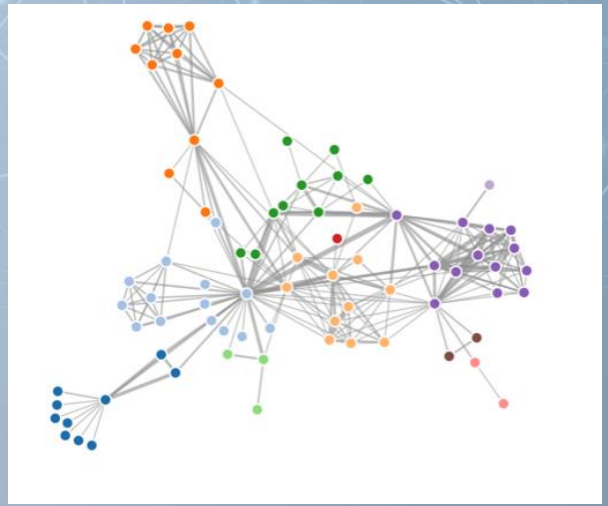
Force-directed Layout

Data Visualization

Force-directed Layout

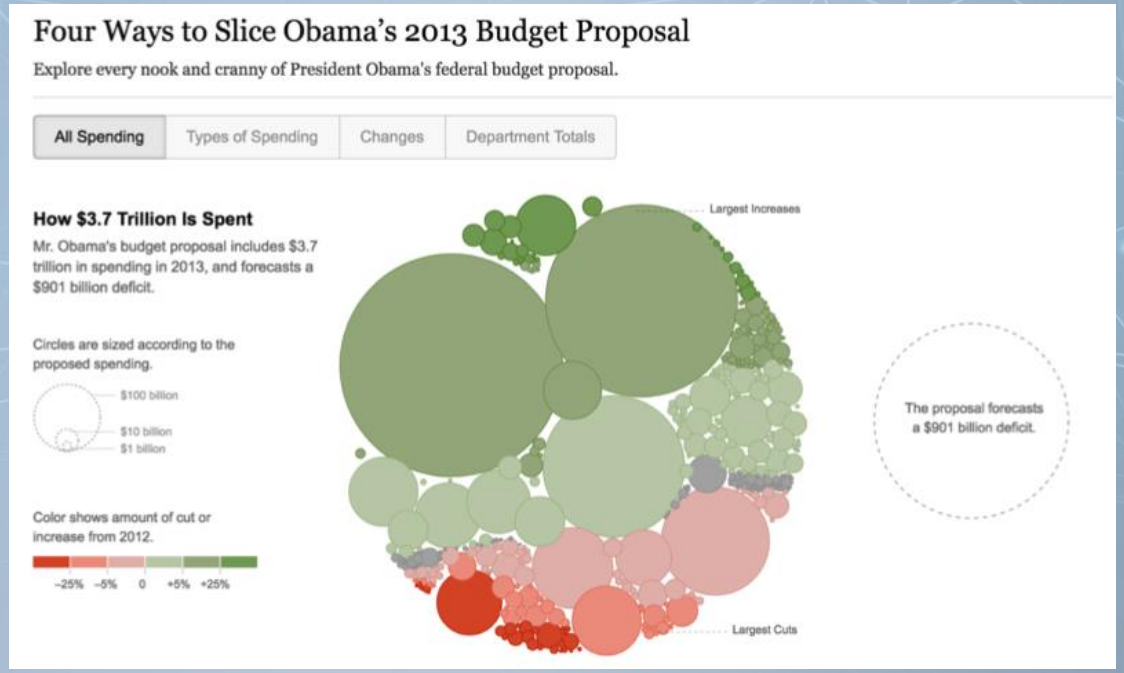
- Use a physics-based simulator for positioning visual elements
  - (Do not define the positions of nodes by you)
  - Given the conditions (force to attract or repulse nodes)
  - Forces move nodes each iteration
  - Settles into a stable configuration



- D3: forced directed layout document
  - https://github.com/d3/d3-force

Application

- https://archive.nytimes.com/www.nytimes.com/interactive/2012/02/13/us/politics/2013-budget-proposal-graphic.html



Steps to Use D3 Force

- Step1: initialize the force simulation
- Step2: add force functions to the system
- Step3: create a callback function to update SVG positions after every “tick”

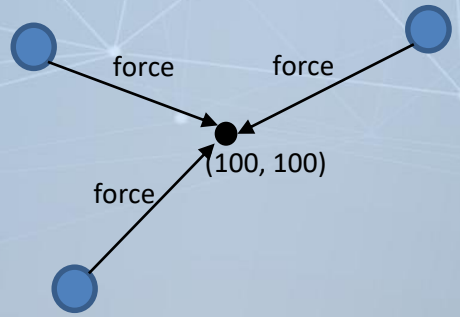
D3 Force Functions

- **forceCenter**: for setting the center of gravity of the system
- **forceManyBody**: for making elements attract or repel one another
- **forceCollide**: for preventing elements overlapping
- **forceX** and **forceY**: for attracting elements to a given point
- **forceRadial**: for force towards a circle of the specified radius centered at (x,y)
- **forceLink**: for creating a fixed distance between connected elements

- Example:
  - simulation.force('charge', d3.forceManyBody())

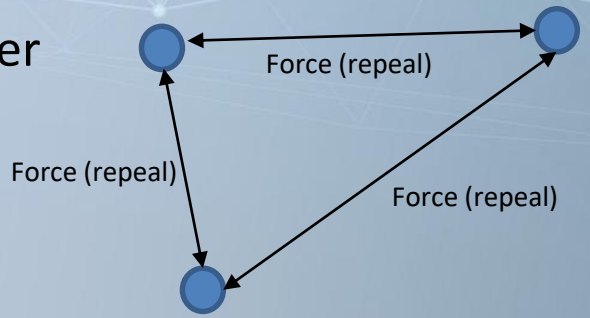
forceCenter

- Centering your elements around a central point of gravity
- simulation.force('center', d3.forceCenter(100,100))
  - All the element will be attracted to (100,100)



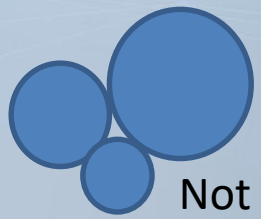
forceManyBody

- Making your elements attract or repeal each other (default strength value: -30)
  - Negative value: repeal each other
  - Positive value: attract each other
  - large positive/negative value: attract/repeal each other with more strength
- simulation.force('charge', d3.forceManyBody.strength(-20))
  - -20: repeal each other



forceCollide

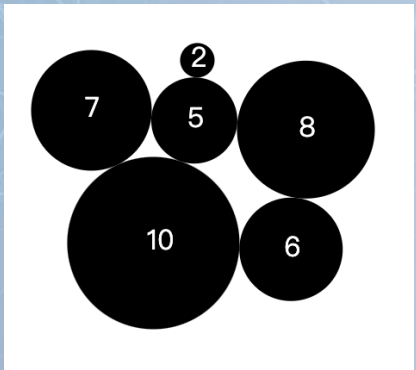
- Stopping elements from overlapping
- Idea
  - Give each element a radius which defined a round area for each element
  - If any two of them overlap with each other, force collide will try to push them apart



Not allow to overlap

Ex09-01

- main.js
- Demonstrate and explain the details of the force simulator



Radius of each circles
To use the simulator, each node (element) must be a dictionary

```
var width = 300, height = 300;
var nodes = [{value: 5}, {value: 10}, {value: 2}, {value: 6}, {value: 7}, {value: 8}];
// var nodes = [{value: 5}, {value: 10, fx: 170, fy:170}, {value: 2}, {value: 6},
//              {value: 7}, {value: 8}];

var value2radius = d3.scaleLinear()
    .domain([0, d3.max(nodes, function(d) {
        return d.value;
    })])
    .range([0, 50]);

var circles = d3.select('svg')
    .selectAll('circle')
    .data(nodes)
    .enter()
    .append('circle')
    .attr('r', function(d) {
        return value2radius(d.value);
    });

var texts = d3.select('svg')
    .selectAll('text')
    .data(nodes)
    .enter()
    .append('text')
    .attr('fill', 'white')
    .text(function(d) {
        return d.value;
    });
```

Scale function to map ‘value’ to real circle radius

Add circles to svg

Add texts to each circle

nodes: your data.
nodes is an array with 6 elements, so simulator knows it has to simulate the force among 6 elements

Ex09-01

- Step1: initialize the force simulation

```
var simulation = d3.forceSimulation(nodes)
    .force('charge', d3.forceManyBody().strength(10))
    .force('center', d3.forceCenter(width / 2, height / 2))
    .force('collision', d3.forceCollide().radius(function(d) {
        return value2radius(d.value);
    }))
    .on('tick', ticked);

function ticked() {
    d3.selectAll('circle')
        .attr('cx', function(d) {
          return d.x
        })
        .attr('cy', function(d) {
          return d.y
        });

    d3.selectAll('text')
        .attr('dx', -4)
        .attr('dy', 4)
        .attr('x', function(d) {
          return d.x
        })
        .attr('y', function(d) {
          return d.y
        });
}
```

Ex09-01

- Step2: add force functions to the system
  - use .force()

d3.forceManyBody().strength(10)
These elements will attract each other

```
var simulation = d3.forceSimulation(nodes)
    .force('charge', d3.forceManyBody().strength(10))
    .force('center', d3.forceCenter(width / 2, height / 2))
    .force('collision', d3.forceCollide().radius(function(d) {
        return value2radius(d.value);
    }))
    .on('tick', ticked);

function ticked() {
    d3.selectAll('circle')
        .attr('cx', function(d) {
          return d.x
        })
        .attr('cy', function(d) {
          return d.y
        });

    d3.selectAll('text')
        .attr('dx', -4)
        .attr('dy', 4)
        .attr('x', function(d) {
          return d.x
        })
        .attr('y', function(d) {
          return d.y
        });
}
```

Ex09-01

- Step2: add force functions to the system
  - use .force()

d3.forceCenter(width/2, height/2)
Attract all element to (width/2, height/2)

```
var simulation = d3.forceSimulation(nodes)
    .force('charge', d3.forceManyBody().strength(10))
    .force('center', d3.forceCenter(width / 2, height / 2))
    .force('collision', d3.forceCollide().radius(function(d) {
        return value2radius(d.value);
    }))
    .on('tick', ticked);

function ticked() {
    d3.selectAll('circle')
        .attr('cx', function(d) {
          return d.x
        })
        .attr('cy', function(d) {
          return d.y
        });

    d3.selectAll('text')
        .attr('dx', -4)
        .attr('dy', 4)
        .attr('x', function(d) {
          return d.x
        })
        .attr('y', function(d) {
          return d.y
        });
}
```

Ex09-01

- Step2: add force functions to the system
  - use .force()

d3.forceCollide()
These elements should not overlap with each other

.radius()
Give it a function to set “the region” of each circle

```
var simulation = d3.forceSimulation(nodes)
    .force('charge', d3.forceManyBody().strength(10))
    .force('center', d3.forceCenter(width / 2, height / 2))
    .force('collision', d3.forceCollide().radius(function(d) {
        return value2radius(d.value);
    }))
    .on('tick', ticked);

function ticked() {
    d3.selectAll('circle')
        .attr('cx', function(d) {
          return d.x
        })
        .attr('cy', function(d) {
          return d.y
        });

    d3.selectAll('text')
        .attr('dx', -4)
        .attr('dy', 4)
        .attr('x', function(d) {
          return d.x
        })
        .attr('y', function(d) {
          return d.y
        });
}
```

Ex09-01

- Step2: add force functions to the system
  - use .force()

The first parameter of .force()?

Just a ‘name’ of the force you assign, you can give it an arbitrary name (such as ‘abc’)

If you want to remove a force from the simulator, you will need the name. For example, “simulation.force("**charge**", null);”

If you never remove a force, the names are useless.

```
var simulation = d3.forceSimulation(nodes)
    .force('charge', d3.forceManyBody().strength(10))
    .force('center', d3.forceCenter(width / 2, height / 2))
    .force('collision', d3.forceCollide().radius(function(d) {
            return value2radius(d.value);
    }))
    .on('tick', ticked);

function ticked() {
    d3.selectAll('circle')
        .attr('cx', function(d) {
          return d.x
        })
        .attr('cy', function(d) {
          return d.y
        });

    d3.selectAll('text')
        .attr('dx', -4)
        .attr('dy', 4)
        .attr('x', function(d) {
          return d.x
        })
        .attr('y', function(d) {
          return d.y
        });
}
```

Ex09-01

- Step3: create a callback function to update SVG positions after every “tick”
  - The simulator iteratively calculates the element positions and try to make all elements positions stable
  - The simulator automatically call the function (in .on()) to update states of elements

```
var simulation = d3.forceSimulation(nodes)
    .force('charge', d3.forceManyBody().strength(10))
    .force('center', d3.forceCenter(width / 2, height / 2))
    .force('collision', d3.forceCollide().radius(function(d) {
            return value2radius(d.value);
    }))
    .on('tick', ticked);

function ticked() {
    d3.selectAll('circle')
        .attr('cx', function(d) {
          return d.x
        })
        .attr('cy', function(d) {
          return d.y
        });

    d3.selectAll('text')
        .attr('dx', -4)
        .attr('dy', 4)
        .attr('x', function(d) {
          return d.x
        })
        .attr('y', function(d) {
          return d.y
        });
}
```

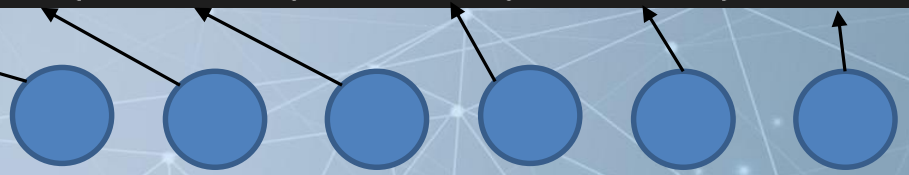
‘tick’: call the function, ticked after every iteration

What are the data bound with these circles? what is ‘d’ here?

Update circle’s states (position)

Ex09-01

```
var nodes = [{value: 5}, {value: 10}, {value: 2}, {value: 6}, {value: 7}, {value: 8}];
```

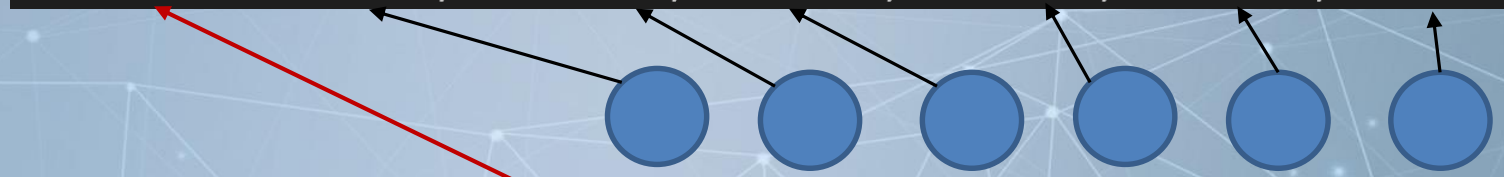


```
var circles = d3.select('svg')
    .selectAll('circle')
    .data(nodes)
    .enter()
    .append('circle')
    .attr('r', function(d) {
        return value2radius(d.value);
    });
```

When you bind data to circles, each circle element just points to a data element of the nodes array

Ex09-01

```
var nodes = [{value: 5}, {value: 10}, {value: 2}, {value: 6}, {value: 7}, {value: 8}];
```



```
var simulation = d3.forceSimulation(nodes)
    .force('charge', d3.forceManyBody().strength(10))
    .force('center', d3.forceCenter(width / 2, height / 2))
    .force('collision', d3.forceCollide().radius(function(d) {
            return value2radius(d.value);
    }))
    .on('tick', ticked);
```

When you bind the data array to the simulator, the simulator also just points to the data array.

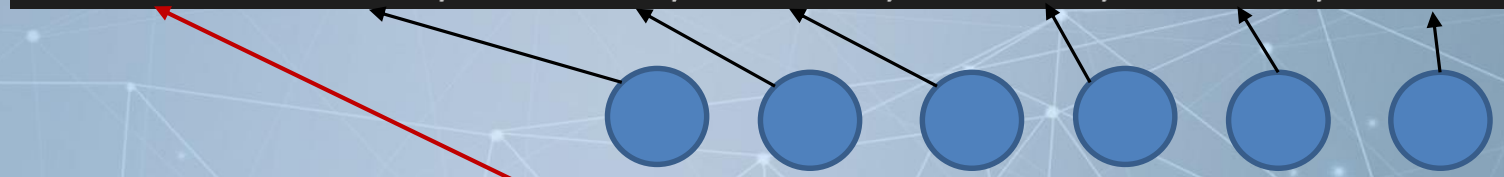
So, if anyone modifies the data array, 'nodes', the data which bind on the circles and the simulator will change as well

```
var circles = d3.select('svg')
    .selectAll('circle')
    .data(nodes)
    .enter()
    .append('circle')
    .attr('r', function(d) {
        return value2radius(d.value);
    });
```

When you bind data to circles, each circle element just points to a data element of the nodes array

Ex09-01

```
var nodes = [{value: 5}, {value: 10}, {value: 2}, {value: 6}, {value: 7}, {value: 8}];
```



```
var simulation = d3.forceSimulation(nodes)
    .force('charge', d3.forceManyBody().strength(10))
    .force('center', d3.forceCenter(width / 2, height / 2))
    .force('collision', d3.forceCollide().radius(function(d) {
            return value2radius(d.value);
    }))
    .on('tick', ticked);
```

When you bind the data array to the simulator, the simulator also just points to the data array.

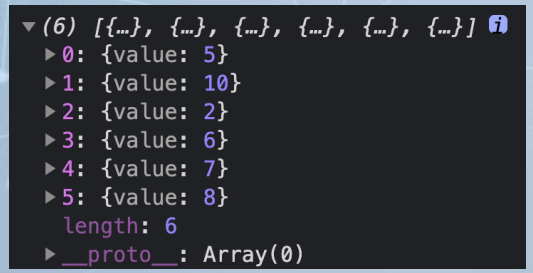
So, if anyone modifies the data array, 'nodes', the data which bind on the circles and the simulator will change as well

```
var circles = d3.select('svg')
    .selectAll('circle')
    .data(nodes)
    .enter()
    .append('circle')
    .attr('r', function(d) {
        return value2radius(d.value);
    });
```

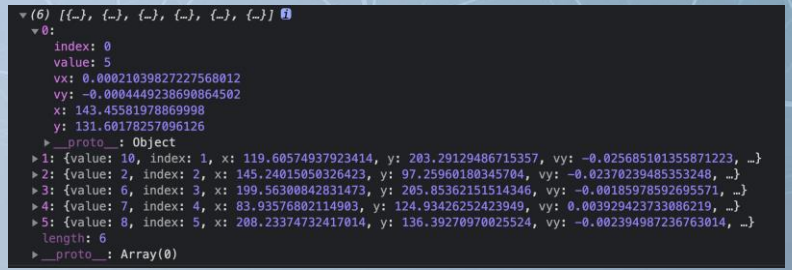
When you bind data to circles, each circle element just points to a data element of the nodes array

Ex09-01

- How does the simulator tell you the information from its force simulation algorithm?
  - When running the simulation, the simulator modifies the **data array** (add more key-value into your data array) to tell you where you show display your circles
    - Index: index is index…
    - x, y: x and y position now
    - vx, vy: x and y velocity now



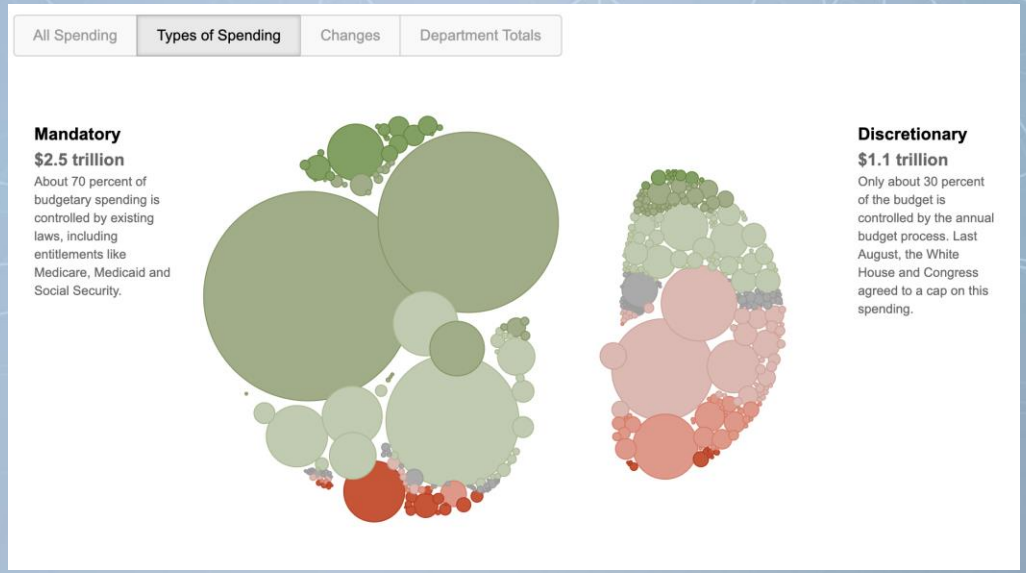
Without the simulation:
console.log(nodes)



With the simulation and after run it:
console.log(nodes)

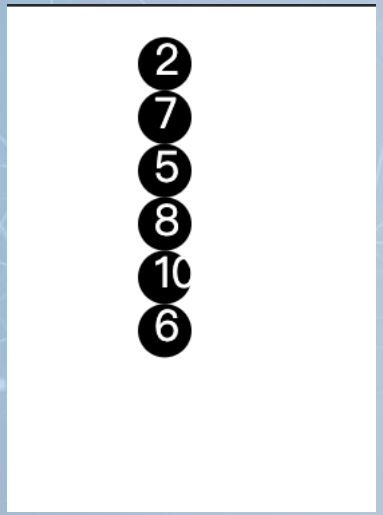
forceX and forceY

- Group different categories on the page
- Causing elements to be attracted towards a certain x/y position



Ex09-02

- main.js
- Modified from Ex09-01




```
var width = 300, height = 300;
var nodes = [{value: 5}, {value: 10}, {value: 2},
             {value: 6}, {value: 7}, {value: 8}];

var nodeRadius = 10;

var circles = d3.select('svg')
    .selectAll('circle')
    .data(nodes)
    .enter()
    .append('circle')
    .attr('r', nodeRadius);

var texts = d3.select('svg')
    .selectAll('text')
    .data(nodes)
    .enter()
    .append('text')
    .attr('fill', 'white')
    .text(function(d) {
        return d.value;
    });
```

All the nodes will be attracted to x = 200

Ex09-02

- main.js
- Modified from Ex09-01

```
var xForce = d3.forceX().x(200).strength(1);
var yForce = d3.forceY().y(function(d){
        return d.value*10;
    });

var simulation = d3.forceSimulation(nodes)
    .force('x', xForce)
    .force('y', yForce)
    .force('collision', d3.forceCollide().radius(nodeRadius))
    .on('tick', ticked);

function ticked() {
    d3.selectAll('circle')
        .attr('cx', function(d) {
          return d.x
        })
        .attr('cy', function(d) {
          return d.y
        });

    d3.selectAll('text')
        .attr('dx', -4)
        .attr('dy', 4)
        .attr('x', function(d) {
          return d.x
        })
        .attr('y', function(d) {
          return d.y
        });
}
```

Each node will be attracted to different y position, which is y = “d.value*10”
d (data)??

Ex09-02

- main.js
- Modified from Ex09-01

```
var xForce = d3.forceX().x(200).strength(1);
var yForce = d3.forceY().y(function(d){
        return d.value*10;
    });

var simulation = d3.forceSimulation(nodes)
    .force('x', xForce)
    .force('y', yForce)
    .force('collision', d3.forceCollide().radius(nodeRadius))
    .on('tick', ticked);

function ticked() {
    d3.selectAll('circle')
        .attr('cx', function(d) {
          return d.x
        })
        .attr('cy', function(d) {
          return d.y
        });

    d3.selectAll('text')
        .attr('dx', -4)
        .attr('dy', 4)
        .attr('x', function(d) {
          return d.x
        })
        .attr('y', function(d) {
          return d.y
        });
}
```

Ex09-02

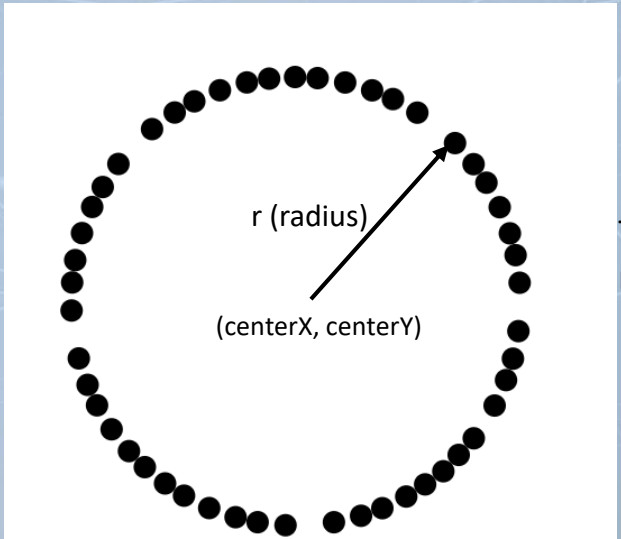
- main.js
- Modified from Ex09-01

The radius for d3.forceCollide() is fixed here (nodeRadius = 10)

```
var xForce = d3.forceX().x(200).strength(1);
var yForce = d3.forceY().y(function(d){
        return d.value*10;
    });

var simulation = d3.forceSimulation(nodes)
    .force('x', xForce)
    .force('y', yForce)
    .force('collision', d3.forceCollide().radius(nodeRadius))
    .on('tick', ticked);

function ticked() {
    d3.selectAll('circle')
        .attr('cx', function(d) {
          return d.x
        })
        .attr('cy', function(d) {
          return d.y
        });

    d3.selectAll('text')
        .attr('dx', -4)
        .attr('dy', 4)
        .attr('x', function(d) {
          return d.x
        })
        .attr('y', function(d) {
          return d.y
        });
}
```

forRadial

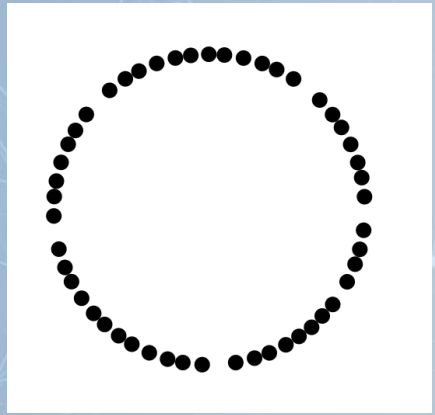
- Push every node away with a fixed distance from a circle center



The desired locations of all nodes and the center are ‘r’

Ex09-03

- main.js



```
var width = 300, height = 300;
var nodeRadius = 5;
var centerX = 150, centerY = 150;
var nodes = d3.range(50).map(function(d) {
    return {};
});

var circles = d3.select('svg')
    .selectAll('circle')
    .data(nodes)
    .enter()
    .append('circle')
    .attr('r', nodeRadius);
```

Bind data and append 50 circles to svg

Create an array with 50 elements and each data element is a empty dictionary (we do not need value in this example)

Ex09-03

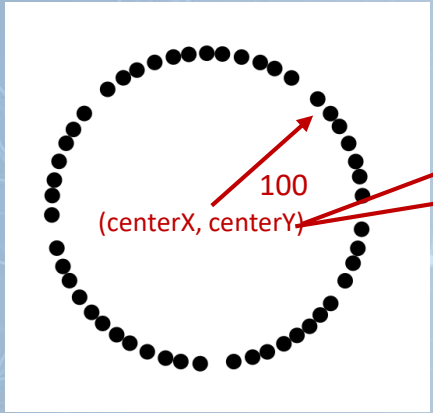
- main.js

Our radial force

```
var radialForce = d3.forceRadial().radius(100);

var simulation = d3.forceSimulation(nodes)
    .force('r', radialForce)
    .force('collision', d3.forceCollide().radius(nodeRadius))
    .on('tick', ticked);

function ticked() {
    d3.selectAll('circle')
        .attr('cx', function(d) {
          return d.x + centerX;
        })
        .attr('cy', function(d) {
          return d.y + centerY;
        });
}
```

Collide force

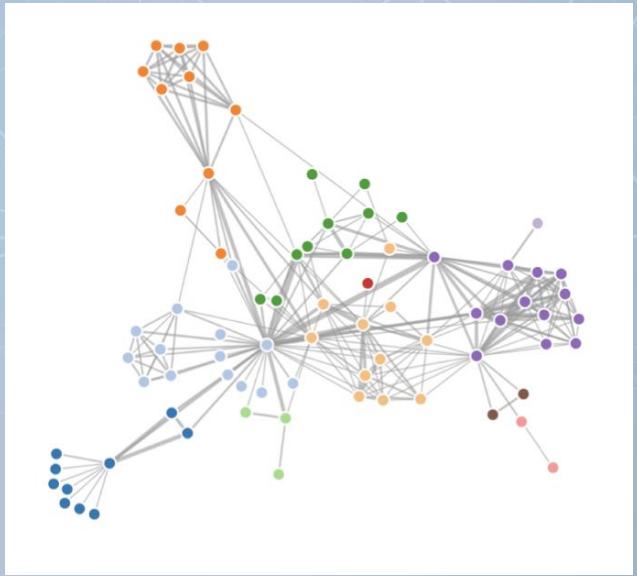
Default center is (0,0)



forceLink

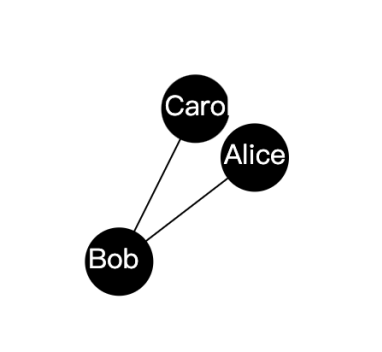
- Force certain nodes on the graph with certain distance apart

- simulation.force(‘link’, d3.forcelink(). link(links))



Ex09-04

- main.js



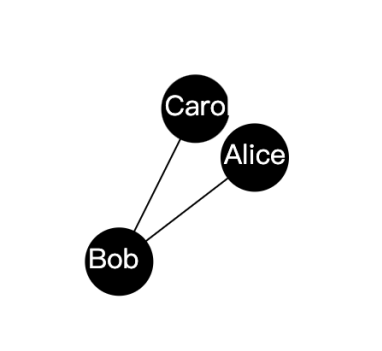
The way to define the node data is the same as usual, array of dictionary.
We assign ”id” for each node to recognize them by link later

Links of the graph are also defined by an array of dictionary. Each dictionary is a link with two keys, ‘source’ and ‘target’. The values of ‘source’ and ‘target’ are two nodes of the link.
You can also use ‘index’ instead of node id to indicate nodes (please check d3 document)

```
var width = 300, height = 300;
var nodeRadius = 20;

var nodes = [
  {"id": "Alice"},
  {"id": "Bob"},
  {"id": "Carol"}
];

var links = [
  {"source": "Alice", "target": "Bob"},
  {"source": "Bob", "target": "Carol"}
];

var lines = d3.select('svg')
    .selectAll('line')
    .data(links)
    .enter()
    .append('line')
    .attr('stroke', 'black');
var circles = d3.select('svg')
    .selectAll('circle')
    .data(nodes)
    .enter()
    .append('circle')
    .attr('r', nodeRadius);
var texts = d3.select('svg')
    .selectAll('text')
    .data(nodes)
    .enter()
    .append('text')
    .attr('fill', 'white')
    .text(function(d) {
        return d.id;
    });
```

Ex09-04

- main.js

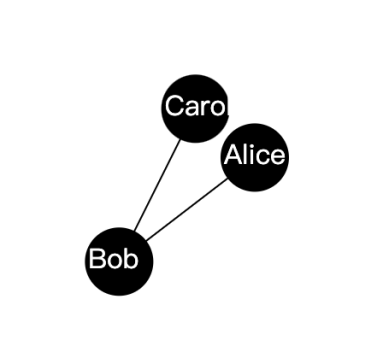


Add nodes, texts, and links to svg

```
var width = 300, height = 300;
var nodeRadius = 20;

var nodes = [
  {"id": "Alice"},
  {"id": "Bob"},
  {"id": "Carol"}
];

var links = [
  {"source": "Alice", "target": "Bob"},
  {"source": "Bob", "target": "Carol"}
];

var lines = d3.select('svg')
    .selectAll('line')
    .data(links)
    .enter()
    .append('line')
    .attr('stroke', 'black');
var circles = d3.select('svg')
    .selectAll('circle')
    .data(nodes)
    .enter()
    .append('circle')
    .attr('r', nodeRadius);
var texts = d3.select('svg')
    .selectAll('text')
    .data(nodes)
    .enter()
    .append('text')
    .attr('fill', 'white')
    .text(function(d) {
        return d.id;
    });
```

Ex09-04

- main.js



The link force

```
var linkForce = d3.forceLink()
    .links(links)
    .id(function(d) {
        return d.id;
    })
    .distance(100);

var simulation = d3.forceSimulation(nodes)
    .force('link', linkForce)
    .force('center', d3.forceCenter(width / 2, height / 2))
    .force('collision', d3.forceCollide().radius(nodeRadius))
    .on('tick', ticked);
```

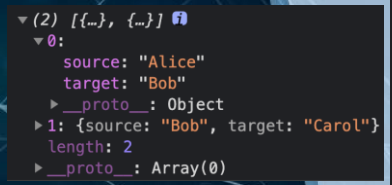
Link information

Define node ID accessor

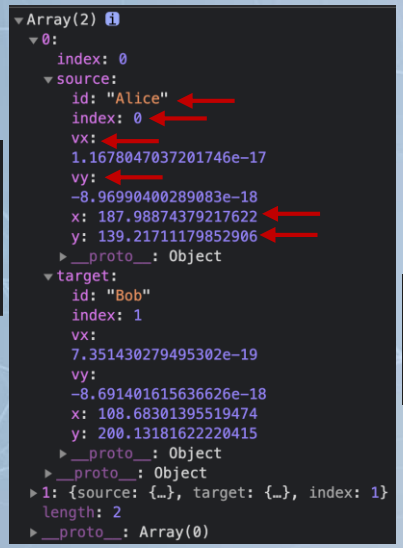
"Desired" length of a link

Ex09-04

- main.js



Without simulator
Console.log(links)



with simulator
Console.log(links)

The simulator modifies the values of source and target to {id, index, vx, vy, x, y}

Update line positions
The concept is the same as updating node position in Ex09-01
The simulator will modify the data bound to d3.forceLink()

```
var lines = d3.select('svg')
    .selectAll('line')
    .data(links)
    .enter()
    .append('line')
    .attr('stroke', 'black');
```

```
var linkForce = d3.forceLink()
    .links(links)
    .id(function(d) {
        return d.id;
    })
    .distance(100);
```

Update node/text positions

```
function ticked() {
    d3.selectAll('circle')
        .attr('cx', function(d) {
          return d.x
        })
        .attr('cy', function(d) {
          return d.y
        });

    d3.selectAll('text')
        .attr('dx', -18)
        .attr('dy', 4)
        .attr('x', function(d) {
          return d.x
        })
        .attr('y', function(d) {
          return d.y
        });

    d3.selectAll('line')
        .attr('x1', function(d) {
          return d.source.x
        })
        .attr('y1', function(d) {
          return d.source.y
        })
        .attr('x2', function(d) {
          return d.target.x
        })
        .attr('y2', function(d) {
          return d.target.y
        });
}
```