

# **This Is Not An Introduction to d3.js**

**Jennifer Piscionere**

**@jpiscionere**

**<http://jpiscionere.github.io/>**

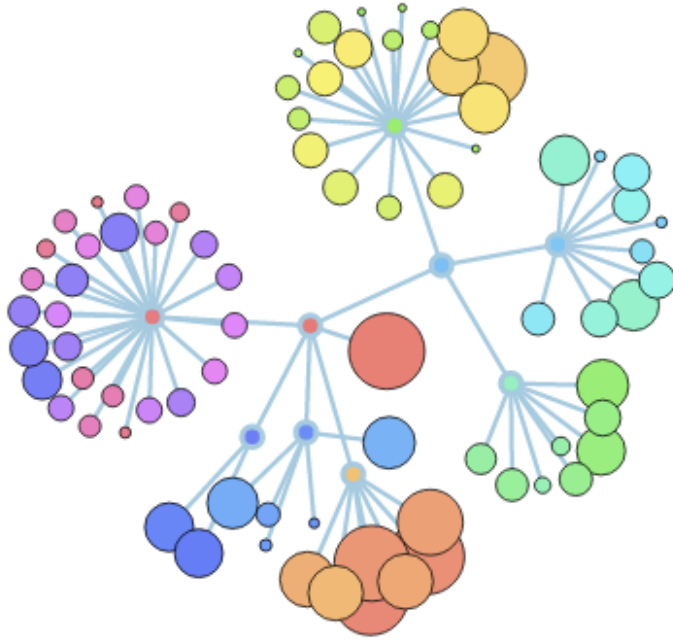
**This Is An Introduction to  
Seeing Something That Looks  
Cool on the Internet and  
Ripping it Off**

**This Is An Introduction to  
Seeing Something That Looks  
Cool on the Internet and  
Ripping it Off**

# **This Is An Introduction to Seeing Something That Looks Cool on the Internet and Ripping it Off**

<https://github.com/mbostock/d3/wiki/Gallery>

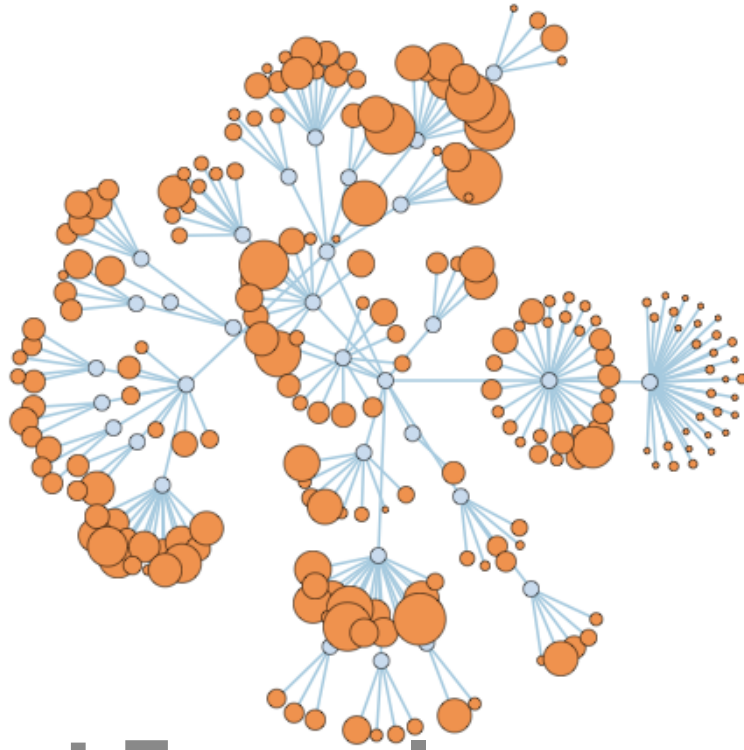
**You've Already Done  
Koalas to the Max Right?**



Final Project

**MAKE YOUR OWN CODEFLOWER**

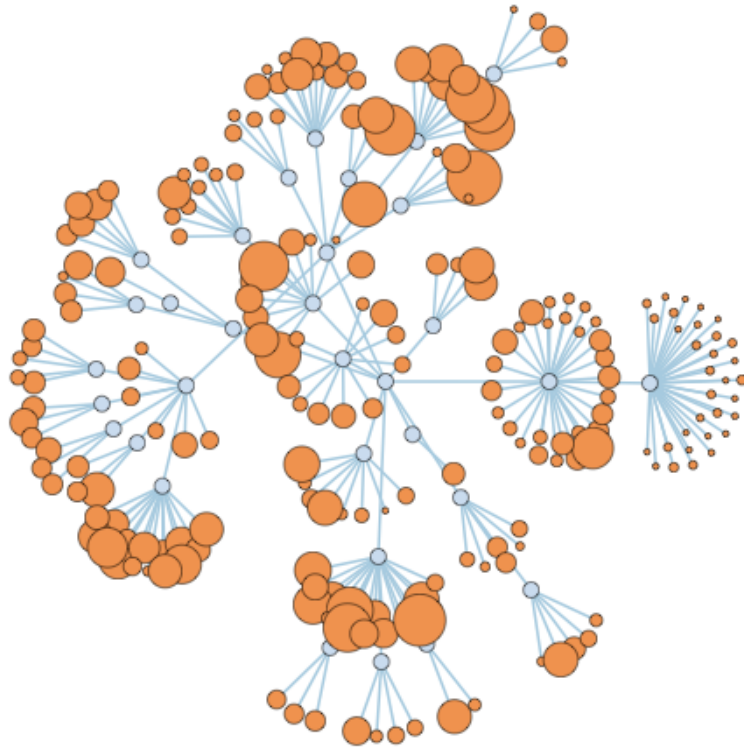
<http://www.redotheweb.com/CodeFlower/>



First Example

**FORCE COLLAPSIBLE**

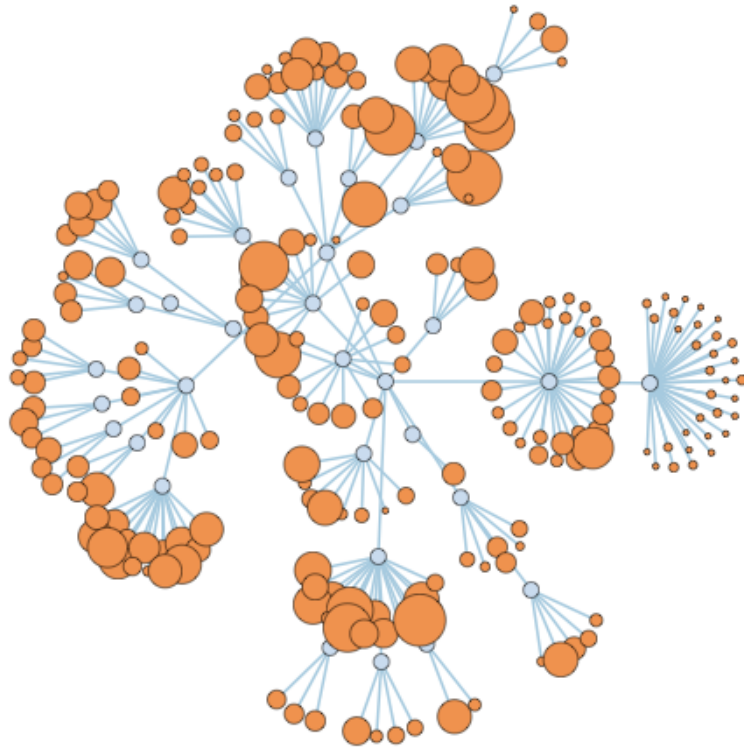
<http://www.redotheweb.com/CodeFlower/>



Step the First

**SAVE FORCE COLLAPSIBLE  
WEBPAGE USING FIREFOX\*  
ON YOUR LOCAL MACHINE**

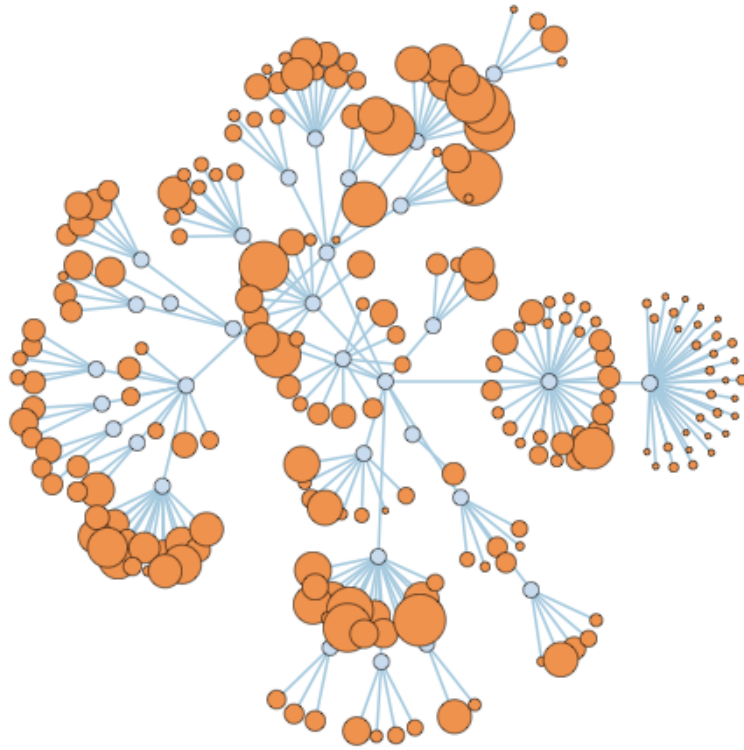




Step the First  
**SAVE FORCE COLLAPSIBLE  
WEBPAGE USING FIREFOX\*  
ON YOUR LOCAL MACHINE**

\*This is non-negotiable for the time being.

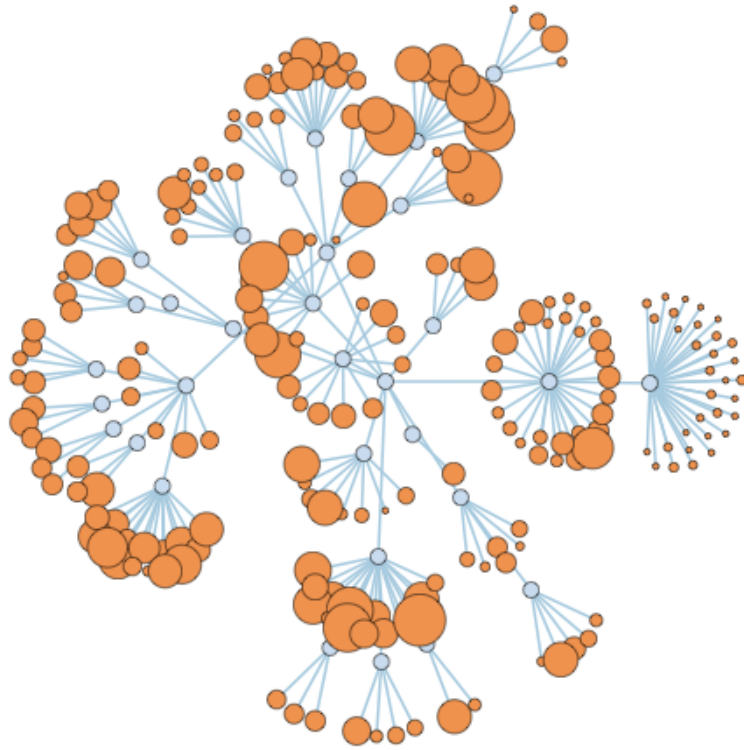
OMG I KNOW chrome/safari/opera/butterfly wings interpreting html IS SO MUCH BETTER



Step the Second

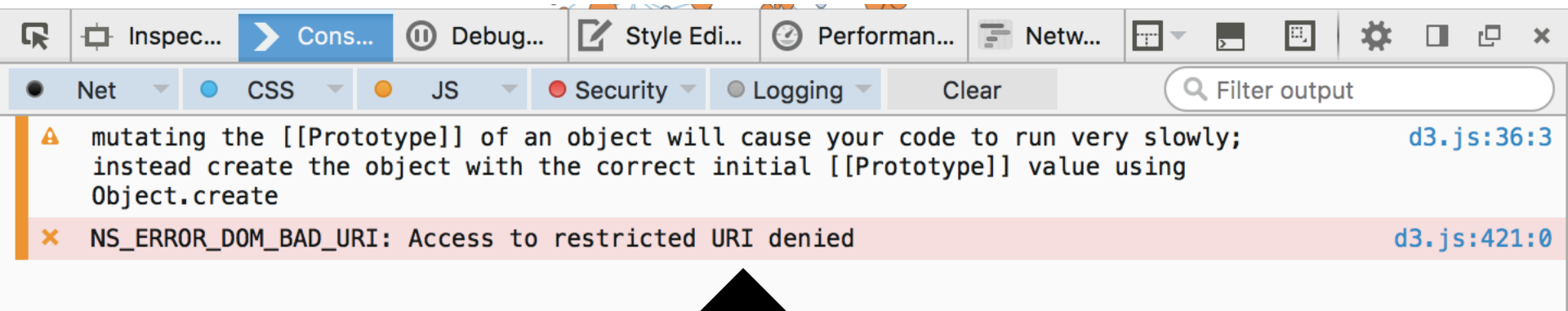
**OPEN FORCE-COLLAPSIBLE.HTML  
USING FIREFOX**

**Why Doesn't My Flower Work?**

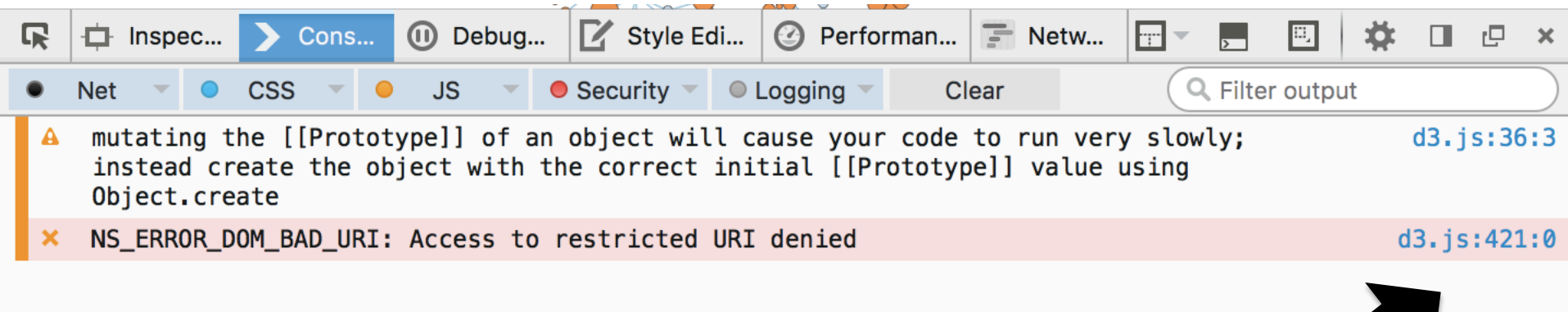


Step the Third

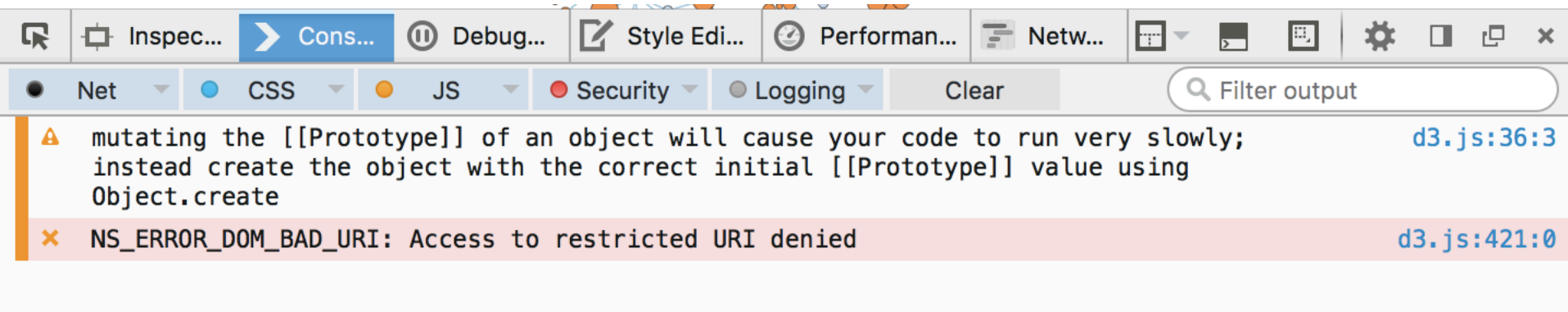
**OPEN THE CONSOLE TO SEE THE  
ERRORS**



**red == not good**



**Not helpful**



**Doctor Google Says:**  
**This Error Means There is A Missing File**

Step the Fourth

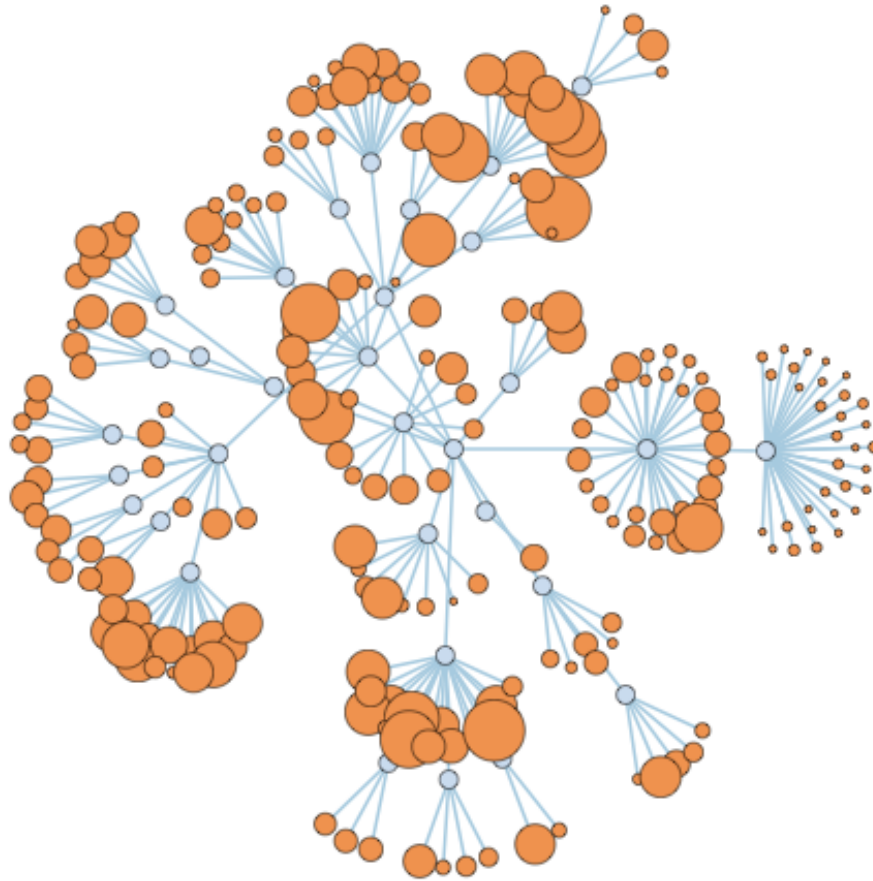
**OPEN FORCE-COLLAPSIBLE.HTML  
USING THE TEXT EDITOR  
OF YOUR CHOICE (!!!)**



# Lines 1:20 HTML / CSS / SVG

```
1 <!DOCTYPE html>
2 <html><head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4     <link type="text/css" rel="stylesheet" href="force-collapsible_files/style.css">
5     <style type="text/css">
6
7     circle.node {
8         cursor: pointer;
9         stroke: #000;
10        stroke-width: .5px;
11    }
12
13    line.link {
14        fill: none;
15        stroke: #9ecae1;
16        stroke-width: 1.5px;
17    }
18
19    </style>
20 </head>
```





**What it Says? HTML**

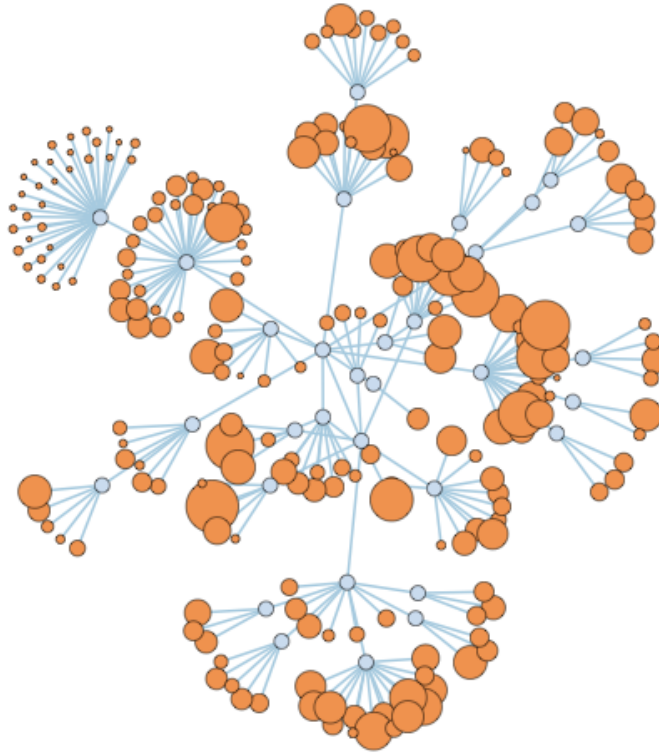
**How it Looks? CSS**

Flare code size  
force-directed graph

Flare code size  
force-directed graph



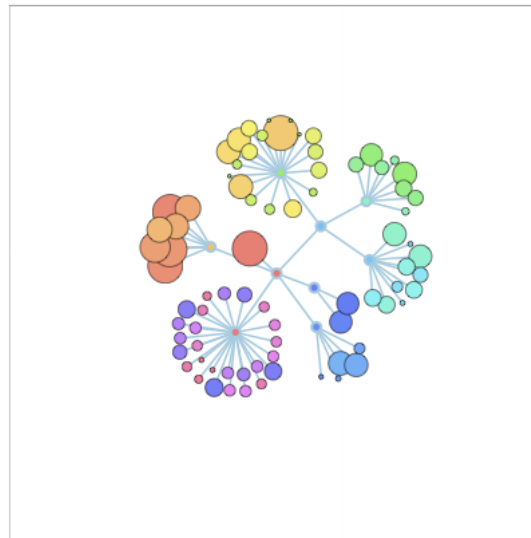
No CSS



## Jennifer A. Piscionere

This website is under [construction](#).

Check out my [codeflower](#) instead. I have a [wix page](#) with some more info.



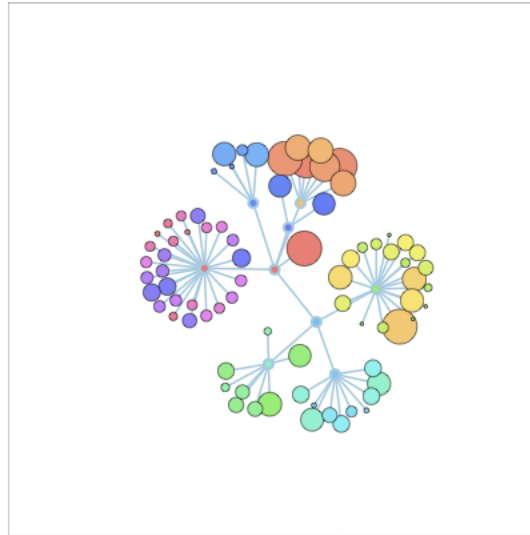
# My Website **With** the Style Sheets

Toggle navigation [JenPi](#)

- [Home](#)
- [CV.pdf](#)
- [New Paper](#)
- [Contact](#)
- [Scatter Matrix](#)

## Jennifer A. Piscionere

This website is under [construction](#).  
Check out my [codeflower](#) instead. I have a [wix page](#) with some more info.




lpiSH- ə-nerēl  
*Like Pictionary without the t*

# My Website **Without** the Style Sheets

# Lines 1:20 HTML / CSS / SVG HEADER

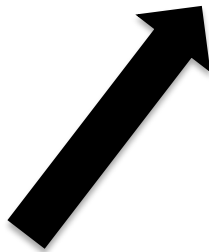
```
1 <!DOCTYPE html>
2 <html><head>
3   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4   <link type="text/css" rel="stylesheet" href="force-collapsible_files/style.css">
5   <style type="text/css">
6
7   circle.node {
8     cursor: pointer;
9     stroke: #000;
10    stroke-width: .5px;
11  }
12
13  line.link {
14    fill: none;
15    stroke: #9ecae1;
16    stroke-width: 1.5px;
17  }
18
19 </style>
20 </head>
```



**Scalable Vector Graphic Attributes**  
There are Connected Circles  
How those Circles Lookish

# Lines 21:28 HTML / Loading d3 Engine

```
21 <body>
22   <h2>
23     Flare code size<br>
24     force-directed graph
25   </h2>
26   <script type="text/javascript" src="force-collapsible_files/d3.js"></script>
27   <script type="text/javascript" src="force-collapsible_files/d3_002.js"></script>
28   <script type="text/javascript" src="force-collapsible_files/d3_003.js"></script>
```



Think of it as  
`import numpy`

# Lines 29:144 d3 Javascript Details

```
29 <script type="text/javascript">
30
31 var w = 1280,
32     h = 800,
33     node,
34     link,
35     root;
36
37 var force = d3.layout.force()
38     .on("tick", tick)
39     .charge(function(d) { return d._children ? -d.size / 100 : -30; })
40     .linkDistance(function(d) { return d.target._children ? 80 : 30; })
41     .size([w, h - 160]);
42
43 var vis = d3.select("body").append("svg:svg")
44     .attr("width", w)
45     .attr("height", h);
46
47 d3.json("flare.json", function(json) {
48     root = json;
49     root.fixed = true;
50     root.x = w / 2;
51     root.y = h / 2 - 80;
```



# Lines 29:144 d3 Javascript Details

```
55 function update() {  
56     var nodes = flatten(root),  
57         links = d3.layout.tree().links(nodes);  
58  
59     // Restart the force layout.  
60     force  
61         .nodes(nodes)  
62         .links(links)  
63         .start();  
64  
65     // Update the links...  
66     link = vis.selectAll("line.link")  
67         .data(links, function(d) { return d.target.id; });  
68  
69     // Enter any new links.  
70     link.enter().insert("svg:line", ".node")  
71         .attr("class", "link")  
72         .attr("x1", function(d) { return d.source.x; })  
73         .attr("y1", function(d) { return d.source.y; })  
74         .attr("x2", function(d) { return d.target.x; })  
75         .attr("y2", function(d) { return d.target.y; });  
76  
77     // Exit any old links.  
78     link.exit().remove();
```

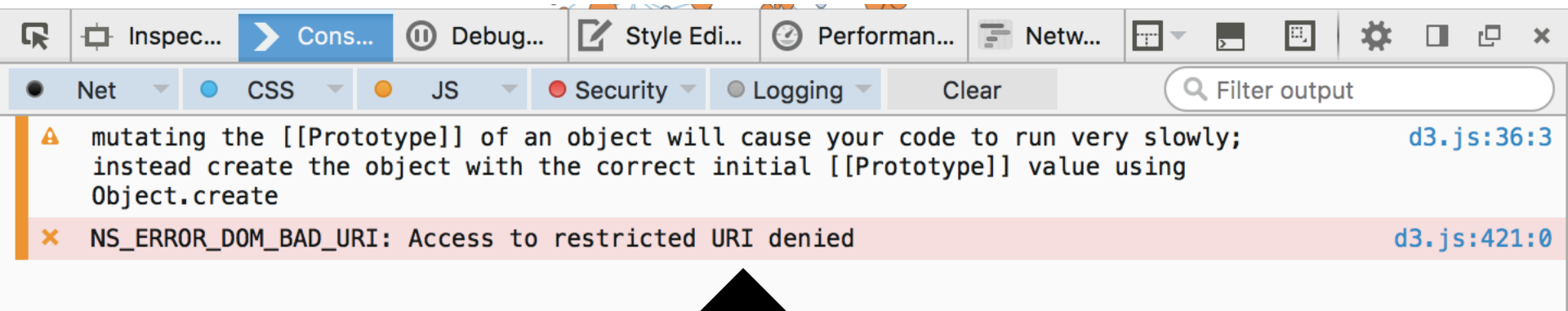
# Lines 29:144 d3 Javascript Details

```
29 <script type="text/javascript">
30
31 var w = 1280,
32     h = 800,
33     node,
34     link,
35     root;
36
37 var force = d3.layout.force()
38     .on("tick", tick)
39     .charge(function(d) { return d._children ? -d.size / 100 : -30; })
40     .linkDistance(function(d) { return d.target._children ? 80 : 30; })
41     .size([w, h - 160]);
42
43 var vis = d3.select("body").append("svg:svg")
44     .attr("width", w)
45     .attr("height", h);
46
47 d3.json("flare.json", function(json) {
48     root = json;
49     root.fixed = true;
50     root.x = w / 2;
51     root.y = h / 2 - 80;
```

**Why Doesn't My Flower Work?**

**Why Doesn't My Flower Work?**

**WHERE IS THE DATA?**



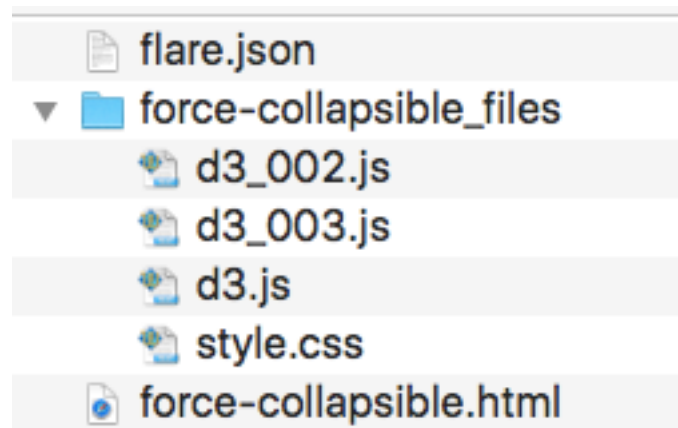
**What File Was This Talking About?**

# Lines 29:144 d3 Javascript Magic

```
29 <script type="text/javascript">
30
31 var w = 1280,
32     h = 800,
33     node,
34     link,
35     root;
36
37 var force = d3.layout.force()
38     .on("tick", tick)
39     .charge(function(d) { return d._children ? -d.size / 100 : -30; })
40     .linkDistance(function(d) { return d.target._children ? 80 : 30; })
41     .size([w, h - 160]);
42
43 var vis = d3.select("body").append("svg:svg")
44     .attr("width", w)
45     .attr("height", h);
46
47 d3.json("flare.json", function(json) {
48     root = json;
49     root.fixed = true;
50     root.x = w / 2;
51     root.y = h / 2 - 80;
```



```
$ wget http://mbostock.github.io/d3/talk/20111116/  
flare.json
```



# flare.json

```
1 {  
2   "name": "flare",  
3   "children": [  
4     {  
5       "name": "analytics",  
6       "children": [  
7         {  
8           "name": "cluster",  
9           "children": [  
10            {"name": "AgglomerativeCluster", "size": 3938},  
11            {"name": "CommunityStructure", "size": 3812},  
12            {"name": "HierarchicalCluster", "size": 6714},  
13            {"name": "MergeEdge", "size": 743}  
14          ]  
15        },  
16        {  
17          "name": "graph",  
18          "children": [  
19            {"name": "BetweennessCentrality", "size": 3534},  
20            {"name": "LinkDistance", "size": 5731},  
21            {"name": "MaxFlowMinCut", "size": 7840},  
22            {"name": "ShortestPaths", "size": 5914},  
23            {"name": "SpanningTree", "size": 3416}  
24          ]  
25        }  
26      ]  
27    }  
28  ]  
29 }
```



# flare.json

```
1 {  
2   "name": "flare",  
3   "children": [  
4     {  
5       "name": "analytics",  
6       "children": [  
7         {  
8           "name": "cluster",  
9           "children": [  
10            {  
11              "name": "HierarchicalCluster", "size": 6714},  
12              {  
13                "name": "MergeEdge", "size": 743  
14            }  
15          ],  
16        },  
17        {  
18          "name": "graph",  
19          "children": [  
20            {  
21              "name": "BetweennessCentrality", "size": 3534},  
22              {  
23                "name": "LinkDistance", "size": 5731},  
24              {  
25                "name": "MaxFlowMinCut", "size": 7840},  
26              {  
27                "name": "ShortestPaths", "size": 5914},  
28              {  
29                "name": "SpanningTree", "size": 3416  
30            }  
31          ]  
32        }  
33      ]  
34    }  
35  ]  
36 }
```

## Code Flower of d3.js itself

**Why Doesn't My Flower Work?**

# Line 144: Firefox Saved A Static Page

```
129 // Returns a list of all nodes under the root.
130 function flatten(root) {
131     var nodes = [], i = 0;
132
133     function recurse(node) {
134         if (node.children) node.size = node.children.reduce(function(p, v) { return p + recurse(v);
135         if (!node.id) node.id = ++i;
136         nodes.push(node);
137         return node.size;
138     }
139
140     root.size = recurse(root);
141     return nodes;
142 }
143
144 </script><svg height="800" width="1280"><line y2="378.5247461881192" x2="726.4042609189993"
145
146
147 </body></html>
```

**Delete :** <svg ... /svg>

**Why Does My Flower**

**STILL NOT WORK?**

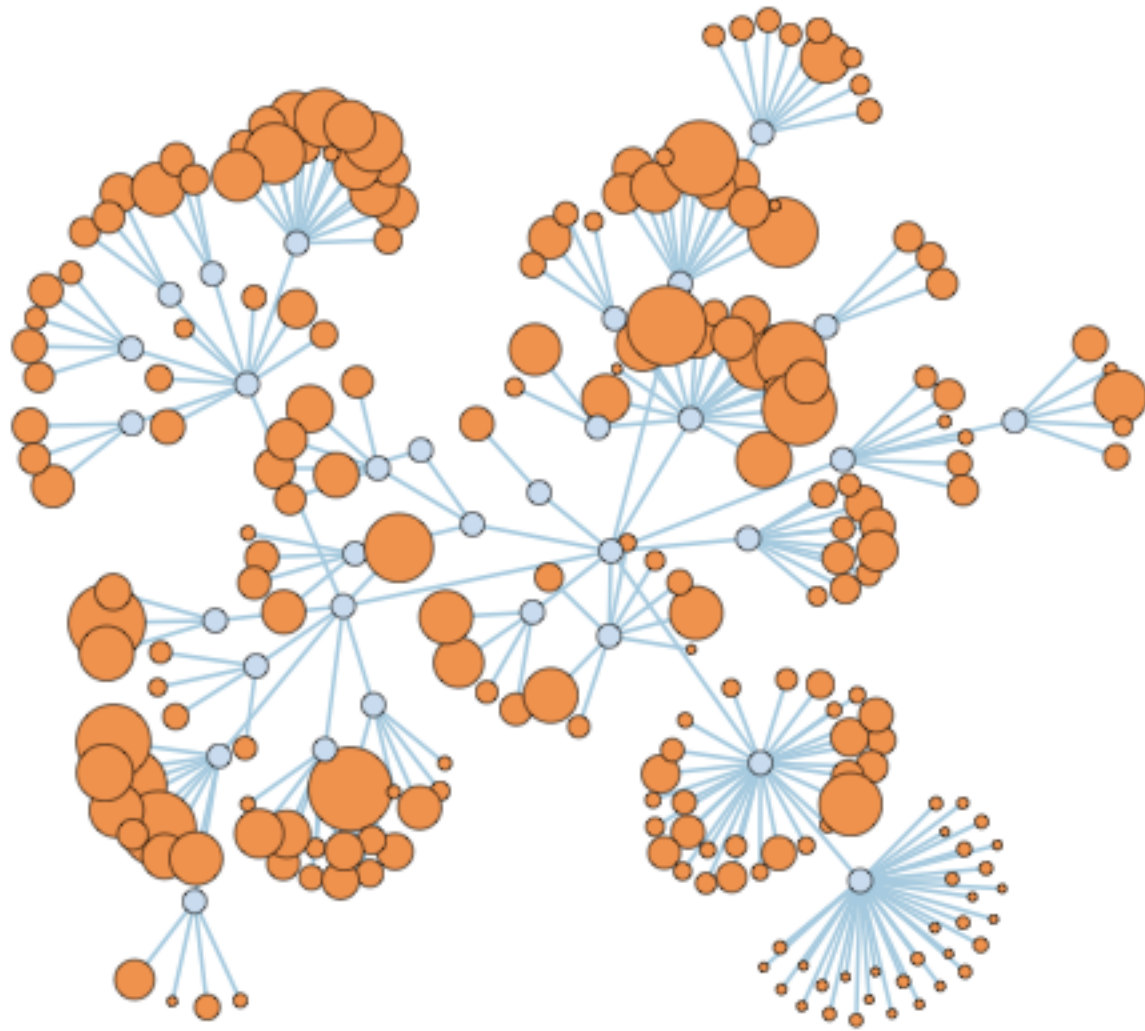
**Same Origin Policy:**

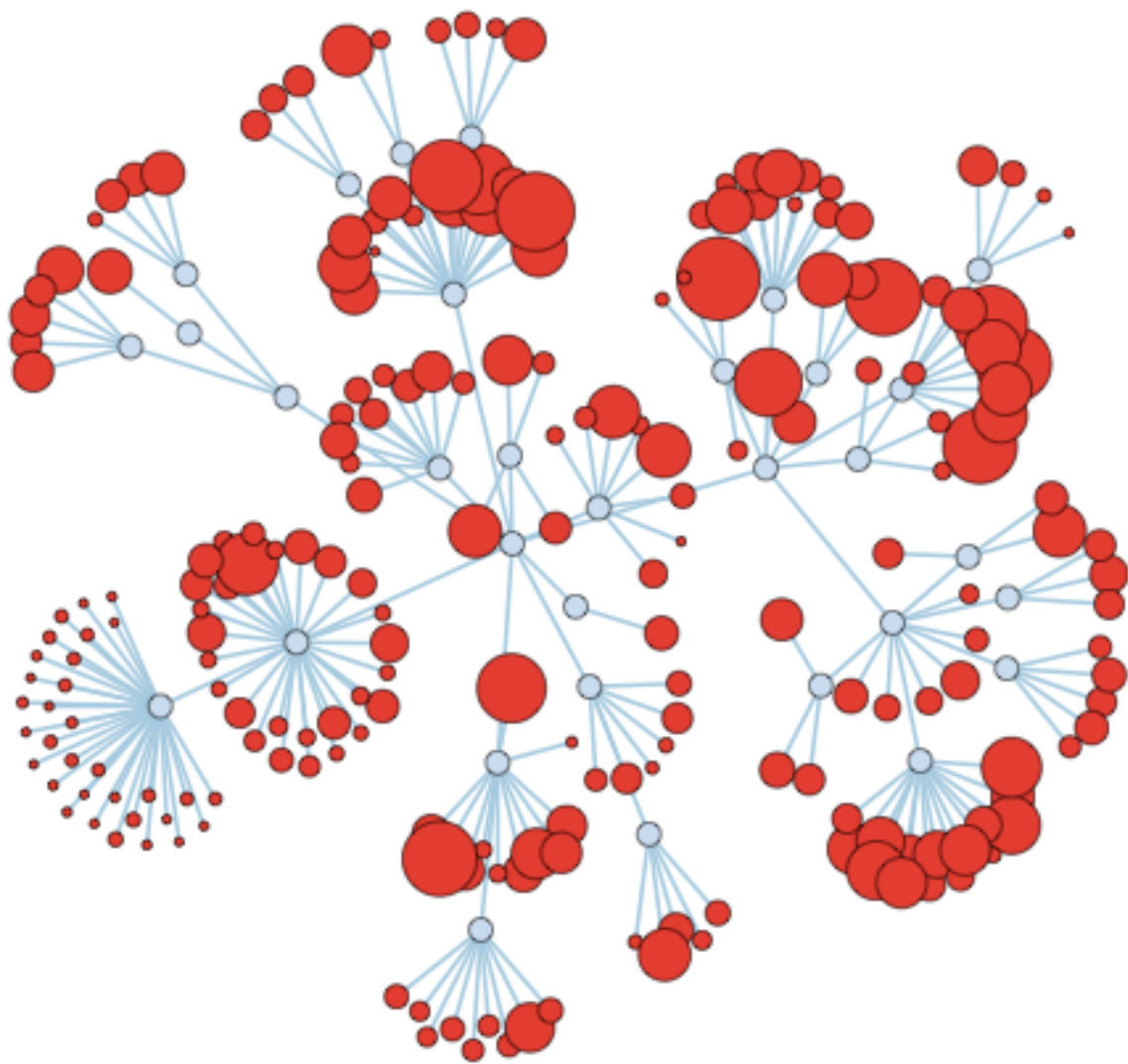
# **Why We Used Firefox to Run Things Locally**

`https://github.com/mrdoob/three.js/wiki/How-to-run-things-locally`

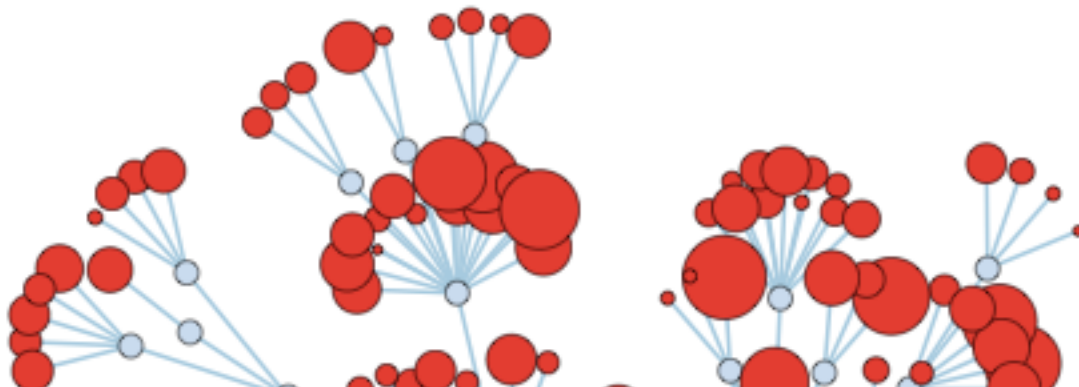
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease\  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
Pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease  
pleasepleasepleasepleasepleasepleasepleasepleasepleaseplease

**Does Everybody's Flower Work?**

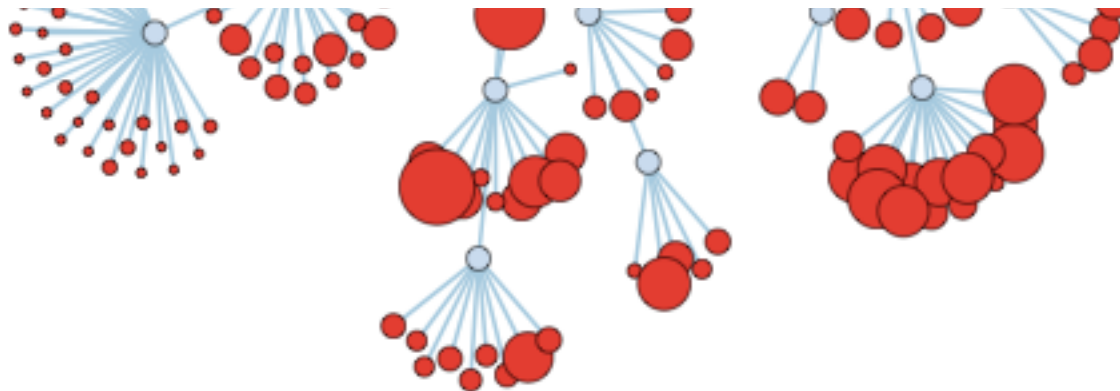








```
111  
112 // Color leaf nodes orange, and packages white or blue.  
113 ▼ function color(d) {  
114     return d._children ? "#3182bd" : d.children ? "#c6dbef" : "#F80000";  
115     ~ }  
116
```





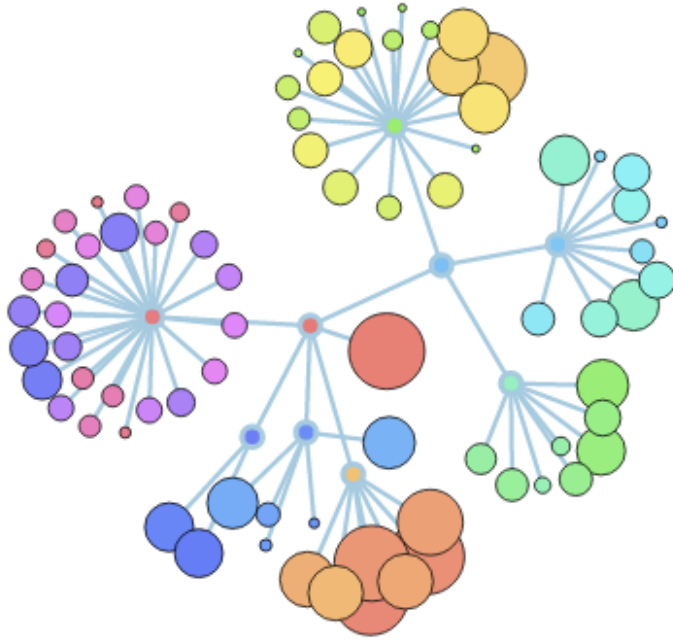


36  
37  
38  
39  
40  
41

```
var force = d3.layout.force()  
  .on("tick", tick)  
  .charge(function(d) { return d._children ? -d.size / 100 : 0; })  
  .linkDistance(function(d) { return d.target._children ? 80 : 30; })  
  .size([w, h - 160]);
```

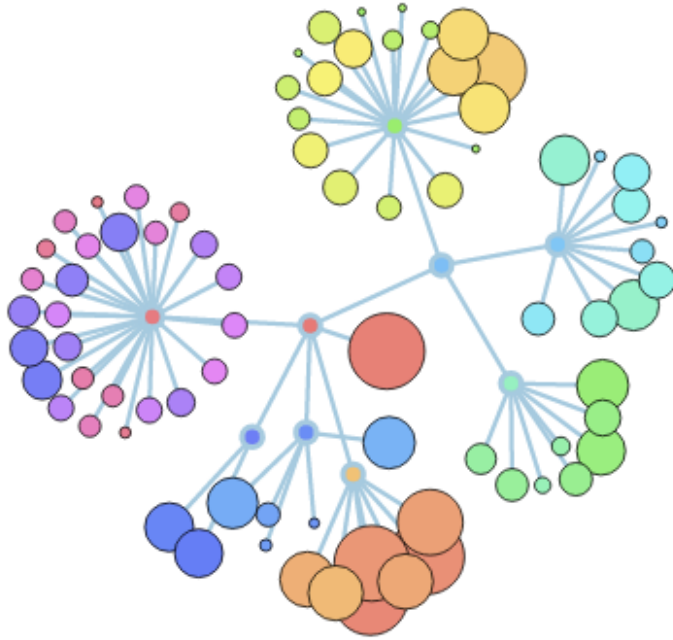
Last Step Before CODEFLOWER  
**ADD THE JAVASCRIPT AS AN  
EXTERNAL LIBRARY**

```
1 <!DOCTYPE html>
2 <html><head>
3   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4   <link type="text/css" rel="stylesheet" href="force-collapsible_files/style.css">
5   <style type="text/css">
6
7   circle.node {
8     cursor: pointer;
9     stroke: #000;
10    stroke-width: .5px;
11  }
12
13  line.link {
14    fill: none;
15    stroke: #9ecae1;
16    stroke-width: 1.5px;
17  }
18
19  </style>
20 </head>
21 <body>
22   <h2>
23     Flare code size<br>
24     force-directed graph
25   </h2>
26   <script type="text/javascript" src="force-collapsible_files/d3.js"></script>
27   <script type="text/javascript" src="force-collapsible_files/d3_002.js"></script>
28   <script type="text/javascript" src="force-collapsible_files/d3_003.js"></script>
29   <script type="text/javascript" src="force-collapsible_files/new_script.js"></script>
30
31 </body></html>
```



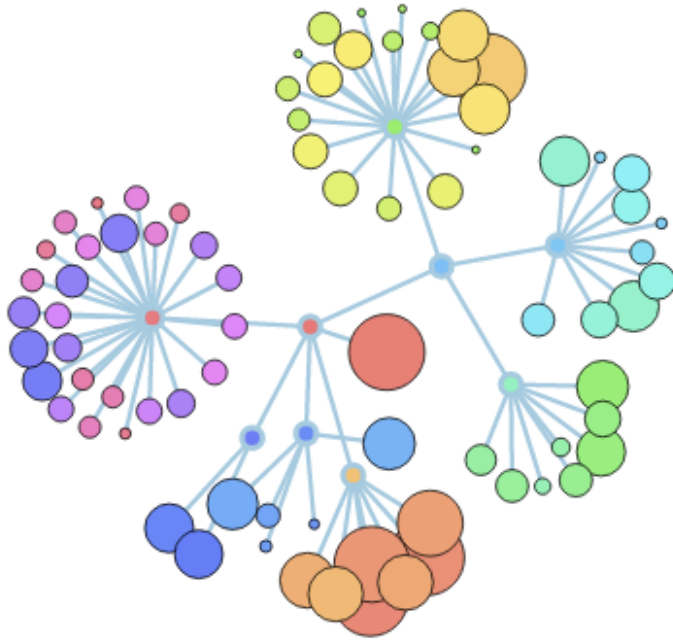
Final Project

**MAKE YOUR OWN CODEFLOWER**



Step the First

**GO TO CODEFLOWER WEBSITE**

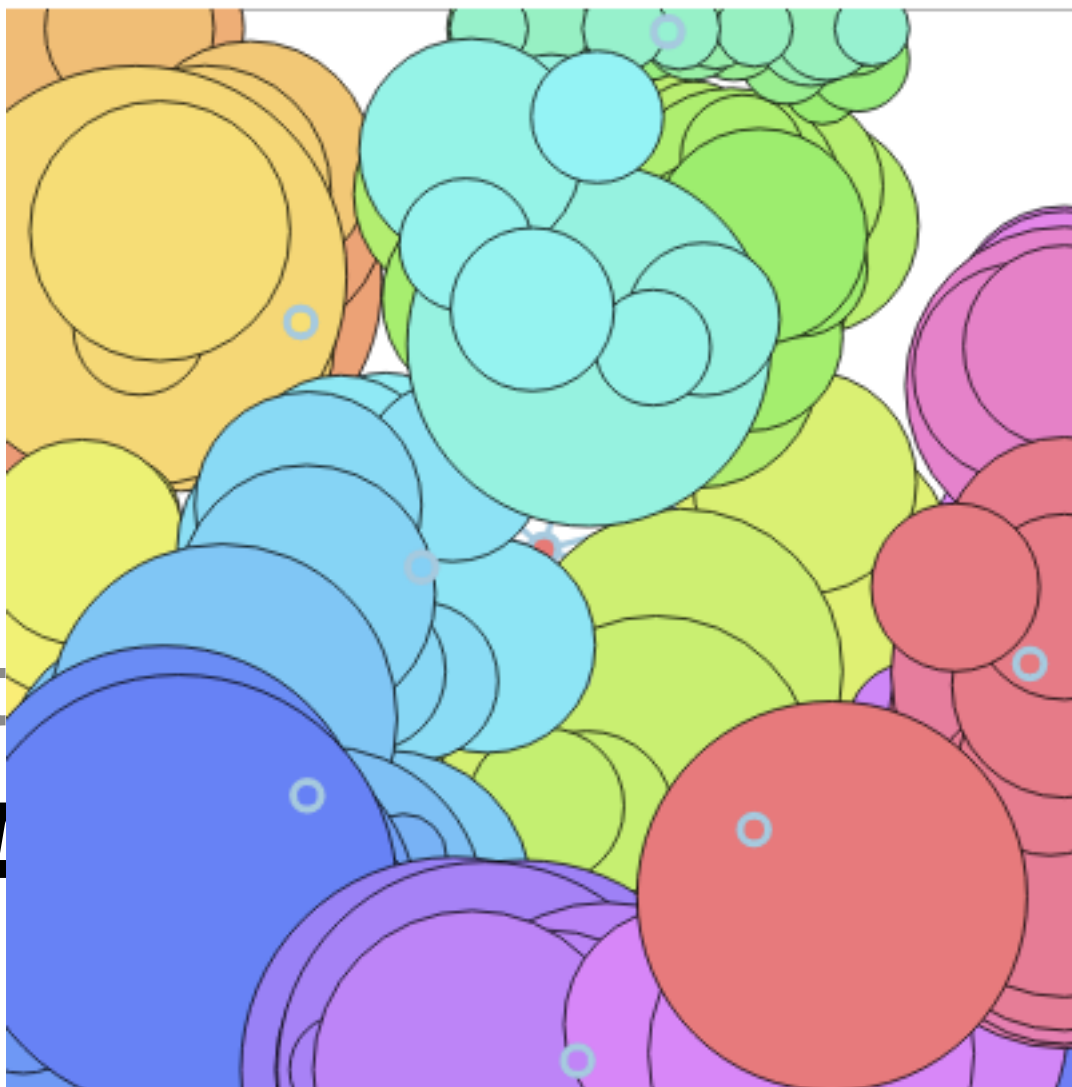


Step the Second

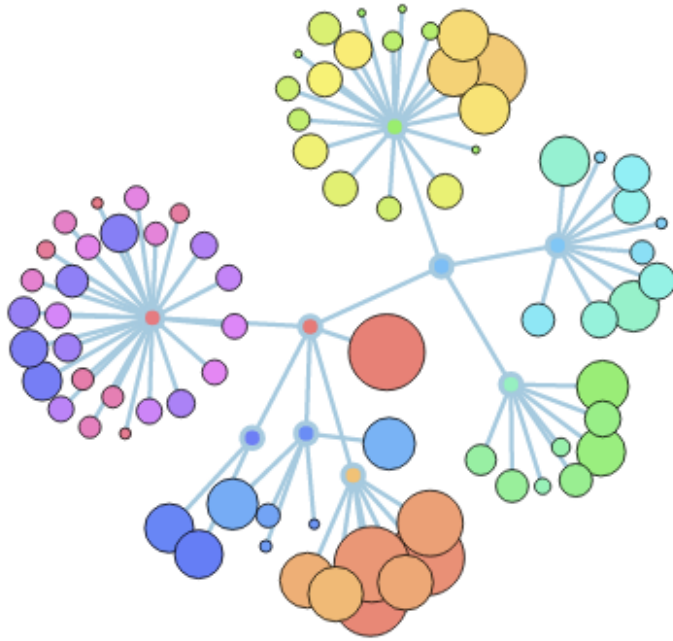
**UPDATE USING FLARE.JSON**



Step 1  
**UPDA**



**ON**



Step the 2<sup>nd</sup>.5

**USING YOUR OWN GITHUB REPO  
AS THE SOURCE OF DATA**

# 1. Install cloc

```
npm install -g cloc           # https://www.npmjs.com/package/cloc
sudo apt-get install cloc     # Debian, Ubuntu
sudo yum install cloc         # Red Hat, Fedora
sudo pacman -S cloc           # Arch
sudo pkg install cloc         # FreeBSD
sudo port install cloc        # Mac OS X with MacPorts
```

# 1. Install cloc

```
npm install -g cloc           # https://www.npmjs.com/package/cloc
sudo apt-get install cloc     # Debian, Ubuntu
sudo yum install cloc         # Red Hat, Fedora
sudo pacman -S cloc           # Arch
sudo pkg install cloc         # FreeBSD
sudo port install cloc        # Mac OS X with MacPorts
```

## 2. Count the Lines in Your Repo

**Your Repo Here**



```
# Using curl and cloc (fast, accurate)
$ curl https://nodeload.github.com/symfony/symfony/tar.gz/master | tar xzv
$ cloc symfony-master --csv --by-file --report-file=symfony.cloc
```

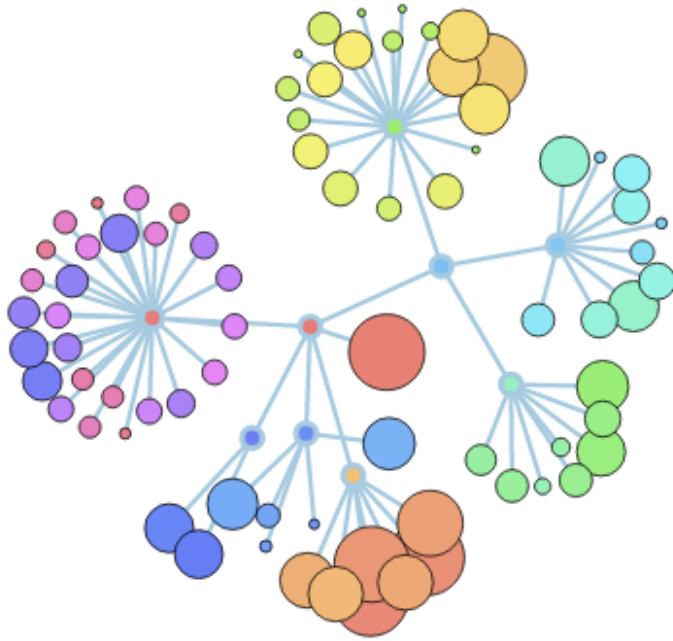
# 1. Install cloc

```
npm install -g cloc           # https://www.npmjs.com/package/cloc
sudo apt-get install cloc     # Debian, Ubuntu
sudo yum install cloc         # Red Hat, Fedora
sudo pacman -S cloc           # Arch
sudo pkg install cloc         # FreeBSD
sudo port install cloc        # Mac OS X with MacPorts
```

# 2. Count the Lines in Your Repo

```
# Using curl and cloc (fast, accurate)
$ curl https://nodeload.github.com/symfony/symfony/tar.gz/master | tar xvz
$ cloc symfony-master --csv --by-file --report-file=symfony.cloc
```

# 3. Put cloc output into .json format use widget on codeflower website



Step the Third

**DOWNLOAD CODEFLOWER  
WEBSITE**

```

36 stroke-width: 1.5px;
37 }
38 </style>
39 </head>
40 <body crossrider_data_store_temp="{>
41 <div class="content">
42 <div class="container">
43 <h1>CodeFlower Source code visualization</h1>
44 <p class="lead">This experiment visualizes source repositories using an interactive tree. Each disc represents a file, with a radius proportional to the number of lines of code.
45 <form class="form-inline">
46 <fieldset>
47 <label>Example projects from GitHub:</label>
48 <select id="project">
49 <option value="data/uptime.json">fzaninotto / uptime</option>
50 <option value="data/faker.json">fzaninotto / faker</option>
51 <option value="data/jquery.json">jquery / jquery</option>
52 <option value="data/twig.json">fabpot / twig</option>
53 <option value="data/lichess.json">ornicar / lila</option>
54 <option value="data/propel2.json">propelorm / Propel2</option>
55 <option value="data/doctrine2.json">doctrine / doctrine2</option>
56 <option value="data/wordpress.json">WordPress / WordPress</option>
57 <option value="data/rails.json">rails / rails</option>
58 <option value="data/symfony.json">symfony / symfony (WARNING: will make your computer scream)</option>
59 <option value="data/zf2.json">zendframework / zf2 (WARNING: will make your computer scream)
60 </option></select>
61 </fieldset>
62 </form>
63 <div id="visualization"><svg width="270" height="270"><rect width="270" height="270" style="stroke: rgb(153, 153, 153); fill: rgb(255, 255, 255);
64 <h2>Purpose</h2>
65 <ul class="unstyled">
66 <li>Did you ever dive into an existing project and wish you could have a bird's eye view of the whole code?</li>
67 <li>Did you ever have to refactor a large application and wish you knew where to start?</li>
68 <li>Did you ever look for a visualization that would help you communicate visually with your teammates about a repository?</li>
69 </ul>
70 <p>CodeFlowers tries to answer these needs. Also, it tries to make code look beautiful, but that's another story.</p>
71 <h2>Usage</h2>
72 <p>To create a CodeFlower, include the <code>CodeFlower.js</code> file together with <code>d3.js</code>, just like in this page. Create a new CodeFlower object:
73 <pre>var myFlower = new CodeFlower("#visualization", 300, 200);
74 myflower.update(jsonData);
75 </pre>
76 <h2>Input data format</h2>
77 <p>The <code>jsonData</code> format taken as input to <code>update()</code> is extremely simple. It's a JavaScript object representing a file tree.
78 <form id="jsonInput">
79 <fieldset>
80 <textarea id="jsonData"></textarea>

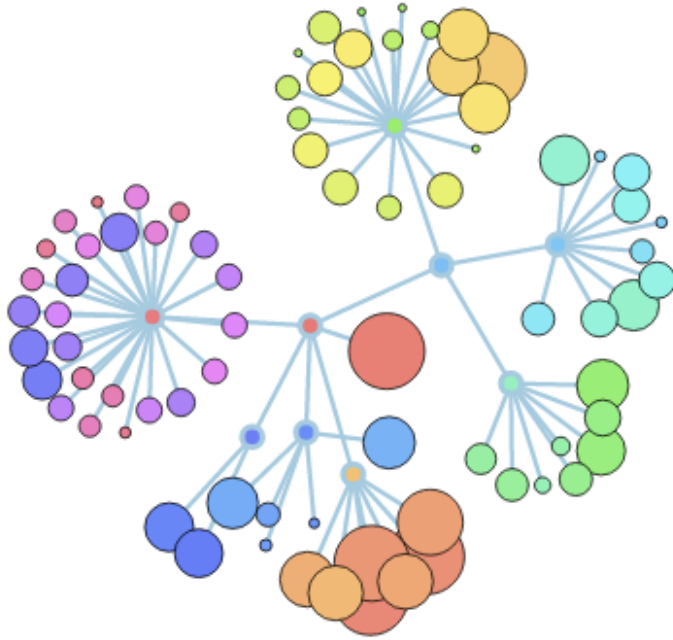
```

```

36 stroke-width: 1.5px;
37 }
38 </style>
39 </head>
40 <body crossridr_data_store_temp="{}">
41 <div class="content">
42 <div class="container">
43 <h1>CodeFlower Source code visualization</h1>
44 <p class="lead">This experiment visualizes source repositories using an interactive tree. Each disc represents a file, with a radius proportional to the number of lines of code it contains.
45 <form class="form-inline">
46 <fieldset>
47 <label>Example projects from GitHub:</label>
48 <select id="project">
49 <option value="data/uptime.json">fzantoni / uptime</option>
50 <option value="data/faker.json">fzaninotto / faker</option>
51 <option value="data/jquery.json">jquery / jquery</option>
52 <option value="data/twig.json">fabpot / twig</option>
53 <option value="data/lichess.json">ornicar / lichess</option>
54 <option value="data/propel2.json">propelorm / propel2</option>
55 <option value="data/doctrine2.json">doctrine / doctrine2</option>
56 <option value="data/wordpress.json">WordPress / WordPress</option>
57 <option value="data/rails.json">rails / rails</option>
58 <option value="data/symfony.json">symfony / symfony</option>
59 <option value="data/zf2.json">zendframework / zf2</option>
60 </select>
61 </fieldset>
62 </form>
63 <div id="visualization"><svg width="270" height="270" style="stroke: rgb(153, 153, 153); fill: rgb(255, 255, 255);"/>
64 <h2>Purpose</h2>
65 <ul class="unstyled">
66 <li>Did you ever dive into an existing project and wonder how could have been the structure of the whole code?</li>
67 <li>Did you ever have to refactor a large codebase and wish you knew the history of the code?</li>
68 <li>Did you ever look for a visualization of a codebase to communicate with your teammates about a repository?</li>
69 </ul>
70 <p>CodeFlowers tries to answer these needs, by providing a way to make code look beautiful, and it's another story.</p>
71 <h2>Usage</h2>
72 <p>To create a CodeFlower, include the <code>CodeFlower.js</code> file together with <code>d3.js</code>, just like in this page. Create a new CodeFlower object and call the <code>update()</code> method.
73 <pre>var myFlower = new CodeFlower("#visualization", 300, 200);
74 myflower.update(jsonData);
75 </pre>
76 <h2>Input data format</h2>
77 <p>The <code>jsonData</code> format taken as input to <code>update()</code> is extremely simple. It's a JavaScript object representing a file tree structure.
78 <form id="jsonData">
79 <fieldset>
80 <textarea id="jsonData"></textarea>

```





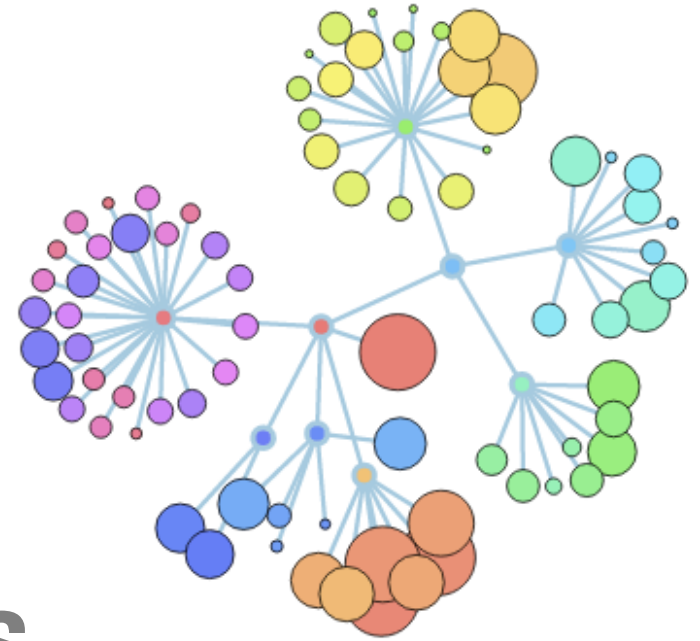
**Final Project**

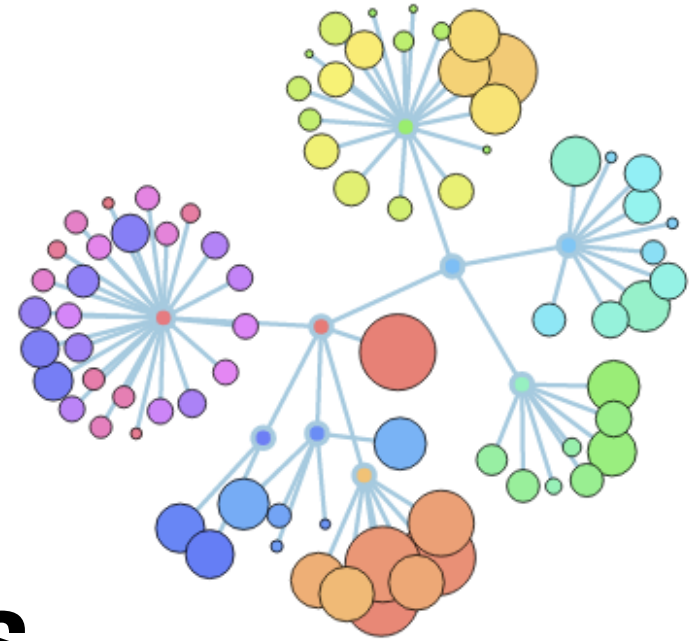
**MAKE YOUR OWN CODEFLOWER**

**Edited html lives here:**

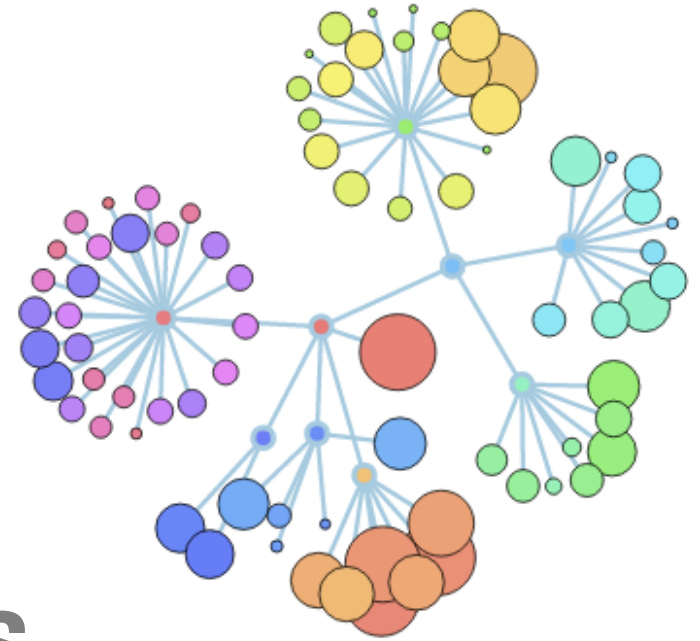
[http://jpiscionere.github.io/code\\_flower\\_try.html](http://jpiscionere.github.io/code_flower_try.html)

- 1. Save Website**
2. Check Dependencies
3. Check the Data Format
4. Check the html
5. Swap Out wtheta.js for flare.json



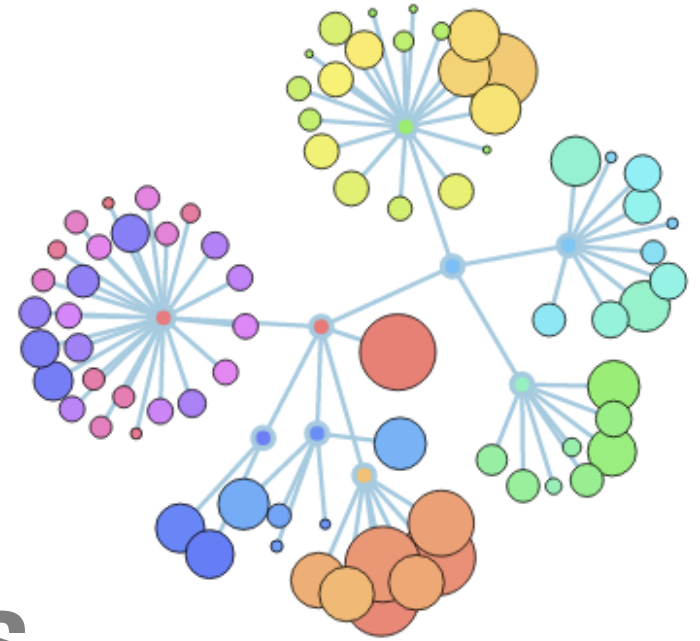


1. Save Website
- 2. Check Dependencies**
3. Check the Data Format
4. Check the html
5. Swap Out wtheta.js for flare.json



1. Save Website
2. Check Dependencies
- 3. Check the Data Format**
4. Check the html
5. Swap Out wtheta.js for flare.json

1. Save Website
2. Check Dependencies
3. Check the Data Format
- 4. Check the html**
5. Swap Out wtheta.js for flare.json



1. Save Website
2. Check Dependencies
3. Check the Data Format
4. Check the html
- 5. Swap Out wtheta.js for flare.json**

