Week 10

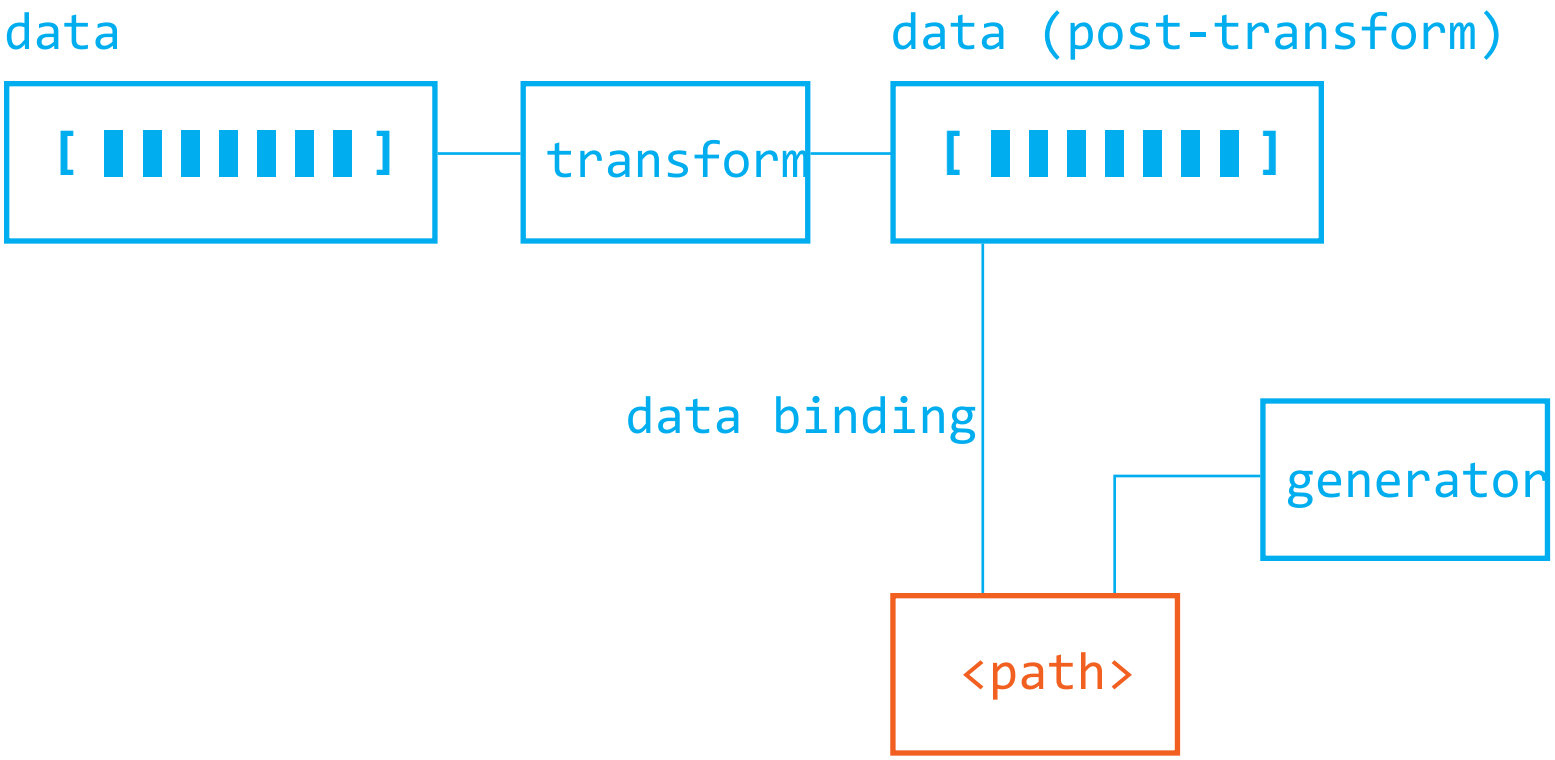
Geographic Representation

WHAT ARE WE TRYING TO DO?

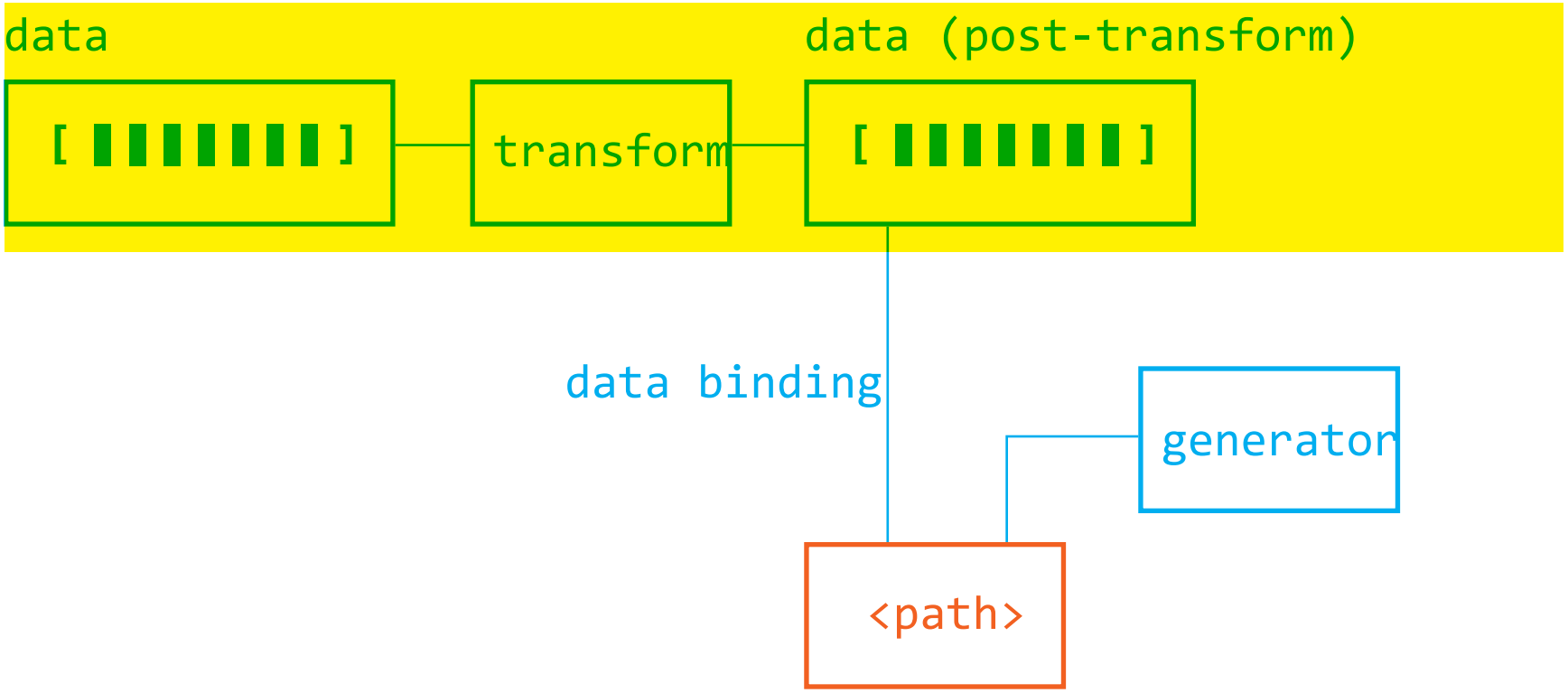
"Mapping" is a huge and vague topic. In this class, we'll focus on building a couple of key capabilities:

- Represent geographic features (points, lines, and polygon features) visually;
- Integrate thematic data into geographic representation i.e. **thematic mapping**;
- Alternative spatial representations, such as cartograms.

Review: Data Transformation and Path Generators



What Does Spatial Data Look Like?



Spatial Data

Spatial data comes in multiple (usually interchangable) formats:

.shp
KML
GeoJSON

The .json Format

You are actually already very familiar with .json data, which is an open-standard format that transmits data objects using **attribute-value pairs**.

```
{
  class: “ARTG5330”,
  graduateLevel: true,
  numStudents: 8,
  students: [
     {name: “Lia Petronio”, id:2334233},
     {name: “Ashley Treni”, id:3433322},
     ...
  ],
  instructor: {
     name: “Siqi Zhu”,
     id: 4333444,
     courses:[“ARTG5330”]
  }
}
```

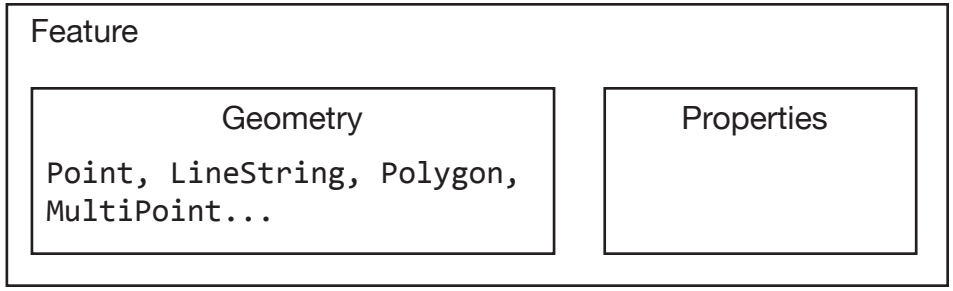
The .json Format

You are actually already very familiar with .json data, which is an open-standard format that transmits data objects using **attribute-value pairs**.

```
{ attribute   value
  class: “ARTG5330”,comma separation btw pairs
  graduateLevel: true,
  numStudents: 8,
  students: [
     {name: “Lia Petronio”, id:2334233},
     {name: “Ashley Treni”, id:3433322},
     ...
  ],
  instructor: {
     name: “Siqi Zhu”,
     id: 4333444,
     courses:[“ARTG5330”]
  }
}
```

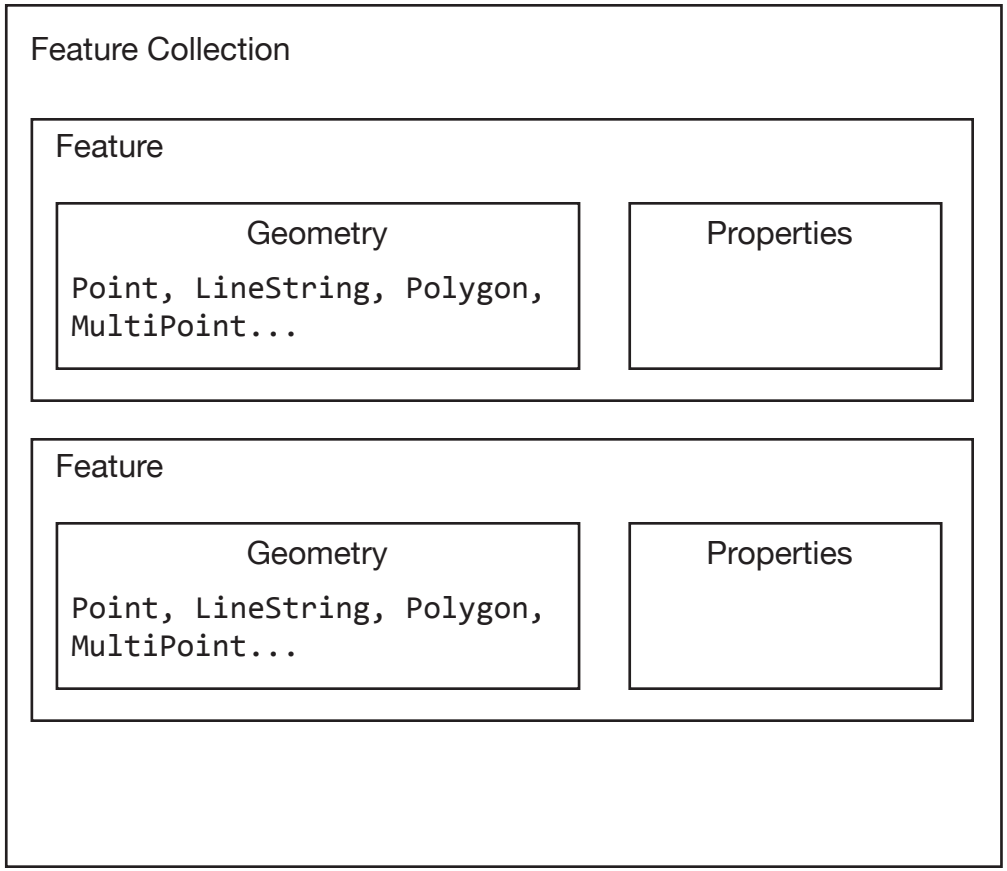
GeoJSON Is a Subset of .json

GeoJSON data is a subset of JSON, with attributes that specifically describe geometries and their properties.



GeoJSON Is a Subset of .json

GeoJSON data is a subset of JSON, with attributes that specifically describe geometries and their properties.



```
{ “type”: “FeatureCollection”,
    “features”: [
      { “type”: “Feature”,
        “geometry”: {“type”: “Point”, “coordinates”: [102.0, 0.5]},
        “properties”: {“prop0”: “value0”}
      },
      { “type”: “Feature”,
        “geometry”: {
          “type”: “LineString”,
          “coordinates”: [[102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]]
          },
        “properties”: {
          “prop1”: 0.0
          }
      },
      { “type”: “Feature”,
         “geometry”: {
           “type”: “Polygon”,
           “coordinates”: [
             [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
               [100.0, 1.0], [100.0, 0.0] ] ]
         },
         “properties”: {
           “prop1”: {“this”: “that”}
           }
         }
```

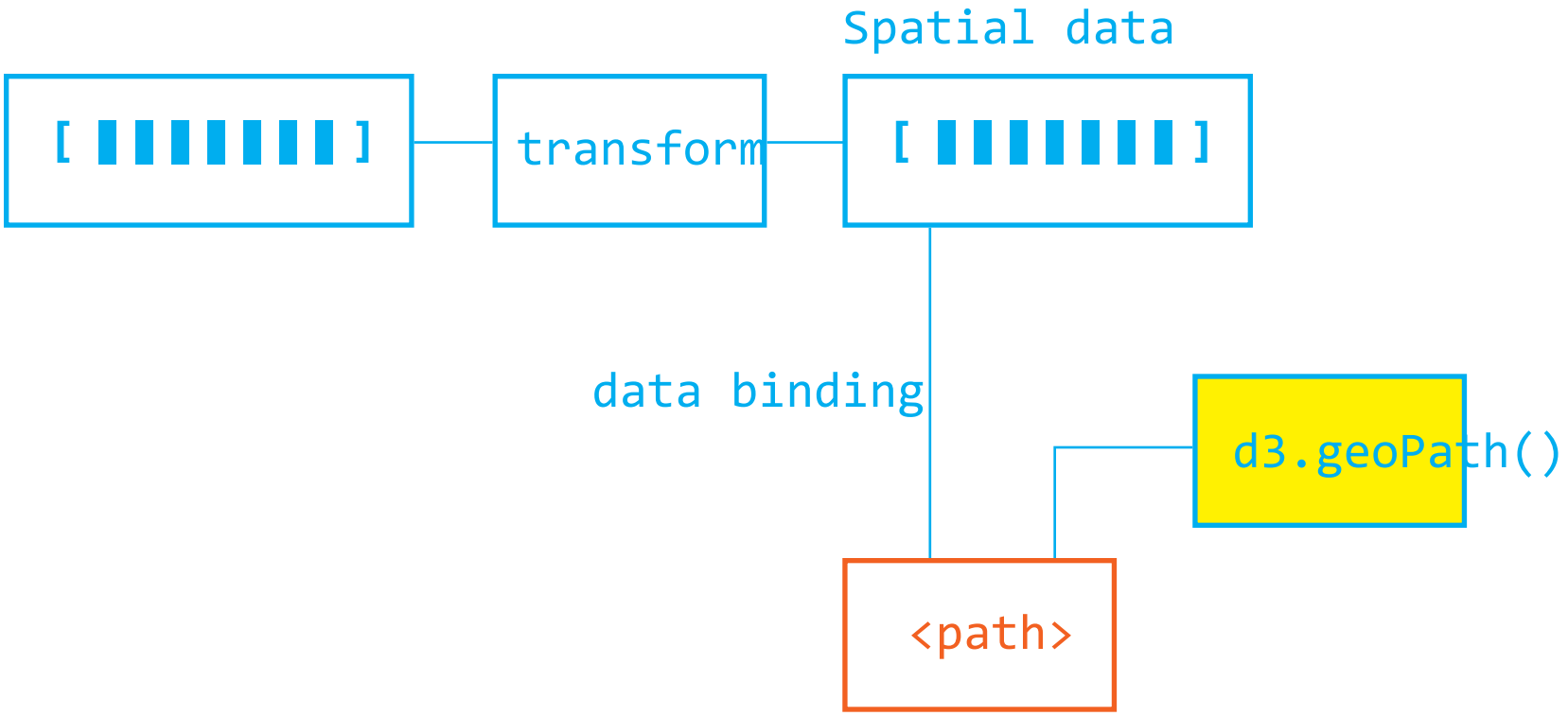
point

line

polygon

```
{ "type": "FeatureCollection",
    "features": [
      { "type": "Feature",
        "geometry": {"type": "Point", "coordinates": [102.0, 0.5]},
        "properties": {"prop0": "value0"}
      },
      { "type": "Feature",
        "geometry": {
          "type": "LineString",
          "coordinates": [[102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]]
          },
        "properties": {
          "prop1": 0.0
          }
      },
      { "type": "Feature",
       "geometry": {
         "type": "Polygon",
         "coordinates": [
           [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
             [100.0, 1.0], [100.0, 0.0] ] ]
       },
       "properties": {
         "prop1": {"this": "that"}
         }
       }
```

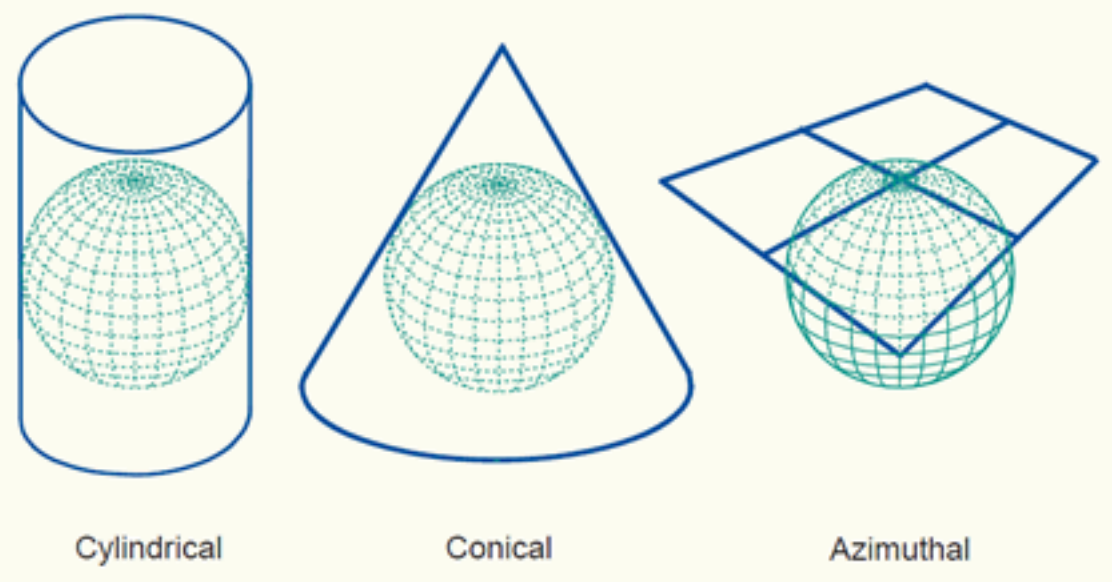
d3.geoPath( ) generator



d3.geoPath() generators will convert GeoJSON data to <path> geometry "d" attribute

`d3.geoPath( )` needs a projection

Not as simple as you think!



Map Projection

Map projection is the process whereby **longitude, latitude coordinates** on the surface of sphere are transformed into **cartesian coordinates** on a plane.

Conceptually, map projection should be a function, where

```
x-y coordinates = projectionFunction([longitude, latitude])
```

Map Projection

See a list of projections here

https://github.com/d3/d3-geo/blob/master/README.md#projections

Settings for projection

```
projection
    .center([lng,lat])
    .translate([x,y])
    .scale(...)
    .fitExtent(extent,obj)
```

Combining Projection with d3.geoPath()

To use a particular projection with a geo path generator

```
var path = d3.geoPath()
    .projection(newProjection);
```

Now the geo path generator is ready to use!

Exercise 1 and 2

Using what we know to:

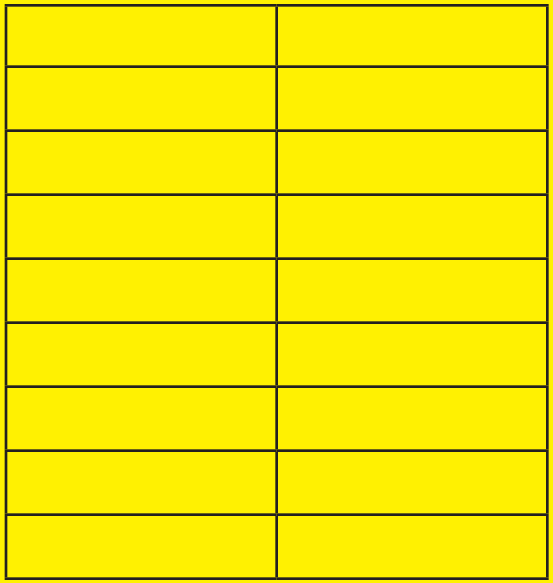
Draw a map
Change projection
Integrate the tooltip patter

Exercise 3: Choropleth

Exercise 3: using `d3.map()`

A “map” in programming speak is similar to a lookup table, which allows us to set (and look up) matching values.

```
//Setting up a map
var newMap = d3.map();
```



Exercise 3: using `d3.map()`

A “map” in programming speak is similar to a lookup table, which allows us to set (and look up) matching values.

```
//Setting up a map
var newMap = d3.map();

//Set
newMap.set(‘orange’, ‘fruit’);
```

orange	fruit

Exercise 3: using `d3.map()`

A “map” in programming speak is similar to a lookup table, which allows us to set (and look up) matching values.

```
//Setting up a map
var newMap = d3.map();

//Set
newMap.set(‘orange’, ‘fruit’);

//Get
newMap.get(‘orange’); // ‘fruit’
```

orange	fruit

Exercise 3: using `d3.map()`

A "map" in programming speak is similar to a lookup table, which allows us to set (and look up) matching values.

```
//Setting up a map
var newMap = d3.map();

//Set
newMap.set('orange', 'fruit');

//Get
newMap.get('orange'); // 'fruit'
```

county	rate
001	.04
002	.033
...	...

Exercise 3: using `d3.map()`

The full d3.map() API is here:
https://github.com/d3/d3-collection/blob/master/README.md#maps

Some other key methods

```
map.each()
map.keys()
map.values()
map.entries()
```

Review: Week 10

Representation

`d3.geoPath()`
`projection`

Data Manipulation

`d3.map()`

Interaction

Extras

.json data format
Basic concepts of a choropleth