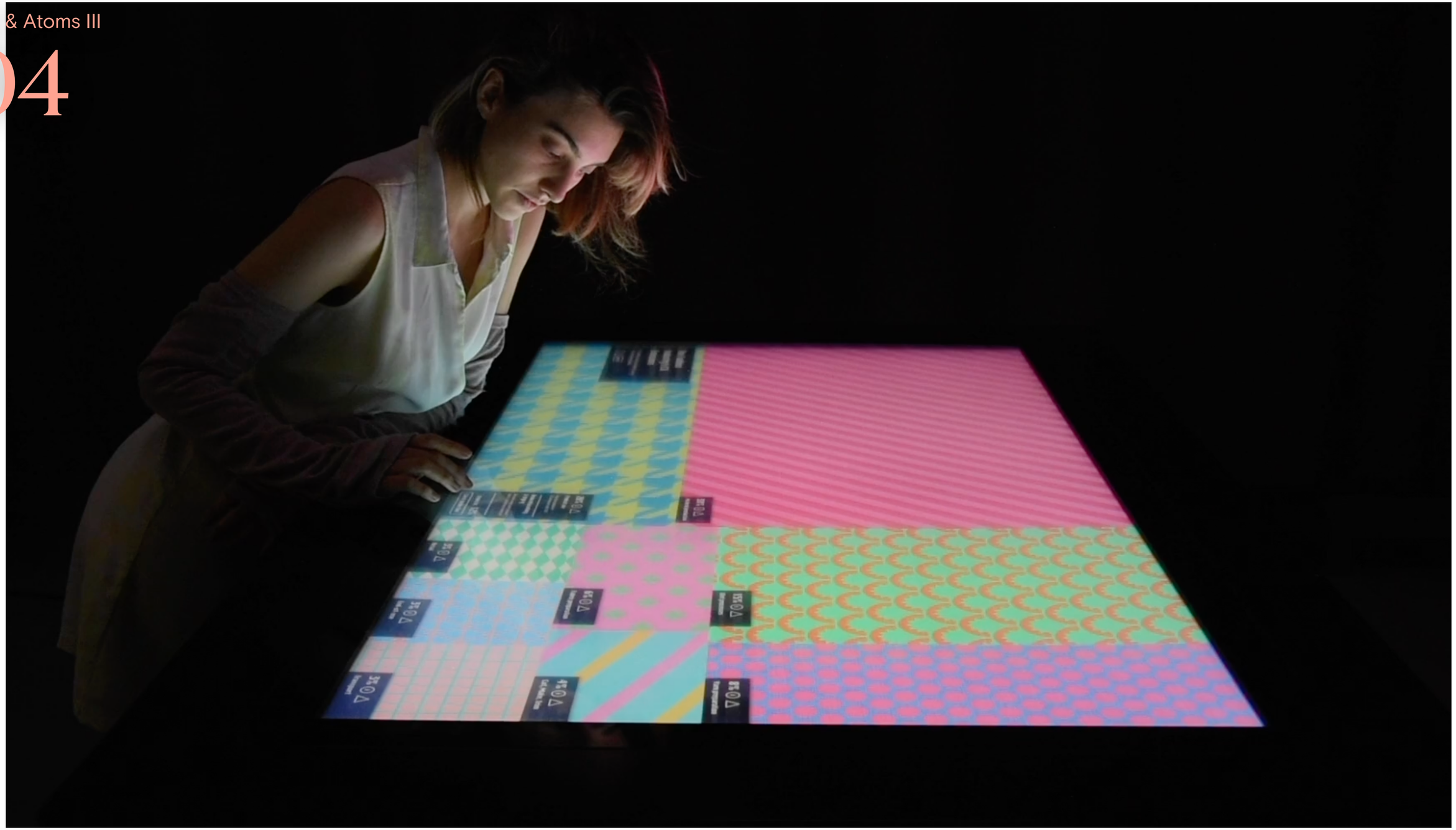
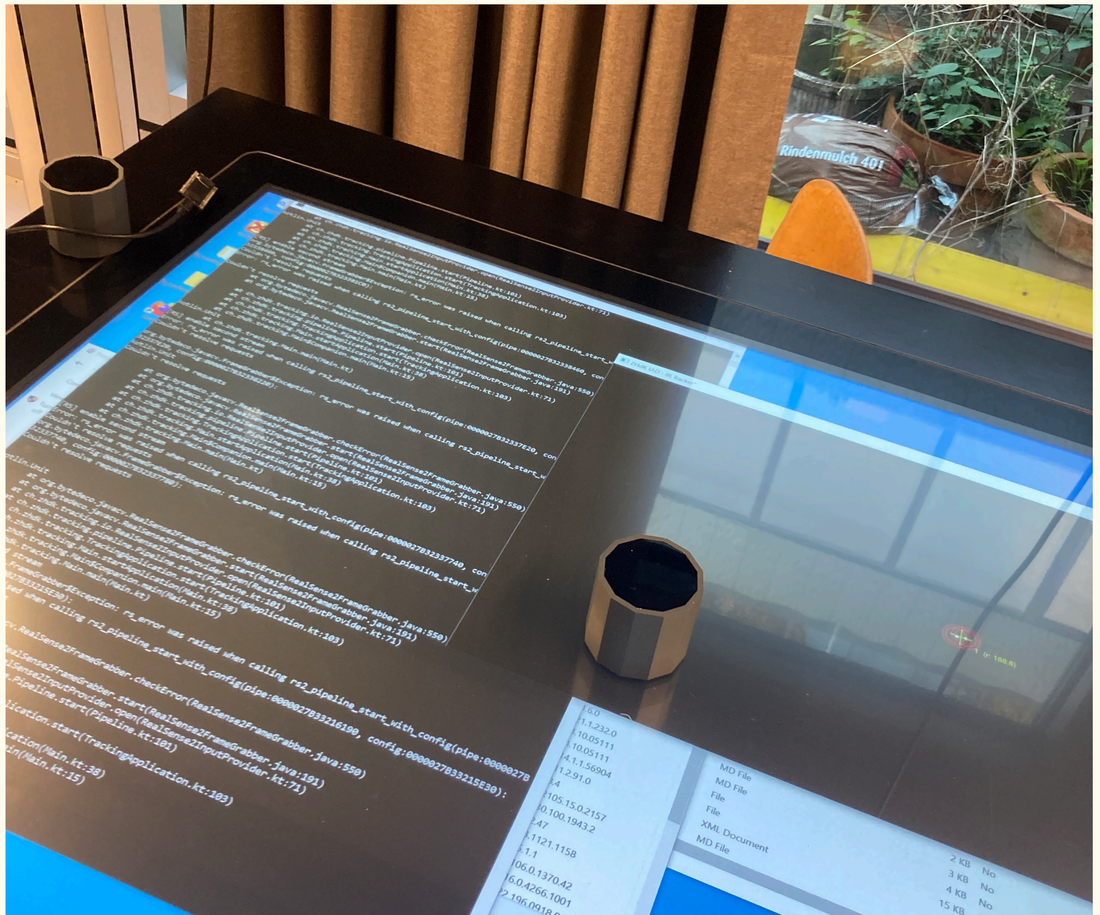
04



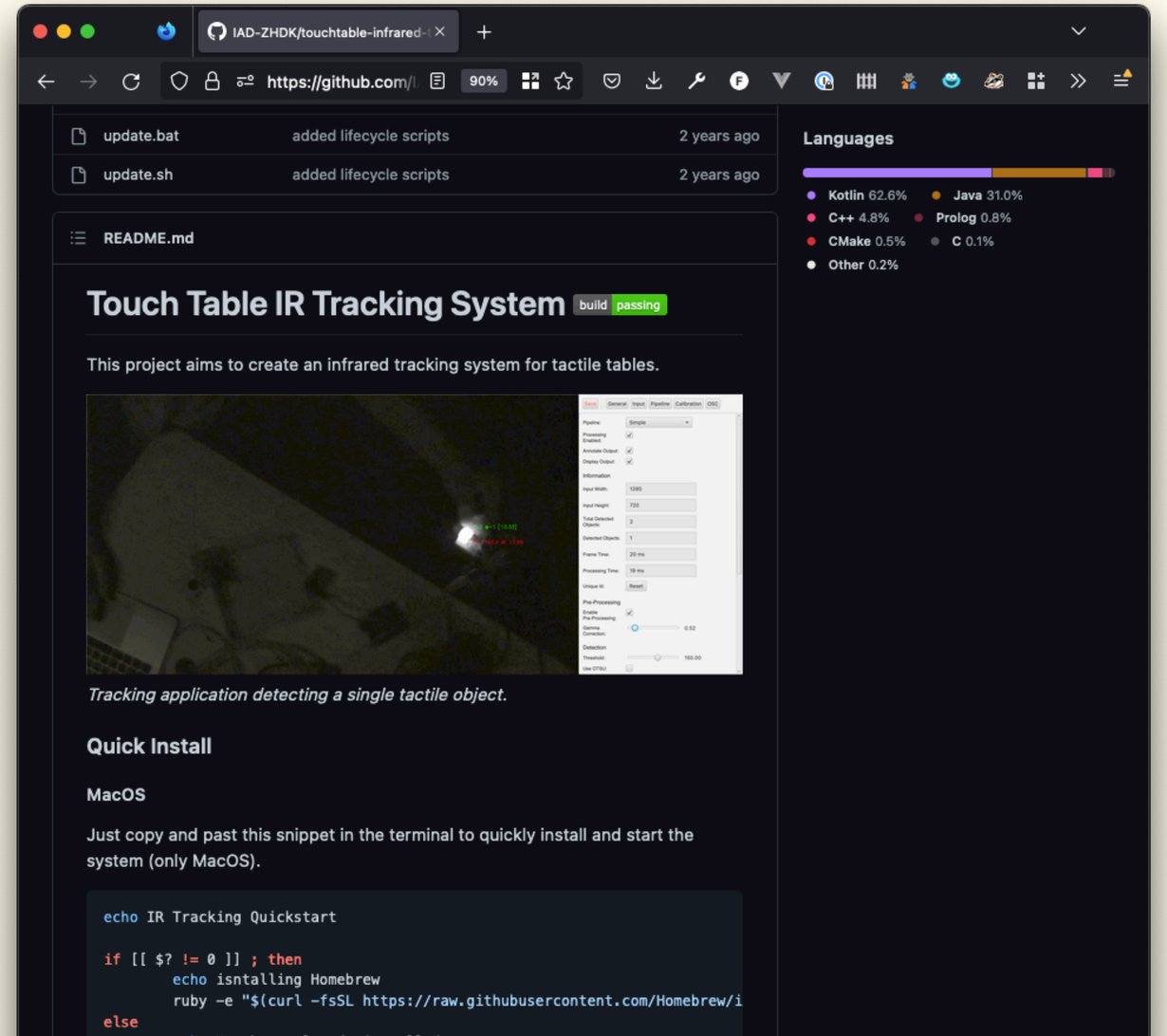
Source: https://vimeo.com/685524336?embedded=true&source=vimeo_logo&owner=9399454

Touch Table Emulator

- Touch table and infrared tracking devices
- Emulator on GitHub to work with 3D-tokens



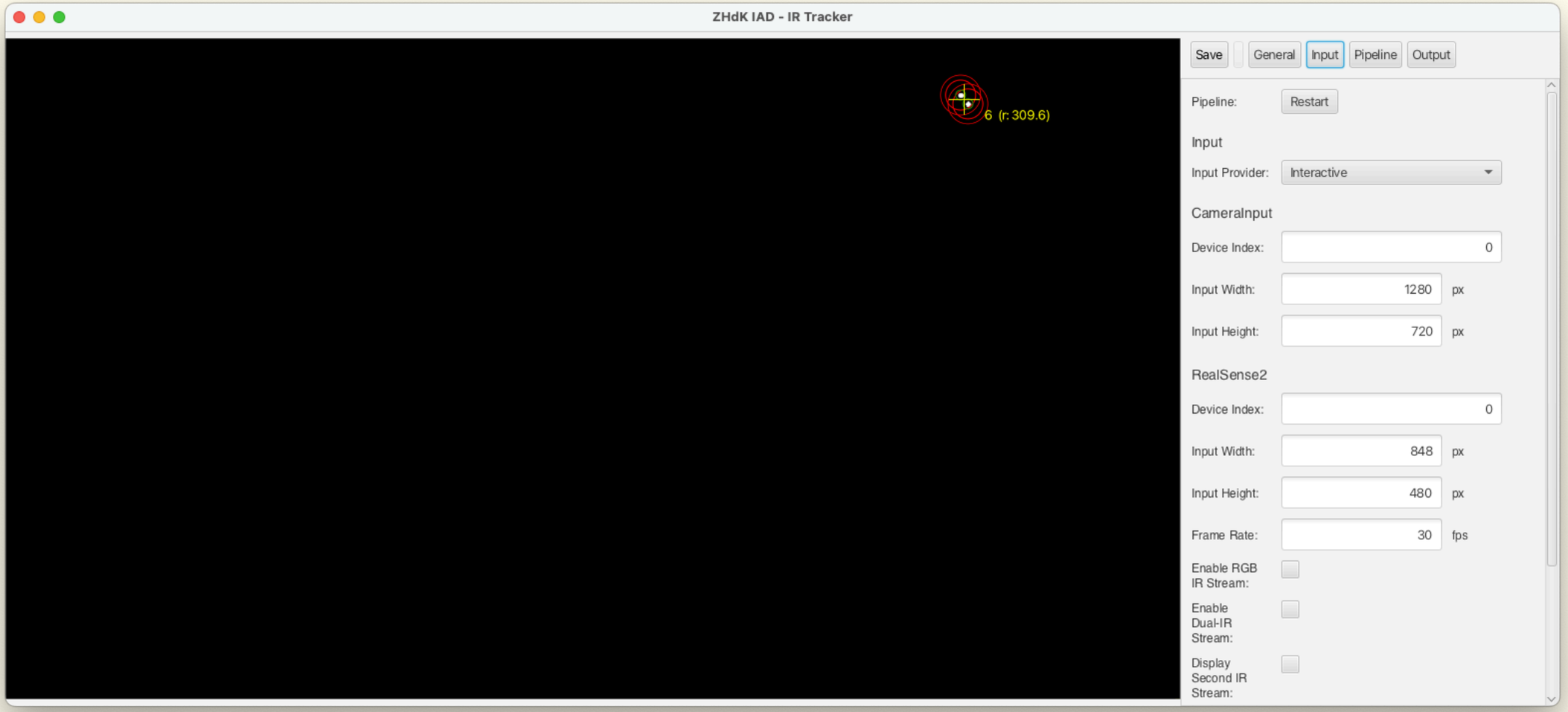
Touch Table Emulator



- https://github.com/IAD-ZHDK/touchtable-infrared-tracker

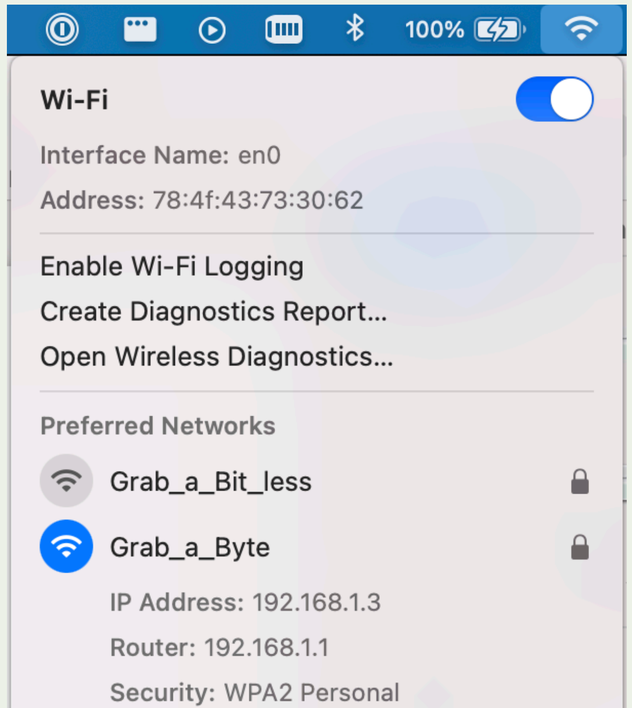
Bits & Atoms III

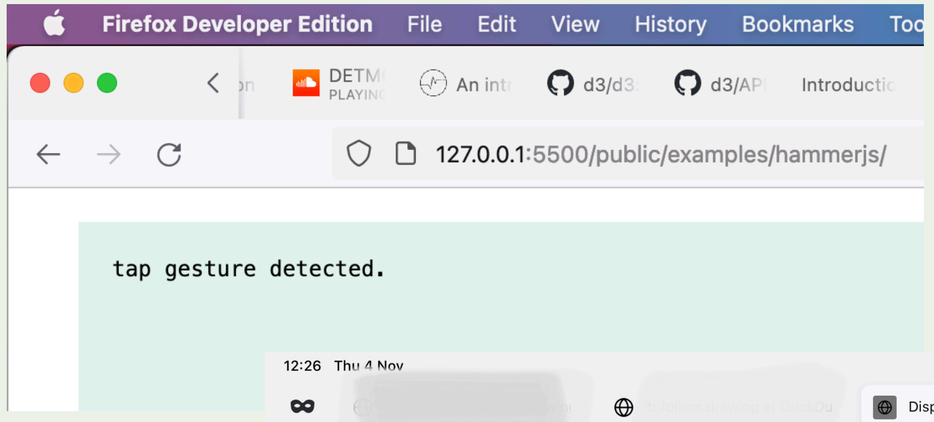
Touch Table Emulator



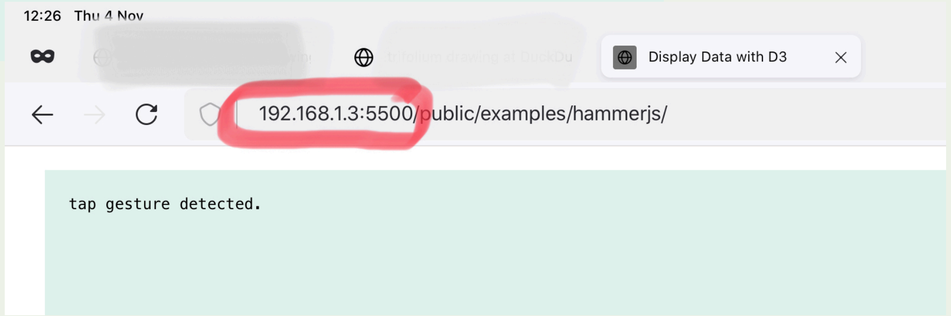
How to debug on touch devices?

- Online web page… but not easy to debug
- Local development (e.g. Live Server)
  - Make sure your touch device and your computer are using the same Wi-Fi
  - Copy computer's IP to your touch device
  - Make sure you set the port & path correctly





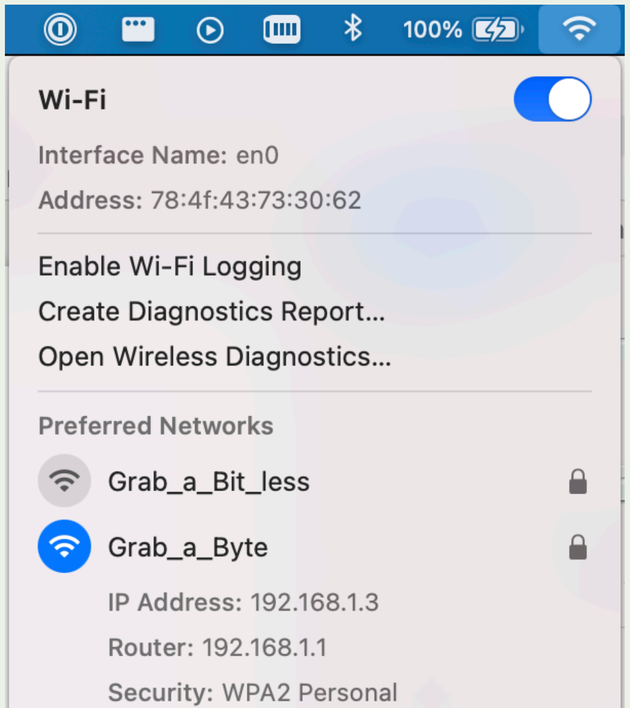
Computer

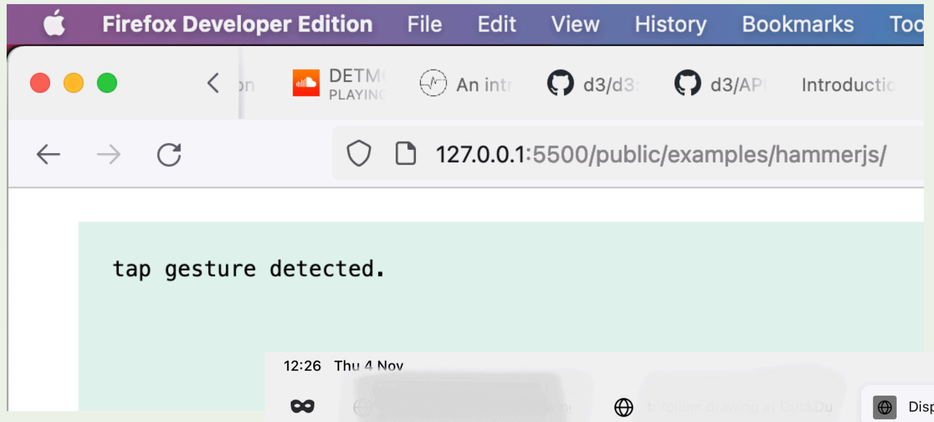


iPad

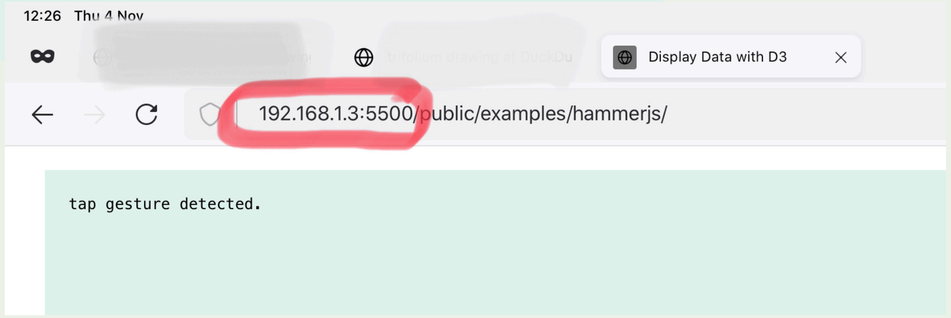
… and @ ZHdK?

- ngrok for the win
- Allows to set a “tunnel” to your localhost, serving it on publicly available url.
- requires shell skills (Terminal.app)





Computer

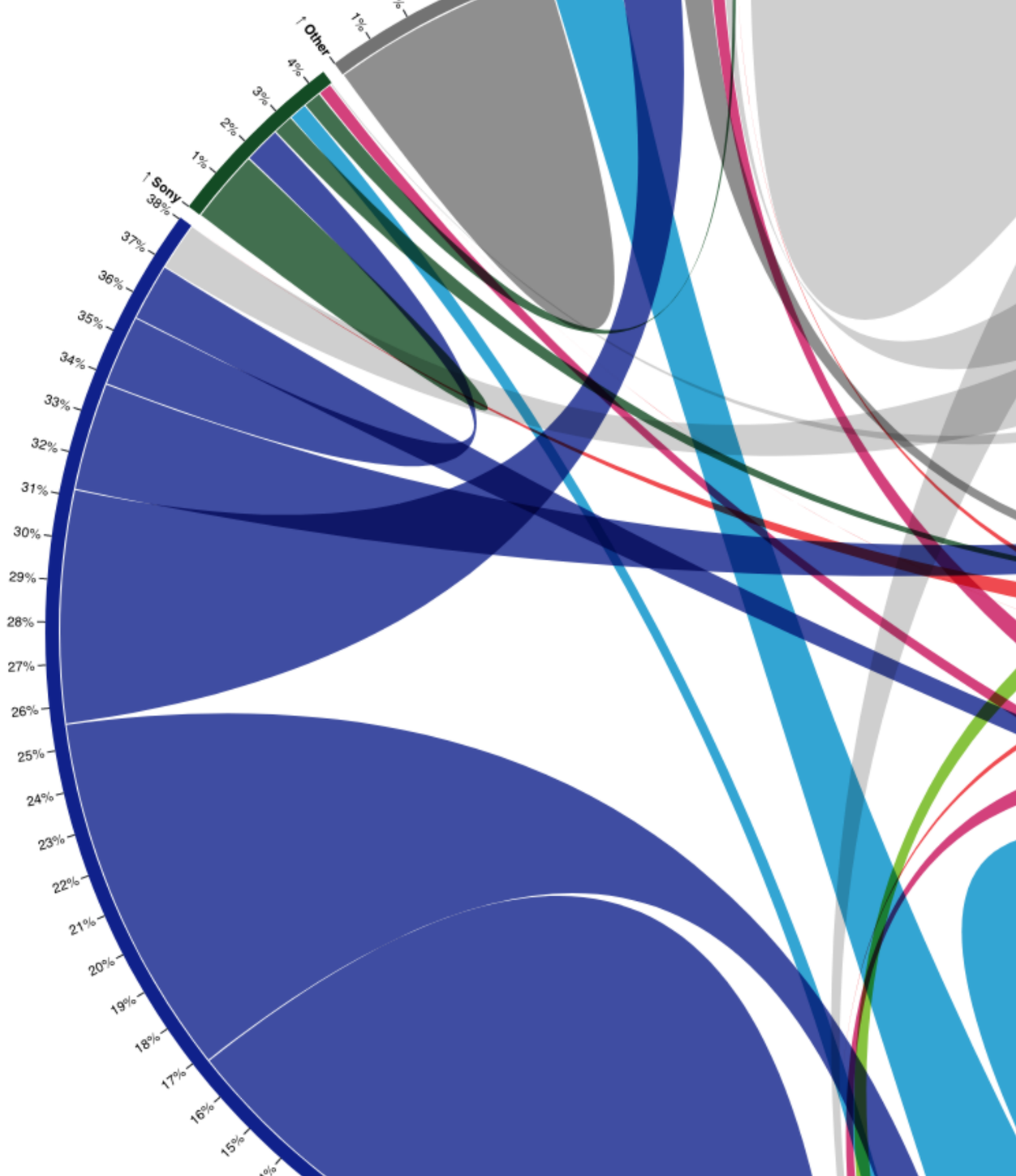


iPad

d3

- «**D3.js** is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to life using HTML, SVG, and CSS... »

- https://d3js.org/

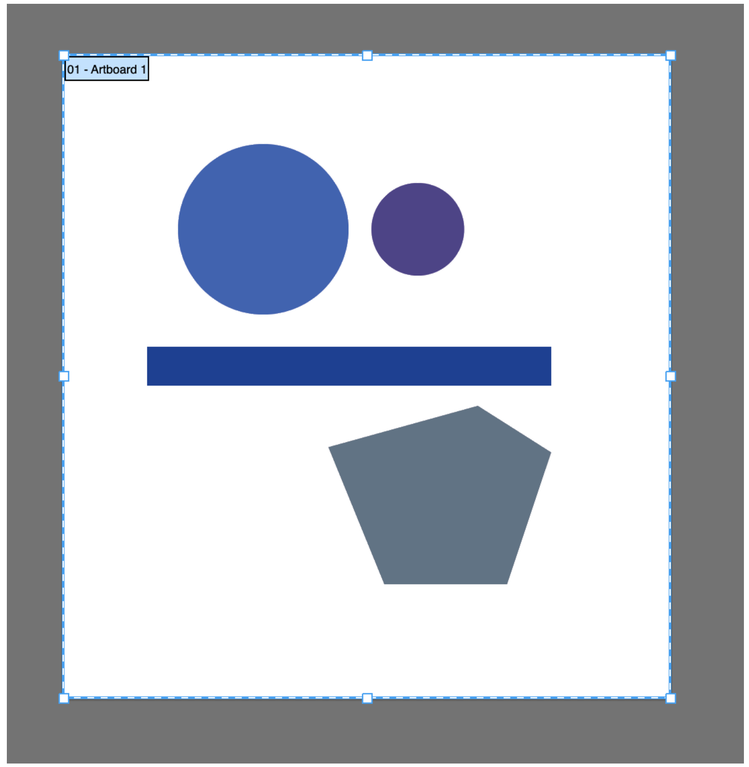


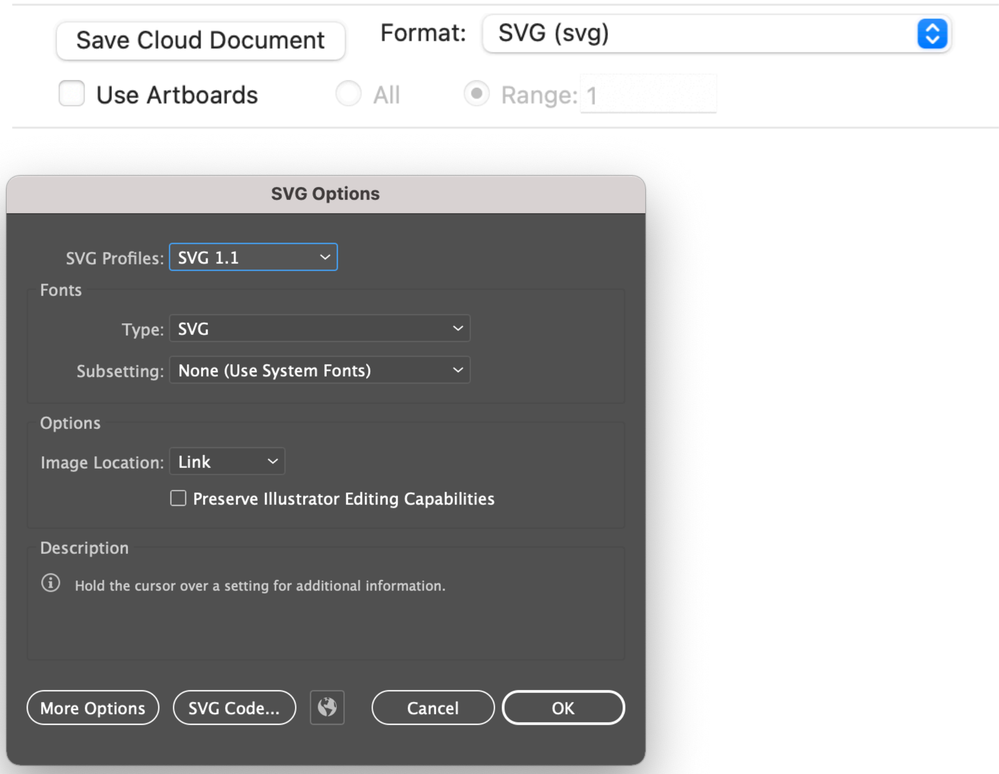
https://observablehq.com/@d3/chord-diagram

Bits & Atoms III

d3 → SVG

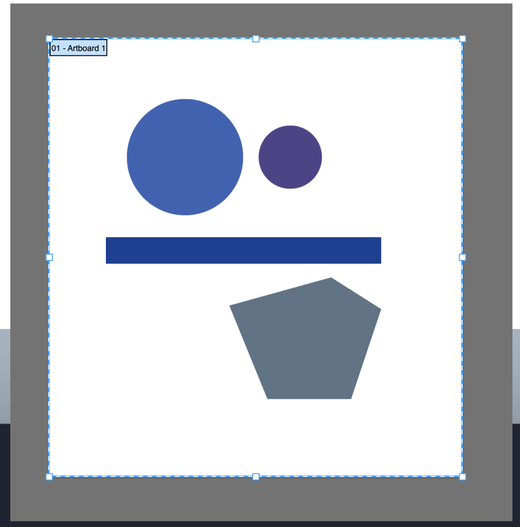
- SVG: Scalable Vector Graphics
- https://developer.mozilla.org/en-US/docs/Web/SVG





Bits & Atoms III

SVG

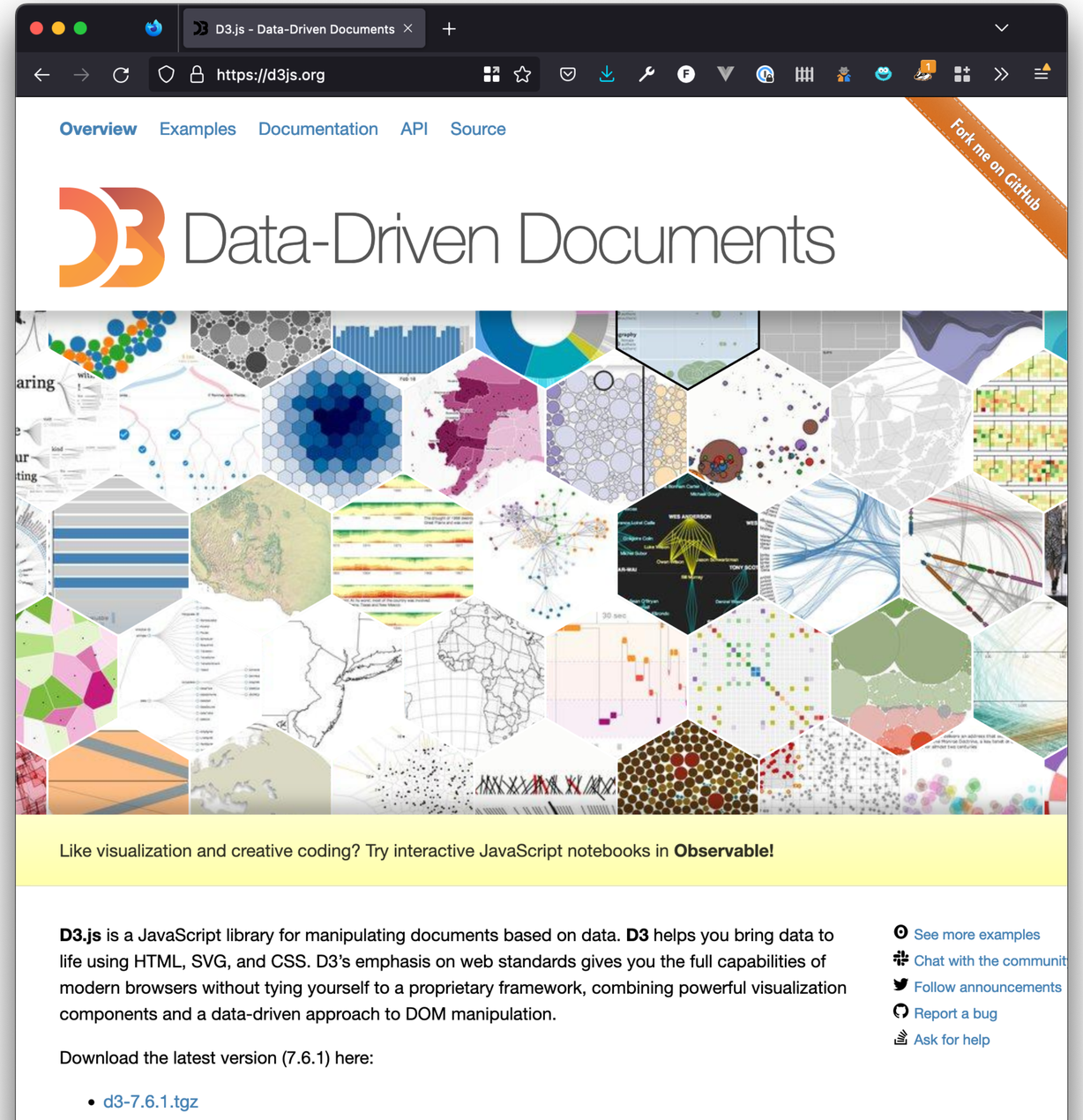



```svg
<?xml version="1.0" encoding="utf-8"?>
<!-- Generator: Adobe Illustrator 25.0.1, SVG Export Plug-In . SVG Version: 6.00 Build 0)  -->
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
   viewBox="0 0 753.3 798.3" style="enable-background:new 0 0 753.3 798.3;" xml:space="preserve">
<style type="text/css">
  .st0{fill:#4668B1;}
  .st1{fill:#524589;}
  .st2{fill:#214193;}
  .st3{fill:#677284;}
</style>
<circle class="st0" cx="248.3" cy="216.7" r="106.7"/>
<circle class="st1" cx="440.2" cy="216.7" r="58.3"/>
<rect x="103.3" y="361.7" class="st2" width="503.3" height="50"/>
<polygon class="st3" points="328.1,486.7 514.8,435 606.7,493.3 551.5,658.3 398.1,658.3 "/>
</svg>
```

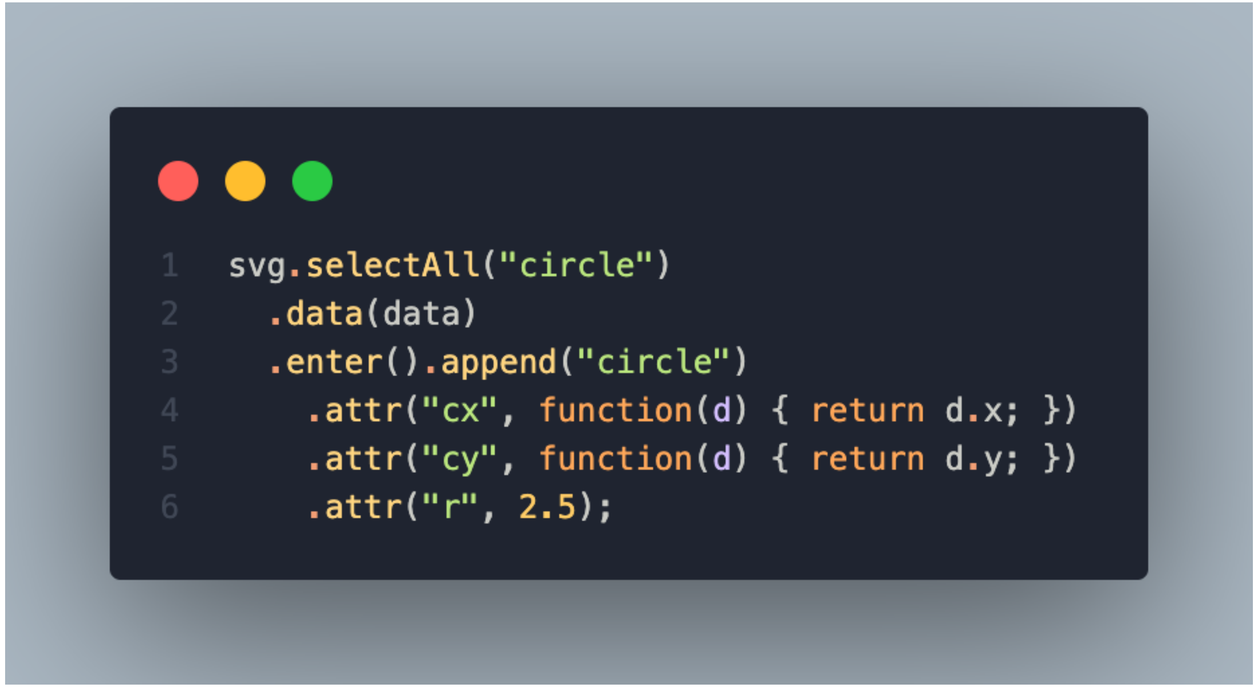
Bits & Atoms III

D3: Intro

- https://d3js.org/
- Read introduction (10–15min)



D3: Basic principles



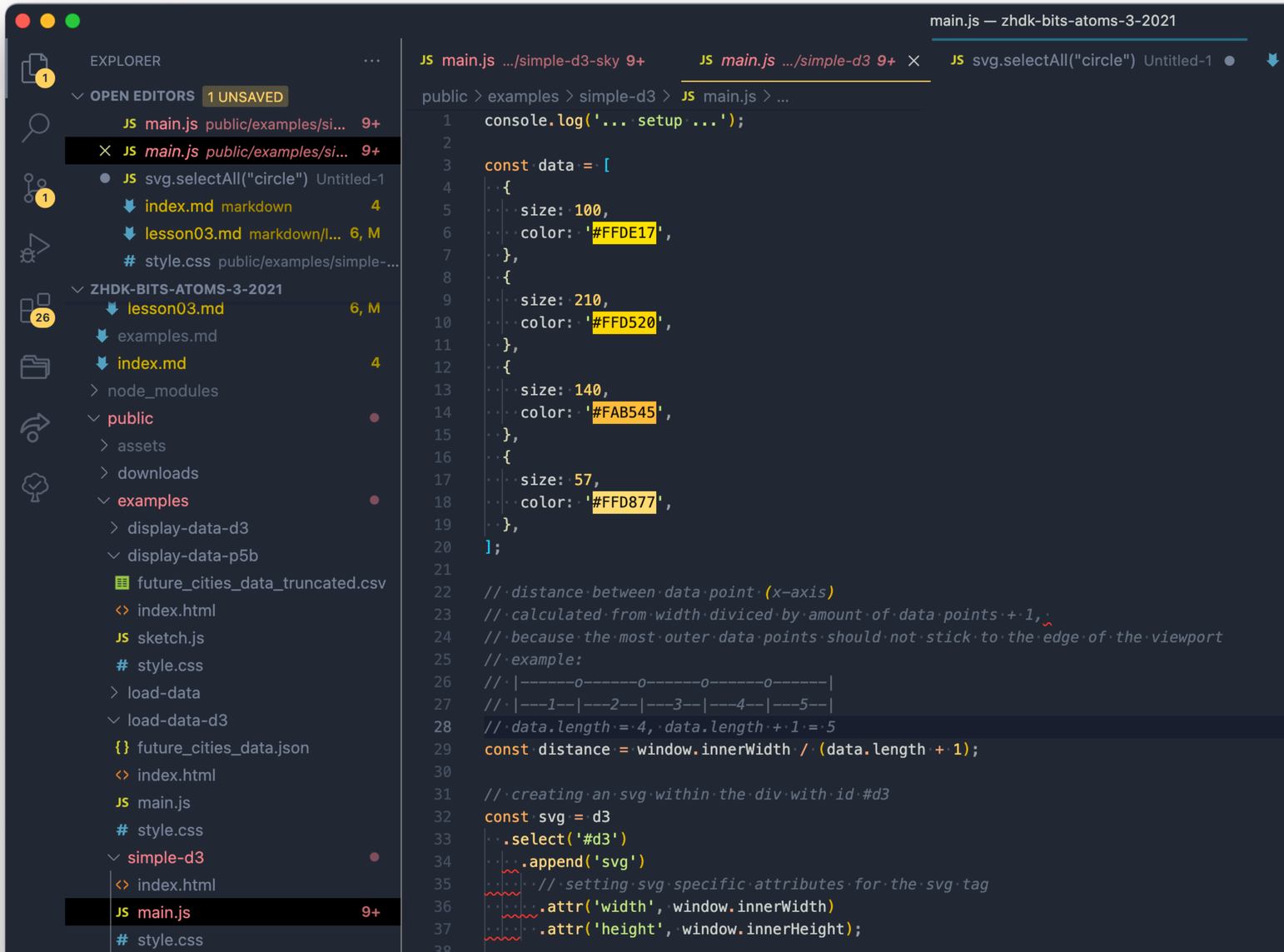
```
svg.selectAll("circle")
  .data(data)
  .enter().append("circle")
    .attr("cx", function(d) { return d.x; })
    .attr("cy", function(d) { return d.y; })
    .attr("r", 2.5);
```

https://www.d3-graph-gallery.com/intro_d3js.html#data
D3 Docs/API: https://github.com/d3/d3/blob/main/API.md

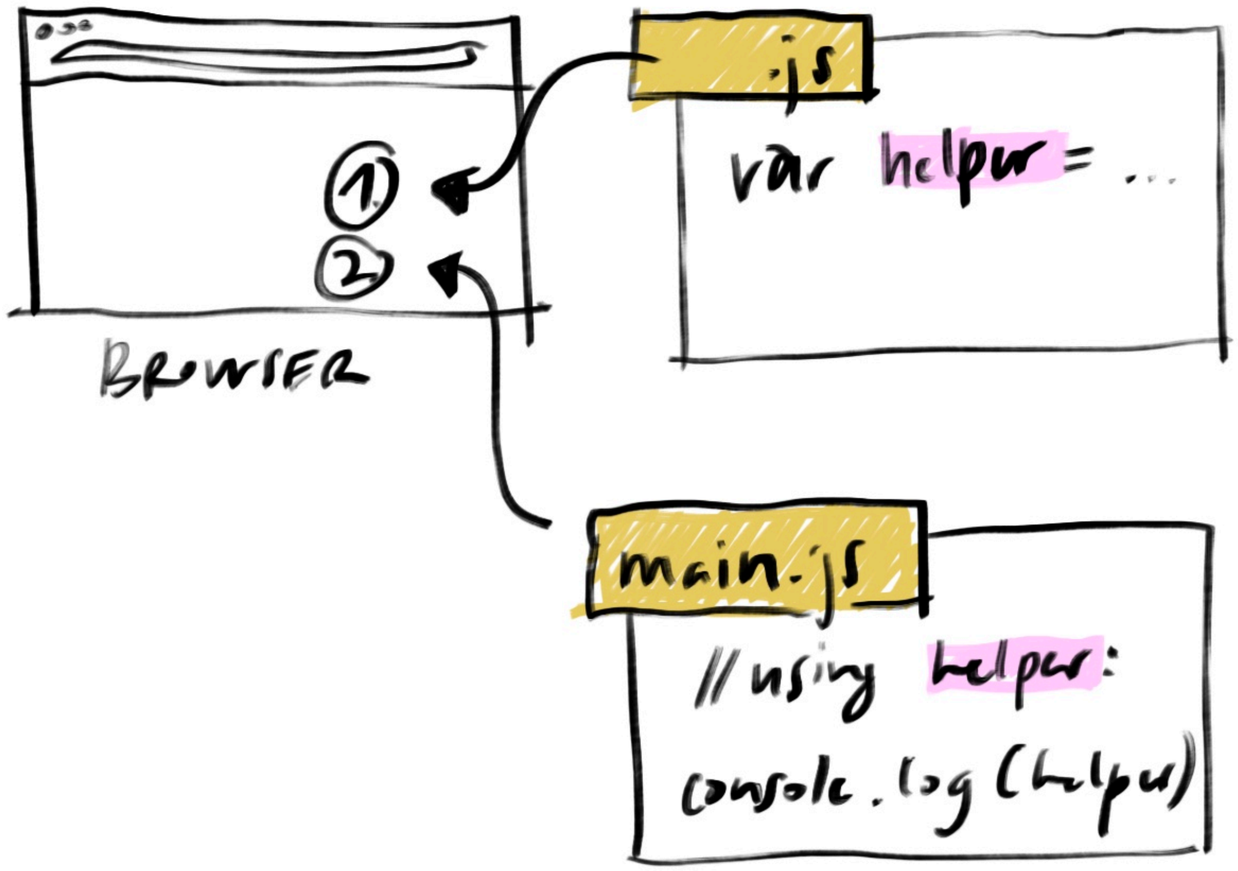
Examples
https://observablehq.com/ respectively: https://observablehq.com/@d3/
https://observablehq.com/@d3/streamgraph-transitions

D3: Example

- Examples from Email:
  → simple-d3
  → simple-d3-sky



d3: Under the hood



d3: Under the hood

```
var d3 = {
  // ...
};
```

```
// main.js
// has access to d3 object

var d3Method = d3.select;
console.log('d3Method: ', d3Method);
```

```
d3Method:  ▸ function Un(t)                    main.js:9:9
```

d3: Under the hood

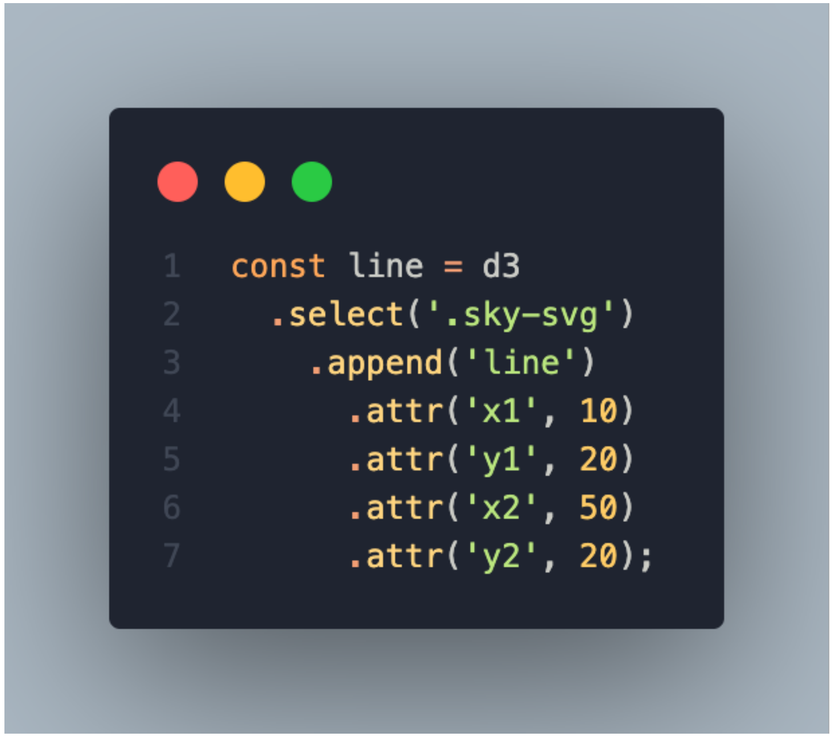
```
var d3 = {
  select: function(selector) {
    // do something (with selector)
    // return something
  },
  size: 100,
  author: 'mbostock'
}
```

```
// main.js
// can now make use of d3 method (function)

var mySelection = d3.select('#div');
```

d3: Under the hood

- But what happens here?... → Methods chaining/ function chaining



```
const line = d3
  .select('.sky-svg')
    .append('line')
      .attr('x1', 10)
      .attr('y1', 20)
      .attr('x2', 50)
      .attr('y2', 20);
```

https://www.tutorialspoint.com/method-chaining-in-javascript
https://medium.com/technofunnel/javascript-function-chaining-8b2fbef76f7f

d3: Under the hood

```jquery
// jQuery
var selectedDiv = $("#myDiv");

selectedDiv.css('color: red');
selectedDiv.height('100px');
selectedDiv.width('100px');
```

Is the same as:

```jquery
// jQuery
var selectedDiv = $("#myDiv");

selectedDiv.css('color: red').height('100px').width('100px');
```

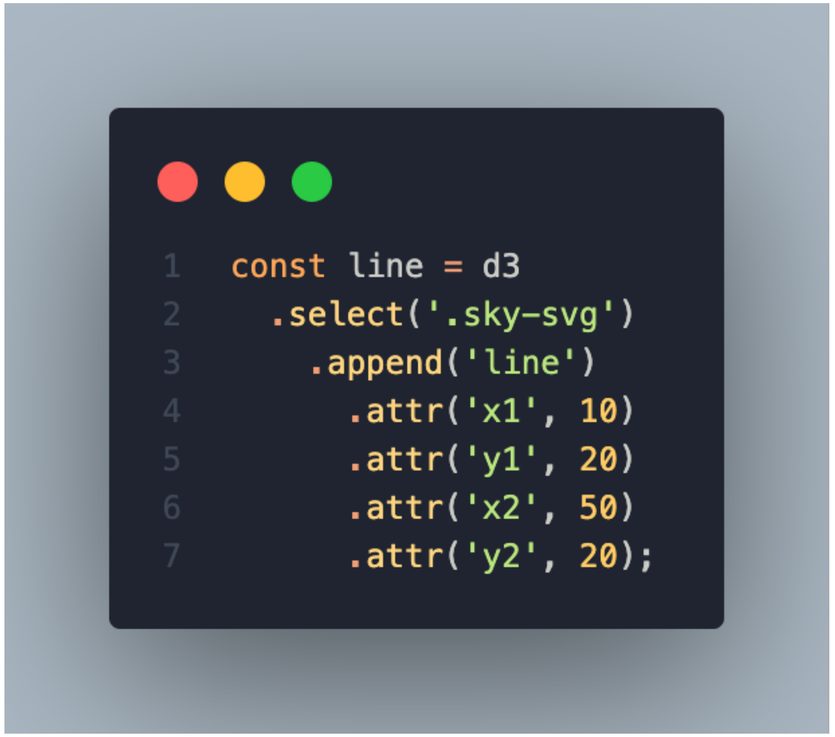
and also the same as:

```jquery
// jQuery
var selectedDiv = $("#myDiv");

selectedDiv.css('color: red')
  .height('100px')
  .width('100px');
```

d3: Under the hood

- ... we use a lot of chained d3 methods. Therefore we only use a semicolon (;) at the end.



```
const line = d3
  .select('.sky-svg')
    .append('line')
      .attr('x1', 10)
      .attr('y1', 20)
      .attr('x2', 50)
      .attr('y2', 20);
```

Exercise

- Work on your homework.

d3: Under the hood

- What about these functions in there?

```
const mySVG = d3.select('#d3-svg')
  .selectAll('whatever')
  .data(data)
  .enter()
  .append('circle')
    .attr('cx', function(d, currentIndex) { return currentIndex * distance + distance })
    .attr('cy', 300)
    .attr('r', function(d) { return d.size / 2 })
    .attr('fill', function(d) { return d.color });
```

d3: Under the hood

First we call the method attr() from d3

```
.attr('cx', function(d, currentIndex) { return currentIndex * distance + distance })
```

What does this function look like?

→ https://github.com/d3/d3-selection/blob/v3.0.0/README.md#selection_attr

→ https://github.com/d3/d3-selection/blob/main/src/selection/attr.js

Modifying Elements

- *selection*.attr - get or set an attribute.

selection.**attr**(*name*[, *value*]) · Source

If a *value* is specified, sets the attribute with the specified *name* to the specified value on the selected elements and returns this selection. If the *value* is a constant, all elements are given the same attribute value; otherwise, if the *value* is a function, it is evaluated for each selected element, in order, being passed the current datum (*d*), the current index (*i*), and the current group (*nodes*), with *this* as the current DOM element (*nodes*[*i*]). The function's return value is then used to set each element's attribute. A null value will remove the specified attribute.

If a *value* is not specified, returns the current value of the specified attribute for the first (non-null) element in the selection. This is generally useful only if you know that the selection contains exactly one element.

The specified *name* may have a namespace prefix, such as `xlink:href` to specify the `href` attribute in the XLink namespace. See namespaces for the map of supported namespaces; additional namespaces can be registered by adding to the map.

```
43  export default function(name, value) {
44    var fullname = namespace(name);
45
46    if (arguments.length < 2) {
47      var node = this.node();
48      return fullname.local
49          ? node.getAttributeNS(fullname.space, fullname.local)
50          : node.getAttribute(fullname);
51    }
52
53    return this.each((value == null
54        ? (fullname.local ? attrRemoveNS : attrRemove) : (typeof value === "func
55        ? (fullname.local ? attrFunctionNS : attrFunction)
56        : (fullname.local ? attrConstantNS : attrConstant)))(fullname, value));
57  }
```

???

???

???

d3: Under the hood

Functions as a parameter?

```
.attr('cx', function(d, currentIndex) { return currentIndex * distance + distance })
```

```
43  export default function(name, value) {
44    var fullname = namespace(name);
45
46    if (arguments.length < 2) {
47      var node = this.node();
48      return fullname.local
49          ? node.getAttributeNS(fullname.space, fullname.local)
50          : node.getAttribute(fullname);
51    }
52
53    return this.each((value == null
54        ? (fullname.local ? attrRemoveNS : attrRemove) : (typeof value === "function"
55        ? (fullname.local ? attrFunctionNS : attrFunction)
56        : (fullname.local ? attrConstantNS : attrConstant)))(fullname, value));
57  }
```

Very complicated and short code to determine, what is passed to the “each” function…
Don’t worry!

d3: Under the hood

```
27  function attrFunction(name, value) {
28    return function() {
29      var v = value.apply(this, arguments);
30      if (v == null) this.removeAttribute(name);
31      else this.setAttribute(name, v);
32    };
33  }
```

```
.attr('cy', 300)
```

```
.attr('cx', function(d, currentIndex) { return currentIndex * distance + distance })
```

.attrFunction(name, value)

.attrFunction(‘cy’, 300)

.attrFunction('cx', function(d, currentIndex) { return currentIndex * distance + distance })

d3: Under the hood

```
27  function attrFunction(name, value) {
28    return function() {
29      var v = value.apply(this, arguments);
30      if (v == null) this.removeAttribute(name);
31      else this.setAttribute(name, v);
32    };
33  }
```

With apply() the function, which is passed in as the variable 'value', is executed.

```
const array = ['a', 'b'];
const elements = [0, 1, 2];
array.push.apply(array, elements);
console.info(array); // ["a", "b", 0, 1, 2]
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/apply

var v = value.apply(this, `function(d, currentIndex) { return currentIndex * distance + distance })`)

This function is executed and we take the return value 'v' to set the attribute:

```
this.setAttribute(name, v);
```

https://developer.mozilla.org/en-US/docs/Web/API/Element/setAttribute

d3: Under the hood

Digging even deeper… → this.each

```
return this.each((value == null
    ? (fullname.local ? attrRemoveNS : attrRemove) : (typeof value === "function"
    ? (fullname.local ? attrFunctionNS : attrFunction)
    : (fullname.local ? attrConstantNS : attrConstant)))(fullname, value));
```

main **d3-selection** / src / selection / **each.js** / <> Jump to

mbostock Fix #57 - rename _nodes to _groups.

1 contributor

https://github.com/d3/d3-selection/blob/main/src/selection/each.js

10 lines (8 sloc) | 289 Bytes

```
1  export default function(callback) {
2
3    for (var groups = this._groups, j = 0, m = groups.length; j < m; ++j) {
4      for (var group = groups[j], i = 0, n = group.length, node; i < n; ++i) {
5        if (node = group[i]) callback.call(node, node.__data__, i, group);
6      }
7    }
8
9    return this;
10 }
```

Bits & Atoms III

d3: Under the hood

d3-selection / src / selection / each.js / <> Jump to

stock Fix #57 - rename _nodes to _groups.

ributor

(8 sloc) | 289 Bytes

```
port default function(callback) {

for (var groups = this._groups, j = 0, m = groups.length; j < m; ++j) {
  for (var group = groups[j], i = 0, n = group.length, node; i < n; ++i) {
    if (node = group[i]) callback.call(node, node.__data__, i, group);
  }
}

return this;
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/call

callback.call() executes our anonymous function we passed into attr()

```
function(d, currentIndex) { return currentIndex * distance + distance })
```

with

```
1  callback.call(node, node.__data__, i, group);
```

This means it executes our function and passes in:
1. node.__data__
2. i (index)
3. group

d3: Under the hood

```
callback.call(node, node.__data__, i, group);
```

This means it executes our function and passes in:
1. node.__data__
2. i (index)
3. group

```
.attr('cx', function(nodeData, index, group) {
  console.log('nodeData: ', nodeData);
  console.log('index: ', index);
  console.log('group: ', group);
  return index * distance + distance;
})
```

```
nodeData:  ▶ Object { size: 100, color: "#FFDE17" }
index:  0
group:
▶ Array(4) [ circle , circle , circle ,
circle ]
nodeData:  ▶ Object { size: 210, color: "#FFD520" }
index:  1
group:
▶ Array(4) [ circle , circle , circle ,
circle ]
nodeData:  ▶ Object { size: 140, color: "#FAB545" }
index:  2
```

d3: Under the hood

- So this is why we can use relatively plain javascript, calling d3 methods and passing in anonymous (=without a name) functions as parameters...

```
const mySVG = d3.select('#d3-svg')
  .selectAll('whatever')
  .data(data)
  .enter()
  .append('circle')
    .attr('cx', function(d, currentIndex) { return currentIndex * distance + distance })
    .attr('cy', 300)
    .attr('r', function(d) { return d.size / 2 })
    .attr('fill', function(d) { return d.color });
```