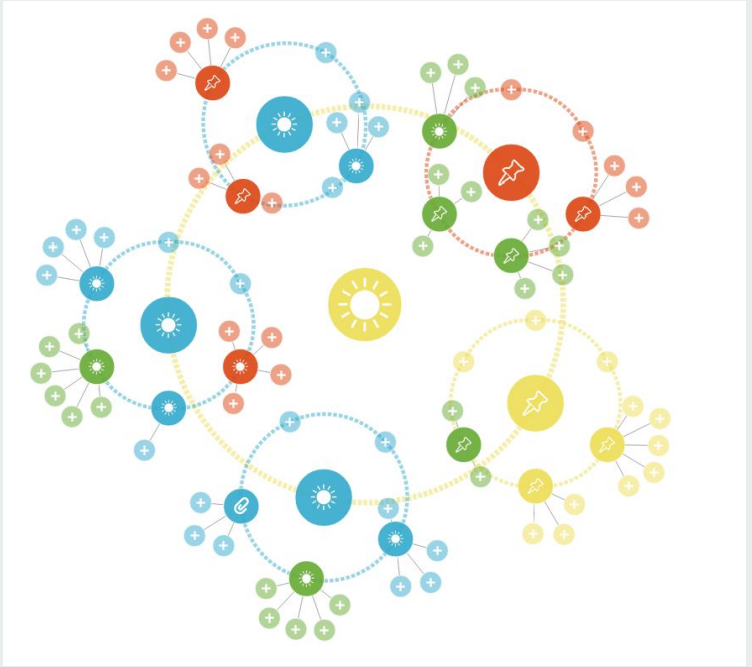
D3 Force Layout

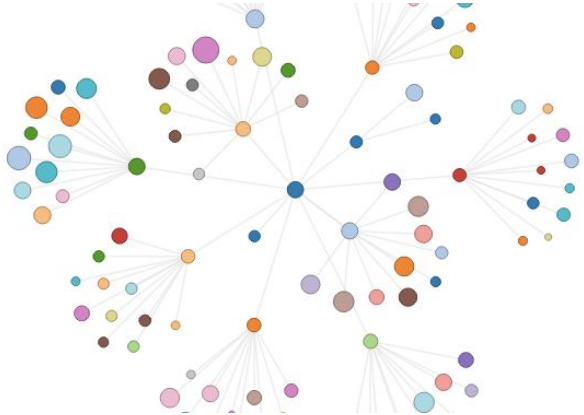
Peter Cook



Physics base simulator

D3's force layout uses a **physics based simulator** for positioning visual elements. Forces can be set up between elements, for example:

- all elements repel one another
- elements are attracted to center(s) of gravity
- linked elements (e.g. friendship) are a fixed distance apart (network visualisation)
- elements may not overlap (collision detection)



Setting up force simulation

- create an array of objects
- call `forceSimulation`, passing in the array of objects
- add one or more force functions (e.g. `forceManyBody`, `forceCenter`, `forceCollide`) to the system
- set up a callback function to update the element positions after each tick

```
var width = 300, height = 300
var nodes = [{}, {}, {}, {}, {}]

var simulation = d3.forceSimulation(nodes)
  .force('charge', d3.forceManyBody())
  .force('center', d3.forceCenter(width / 2, height / 2))
  .on('tick', ticked);
```

forceCollide

`forceCollide` is used to stop elements overlapping and is particularly useful when 'clumping' circles together.

We must specify the radius of the elements using `.radius()`:

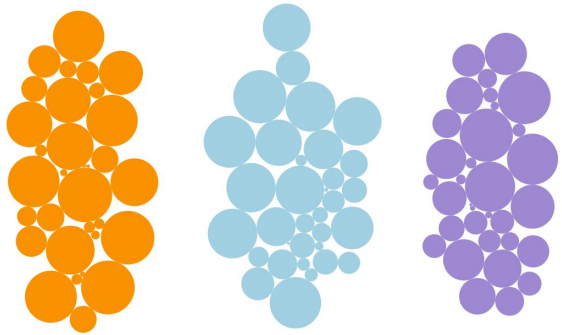
```
var numNodes = 100
var nodes = d3.range(numNodes).map(function(d) {
  return {radius: Math.random() * 25}})

var simulation = d3.forceSimulation(nodes)
  .force('charge', d3.forceManyBody().strength(5))
  .force('center', d3.forceCenter(width / 2, height / 2))
  .force('collision', d3.forceCollide().radius(function(d) { return d.radius  }))
```

forceX

`forceX` and `forceY` cause elements **to be attracted towards** specified position(s). We can use a single center for all elements or apply the force on a per-element basis. The strength of attraction can be configured using `.strength()`.

As an example suppose we have a number of elements, each of which ha category `0`, `1` or `2`. We can add a `forceX` force function to attract the elements to an x-coordinate `100`, `300` or `500` based on the element's category:

```
simulation.force('x', d3.forceX().x(function(d) {
  return xCenter[d.category];
}))
```



forceY

```
simulation.force('x', d3.forceX().x(function(d) {
  return xScale(d.value);
}))
.force('y', d3.forceY().y(function(d) {
  return 0;
}))
```

forceLink

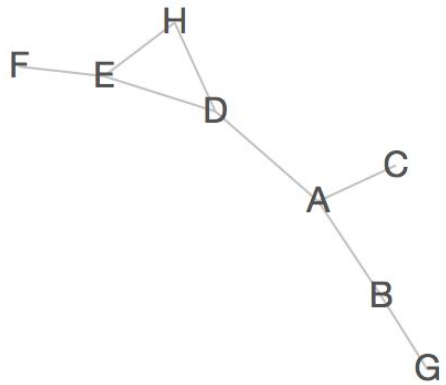
`forceLink` pushes linked elements to be a **fixed distance apart**. It requires an **array of links** that specify which elements we want to link together. Each link object specifies a source and target element, where the value is the element's array index:

```
var links = [
  {source: 0, target: 1},
  {source: 0, target: 2},
  {source: 0, target: 3},
  {source: 1, target: 6},
  {source: 3, target: 4},
  {source: 3, target: 7},
  {source: 4, target: 5},
  {source: 4, target: 7}
]
```

forceLink

We can then pass our links array into the `forceLink` functi
using `.links()`:

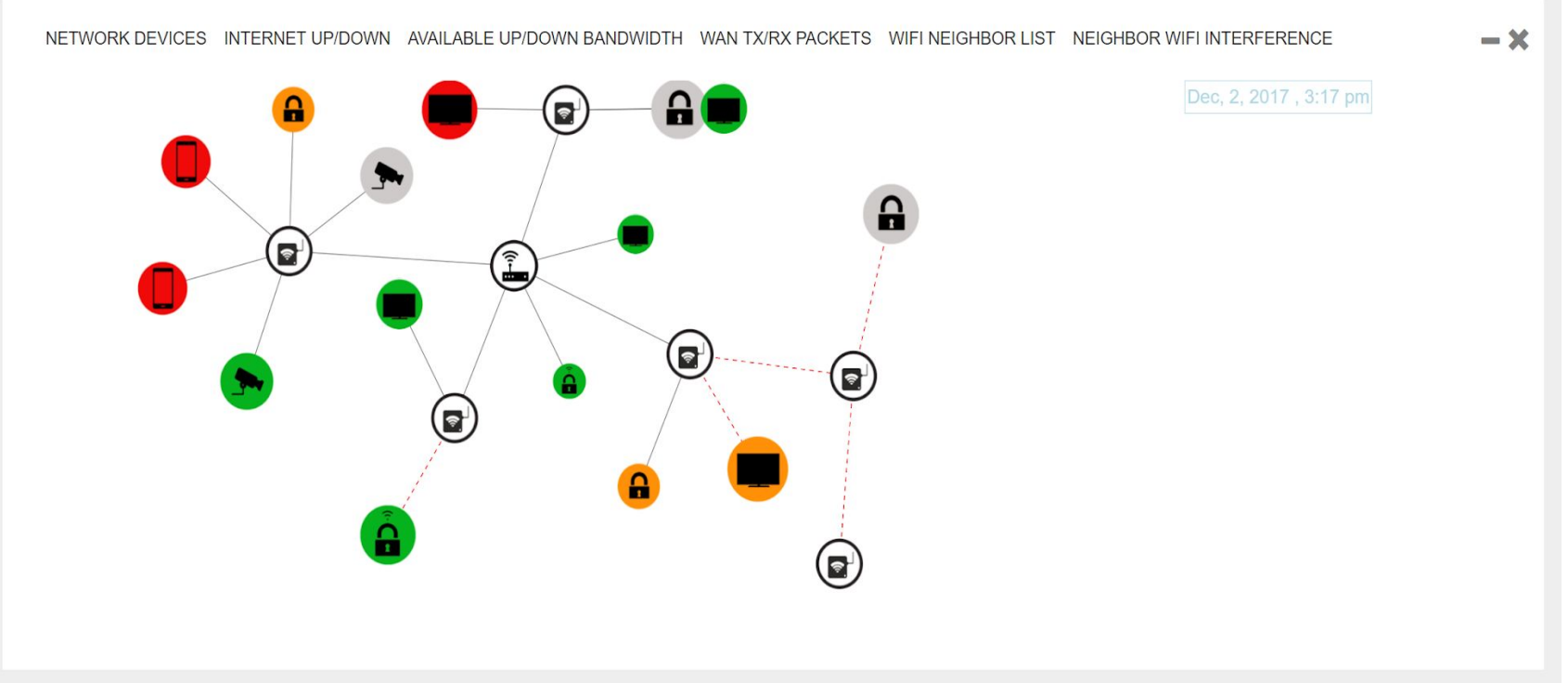
```
simulation.force('link', d3.forceLink().links(links))
```



Complete list of funcs

- d3.forceSimulation - create a new force simulation.
- *simulation*.restart - reheat and restart the simulation's timer.
- *simulation*.stop - stop the simulation's timer.
- *simulation*.tick - advance the simulation one step.
- *simulation*.nodes - set the simulation's nodes.
- *simulation*.alpha - set the current alpha.
- *simulation*.alphaMin - set the minimum alpha threshold.
- *simulation*.alphaDecay - set the alpha exponential decay rate.
- *simulation*.alphaTarget - set the target alpha.
- *simulation*.velocityDecay - set the velocity decay rate.
- *simulation*.force - add or remove a force.
- *simulation*.find - find the closest node to the given position.
- *simulation*.on - add or remove an event listener.
- *force* - apply the force.
- *force*.initialize - initialize the force with the given nodes.
- d3.forceCenter - create a centering force.
- *center*.x - set the center *x*-coordinate.
- *center*.y - set the center *y*-coordinate.
- d3.forceCollide - create a circle collision force.
- *collide*.radius - set the circle radius.
- *collide*.strength - set the collision resolution strength.
- *collide*.iterations - set the number of iterations.
- d3.forceLink - create a link force.
- *link*.links - set the array of links.
- *link*.id - link nodes by numeric index or string identifier.
- *link*.distance - set the link distance.
- *link*.strength - set the link strength.
- *link*.iterations - set the number of iterations.
- d3.forceManyBody - create a many-body force.
- *manyBody*.strength - set the force strength.
- *manyBody*.theta - set the Barnes–Hut approximation accuracy.
- *manyBody*.distanceMin - limit the force when nodes are close.
- *manyBody*.distanceMax - limit the force when nodes are far.
- d3.forceX - create an *x*-positioning force.
- x.strength - set the force strength.
- x.x - set the target *x*-coordinate.
- d3.forceY - create an *y*-positioning force.
- y.strength - set the force strength.
- y.y - set the target *y*-coordinate.
- d3.forceRadial - create a radial positioning force.
- *radial*.strength - set the force strength.
- *radial*.radius - set the target radius.
- *radial*.x - set the target center *x*-coordinate.
- *radial*.y - set the target center *y*-coordinate.

Some of my work

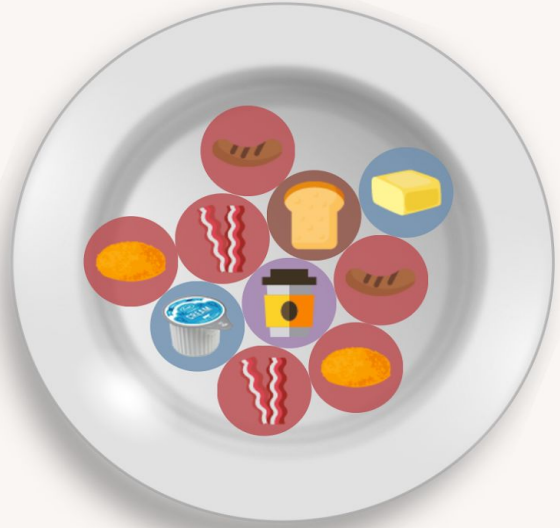


Choose Your Breakfast: What The Experts Recommend

Fruits

Pastries

Beverages



Eggs & Meats

Cereals

Other