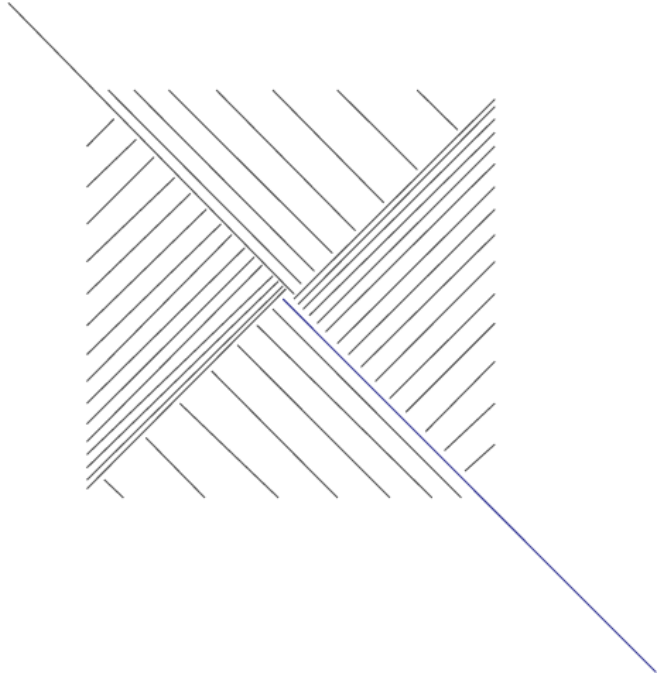


9

Vector Graphics in Raphael

A lightweight vector graphics library



Mystery solved!

The Ulam spiral

A means of visualizing prime numbers

The Ulam spiral

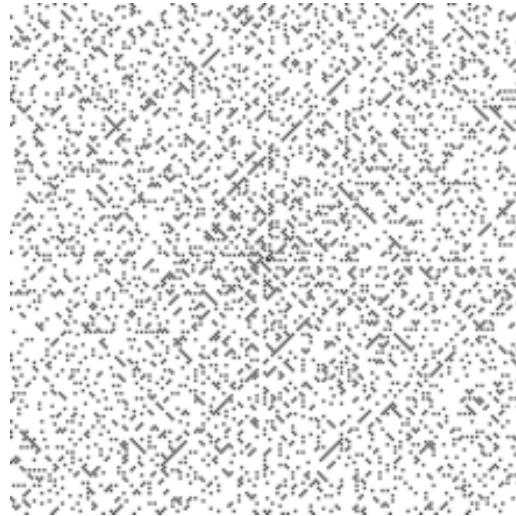
A means of visualizing prime numbers

```
37-36-35-34-33-32-31
|
38 17-16-15-14-13 30
|
39 18 5-4-3 12 29
|
40 19 6 1-2 11 28
|
41 20 7-8-9-10 27
|
42 21-22-23-24-25-26
|
43-44-45-46-47-48-49...
```

The Ulam spiral

A means of visualizing prime numbers

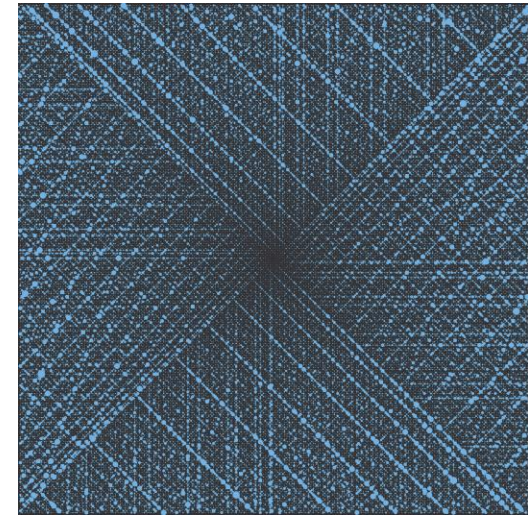
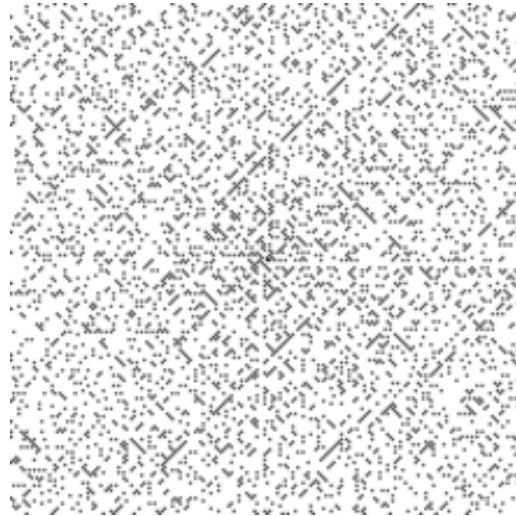
37	36	35	34	33	32	31
38	17	16	15	14	13	30
39	18	5	4	3	12	29
40	19	6	1	2	11	28
41	20	7	8	9	10	27
42	21	22	23	24	25	26
43	44	45	46	47	48	49...



The Ulam spiral

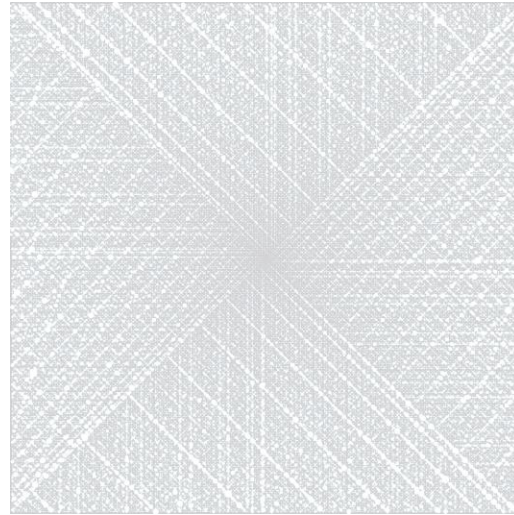
A means of visualizing prime numbers

37	36	35	34	33	32	31
38	17	16	15	14	13	30
39	18	5	4	3	12	29
40	19	6	1	2	11	28
41	20	7	8	9	10	27
42	21	22	23	24	25	26
43	44	45	46	47	48	49...



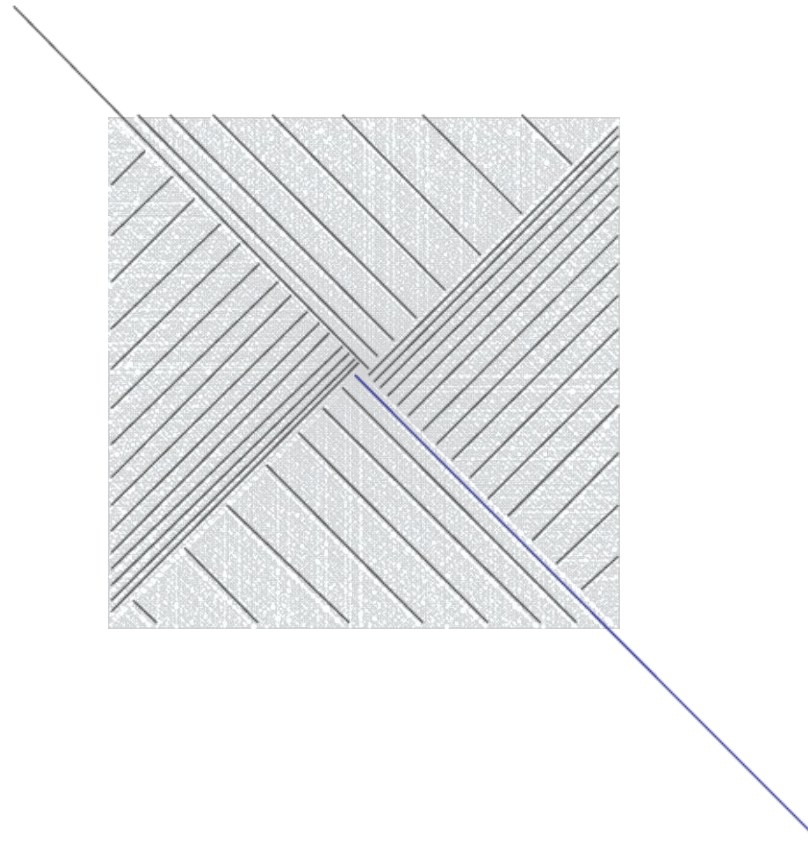
The Ulam spiral

A means of visualizing prime numbers



The Ulam spiral

A means of visualizing prime numbers



Congrats to **Matt Voda**



+



=



+



=





Raphael

A simple, lightweight JavaScript library for vector graphics creation and manipulation. Includes cool stuff such as interactivity and animation.

```
<!DOCTYPE html>  
<html>  
<head>  
  
</head>  
<body>  
  
</body>  
</html>
```

raphael_example.html

An ordinary HTML page

```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>

</body>
</html>
```

raphael_example.html

**TURBOCHARGED
WITH RAPHAEL!**

```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>

</body>
</html>
```

raphael_example.html

TURBOCHARGED WITH RAPHAEL!



```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>

</body>
</html>
```

raphael_example.html

```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>
  <script>

  </script>
</body>
</html>
```

raphael_example.html

Adding JavaScript tags
gives us a place to work


```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>
  <script>

  </script>
</body>
</html>
```

raphael_example.html

```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>
  <script>
    var paper = Raphael(10,10,600,600);

  </script>
</body>
</html>
```

raphael_example.html

The **paper** is defined as the region in which Raphael works its magic.

```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>
  <script>
    var paper = Raphael(10,10,600,600);
    var circle = paper.circle(50,50,10);

  </script>
</body>
</html>
```

raphael_example.html

The paper holds functions
that allow you to draw
SVG objects

```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>
  <script>
    var paper = Raphael(10,10,600,600);
    var circle = paper.circle(50,50,10);
    circle.attr("fill", "rgb(255,0,255)");
  </script>
</body>
</html>
```

raphael_example.html

Huzzah, and now you've changed the circle's color

```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>
  <script>
    var paper = Raphael(10,10,600,600);
    var circle = paper.circle(50,50,10);
    circle.attr("fill", "rgb(255,0,255)");
  </script>
</body>
</html>
```

raphael_example.html



```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>
  <script>
    var paper = Raphael(10,10,600,600);
    var circle = paper.circle(50,50,10);
    circle.attr("fill", "rgb(255,0,255)");
  </script>
</body>
</html>
```

raphael_example.html



AMAZING!

```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>
  <script>
    var paper = Raphael(10,10,600,600);
    var circle = paper.circle(50,50,10).attr("fill", "rgb(255,0,255)");
  </script>      (alternatively, put it all on one line!)
</body>
</html>
```

raphael_example.html



Of course, that's just part of the fun!

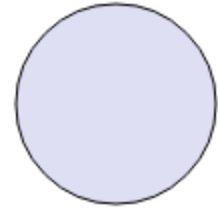
CIRCLE

CIRCLE

```
paper.circle(100,100,50).attr({  
    "fill": "rgb(100,100,200)",  
    "fill-opacity": 0.2  
});
```

CIRCLE

```
paper.circle(100,100,50).attr({  
  "fill": "rgb(100,100,200)",  
  "fill-opacity": 0.2  
});
```

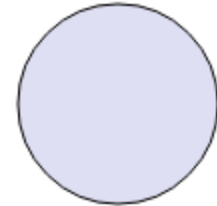


CIRCLE

x, y, radius



```
paper.circle(100,100,50).attr({  
  "fill": "rgb(100,100,200)",  
  "fill-opacity": 0.2  
});
```



Multiple attributes given to .attr
as a JSON object!

RECT

RECT

```
paper.rect(50,50,200,100).attr({  
  "fill": "none",  
  "stroke-width": 5,  
  "stroke-opacity": 0.1  
});
```

RECT

```
paper.rect(50,50,200,100).attr({  
  "fill": "none",  
  "stroke-width": 5,  
  "stroke-opacity": 0.1  
});
```



RECT

x, y, width, height



```
paper.rect(50,50,200,100).attr({  
  "fill": "none",  
  "stroke-width": 5,  
  "stroke-opacity": 0.1  
});
```



“none” is an acceptable value
as defined by the SVG specification

TEXT

TEXT

```
paper.text(150,50, "CS1501 is\nthe best!").attr({  
    "font-size": 20  
});
```

TEXT

```
paper.text(150,50, "CS1501 is\nthe best!").attr({  
    "font-size": 20  
});
```

CS1501 is
the best!

TEXT

x, y, text



```
paper.text(150,50, "CS1501 is\nthe best!").attr({  
  "font-size": 20  
});
```



Newline character
is accepted!

CS1501 is
the best!

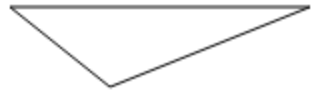
PATH

PATH

```
paper.path("M20,20L170,20L70,60Z");
```

PATH

```
paper.path("M20,20L170,20L70,60Z");
```



PATH

```
paper.path("M20,20L170,20L70,60Z");
```



Here is short list of commands available, for more details see [SVG path string format](#).

Command	Name	Parameters
M	moveto	(x y)+
Z	closepath	(none)
L	lineto	(x y)+
H	horizontal lineto	x+
V	vertical lineto	y+
C	curveto	(x1 y1 x2 y2 x y)+
S	smooth curveto	(x2 y2 x y)+
Q	quadratic Bézier curveto	(x1 y1 x y)+
T	smooth quadratic Bézier curveto	(x y)+
A	elliptical arc	(rx ry x-axis-rotation large-arc-flag sweep-flag x y)+
R	Catmull-Rom curveto *	x1 y1 (x y)+

* "Catmull-Rom curveto" is a not standard SVG command and added in 2.0 to make life easier. Note: there is a special case when path consist of just three commands: "M10,10R...z". In this case path will smoothly connects to its beginning.

PATH



```
paper.path("M20,20L170,20L70,60Z");
```



Here is short list of commands available, for more details see [SVG path string format](#).

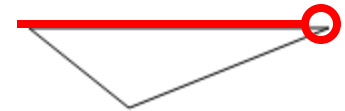
Command	Name	Parameters
M	moveto	(x y)+
Z	closepath	(none)
L	lineto	(x y)+
H	horizontal lineto	x+
V	vertical lineto	y+
C	curveto	(x1 y1 x2 y2 x y)+
S	smooth curveto	(x2 y2 x y)+
Q	quadratic Bézier curveto	(x1 y1 x y)+
T	smooth quadratic Bézier curveto	(x y)+
A	elliptical arc	(rx ry x-axis-rotation large-arc-flag sweep-flag x y)+
R	Catmull-Rom curveto *	x1 y1 (x y)+

* "Catmull-Rom curveto" is a not standard SVG command and added in 2.0 to make life easier. Note: there is a special case when path consist of just three commands: "M10,10R...z". In this case path will smoothly connects to its beginning.

PATH



```
paper.path("M20,20L170,20L70,60Z");
```



Here is short list of commands available, for more details see [SVG path string format](#).

Command	Name	Parameters
M	moveto	(x y)+
Z	closepath	(none)
L	lineto	(x y)+
H	horizontal lineto	x+
V	vertical lineto	y+
C	curveto	(x1 y1 x2 y2 x y)+
S	smooth curveto	(x2 y2 x y)+
Q	quadratic Bézier curveto	(x1 y1 x y)+
T	smooth quadratic Bézier curveto	(x y)+
A	elliptical arc	(rx ry x-axis-rotation large-arc-flag sweep-flag x y)+
R	Catmull-Rom curveto *	x1 y1 (x y)+

* "Catmull-Rom curveto" is a not standard SVG command and added in 2.0 to make life easier. Note: there is a special case when path consist of just three commands: "M10,10R...z". In this case path will smoothly connects to its beginning.

PATH

paper.path("M20,20L170,20L70,60Z");



Here is short list of commands available, for more details see [SVG path string format](#).

Command	Name	Parameters
M	moveto	(x y)+
Z	closepath	(none)
L	lineto	(x y)+
H	horizontal lineto	x+
V	vertical lineto	y+
C	curveto	(x1 y1 x2 y2 x y)+
S	smooth curveto	(x2 y2 x y)+
Q	quadratic Bézier curveto	(x1 y1 x y)+
T	smooth quadratic Bézier curveto	(x y)+
A	elliptical arc	(rx ry x-axis-rotation large-arc-flag sweep-flag x y)+
R	Catmull-Rom curveto *	x1 y1 (x y)+

* "Catmull-Rom curveto" is a not standard SVG command and added in 2.0 to make life easier. Note: there is a special case when path consist of just three commands: "M10,10R...z". In this case path will smoothly connects to its beginning.

PATH

paper.path("M20,20L170,20L70,60Z");



Here is short list of commands available, for more details see [SVG path string format](#).

Command	Name	Parameters
M	moveto	(x y)+
Z	closepath	(none)
L	lineto	(x y)+
H	horizontal lineto	x+
V	vertical lineto	y+
C	curveto	(x1 y1 x2 y2 x y)+
S	smooth curveto	(x2 y2 x y)+
Q	quadratic Bézier curveto	(x1 y1 x y)+
T	smooth quadratic Bézier curveto	(x y)+
A	elliptical arc	(rx ry x-axis-rotation large-arc-flag sweep-flag x y)+
R	Catmull-Rom curveto *	x1 y1 (x y)+

* "Catmull-Rom curveto" is a not standard SVG command and added in 2.0 to make life easier. Note: there is a special case when path consist of just three commands: "M10,10R...z". In this case path will smoothly connects to its beginning.

PATH

```
paper.path("M20,20L170,20L70,60Z");
```



<http://raphaeljs.com/reference.html#Paper.path>

Here is short list of commands available, for more details see [SVG path string format](#).

Command	Name	Parameters
M	moveto	(x y)+
Z	closepath	(none)
L	lineto	(x y)+
H	horizontal lineto	x+
V	vertical lineto	y+
C	curveto	(x1 y1 x2 y2 x y)+
S	smooth curveto	(x2 y2 x y)+
Q	quadratic Bézier curveto	(x1 y1 x y)+
T	smooth quadratic Bézier curveto	(x y)+
A	elliptical arc	(rx ry x-axis-rotation large-arc-flag sweep-flag x y)+
R	Catmull-Rom curveto *	x1 y1 (x y)+

* "Catmull-Rom curveto" is a not standard SVG command and added in 2.0 to make life easier. Note: there is a special case when path consist of just three commands: "M10,10R...z". In this case path will smoothly connects to its beginning.



Raphaël—JavaScript Library

? What is it?

Raphaël is a small JavaScript library that should simplify your work with vector graphics on the web. If you want to create your own specific chart or image crop and rotate widget, for example, you can achieve it simply and easily with this

**Most definitely consult the
Raphael documentation
if you get stuck or curious**

Compatible cross-browser and easy.

Raphaël currently supports Firefox 3.0+, Safari 3.0+, Chrome 5.0+, Opera 9.5+ and Internet Explorer 6.0+.



[Download v. 2.1.2 \(91 Kb\)](#)

Our recommendation is to GZIP it. It will help to reduce file size. You can download [uncompressed source \(299 Kb\)](#) as well.



[Documentation](#)



[Discussion Group](#)



[Twitter](#)

Raphaël Reference

Animation

Don't be a fool, use the search bar to find what you need!

Animation.delay(delay)

Creates a copy of existing animation object with given delay.

Parameters

delay number number of ms to pass between animation start and actual animation

Returns: object new altered Animation object

```
var anim = Raphael.animation({cx: 10, cy: 20}, 2e3);
circle1.animate(anim); // run the given animation immediately
circle2.animate(anim.delay(500)); // run the given animation after 500 ms
```

- Animation
 - Animation.delay()
 - Animation.repeat()
- Element
 - Element.animate()
 - Element.animateWith()
 - Element.attr()
 - Element.click()
 - Element.clone()
 - Element.data()
 - Element.dbclick()
 - Element.drag()
 - Element.getBBox()
 - Element.getPointAtLength()
 - Element.getSubpath()
 - Element.getTotalLength()
 - Element.glow()

Time for something a little more practical.

Time for something a little more practical.
(this is the part where things start to get complicated)

Imagine we have a dataset

```
var dataset = [50, 80, 35, 41];
```

Imagine we have a dataset

```
var dataset = [50, 80, 35, 41];
```

We want to visualize this dataset

(A bar chart will work just fine)

Imagine we have a dataset

```
var dataset = [50, 80, 35, 41];
```

We want to visualize this dataset

(A bar chart will work just fine)

Fortunately we know *just* enough JavaScript

Bear with me

Let's begin with our blank Raphael ready webpage

```
<!DOCTYPE html>  
<html>  
<head>  
  <script src="raphael-min.js" type="text/javascript"></script>  
</head>  
<body>  
  <script>  
  
  </script>  
</body>  
</html>
```

Let's begin with our blank Raphael ready webpage

```
<!DOCTYPE html>
<html>
<head>
  <script src="raphael-min.js" type="text/javascript"></script>
</head>
<body>
  <script>
    Great, now note that everything we
    do falls within these <script> tags
  </script>
</body>
</html>
```

Let's do something fun with data!

```
<script>
```

```
</script>
```

Let's do something fun with data!

```
<script>
```

```
var paper = Raphael(10,10,600,600);
```

```
var dataset = [50, 80, 35, 41];
```

Create the paper
Create the dataset

```
</script>
```


Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];

for(var i = 0; i < dataset.length; i++) {
}

```

Iterate through the dataset
We're going to make a bar chart

```
</script>
```

Let's do something fun with data!

```
<script>
```

```
var paper = Raphael(10,10,600,600);
```

```
var dataset = [50, 80, 35, 41];
```

```
var bars = [];
```

```
var labels = [];
```

Create arrays to hold the x number
of bars our chart will hold.

```
for(var i = 0; i < dataset.length; i++) {
```

```
}
```

```
</script>
```

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

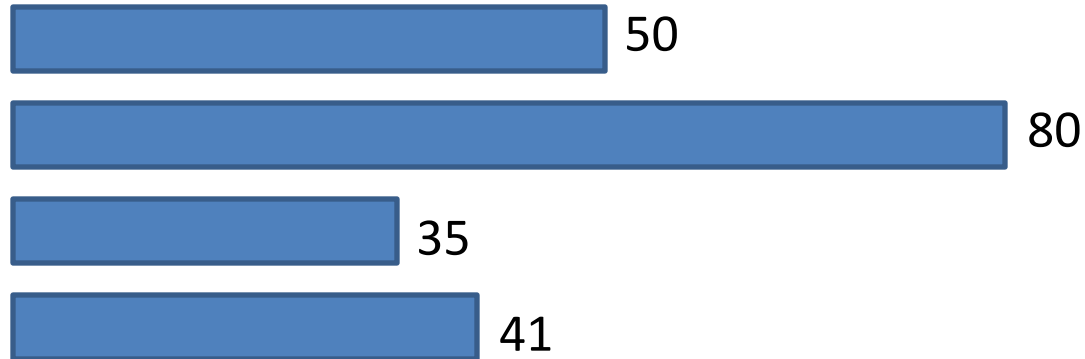
for(var i = 0; i < dataset.length; i++) {

}

</script>
```

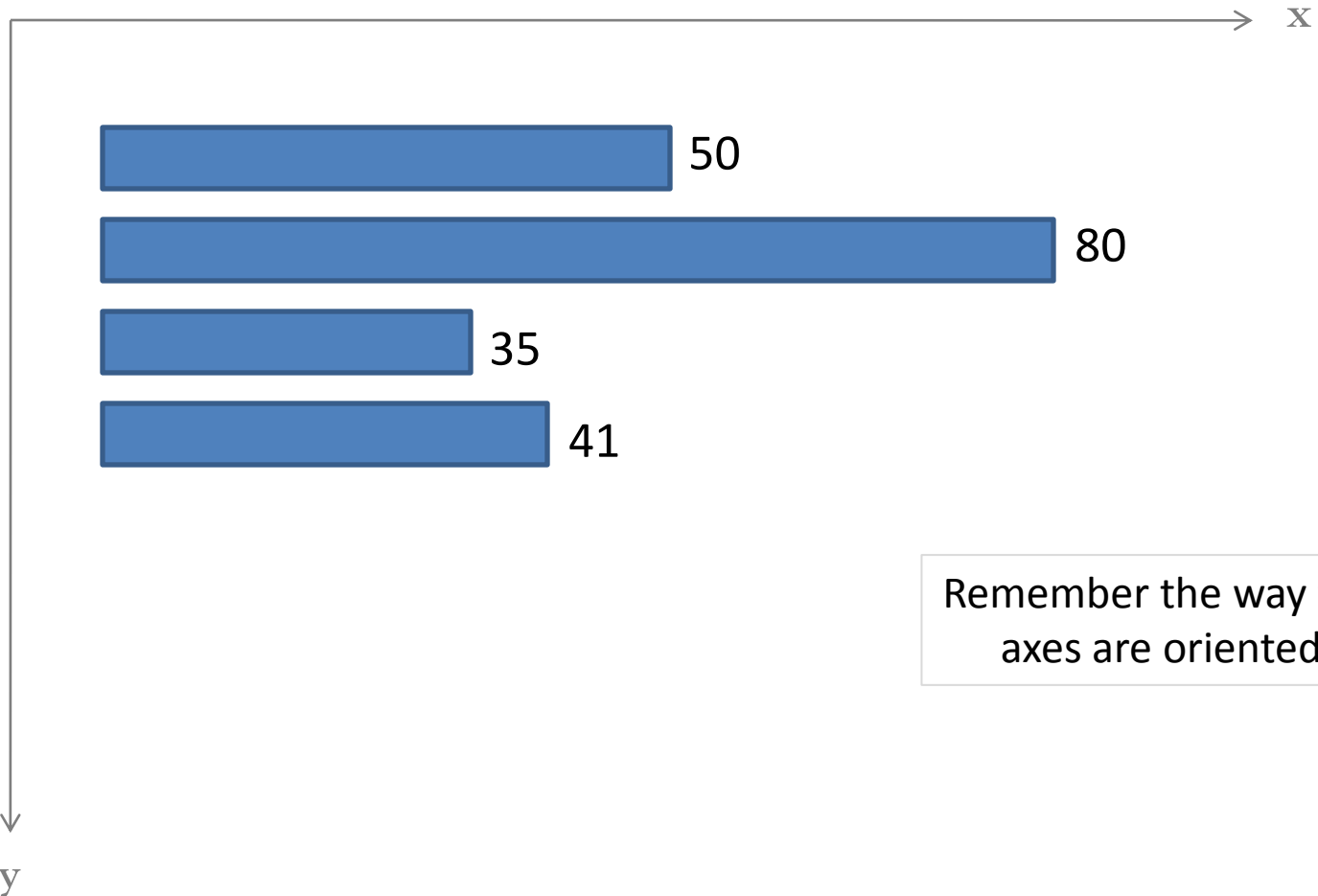
Now begin to think, how
should a bar chart be drawn?
Think **procedurally!**

Let's do something fun with data!

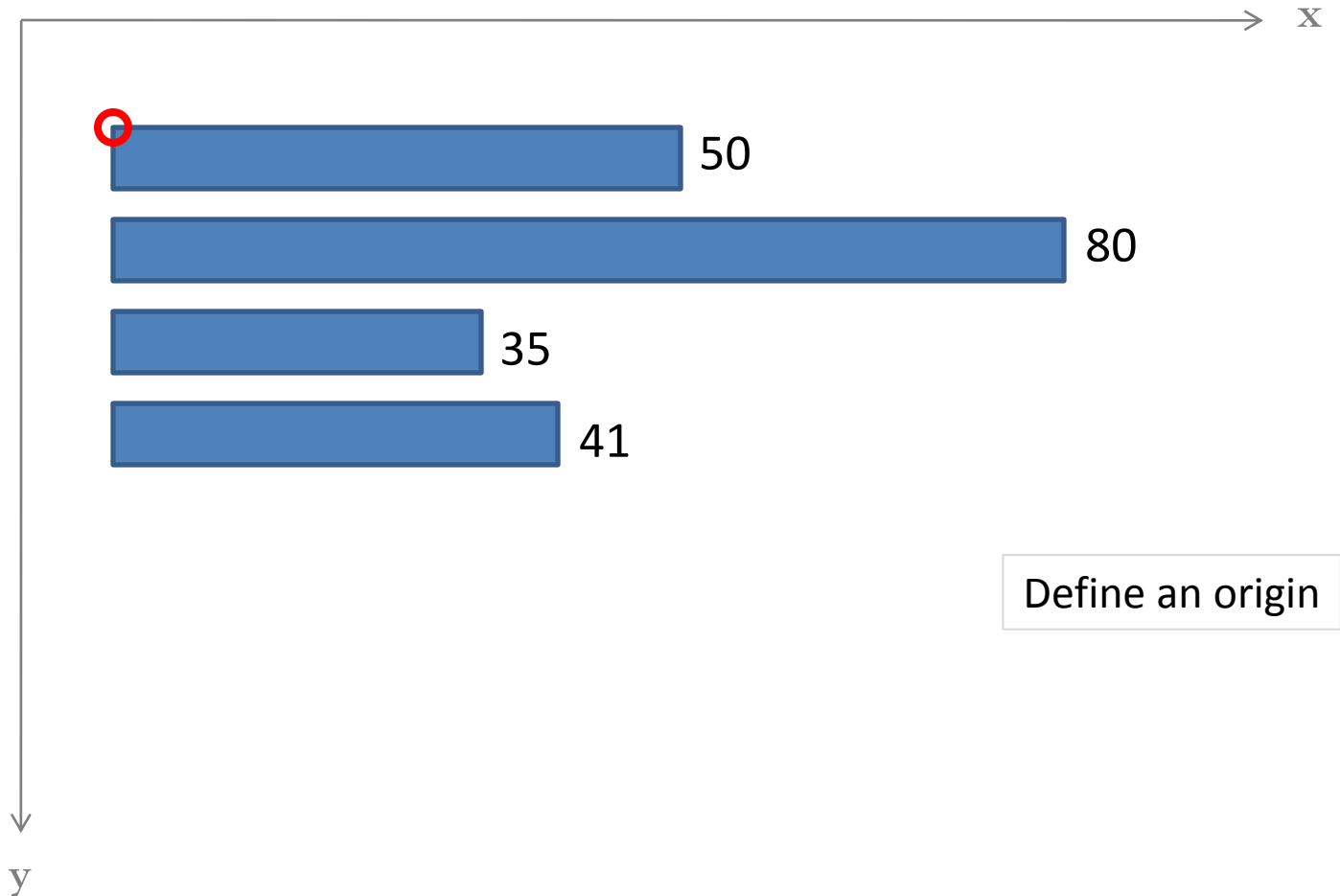


Here's an example bar chart.
We need to think about how it
is represented mathematically.

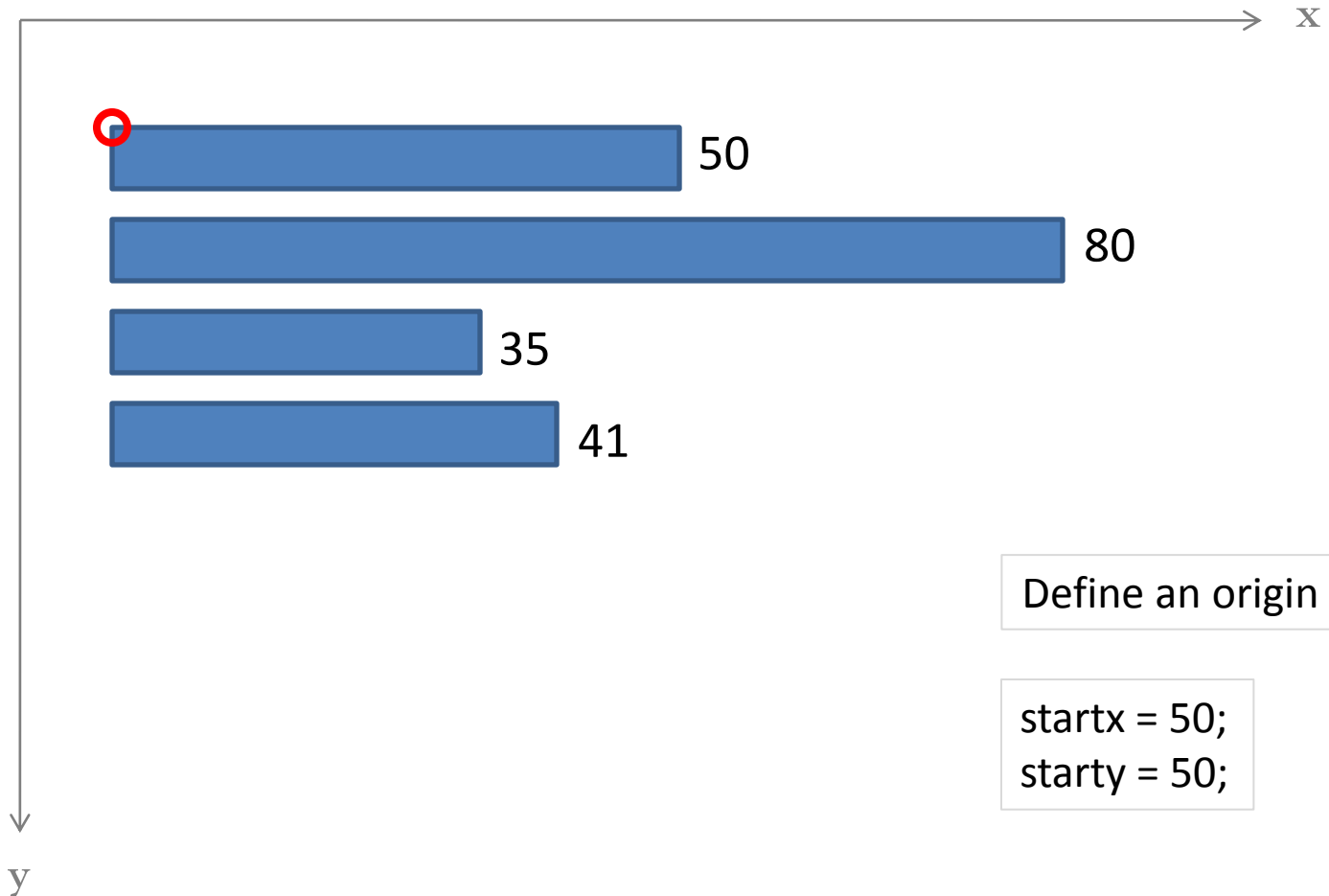
Let's do something fun with data!



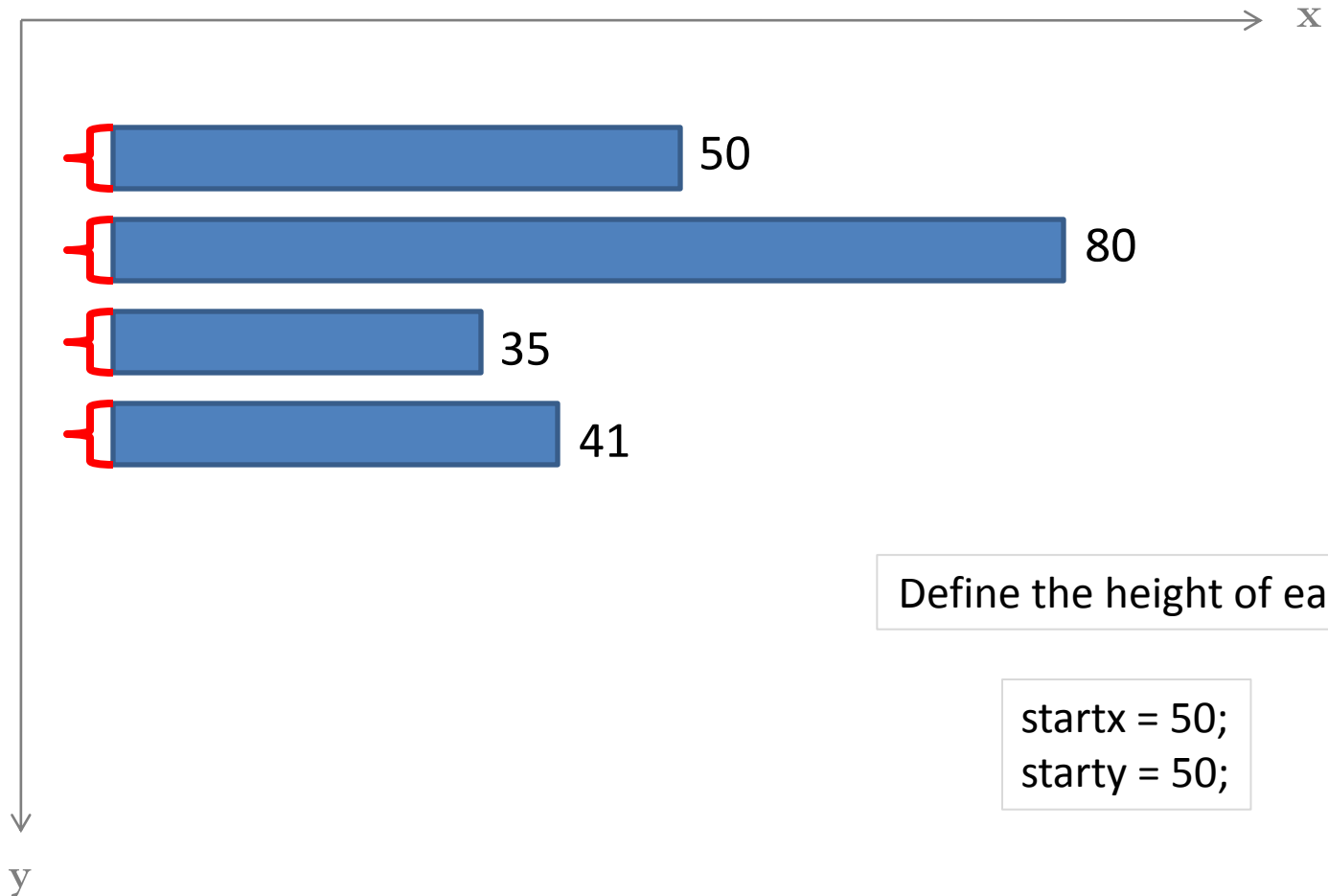
Let's do something fun with data!



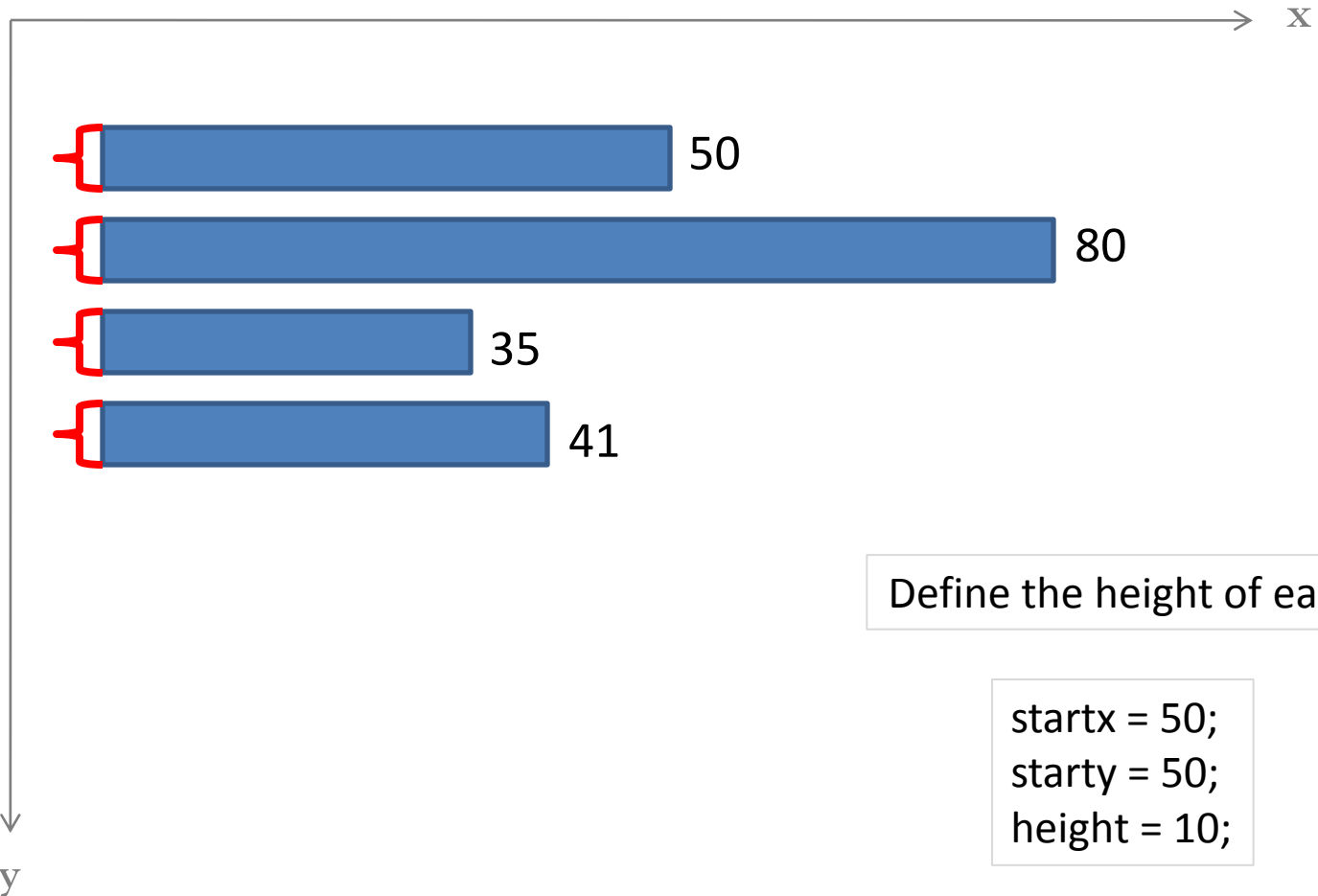
Let's do something fun with data!



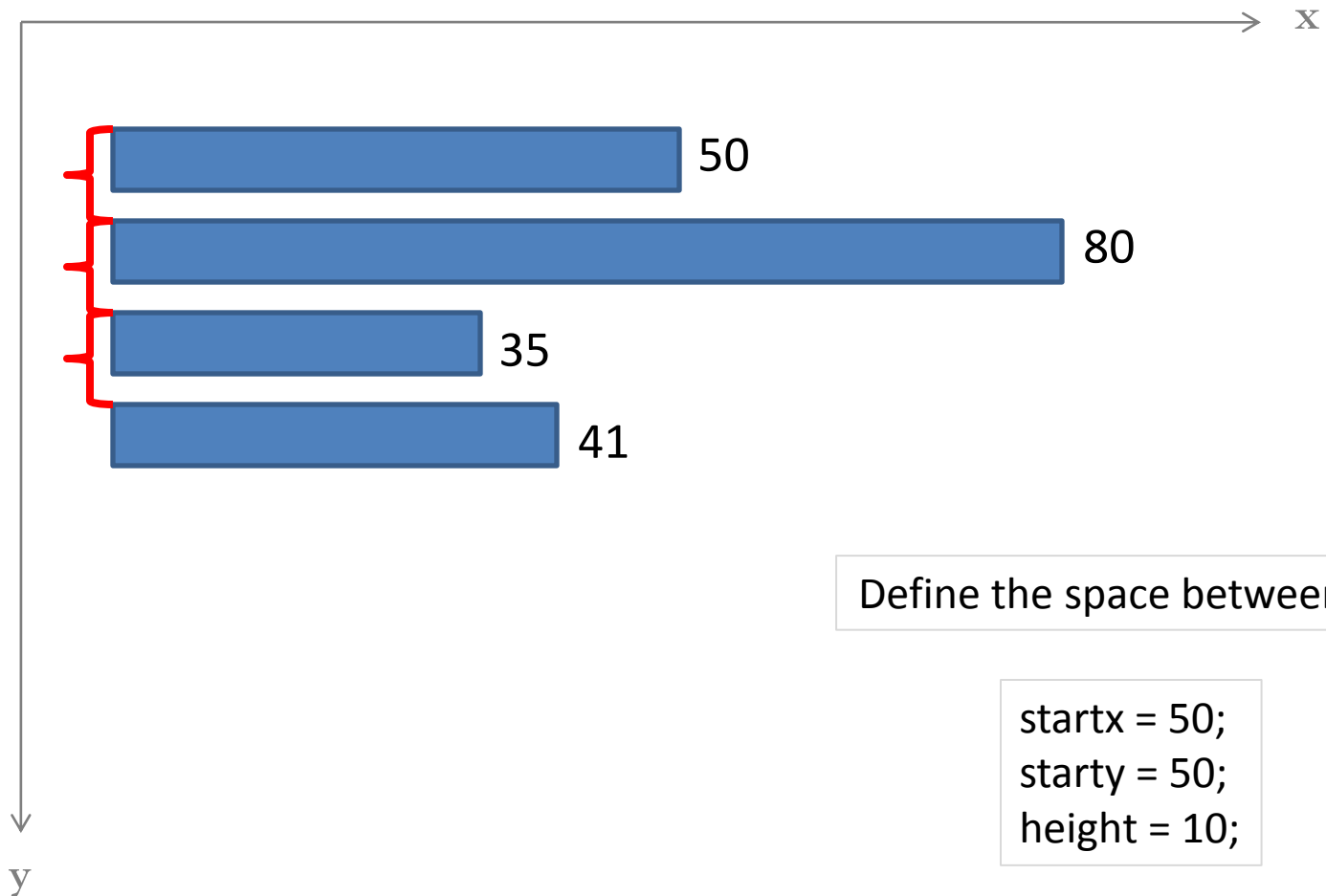
Let's do something fun with data!



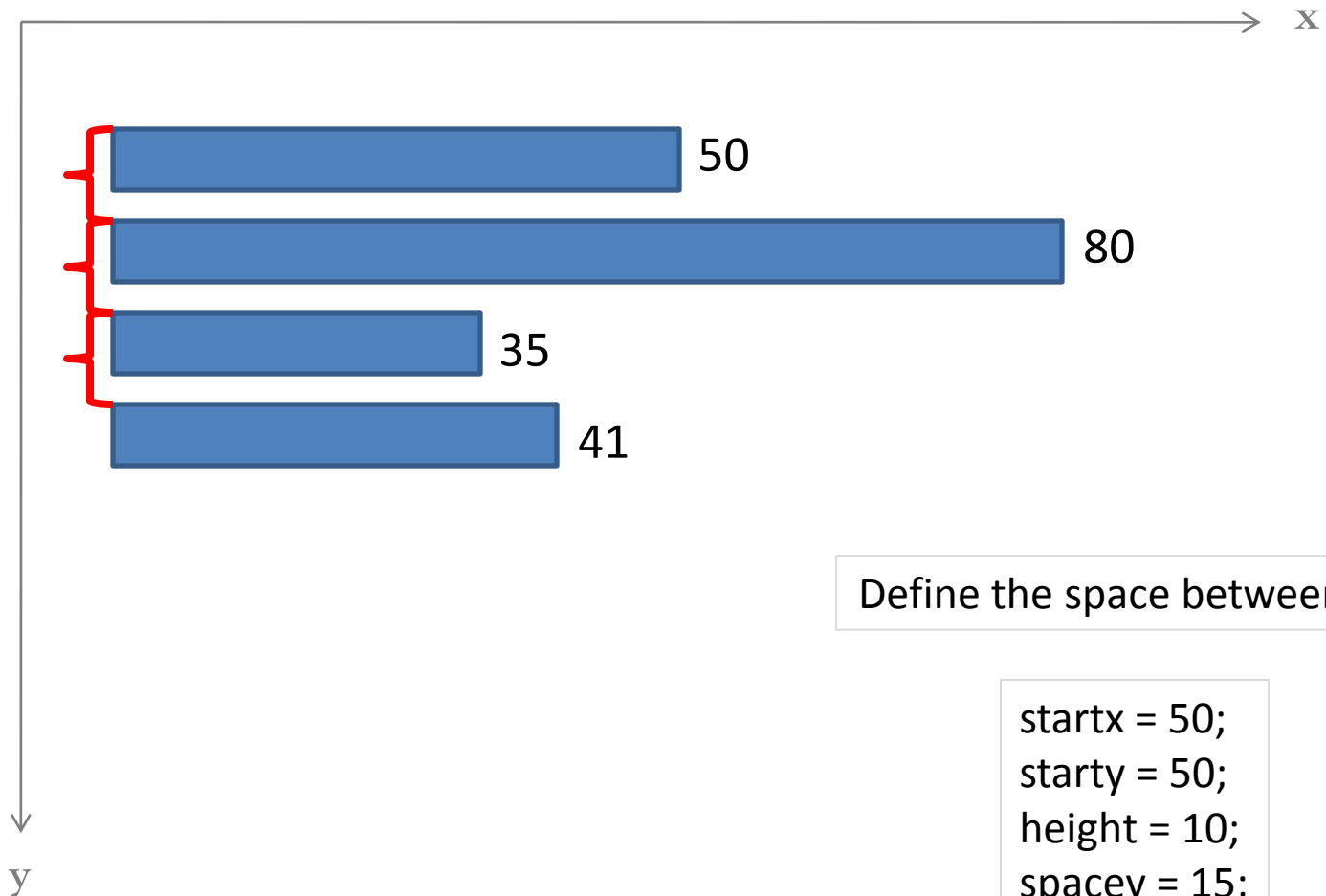
Let's do something fun with data!



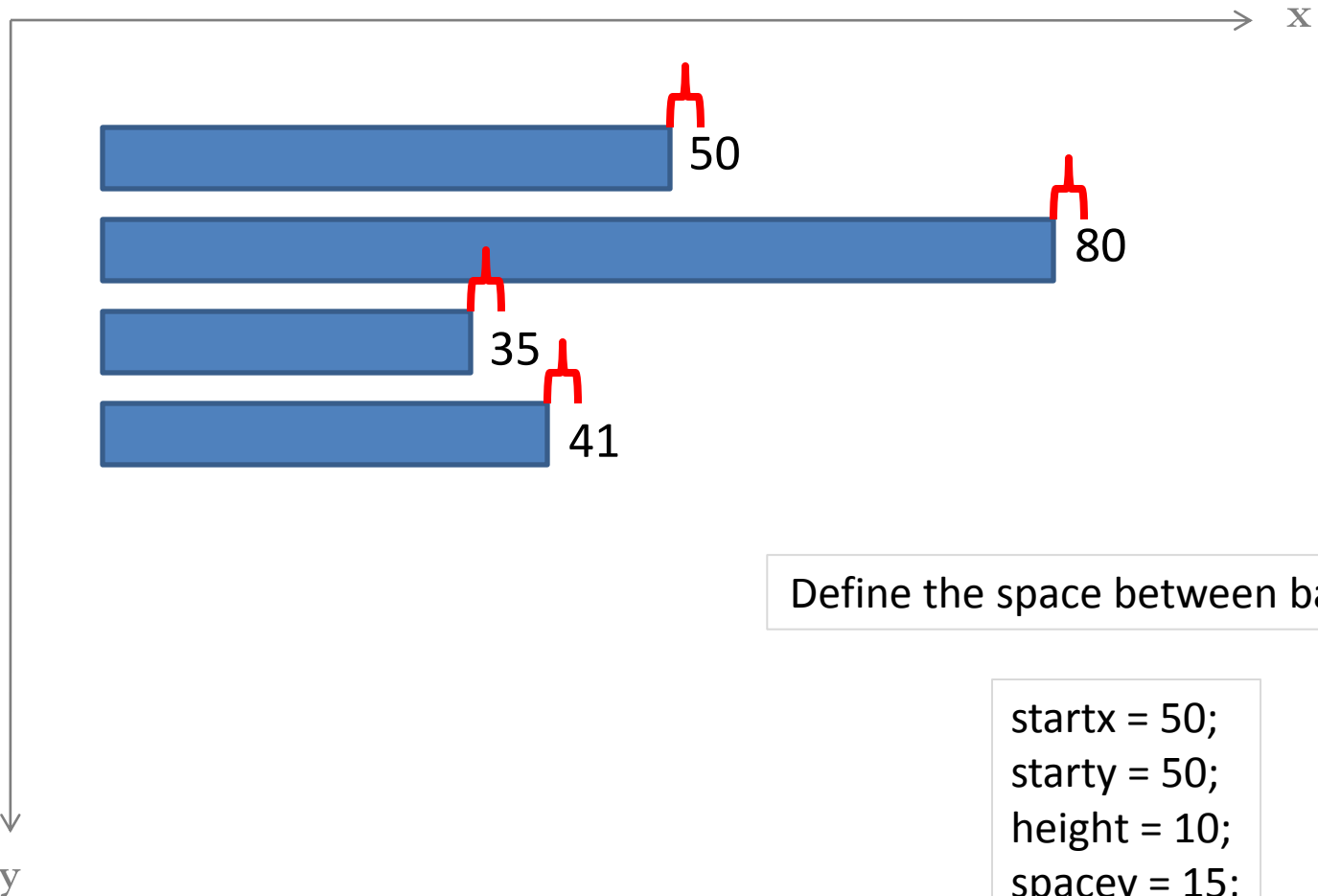
Let's do something fun with data!



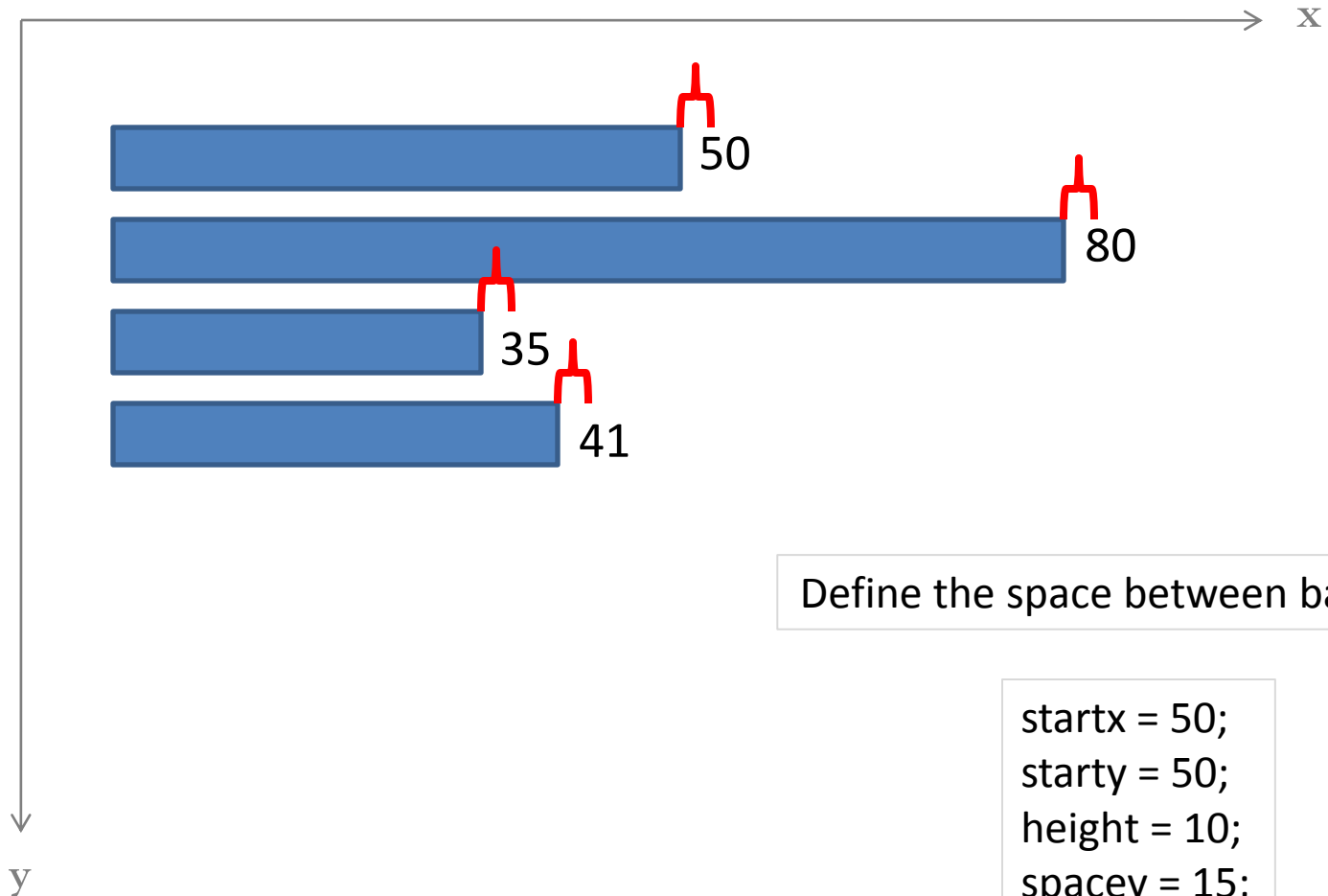
Let's do something fun with data!



Let's do something fun with data!



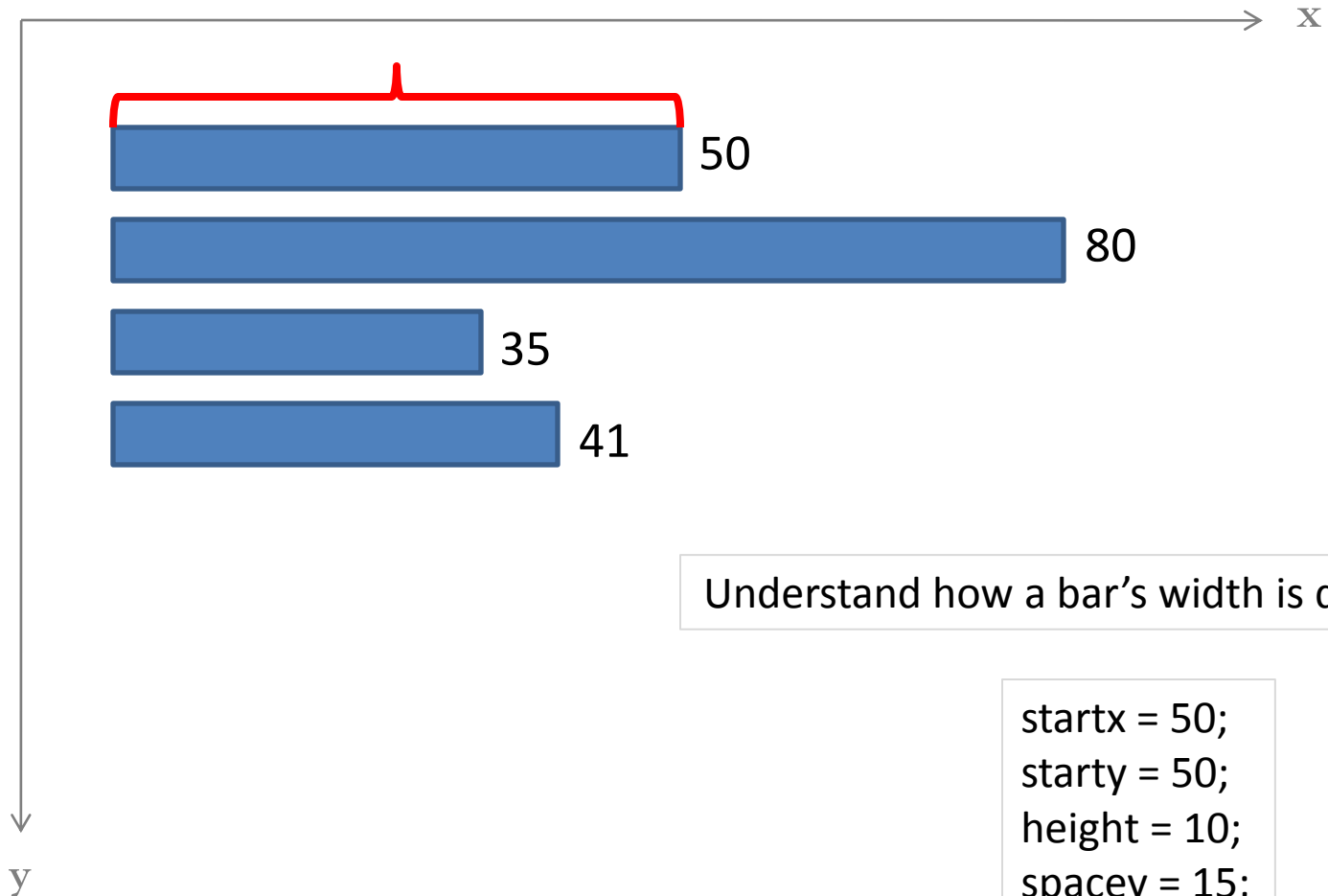
Let's do something fun with data!



Define the space between bar and label

```
startx = 50;  
starty = 50;  
height = 10;  
spacey = 15;  
spacex = 10;
```

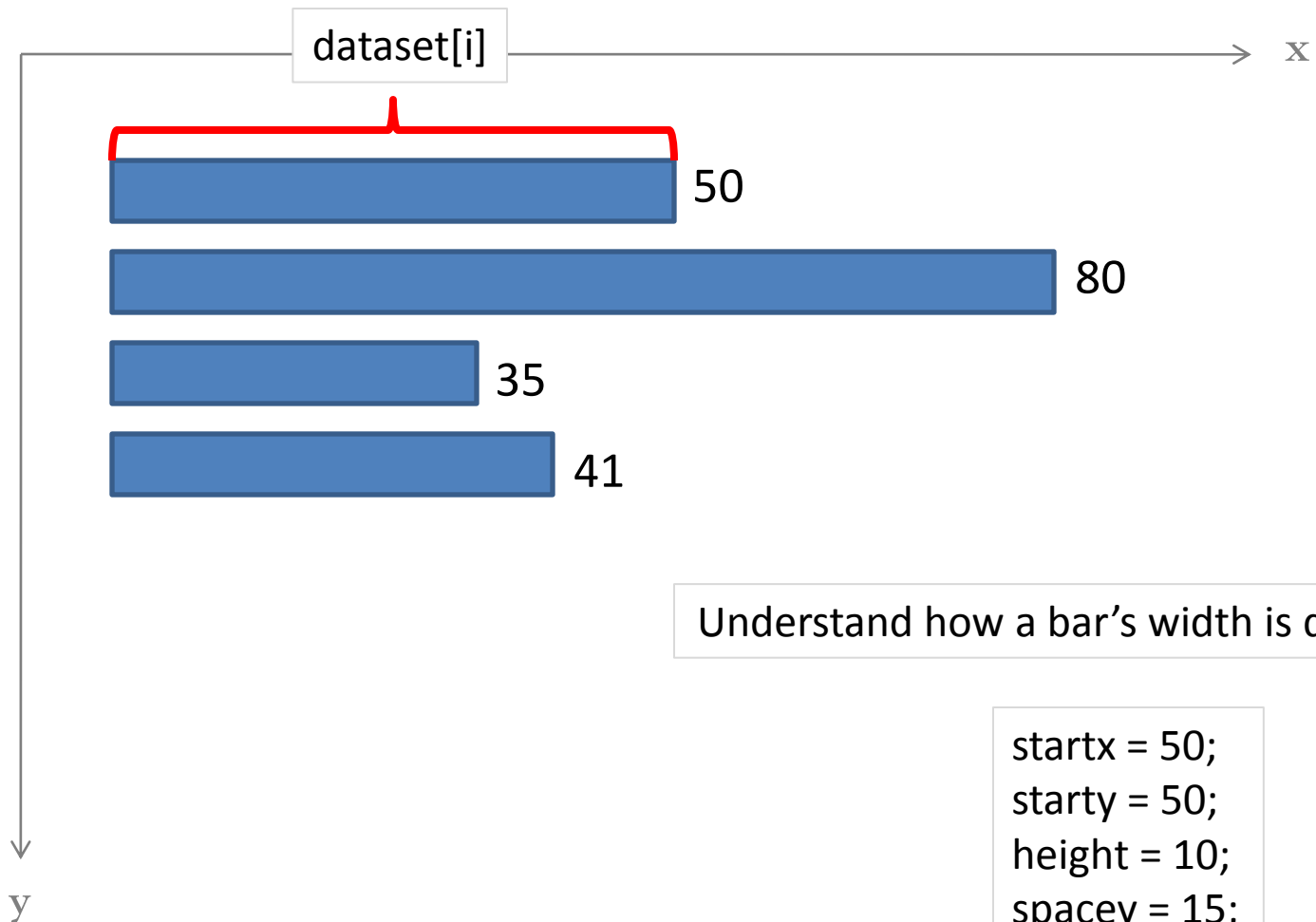
Let's do something fun with data!



Understand how a bar's width is determined

```
startx = 50;  
starty = 50;  
height = 10;  
spacey = 15;  
spacex = 10;
```

Let's do something fun with data!



Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

for(var i = 0; i < dataset.length; i++) {

}

</script>
```

Now begin to think, how
should a bar chart be drawn?
Think **procedurally!**

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {

}

</script>
```

Add our newly
discovered definitions

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {

}

</script>
```

Great, now we can draw the bars themselves.

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect();
}

</script>
```

We want a rectangle

}

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, )
    With origin at startx
}

</script>
```

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey))
  

With origin at startx and modulated starty

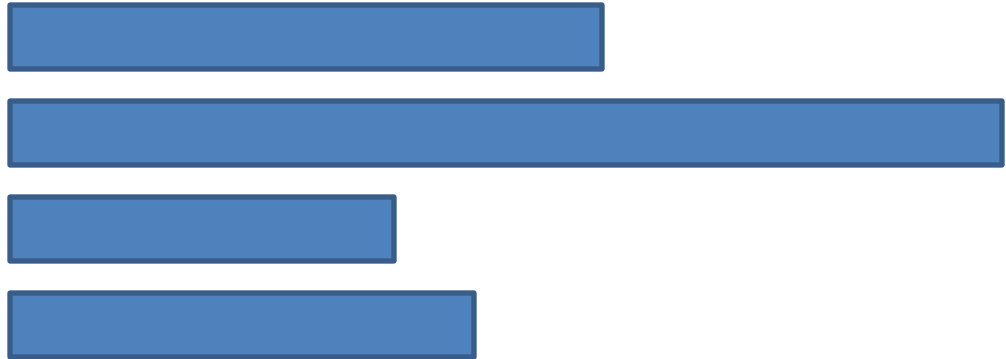

}

</script>
```

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;
```



```
for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey))
  // With origin at startx and modulated starty
}
```

```
</script>
```

Let's do something fun with data!

```
<script>
```

```
var paper = Raphael(10,10,600,600);
```

```
var dataset = [50, 80, 35, 41];
```

```
var bars = [];
```

```
var labels = [];
```

```
var startx = 50;
```

```
var starty = 50;
```

```
var height = 10;
```

```
var spacey = 15;
```

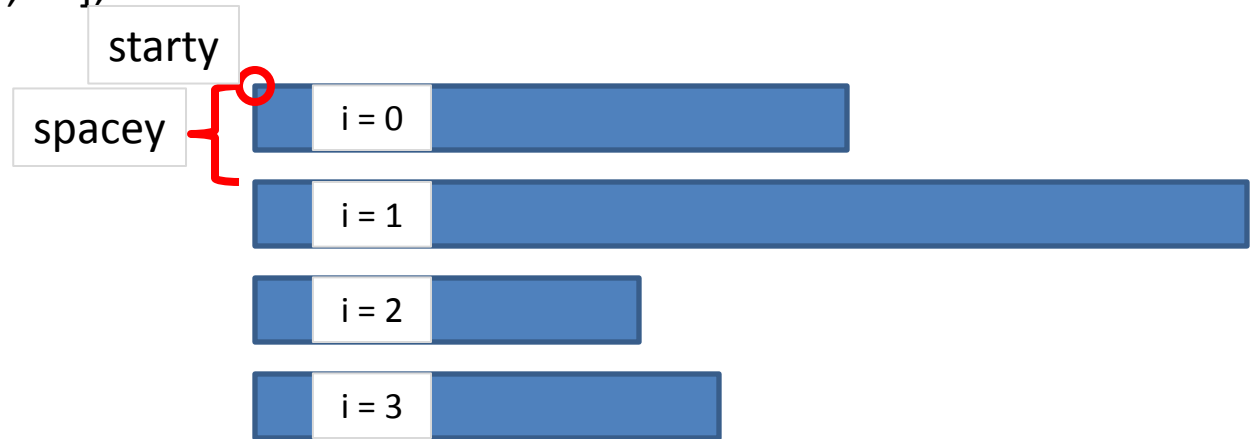
```
var spacex = 10;
```

```
for(var i = 0; i < dataset.length; i++) {  
  paper.rect(startx, starty + (i * spacey))
```

With origin at startx and modulated starty

```
}
```

```
</script>
```



Let's do something fun with data!

```
<script>
```

```
var paper = Raphael(10,10,600,600);
```

```
var dataset = [50, 80, 35, 41];
```

```
var bars = [];
```

```
var labels = [];
```

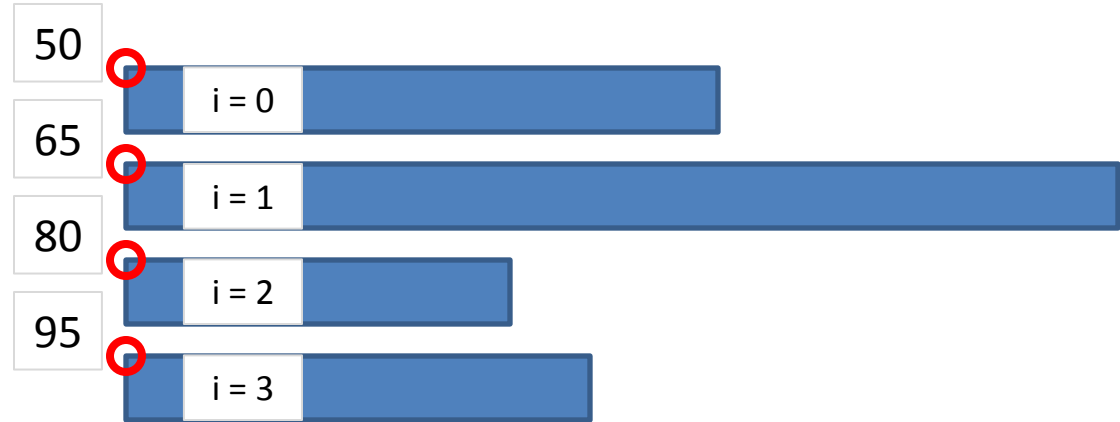
```
var startx = 50;
```

```
var starty = 50;
```

```
var height = 10;
```

```
var spacey = 15;
```

```
var spacex = 10;
```



```
for(var i = 0; i < dataset.length; i++) {  
  paper.rect(startx, starty + (i * spacey))
```

```
    With origin at startx and modulated starty
```

```
}
```

```
</script>
```


Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey), dataset[i])
}

</script>
```

With a width set by the data set

Let's do something fun with data!

```
<script>
```

```
var paper = Raphael(10,10,600,600);
```

```
var dataset = [50, 80, 35, 41];
```

```
var bars = [];
```

```
var labels = [];
```

```
var startx = 50;
```

```
var starty = 50;
```

```
var height = 10;
```

```
var spacey = 15;
```

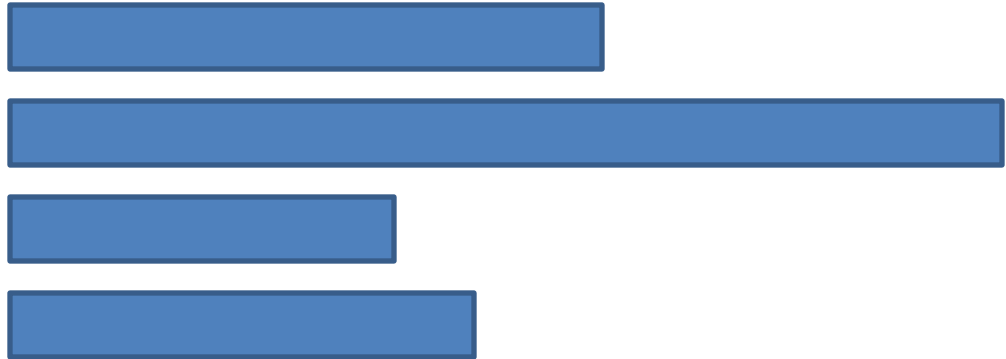
```
var spacex = 10;
```

```
for(var i = 0; i < dataset.length; i++) {
```

```
  paper.rect(startx, starty + (i * spacey), dataset[i])
```

```
}
```

```
</script>
```



With a width set by the data set

Let's do something fun with data!

```
<script>
```

```
var paper = Raphael(10,10,600,600);
```

```
var dataset = [50, 80, 35, 41];
```

```
var bars = [];
```

```
var labels = [];
```

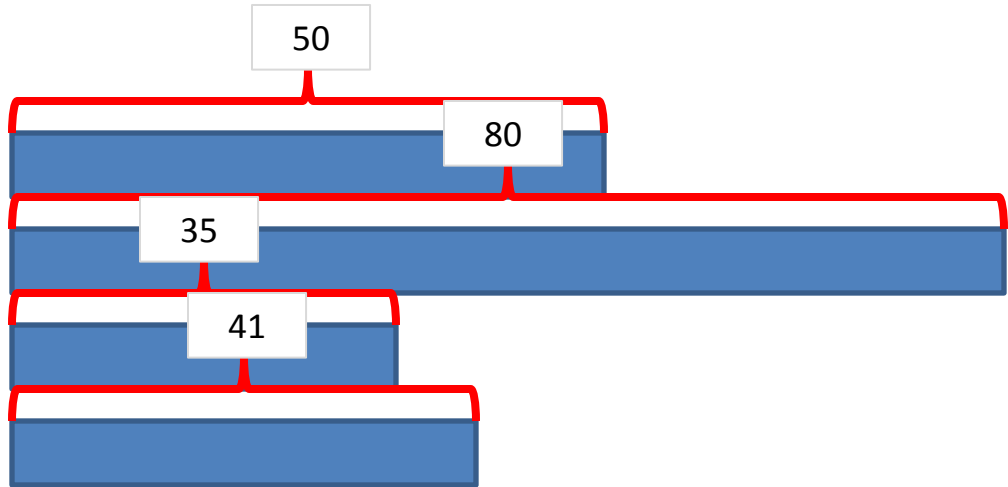
```
var startx = 50;
```

```
var starty = 50;
```

```
var height = 10;
```

```
var spacey = 15;
```

```
var spacex = 10;
```



```
for(var i = 0; i < dataset.length; i++) {
```

```
  paper.rect(startx, starty + (i * spacey), dataset[i])
```

```
}
```

With a width set by the data set

```
</script>
```

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey), dataset[i], height)
}

</script>
```

And predetermined height

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"});
}

</script>
```

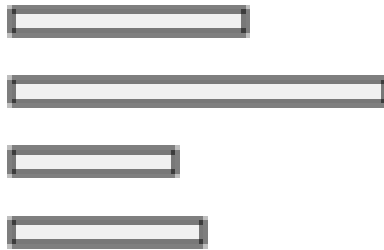
(and just for fun let's color them grey)

Let's do something fun with data!

What do we have?

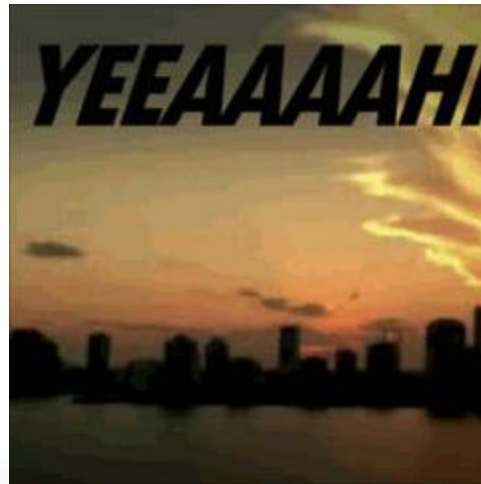
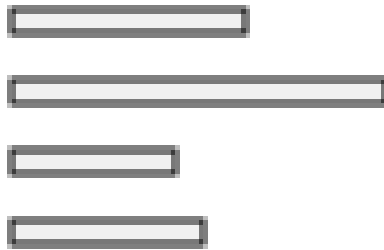
Let's do something fun with data!

What do we have?



Let's do something fun with data!

What do we have?



Let's do something fun with data!

What if we change the dataset?

Let's do something fun with data!

What if we change the dataset?

```
var dataset = [50, 80, 35, 41, 100, 28, 92, 84, 37, 59, 77, 69, 5, 99, 22, 61, 12, 43, 21, 44];
```

Let's do something fun with data!

What if we change the dataset?

```
var dataset = [50, 80, 35, 41, 100, 28, 92, 84, 37, 59, 77, 69, 5, 99, 22, 61, 12, 43, 21, 44];
```



Let's do something fun with data!

What if we change the dataset?

```
var dataset = [50, 80, 35, 41, 100, 28, 92, 84, 37, 59, 77, 69, 5, 99, 22, 61, 12, 43, 21, 44];
```

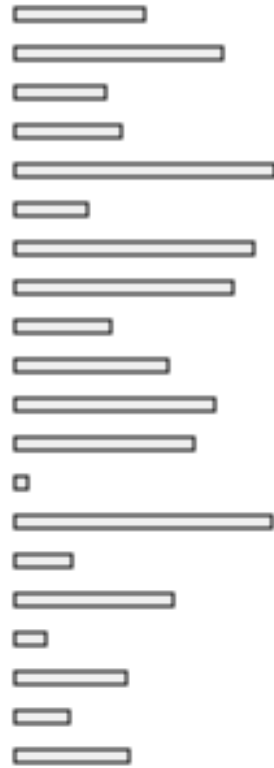


AMAZING.

Let's do something fun with data!

What if we change the dataset?

```
var dataset = [50, 80, 35, 41, 100, 28, 92, 84, 37, 59, 77, 69, 5, 99, 22, 61, 12, 43, 21, 44];
```



AMAZING.
(but we can do better)

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"});
}

</script>
```

Let's add labels!

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"});
  paper.text(startx + dataset[i] + spacex, starty + (i * spacey), dataset[i]);
}

</script>
```

Let's add labels!

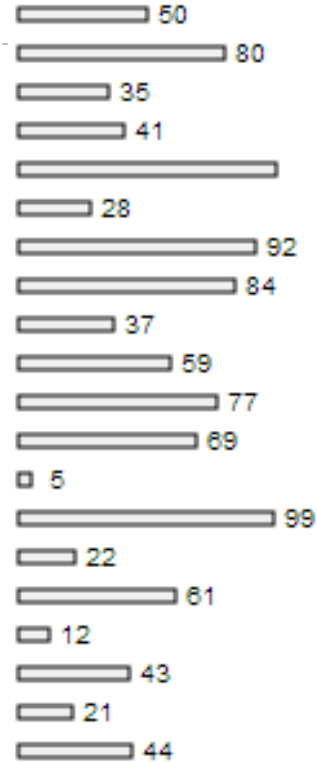
Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"});
  paper.text(startx + dataset[i] + spacex, starty + (i * spacey), dataset[i]);
}

</script>
```

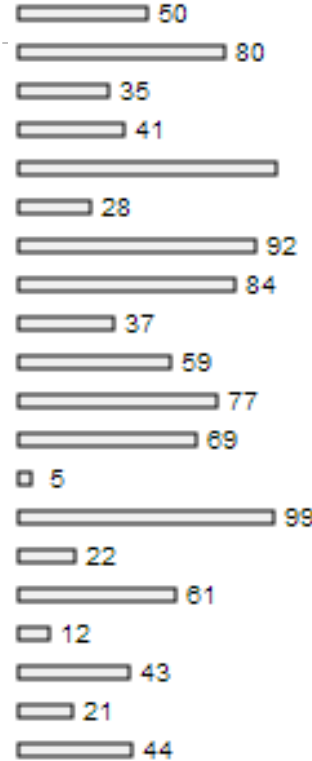


Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"});
  paper.text(startx + dataset[i] + spacex, starty + (i * spacey), dataset[i]);
}
</script>
```



AMAZING.
(but we can still do better)

Interactivity in Raphael

Interactivity in Raphael

mouseover
mousedown

mouseout
mouseup

hover
click



All fields for interactivity!

Interactivity in Raphael

mouseover
mousedown

mouseout
mouseup

hover
click



All fields for interactivity!

Element.click(handler)

Adds event handler for click for the element.

Parameters

handler function handler for the event

Interactivity in Raphael

mouseover

mouseout

hover

mousedown

mouseup

click

All fields for interactivity!

Element.click(handler)

Adds event handler for click for the element.

Parameters

handler

function

handler for the event

```
var paper = Raphael(10,10,600,600);  
var circ = paper.circle(30,30,80);
```

Interactivity in Raphael

mouseover

mouseout

hover

mousedown

mouseup

click

All fields for interactivity!

Element.click(handler)

Adds event handler for click for the element.

Parameters

handler

function

handler for the event

```
var paper = Raphael(10,10,600,600);  
var circ = paper.circle(30,30,80);  
circ.click(function() {  
    console.log("lol click!");  
});
```

Interactivity in Raphael

mouseover
mousedown

mouseout
mouseup

hover
click



All fields for interactivity!

Element.click(handler)

Adds event handler for click for the element.

Parameters

handler

function

handler for the event

```
var paper = Raphael(10,10,600,600);  
var circ = paper.circle(30,30,80);  
circ.click(function() {  
    console.log("lol click!");  
});
```



function is passed as an argument

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"});
  paper.text(startx + dataset[i] + spacex, starty + (i * spacey), dataset[i]);
}

</script>
```


Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"});
  paper.text(startx + dataset[i] + spacex, starty + (i * spacey), dataset[i]);
}
</script>
```

Suppose we only want to see labels while we hover over the bar.

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"});
  paper.text(startx + dataset[i] + spacex, starty + (i * spacey), dataset[i]).attr({"opacity": 0.0});
}

</script>
```

We'll need to start by initially drawing every label transparent.

Let's do something fun with data!

```
<script>
var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41];
var bars = [];
var labels = [];

var startx = 50;
var starty = 50;
var height = 10;
var spacey = 15;
var spacex = 10;

for(var i = 0; i < dataset.length; i++) {
  paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"});
  paper.text(startx + dataset[i] + spacex, starty + (i * spacey), dataset[i]).attr({"opacity": 0.0});
}

</script>
```

Now we use interactivity to make
the text reappear!

Let's do something fun with data!

```
...  
var bars = [];  
var labels = [];  
  
for(var i = 0; i < dataset.length; i++) {  
  paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"});  
  paper.text(startx + dataset[i] + spacex, starty + (i * spacey), dataset[i]).attr({"opacity": 0.0});  
}
```

We'll need to actually **STORE** the objects instead of just drawing them and forgetting about them if we wish to **recall** them.

Let's do something fun with data!

```
...  
var bars = [];  
var labels = [];  
  
for(var i = 0; i < dataset.length; i++) {  
  bars.push(paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill":  
"rgb(240,240,240)"}));  
  
  labels.push(paper.text(startx + dataset[i] + spacex, starty + (i * spacey),  
dataset[i]).attr({"opacity": 0.0}));  
  
}
```

Fortunately we have arrays for that.
As we draw the objects we **push** them
to the end of the arrays.

Let's do something fun with data!

```
...  
var bars = [];  
var labels = [];  
  
for(var i = 0; i < dataset.length; i++) {  
  bars.push(paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill":  
"rgb(240,240,240)"}));  
  
  labels.push(paper.text(startx + dataset[i] + spacex, starty + (i * spacey),  
dataset[i]).attr({"opacity": 0.0}));  
  
  bars[i].mouseover(  
  
  );  
  bars[i].mouseout(  
  
  );  
}
```

We can now add interactivity to our stored bar and label objects

Let's do something fun with data!

```
...  
var bars = [];  
var labels = [];  
  
for(var i = 0; i < dataset.length; i++) {  
  bars.push(paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill":  
"rgb(240,240,240)"}));  
  
  labels.push(paper.text(startx + dataset[i] + spacex, starty + (i * spacey),  
dataset[i]).attr({"opacity": 0.0}));  
  
  bars[i].mouseover(function() {  
  
});  
  bars[i].mouseout(function() {  
  
});  
}  
...
```

Remember, **functions** are objects!

Let's do something fun with data!

```
...  
var bars = [];  
var labels = [];  
  
for(var i = 0; i < dataset.length; i++) {  
  bars.push(paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill":  
"rgb(240,240,240)"}));  
  
  labels.push(paper.text(startx + dataset[i] + spacex, starty + (i * spacey),  
dataset[i]).attr({"opacity": 0.0}));  
  
  bars[i].mouseover(function() {  
    labels[i].attr({"opacity": 1.0});  
  });  
  bars[i].mouseout(function() {  
    labels[i].attr({"opacity": 0.0});  
  });  
}
```

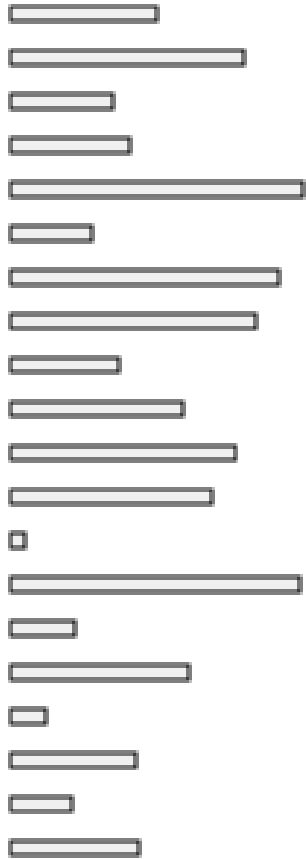
Turn on the label

Turn off the label

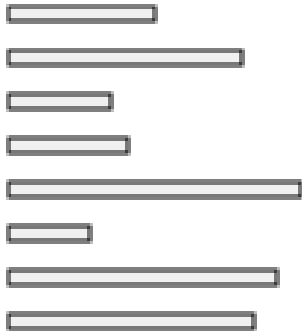
Let's do something fun with data!

```
...  
var bars = [];  
var labels = [];  
  
for(var i = 0; i < dataset.length; i++) {  
  bars.push(paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill":  
"rgb(240,240,240)"}));  
  
  labels.push(paper.text(startx + dataset[i] + spacex, starty + (i * spacey),  
dataset[i]).attr({"opacity": 0.0}));  
  
  bars[i].mouseover(function() {  
    labels[i].attr({"opacity": 1.0});  
  });  
  bars[i].mouseout(function() {  
    labels[i].attr({"opacity": 0.0});  
  });  
}
```

Cool, so let's try to run it!



So far so good.
Let's hover over something...



OH NO!

Uncaught TypeError: Cannot call method 'attr' of undefined



Let's do something fun with data!

```
...  
var bars = [];  
var labels = [];  
  
for(var i = 0; i < dataset.length; i++) {  
    bars.push(paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill":  
"rgb(240,240,240)"}));  
  
    labels.push(paper.text(startx + dataset[i] + spacex, starty + (i * spacey),  
dataset[i]).attr({"opacity": 0.0}));  
  
    bars[i].mouseover(function() {  
        labels[i].attr({"opacity": 1.0});  
    });  
    bars[i].mouseout(function() {  
        labels[i].attr({"opacity": 0.0});  
    });  
}  
}
```

The problem lies here.

Let's do something fun with data!

```
...  
var bars = [];  
var labels = [];  
  
for(var i = 0; i < dataset.length; i++) {  
  bars.push(paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill":  
"rgb(240,240,240)"}));  
  
  labels.push(paper.text(startx + dataset[i] + spacex, starty + (i * spacey),  
dataset[i]).attr({"opacity": 0.0}));  
  
  bars[i].mouseover(function() {  
    labels[i].attr({"opacity": 1.0});  
  });  
  bars[i].mouseout(function() {  
    labels[i].attr({"opacity": 0.0});  
  });  
}  
}
```

More specifically, [here](#).

Let's do something fun with data!

More specifically, [here](#).



Let's do something fun with data!

More specifically, **here**.
(Okay, what's actually happening?)



Let's do something fun with data!

```
...  
var bars = [];  
var labels = [];  
  
for(var i = 0; i < dataset.length; i++) {  
    bars.push(paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill":  
"rgb(240,240,240)"}));  
  
    labels.push(paper.text(startx + dataset[i] + spacex, starty + (i * spacey),  
dataset[i]).attr({"opacity": 0.0}));  
  
    bars[i].mouseover(function() {  
        labels[i].attr({"opacity": 1.0});  
    });  
    bars[i].mouseout(function() {  
        labels[i].attr({"opacity": 0.0});  
    });  
}  
}
```

i is not known to the anonymous function
Given to mouseover and mouseout.

Let's do something fun with data!

...

```
bars[i].mouseover(function() {  
  labels[i].attr({"opacity": 1.0});  
});
```

```
bars[i].mouseout(function() {  
  labels[i].attr({"opacity": 0.0});  
});
```

Prepare for some **Inception**-style code.

...

Let's do something fun with data!

...

```
bars[i].mouseover((function(i) {  
  return function() {  
    labels[i].attr({"opacity": 1.0});  
  }  
})(i));
```

**WE HAVE TO GO DEEPER.
FUNCTIONS IN FUNCTIONS.**

```
bars[i].mouseout((function(i) {  
  return function() {  
    labels[i].attr({"opacity": 0.0});  
  }  
})(i));
```

...

Let's do something fun with data!

...

```
bars[i].mouseover((function(i) {  
  return function() {  
    labels[i].attr({"opacity": 1.0});  
  }  
})(i));
```

To get our code to work, we needed a **wrapper function** that **returns** our initial desired function.

```
bars[i].mouseout((function(i) {  
  return function() {  
    labels[i].attr({"opacity": 0.0});  
  }  
})(i));
```

...

Let's do something fun with data!

...

```
bars[i].mouseover((function(i) {  
    return function() {  
        labels[i].attr({"opacity": 1.0});  
    }  
})(i));
```

The wrapper function takes our needed **abstract looping variable** as a parameter.

```
bars[i].mouseout((function(i) {  
    return function() {  
        labels[i].attr({"opacity": 0.0});  
    }  
})(i));
```

...

Let's do something fun with data!

...

```
bars[i].mouseover((function(i) {  
  return function() {  
    labels[i].attr({"opacity": 1.0});  
  }  
})(i));
```

```
bars[i].mouseout((function(i) {  
  return function() {  
    labels[i].attr({"opacity": 0.0});  
  }  
})(i));
```

...

But JavaScript doesn't call it with any arguments. We have to do that ourselves.

Let's do something fun with data!

...

```
bars[i].mouseover((function(i) {  
    return function() {  
        labels[i].attr({"opacity": 1.0});  
    }  
}))(i));
```

But JavaScript doesn't call it with any arguments. We have to do that ourselves.

The wrapper function is itself wrapped in **parens.**

```
bars[i].mouseout((function(i) {  
    return function() {  
        labels[i].attr({"opacity": 0.0});  
    }  
}))(i));
```

...

Let's do something fun with data!

...

```
bars[i].mouseover((function(i) {  
    return function() {  
        labels[i].attr({"opacity": 1.0});  
    }  
}))(i));
```

```
bars[i].mouseout((function(i) {  
    return function() {  
        labels[i].attr({"opacity": 0.0});  
    }  
}))(i));
```

...

But JavaScript doesn't call it with any arguments. We have to do that ourselves.

The wrapper function is itself wrapped in parens.

And then gets **forcibly called immediately following its definition.**

Let's do something fun with data!

...

```
bars[i].mouseover((function(i) {  
  return function() {  
    labels[i].attr({"opacity": 1.0});  
  }  
})(i));
```

```
bars[i].mouseout((function(i) {  
  return function() {  
    labels[i].attr({"opacity": 0.0});  
  }  
})(i));
```

...

But JavaScript doesn't call it with any arguments. We have to do that ourselves.

The wrapper function is itself wrapped in parens.

And then gets forcibly called immediately following its definition.

The wrapper function then constructs the interior function giving it the **variable that it needs, and then returns it anonymously.**

Let's do something fun with data!

...

```
bars[i].mouseover((function(i) {  
  return function() {  
    labels[i].attr({"opacity": 1.0});  
  }  
})(i));
```

```
bars[i].mouseout((function(i) {  
  return function() {  
    labels[i].attr({"opacity": 0.0});  
  }  
})(i));
```

...

But JavaScript doesn't call it with any arguments. We have to do that ourselves.

The wrapper function is itself wrapped in parens.

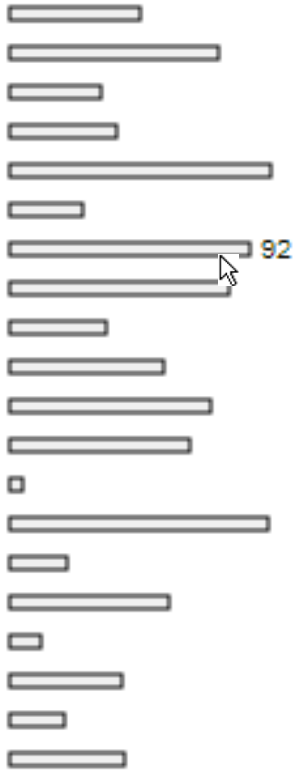
And then gets forcibly called immediately following its definition.

The wrapper function then constructs the interior function giving it the variable that it needs, and then returns it anonymously.

Functions that write functions.



Let's do something fun with data!



Hooray!
Look at how great we are.

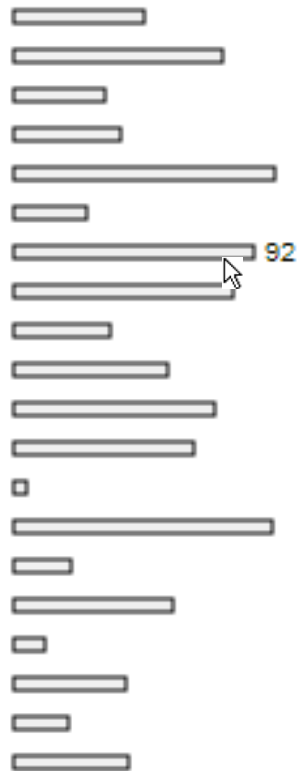
Let's do something fun with data!

```
var paper = Raphael(10,10,600,600);  
var dataset = [50, 80, 35, 41, 100, 28, 92, 84, 37, 59, 77, 69, 5, 99, 22, 61, 12, 43, 21, 44];
```

```
var startx = 50;  
var starty = 50;  
var spacex = 10;  
var spacey = 15;  
var height = 5;
```

```
var labels = [];  
var bars = [];
```

```
for(var i = 0; i < dataset.length; i++) {  
  labels.push(paper.text(startx + dataset[i] + spacex, starty + (i * spacey) + 2, dataset[i]).attr({"opacity": 0.0}));  
  bars.push(paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"}));  
  
  bars[i].mouseover(  
    (function(i){  
      return function() {  
        labels[i].attr({"opacity": 1.0});  
      };  
    })(i)  
  );  
  
  bars[i].mouseout(  
    (function(i){  
      return function(){  
        labels[i].attr({"opacity": 0.0});  
      };  
    })(i)  
  );  
}
```



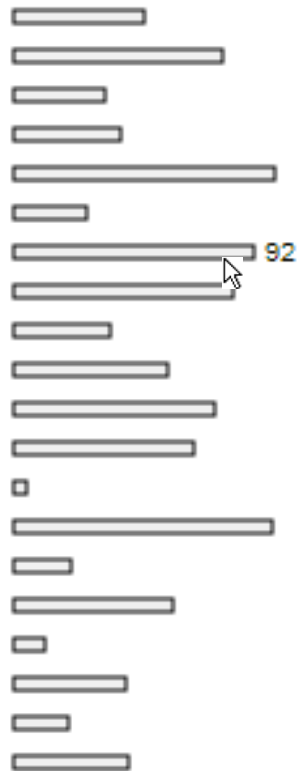
Let's do something fun with data!

```
var paper = Raphael(10,10,600,600);  
var dataset = [50, 80, 35, 41, 100, 28, 92, 84, 37, 59, 77, 69, 5, 99, 22, 61, 12, 43, 21, 44];
```

```
var startx = 50;  
var starty = 50;  
var spacex = 10;  
var spacey = 15;  
var height = 5;
```

```
var labels = [];  
var bars = [];
```

```
for(var i = 0; i < dataset.length; i++) {  
  labels.push(paper.text(startx + dataset[i] + spacex, starty + (i * spacey) + 2, dataset[i]).attr({"opacity": 0.0}));  
  bars.push(paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"}));  
  
  bars[i].mouseover(  
    (function(i){  
      return function() {  
        labels[i].attr({"opacity": 1.0});  
      };  
    })(i)  
  );  
  
  bars[i].mouseout(  
    (function(i){  
      return function(){  
        labels[i].attr({"opacity": 0.0});  
      };  
    })(i)  
  );  
}
```



Now there's some code

ONE. LAST. FEATURE.

ONE. LAST. FEATURE.

Animation in Raphael

Let's do something fun with data!

```
bars[i].mouseover(  
  (function(i){  
    return function() {  
      labels[i].attr({"opacity": 1.0});  
    };  
  })(i)  
);
```

```
bars[i].mouseout(  
  (function(i){  
    return function(){  
      labels[i].attr({"opacity": 0.0});  
    };  
  })(i)  
);
```

Let's make the bars and labels animate when we do our hover.

Let's do something fun with data!

```
bars[i].mouseover(  
  (function(i){  
    return function() {  
      labels[i].animate({"opacity": 1.0}, 300);  
    };  
  })(i)  
);  
  
bars[i].mouseout(  
  (function(i){  
    return function(){  
      labels[i].animate({"opacity": 0.0}, 300);  
    };  
  })(i)  
);
```

The **animate** function in Raphael allows you to animate any number of attributes.

Let's do something fun with data!

```
bars[i].mouseover(  
  (function(i){  
    return function() {  
      labels[i].animate({"opacity": 1.0}, 300);  
    };  
  })(i)  
);
```

```
bars[i].mouseout(  
  (function(i){  
    return function(){  
      labels[i].animate({"opacity": 0.0}, 300);  
    };  
  })(i)  
);
```

Notice the similarity to attr.
You provide your **attributes** as a JSON string and provide a second parameter of **time to take in ms**.

Let's do something fun with data!

```
bars[i].mouseover(  
  (function(i){  
    return function() {  
      labels[i].animate({"opacity": 1.0}, 300);  
      bars[i].animate({"fill": "rgb(170,170,170)"}, 300);  
    };  
  })(i)  
);
```

And now we add the part for the bars.
NOW we're done. =)

```
bars[i].mouseout(  
  (function(i){  
    return function(){  
      labels[i].animate({"opacity": 0.0}, 300);  
      bars[i].animate({"fill": "rgb(240,240,240)"}, 300);  
    };  
  })(i)  
);
```

```

var paper = Raphael(10,10,600,600);
var dataset = [50, 80, 35, 41, 100, 28, 92, 84, 37, 59, 77, 69, 5, 99, 22, 61, 12, 43, 21, 44];

var startx = 50;
var starty = 50;
var spacex = 10;
var spacey = 15;
var height = 5;

var labels = [];
var bars = [];

for(var i = 0; i < dataset.length; i++) {
  labels.push(paper.text(startx + dataset[i] + spacex, starty + (i * spacey) + 2, dataset[i]).attr({"opacity": 0.0}));
  bars.push(paper.rect(startx, starty + (i * spacey), dataset[i], height).attr({"fill": "rgb(240,240,240)"}));

  bars[i].mouseover(
    (function(i){
      return function() {
        bars[i].animate({"fill": "rgb(170,170,170)"}, 300);
        labels[i].animate({"opacity": 0.6}, 300);
      };
    })(i)
  );

  bars[i].mouseout(
    (function(i){
      return function(){
        bars[i].animate({"fill": "rgb(240,240,240)"}, 300);
        labels[i].animate({"opacity": 0.0}, 300);
      };
    })(i)
  );
}

```

Our completed bar graph generator

HOMEWORK

(This is on Collab)

This week you'll be implementing a friendly fellow called the "timid circle" in Raphael. Woo-hoo!