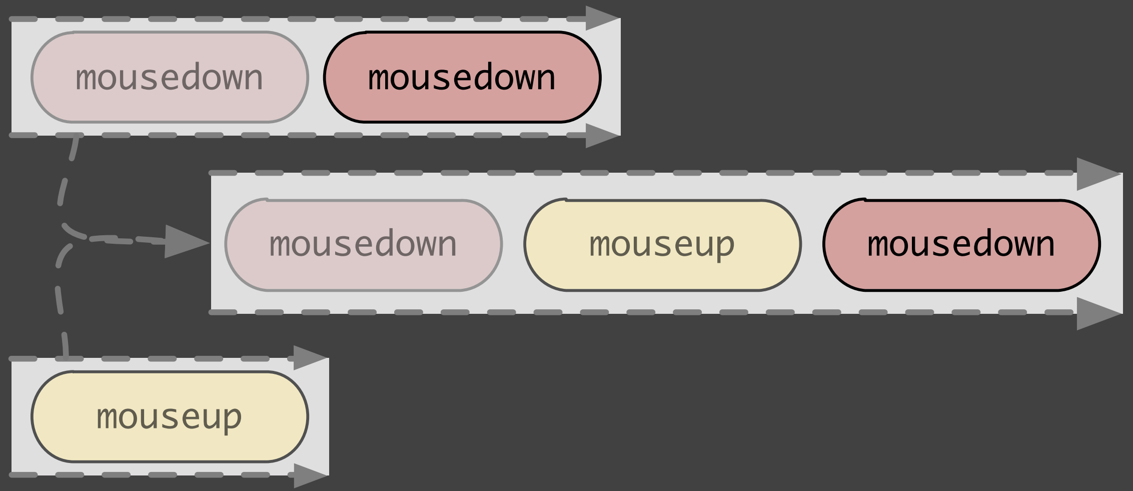


A stream of `mousemove` events that occur on `rect` marks .

```
rect:mousemove
```

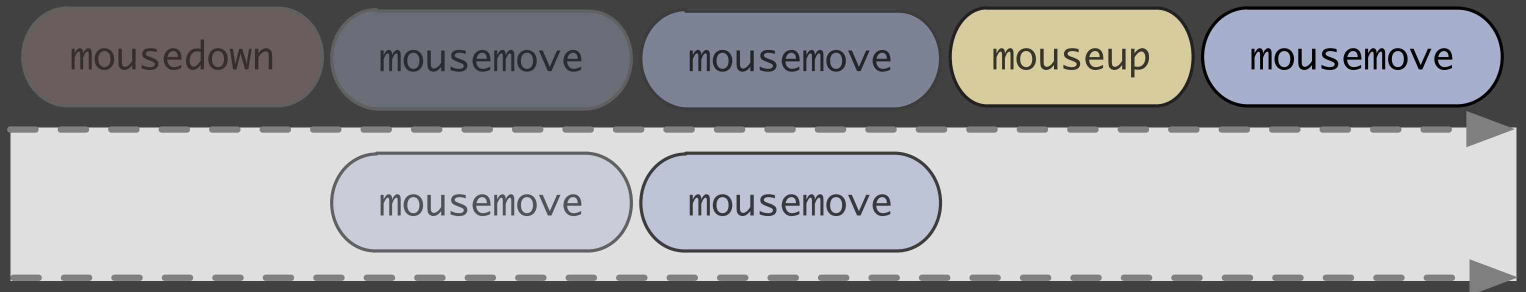
`*:mousedown, *:mouseup`

a single stream merges `mousedown` and `mouseup` streams



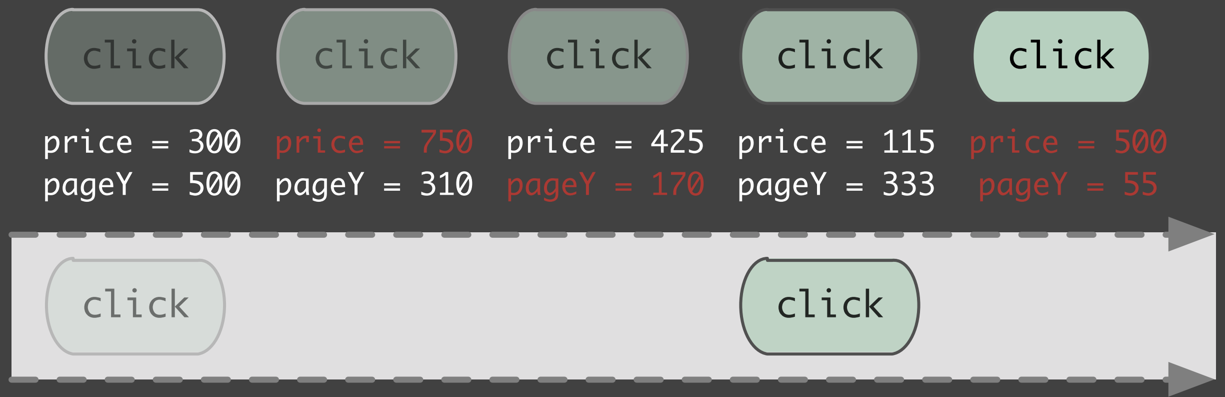
`[*:mousedown, *:mouseup] > *:mousemove`

A stream of `mousemove` events that occur between a `mousedown` and a `mouseup` (aka drag)



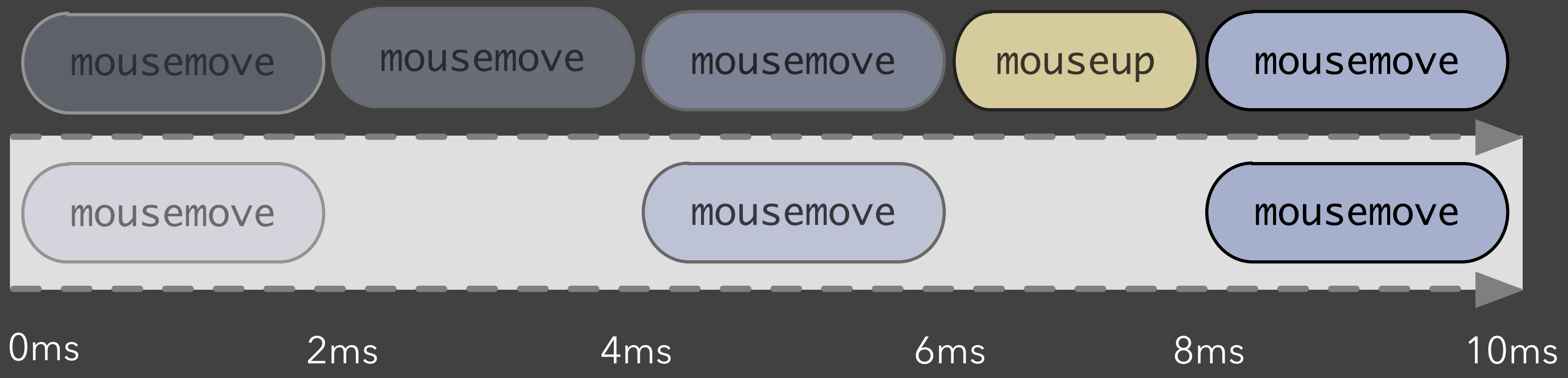
`*:click[event.pageY >= 300]`
`[data.price < 500]`

filtered stream of `click` events

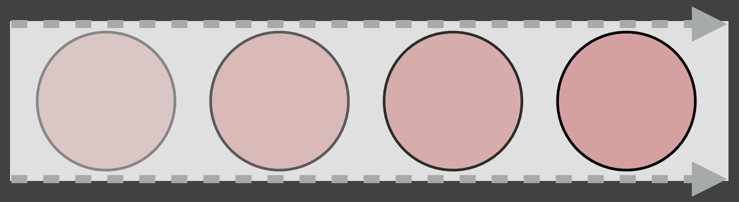


`*:mousemove{3ms,5ms}`

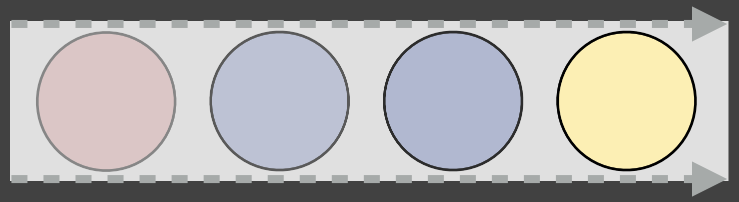
stream of `mousemove` events that occur at least 3ms, and at most 5ms, apart (debouncing/throttling)



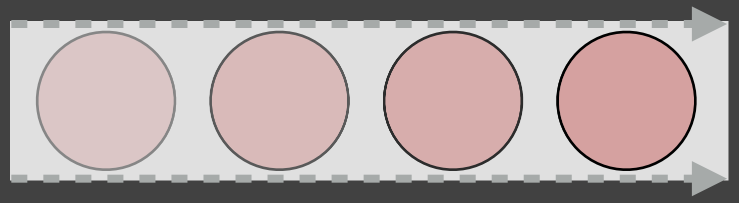
```
mousedown
```



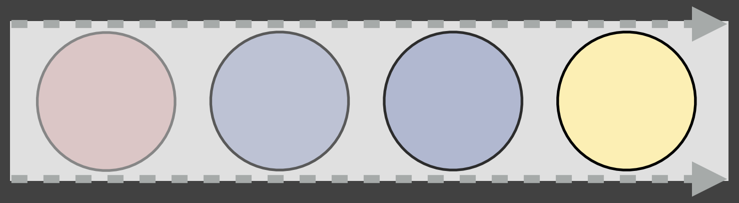
```
[mousedown, mouseup] >
    mousemove
```

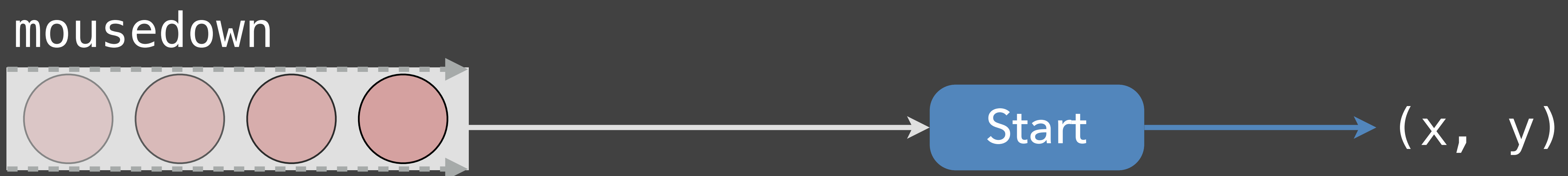


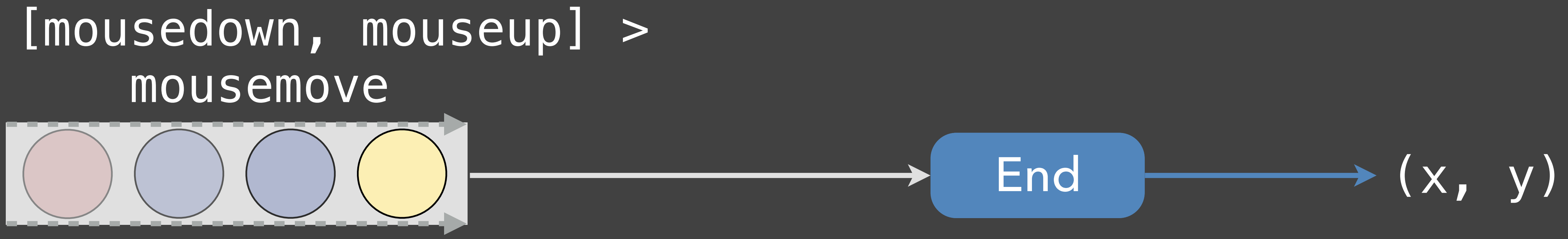
```
mousedown
```

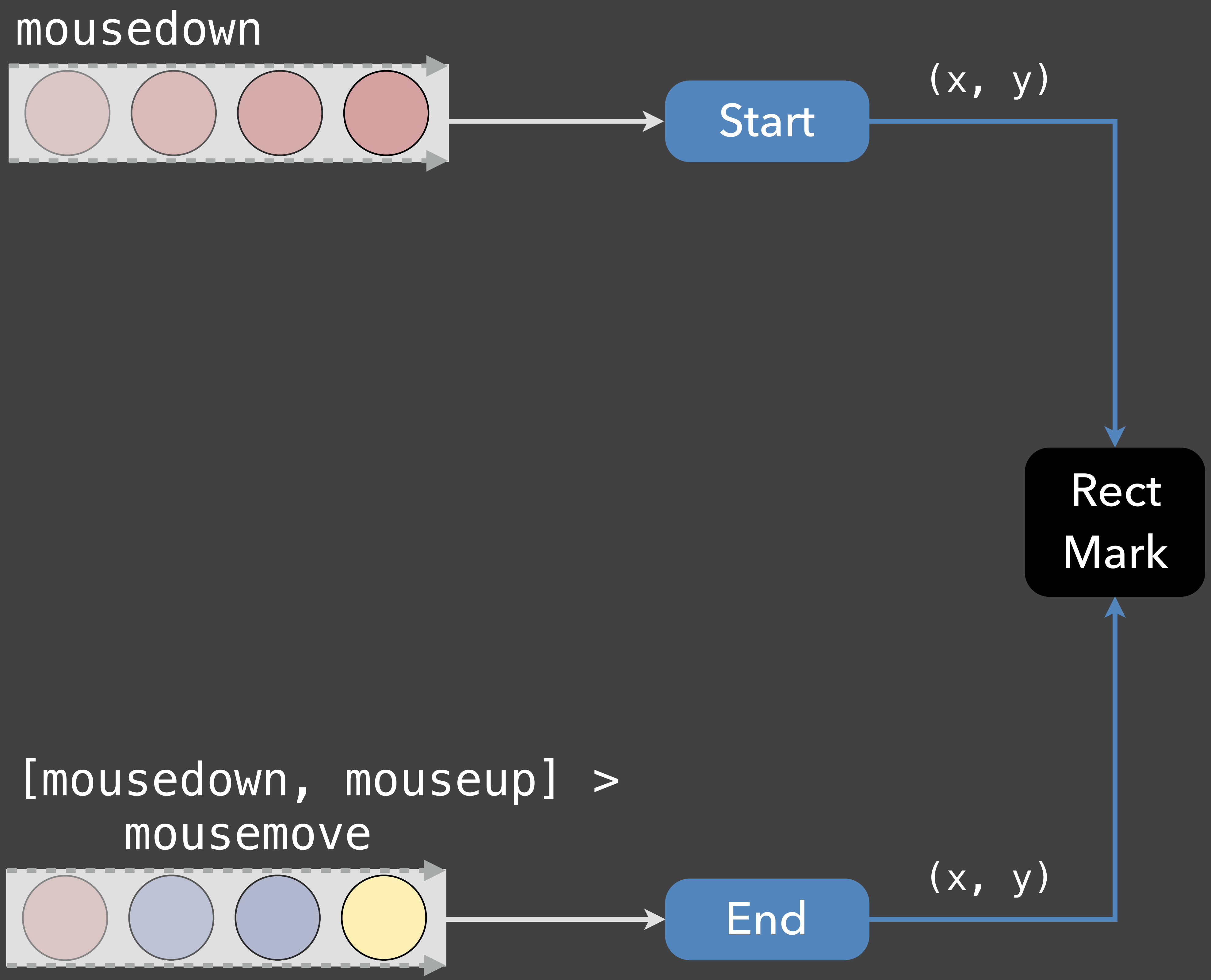


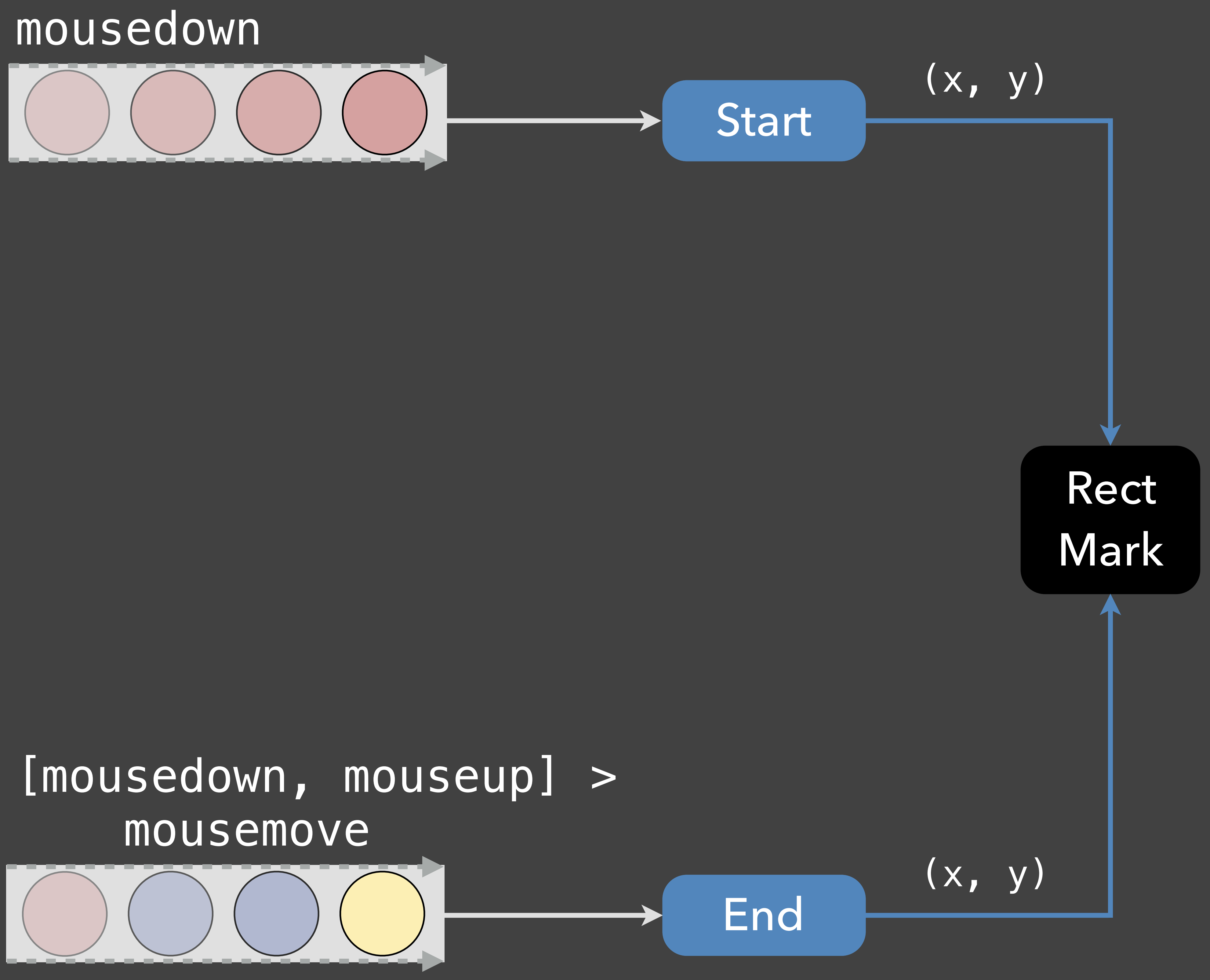
```
[mousedown, mouseup] >
    mousemove
```

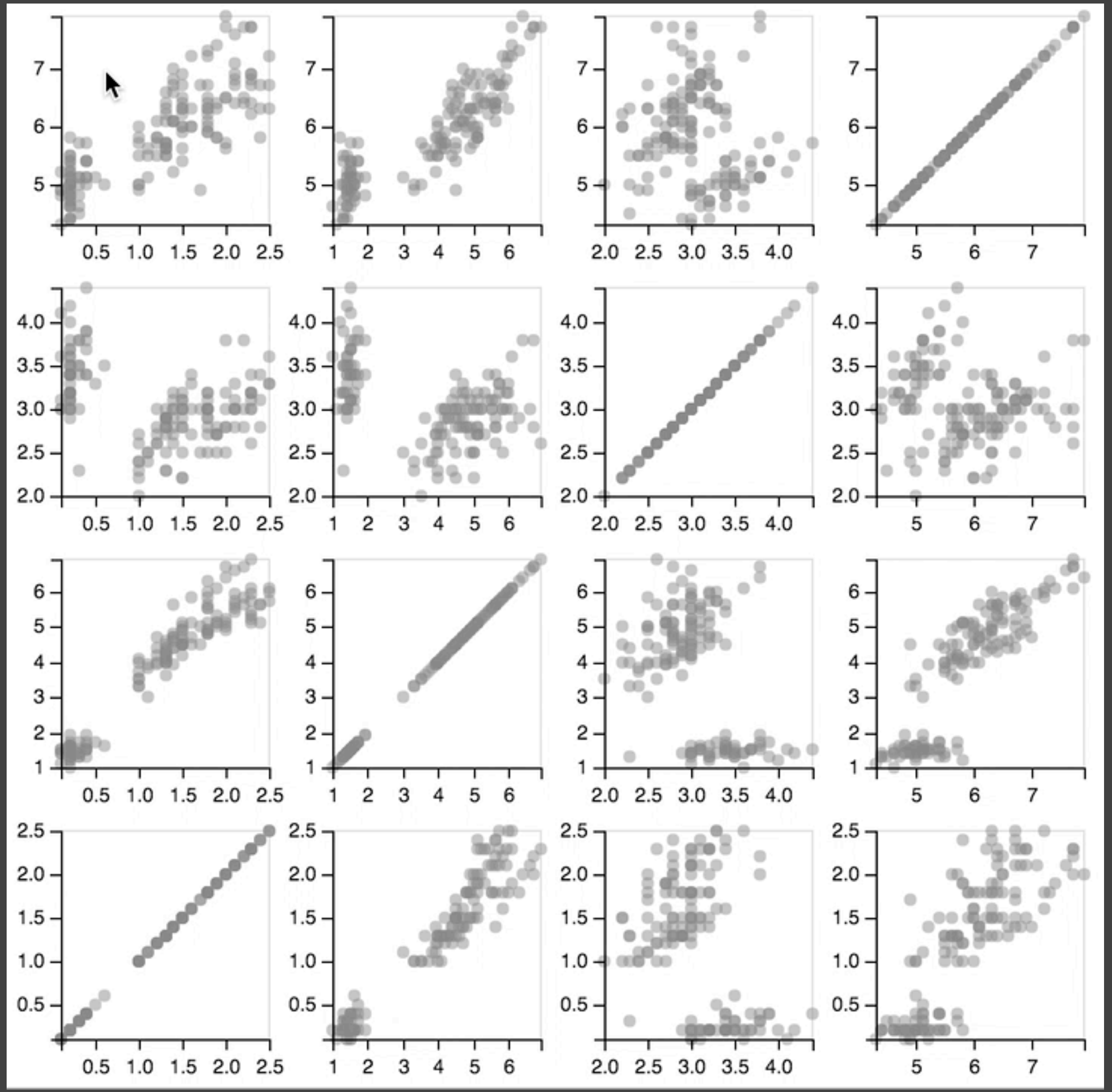


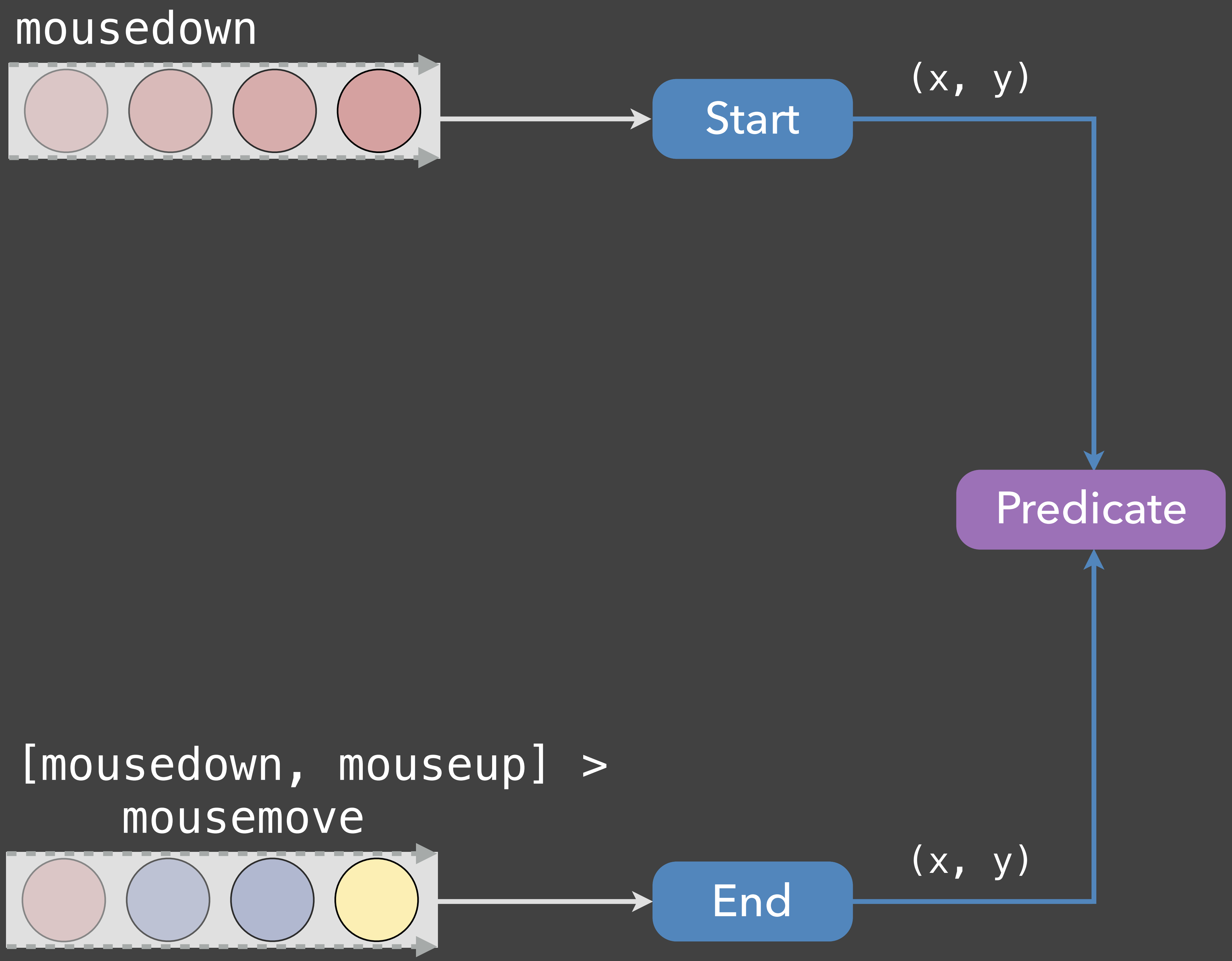

mousedown
Start
(x, y)

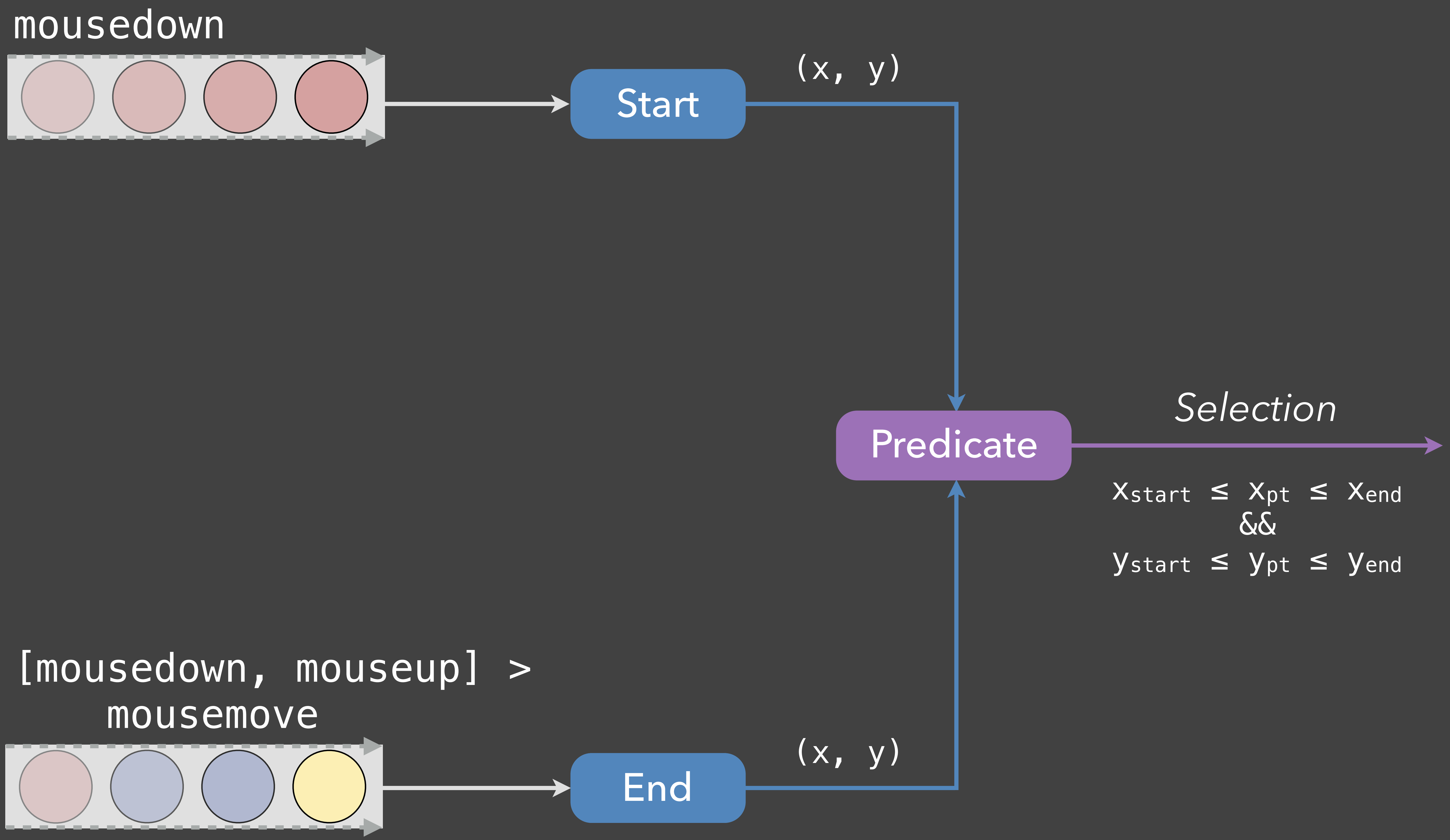

[mousedown, mouseup] >
mousemove
End
(x, y)


mousedown
Start
(x, y)
Rect Mark
[mousedown, mouseup] > mousemove
End
(x, y)

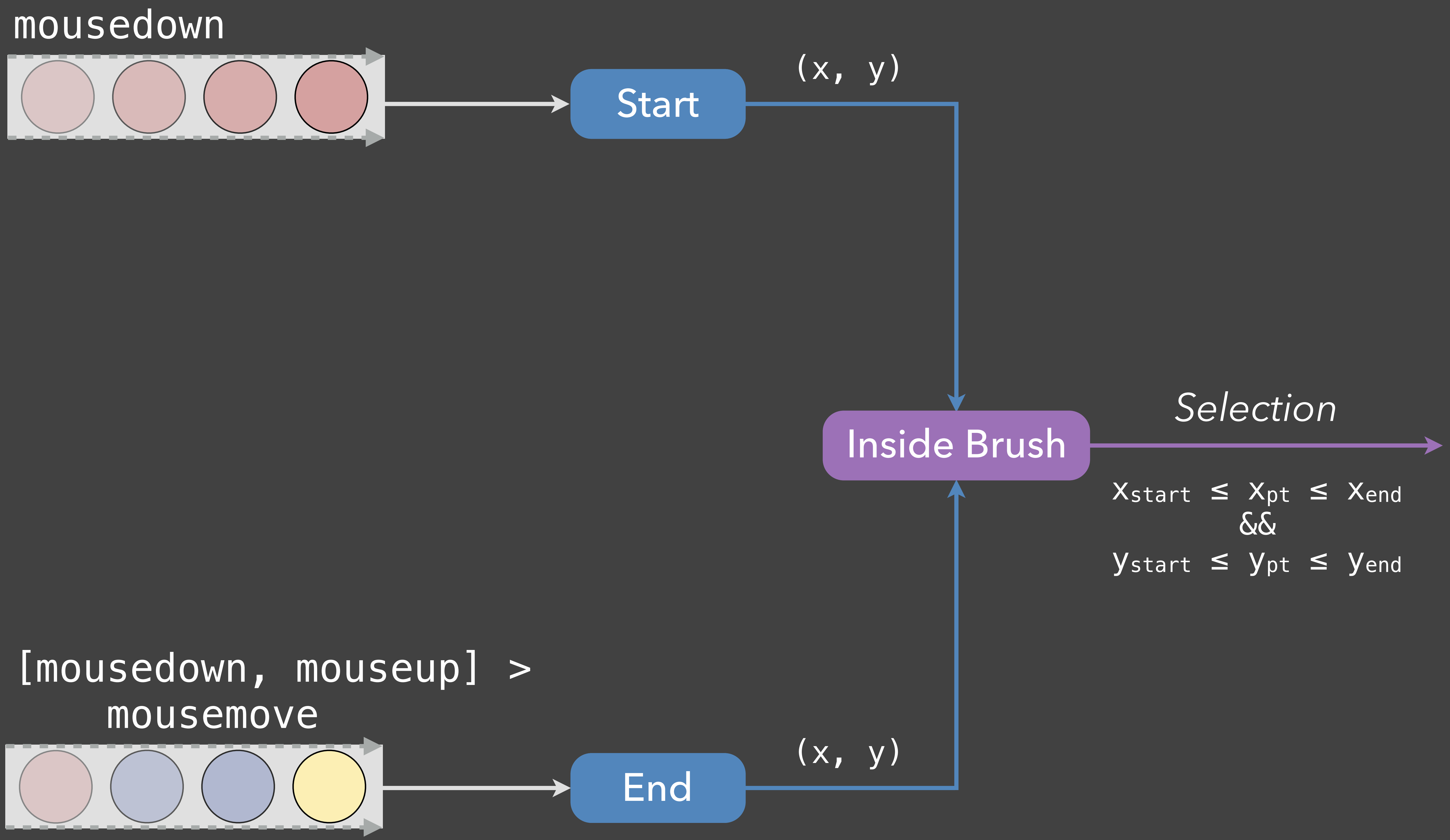

mousedown
Start
(x, y)
Rect Mark
[mousedown, mouseup] > mousemove
End
(x, y)

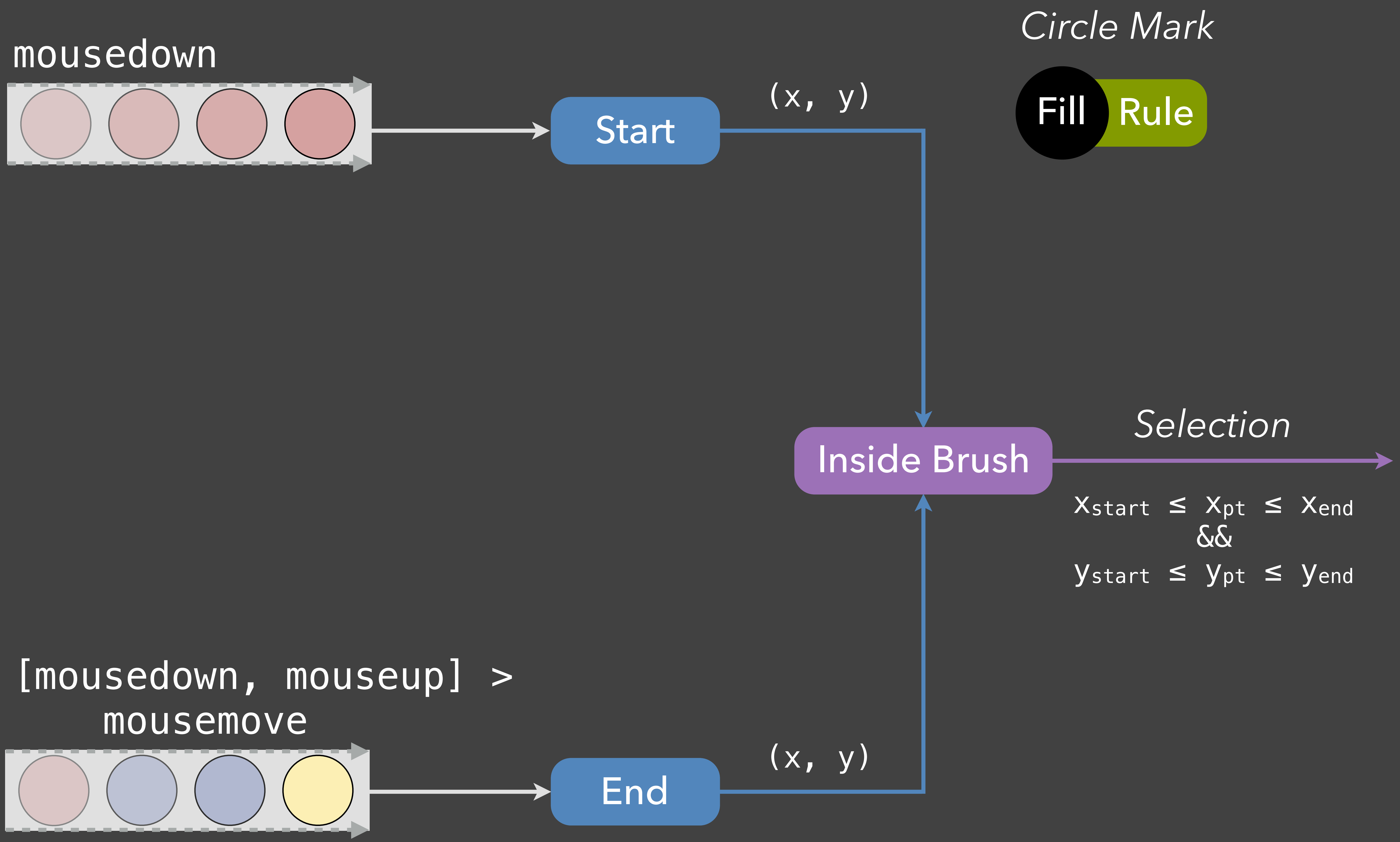



mousedown
Start
(x, y)
Predicate
[mousedown, mouseup] >
mousemove
End
(x, y)

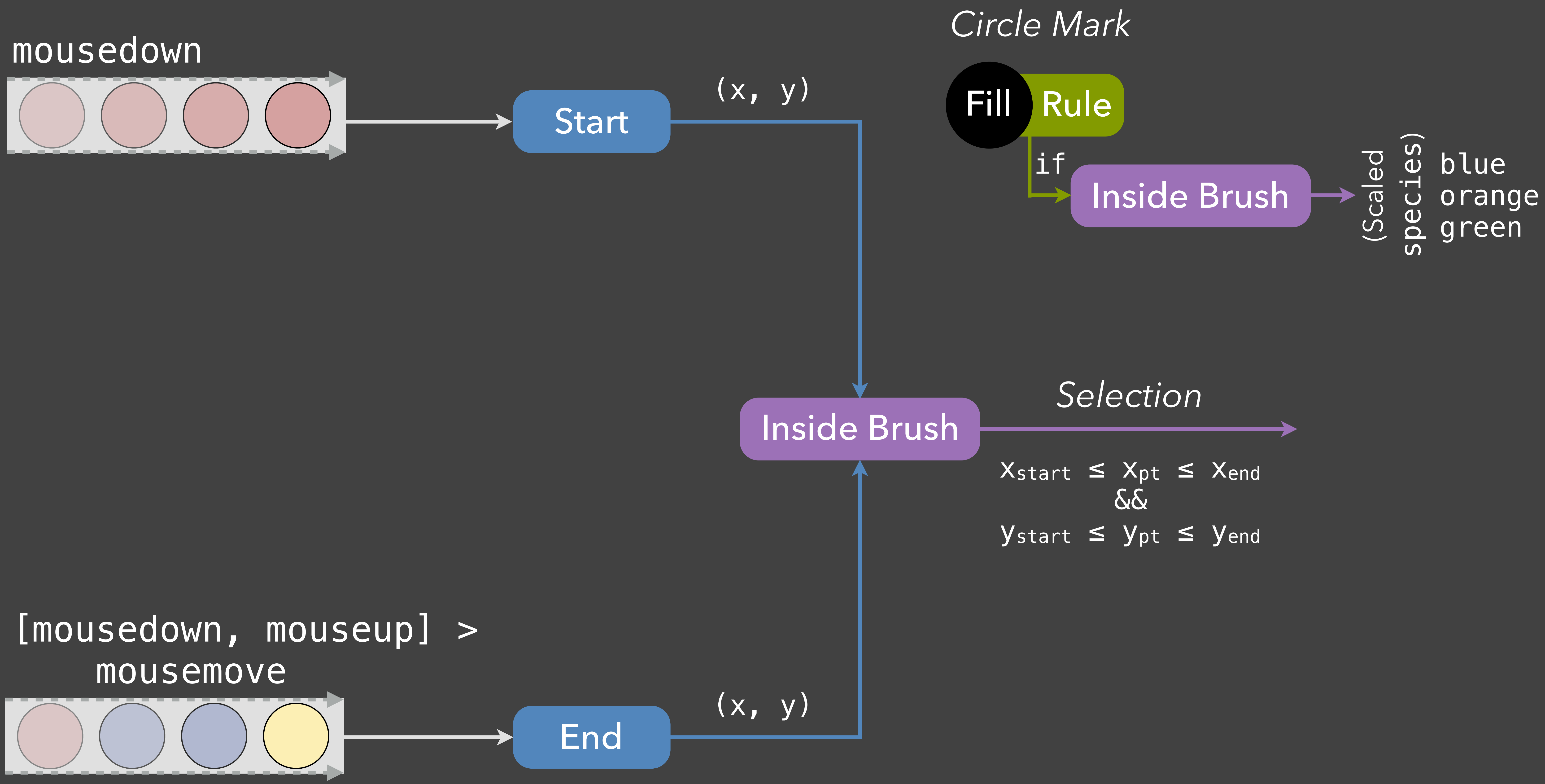


$x_{start} \le x_{pt} \le x_{end}$ && $y_{start} \le y_{pt} \le y_{end}$

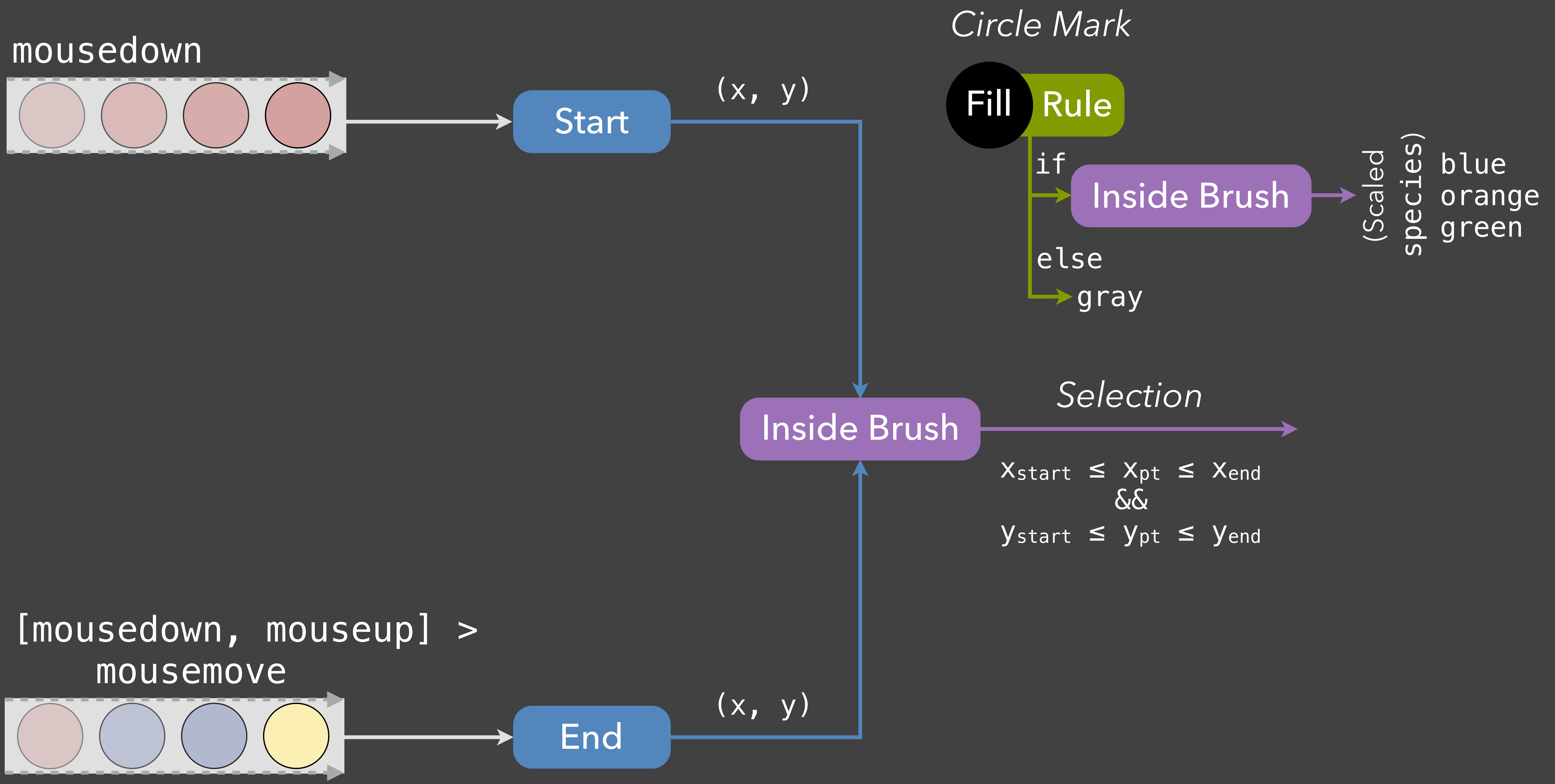

mousedown
Start
(x, y)
Inside Brush
Selection
$x_{start} \le x_{pt} \le x_{end}$
&&
$y_{start} \le y_{pt} \le y_{end}$
[mousedown, mouseup] > mousemove
End
(x, y)



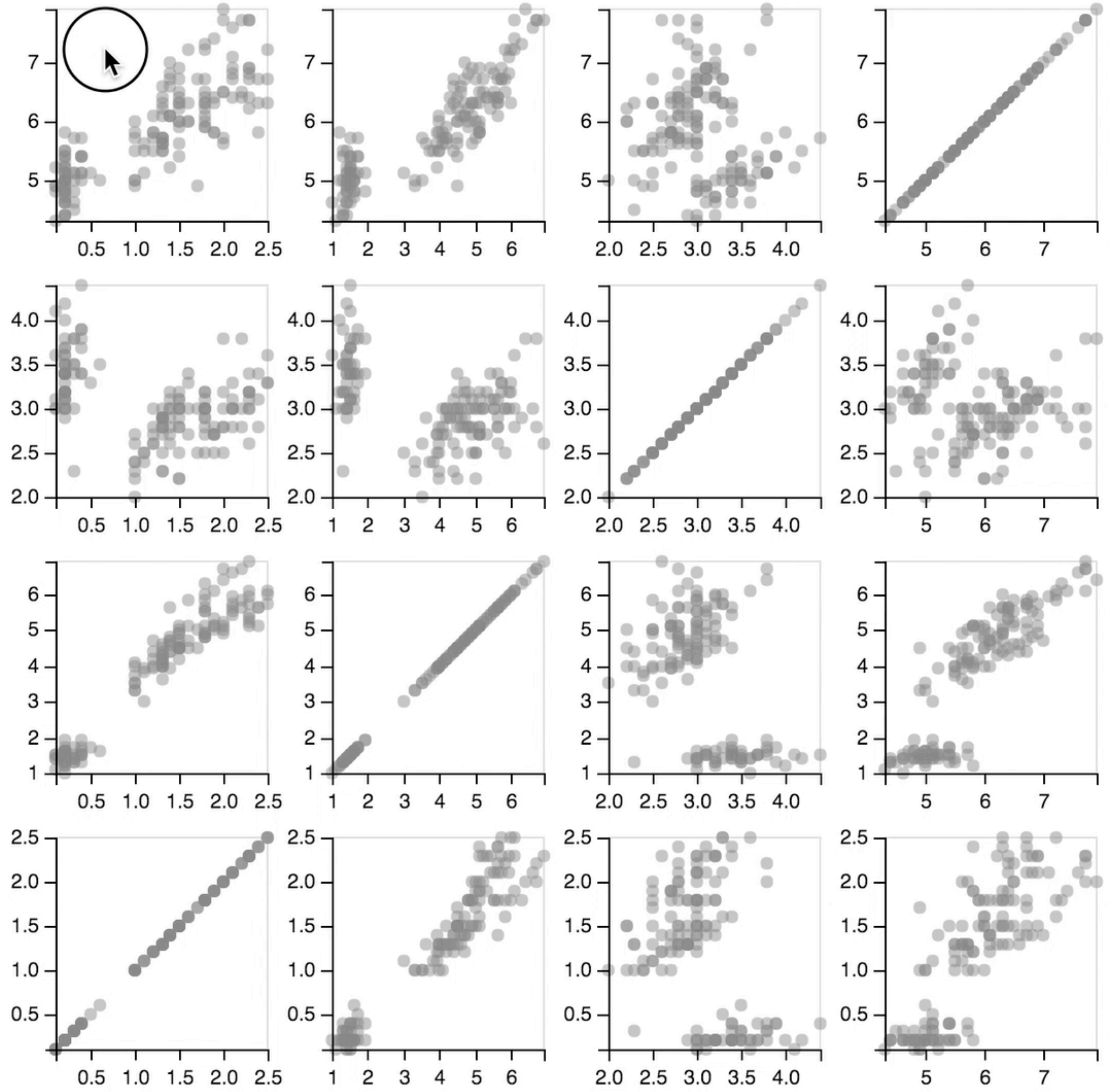
$$x_{start} \le x_{pt} \le x_{end}$$
$$\&\&$$
$$y_{start} \le y_{pt} \le y_{end}$$



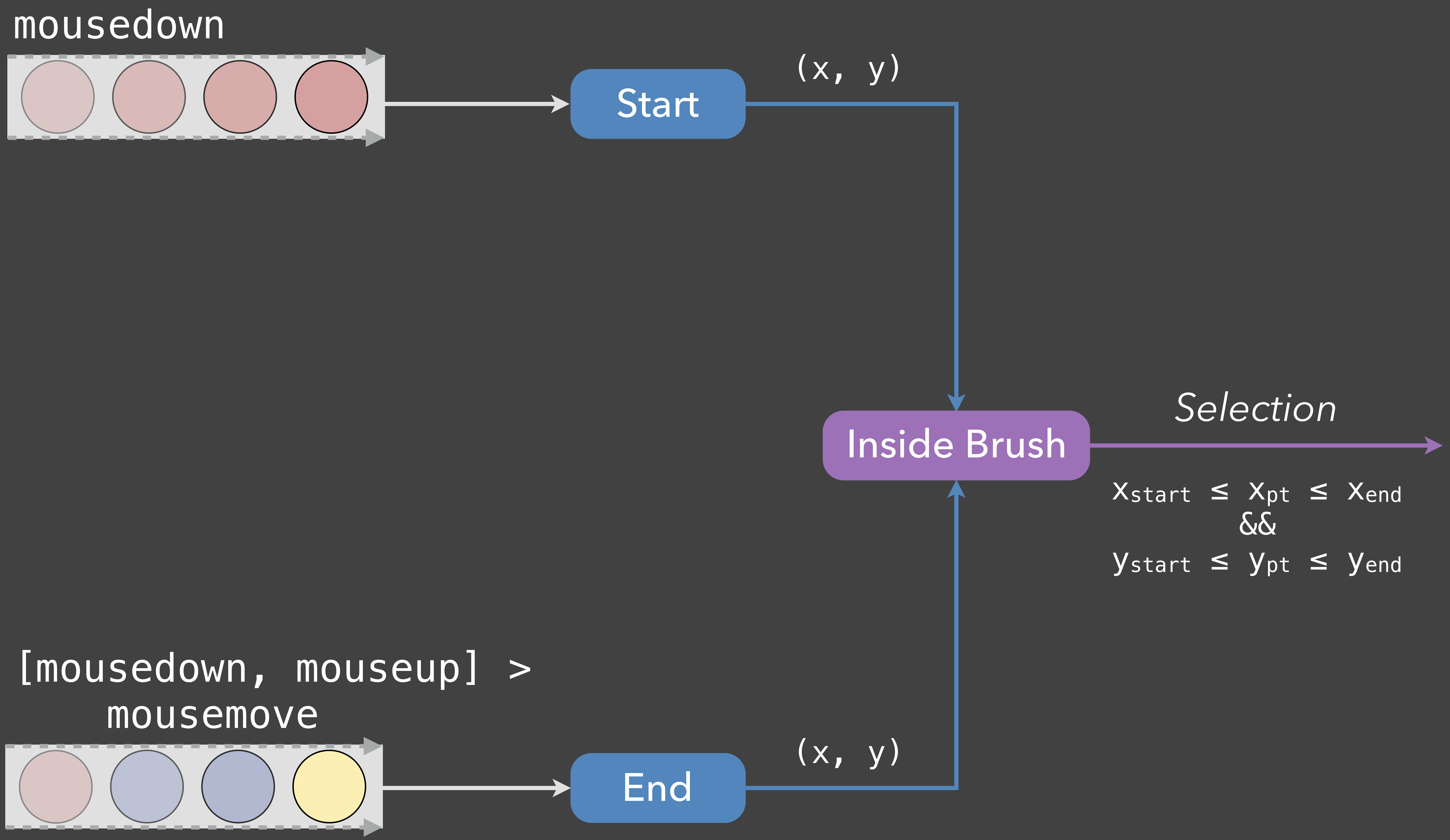
$$x_{start} \le x_{pt} \le x_{end}$$
$$\&\&$$
$$y_{start} \le y_{pt} \le y_{end}$$



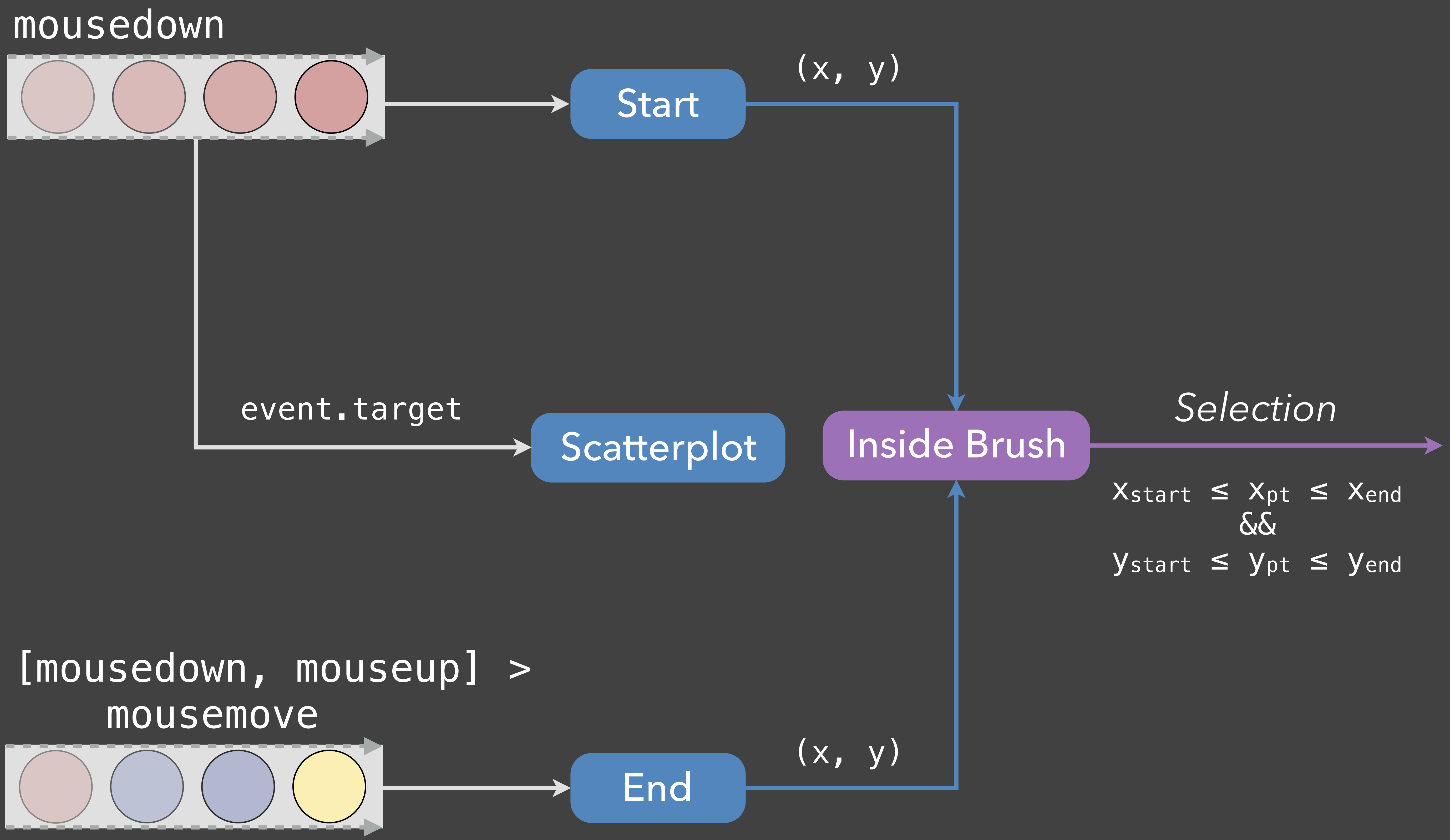
$x_{start} \leq x_{pt} \leq x_{end}$ && $y_{start} \leq y_{pt} \leq y_{end}$



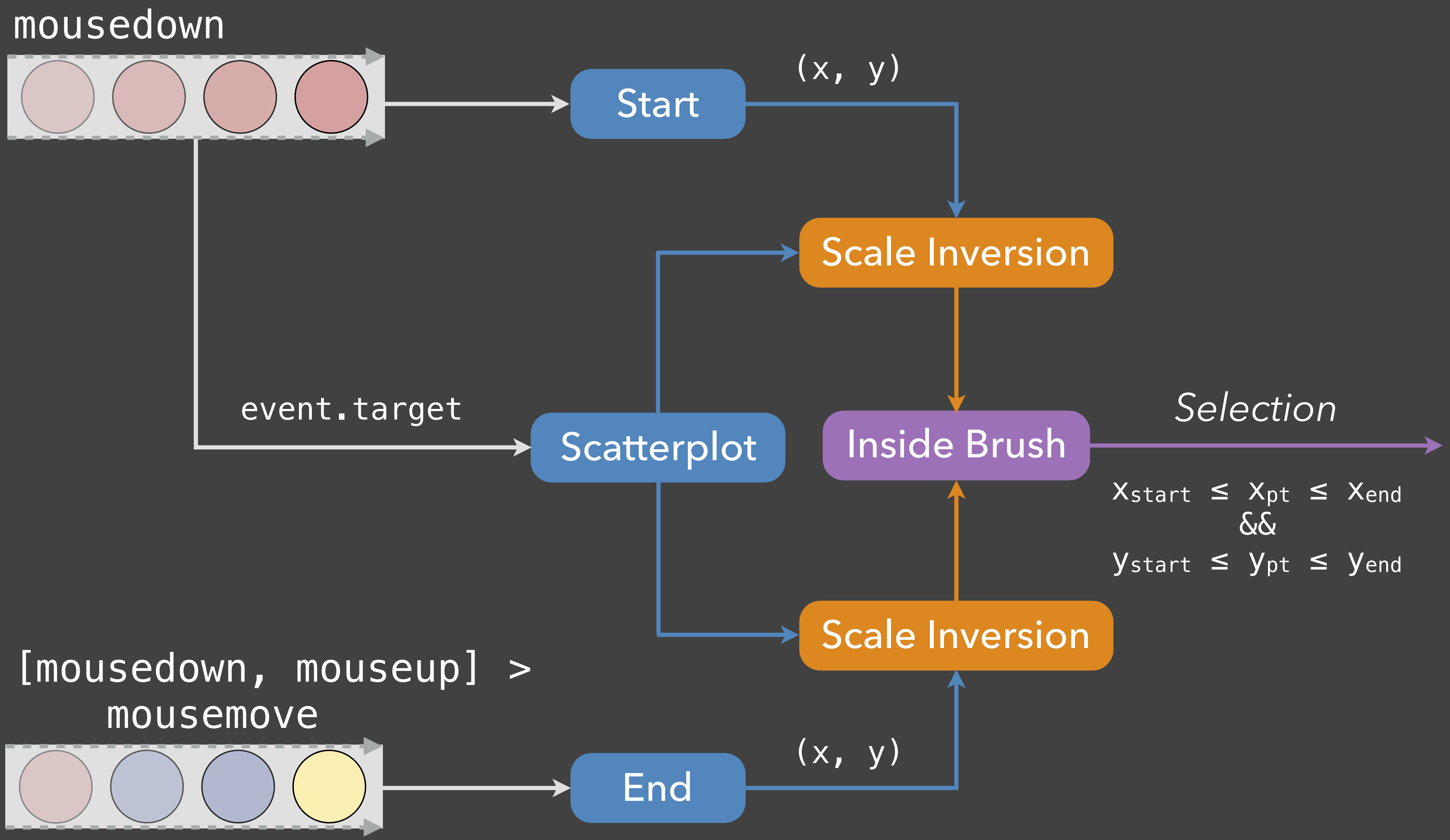
Species
● setosa
● versicolor
● virginica

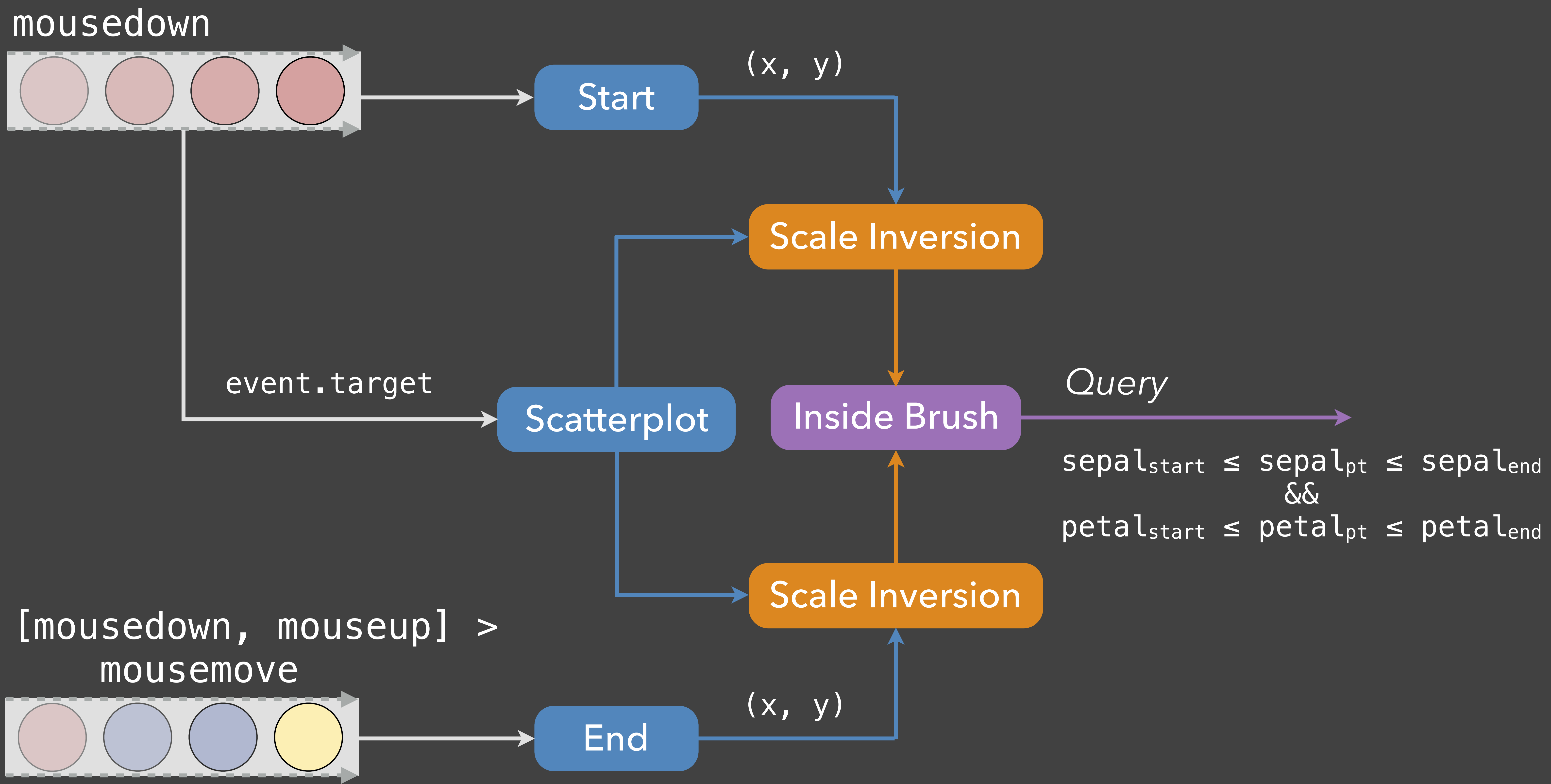


$$x_{start} \le x_{pt} \le x_{end} \;\&\&\; y_{start} \le y_{pt} \le y_{end}$$

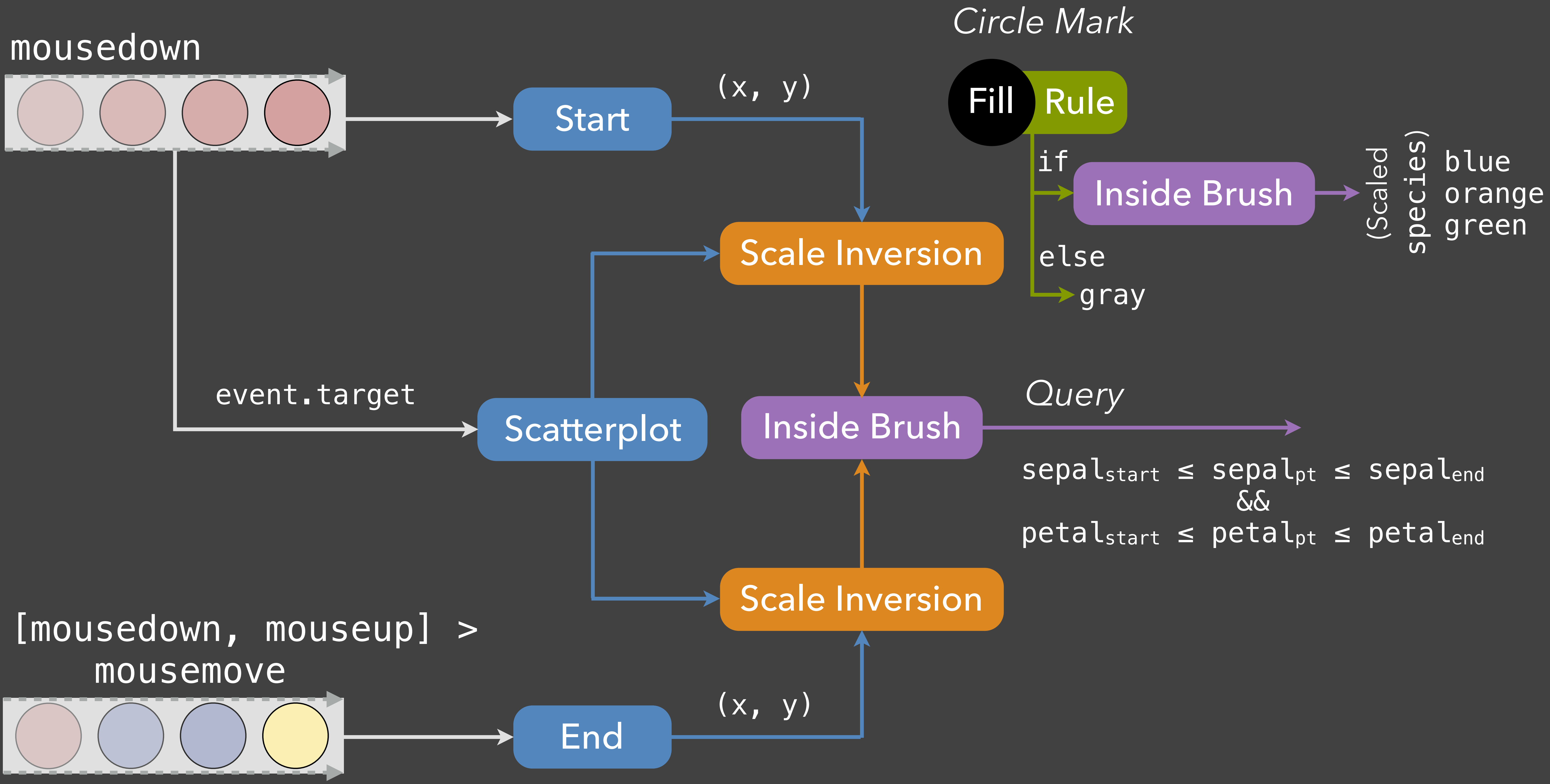


$x_{start} \leq x_{pt} \leq x_{end}$
&&
$y_{start} \leq y_{pt} \leq y_{end}$

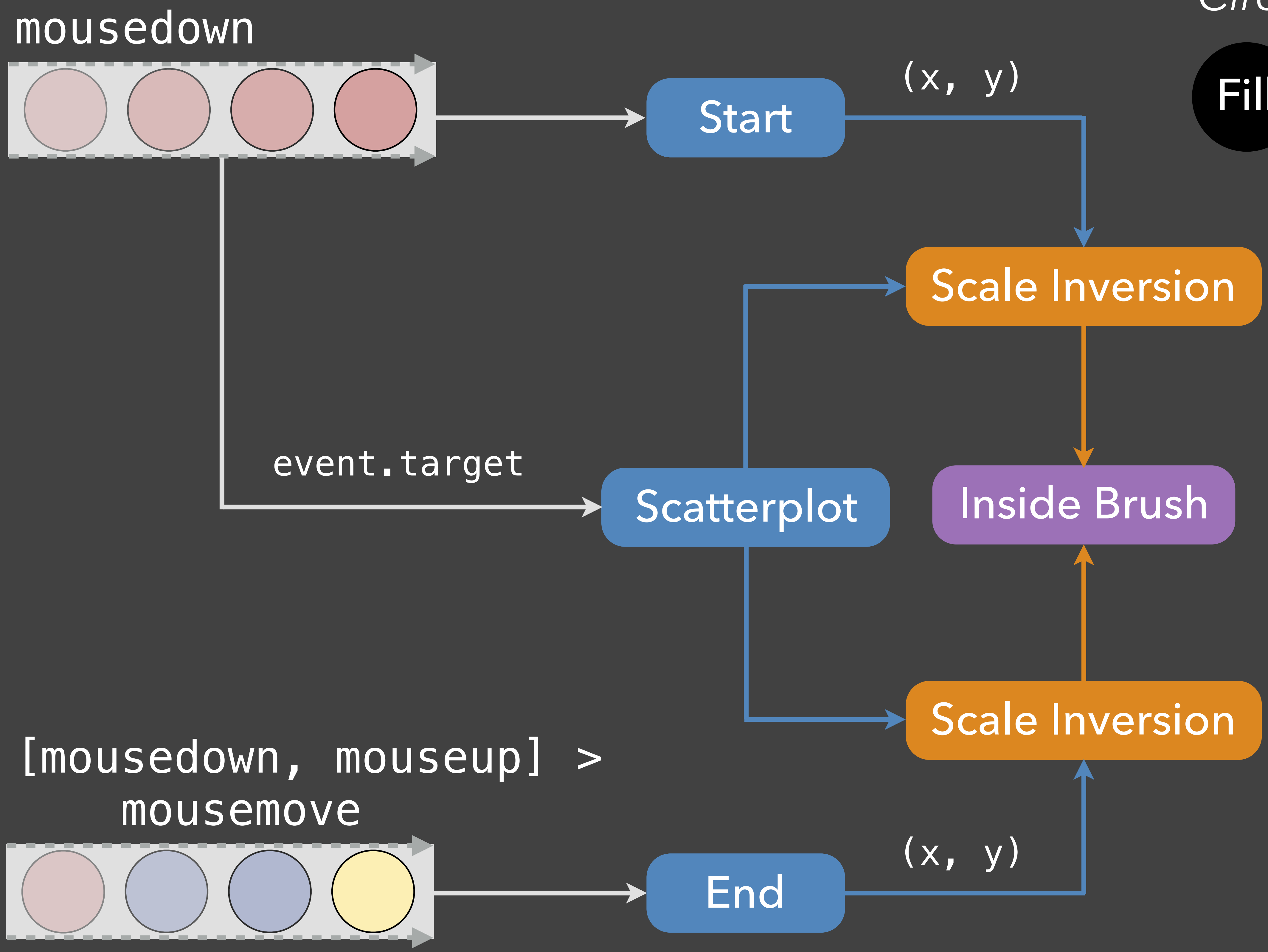

mousedown
Start
(x, y)
Scale Inversion
event.target
Scatterplot
Inside Brush
Selection
$x_{start} \le x_{pt} \le x_{end}$
&&
$y_{start} \le y_{pt} \le y_{end}$
Scale Inversion
[mousedown, mouseup] > mousemove
End
(x, y)

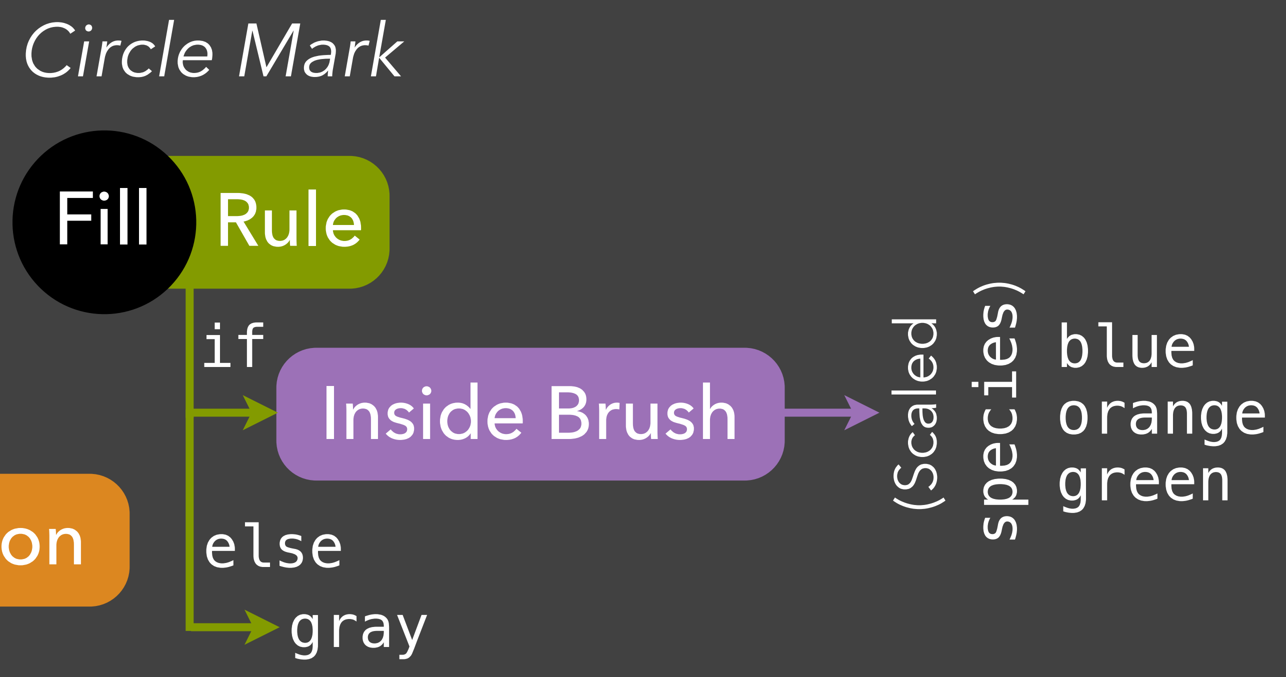


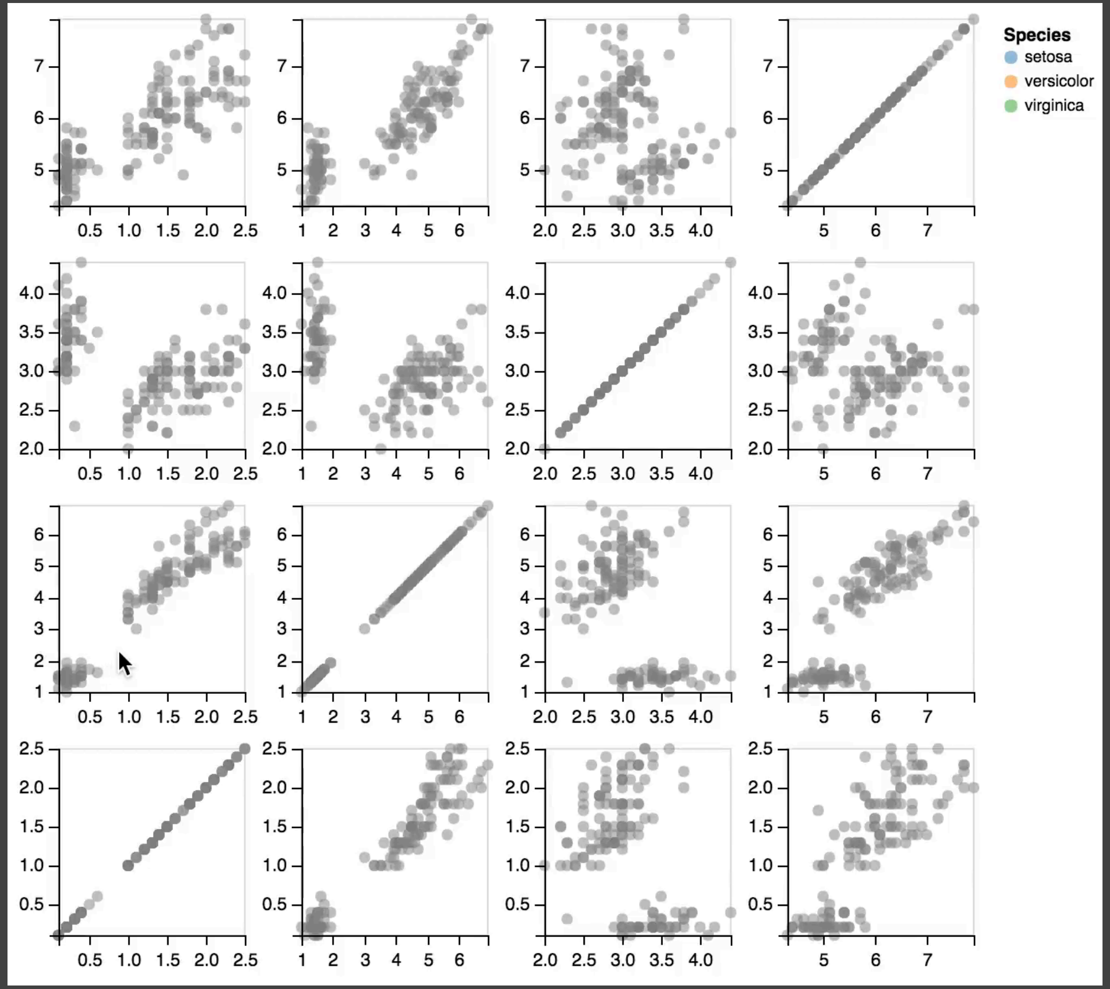
$$\text{sepal}_{start} \le \text{sepal}_{pt} \le \text{sepal}_{end}$$
&&
$$\text{petal}_{start} \le \text{petal}_{pt} \le \text{petal}_{end}$$

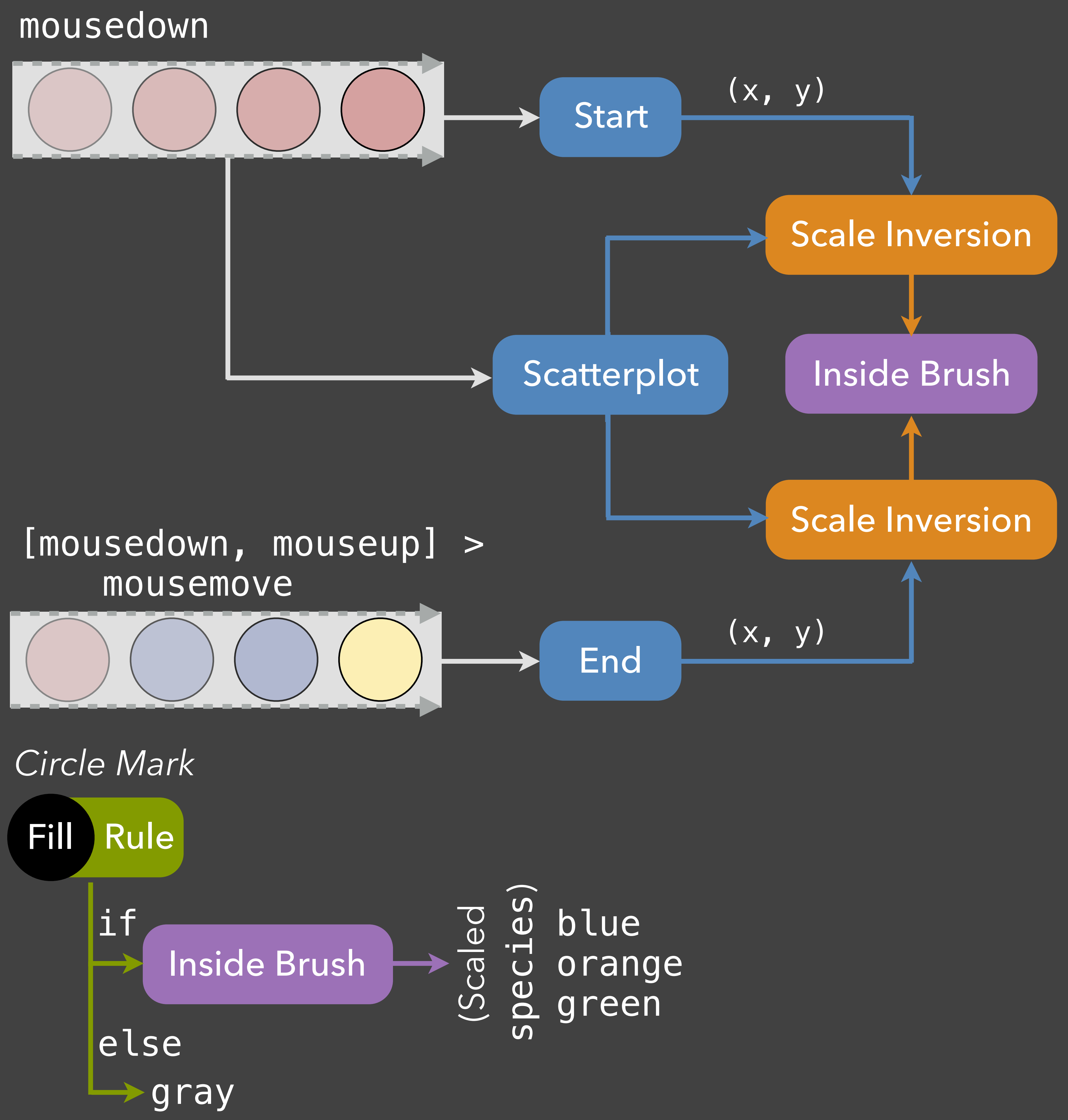


mousedown
Start
(x, y)
Circle Mark
Fill
Rule
if
Inside Brush
(Scaled species)
blue
orange
green
else
gray
Scale Inversion
event.target
Scatterplot
Inside Brush
Query
sepal_start ≤ sepal_pt ≤ sepal_end
&&
petal_start ≤ petal_pt ≤ petal_end
Scale Inversion
[mousedown, mouseup] > mousemove
End
(x, y)

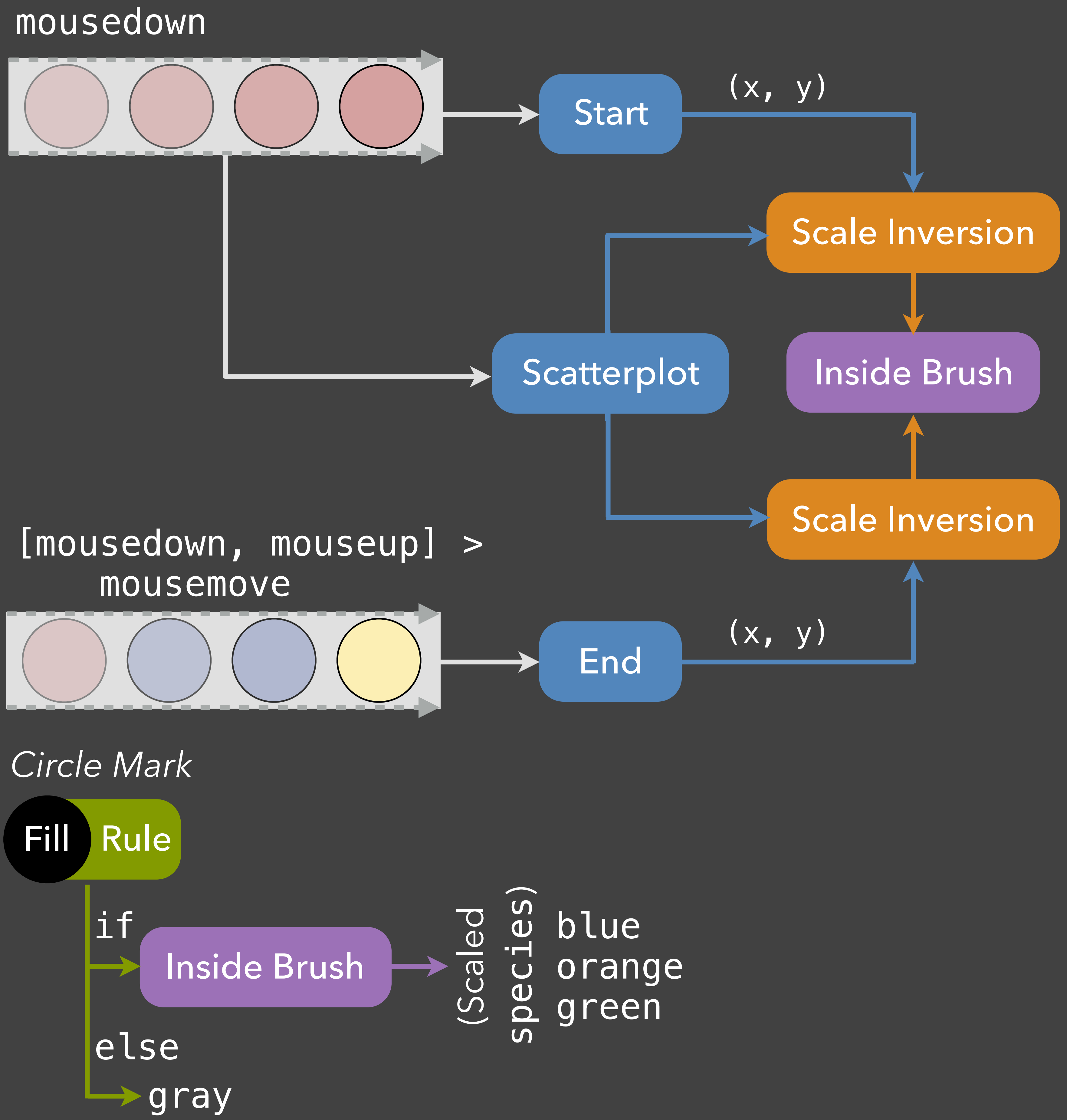

mousedown
Start
(x, y)
Scale Inversion
event.target
Scatterplot
Inside Brush
Scale Inversion
[mousedown, mouseup] > mousemove
End
(x, y)


Circle Mark
Fill
Rule
if
Inside Brush
(Scaled species)
blue
orange
green
else
gray


Species
setosa
versicolor
virginica

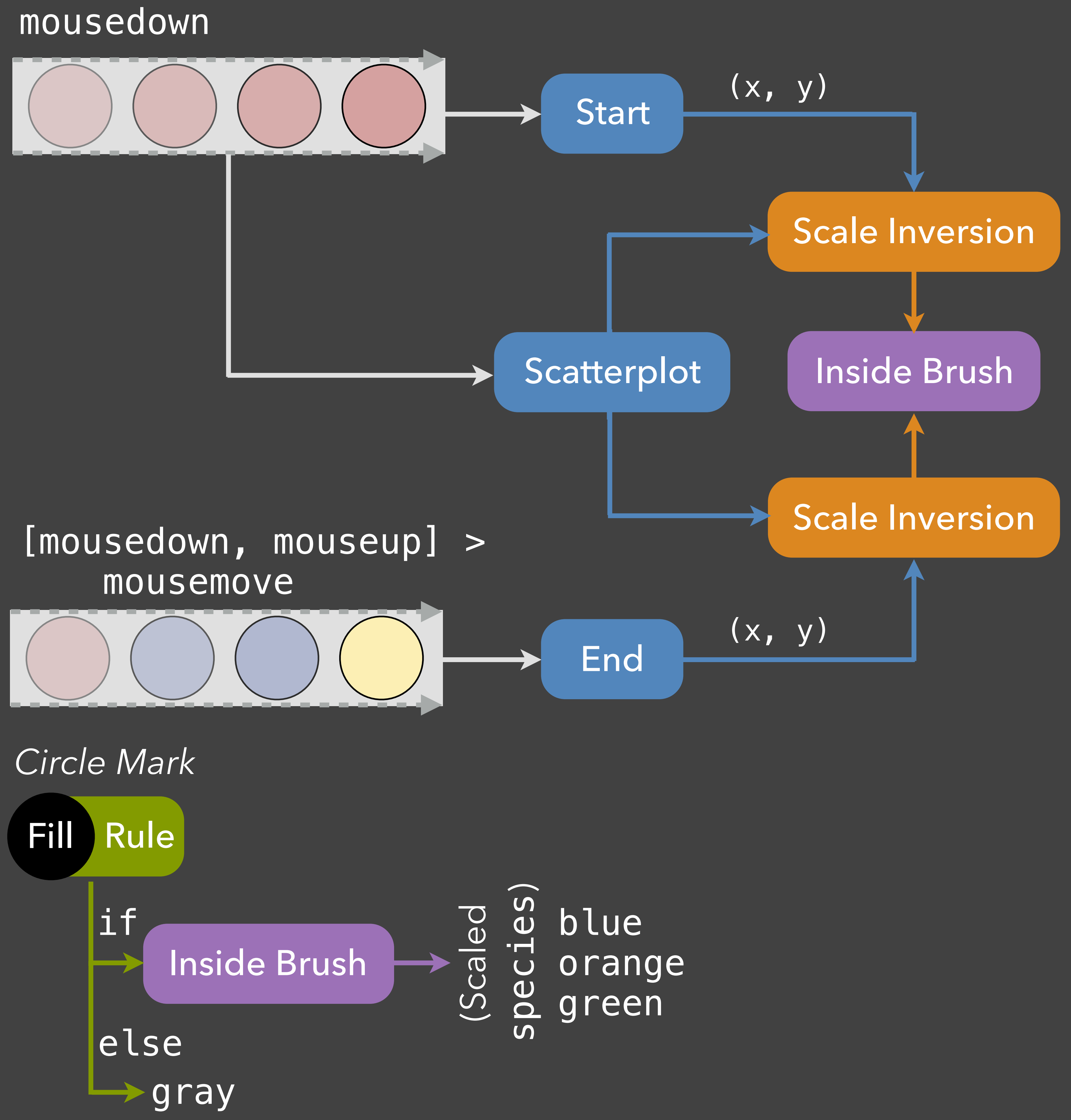

mousedown
Start
(x, y)
Scale Inversion
Scatterplot
Inside Brush
Scale Inversion
[mousedown, mouseup] > mousemove
End
(x, y)
Circle Mark
Fill
Rule
if
Inside Brush
(Scaled species)
blue
orange
green
else
gray

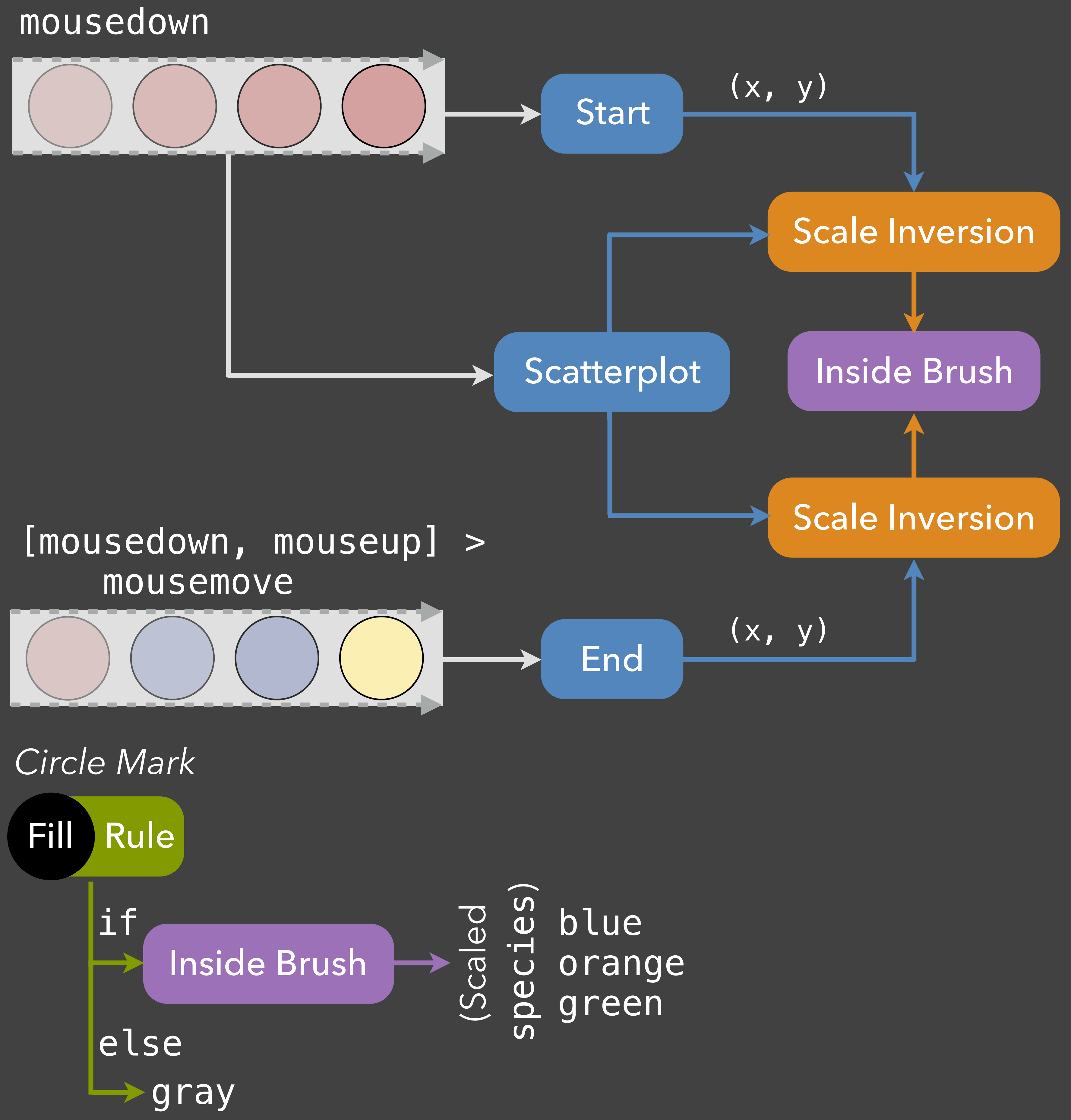
Declarative Interaction Design



Declarative Interaction Design

✓ Faster iteration + accessible to a larger audience.





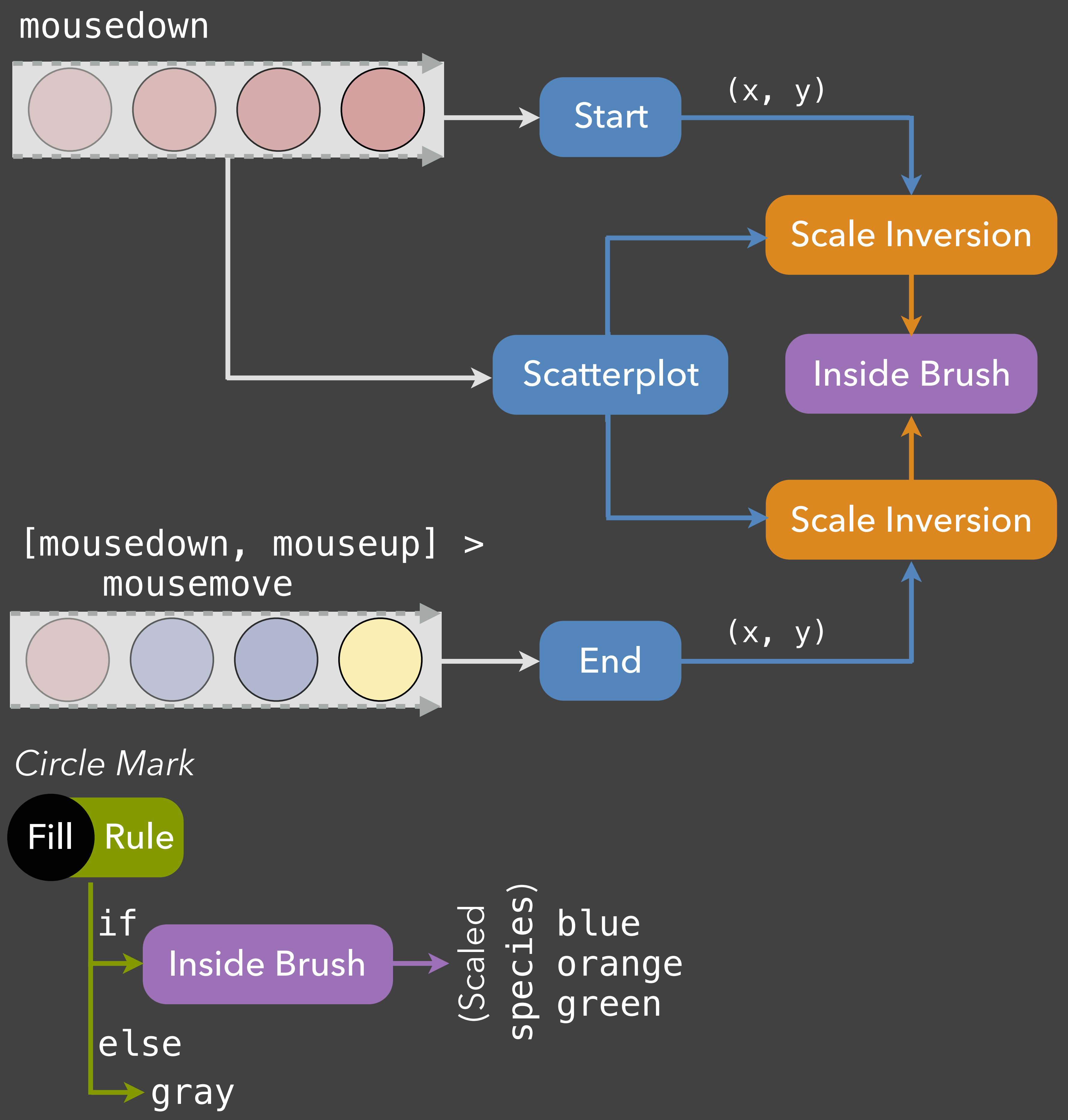
Declarative Interaction Design

✓ Faster iteration + accessible to a larger audience.

✓ Performance + scalability.

At least 2x faster than D3 + callbacks†.

† http://github.com/vega/vega-benchmarks



Declarative Interaction Design

✓ Faster iteration + accessible to a larger audience.

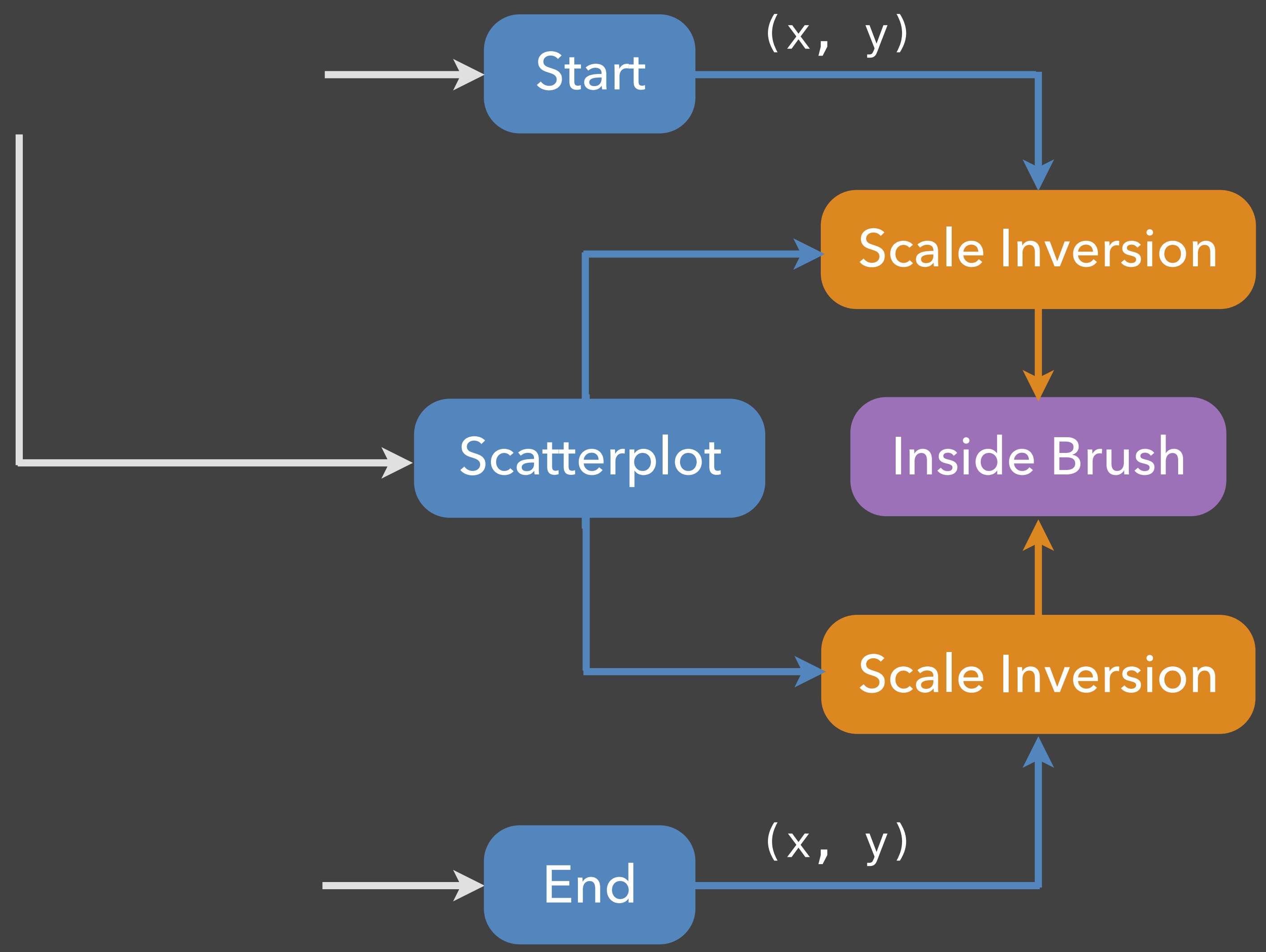
✓ Performance + scalability.

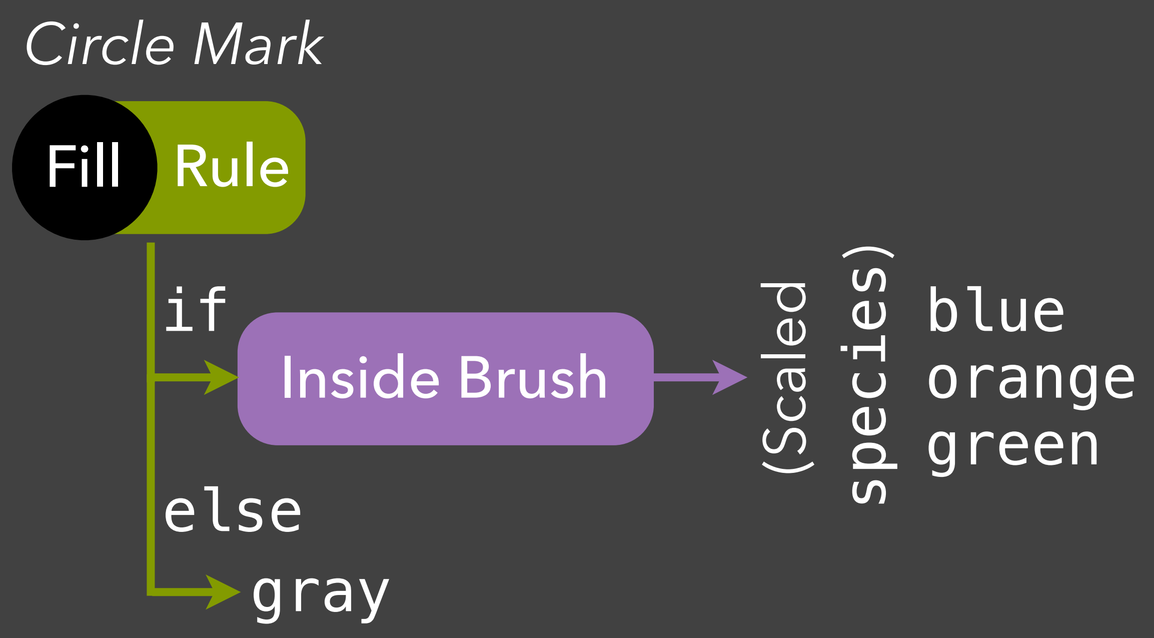
At least 2x faster than D3 + callbacks†.

✓ Reuse + portability.

Write once. Re-apply with different input data. Re-target to multiple devices, renderers, or modalities.

† http://github.com/vega/vega-benchmarks





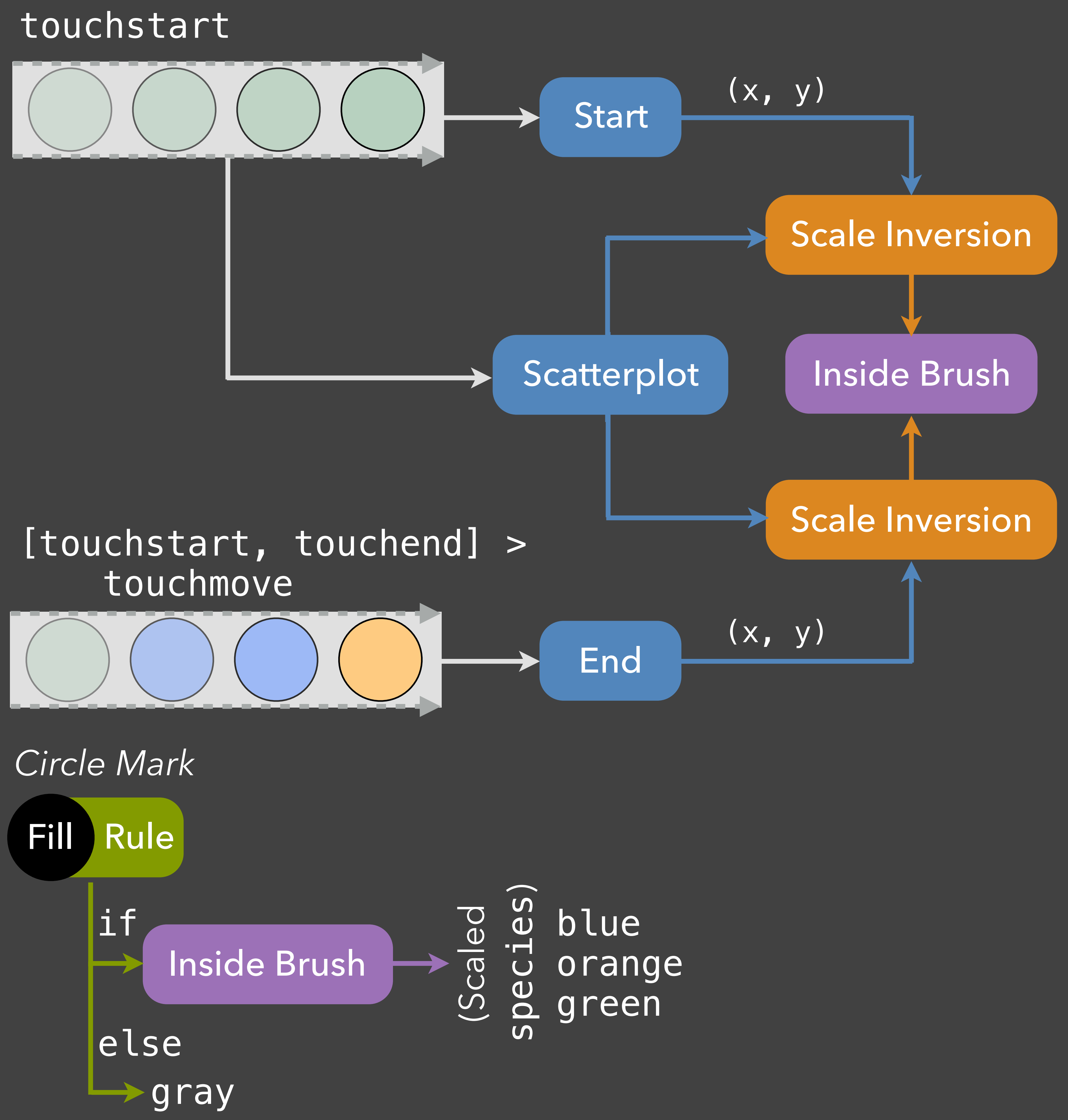
Declarative Interaction Design

- ✓ Faster iteration + accessible to a larger audience.
- ✓ Performance + scalability.

  At least 2x faster than D3 + callbacks†.
- ✓ Reuse + portability.

  Write once. Re-apply with different input data. Re-target to multiple devices, renderers, or modalities.

† http://github.com/vega/vega-benchmarks



Declarative Interaction Design

- ✓ Faster iteration + accessible to a larger audience.
- ✓ Performance + scalability.

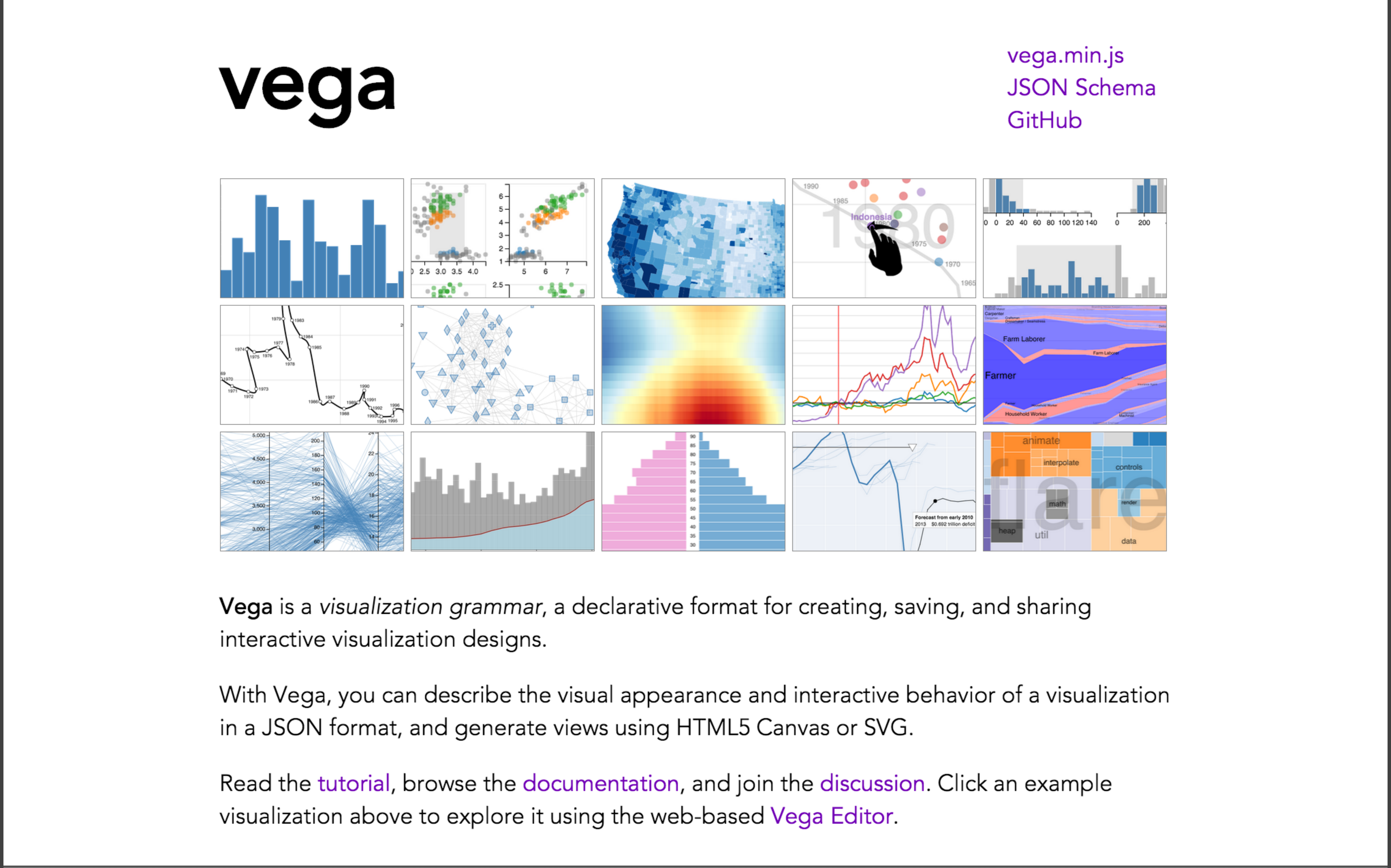
  At least 2x faster than D3 + callbacks†.
- ✓ Reuse + portability.

  Write once. Re-apply with different input data. Re-target to multiple devices, renderers, or modalities.

† http://github.com/vega/vega-benchmarks

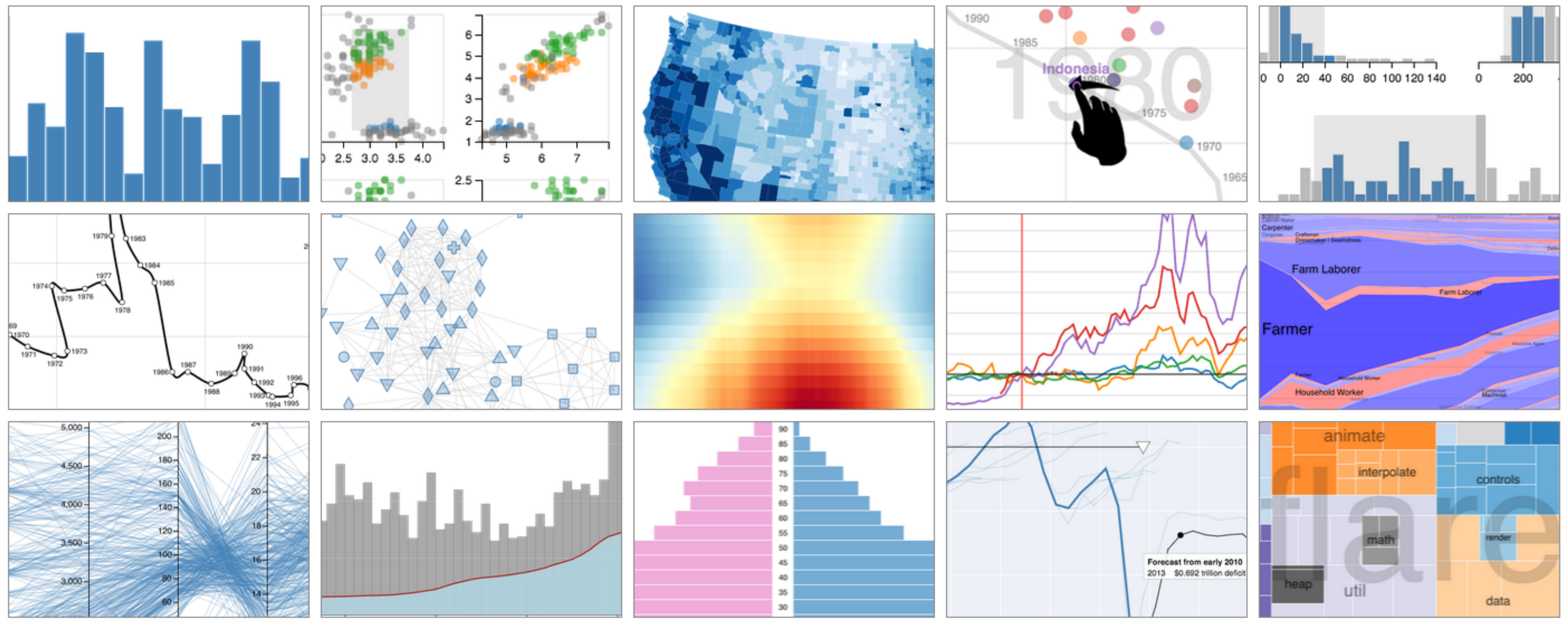
Demo

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



vega

vega.min.js
JSON Schema
GitHub



Vega is a *visualization grammar*, a declarative format for creating, saving, and sharing interactive visualization designs.

With Vega, you can describe the visual appearance and interactive behavior of a visualization in a JSON format, and generate views using HTML5 Canvas or SVG.

Read the tutorial, browse the documentation, and join the discussion. Click an example visualization above to explore it using the web-based Vega Editor.

vega.github.io/vega/

One more thing...

Interactive Vega-Lite

Interactive Vega-Lite

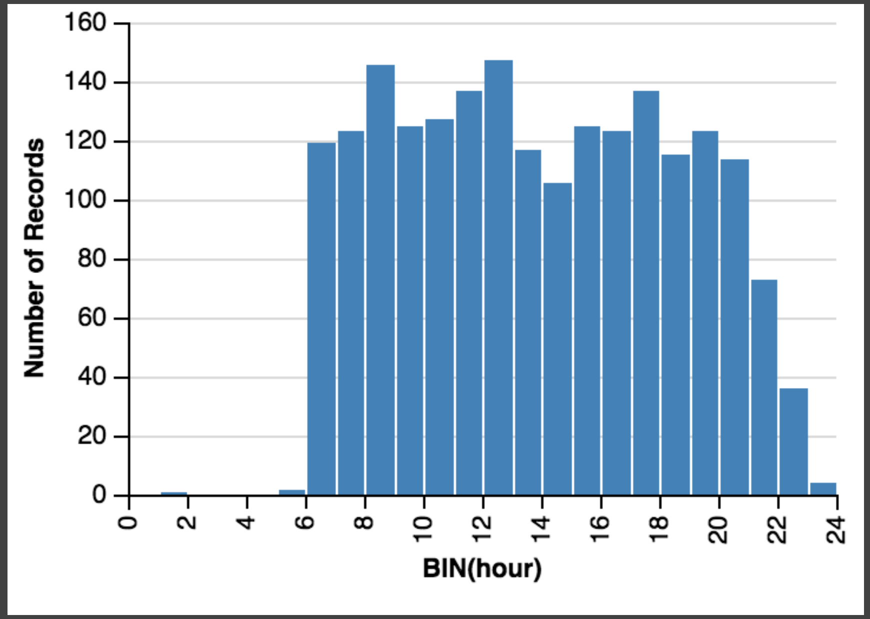
(A Sneak Peak)

```
{
  "data": {"url": "data/flights.json"},
  "mark": "bar",
  "encoding": {
    "x": {"field": "hour", "bin": true, "type": "quantitative",},
    "y": {"field": "*", "aggregate": "count", "type": "quantitative"}
  }
}
```

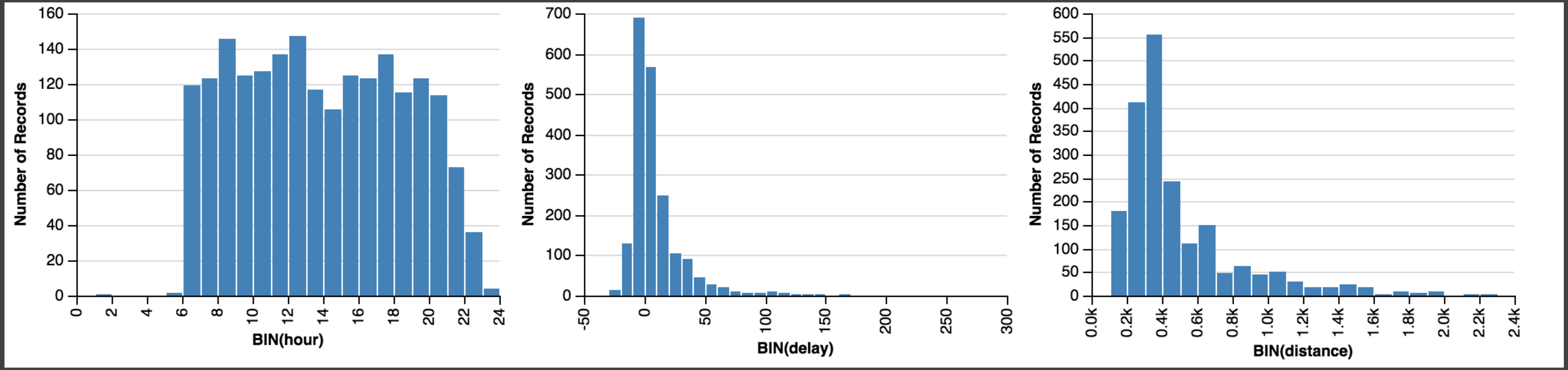
Data

Mark

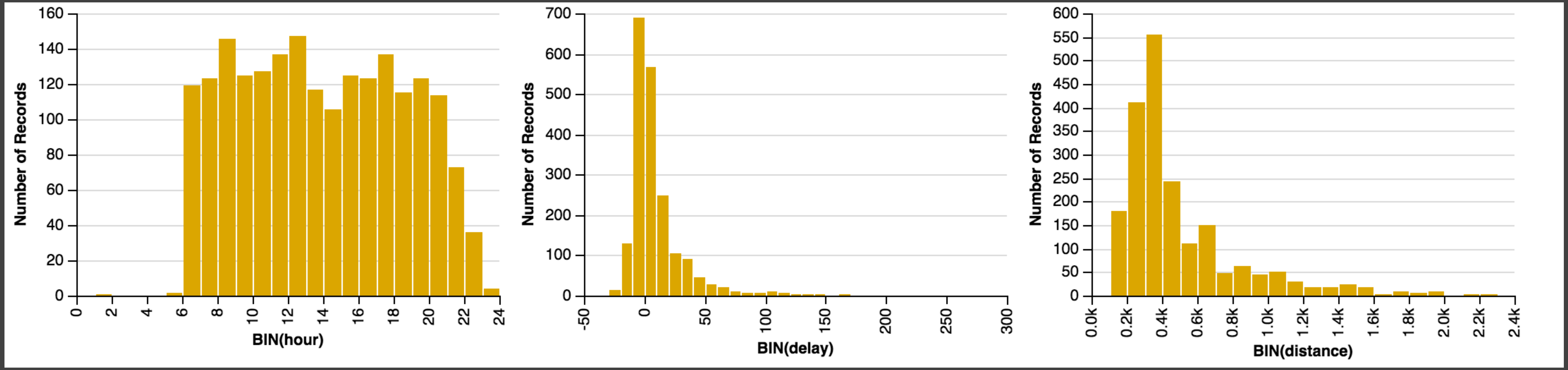
Transforms + *Scales & Guides (not shown)*



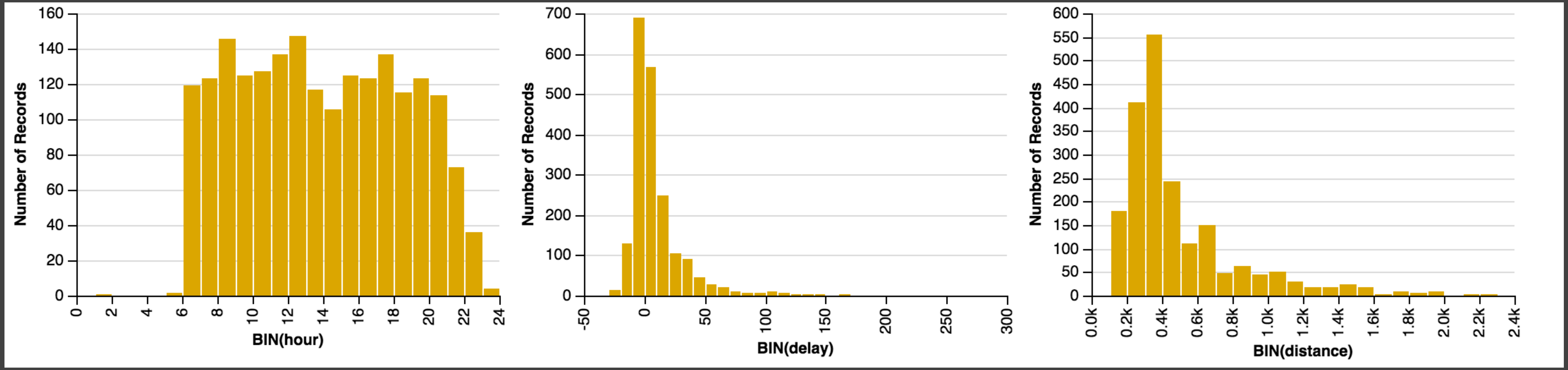
```
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "data": {"url": "data/flights.json"},
    "mark": "bar",
    "encoding": {
      "x": {"field": {"repeat": "column"}, "bin": true, "type": "quantitative"},
      "y": {"field": "*", "aggregate": "count", "type": "quantitative"}
    }
  }
}
```



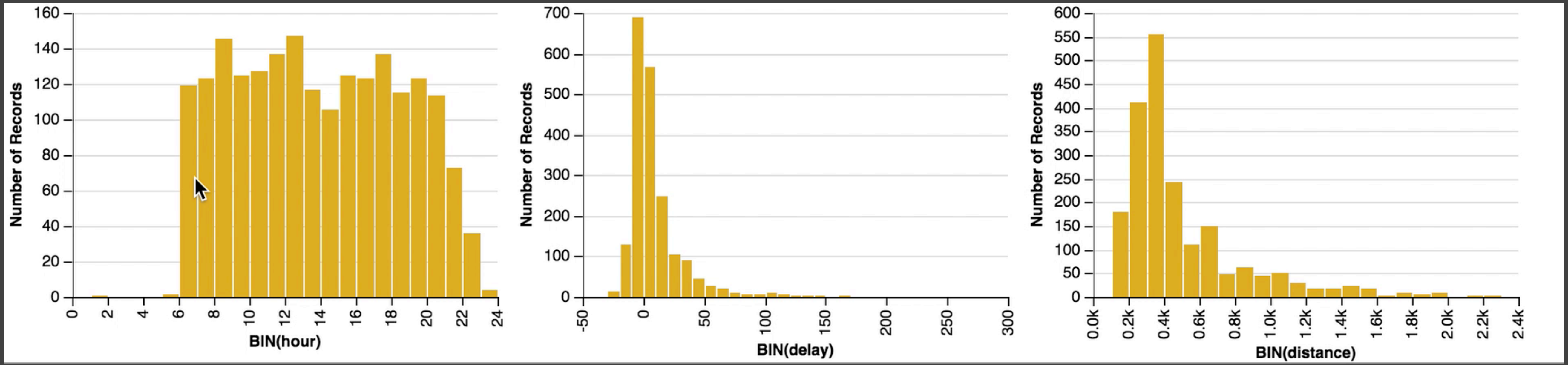
```
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "layers": [{
      "data": {"url": "data/flights.json"},
      "mark": "bar",
      "encoding": {
        "x": {"field": {"repeat": "column"}, "bin": true, "type": "quantitative"},
        "y": {"field": "*", "aggregate": "count", "type": "quantitative"}
      }
    }, {
      ...,
      "encoding": {
        ...,
        "color": {"value": "goldenrod"}
      }
    }]
  }
}
```



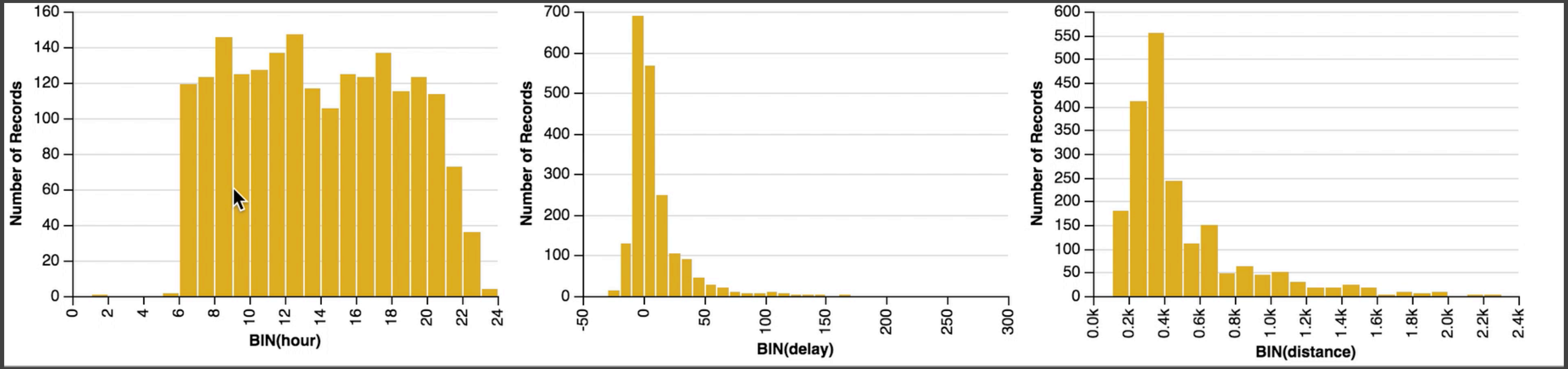
```
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "layers": [{
      ...,






    }, {
      ...,

    }]
  }
}
```



```
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "layers": [{
      ...,
      "select": {
        "region": {
          "type": "interval", "project": {"channels": ["x"]}, ...
        }
      }
    }, {
      ...,

    }]
  }
}
```

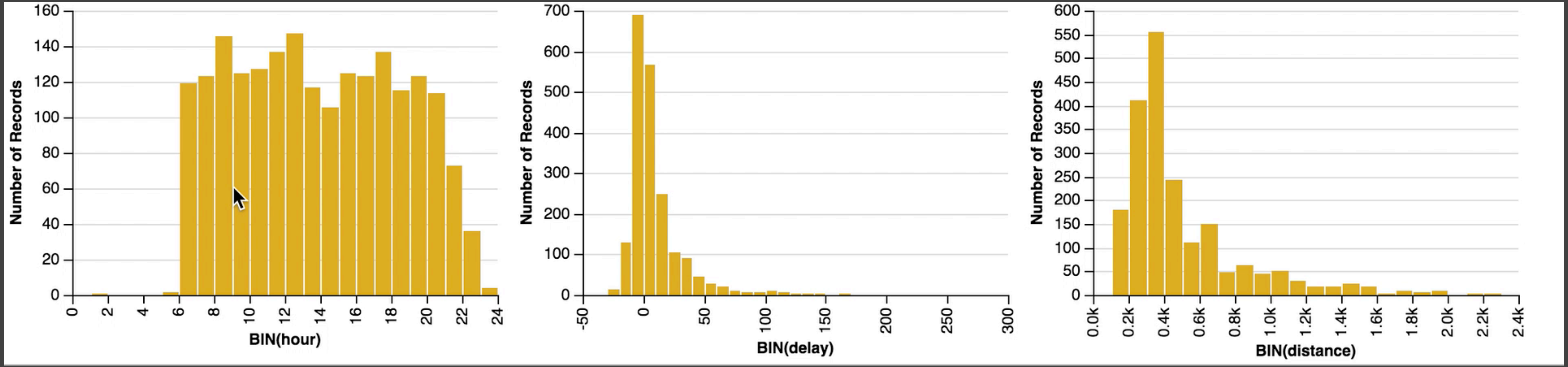


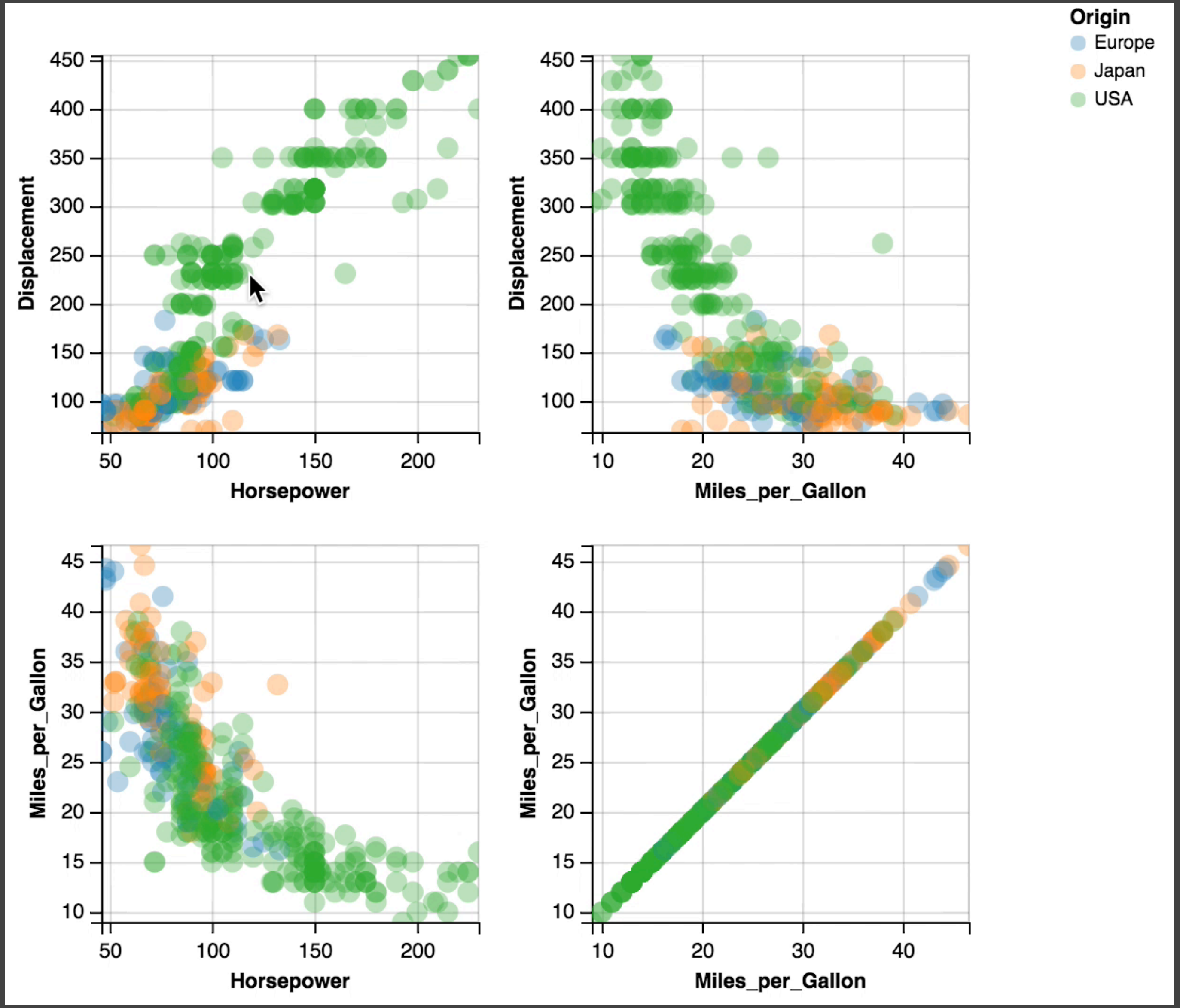
```
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "layers": [{
      ...,
      "select": {
        "region": {
          "type": "interval", "project": {"channels": ["x"]}, ...
        }
      }
    }, {
      ...,
      "transform": {"filterWith": "region"}
    }]
  }
}
```

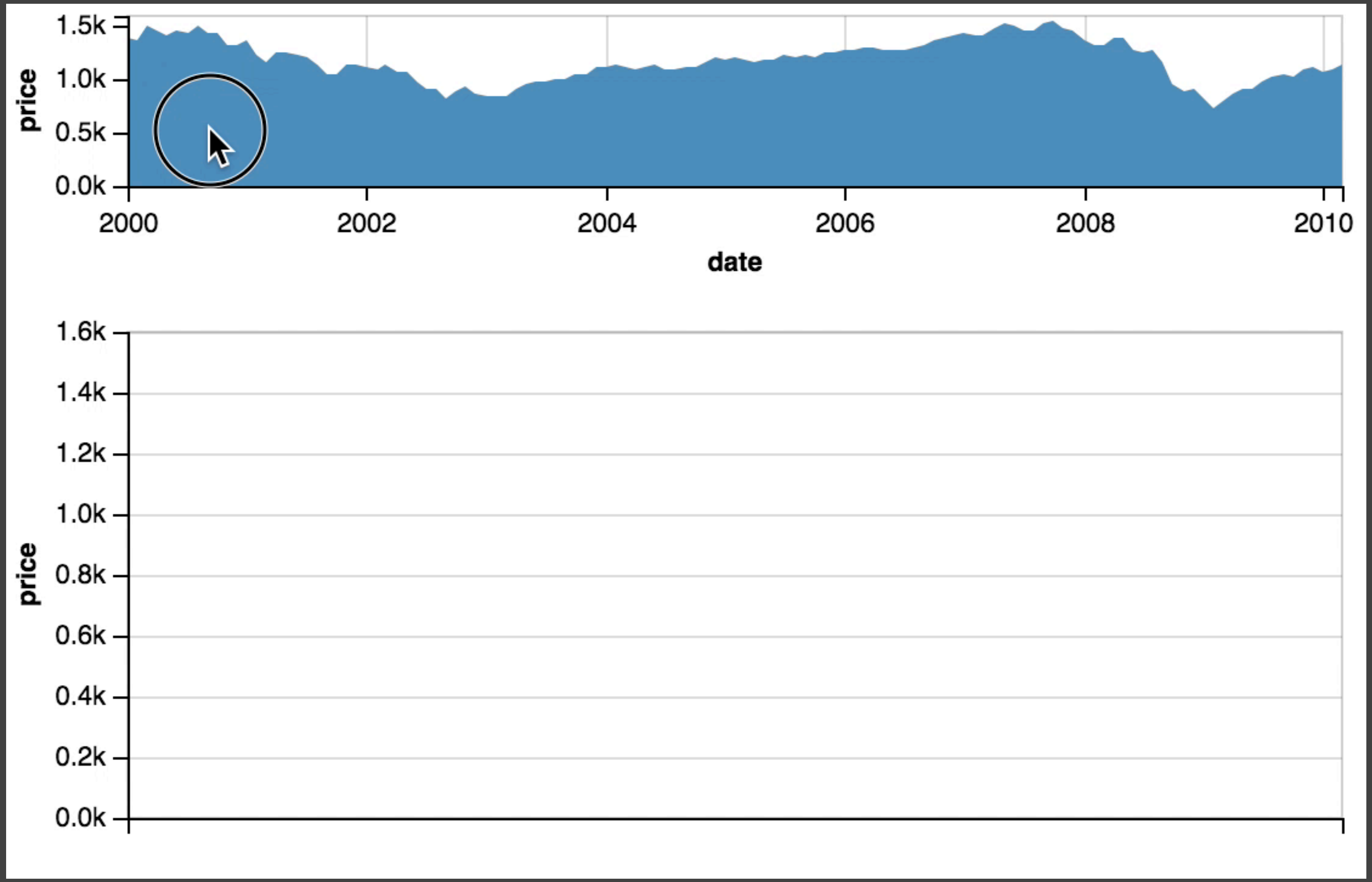


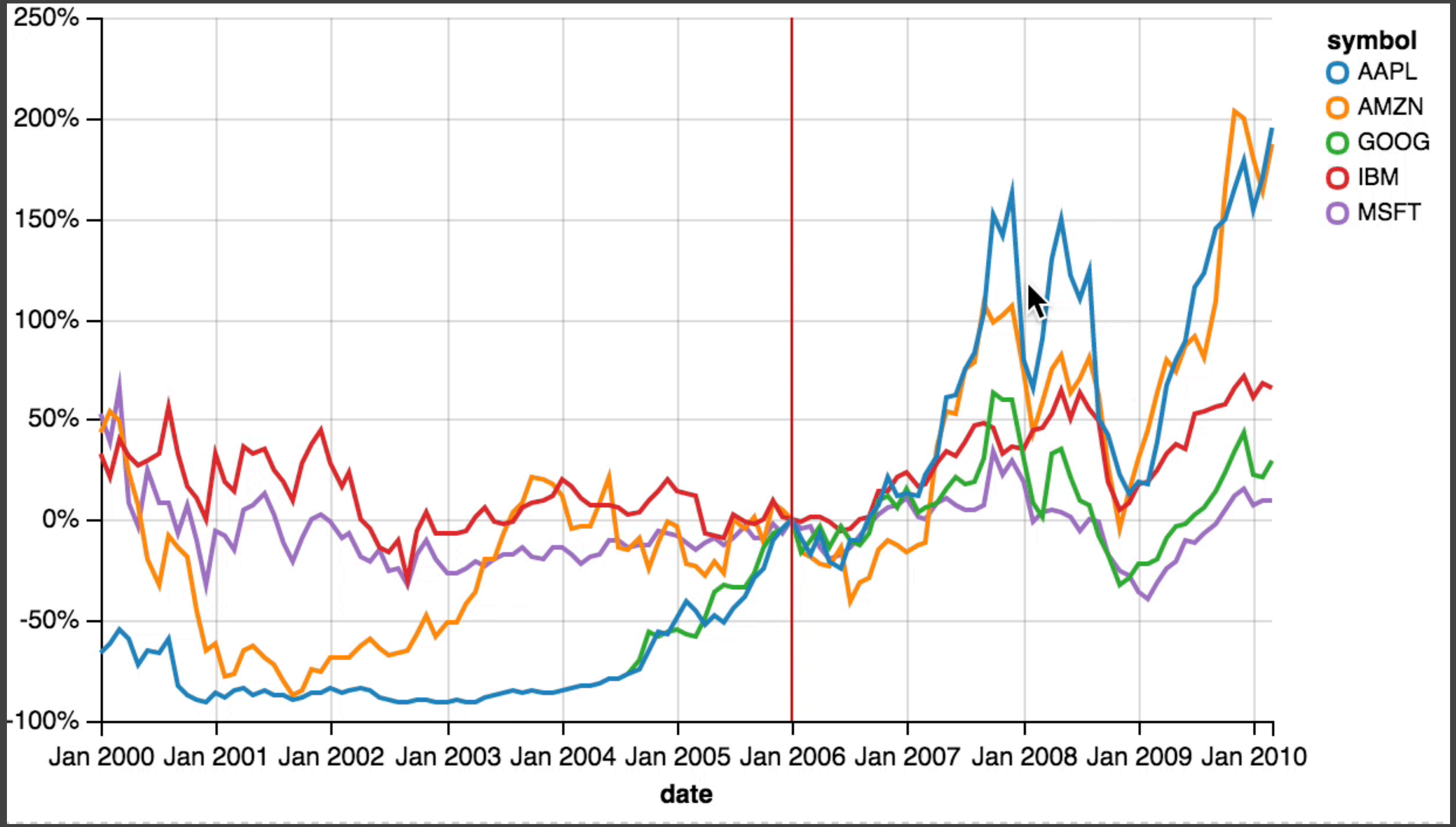
```
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "layers": [{
      ...,
      "select": {
        "region": {
          "type": "interval", "project": {"channels": ["x"]}, ...
        }
      }
    }, {
      ...,
      "transform": {"filterWith": "region"}
    }]
  }
}
```

35 Lines of JSON!

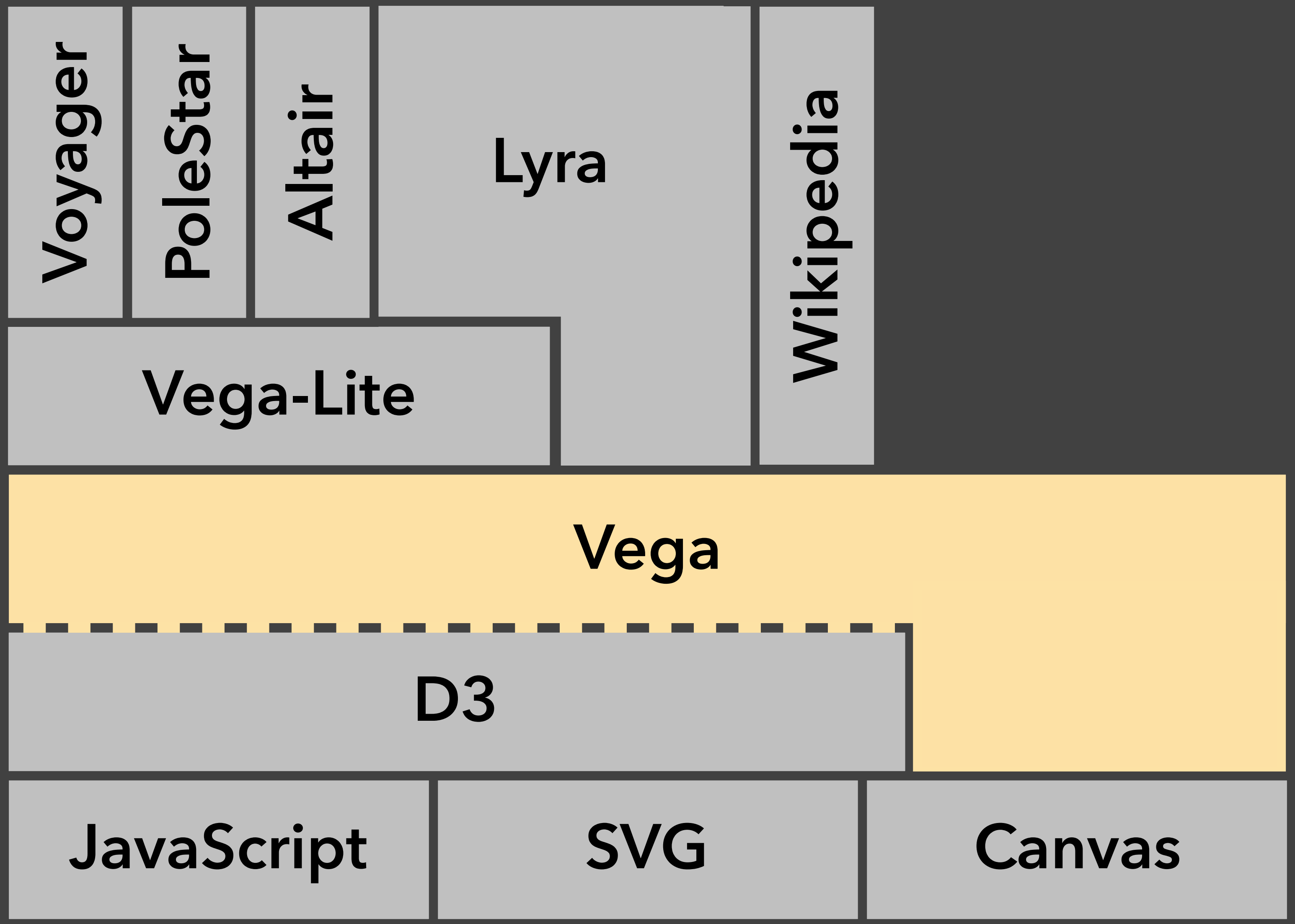



Origin
Europe
Japan
USA
Displacement
Horsepower
Miles_per_Gallon

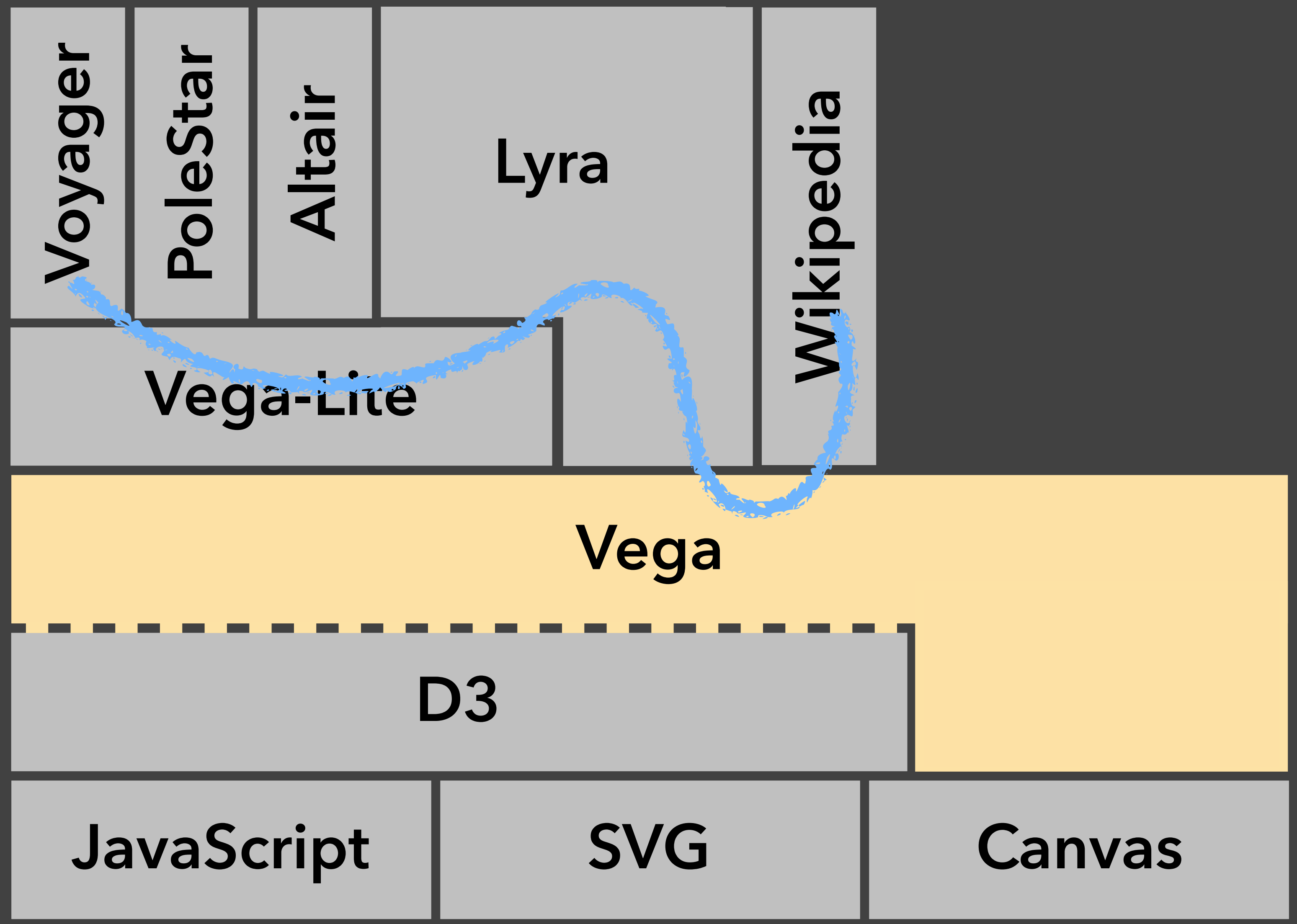

price
date
2000
2002
2004
2006
2008
2010


symbol
AAPL
AMZN
GOOG
IBM
MSFT
date
Jan 2000
Jan 2001
Jan 2002
Jan 2003
Jan 2004
Jan 2005
Jan 2006
Jan 2007
Jan 2008
Jan 2009
Jan 2010

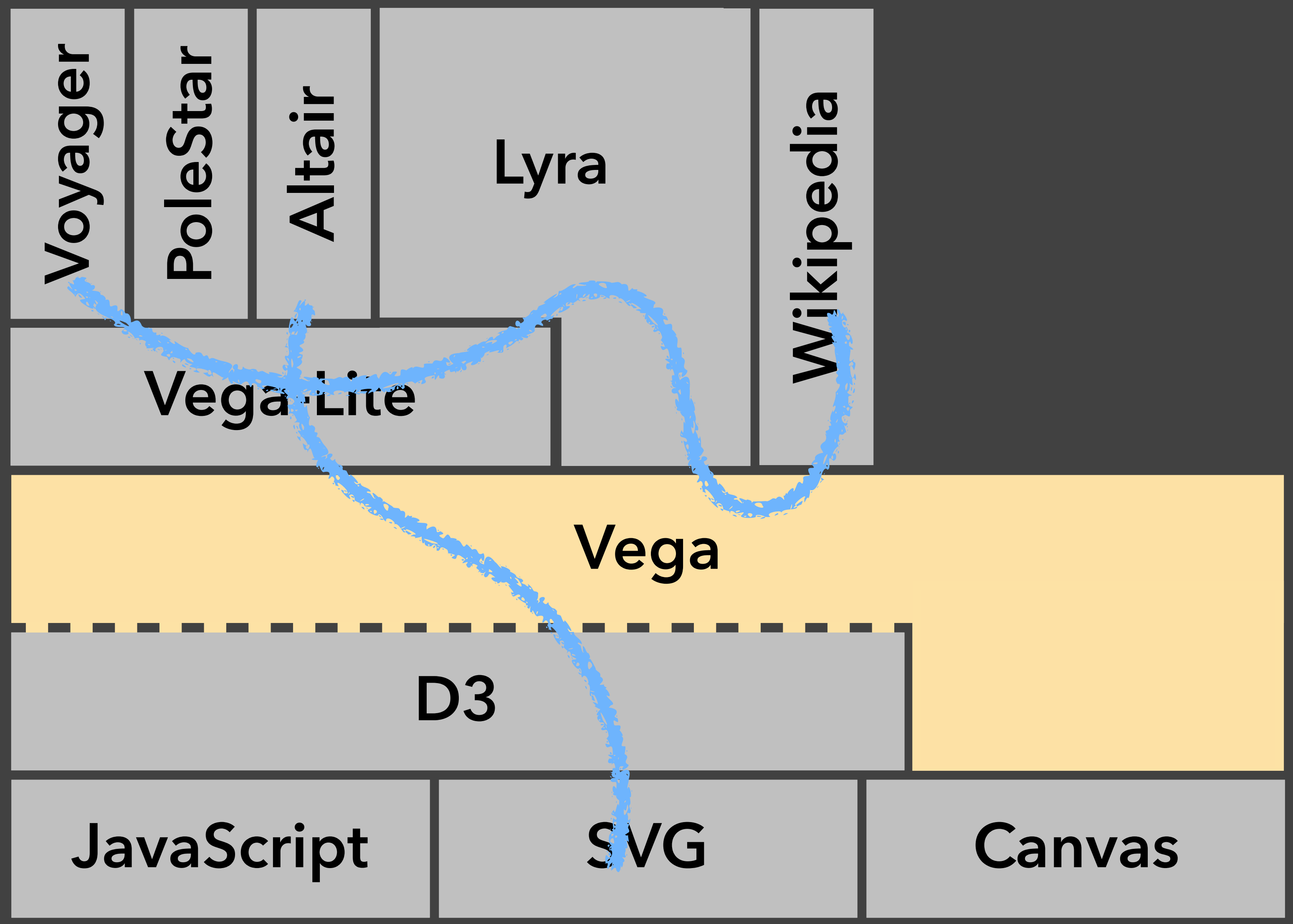
Vega: A Platform for Visualization



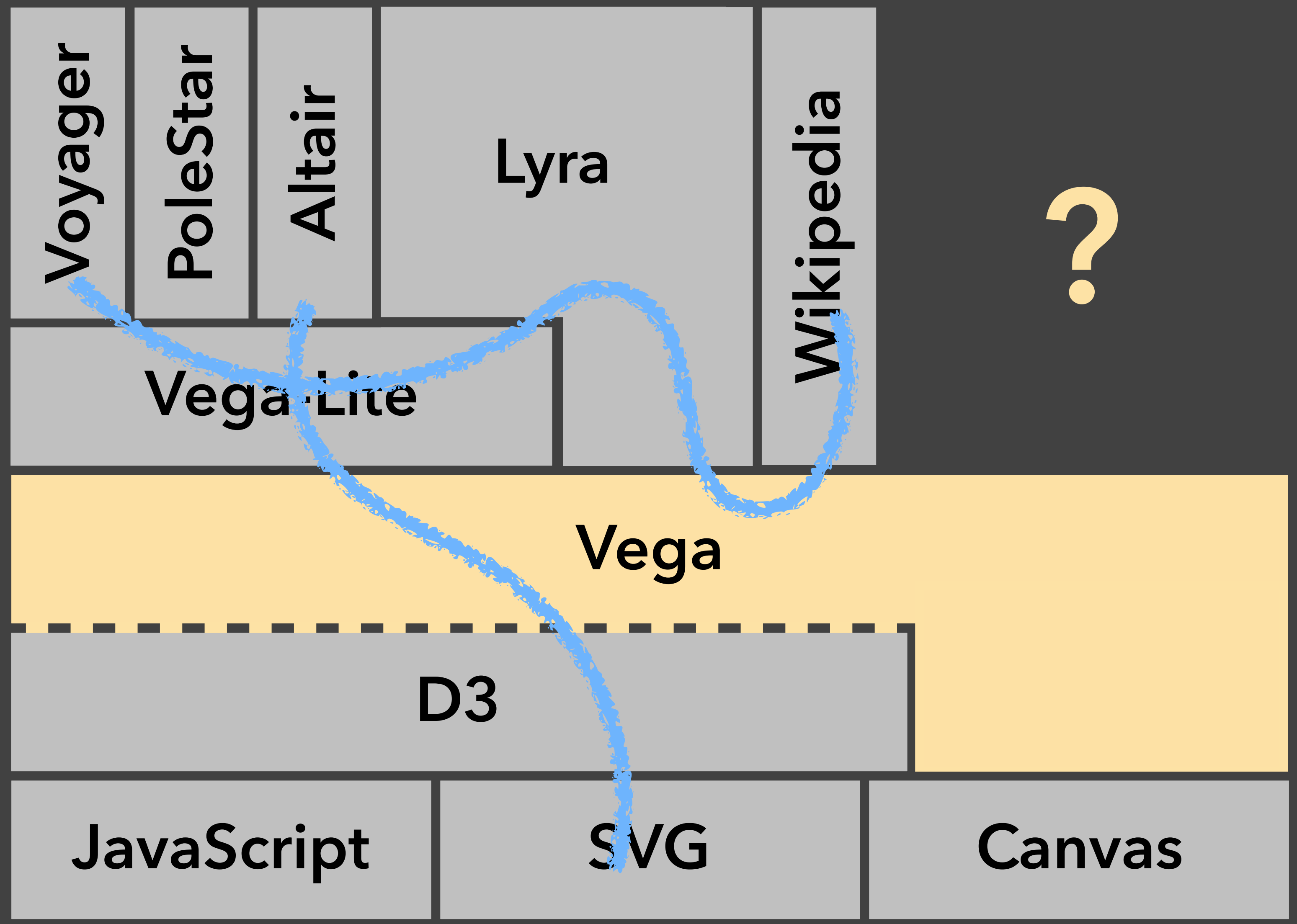
Vega: A Platform for Visualization



Vega: A Platform for Visualization



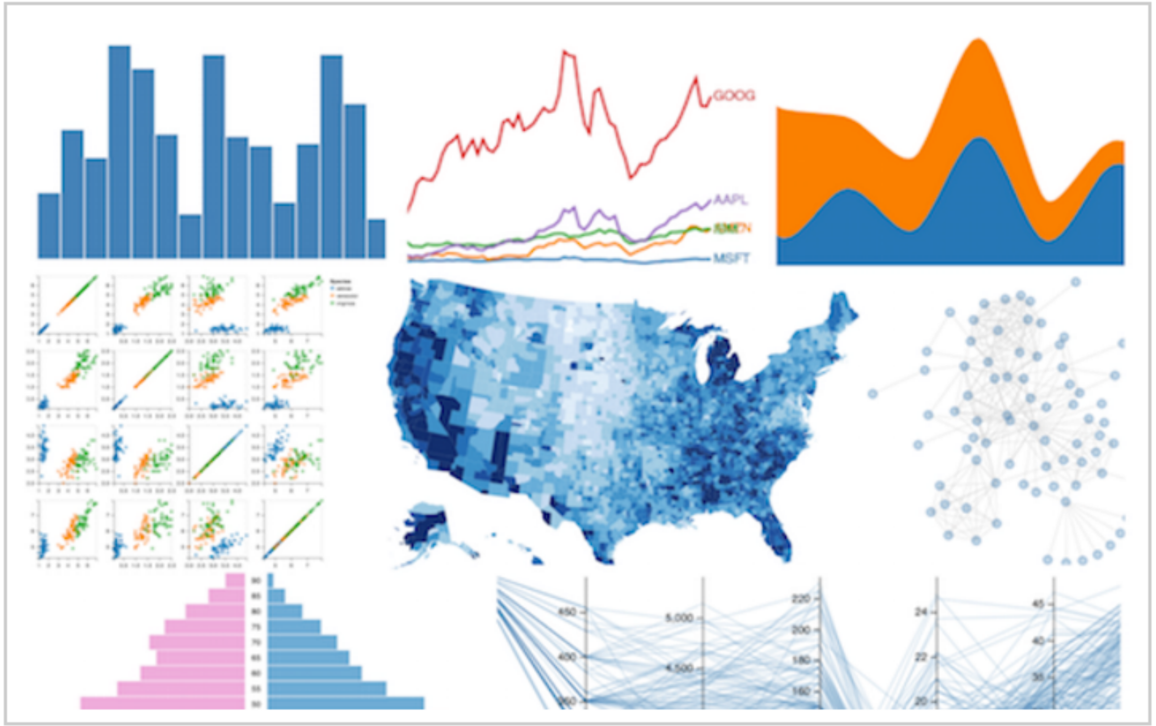
Vega: A Platform for Visualization

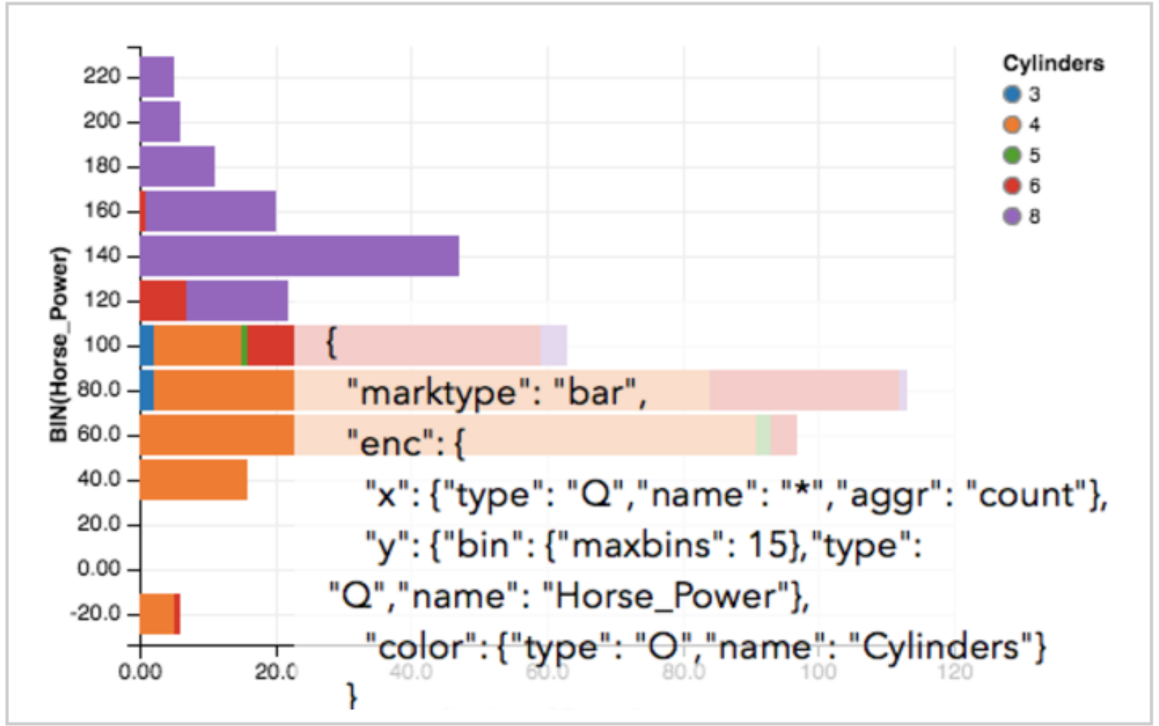


Vega A VISUALIZATION GRAMMAR

Vega is a declarative format for creating, saving, and sharing visualization designs. With Vega, visualizations are described in JSON, and generate interactive views using either HTML5 Canvas or SVG.

TOOLKITS





```
{
  "marktype": "bar",
  "enc": {
    "x": {"type": "Q","name": "*","aggr": "count"},
    "y": {"bin": {"maxbins": 15},"type":
 "Q","name": "Horse_Power"},
    "color": {"type": "O","name": "Cylinders"}
  }
```

VEGA 2.0 offers a full declarative visualization grammar, suitable for expressive custom interactive visualization design and programmatic generation.

Online Editor & Examples | Documentation | GitHub

VEGA-LITE provides a higher-level grammar for visual analysis, comparable to ggplot or Tableau, that generates complete Vega specifications.

Online Editor | Examples | Documentation | GitHub

vega.github.io