**June 28, 2016**

* Pointer events, allow you to do hovers and other things, pointer dash events , none (css style)
* Finds an area that can fit as much as possible without interfering with another coordinate (Voronoi): Read up on this here: <https://bl.ocks.org/mbostock/8033015>
* Hover over, add invisible rectangles or circles. It’s a path. Wrap data around voronoi function, like a line generator but more complicated
* Map returns the numbers

Next class we'll be going through another fundamental, but difficult, part of D3, which is called the "General Update Pattern."

For that, read through these two blog posts:

* [Thinking with joins](https://bost.ocks.org/mike/join/)
* [Three little circles](https://bost.ocks.org/mike/circles/)
* <https://bost.ocks.org/mike/algorithms/>: visualizing how algorithms work
* d3:: 4.0, new version of d3 coming out!! Read this: <https://github.com/d3/d3/blob/master/CHANGES.md>

can load modules instead of all of the library (API directory).

These all take work to understand, but you can see Mike's explanations get better and better.

* [Three little circles](http://bost.ocks.org/mike/circles/)
* [How selections work](http://bost.ocks.org/mike/selection/)
* [selection.data()](https://github.com/mbostock/d3/wiki/Selections#data)
* [Thinking with joins](http://bost.ocks.org/mike/join/)
* [General Update Pattern](http://bl.ocks.org/mbostock/3808218)
* [General Update Pattern II](http://bl.ocks.org/mbostock/3808221)
* [General Update Pattern III](http://bl.ocks.org/mbostock/3808234)

1. Make a checklist for your
   1. Make new data join
   2. Get rid of old elements
   3. Enter new elements
   4. append elements as needed
   5. update new selection

Core idea of the update pattern

select it and data

exit method for when its not the same number of things connects

more data than elements : add enter

the same amount of data as before: update

if more elements than new

.data (data, function(d) { return d; ) the second part of this is the ID, so d.company (needs to be distinct: different for every single one). So that when you are updating.

SQL, what are you joining on to make sure you connect the data the right way

<https://esri.github.io/cedar/examples/drill-down.html>

if you have over 10,000 html objects on the page, it goes slow. Can use react

do your css : designer does its css page. Directory, separate page

functions are reusable chart functions, that can be called in other files

input type : range min and max with steps with a slider, give it an id

within update chart function::can’t stick it into a

d3.select(“#scaleRange”)

.on(“input”, function(){})

var this.value: slider value

slider:: using the slider!