Reactive Vega

A Streaming Dataflow Architecture for Declarative Interactive Visualization

Arvind Satyanarayan @arvindsatya1
Stanford University

Ryan Russell
Jane Hoffswell
Jeffrey Heer @jeffrey_heer
University of Washington

Reactive Vega

A Streaming Dataflow Architecture for
Declarative Interactive Visualization

Arvind Satyanarayan @arvindsatya1
Stanford University

Ryan Russell
Jane Hoffswell
Jeffrey Heer @jeffrey_heer
University of Washington

Reactive Vega

A Streaming Dataflow Architecture for Declarative Interactive Visualization

Arvind Satyanarayan @arvindsatya1
Stanford University

Ryan Russell
Jane Hoffswell
Jeffrey Heer @jeffrey_heer
University of Washington

Reactive Vega

A Streaming Dataflow Architecture for Declarative Interactive Visualization

Arvind Satyanarayan @arvindsatya1
Stanford University

Ryan Russell
Jane Hoffswell
Jeffrey Heer @jeffrey_heer
University of Washington

Reactive Vega

A Streaming Dataflow Architecture for Declarative Interactive Visualization

Arvind Satyanarayan @arvindsatya1
Stanford University

Ryan Russell
Jane Hoffswell
Jeffrey Heer @jeffrey_heer
University of Washington

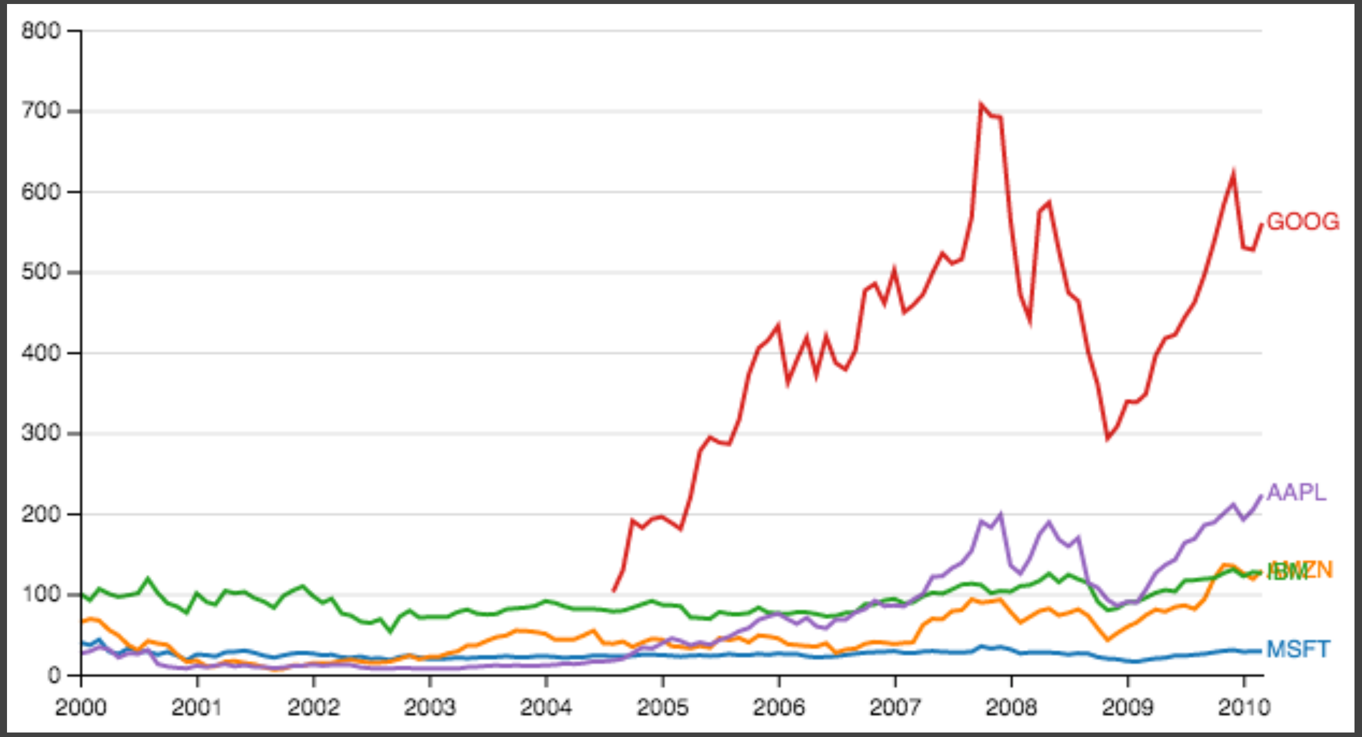
Reactive Vega

A Streaming Dataflow Architecture for Declarative Interactive Visualization

Arvind Satyanarayan @arvindsatya1
Stanford University

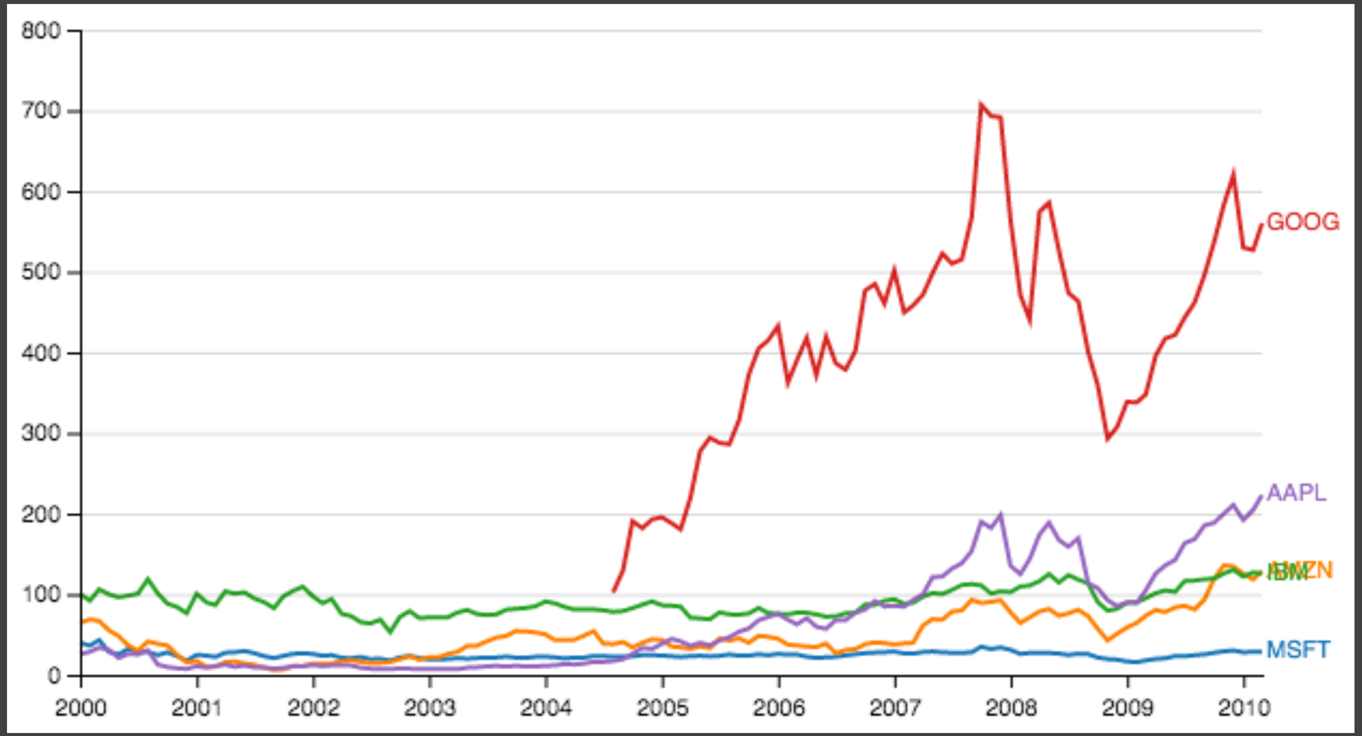
Ryan Russell
Jane Hoffswell
Jeffrey Heer @jeffrey_heer
University of Washington

```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```



```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```

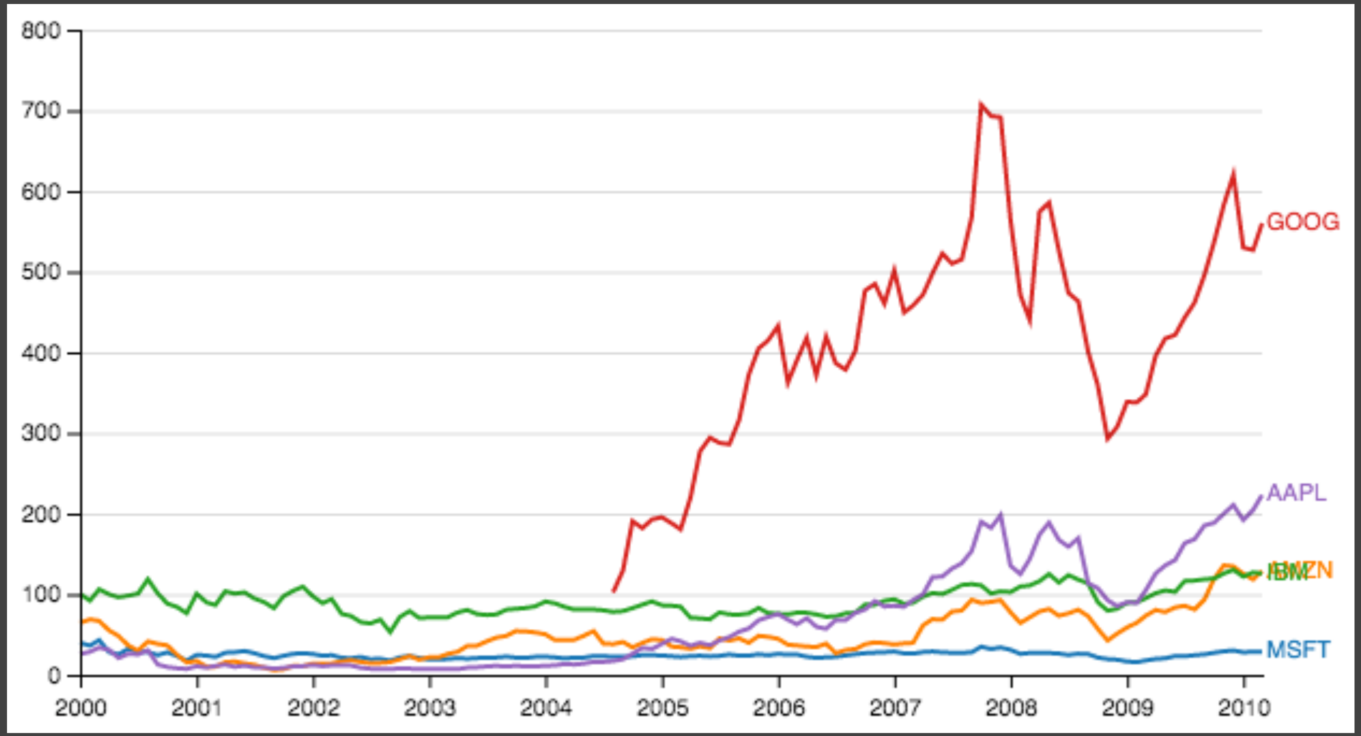
Data + Transforms



```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```

Data + Transforms

Scales

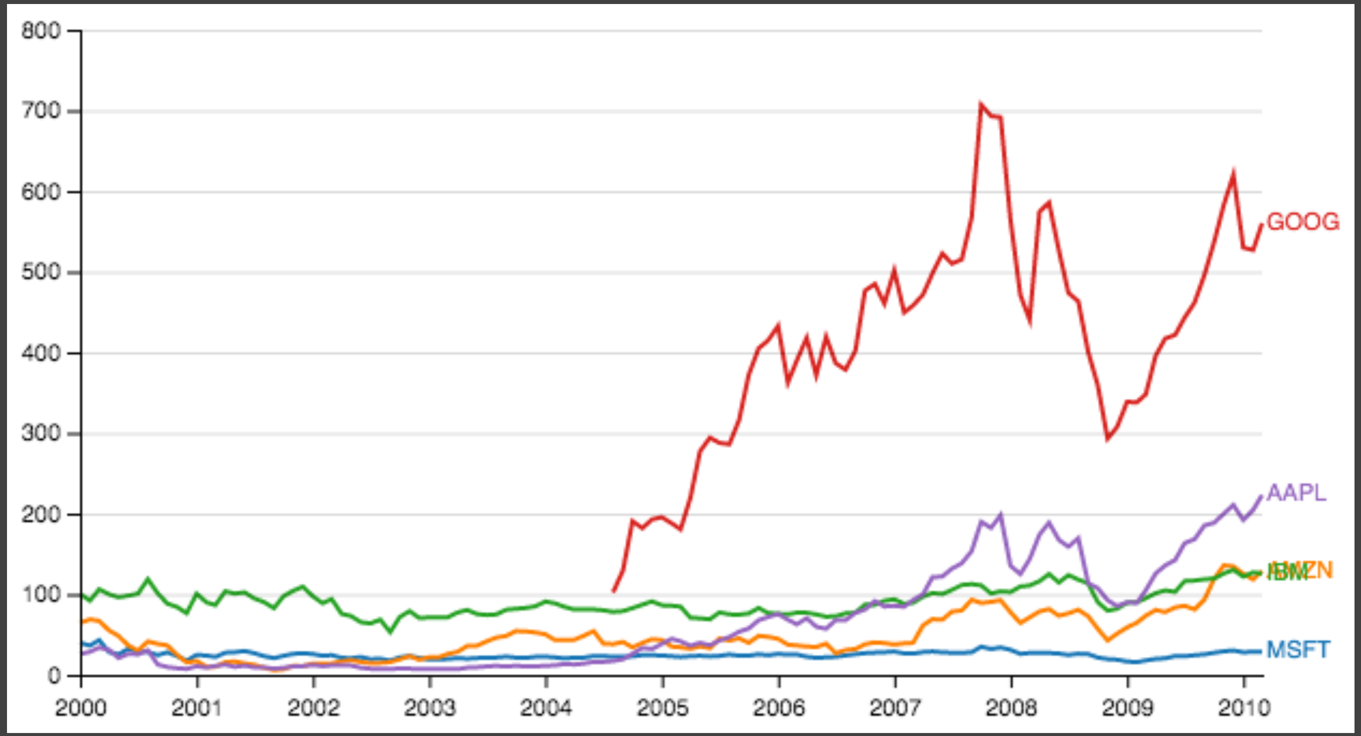


```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```

Data +
Transforms

Scales

Guides



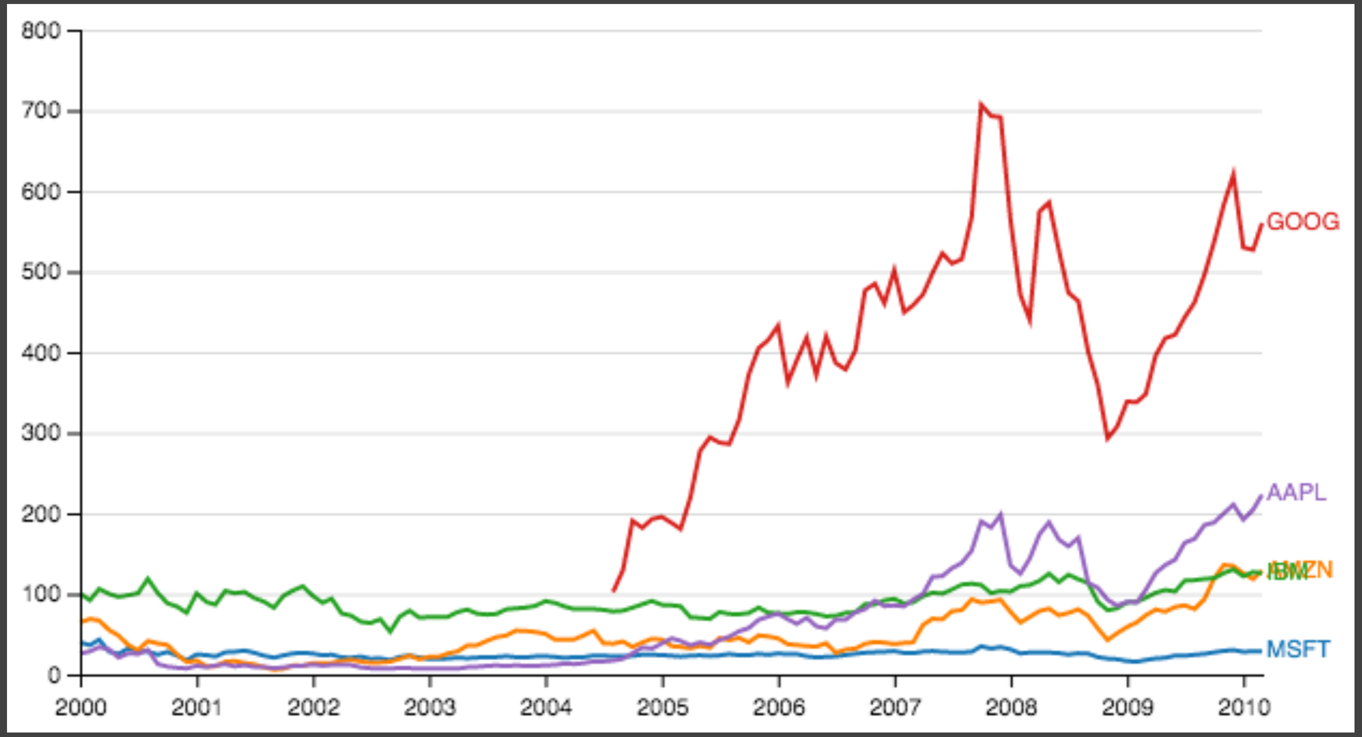
```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```

Data +
Transforms

Scales

Guides

Marks



```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```

Data + Transforms

Scales

Guides

Marks

Declarative specification: describes *what* the visualization should look like vs. *how* it should be computed.

```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```

Data +
Transforms

Scales

Guides

Marks

Declarative specification: describes *what* the visualization should look like vs. *how* it should be computed.

✓ Less code + faster iteration.

Accessible to a larger audience.

```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```

Data + Transforms

Scales

Guides

Marks

Declarative specification: describes *what* the visualization should look like vs. *how* it should be computed.

✓ Less code + faster iteration.

Accessible to a larger audience.

✓ Performance + scalability.

```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```

Data + Transforms

Scales

Guides

Marks

Declarative specification: describes *what* the visualization should look like vs. *how* it should be computed.

✓ Less code + faster iteration.

Accessible to a larger audience.

✓ Performance + scalability.

✓ **Reuse + portability.**

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

```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```

Data + Transforms

Scales

Guides

Marks

Declarative specification: describes *what* the visualization should look like vs. *how* it should be computed.

✓ Less code + faster iteration.

Accessible to a larger audience.

✓ Performance + scalability.

✓ **Reuse + portability.**

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

✓ **Programmatic Generation.**

Higher-level software for creating and recommending visualizations.

What about *interaction*?

What about *interaction*?

“A graphic is not ‘drawn’ once and for all; it is ‘constructed’ and reconstructed until it reveals all the relationships constituted by the interplay of the data. The best graphic operations are those carried out by the decision-maker himself.”

– Jacques Bertin, 1981.

The Problem with *Imperative* Interaction

```
d3.selectAll("rect")
  .on("mousedown", function() {

  })
  .on("mouseup", function() {


  })
  .on("mousemove", function() {





  });
```

The Problem with *Imperative* Interaction

```
d3.selectAll("rect")
  .on("mousedown", function() {

  })
  .on("mouseup", function() {

    d3.event.stopPropagation();
  })
  .on("mousemove", function() {
    var e = d3.event;

    d3.select(this)


  });
```

1. Inconsistent syntactic forms for similar semantics.
   Blackwell et al. *Cognitive Dimensions of Notations*. 2001.

The Problem with *Imperative* Interaction

```
var dragging = false;
d3.selectAll("rect")
  .on("mousedown", function() {
     dragging = true;
  })
  .on("mouseup", function() {
      dragging = false;
     d3.event.stopPropagation();
  })
  .on("mousemove", function() {
     var e = d3.event;
     if(!dragging) return;
     d3.select(this)


  });
```

1. Inconsistent syntactic forms for similar semantics.
   Blackwell et al. *Cognitive Dimensions of Notations.* 2001.

2. Manually maintain state and dependencies.
   Myers. *Eliminating the Spaghetti of Callbacks.* UIST 2001.
   Cooper et al. *Programming Languages and Systems.* 2006.

The Problem with *Imperative* Interaction

```
var dragging = false;
d3.selectAll("rect")
  .on("mousedown", function() {
     dragging = true;
  })
  .on("mouseup", function() {
      dragging = false;
     d3.event.stopPropagation();
  })
  .on("mousemove", function() {
     var e = d3.event;
     if(!dragging) return;
     d3.select(this)
         .attr("x", e.pageX)
         .attr("y", e.pageY);
  });
```

1. Inconsistent syntactic forms for similar semantics.
   Blackwell et al. *Cognitive Dimensions of Notations.* 2001.

2. Manually maintain state and dependencies.
   Myers. *Eliminating the Spaghetti of Callbacks.* UIST 2001.
   Cooper et al. *Programming Languages and Systems.* 2006.

3. "Side-effects" break encapsulation.
   Cooper. *Integrating dataflow evaluation into a practical higher-order call-by-value language.* 2008.

The Problem with *Imperative* Interaction

```
var dragging = false;
d3.selectAll("rect")
  .on("mousedown", function() {
     dragging = true;
  })
  .on("mouseup", function() {
      dragging = false;
     d3.event.stopPropagation();
  })
  .on("mousemove", function() {
     var e = d3.event;
     if(!dragging) return;
     d3.select(this)
         .attr("x", e.pageX)
         .attr("y", e.pageY);
  });
```

1. Inconsistent syntactic forms for similar semantics.
   Blackwell et al. *Cognitive Dimensions of Notations*. 2001.

2. Manually maintain state and dependencies.
   Myers. *Eliminating the Spaghetti of Callbacks*. UIST 2001.
   Cooper et al. *Programming Languages and Systems*. 2006.

3. "Side-effects" break encapsulation.
   Cooper. *Integrating dataflow evaluation into a practical higher-order call-by-value language*. 2008.

4. "Callback hell": execution order can be unpredictable and interleaved.
   Edwards. *Coherent Reaction*. SIGPLAN 2009.

Declarative Interaction

Data

Transforms

Scales

Guides

Marks

?

Declarative Interaction

Data

Transforms

Scales

Guides

Marks

?

Key Insights

Model user input as streaming data.

Declarative Interaction

Data

Transforms

Scales

Guides

Marks

?

Key Insights

Model user input as streaming data.

Adapt techniques from Functional Reactive Programming (FRP).

Declarative Interaction

Data

Transforms

Scales

Guides

Marks

Event Streams `[mousedown, mouseup] > mousemove`

Key Insights

Model user input as streaming data.

Adapt techniques from Functional Reactive Programming (FRP).

Satyanarayan et al. *UIST 2014*.

Declarative Interaction

Data	Event Streams	`[mousedown, mouseup] > mousemove`
Transforms	Signals	`minX := min(downX, event.x)`
Scales		
Guides		
Marks		

Key Insights
Model user input as streaming data.
Adapt techniques from Functional Reactive Programming (FRP).

Satyanarayan et al. *UIST 2014*.

Declarative Interaction

Data	Event Streams	`[mousedown, mouseup] > mousemove`
Transforms	Signals	`minX := min(downX, event.x)`
Scales	Scale Inversions	`minVal := xScale.invert(minX)`
Guides		
Marks		

Key Insights

Model user input as streaming data.

Adapt techniques from Functional Reactive Programming (FRP).

Declarative Interaction

Data	Event Streams	`[mousedown, mouseup] > mousemove`
Transforms	Signals	`minX := min(downX, event.x)`
Scales	Scale Inversions	`minVal := xScale.invert(minX)`
Guides	Predicates	`p(t) := t.value ∈ [minVal, maxVal]`
Marks		

Key Insights

Model user input as streaming data.

Adapt techniques from Functional Reactive Programming (FRP).

Declarative Interaction

Data	Event Streams	`[mousedown, mouseup] > mousemove`
Transforms	Signals	`minX := min(downX, event.x)`
Scales	Scale Inversions	`minVal := xScale.invert(minX)`
Guides	Predicates	`p(t) := t.value ∈ [minVal, maxVal]`

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

Key Insights

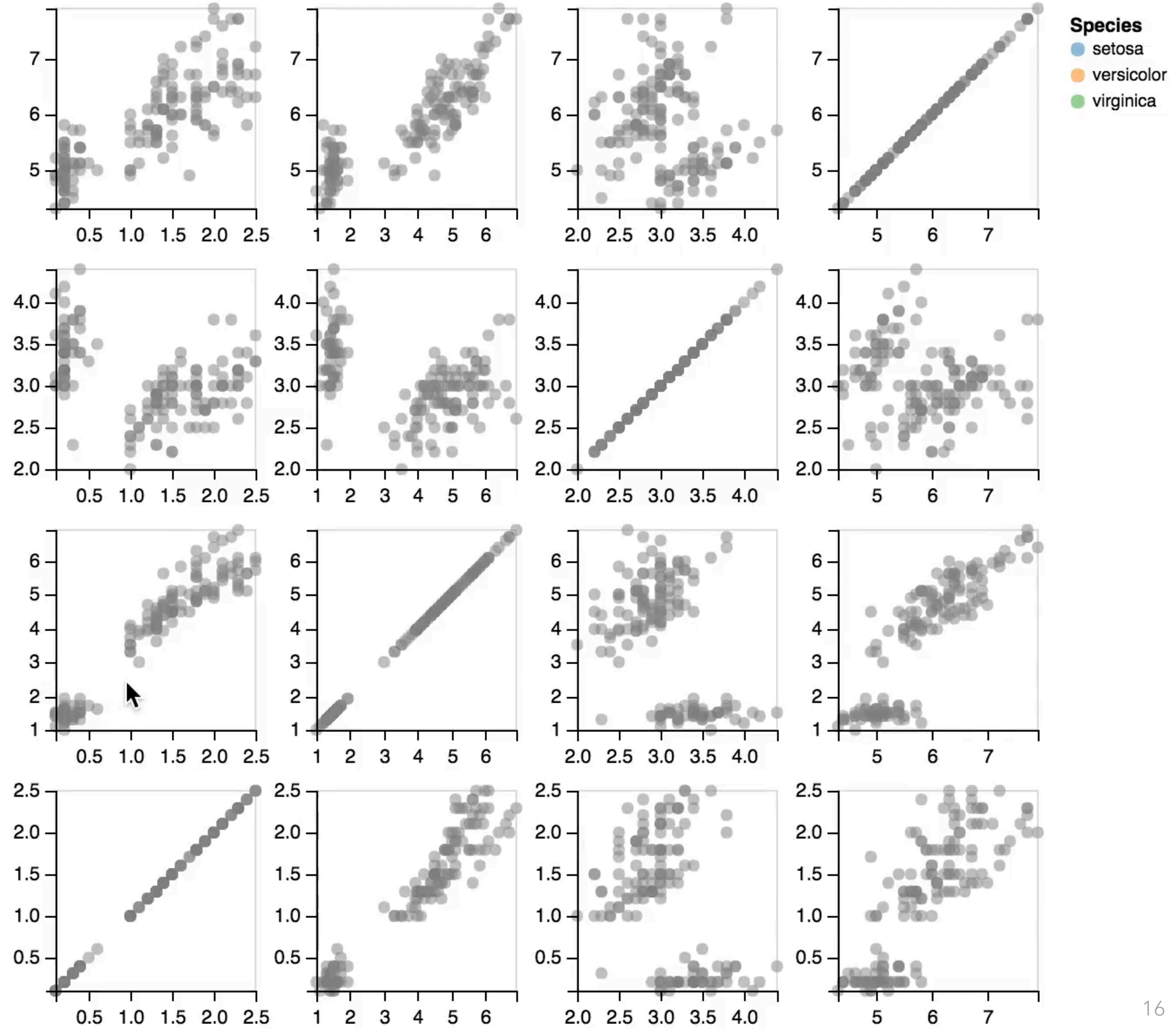
Model user input as streaming data.

Adapt techniques from Functional Reactive Programming (FRP).

Satyanarayan et al. *UIST 2014*.

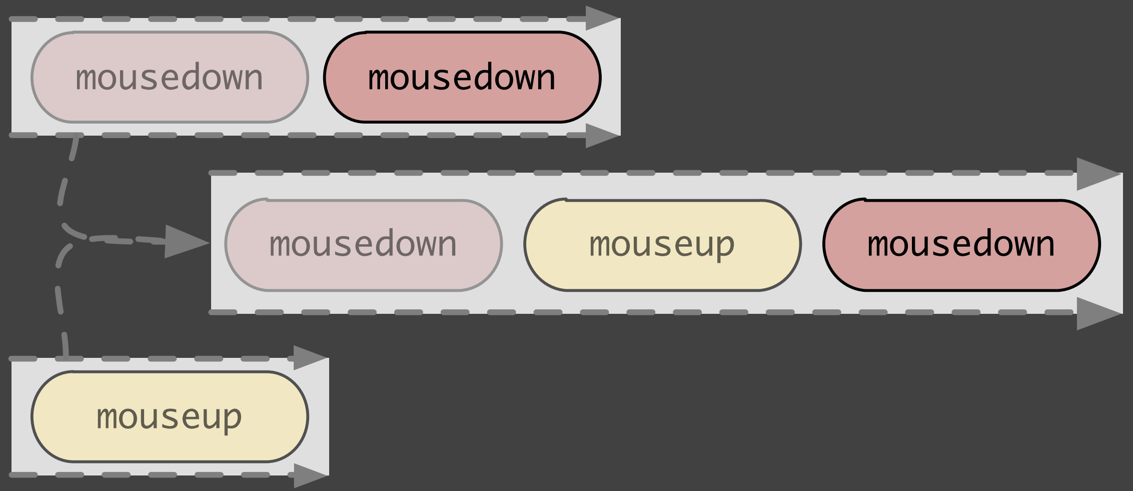
Example

Brushing & Linking



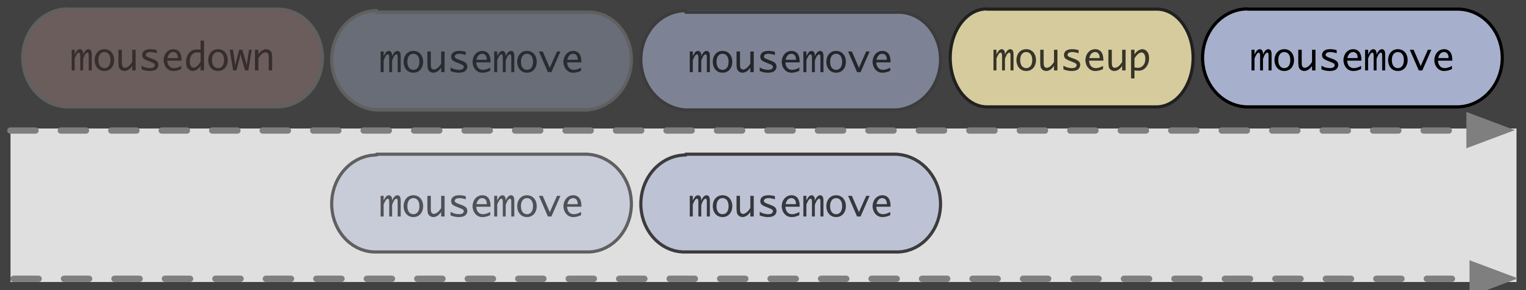
`*:mousedown, *:mouseup`

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



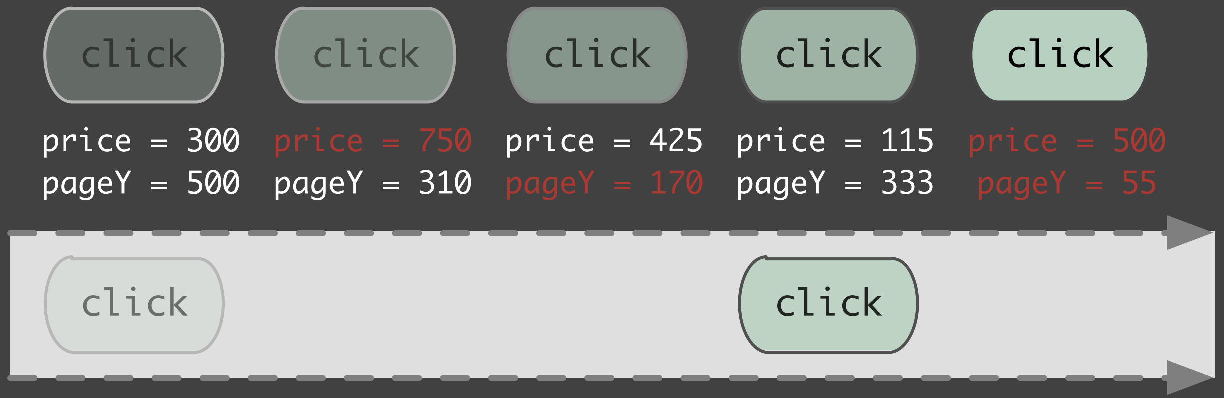
`[*:mousedown, *:mouseup] > *:mousemove`

drag



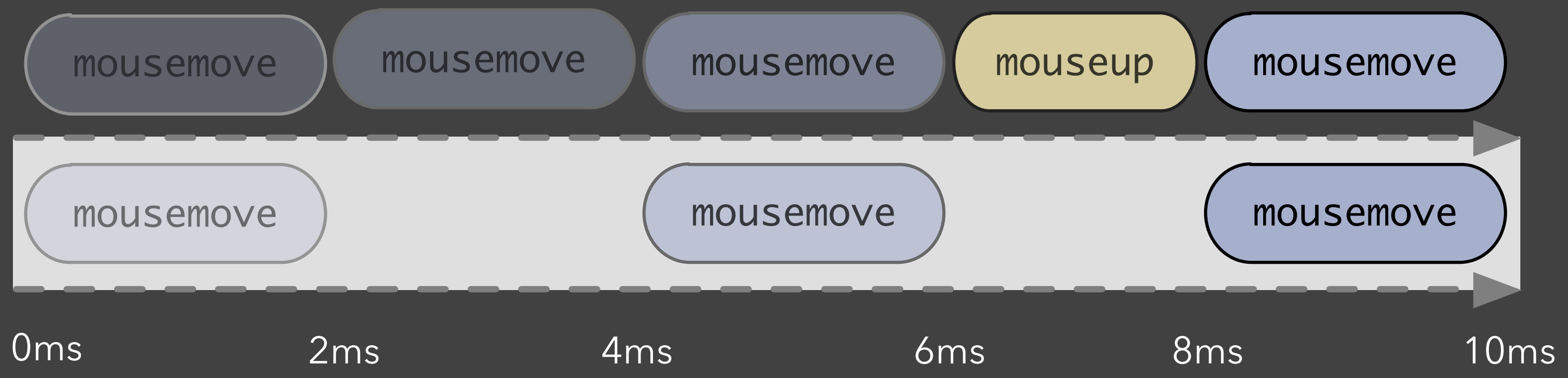
`*:click[event.pageY >= 300]`
`        [data.price < 500]`

filtered stream of `click` events

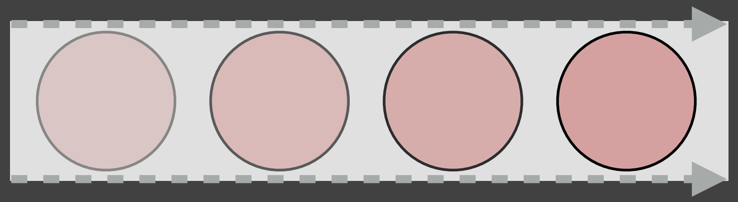


`*:mousemove{3ms,5ms}`

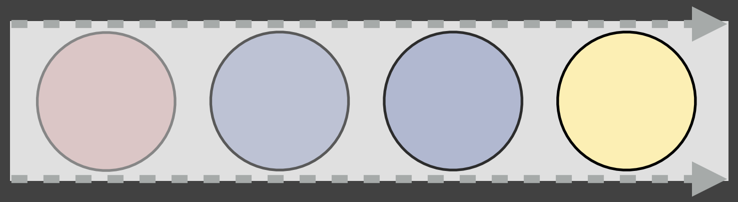
stream of `mousemove` events that occur at least 3ms, and at most 5ms, apart



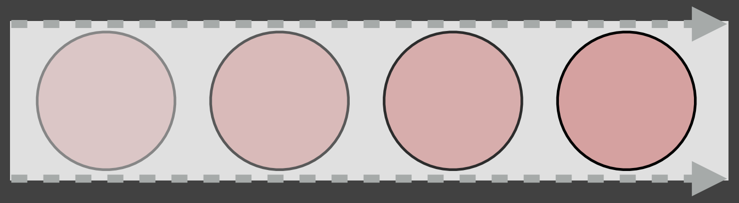
```
mousedown
```



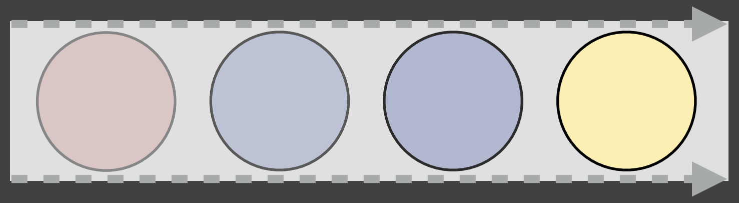
```
[mousedown, mouseup] >
    mousemove
```

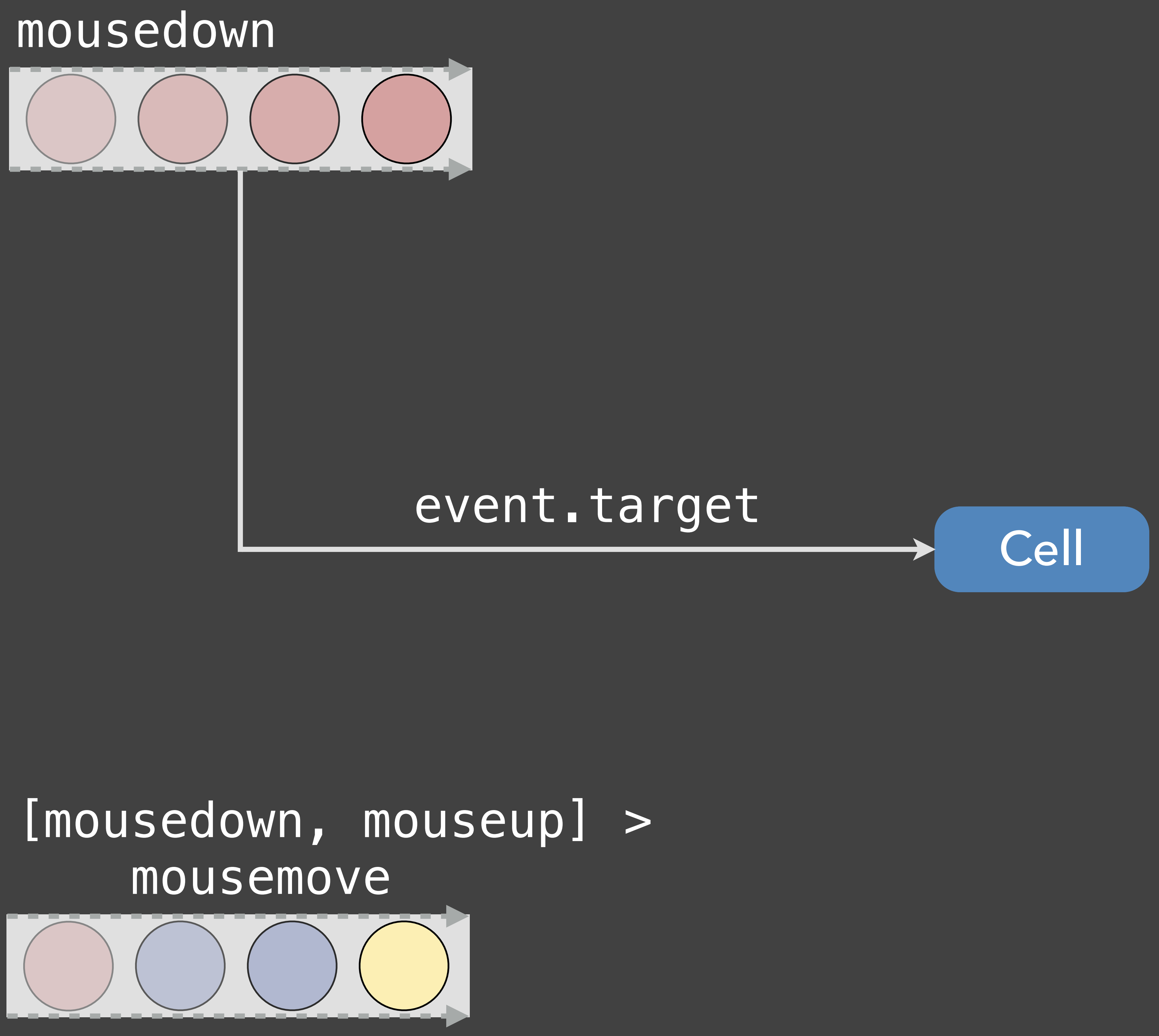


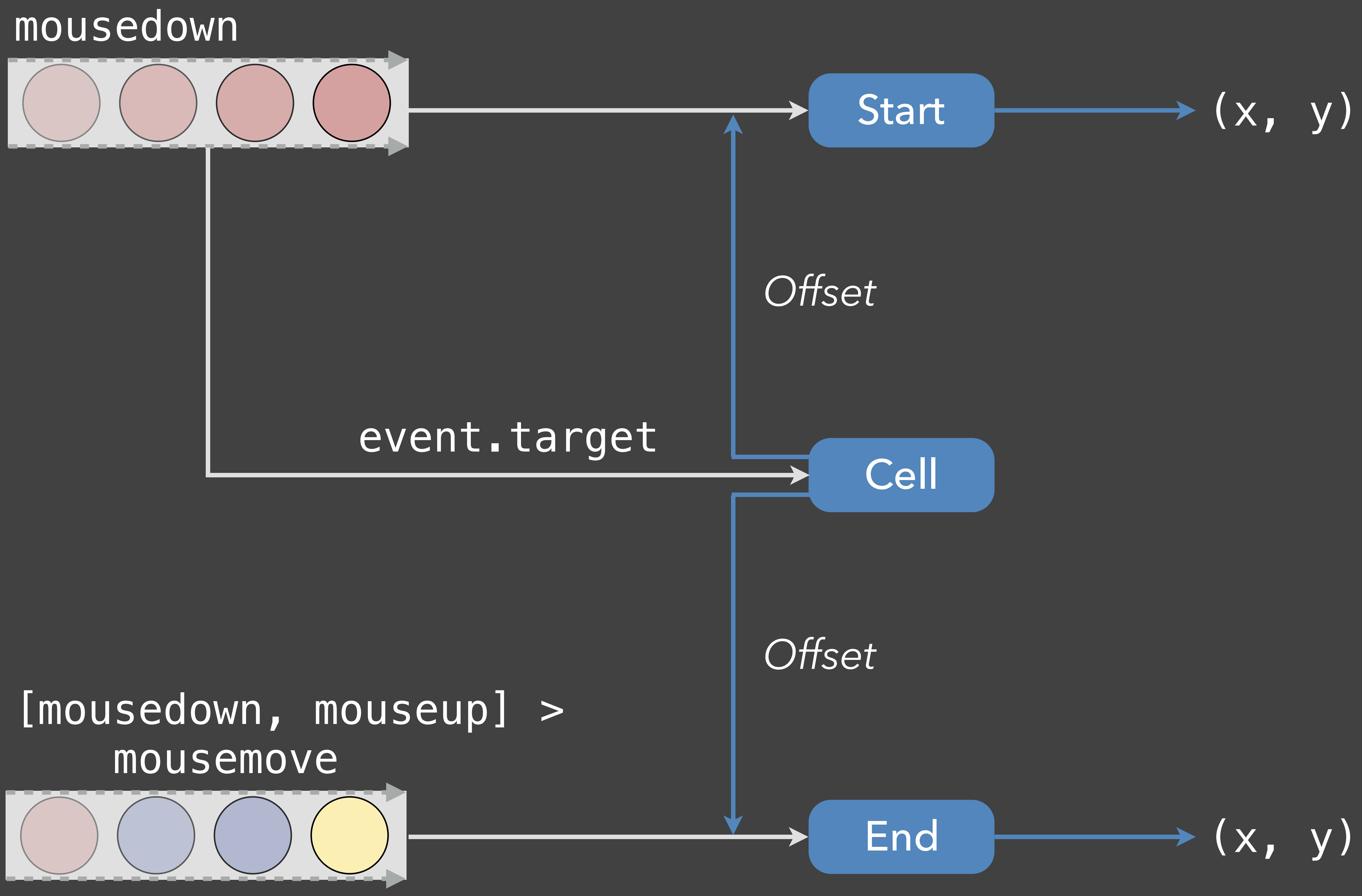
`mousedown`

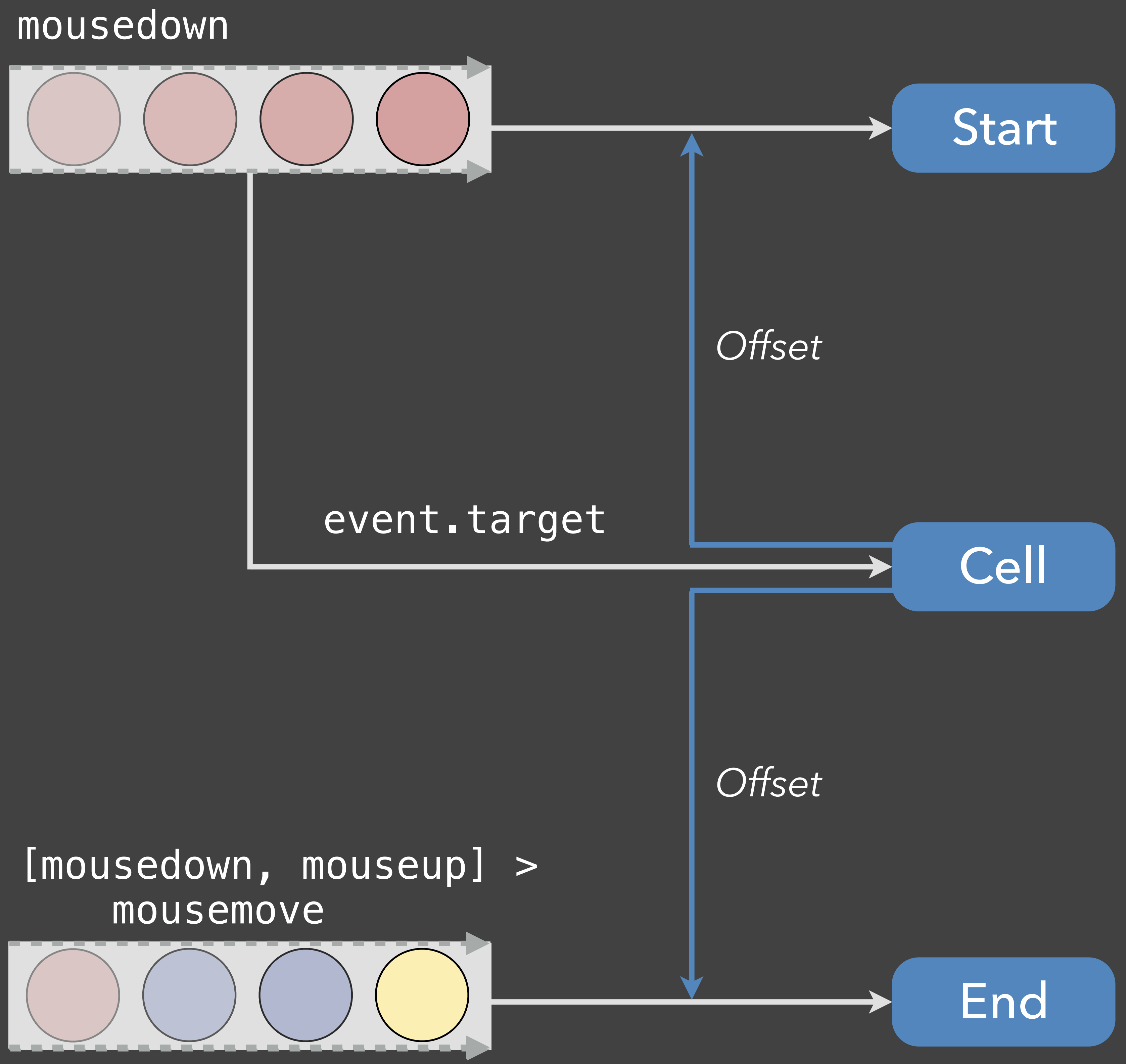


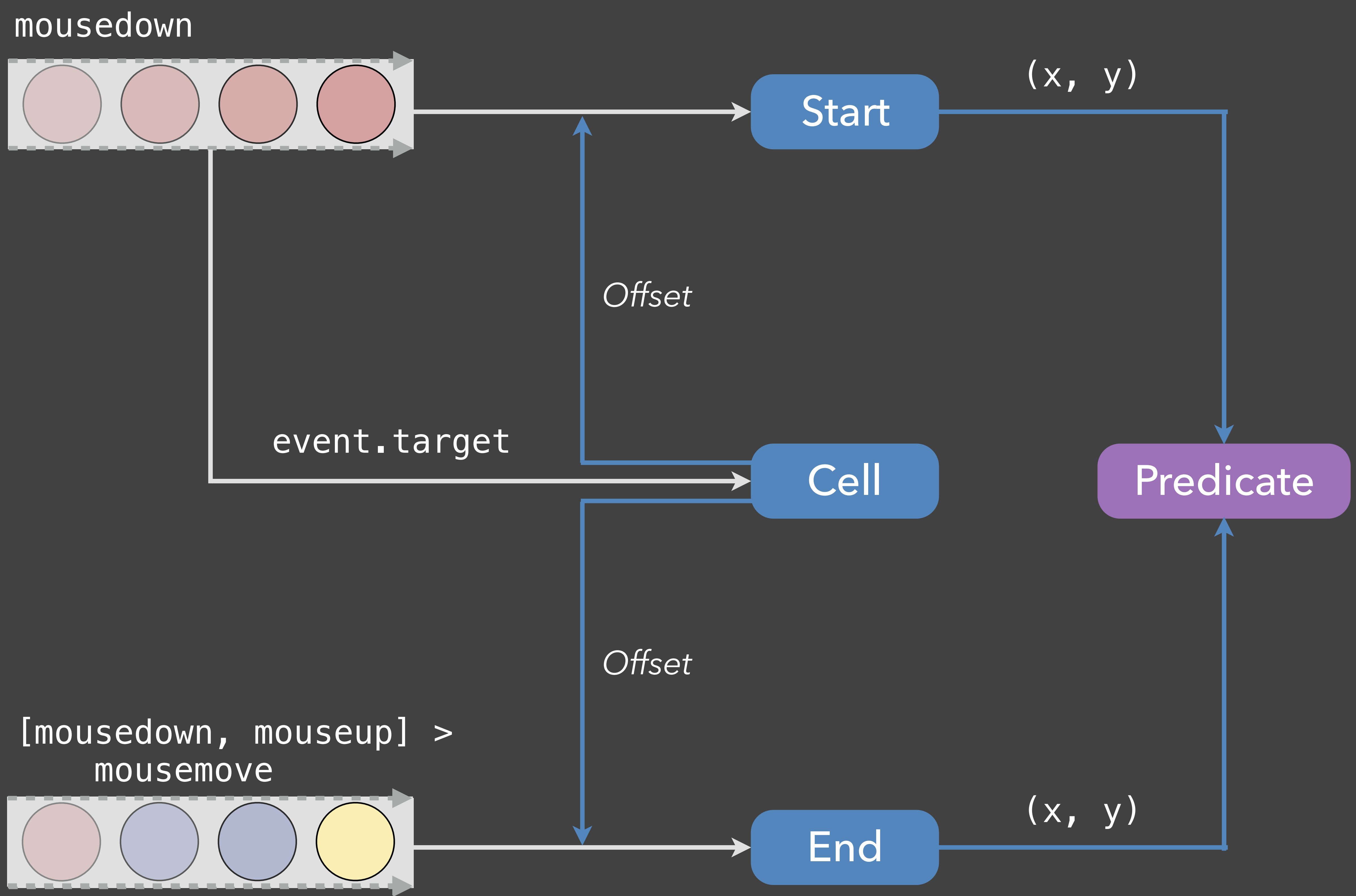
```
[mousedown, mouseup] >
    mousemove
```

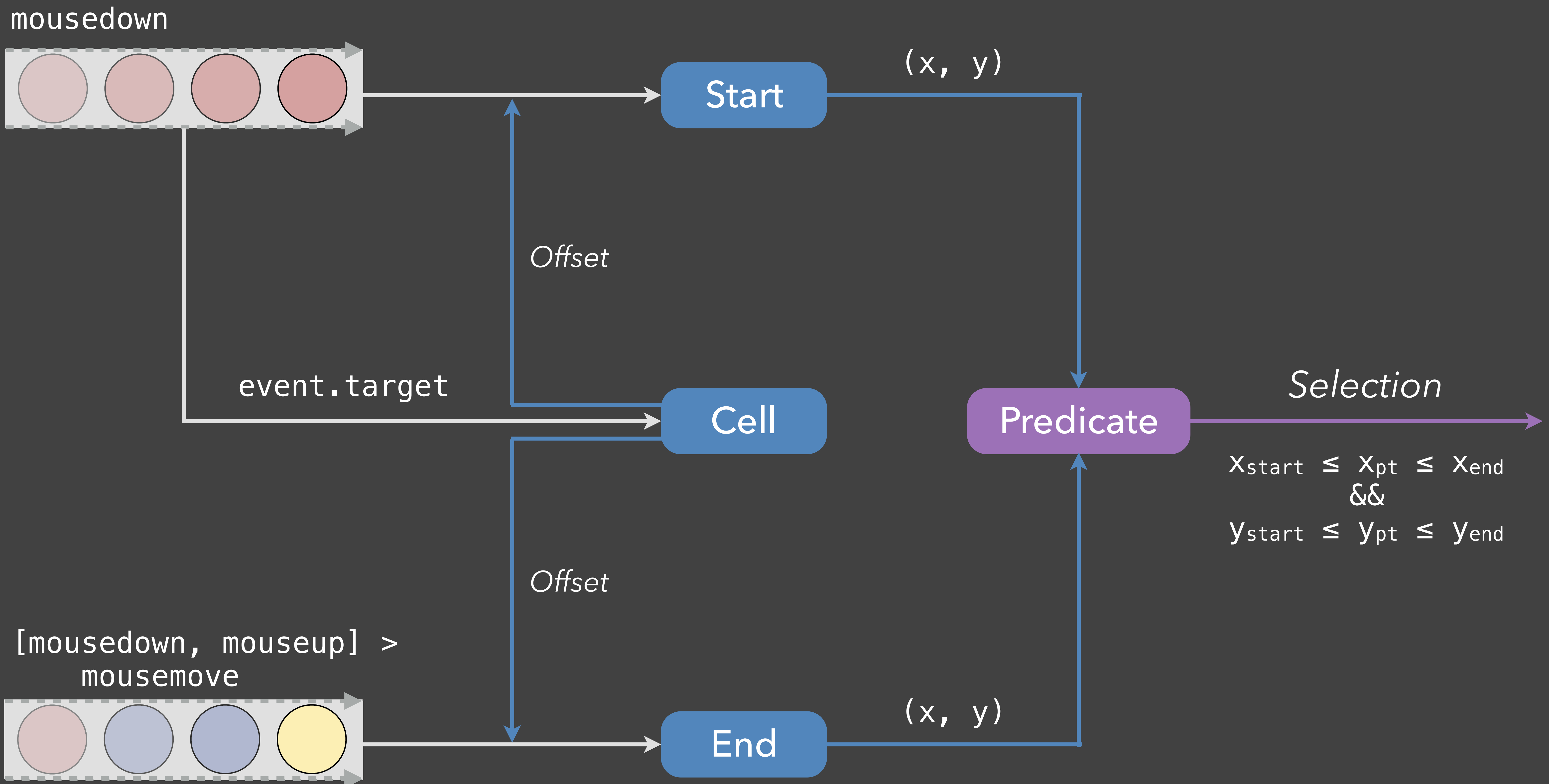



mousedown
event.target
Cell
[mousedown, mouseup] >
mousemove

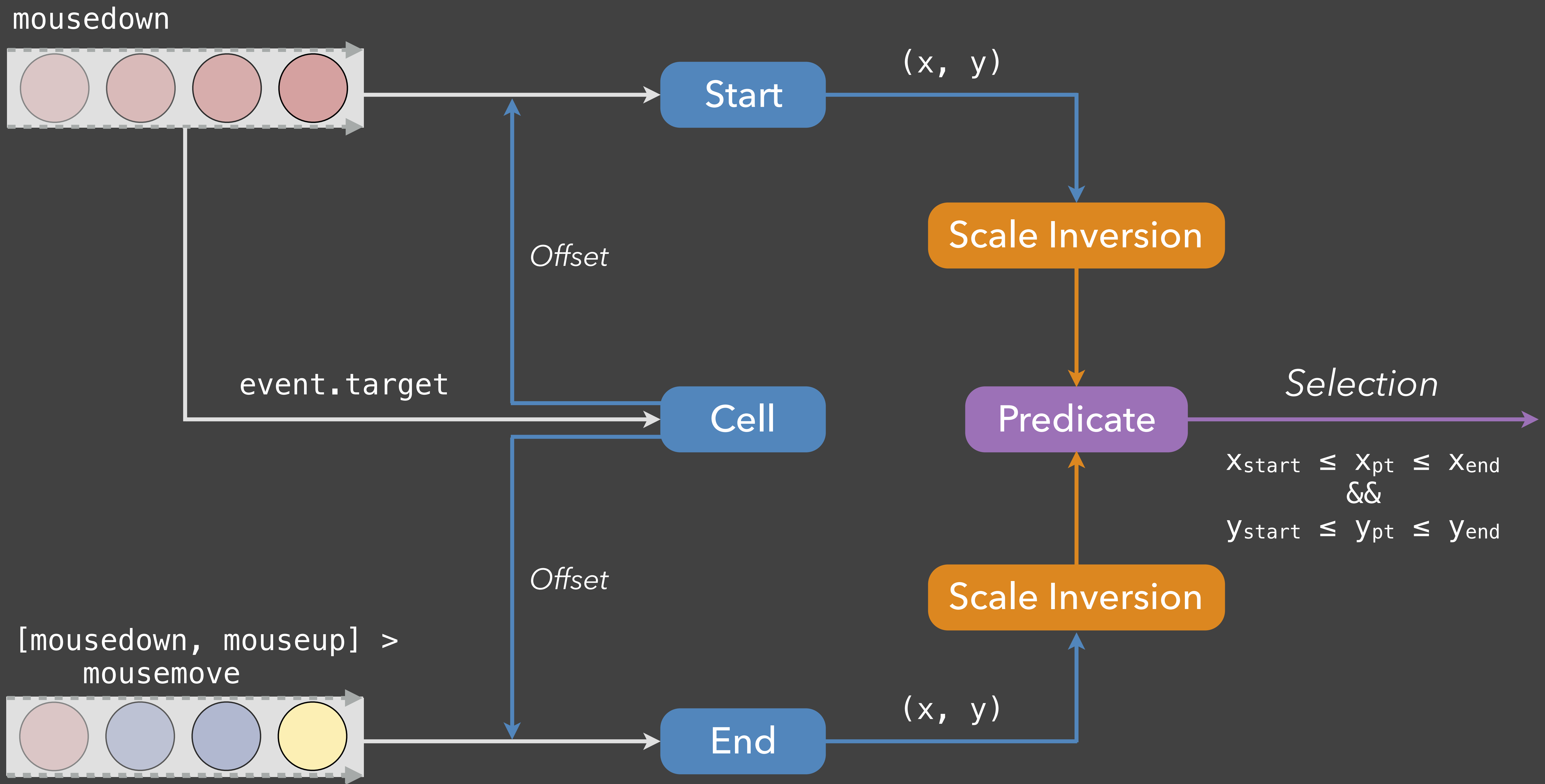

mousedown
Start
(x, y)
Offset
event.target
Cell
Offset
[mousedown, mouseup] > mousemove
End
(x, y)


mousedown
Start
Offset
event.target
Cell
Offset
[mousedown, mouseup] >
mousemove
End

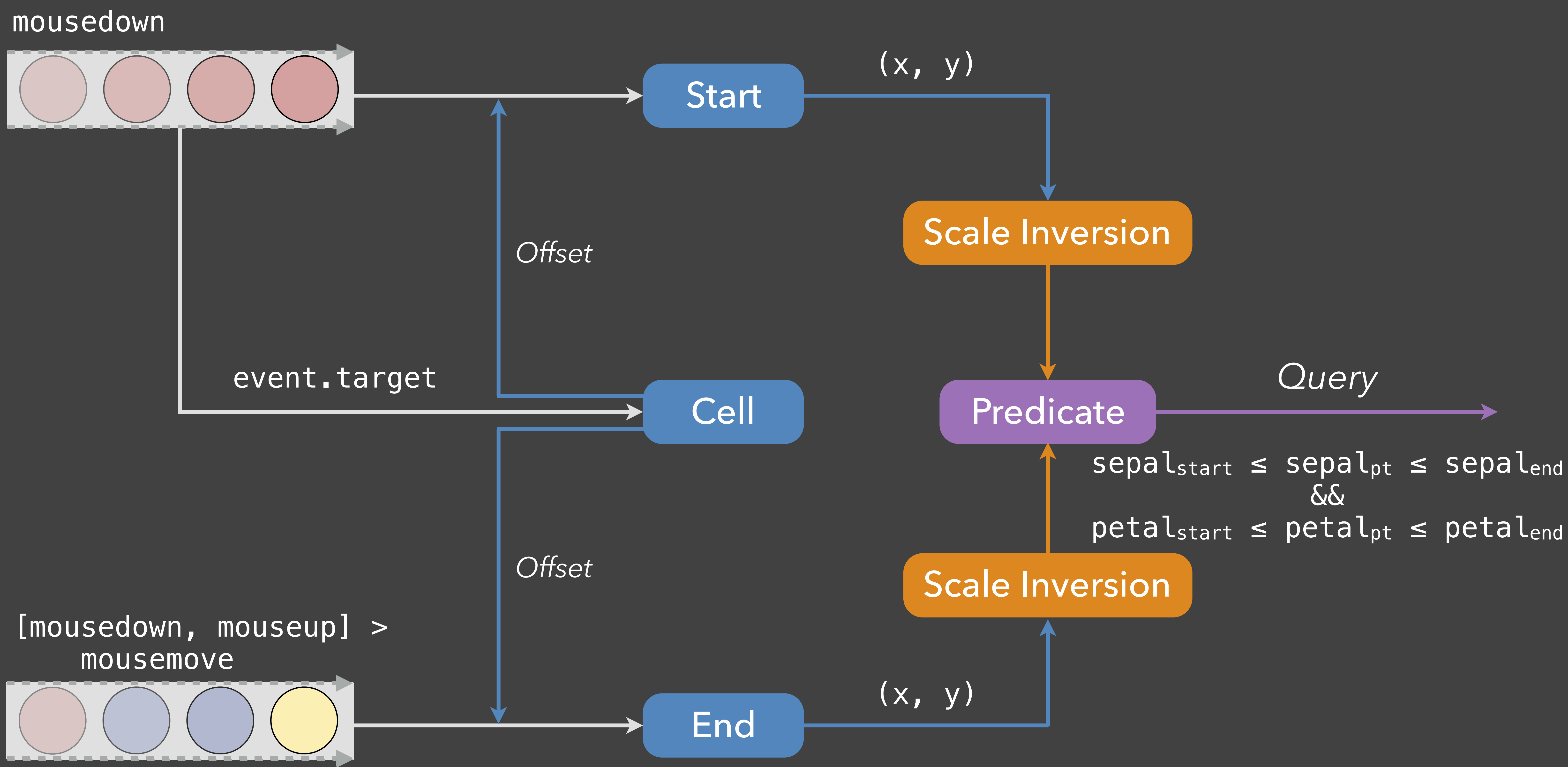

mousedown
Start
(x, y)
Offset
event.target
Cell
Predicate
Offset
[mousedown, mouseup] >
mousemove
End
(x, y)



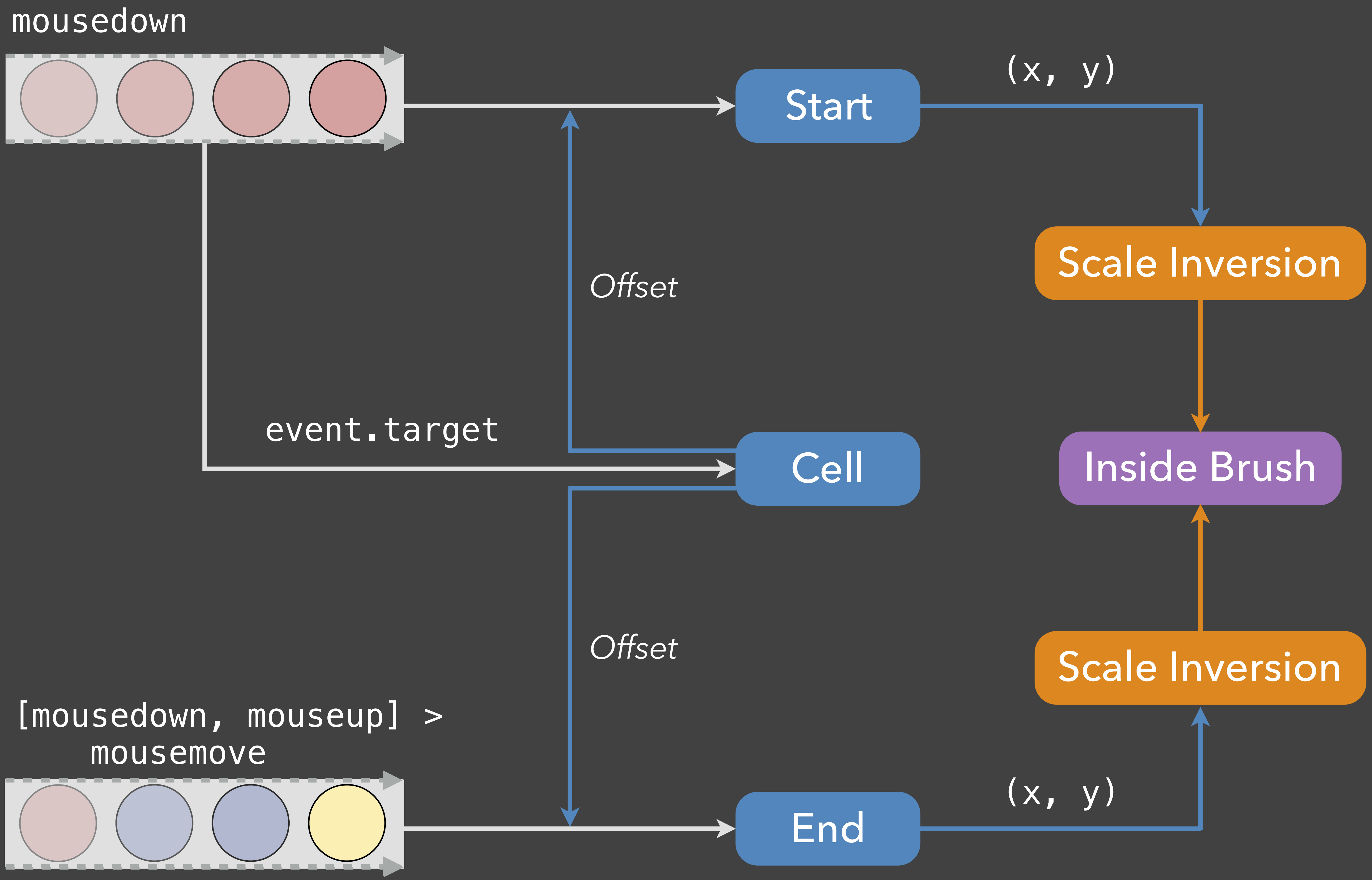
$$x_{start} \leq x_{pt} \leq x_{end}$$
&&
$$y_{start} \leq y_{pt} \leq y_{end}$$

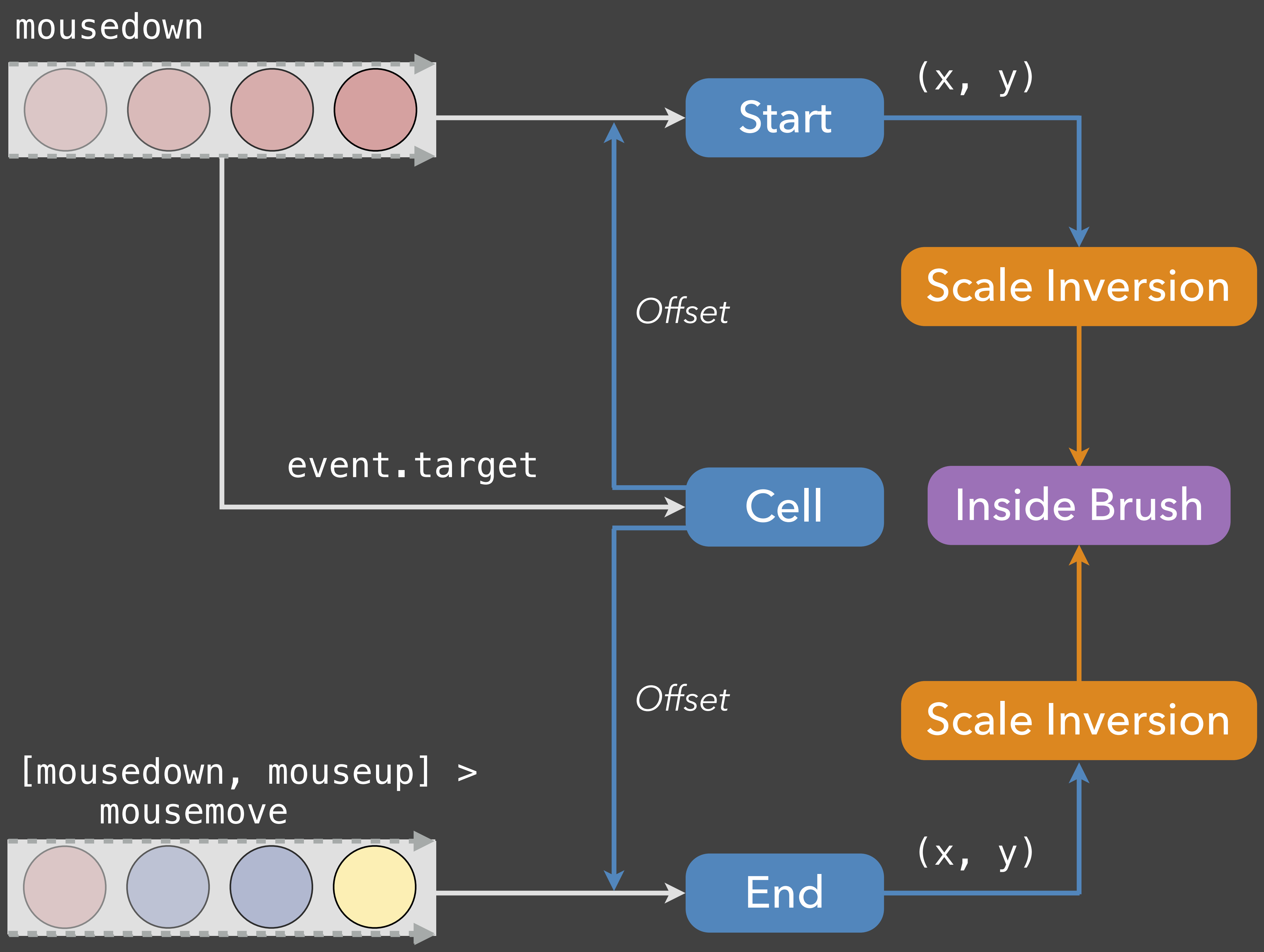


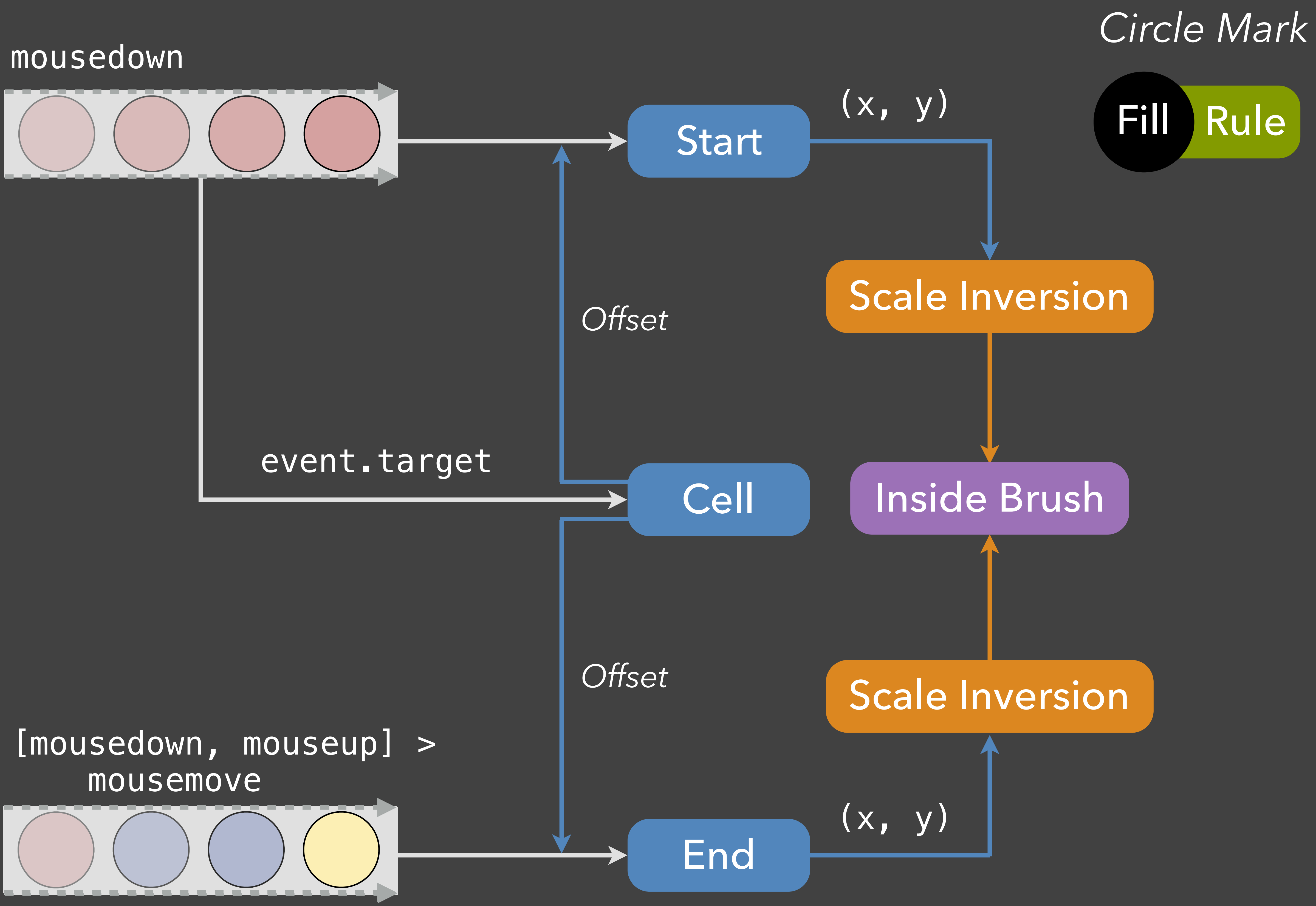
$$x_{start} \le x_{pt} \le x_{end}$$
$$\&\&$$
$$y_{start} \le y_{pt} \le y_{end}$$

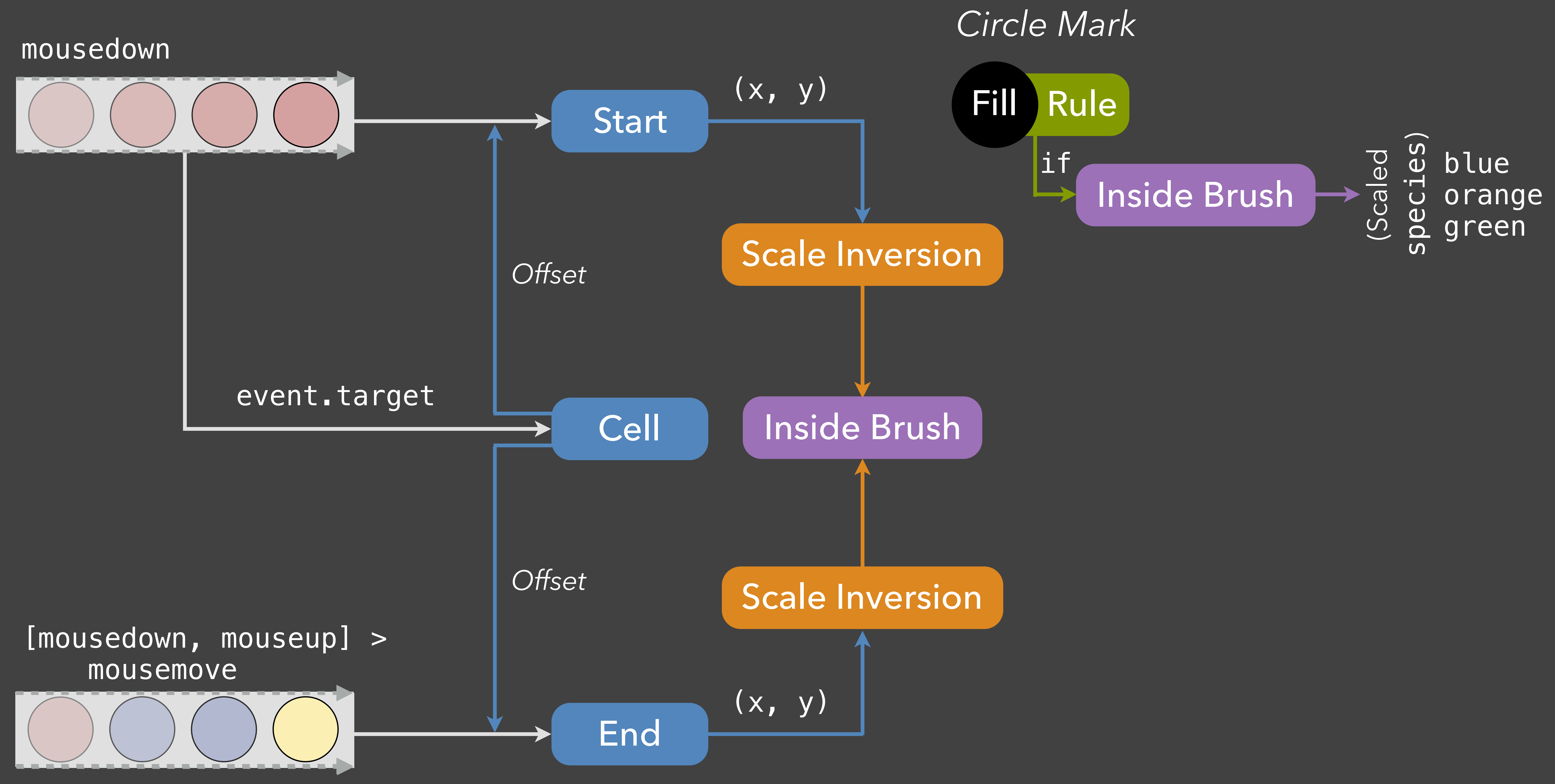


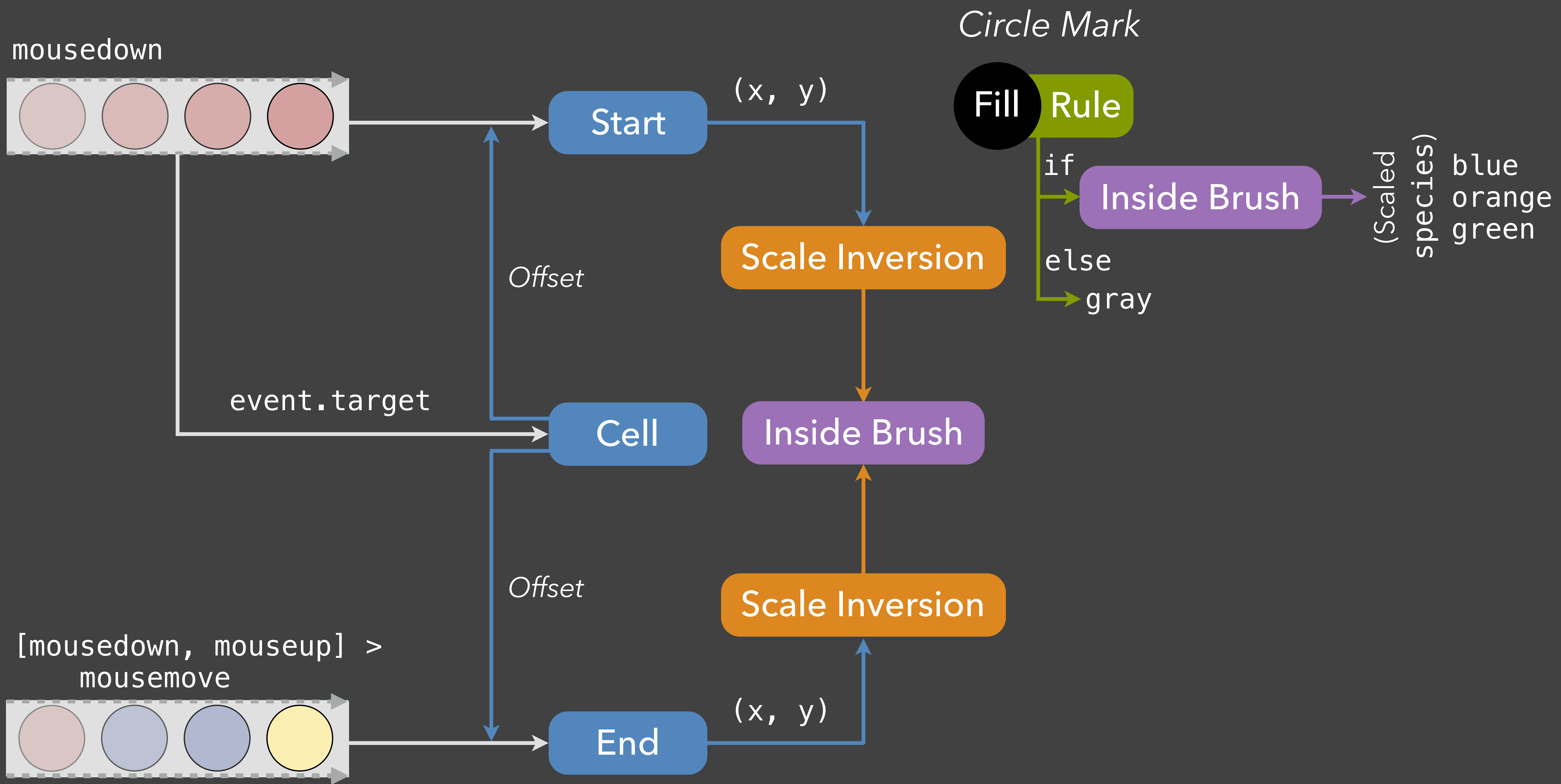
$$\text{sepal}_{start} \le \text{sepal}_{pt} \le \text{sepal}_{end}$$
$$\&\&$$
$$\text{petal}_{start} \le \text{petal}_{pt} \le \text{petal}_{end}$$


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

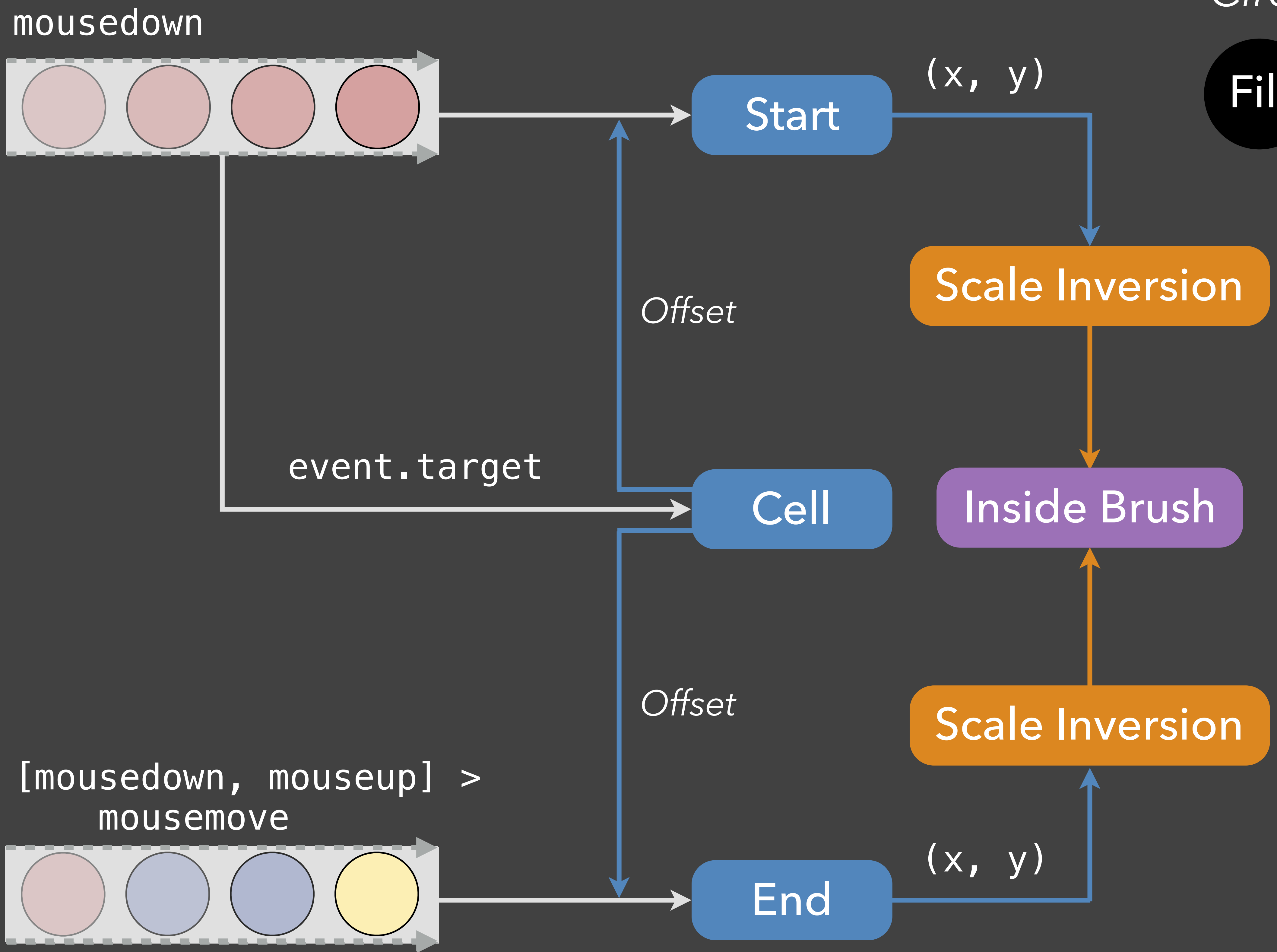

mousedown
Start
(x, y)
Scale Inversion
Offset
event.target
Cell
Inside Brush
Offset
Scale Inversion
[mousedown, mouseup] > mousemove
End
(x, y)

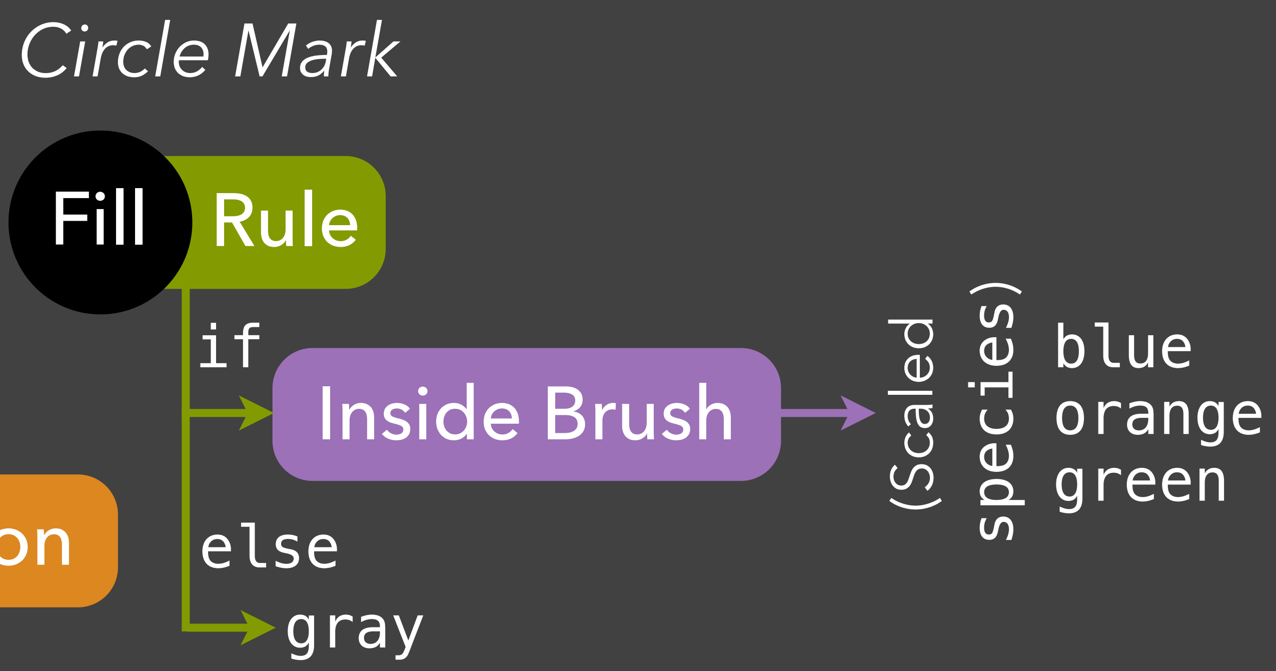

Circle Mark
Fill
Rule
mousedown
Start
(x, y)
Scale Inversion
Offset
event.target
Cell
Inside Brush
Offset
Scale Inversion
[mousedown, mouseup] > mousemove
End
(x, y)

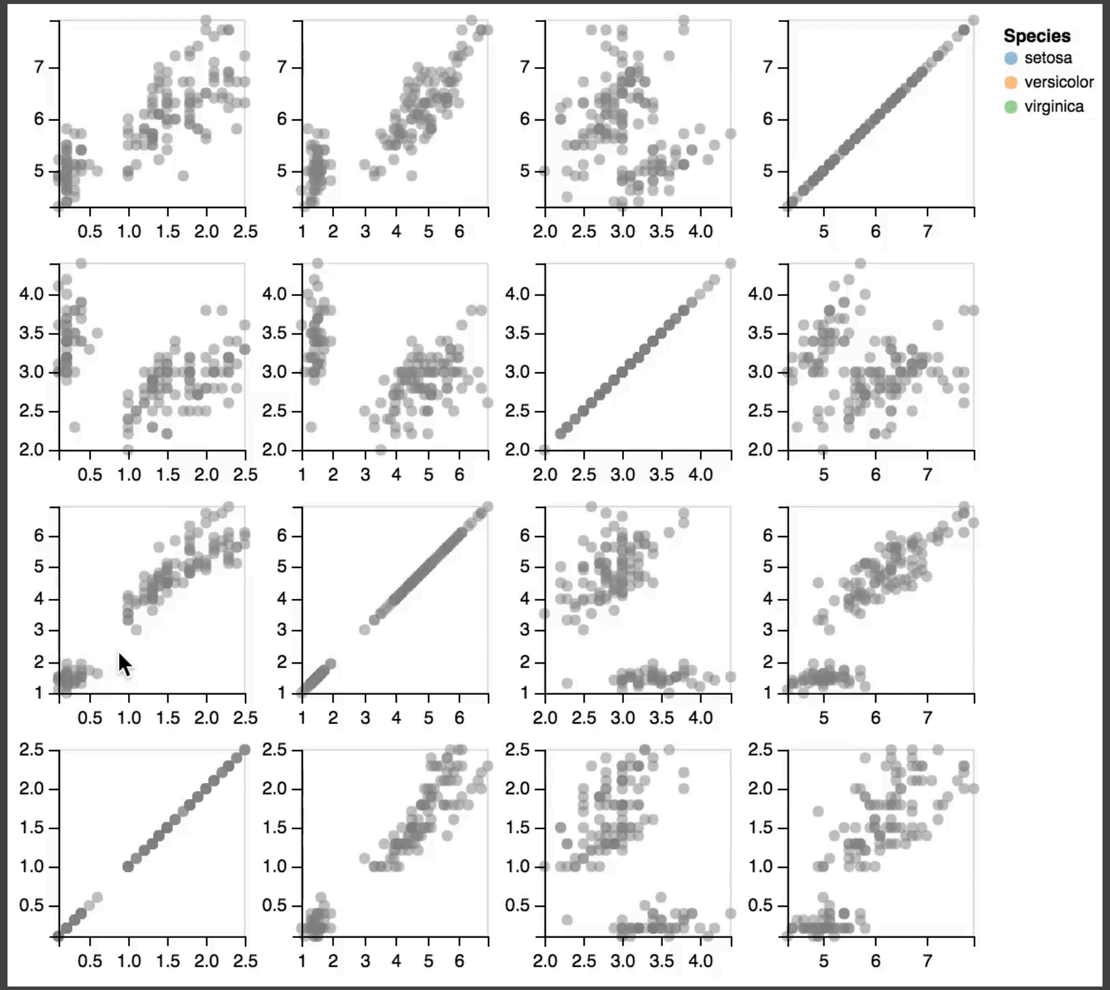

mousedown
Start
(x, y)
Circle Mark
Fill
Rule
if
Inside Brush
(Scaled species)
blue
orange
green
Scale Inversion
Offset
event.target
Cell
Inside Brush
Offset
Scale Inversion
[mousedown, mouseup] > mousemove
End
(x, y)



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


mousedown
Start
(x, y)
Scale Inversion
Offset
event.target
Cell
Inside Brush
Offset
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

Demo

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

Reactive Vega

A Streaming Dataflow Architecture for
Declarative Interactive Visualization

Arvind Satyanarayan @arvindsatya1
Stanford University

Ryan Russell
Jane Hoffswell
Jeffrey Heer @jeffrey_heer
University of Washington

Reactive Vega

A Streaming Dataflow Architecture for Declarative Interactive Visualization

Arvind Satyanarayan @arvindsatya1
Stanford University

Ryan Russell
Jane Hoffswell
Jeffrey Heer @jeffrey_heer
University of Washington

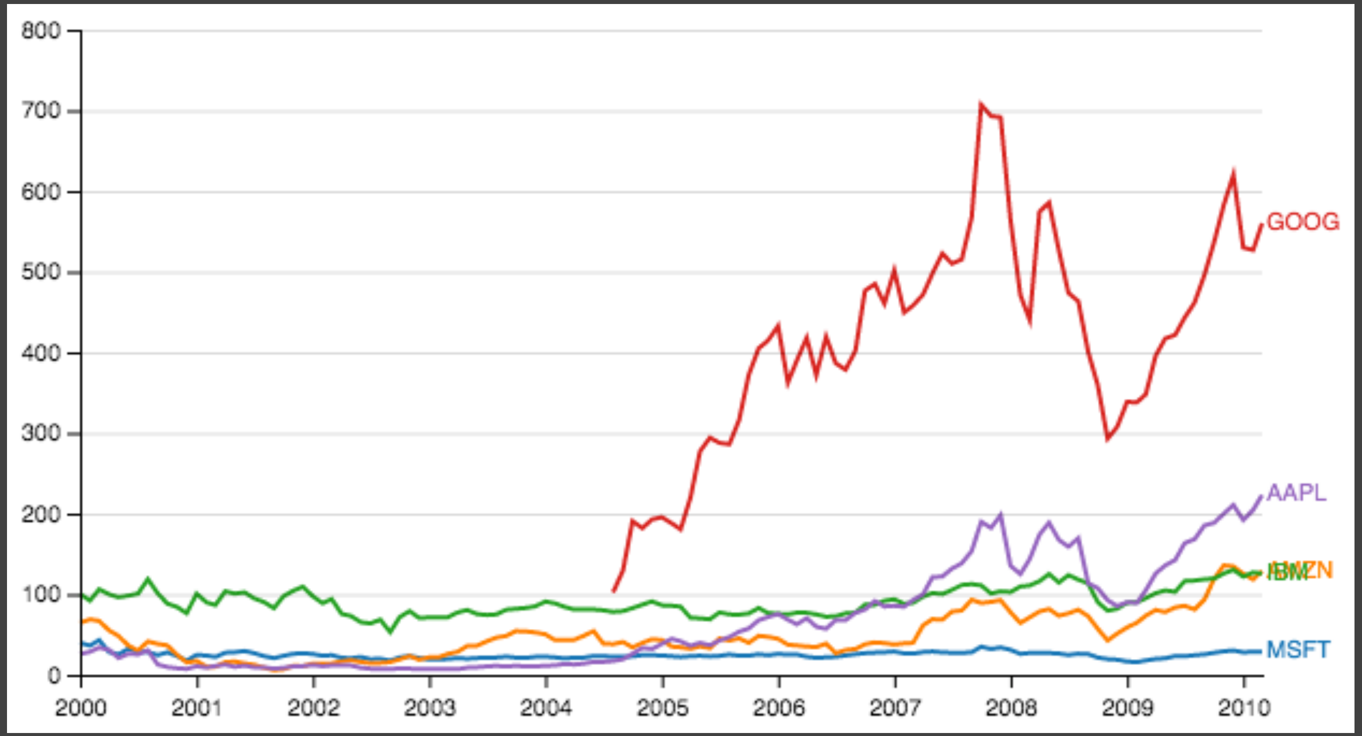
```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```

Data + Transforms

Scales

Guides

Marks



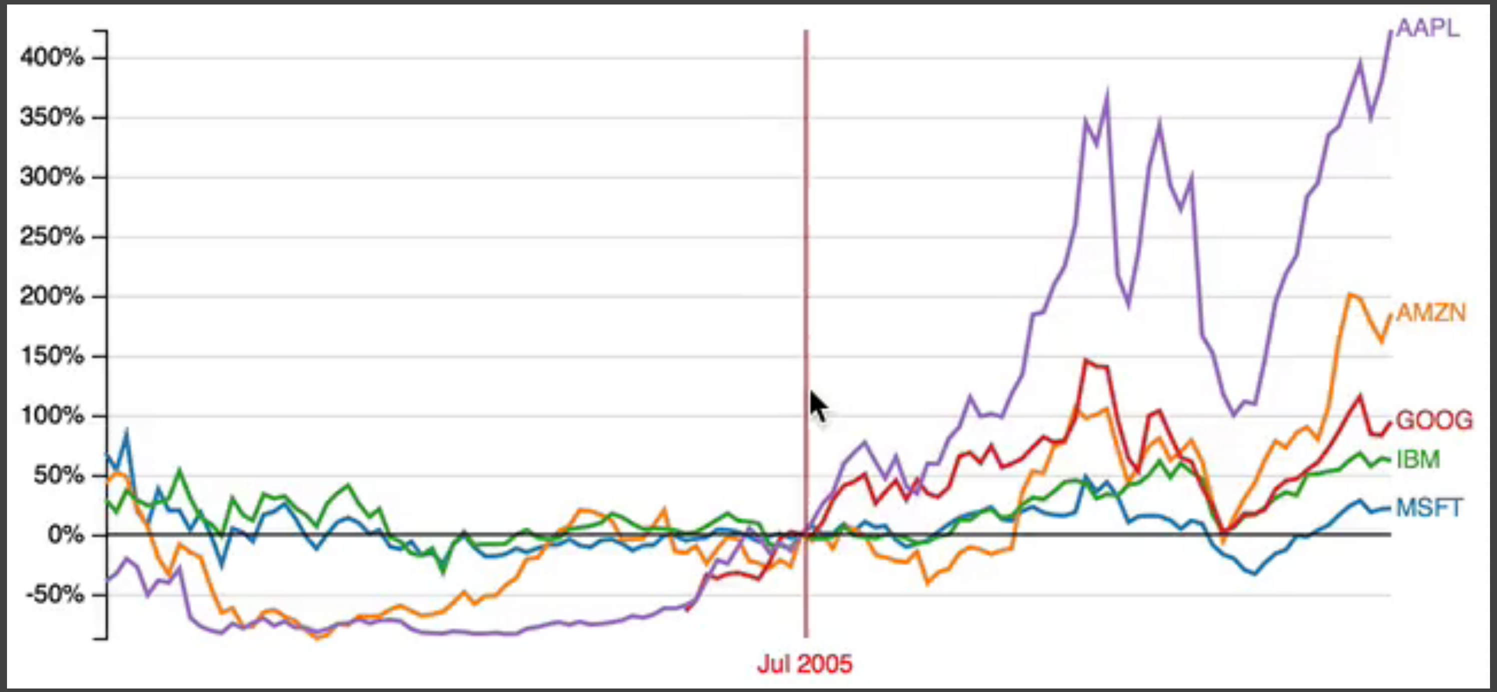
```
{
  "width": 650, "height": 300,
  "data": [
    {"name": "stocks", "url": "data/stocks.json"}
  ],
  "scales": [
    {
      "name": "sx", "type": "ordinal",
      "domain": {"data": "stocks", "field": "date"}
      "range": "width"
    }, ...
  ],
  "axes": [
    {"type": "x", "scale": "sx"}, ...
  ],
  "marks": [{
    "type": "group",
    "from": {
      "data": "stocks",
      "transform": [
        {"type": "facet", "groupby": ["symbol"]}
      ]
    },
    "marks": [{
      "type": "line",
      "properties": { "enter": {
        "x": {"scale": "sx", "field": "date"},
        "y": {"scale": "sy", "field": "price"},
        "stroke": {"scale": "sc", "field": "symbol"}
      }}
    }, {
      "type": "text",
      ...
    }]
```

Data +
Transforms

Scales

Guides

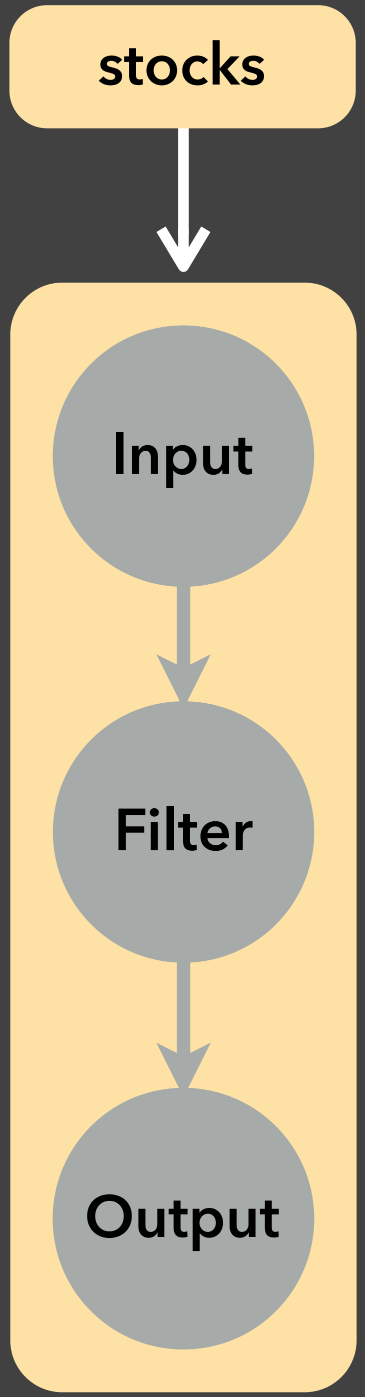
Marks



```
{
  "data": [
    {...},
    {
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```

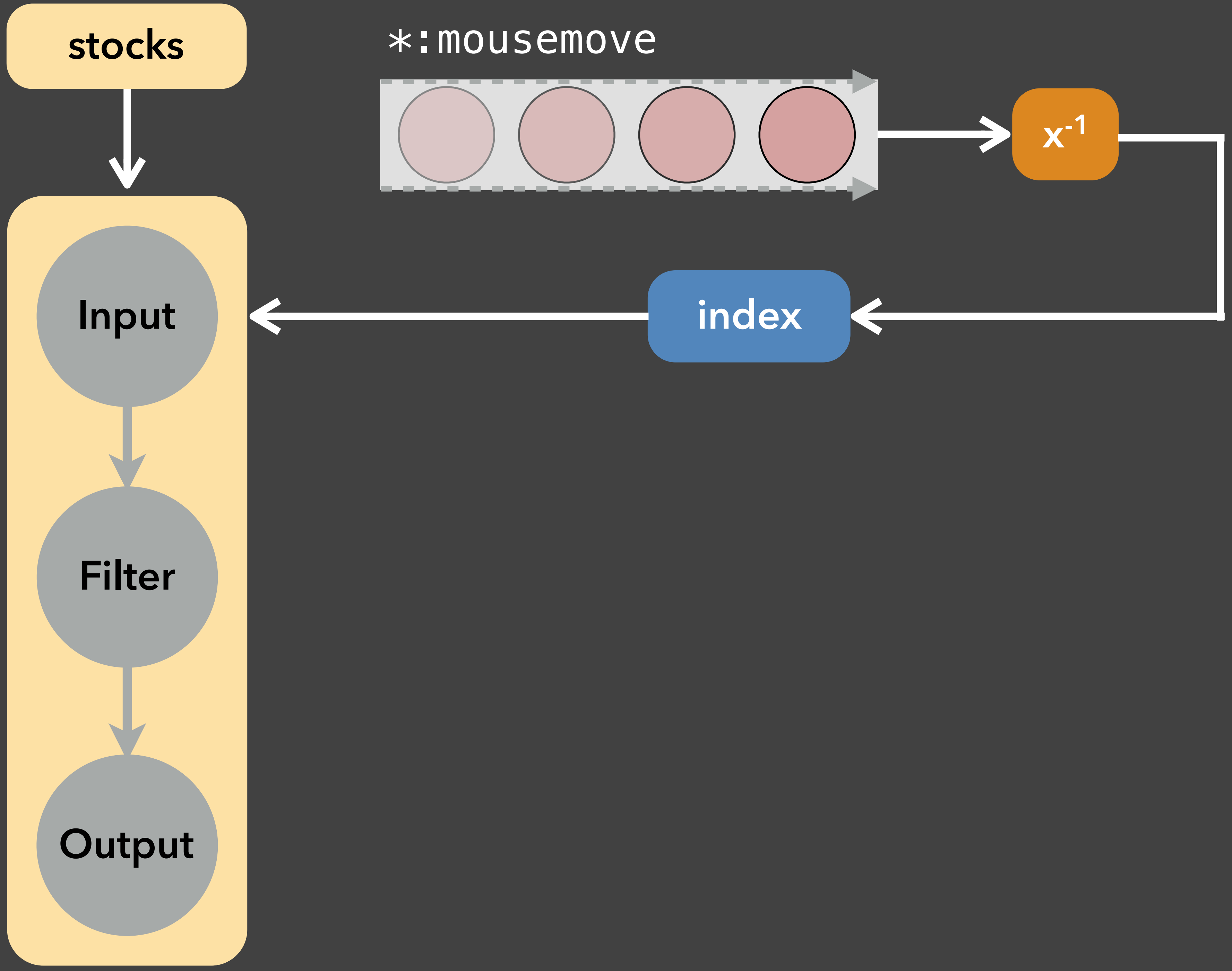
Compile Time

```
{
  "data": [
    {...},
    {
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



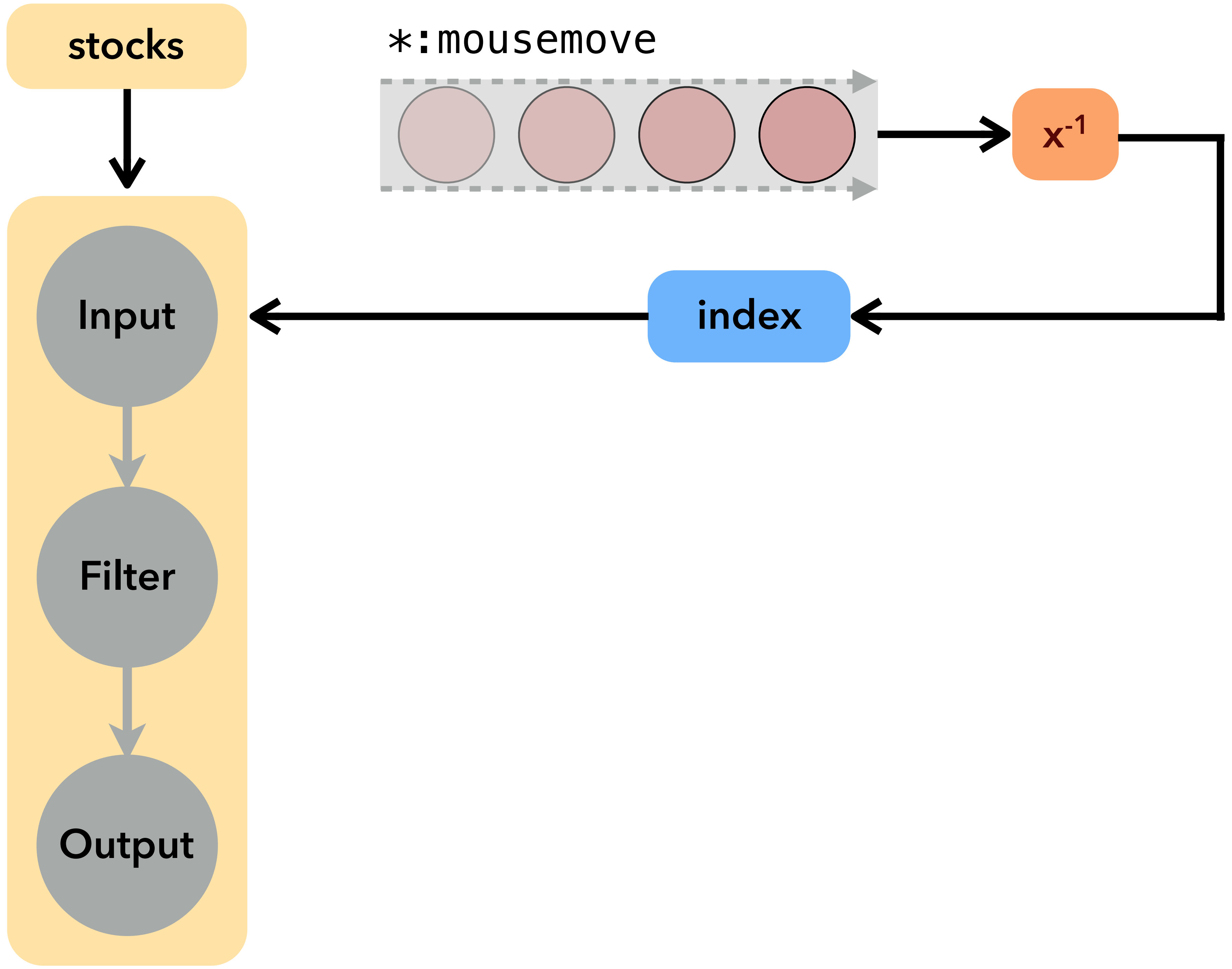
Compile Time

```
{
  "data": [
    {...},
    {
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



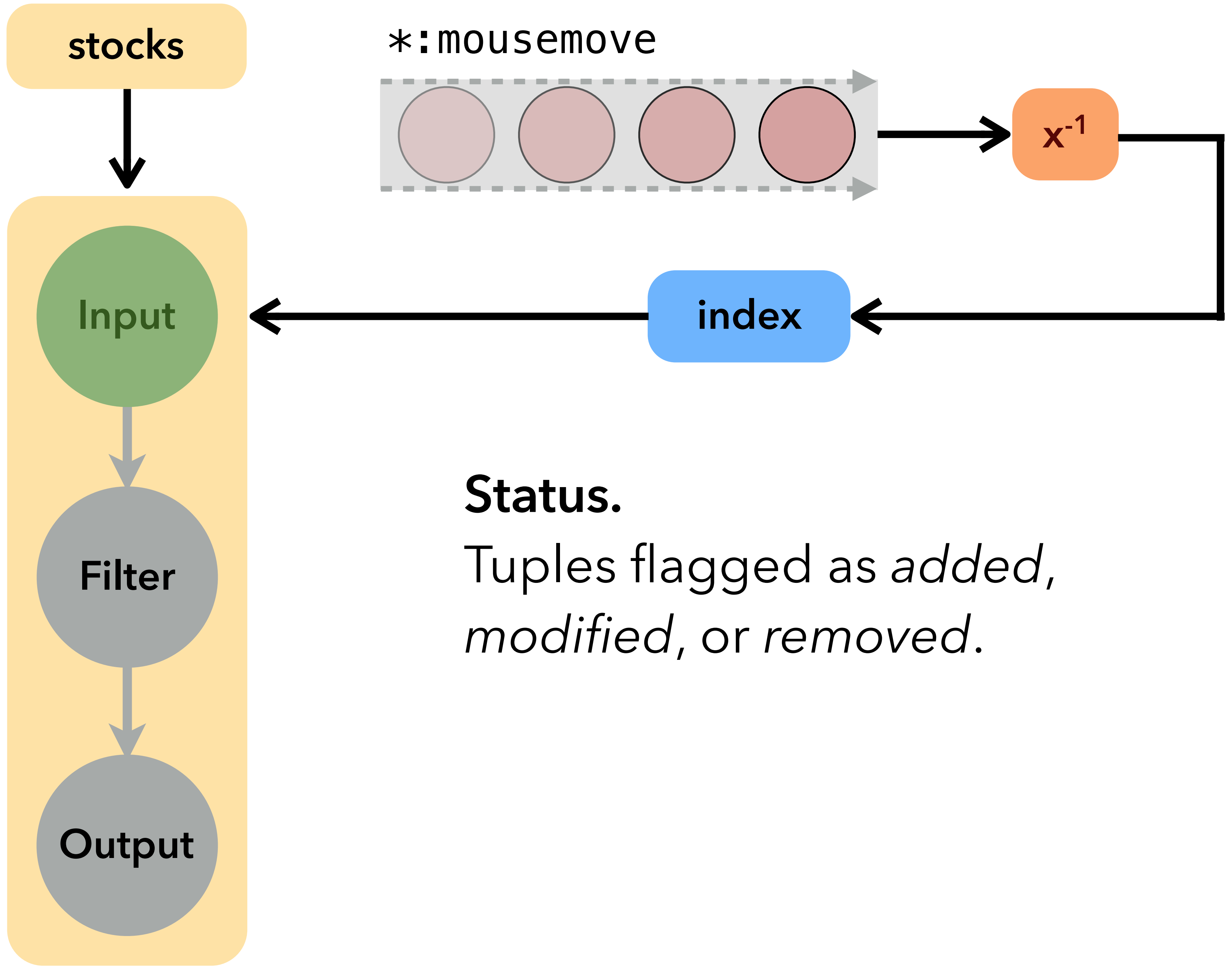
Run Time

```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



Run Time

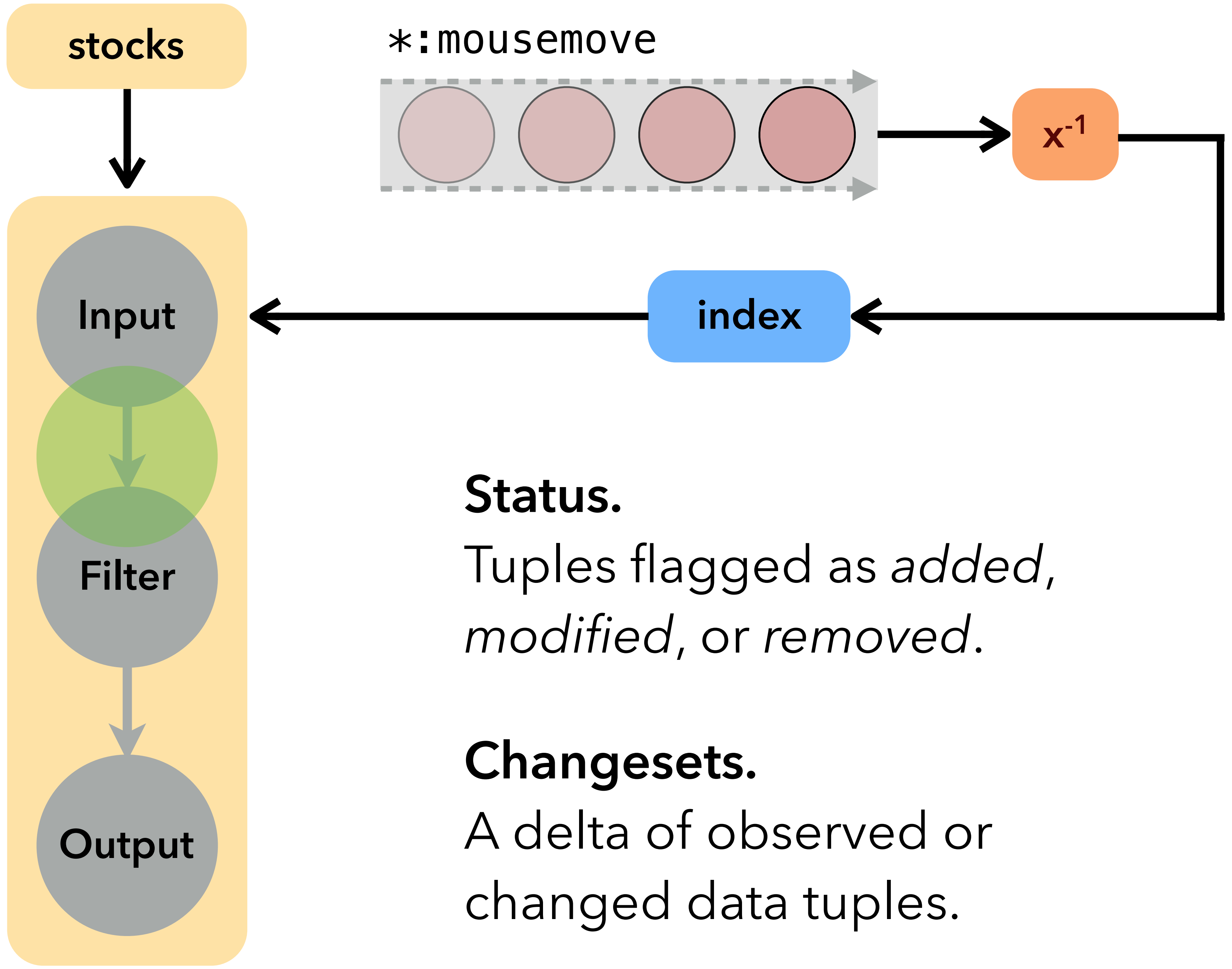
```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



Status.
Tuples flagged as *added*, *modified*, or *removed*.

Run Time

```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```

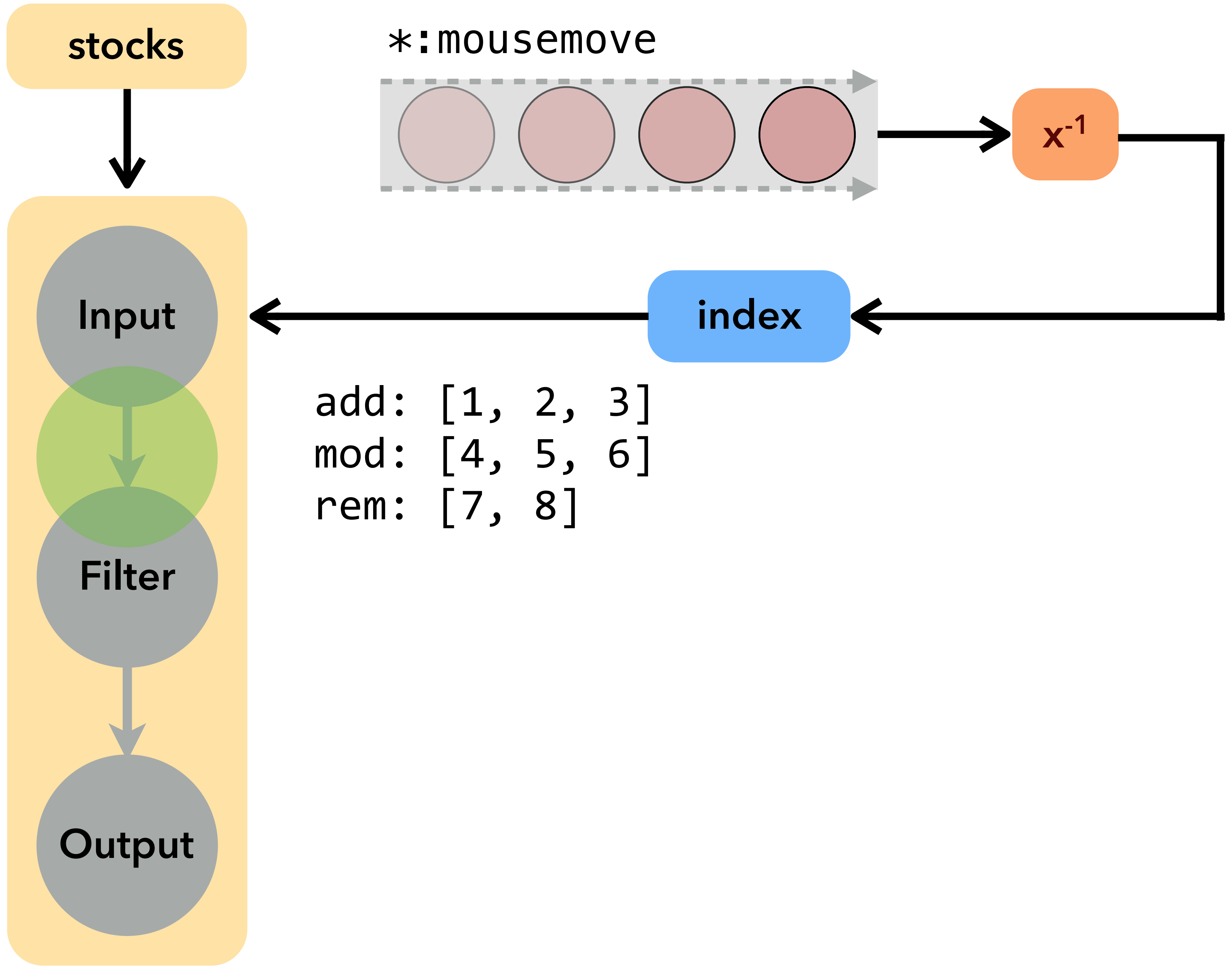


Status.
Tuples flagged as *added*, *modified*, or *removed*.

Changesets.
A delta of observed or changed data tuples.

Run Time

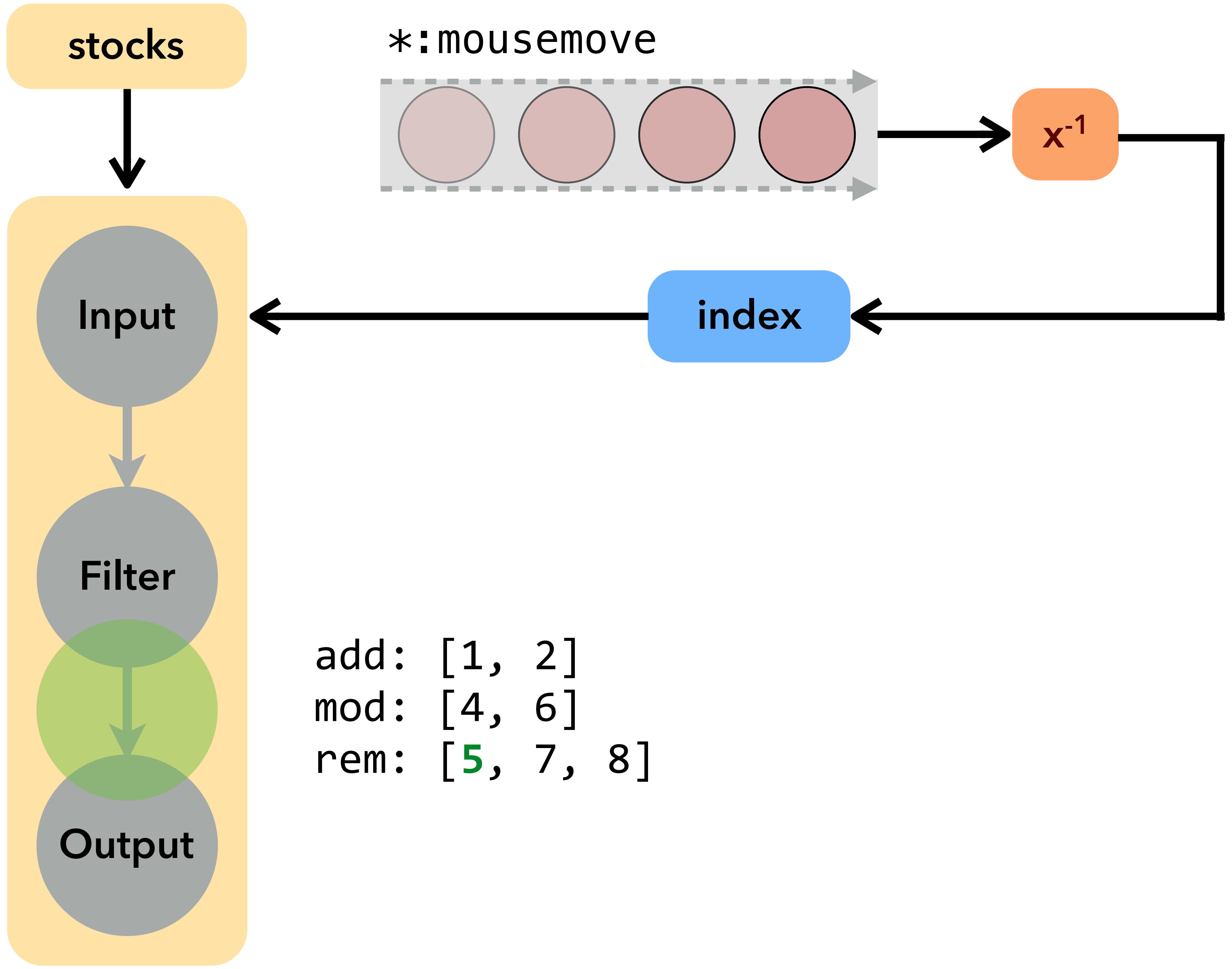
```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



```
add: [1, 2, 3]
mod: [4, 5, 6]
rem: [7, 8]
```

Run Time

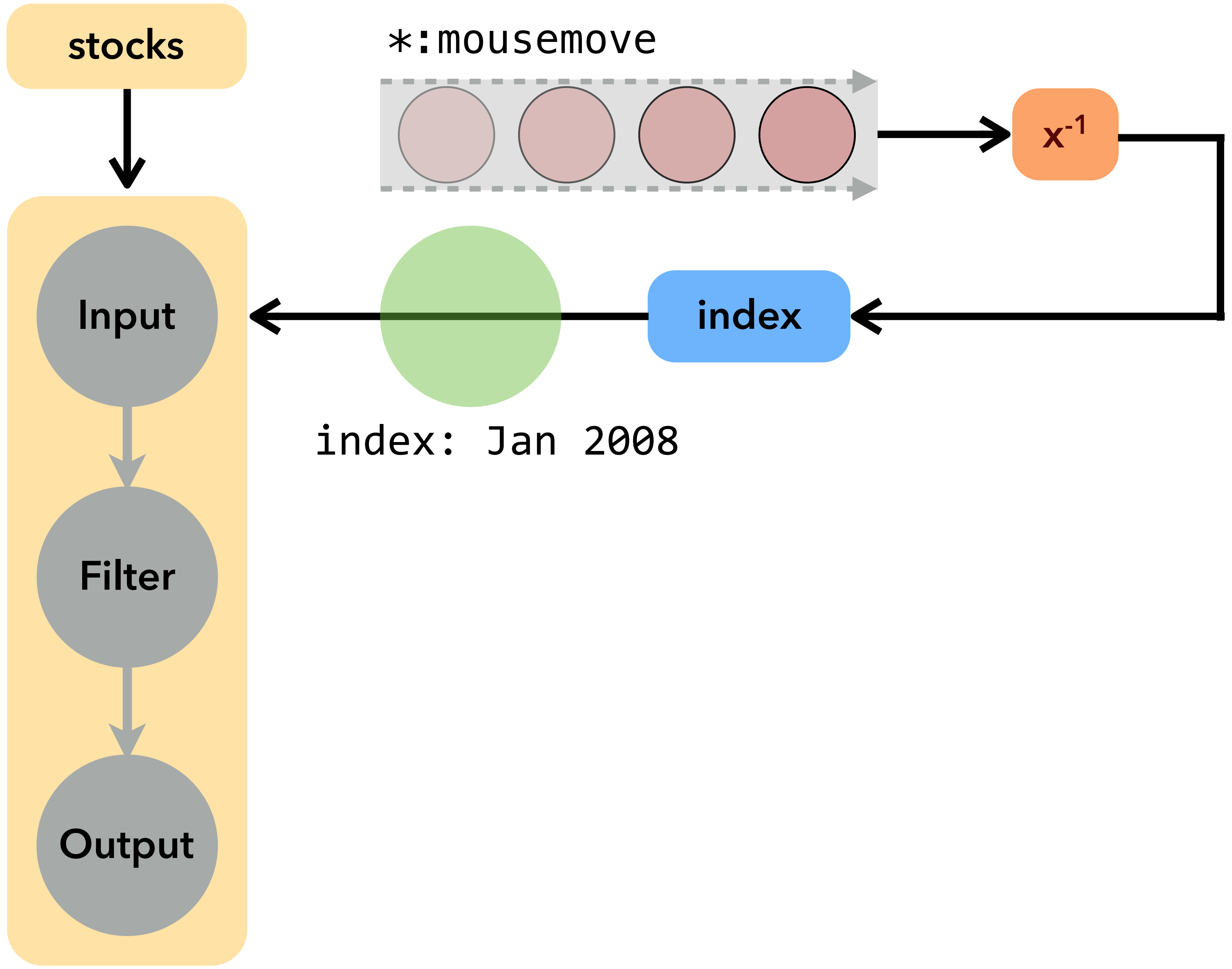
```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



```
add: [1, 2]
mod: [4, 6]
rem: [5, 7, 8]
```

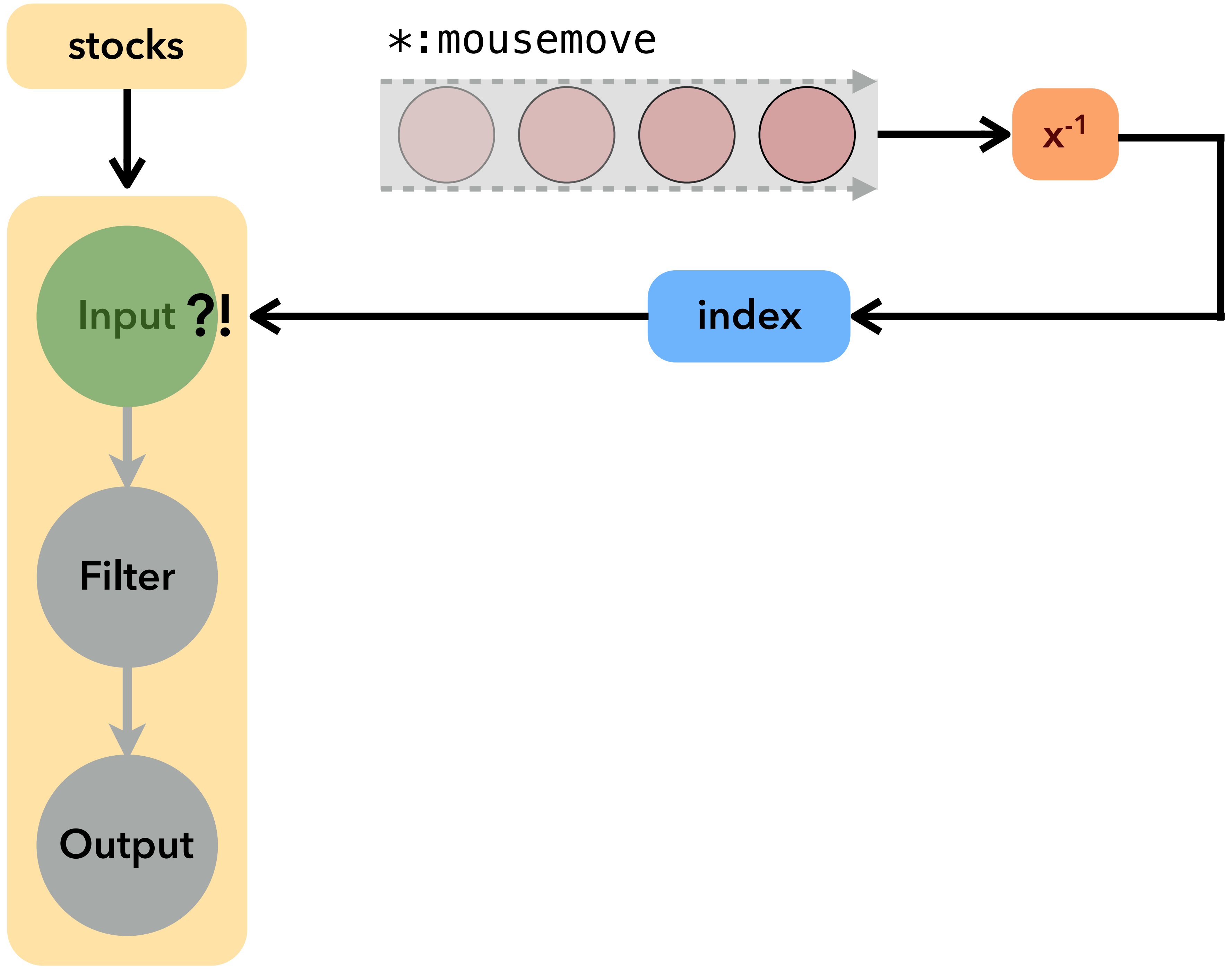
Run Time

```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



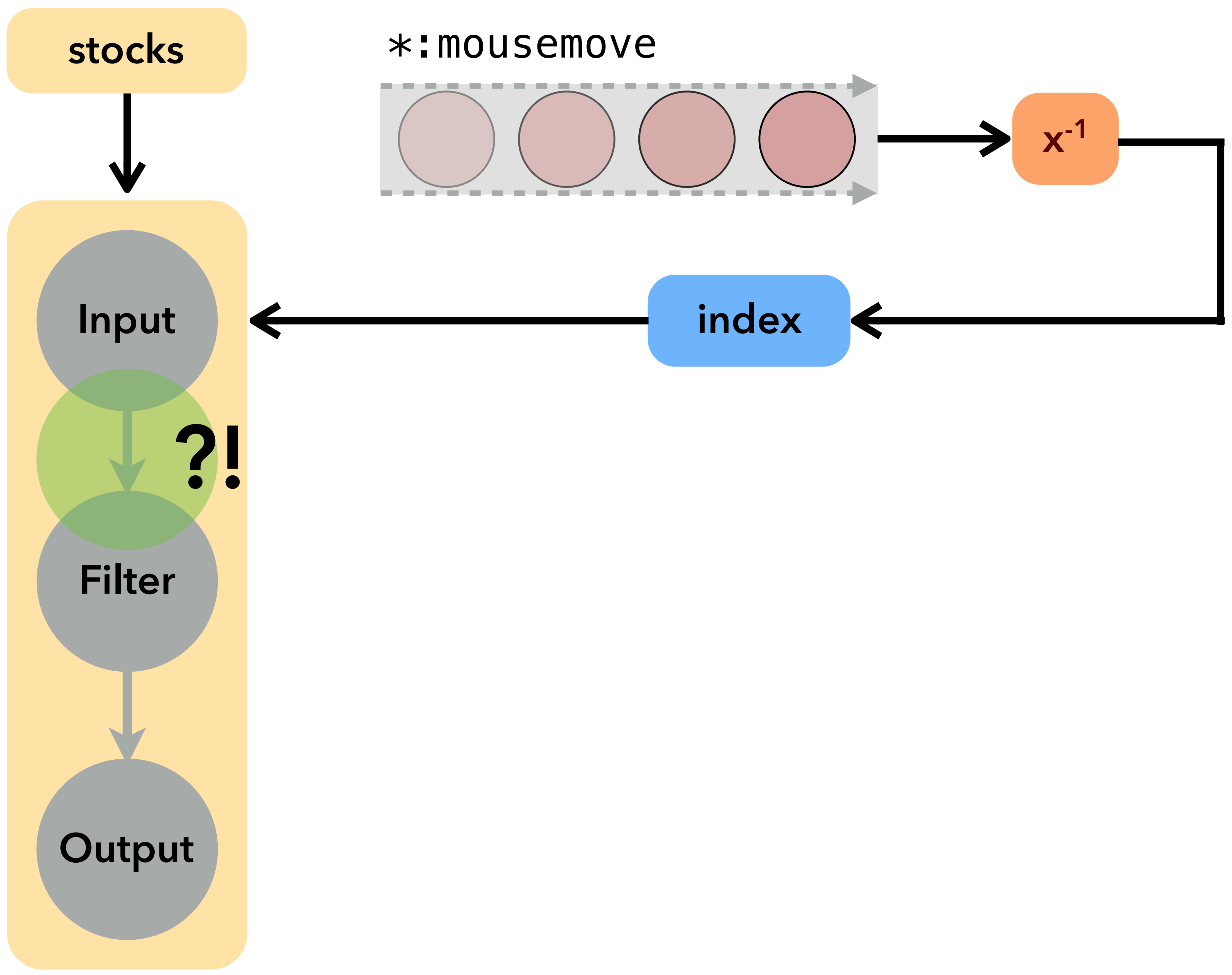
Run Time

```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



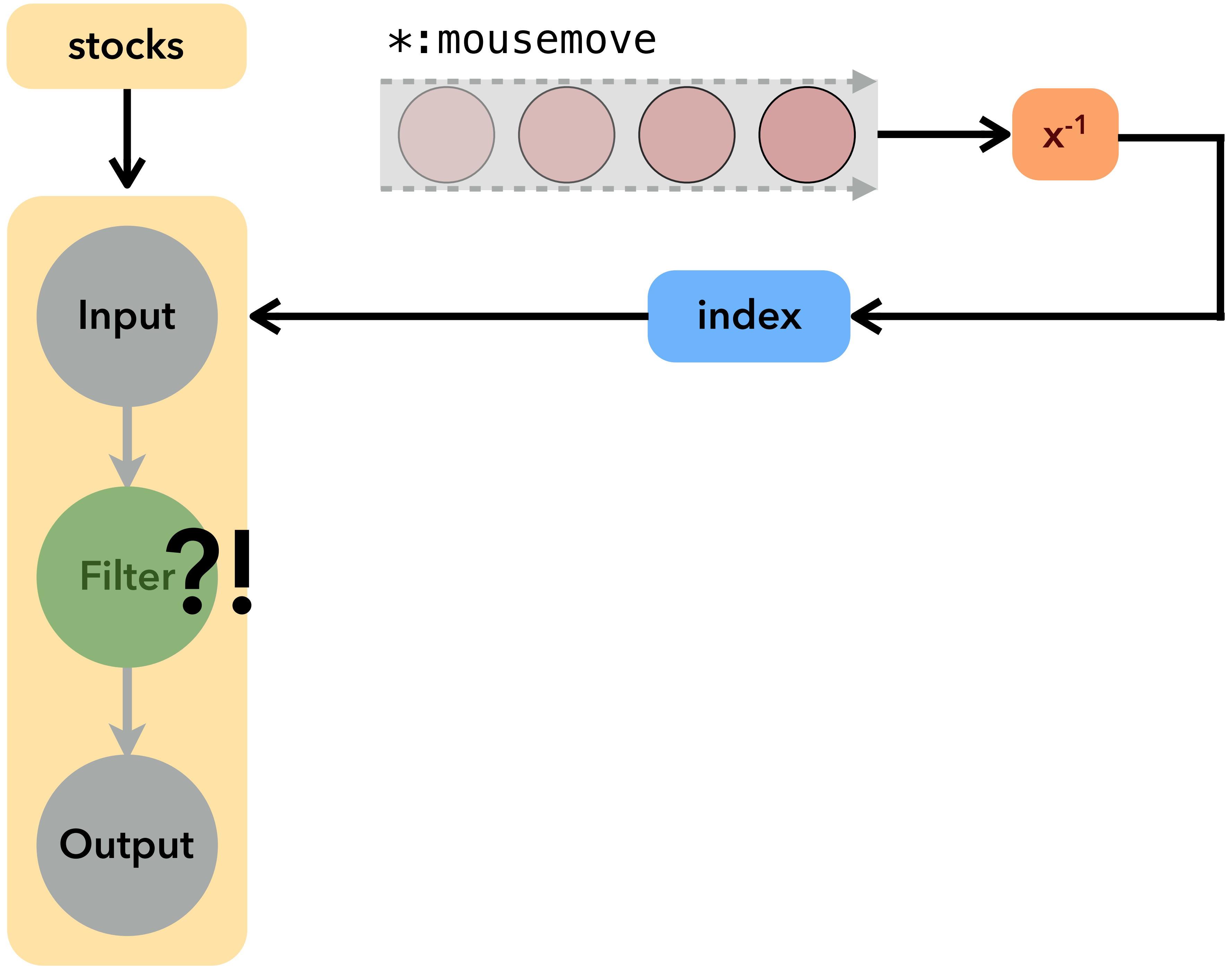
Run Time

```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



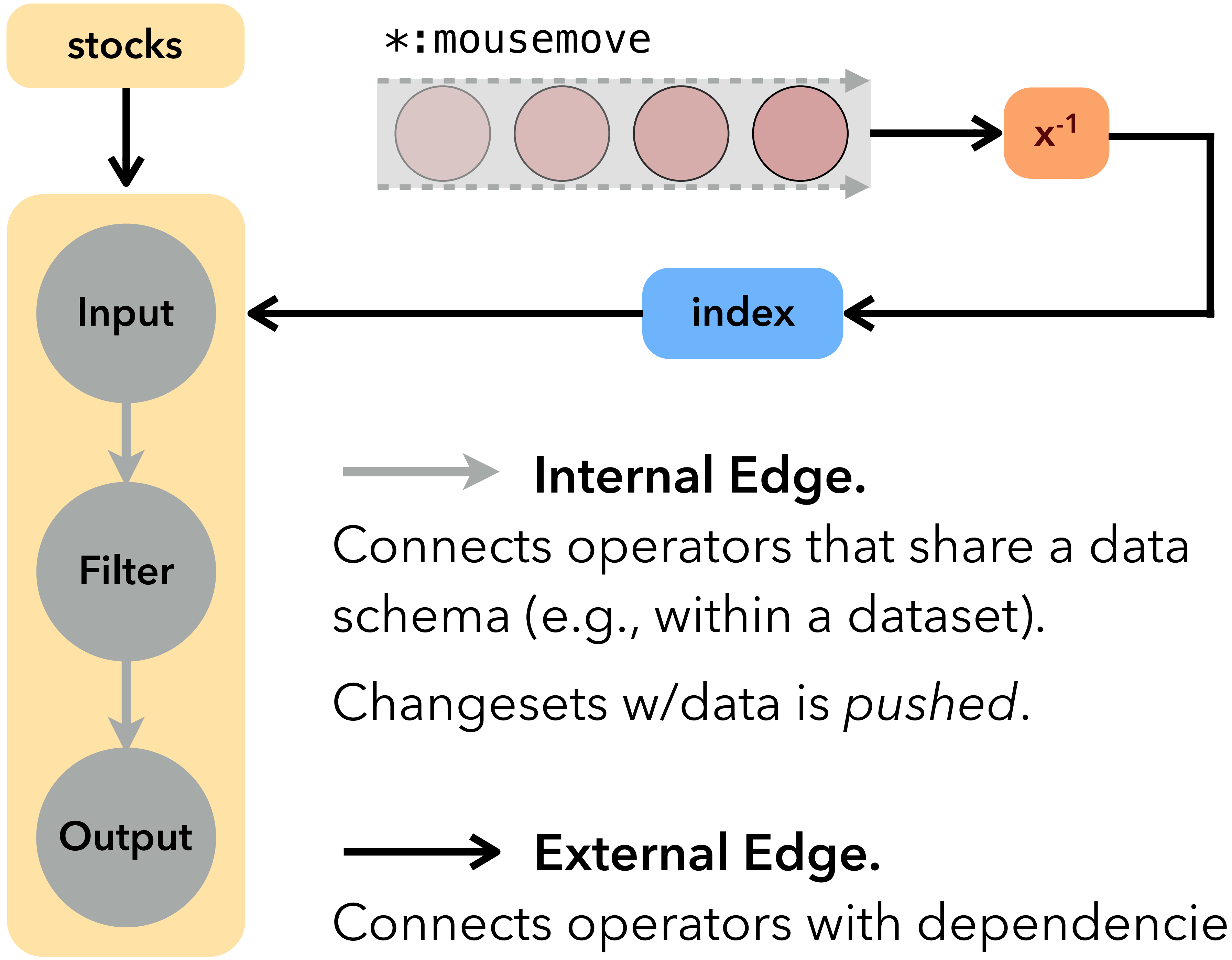
Run Time

```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



Run Time

```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



Internal Edge. Connects operators that share a data schema (e.g., within a dataset).

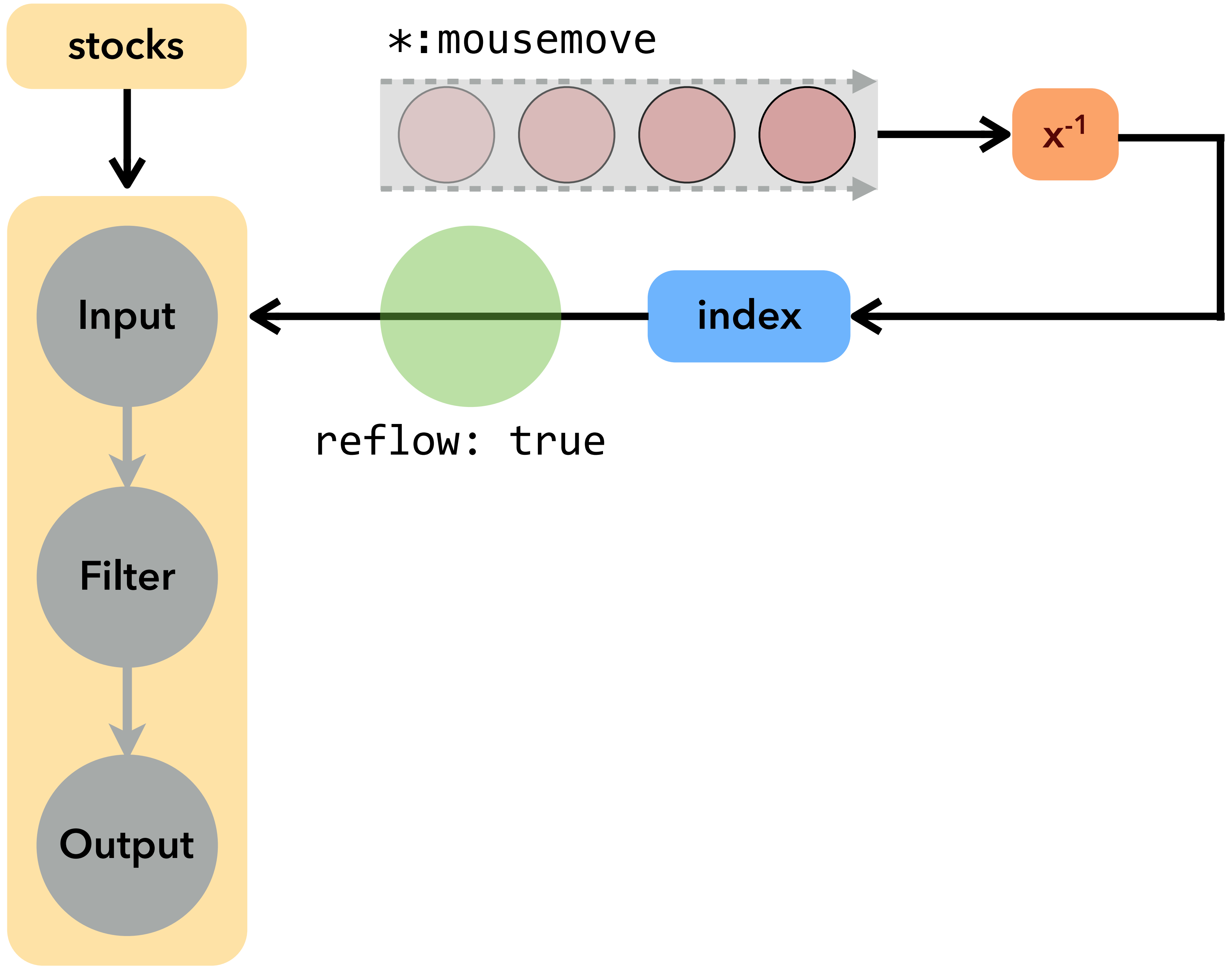
Changesets w/data is *pushed*.

External Edge. Connects operators with dependencies (e.g., signals and datasets).

Reflow changesets. Data is *pulled*.

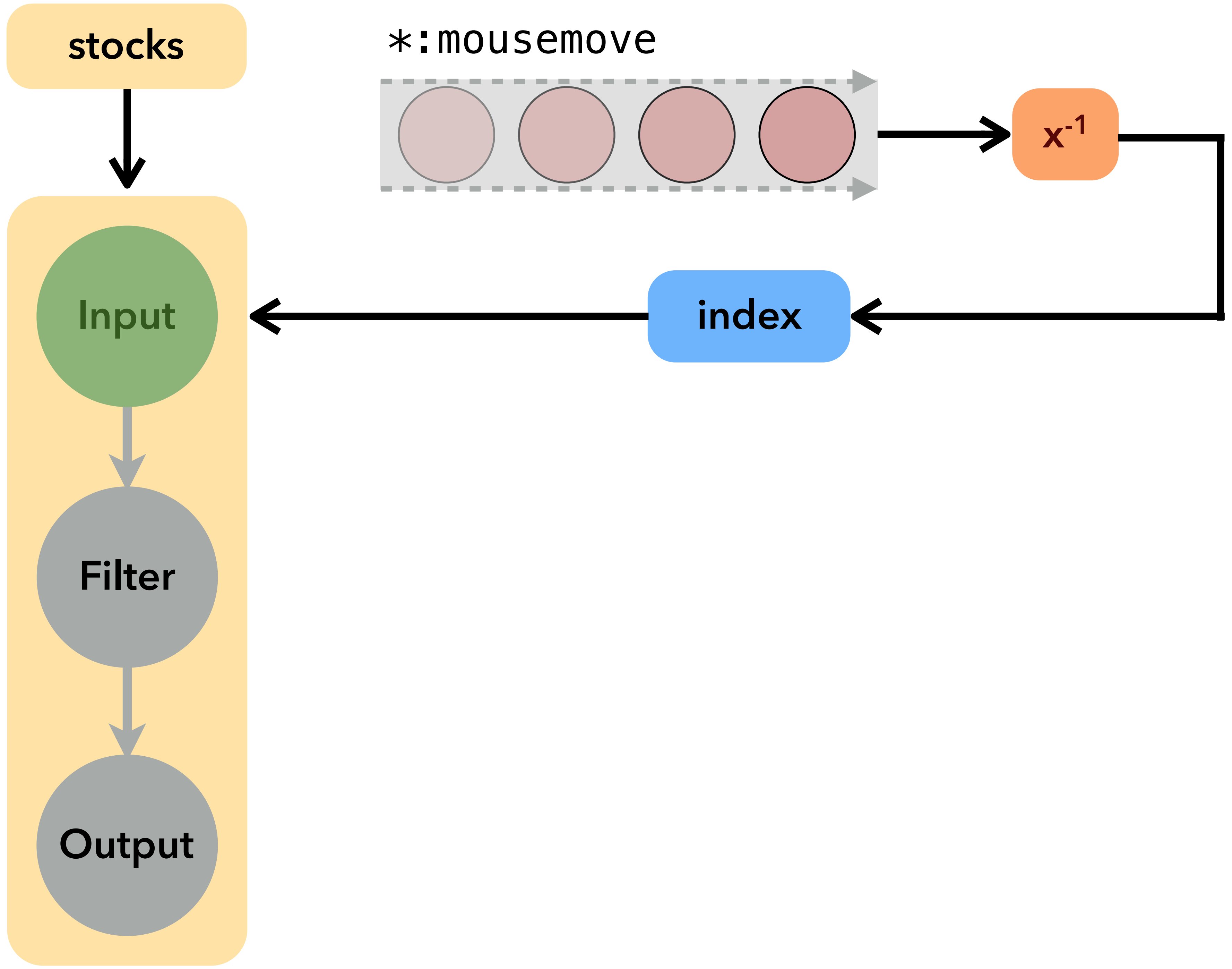
Run Time

```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



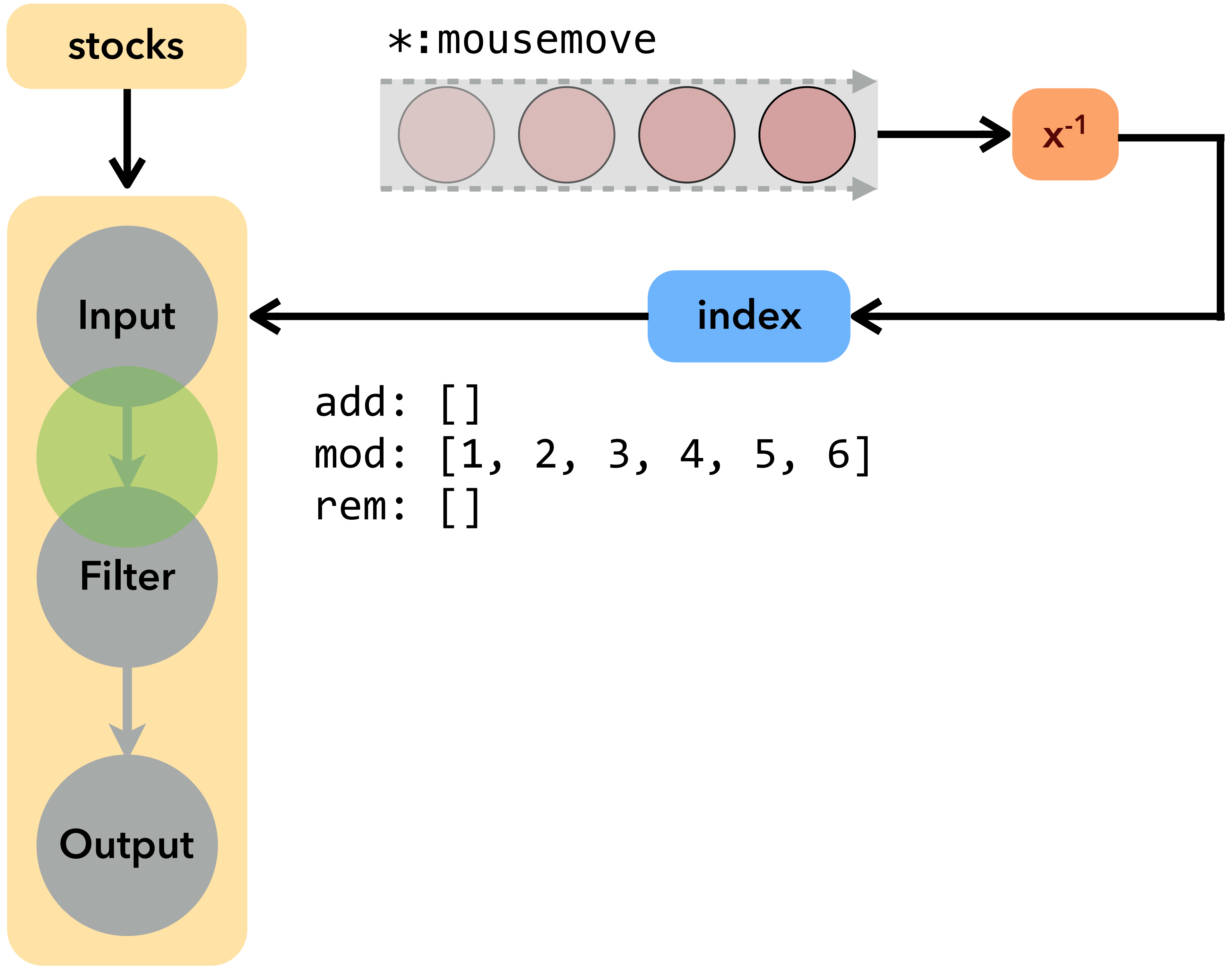
Run Time

```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



Run Time

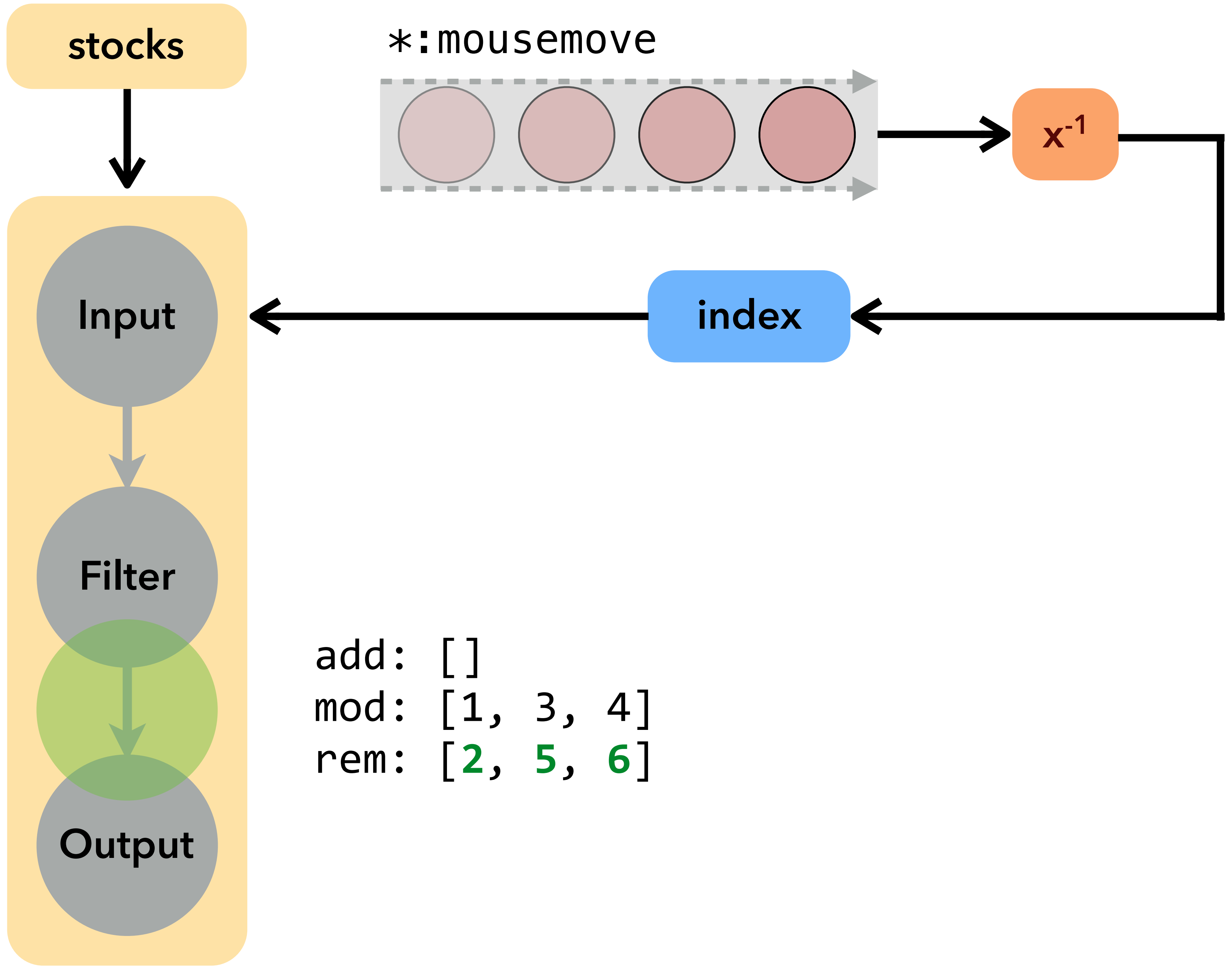
```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



```
add: []
mod: [1, 2, 3, 4, 5, 6]
rem: []
```

Run Time

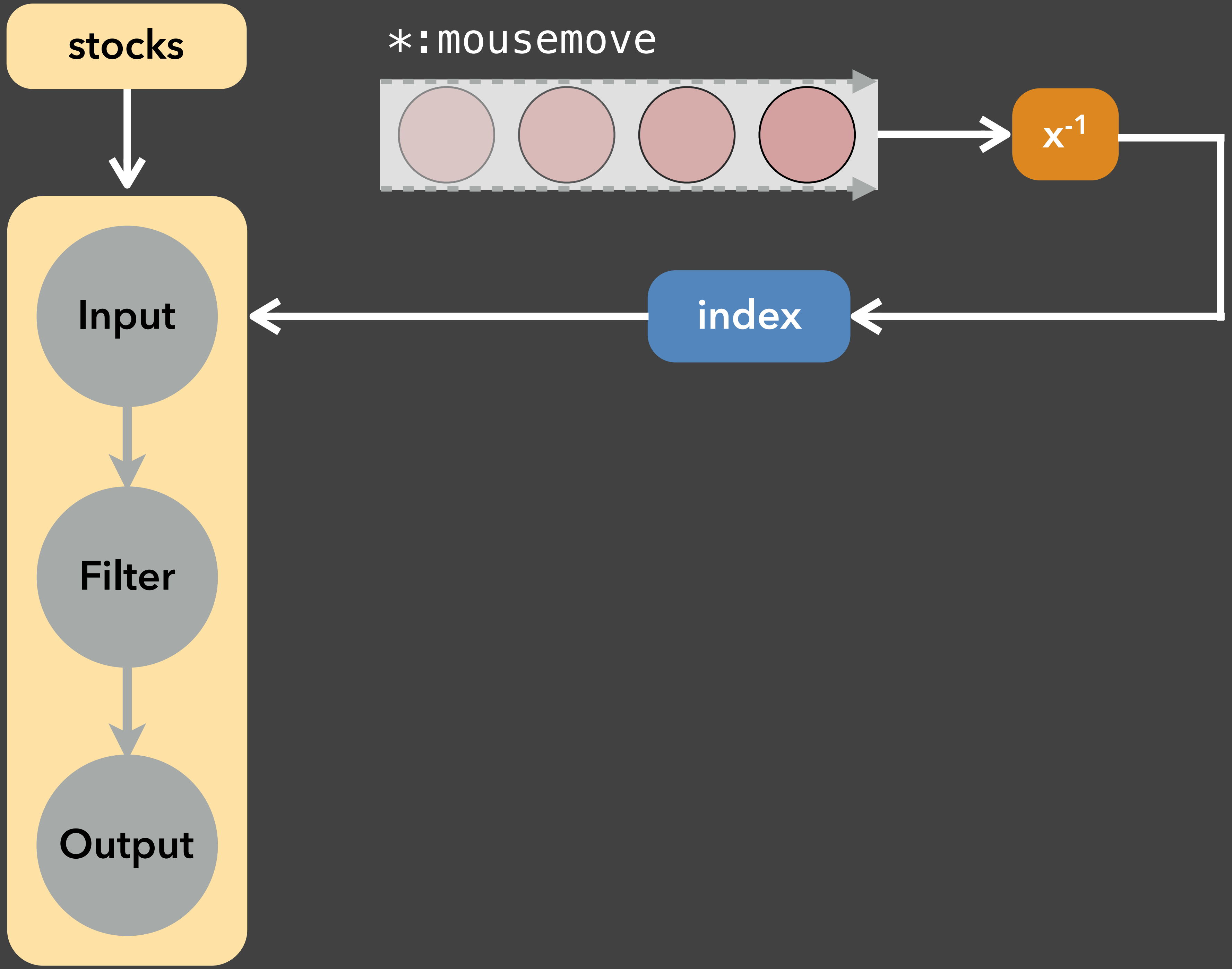
```
{
  "data": [
    {...},
    {...
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



```
add: []
mod: [1, 3, 4]
rem: [2, 5, 6]
```

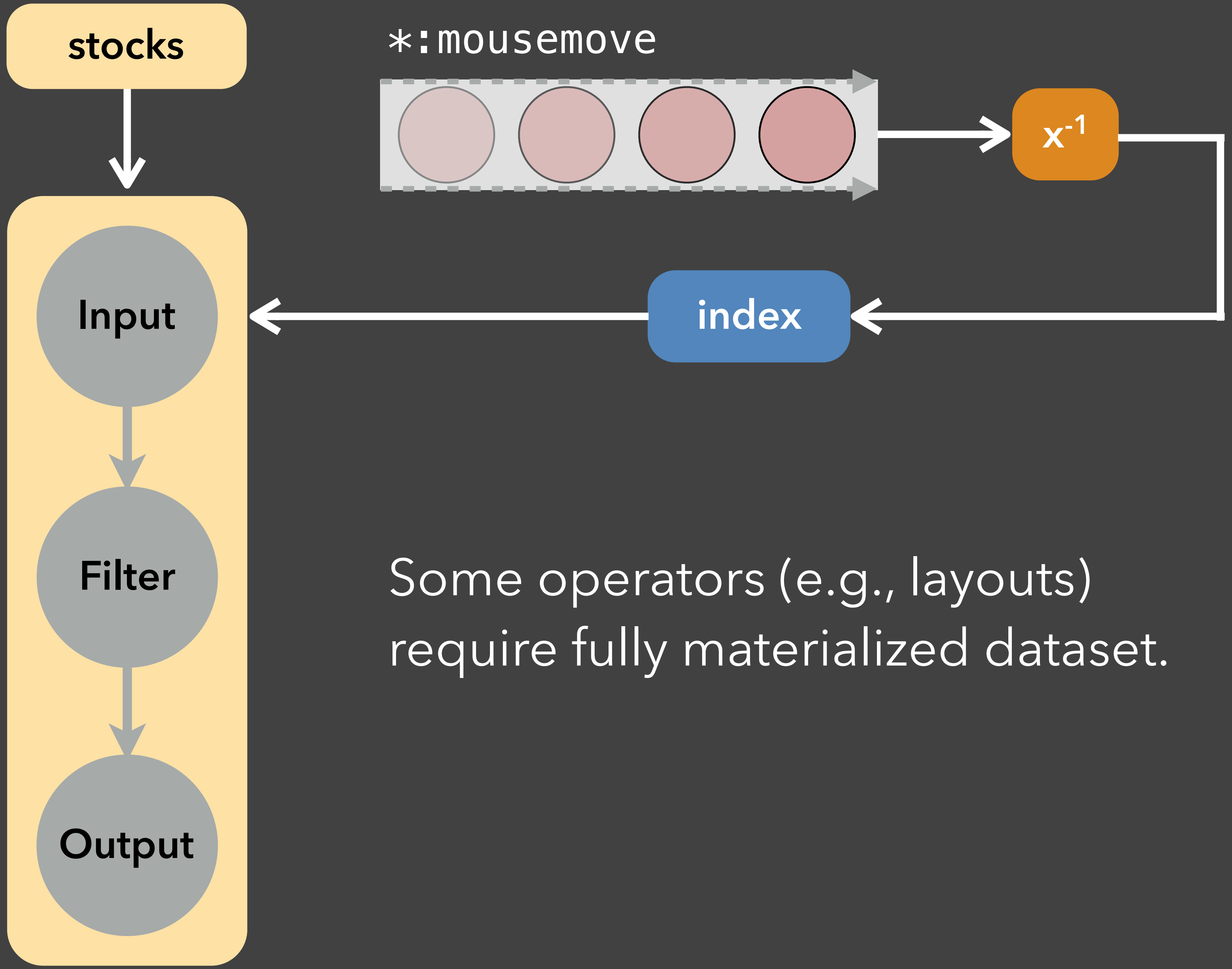
Compile Time

```
{
  "data": [
    {...},
    {
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



Compile Time

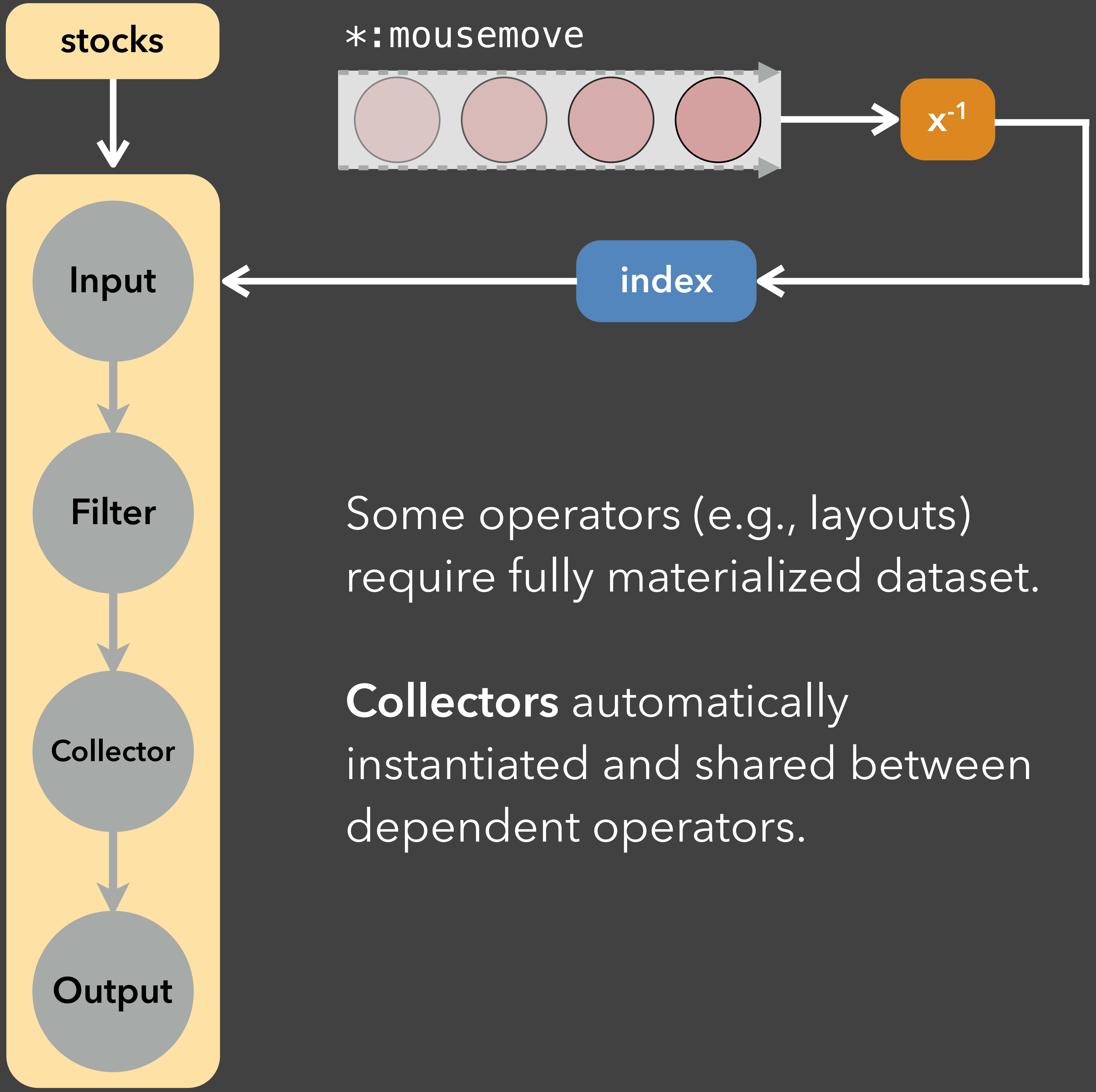
```
{
  "data": [
    {...},
    {
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



Some operators (e.g., layouts) require fully materialized dataset.

Compile Time

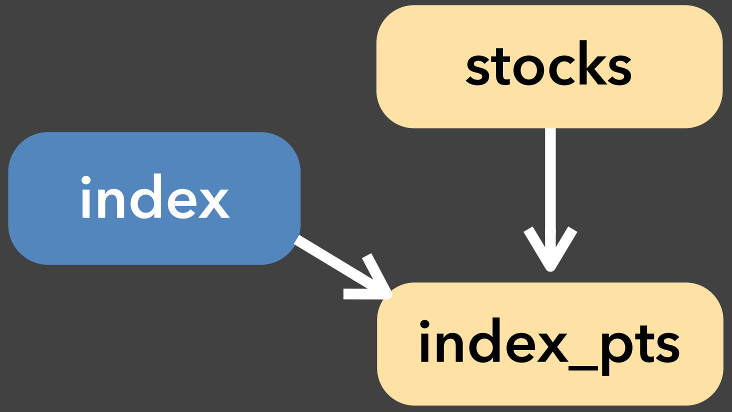
```
{
  "data": [
    {...},
    {
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```



Some operators (e.g., layouts) require fully materialized dataset.

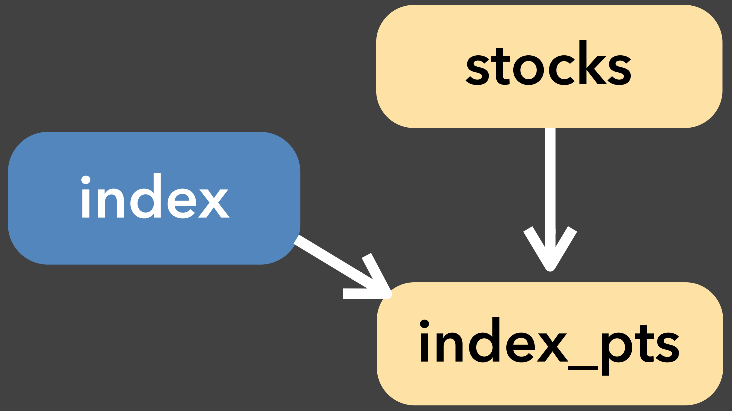
Collectors automatically instantiated and shared between dependent operators.

Compile Time



```
{
  "data": [
    {...},
    {
      "name": "index_pts",
      "source": "stocks",
      "transform": [{
        "type": "filter",
        "test": "month(datum) ==
month(index) && year(datum) ==
year(index)"
      }]
    },
    {...}
  ]
}
```

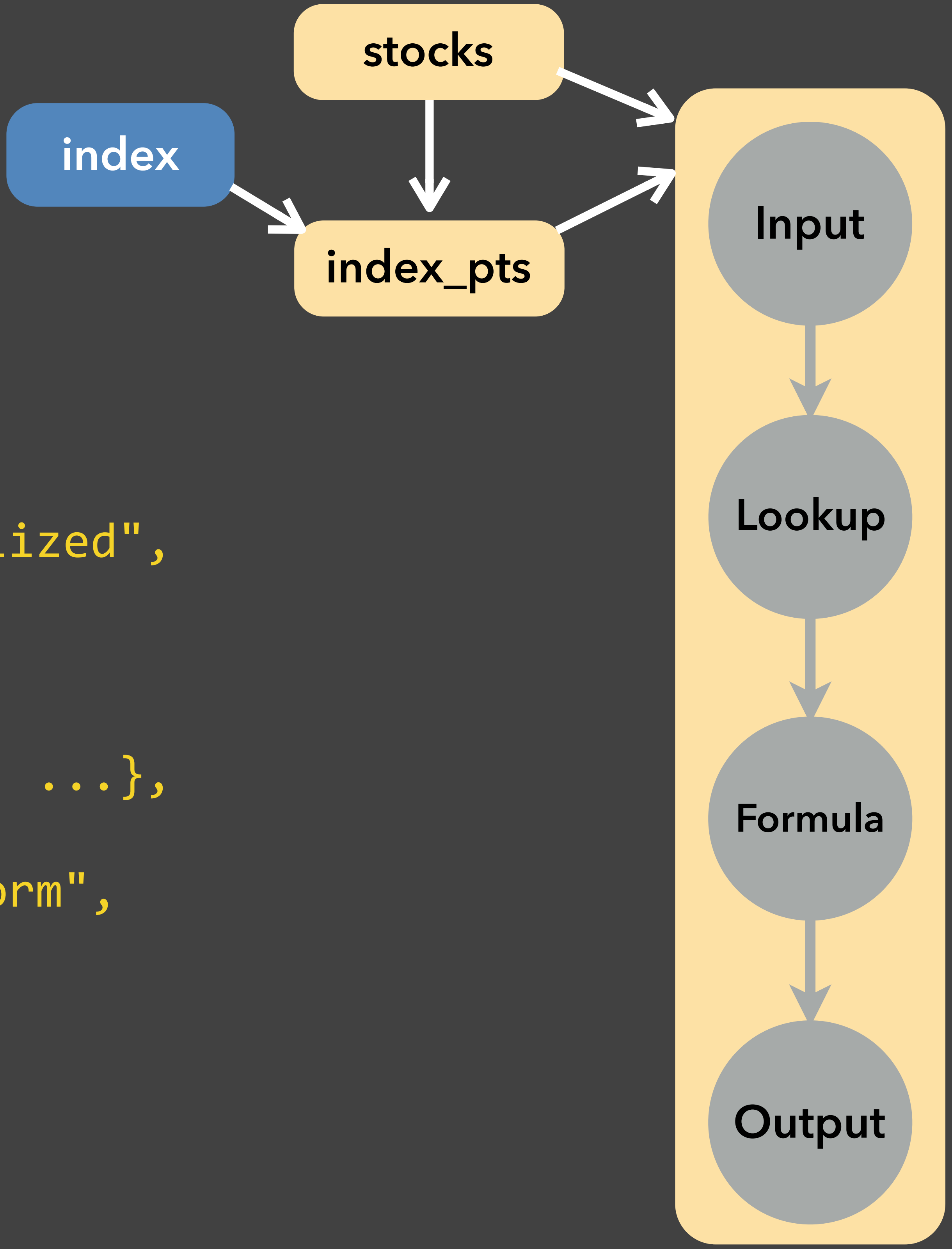
Compile Time



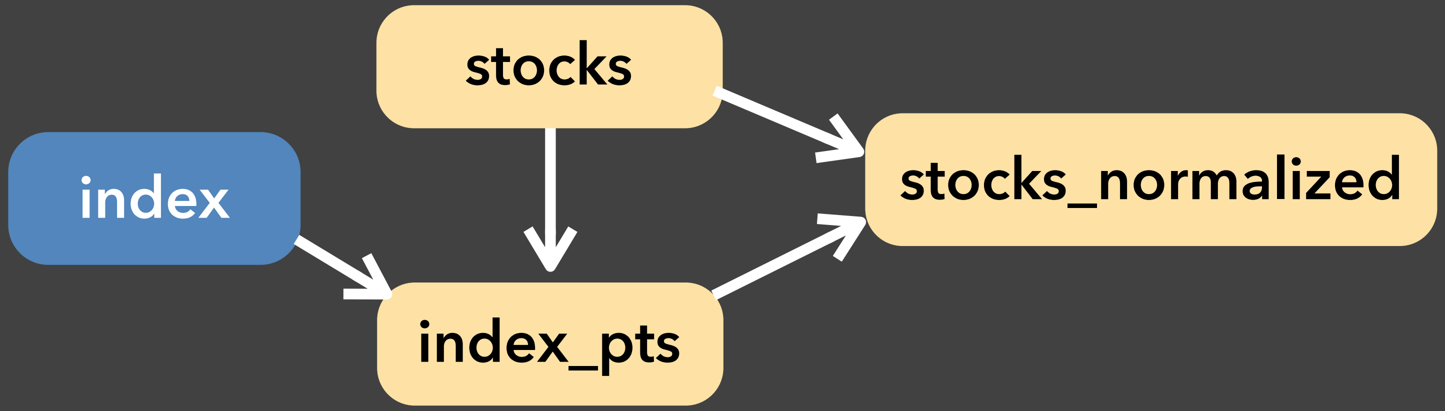
```
{
  "data": [
    {...},
    {...},
    {
      "name": "stocks_normalized",
      "source": "stocks",
      "transform": [
        { "type": "lookup",
          "on": "index_pts", ...},
        { "type": "formula",
          "field": "price_norm",
          "expr": ... }
      ]
    }
  ]
}
```

Compile Time

```
{
  "data": [
    {...},
    {...},
    {
      "name": "stocks_normalized",
      "source": "stocks",
      "transform": [
        { "type": "lookup",
          "on": "index_pts", ...},
        { "type": "formula",
          "field": "price_norm",
          "expr": ... }
      ]
    }
  ]
}
```



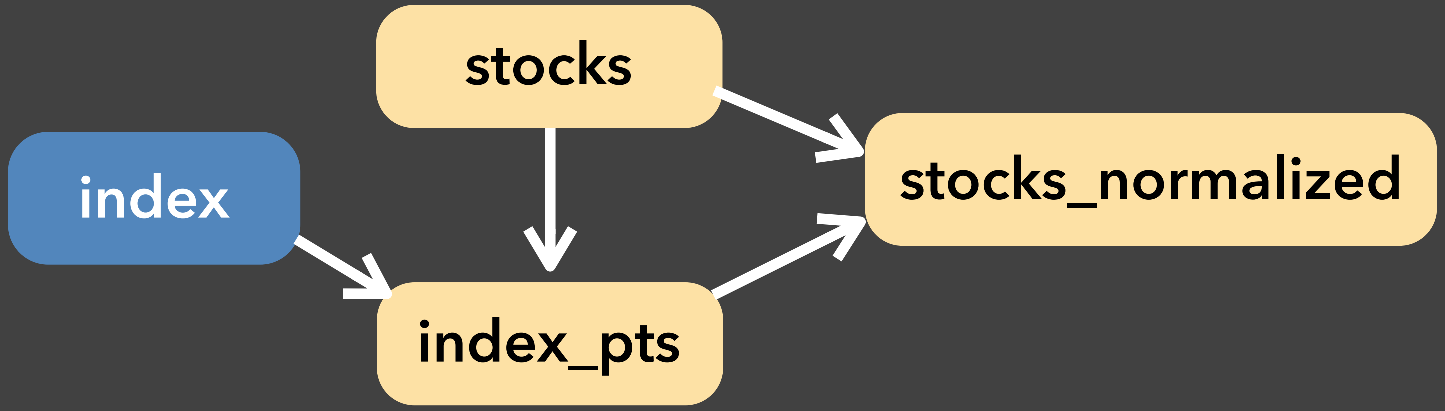
Compile Time



```
{
  "data": [
    {...},
    {...},
    {
      "name": "stocks_normalized",
      "source": "stocks",
      "transform": [
        { "type": "lookup",
          "on": "index_pts", ...},
        { "type": "formula",
          "field": "price_norm",
          "expr": ... }
      ]
    }
  ]
}
```

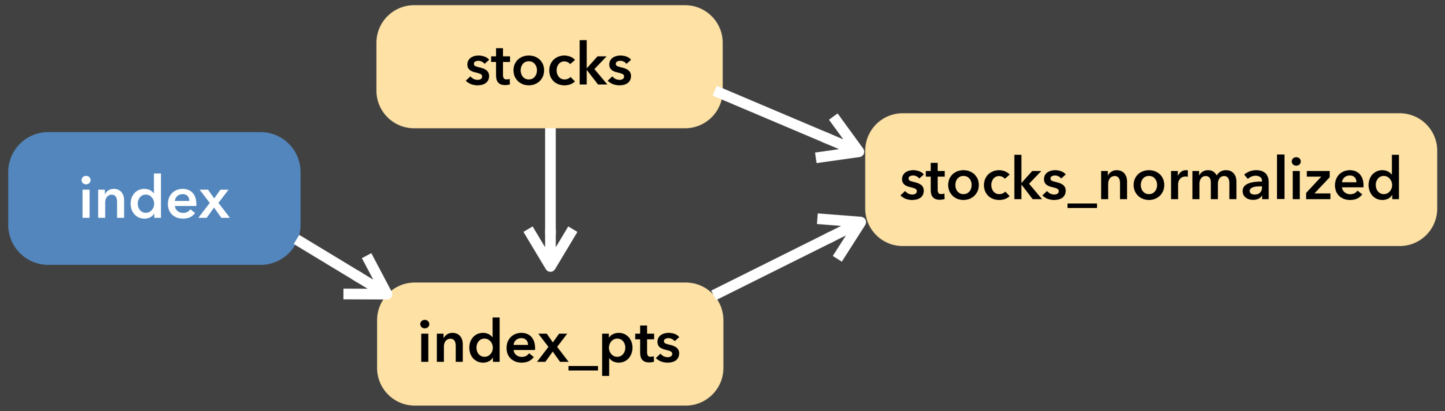
Compile Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    }
  ]
}
```


$$\begin{tikzcd} & \text{stocks} \arrow[d] \arrow[r] & \text{stocks\_normalized} \\ \text{index} \arrow[r] & \text{index\_pts} \arrow[ur] & \end{tikzcd}$$

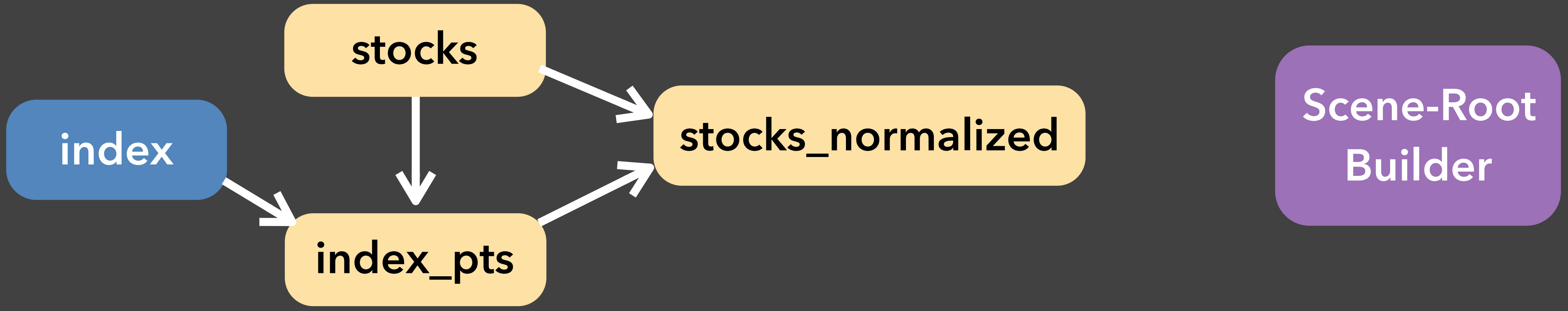
Compile Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```


$$\begin{tikzcd}
 & \text{stocks} \arrow[d] \arrow[r] & \text{stocks\_normalized} \\
\text{index} \arrow[r] & \text{index\_pts} \arrow[ur] &
\end{tikzcd}$$

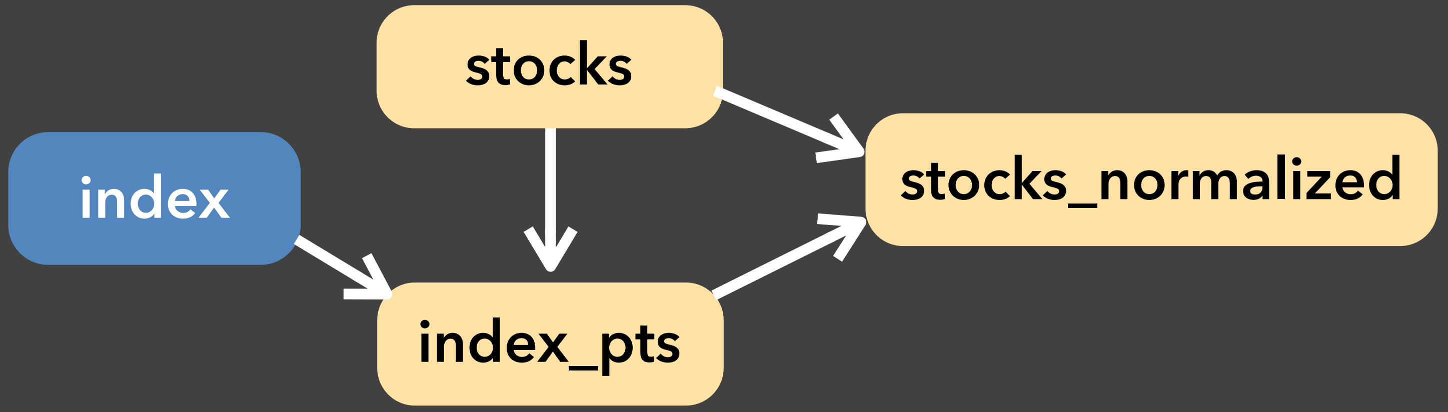
Compile Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



Compile Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



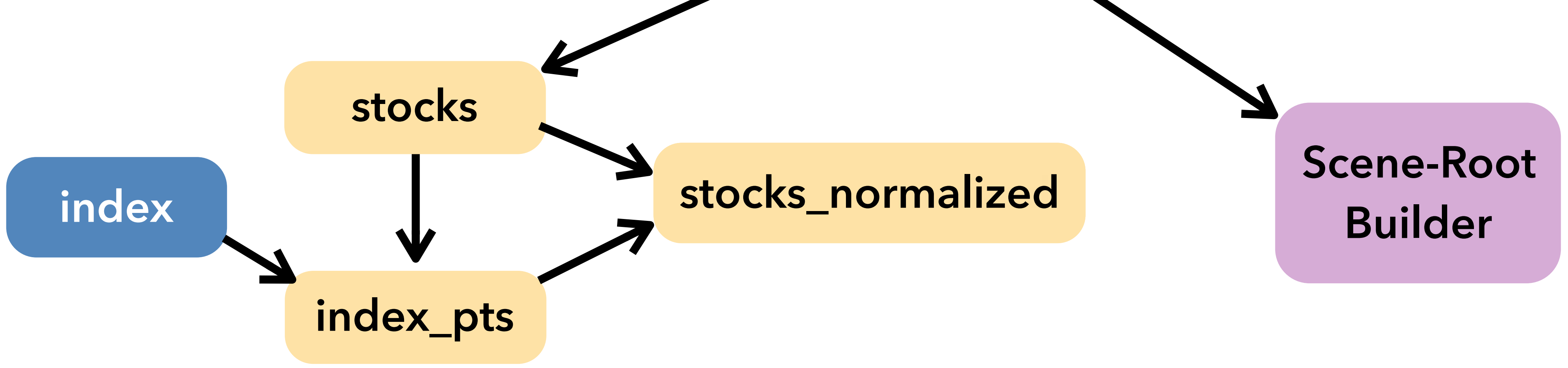
At compile-time, we do not know what data we will see.

What are the different ticker symbols?
How many facets will be constructed?
How will groups build out with nested children?

Dynamic Self-Instantiating Dataflows.
Operators extend/prune dataflow branches at runtime.

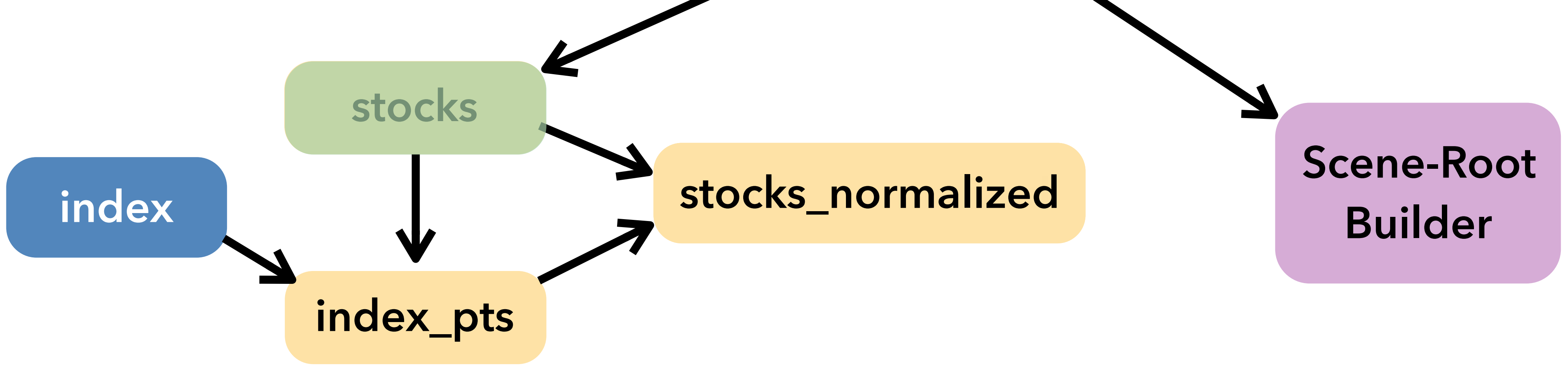
Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



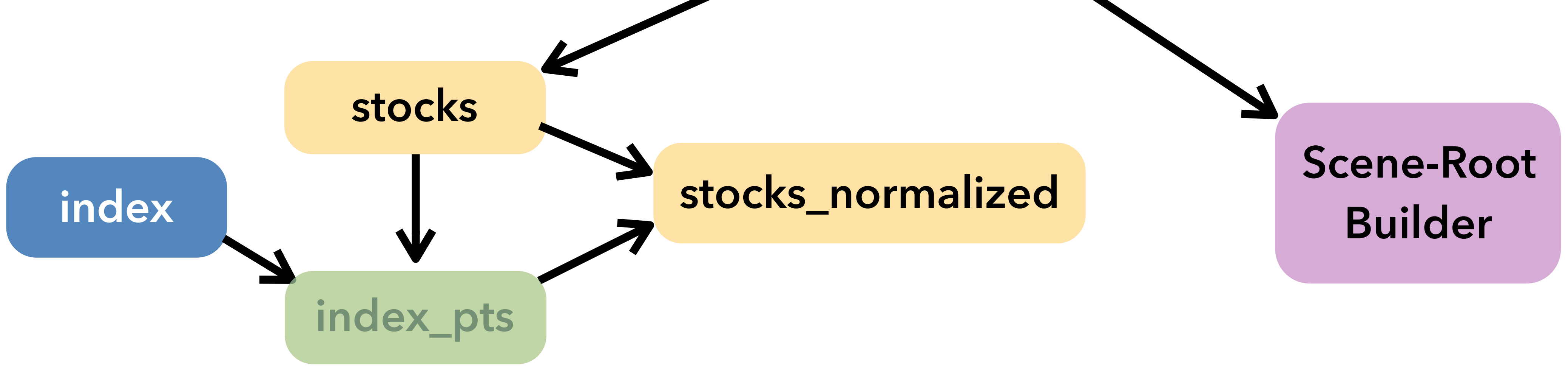
Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



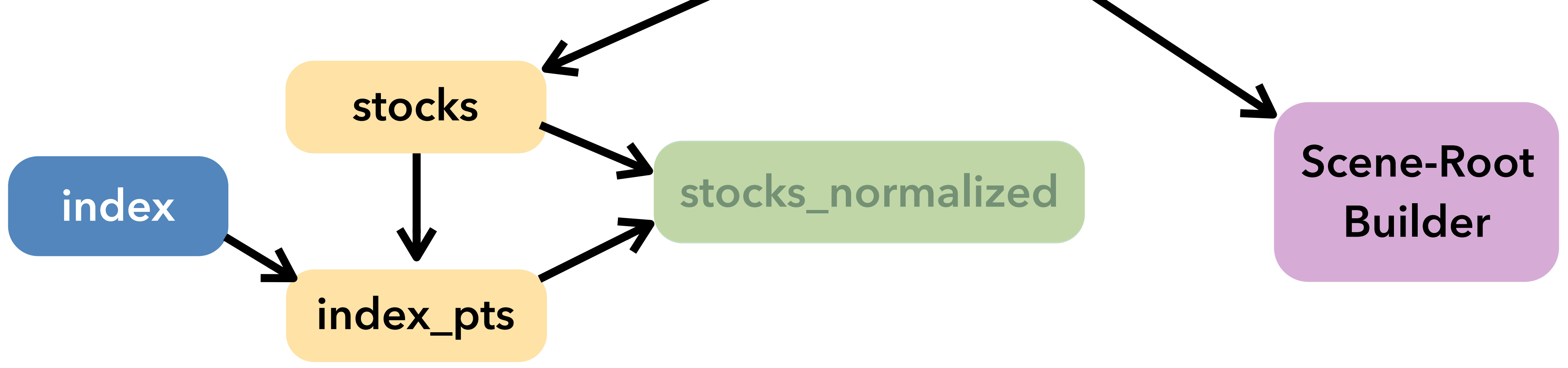
Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



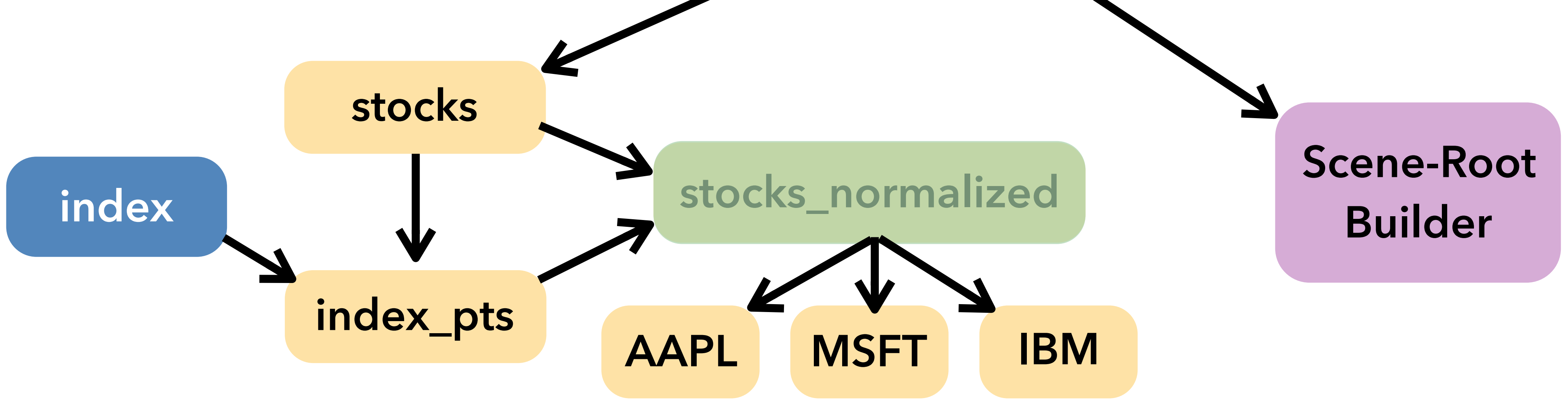
Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



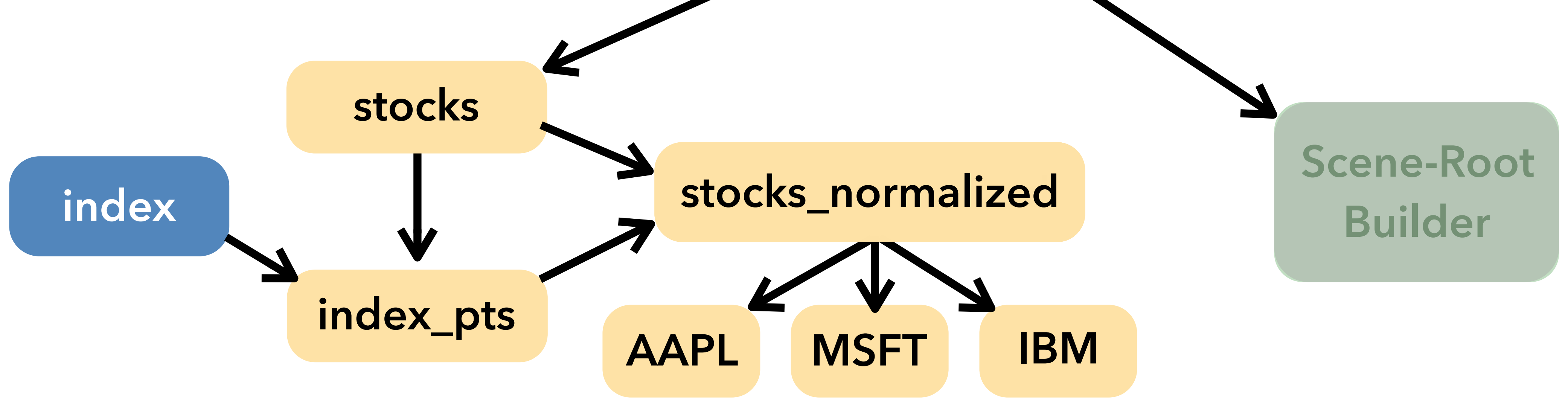
Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



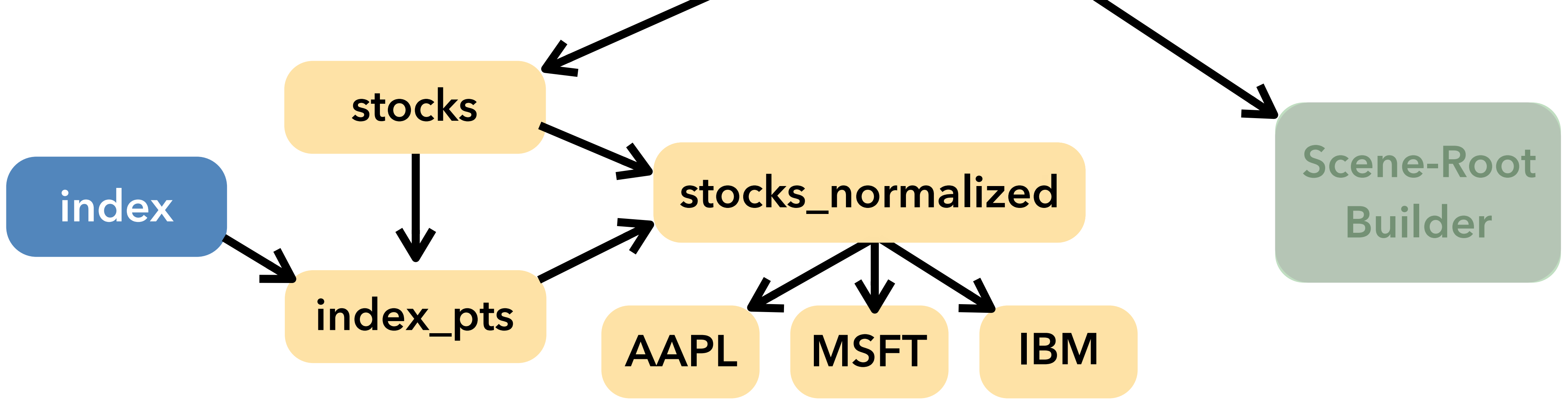
Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



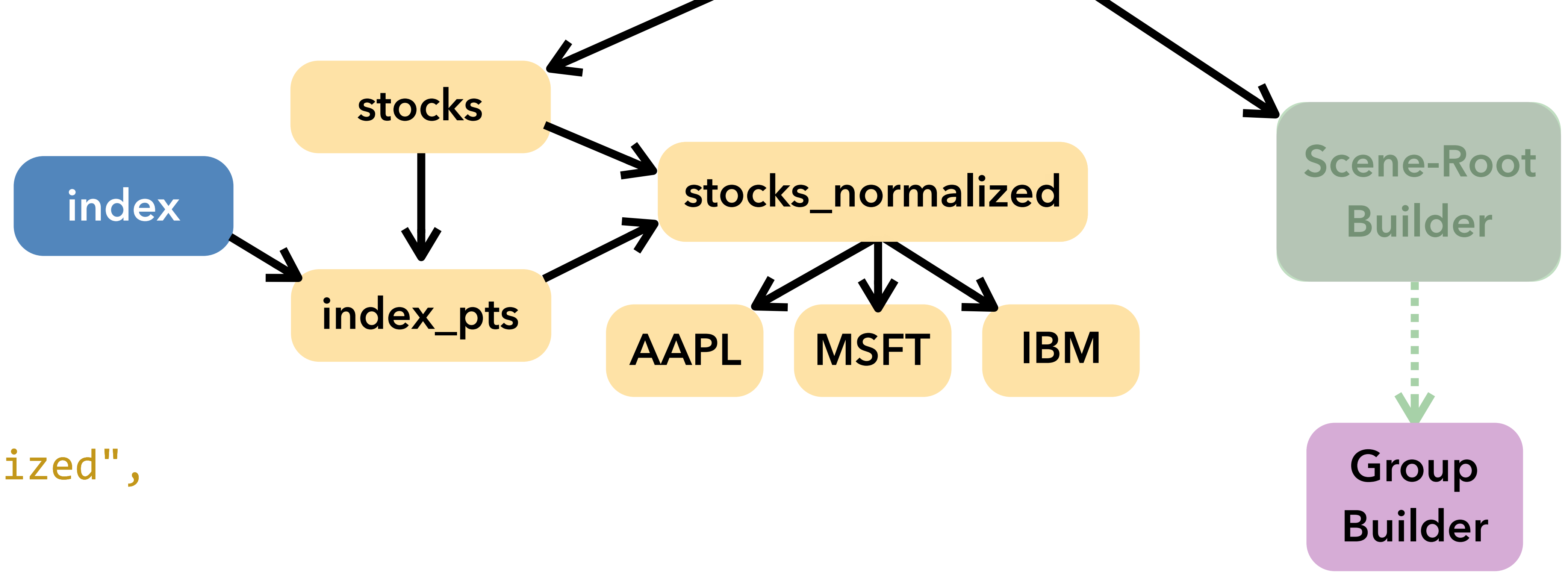
Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



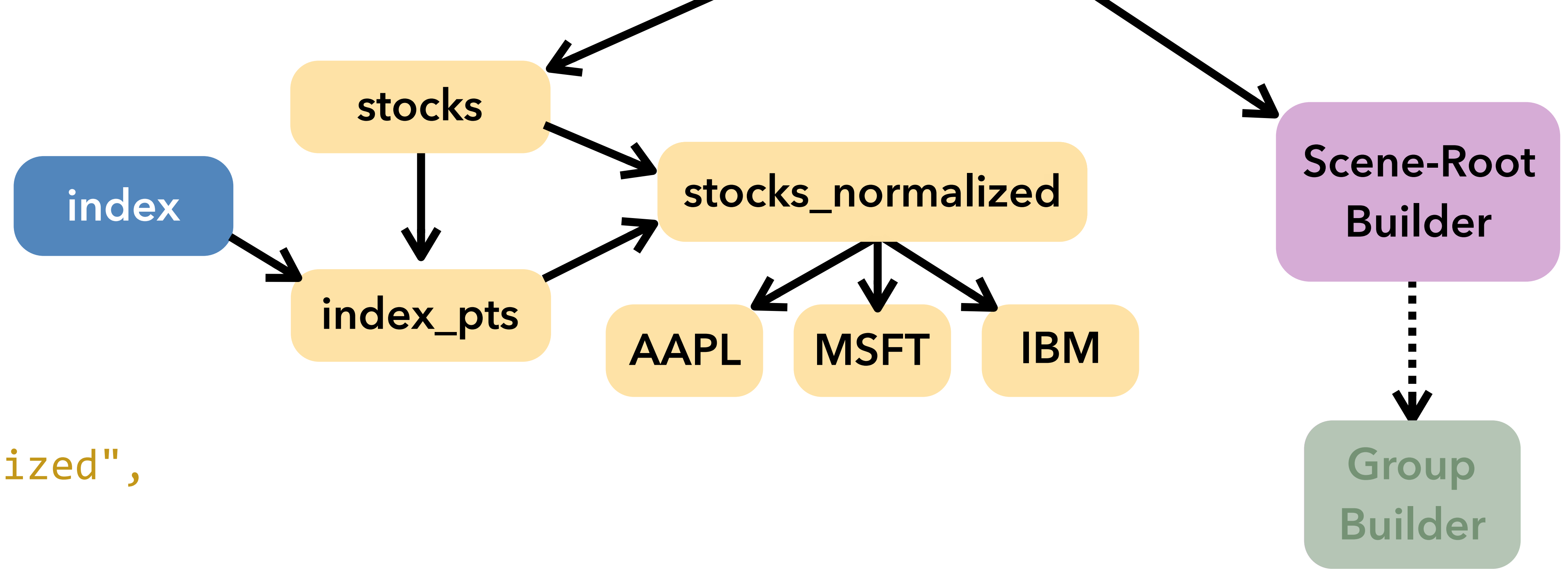
Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



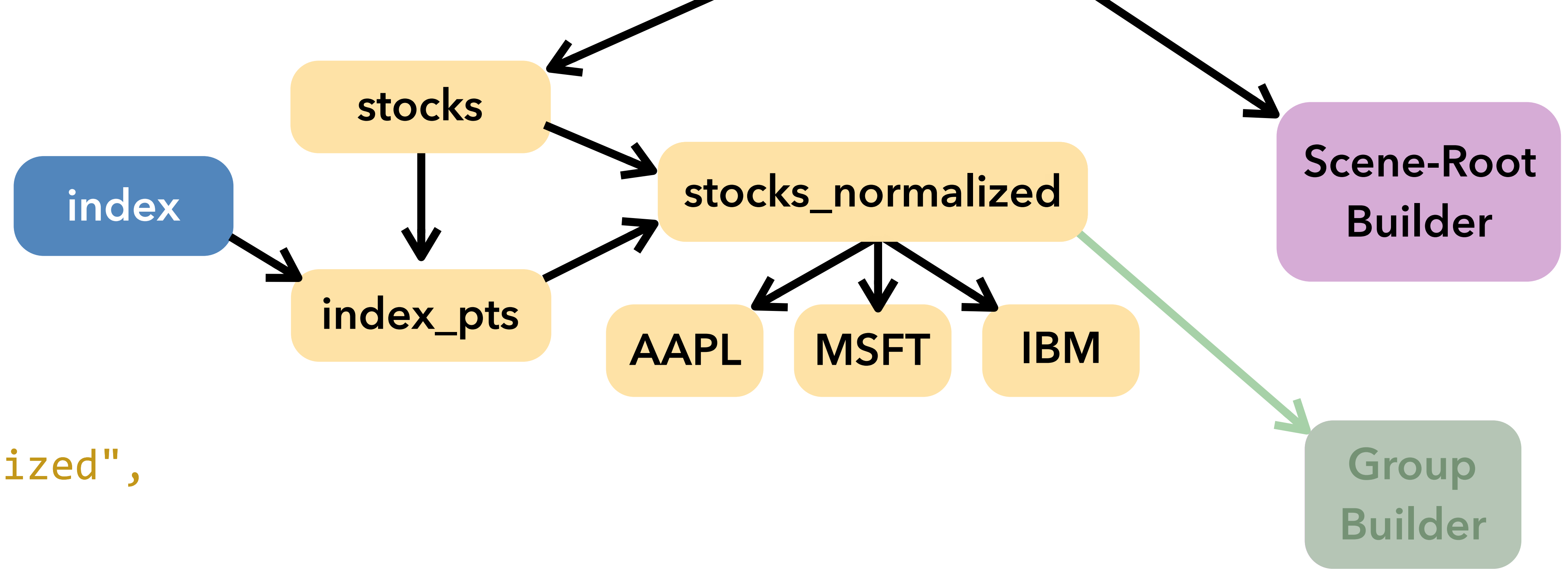
Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



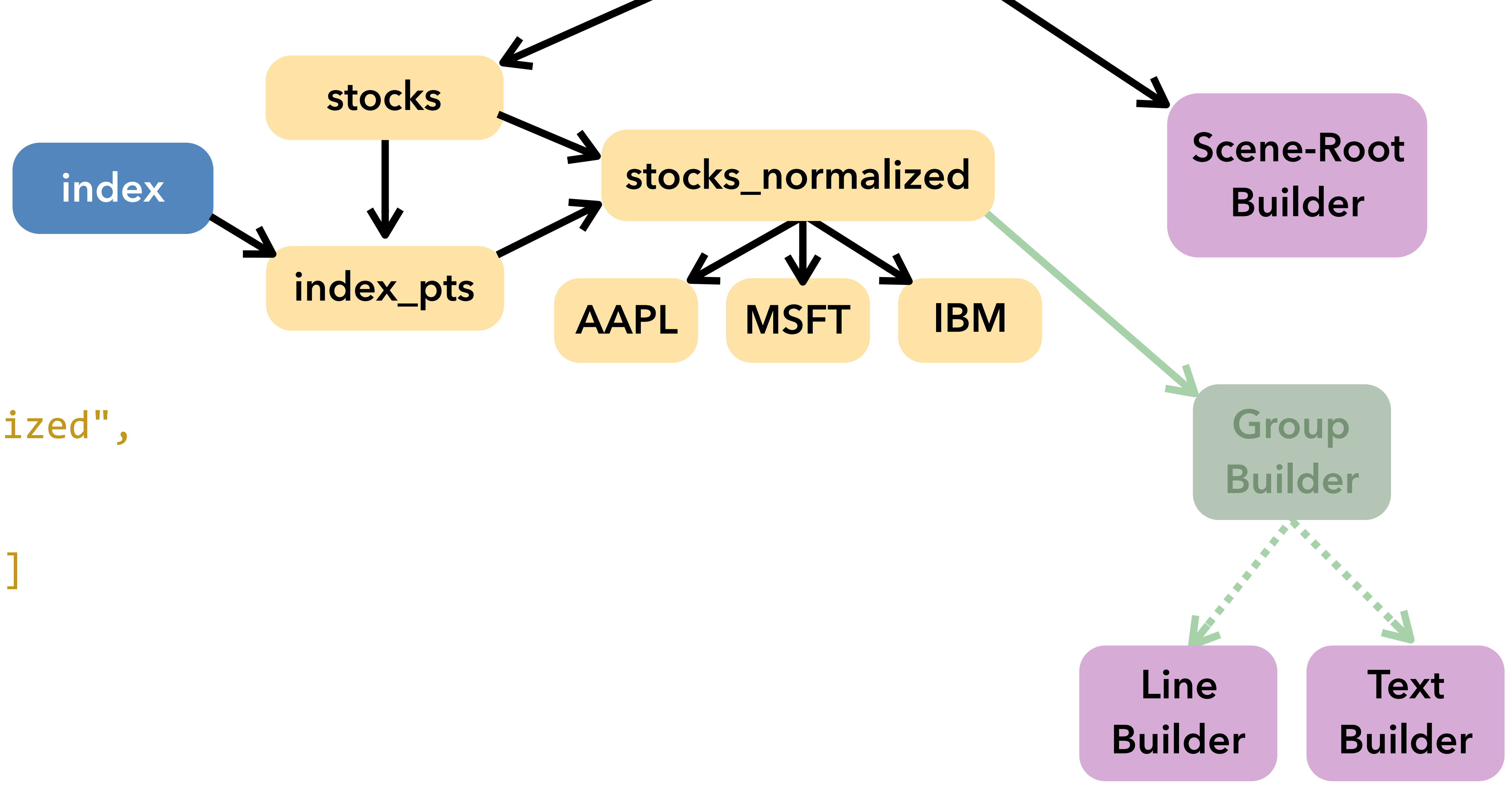
Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



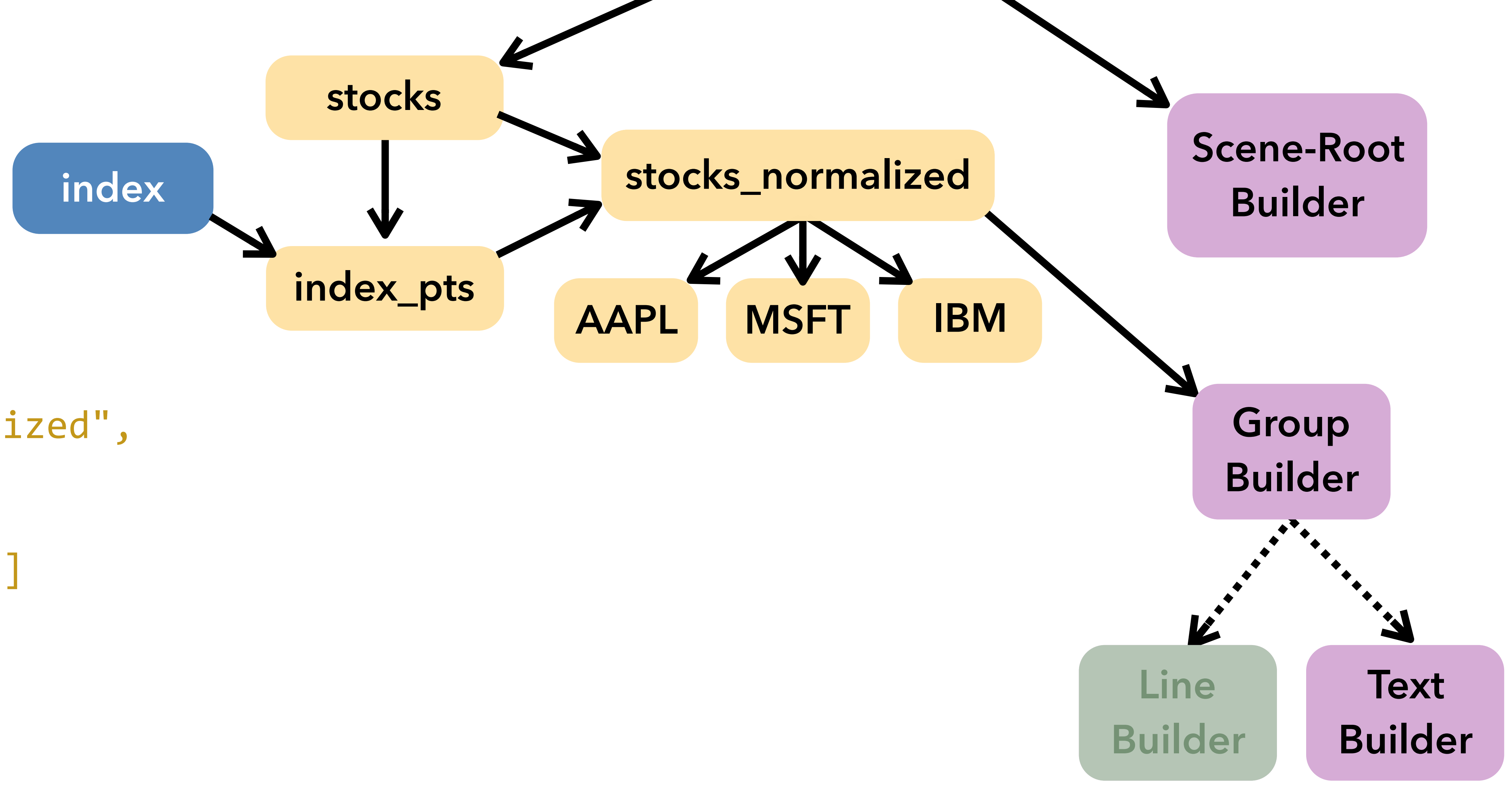
Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



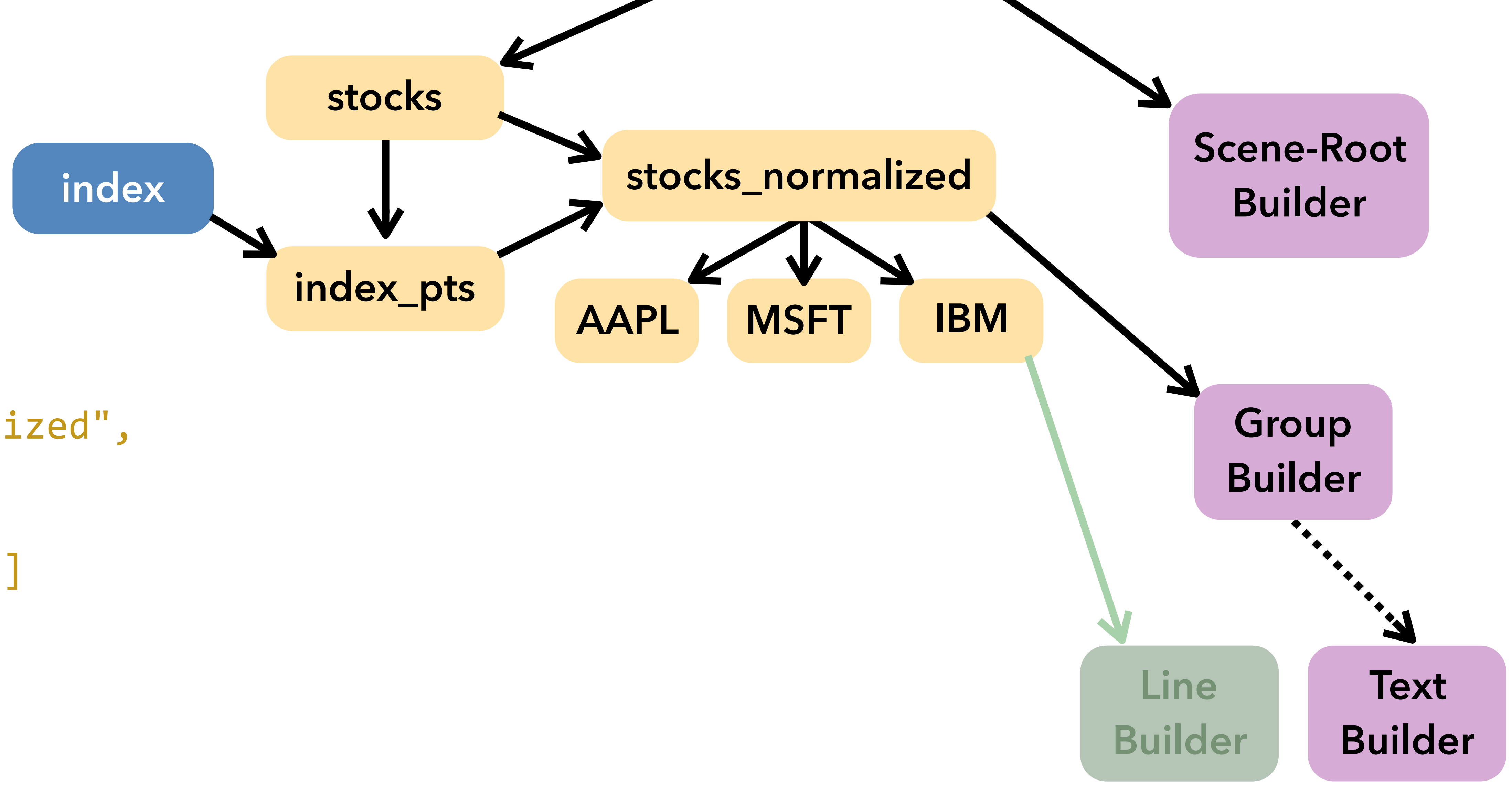
Run Time

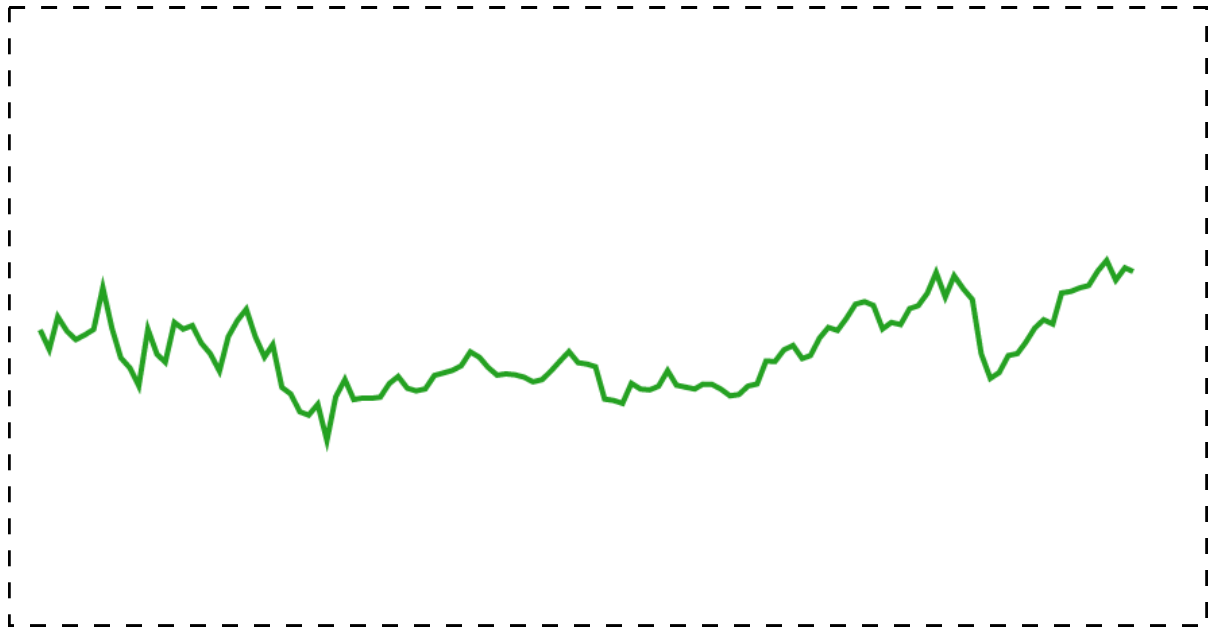
```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```



Run Time

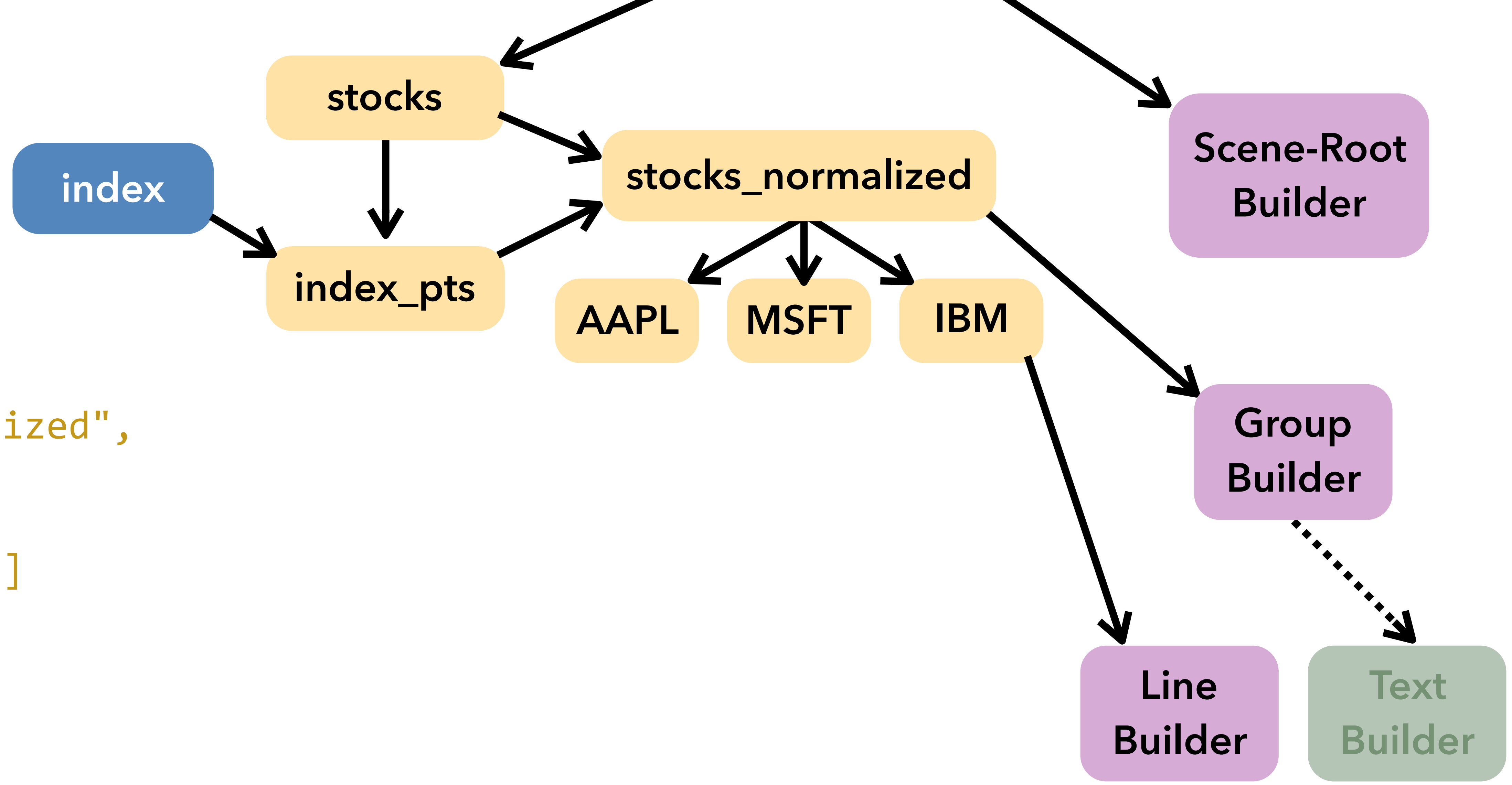
```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```

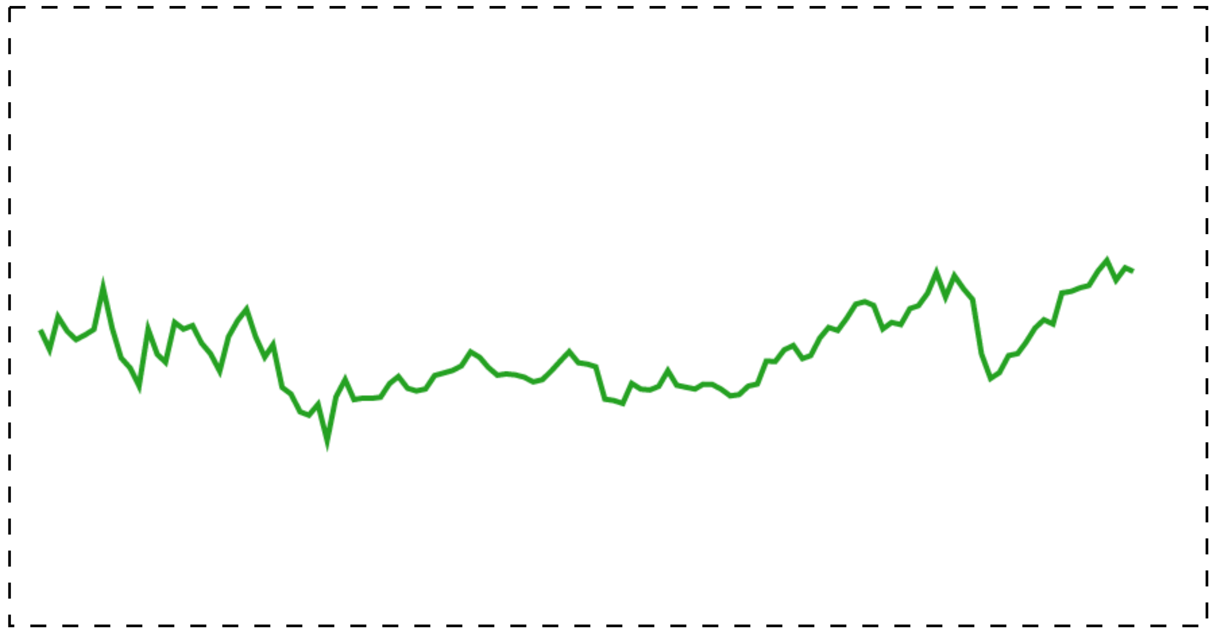




Run Time

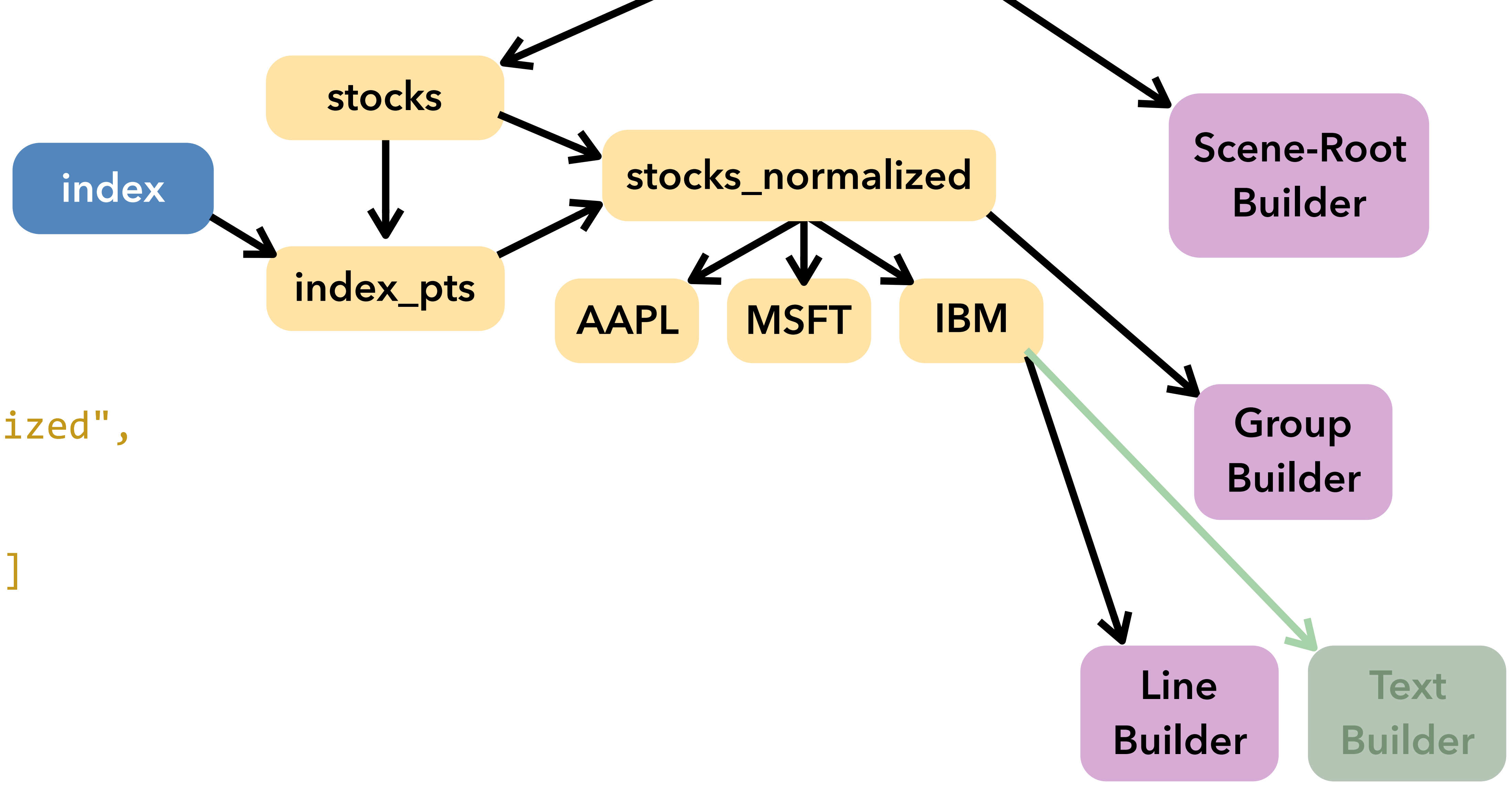
```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```

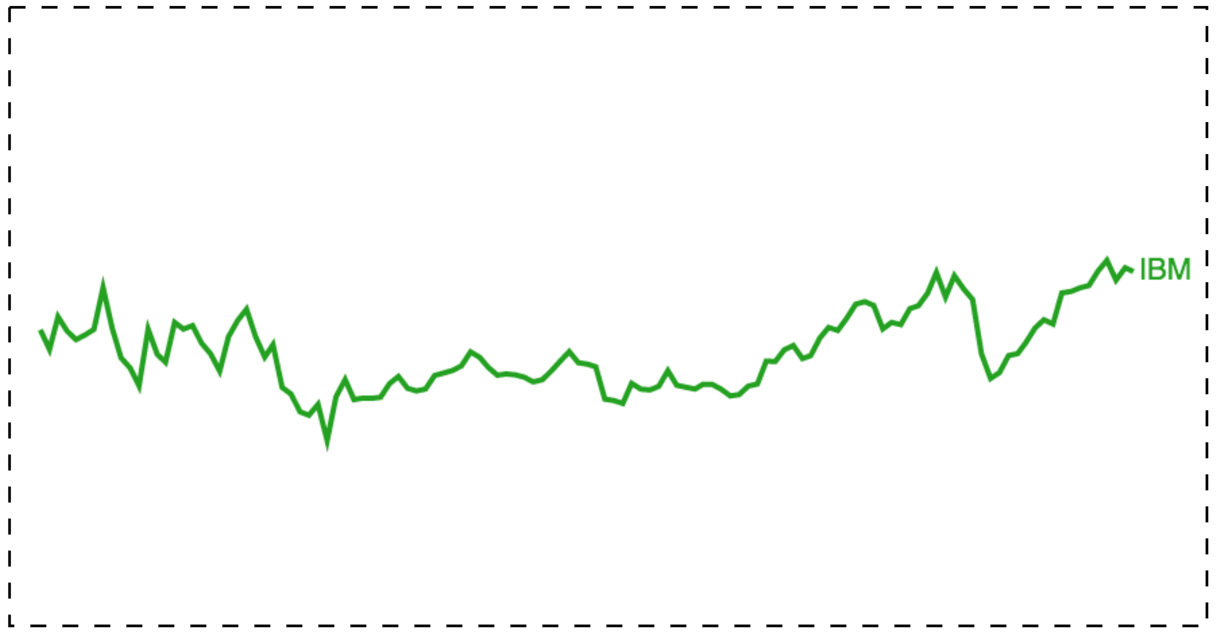




Run Time

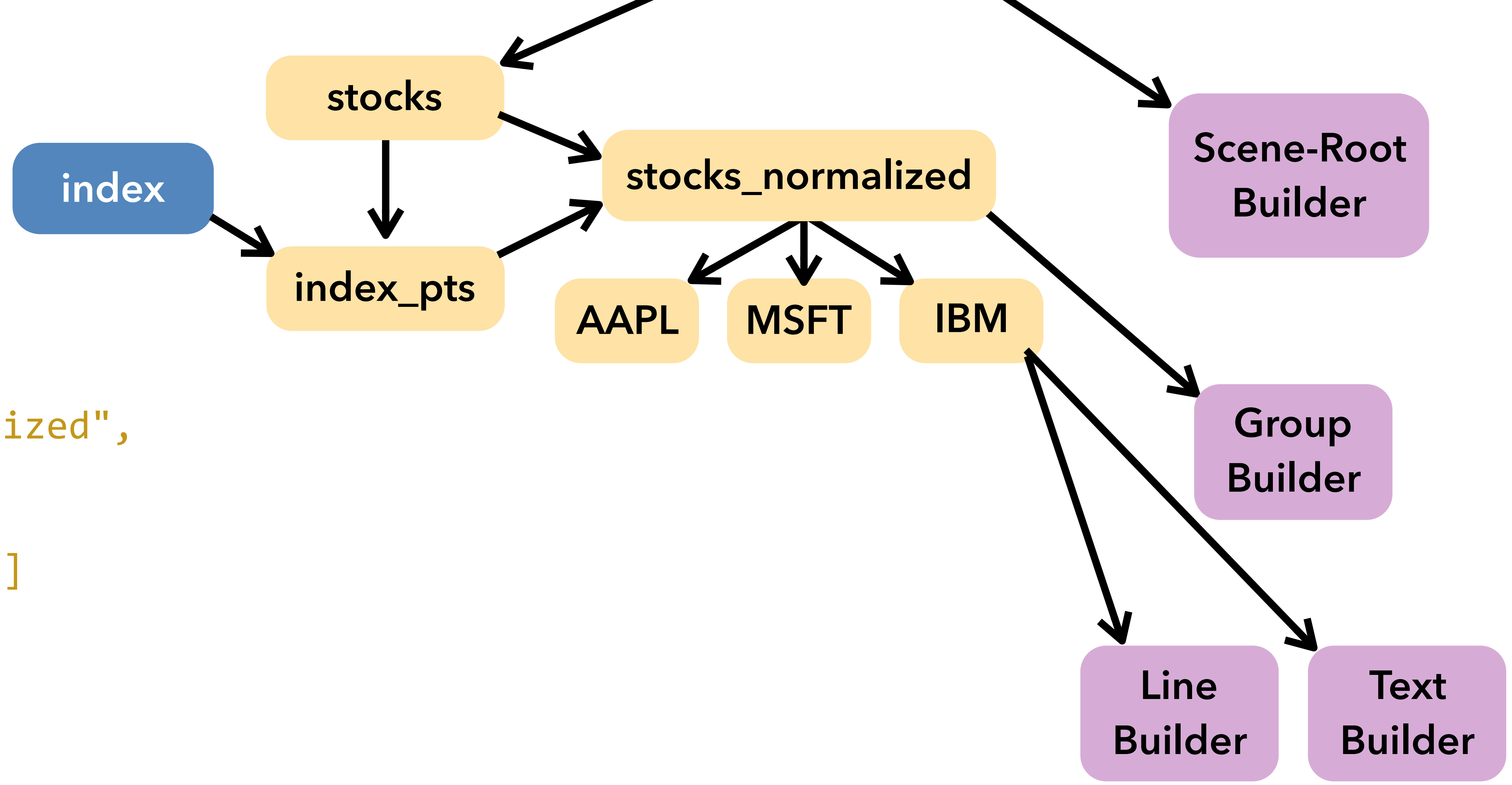
```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```

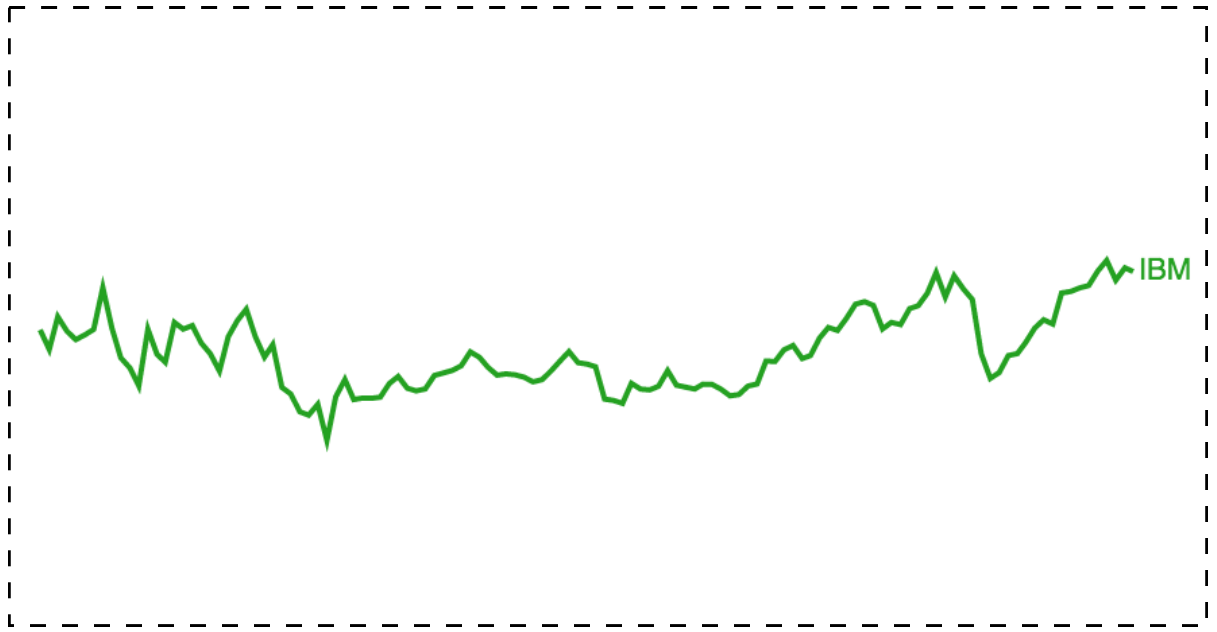




Run Time

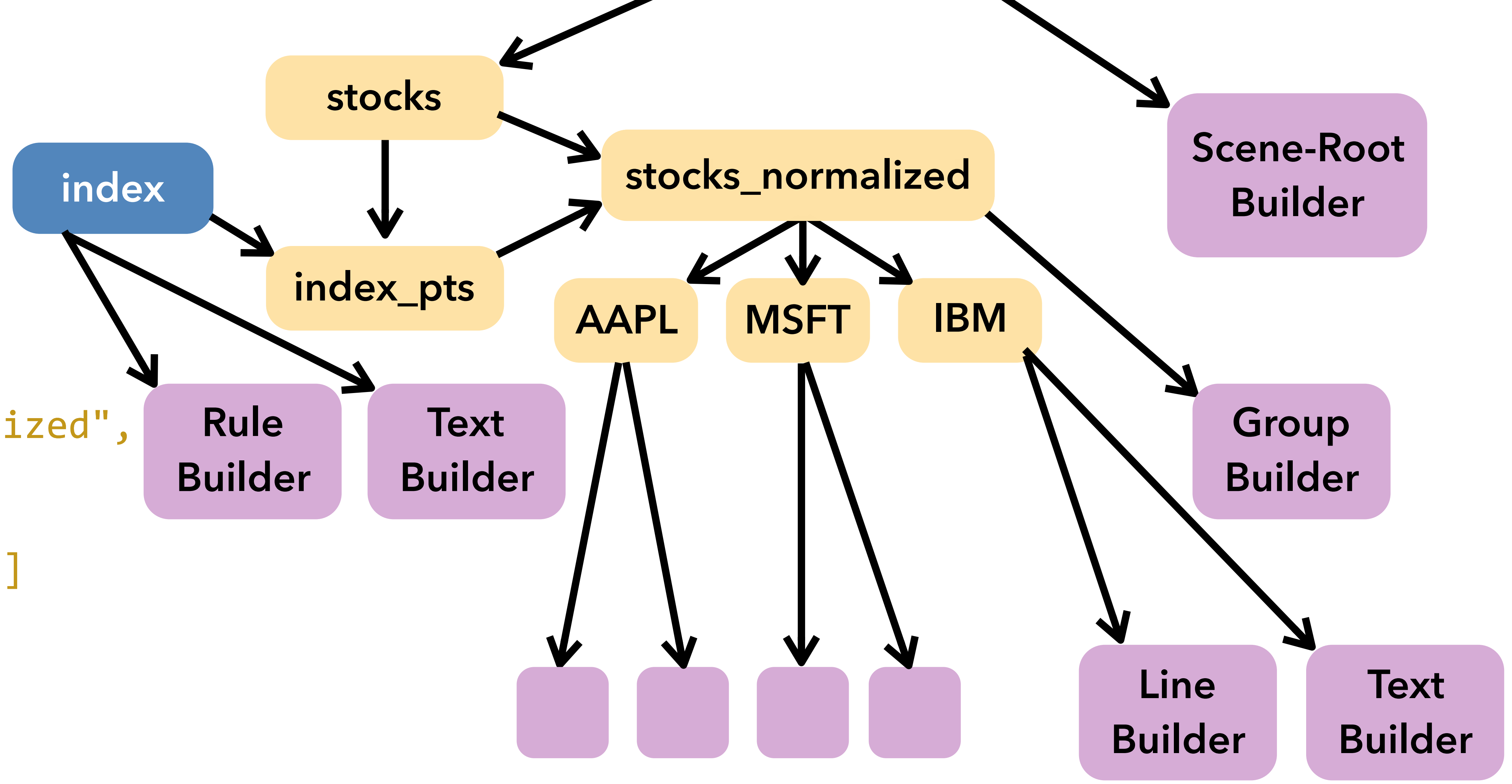
```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```

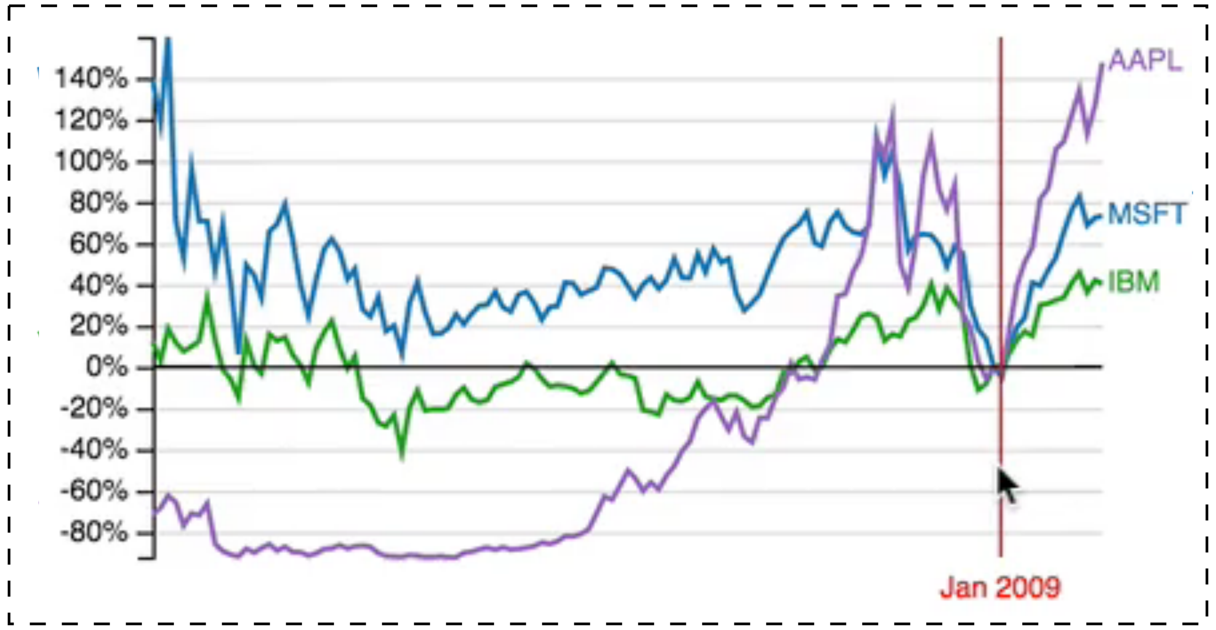




Run Time

```
{
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "stocks_normalized",
        "transform": [{
          "type": "facet",
          "groupby": ["symbol"]
        }]
      },

      "marks": [
        {"type": "line", ...},
        {"type": "text", ...}
      ]
    },
    {"type": "rule", ...},
    {"type": "text", ...},
  ]
}
```





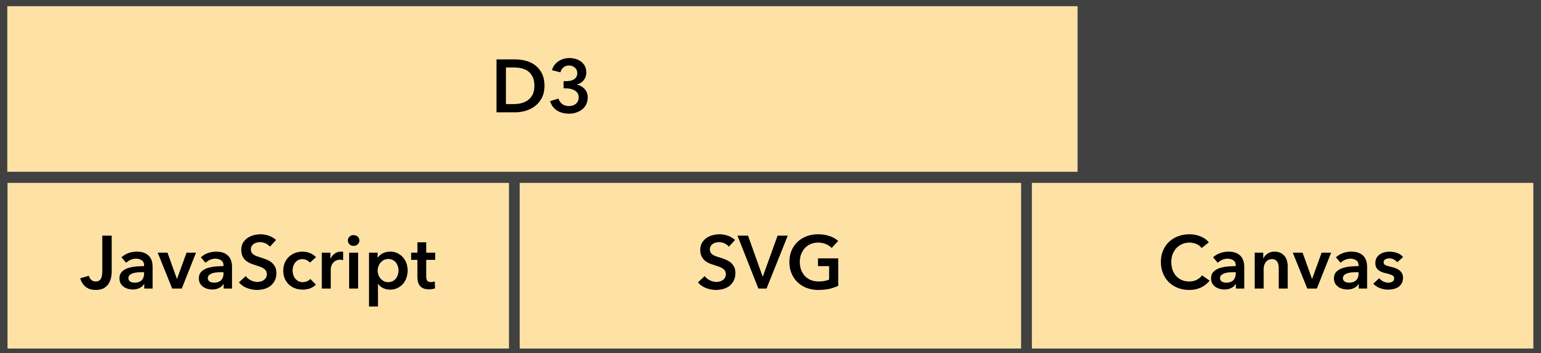
What about performance?

~2x faster than D3.

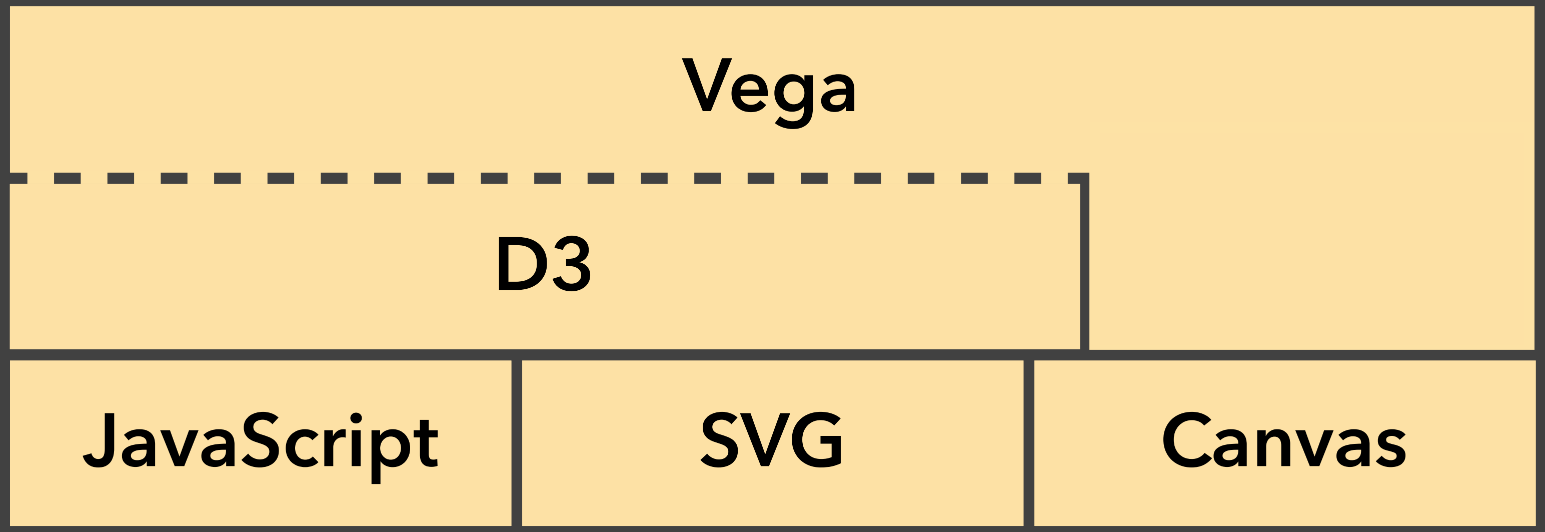
Full benchmark studies in the paper and online:
http://github.com/vega/vega-benchmarks

Future Work

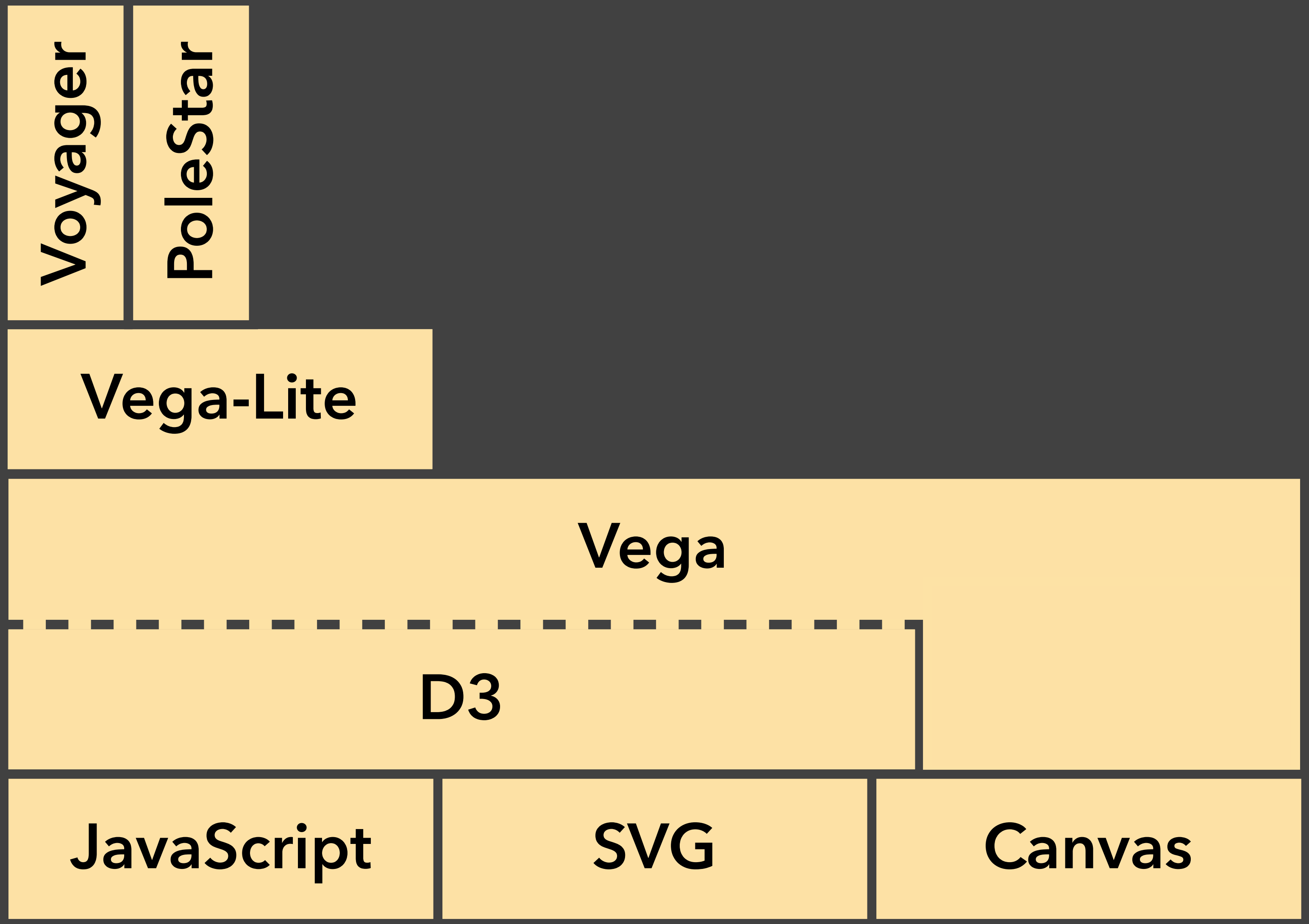
Future Work



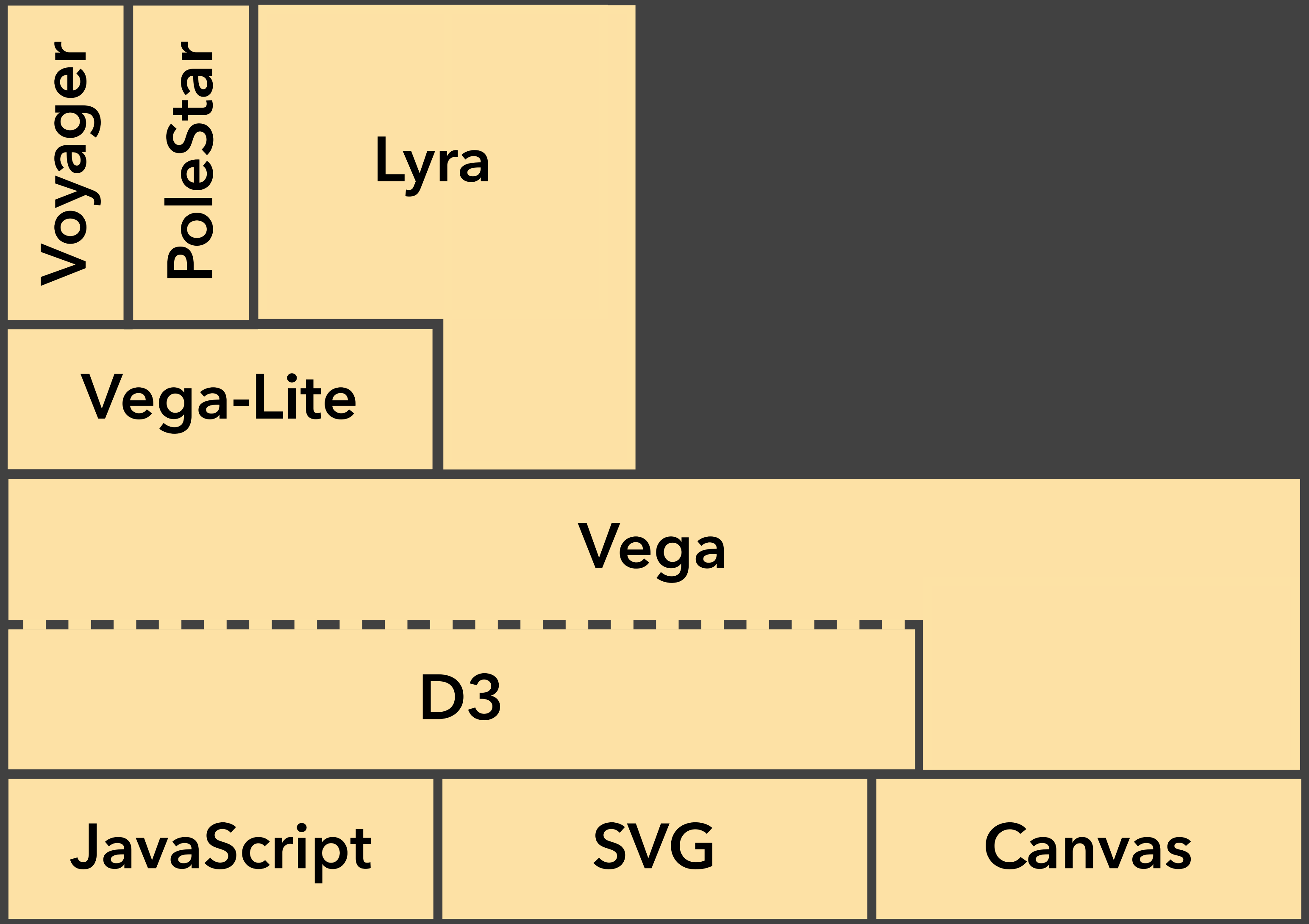
Future Work



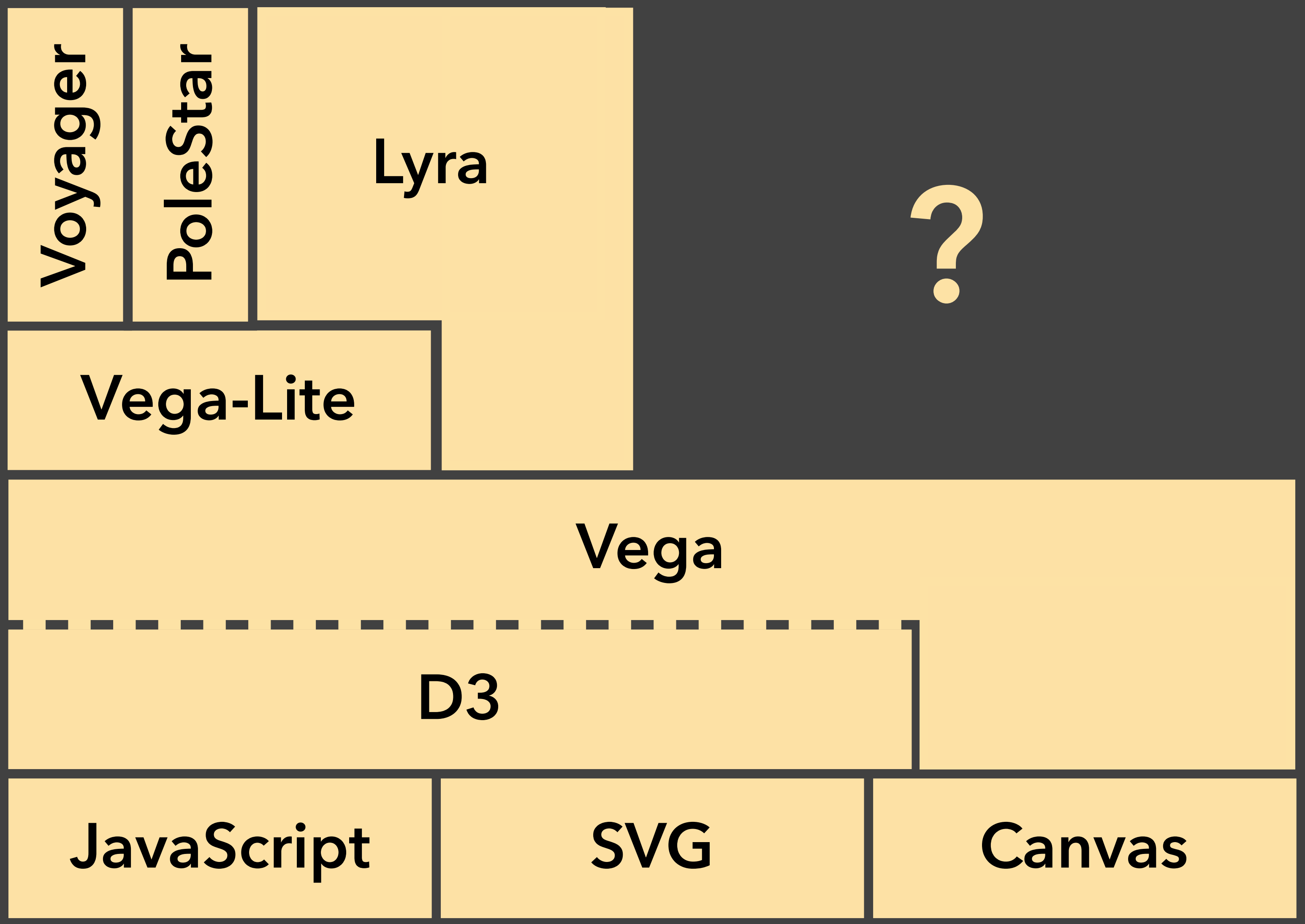
Future Work



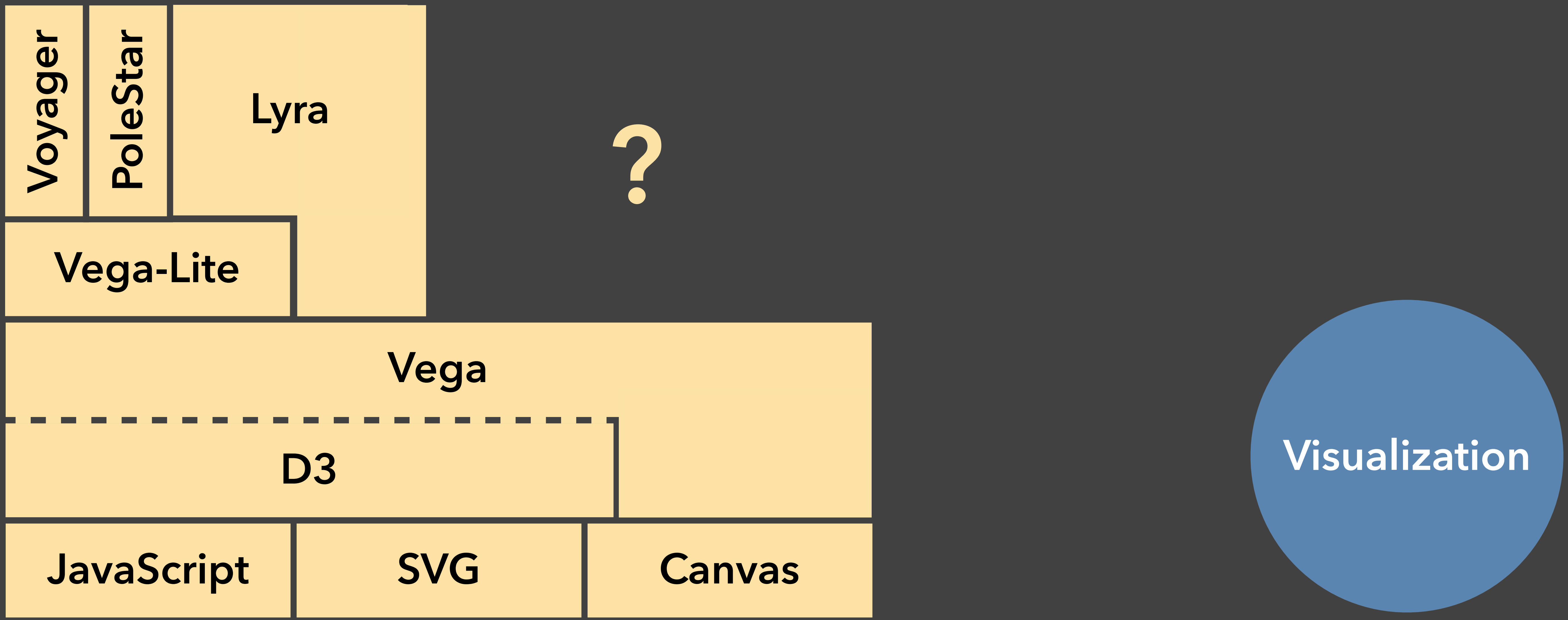
Future Work


Voyager
PoleStar
Lyra
Vega-Lite
Vega
D3
JavaScript
SVG
Canvas

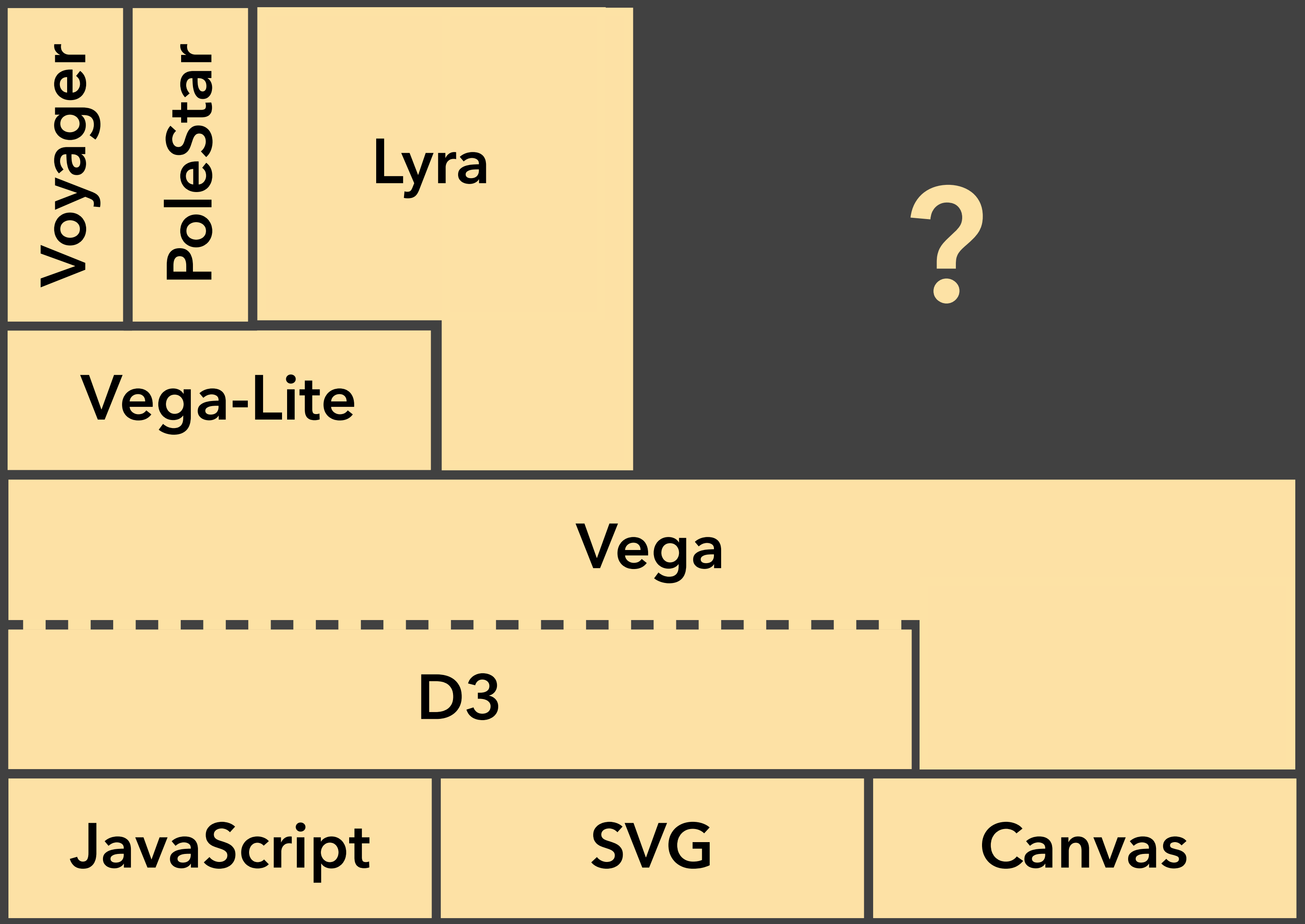
Future Work


Voyager
PoleStar
Lyra
?
Vega-Lite
Vega
D3
JavaScript
SVG
Canvas

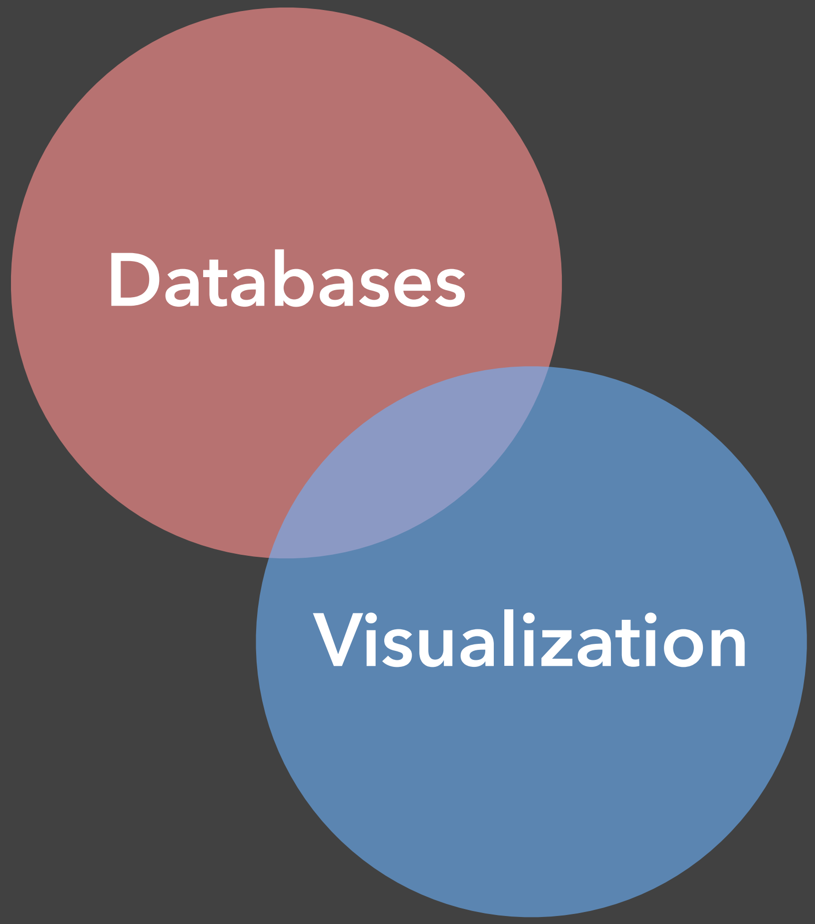
Future Work


Voyager
PoleStar
Lyra
?
Vega-Lite
Vega
D3
JavaScript
SVG
Canvas
Visualization

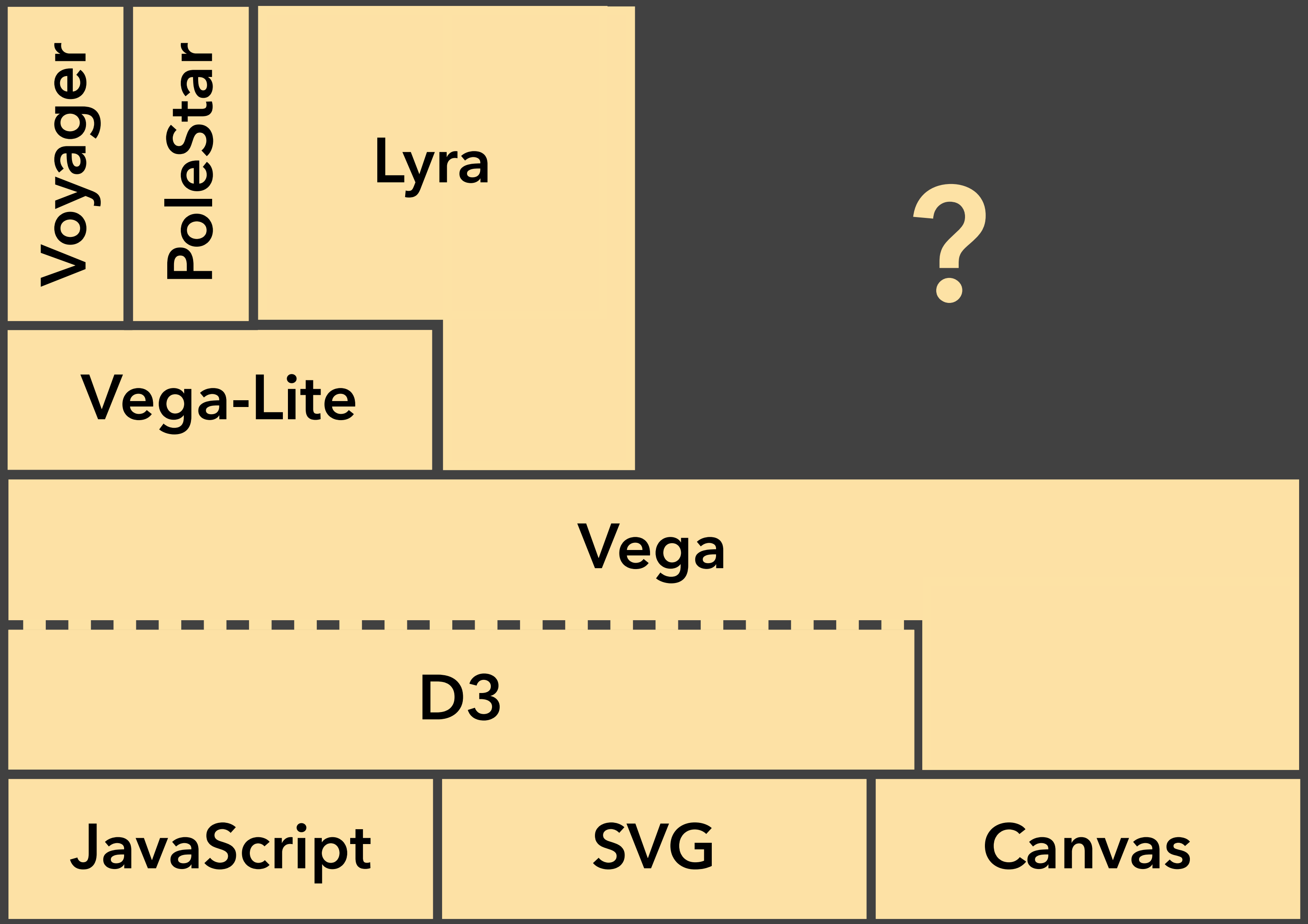
Future Work


Voyager
PoleStar
Lyra
Vega-Lite
Vega
D3
JavaScript
SVG
Canvas

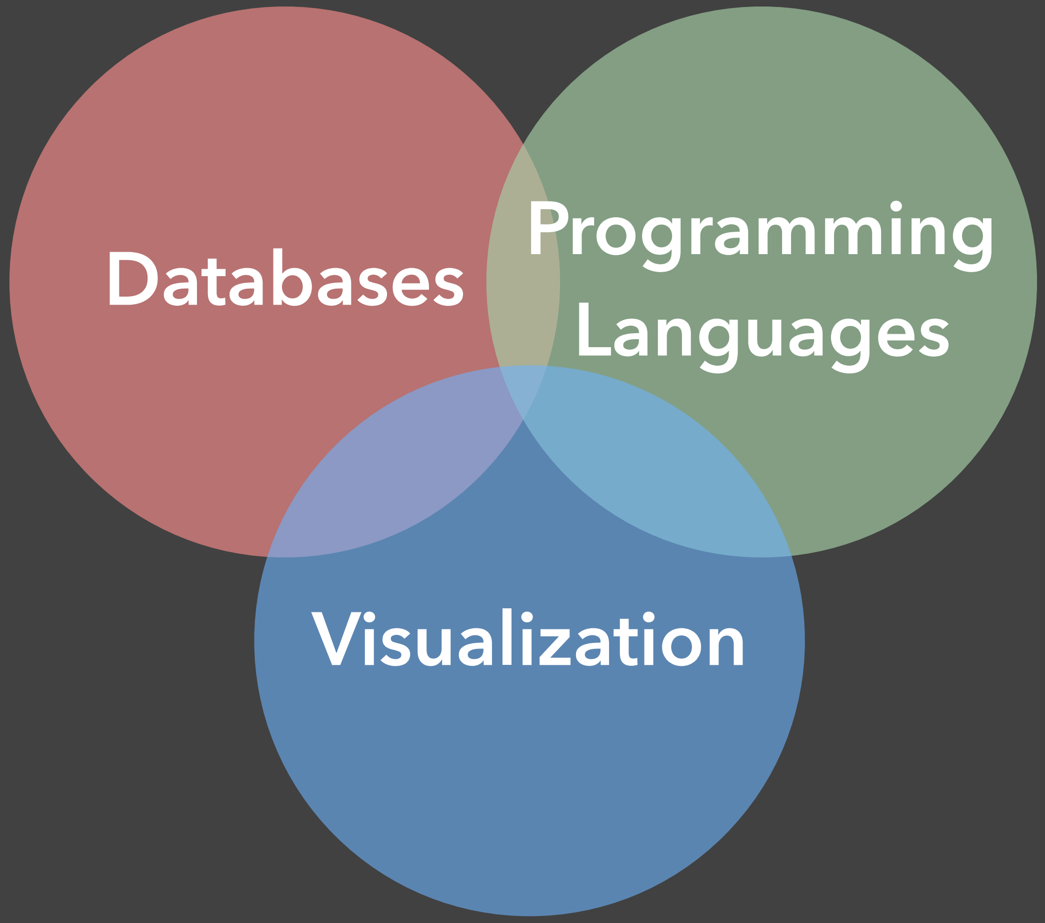
?


Databases
Visualization

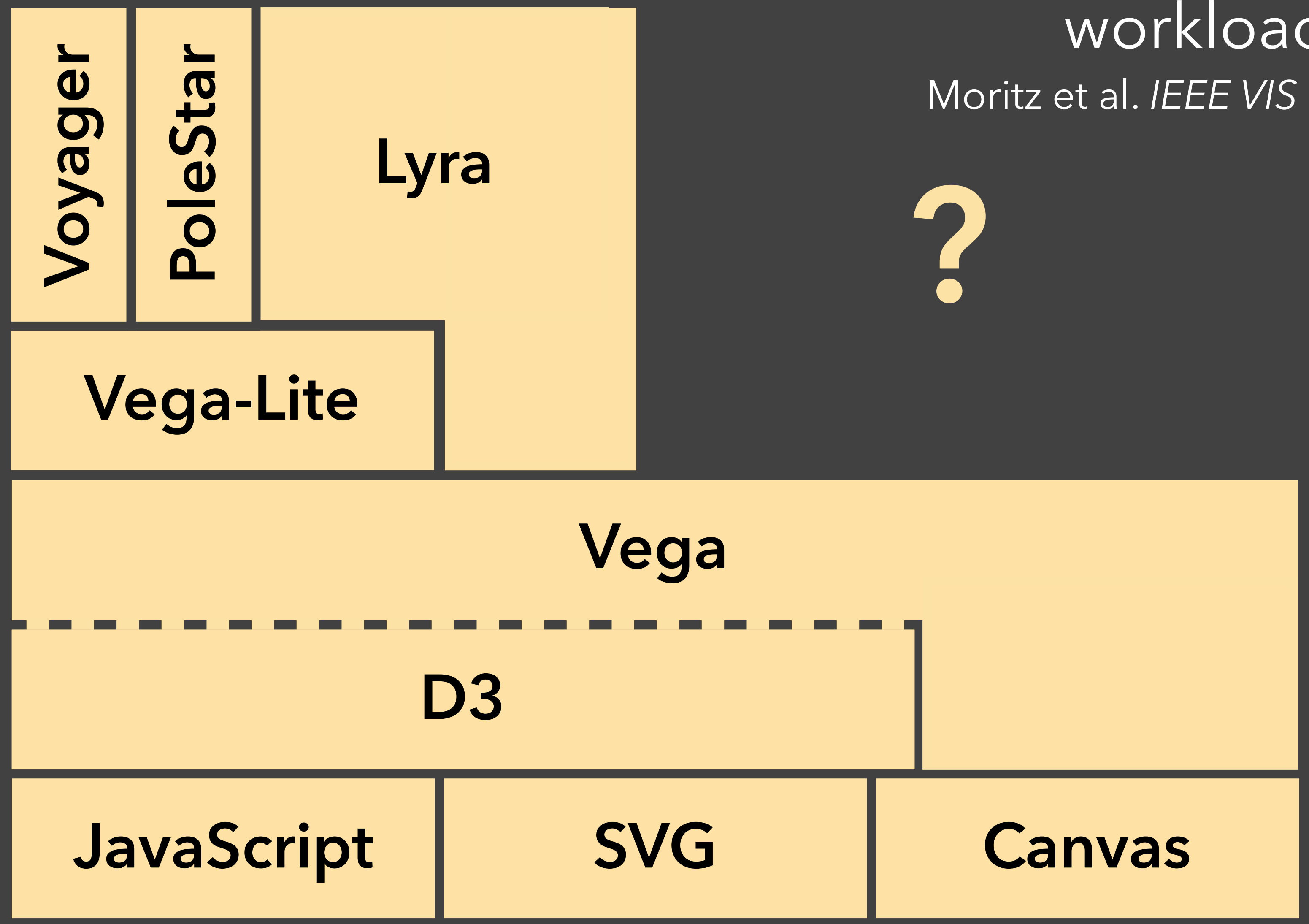
Future Work


Voyager
PoleStar
Lyra
Vega-Lite
Vega
D3
JavaScript
SVG
Canvas

?

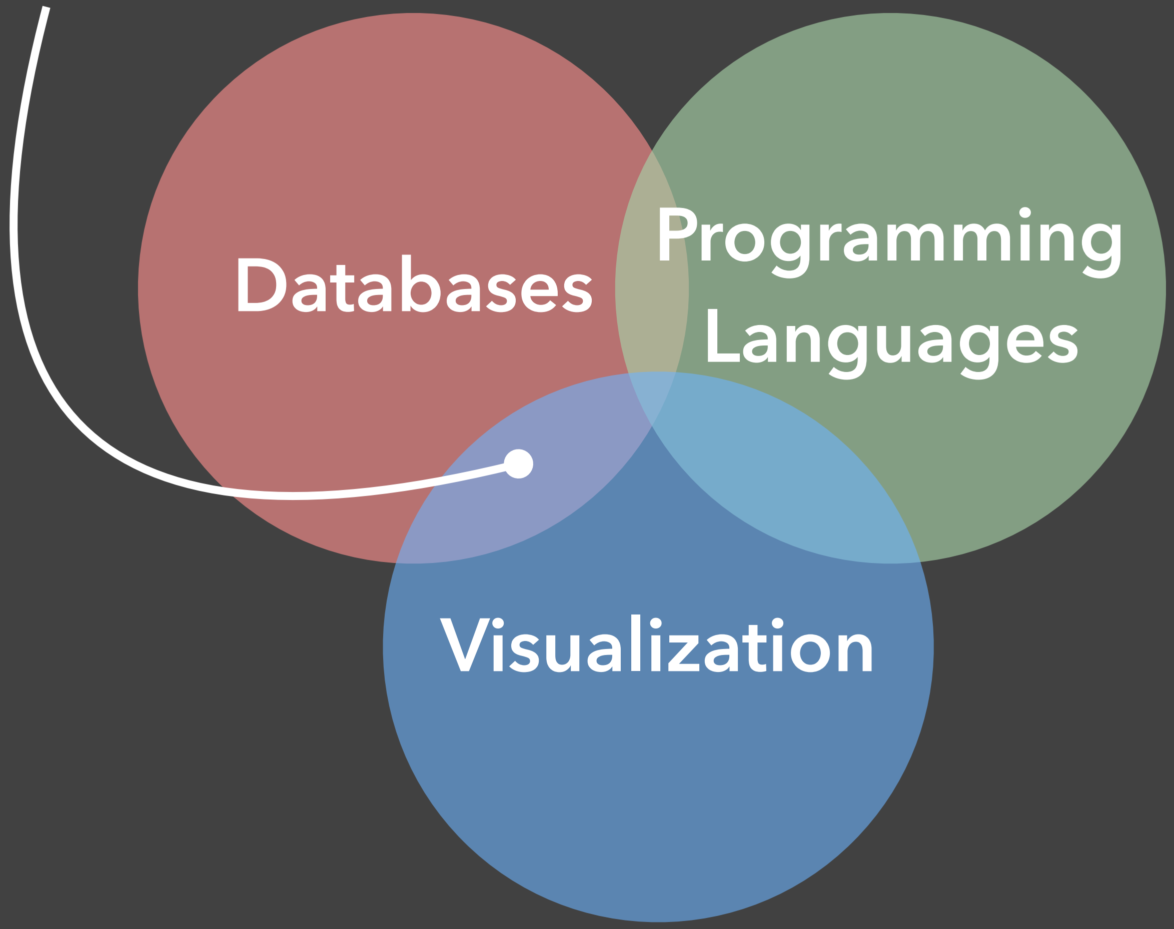

Databases
Programming Languages
Visualization

Future Work

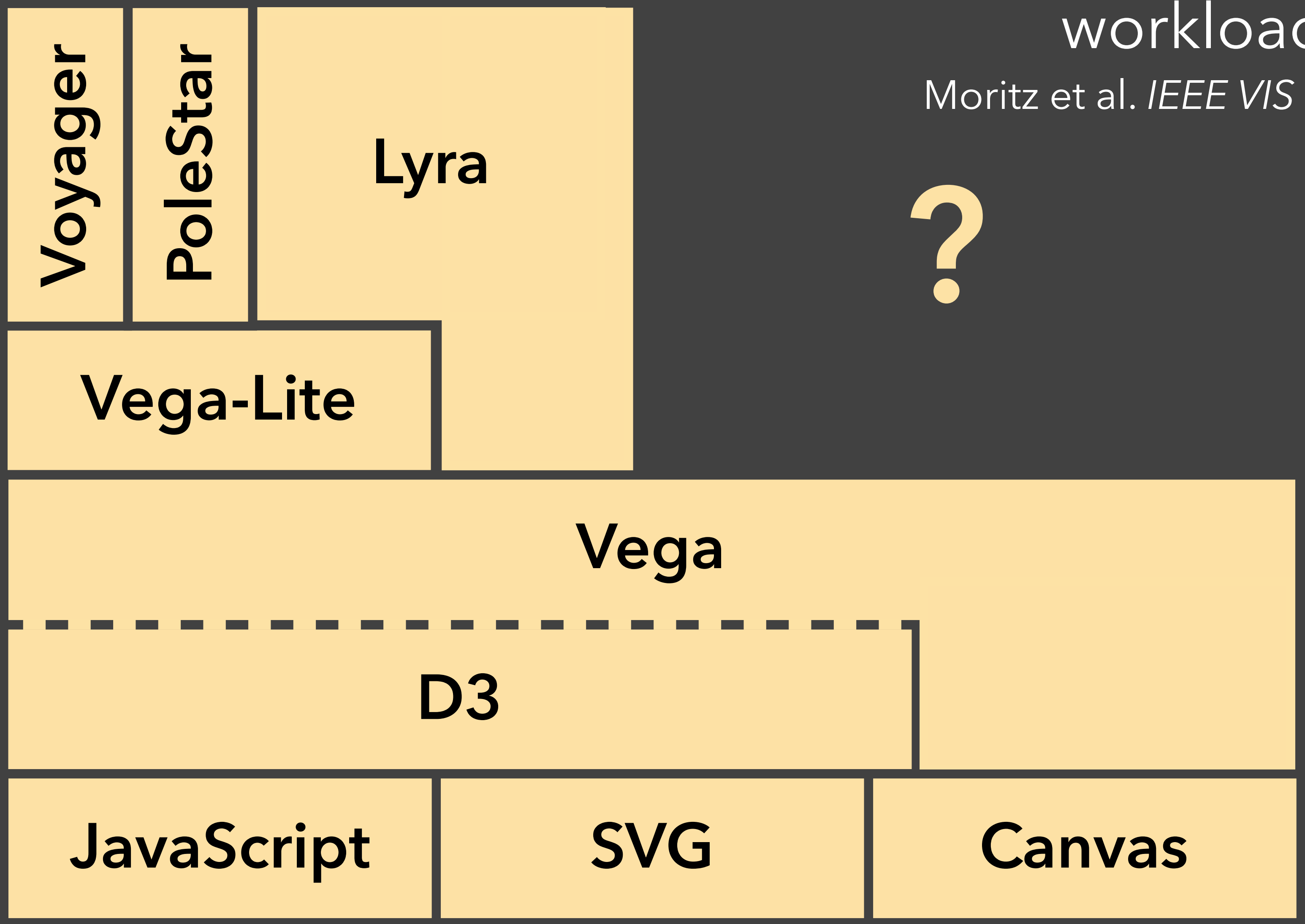


How can we **automatically partition** workloads between client and server?

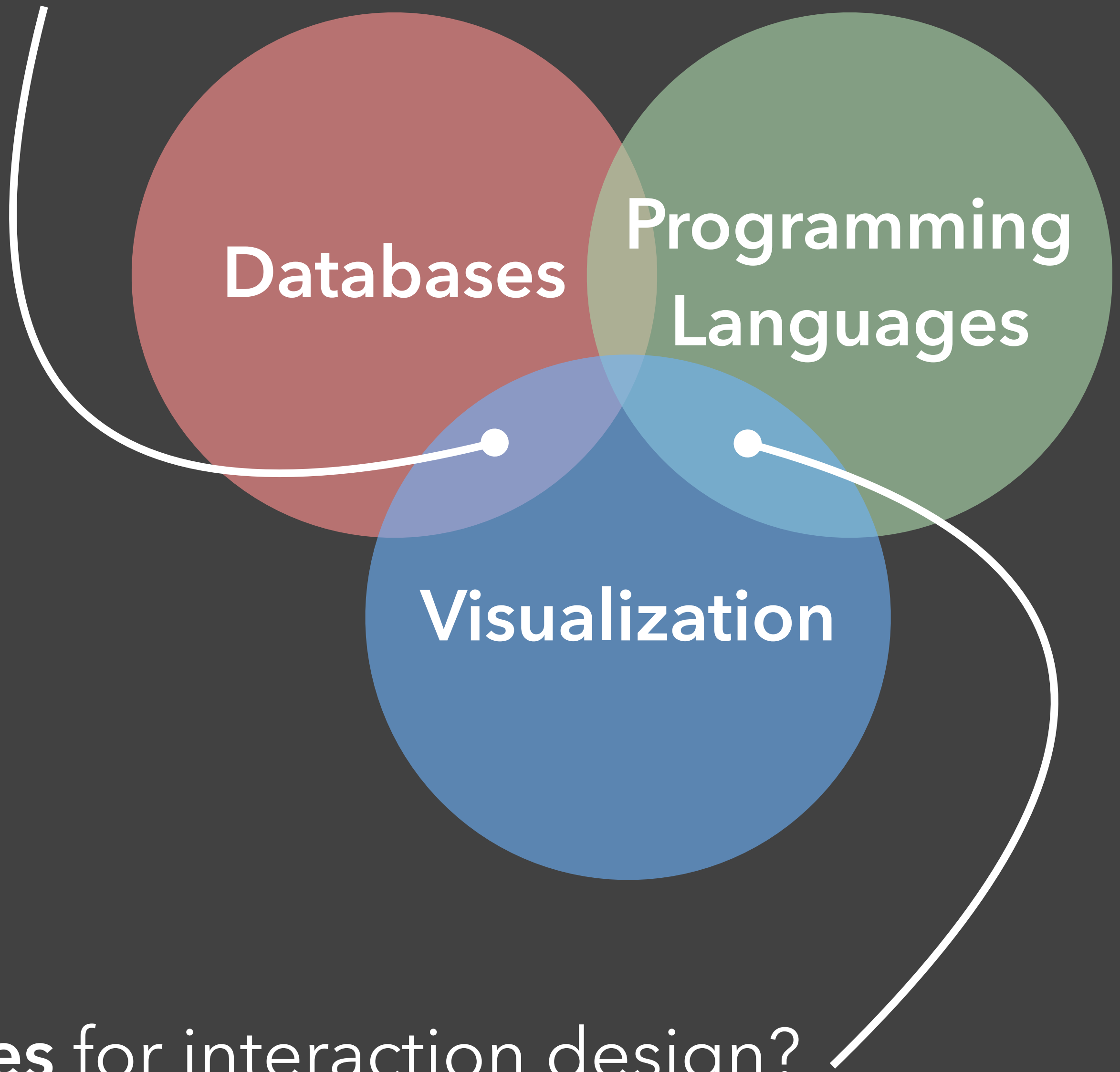
Moritz et al. *IEEE VIS Data Systems for Interactive Analysis Workshop 2015.*



Future Work

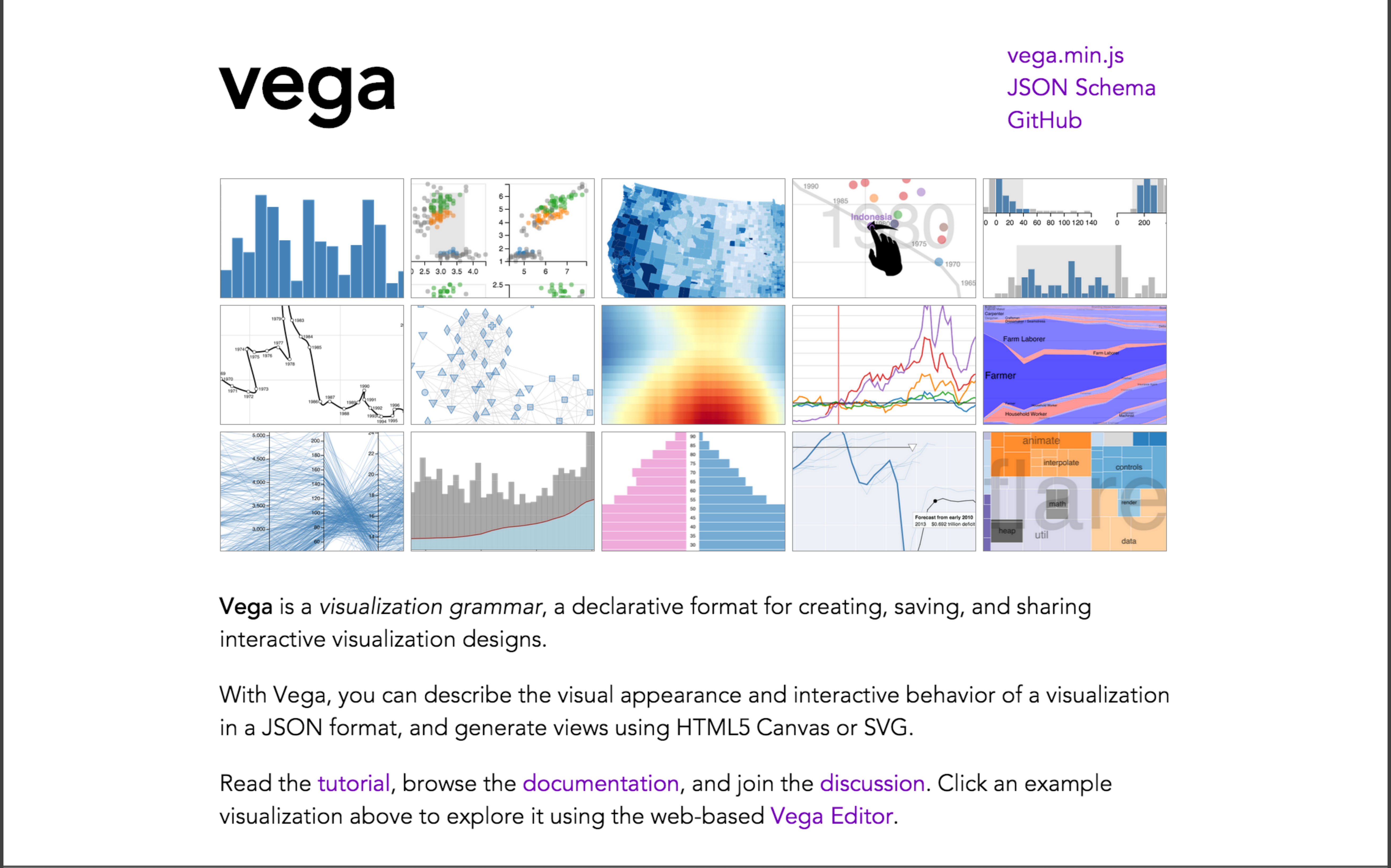


How can we **automatically partition** workloads between client and server?
Moritz et al. *IEEE VIS Data Systems for Interactive Analysis Workshop 2015.*



What are **higher-level languages** for interaction design?

Can **visual debugging tools** improve learnability of declarative specification?



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/