

This example has been archived.

This example is no longer maintained. You can still access the code below, but it may use an old version of D3, have bugs, lack comments, or otherwise not represent current best practices.

The D3 team now maintains [examples and tutorials](#) on [Observable](#), the best place to collaborate on visualization. Observable makes it easy to play with, fork, import, and share code on the web. You can download or embed whatever you make on Observable so it can live anywhere on the web (or off of it).

Observable uses dataflow, so you may not be able to copy-paste Observable code directly into vanilla JavaScript, but once you [know the differences](#) you can port if desired. Or [use it directly](#) in your application with Observable's lightweight open-source runtime. Either way, dataflow makes code more approachable and helps you spend more time “thinking with data” instead of programming. You can find thousands of notebooks on Observable to help you learn and get inspired.

[See this example on Observable.](#) →

[See my notebooks on Observable.](#) →

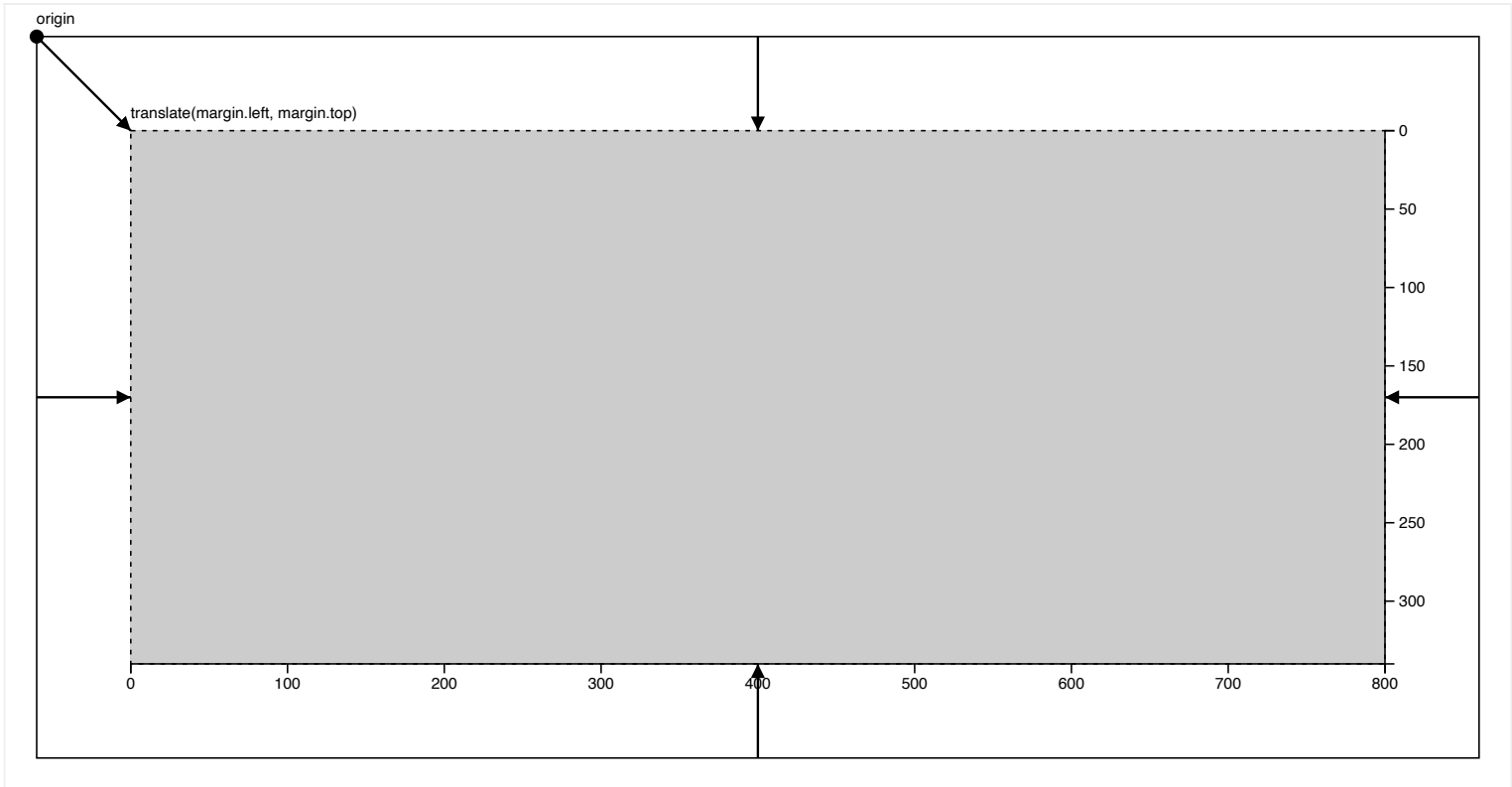
[See the D3 gallery.](#) →

[Hide this message.](#)

[Mike Bostock's Block 3019563](#)
Updated November 3, 2021

[Popular / About](#)

Margin Convention



First define the margin object with properties for the four sides (clockwise from the top, as in CSS).

[Open](#)

```
var margin = {top: 20, right: 10, bottom: 20, left: 10};
```

Then define width and height as the inner dimensions of the chart area.

```
var width = 960 - margin.left - margin.right,
    height = 500 - margin.top - margin.bottom;
```

Lastly, define svg as a G element that translates the origin to the top-left corner of the chart area.

```
var svg = d3.select("body").append("svg")
  .attr("width", width + margin.left + margin.right)
  .attr("height", height + margin.top + margin.bottom)
  .append("g")
  .attr("transform", "translate(" + margin.left + "," + margin.top + ")")
```

With this convention, all subsequent code can ignore margins. For example, to create x and y scales, simply say:

```
var x = d3.scale.linear()
  .range([0, width]);

var y = d3.scale.linear()
  .range([height, 0]);
```

If you want to add `axes` to the chart, they will be positioned correctly by default in the "left" and "top" orientations. For "right" or "bottom" orientation, translate the axis G element by the width or height, respectively.

index.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

body {
  font: 10px sans-serif;
}

.axis line,
.axis path {
  fill: none;
  stroke: #000;
  shape-rendering: crispEdges;
}

.arrow {
  stroke: #000;
  stroke-width: 1.5px;
}

.outer,
.inner {
  shape-rendering: crispEdges;
}

.outer {
  fill: none;
  stroke: #000;
}

.inner {
  fill: #ccc;
  stroke: #000;
  stroke-dasharray: 3, 4;
}

</style>
<body>
<script src="//d3js.org/d3.v3.min.js"></script>
<script>

var margin = {top: 20, right: 20, bottom: 20, left: 20},
    padding = {top: 60, right: 60, bottom: 60, left: 60},
    outerWidth = 960,
    outerHeight = 500,
    innerWidth = outerWidth - margin.left - margin.right,
    innerHeight = outerHeight - margin.top - margin.bottom,
    width = innerWidth - padding.left - padding.right,
    height = innerHeight - padding.top - padding.bottom;

var x = d3.scale.identity()
```

```
.domain([0, width]);

var y = d3.scale.identity()
  .domain([0, height]);

var xAxis = d3.svg.axis()
  .scale(x)
  .orient("bottom");

var yAxis = d3.svg.axis()
  .scale(y)
  .orient("right");

var svg = d3.select("body").append("svg")
  .attr("width", outerWidth)
  .attr("height", outerHeight)
  .append("g")
  .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

var defs = svg.append("defs");

defs.append("marker")
  .attr("id", "triangle-start")
  .attr("viewBox", "0 0 10 10")
  .attr("refX", 10)
  .attr("refY", 5)
  .attr("markerWidth", -6)
  .attr("markerHeight", 6)
  .attr("orient", "auto")
  .append("path")
  .attr("d", "M 0 0 L 10 5 L 0 10 z");

defs.append("marker")
  .attr("id", "triangle-end")
  .attr("viewBox", "0 0 10 10")
  .attr("refX", 10)
  .attr("refY", 5)
  .attr("markerWidth", 6)
  .attr("markerHeight", 6)
  .attr("orient", "auto")
  .append("path")
  .attr("d", "M 0 0 L 10 5 L 0 10 z");

svg.append("rect")
  .attr("class", "outer")
  .attr("width", innerWidth)
  .attr("height", innerHeight);

var g = svg.append("g")
  .attr("transform", "translate(" + padding.left + "," + padding.top + ")");

g.append("rect")
  .attr("class", "inner")
  .attr("width", width)
  .attr("height", height);

g.append("g")
  .attr("class", "x axis")
  .attr("transform", "translate(0," + height + ")")
  .call(xAxis);
```

```
g.append("g")
  .attr("class", "y axis")
  .attr("transform", "translate(" + width + ",0)")
  .call(yAxis);

svg.append("line")
  .attr("class", "arrow")
  .attr("x2", padding.left)
  .attr("y2", padding.top)
  .attr("marker-end", "url(#triangle-end)");

svg.append("line")
  .attr("class", "arrow")
  .attr("x1", innerWidth / 2)
  .attr("x2", innerWidth / 2)
  .attr("y2", padding.top)
  .attr("marker-end", "url(#triangle-end)");

svg.append("line")
  .attr("class", "arrow")
  .attr("x1", innerWidth / 2)
  .attr("x2", innerWidth / 2)
  .attr("y1", innerHeight - padding.bottom)
  .attr("y2", innerHeight)
  .attr("marker-start", "url(#triangle-start)");

svg.append("line")
  .attr("class", "arrow")
  .attr("x2", padding.left)
  .attr("y1", innerHeight / 2)
  .attr("y2", innerHeight / 2)
  .attr("marker-end", "url(#triangle-end)");

svg.append("line")
  .attr("class", "arrow")
  .attr("x1", innerWidth)
  .attr("x2", innerWidth - padding.right)
  .attr("y1", innerHeight / 2)
  .attr("y2", innerHeight / 2)
  .attr("marker-end", "url(#triangle-end)");

svg.append("text")
  .text("origin")
  .attr("y", -8);

svg.append("circle")
  .attr("class", "origin")
  .attr("r", 4.5);

g.append("text")
  .text("translate(margin.left, margin.top)")
  .attr("y", -8);

</script>
```

LICENSE

Released under the [GNU General Public License, version 3](#).