

用前端框架简化 D3 编程

演讲人：汪志成（雪狼）

跨界互联
数聚未来

第四届中国数据分析师行业峰会
CHINA DATA ANALYST SUMMIT

北京 中国大饭店 2017.07

日程与相关知识

日程

D3 简介

这是什么？

有什么优缺点？

换个角度看 D3

架构

设计思想

换种用法

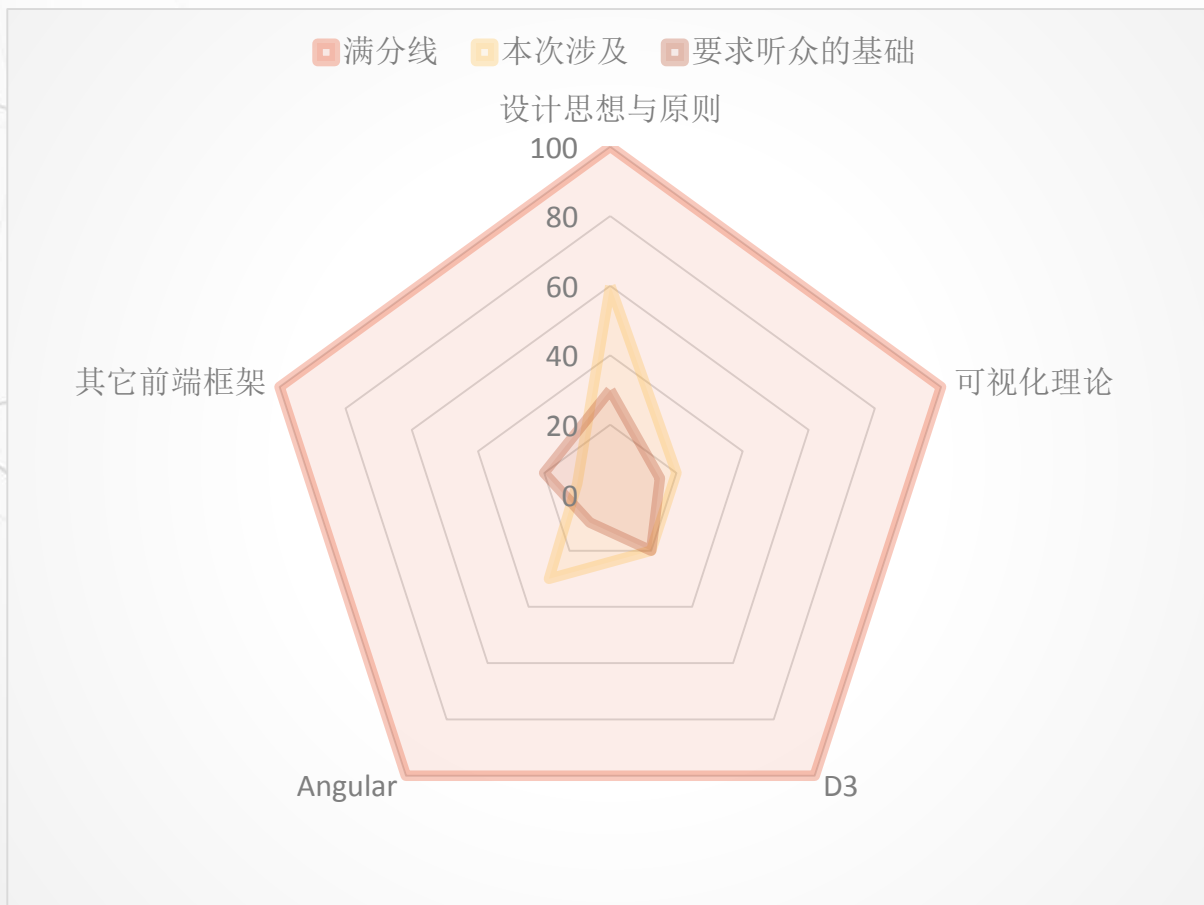
利用前端框架简化 D3 编程

集成方案

实例演示

背后的思想

相关知识



深度涉及设计思想与原则

中度涉及前端框架

轻度涉及 D3

轻度涉及可视化理论

关于我

ThoughtWorker

Google 开发技术专家 - GDE

Angular Contributor

Angular 官方文档中文版译者

《AngularJS 深度剖析与最佳实践》作者

《Angular 权威指南》（ng-book2）译者





Data-Driven

CDA 数据分析师
CERTIFIED DATA ANALYST

IT大咖说
知识分享平台

D3 简介



跨界互联
数聚未来

第四届中国数据分析师行业峰会
CHINA DATA ANALYST SUMMIT



CDA 数据分析师
www.cda.cn

什么是 D3?

常规认知

- 一个图表库，用于制作精美、复杂的图表
- Highcharts 等的竞品

本质：数据驱动文档

- 名字的由来：Data-Driven Documents
- 事实上，它自带了半个前端框架
- 听说曾有大牛基于它设计了全功能的前端框架

D3 的优点

良好的抽象

- 涉及到数据可视化的方方面面

自由的定制方式

- SoC 赋予的自由度（本课程的重点）

完善的生态系统

- 几乎无所不包
- 丰富的文档、书籍

D3 的缺点

入门难度较高

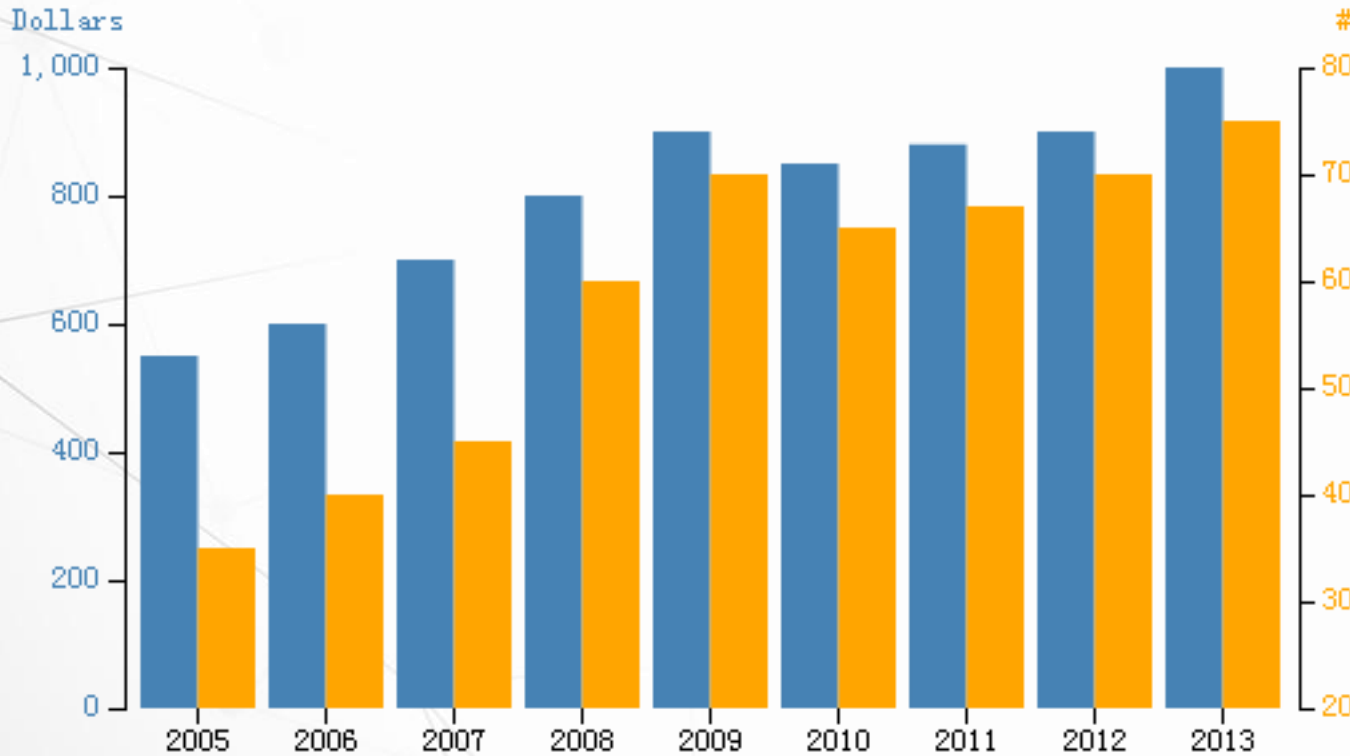
- 数据可视化
- SVG
- 数据驱动
- 相对其它技能来说还不够主流

生态系统良莠不齐

- 第三方库多而杂
- 筛选成本较高

换个角度看 D3

仔细看这张图，它包含哪几个方面的内容？



两个数据序列

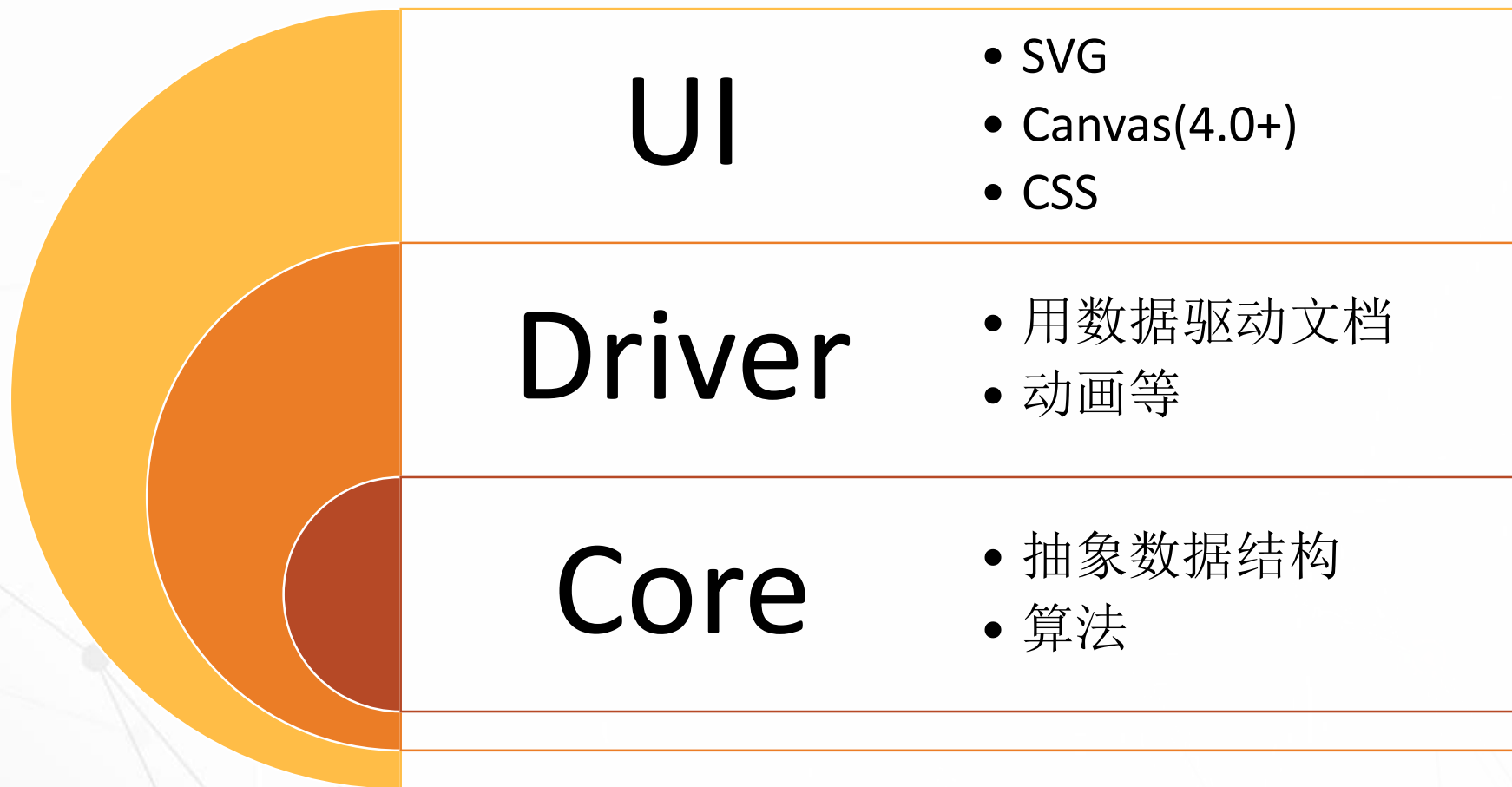
数据序列的对比

（可能存在）数据的更新

布局、样式、动画等

它可能有多少种变体？

架构



D3 的设计思想（个人理解）

数据为王

- 界面是表象，数据是核心

分层设计

- Core 层是灵魂
- Driver 层是骨架
- UI 层是血肉

SoC - 关注点分离

- Core 层关注可视化中涉及到的概念和算法
- Driver 层关注界面与数据的同步
- UI 层关注外观表现

可见，在 D3 中

最有价值的是

- 概念抽象
- 算法

Driver 层与主流前端框架的职责高度重叠

- 处理事件
- 在数据与（SVG）DOM 之间进行同步

UI 层易变

- D3 4.0 引入了 Canvas 渲染方式
- 实际业务中定制 UI 的需求很强劲

所以，我们可以把 D3 看成两个库！

那么，问题来了.....

我们能不能只用“半个”D3?

当然可以！

我们用 D3 Core 写算法与数据结构

- 与 SVG、CSS、Canvas 无关
- 极其纯粹
- 干干净净

我们用主流前端框架代替 Driver

- 人力资源丰富
- 技术成熟
- 有利于分工协作

这 will 把以前的分工方式改造成.....

建模工作

- 设计前端数据模型
- 写算法
- 可由数据工程师兼任

前端工作

- SVG（由 UX 给出基准设计稿）
- 样式
- 动画
- 由熟悉前端框架的工程师完成

怎么做？！

一、选一个前端框架

此处以 Angular 为例，是因为

- Angular 提供了非常全面、强大的基础设施
- Angular 的核心概念都来自后端，特别适合后端使用
- 我是 Angular 方面的专家

你也可以用其它框架实现

- 重要的是思想，不要受困于表象
- 框架只要支持模型与视图同步就可以
- 对于大中型组织来说，自己补齐基础设施也是值得的

二、学习 SVG 和 CSS

这是互联网的基础设施

- SVG 相当于图形领域的 HTML
- SVG 原来的样式（包括动画）已经移入了 CSS

这种投资不会浪费

- 这些知识在普通的 Web 开发中也很有用
- 这些都已经都是标准，不用担心突然消失

三、调整团队结构

根据康威定律和逆康威定律

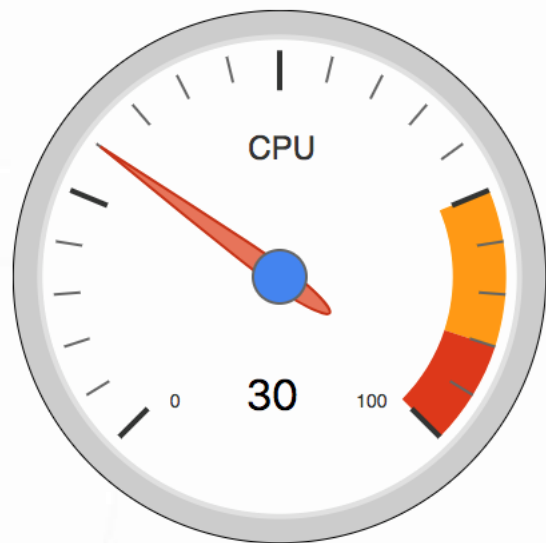
- 软件架构与组织结构会互相影响

所以，理想的分工方式是这样的

- UX/UI 用设计软件制作 SVG 格式的图表
- 数据建模人员设计 D3 Core 的数据模型
- 前端程序员把数据模型绑定到 SVG
- 前端程序员和 UX 合作，调整样式、设计动画

Talk is cheap, show me the code!

做这么个玩意儿要多少行代码？



Percent:

30

能显示进度文字

指针随着进度变化

左下角为 0%，右下角为 100%

指针值变化时有个缓动动画

不用除 D3 Core 之外的第三方库

估一下：500行？1000行？

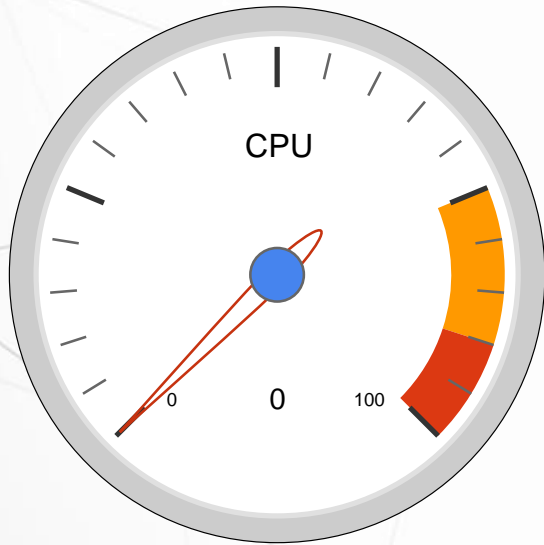
UX 制作图表

SVG Code

```
<svg width="202px" height="202px" viewBox="0 0 202 202" version="1.1"
xmlns="http://www.w3.org/2000/svg">
  <g stroke="none" stroke-width="1" fill="none" fill-rule="evenodd">
    <g>...</g>
    <path class="pointer"
          d="M100,17 C100,17 100,17 ..."
          stroke="#C63310" fill-opacity="0.7" fill="#DC3912" fill-rule="nonzero"
          transform="rotate(-135)"
    ></path>
    <circle stroke="#666666" ... cx="100" cy="100" r="10"></circle>
    <text fill="#000000">
      <tspan x="97.2053131" y="150" text-anchor="middle">0</tspan>
    </text>
  </g>
</svg>
```

pointer 类的 CSS 会把旋转的中心设置到全图中心

画出指针



数据建模程序员设计 D3 Core 模型

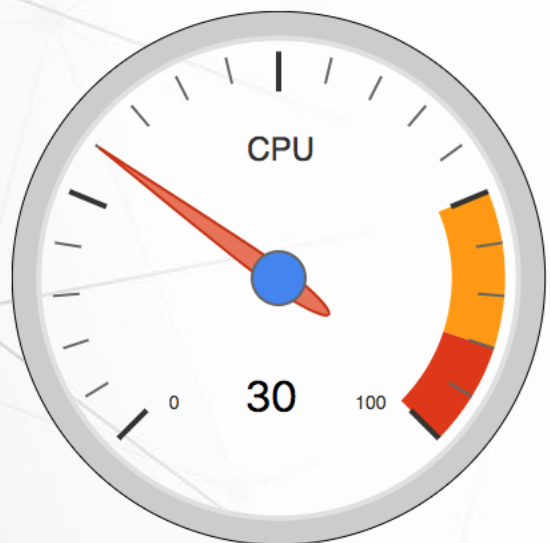
如果只需要显示 CPU 占用率

- 那就只有一项数据：百分比

我们还需要设计一个算法

- 把百分比折算成转动 (rotate) 的角度
- 已知：0%是-135度，100%是135度
- 求：x%对应的角度是多少？
- D3 的 scale 函数已经给出了答案：
`scaleLinear().domain([0, 100]).range([-135, 135])(x)`

前端程序员把模型应用到 SVG



Percent:

30

```
<svg width="202px" height="202px" viewBox="0 0 202 202" version="1.1"
xmlns="http://www.w3.org/2000/svg">
  <g stroke="none" stroke-width="1" fill="r" 把上页计算出的值放在这里
  <g>...</g>
  <path class="pointer"
    d="M100,17 C100,17 100,17 ..."
    stroke="#C63310" fill-opacity="0.7" fill="#DC3912" fill-rule="nonzero"
    attr.transform="rotate({{rotateAngle}})"
  ></path>
  <circle stroke="#666666" ... cx="100" cy="100" r="10"></circle>
  <text fill="#000000">
    <tspan x="97.2053131" y="150" text-anchor="middle">{{value}}</tspan>
  </text>
  </g>
</svg>
```

rotateAngle 是由 value 计算得来的

前端和 UX 合作调整样式、设计动画

样式咱就不调了吧，免得UX看了程序员的审美会发飙


随便加个动画吧！

```
.pointer {  
  transform-origin: 100px 100px;  
  transition: 2s;  
}
```

简单的过渡动画效果只要加这一句就行了

意犹未尽？来看更多的例子.....

折线图 - 模板



```
<svg viewBox="0 0 1000 1000" width="250" height="250">  
  <g fill="none" stroke="black">  
    <line stroke-width="5"  
      *ngFor="let point of points;let index=index"  
      [attr.x1]="point.x1" [attr.y1]="point.y1"  
      [attr.x2]="point.x2" [attr.y2]="point.y2"  
      [style.stroke]="index|d3Colors:points.length"  
    ></line>  
  </g>  
</svg>
```

折线图 - 模型代码

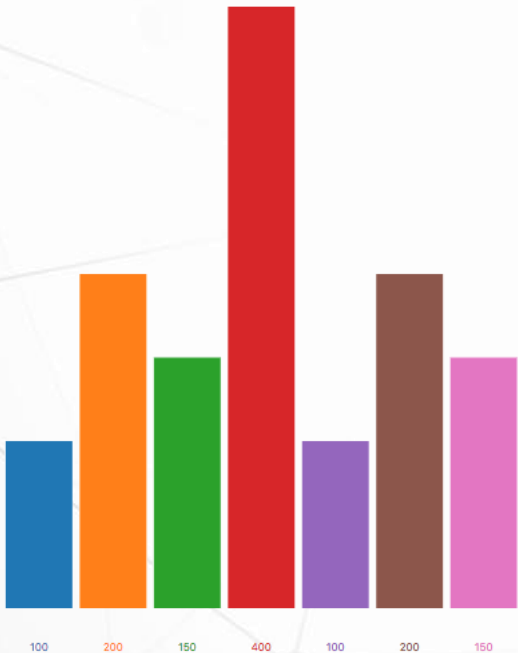
组件的实现代码

```
@Component({
  selector: 'app-d3-line-chart',
  templateUrl: './d3-line-chart.component.html',
  styleUrls: ['./d3-line-chart.component.scss'],
})
export class D3LineChartComponent {
  xScale = scaleLinear().domain([0, 6]).range([0, 1000]);
  yScale = scaleLinear().domain([0, 10]).range([0, 1000]);
  points = lines([1, 5, 3, 10, 1, 0, 3].map((value, index) => ({
    x: this.xScale(index),
    y: this.yScale(value),
  })));
}
```

lines 的实现代码

```
class Point { x: number; y: number; }
class Line { x1: number; y1: number; x2: number; y2: number; }
export function lines(data: Point[]): Line[] {
  return data.map((point, index, points) => {
    const nextPoint = points[index + 1];
    if (nextPoint) {
      return {
        x1: point.x, y1: point.y,
        x2: nextPoint.x, y2: nextPoint.y,
      };
    }
  }).filter(isDefined);
}
```


柱状图 - 模板



```
<svg viewBox="0 0 1000 1000" width="250" height="250">  
  <g *ngFor="let item of items; let index=index" [style.fill]="colorOf(index)"  
    attr.transform="translate({{offsetOf(index)}}, 900)">  
    <rect [style.height]="heightOf(item)" width="100"  
      x="-50" [style.y]="-heightOf(item)"></rect>  
    <text class="axis-label" dy="4em" text-anchor="middle">{{item}}</text>  
  </g>  
</svg>
```

柱状图 - 模型代码

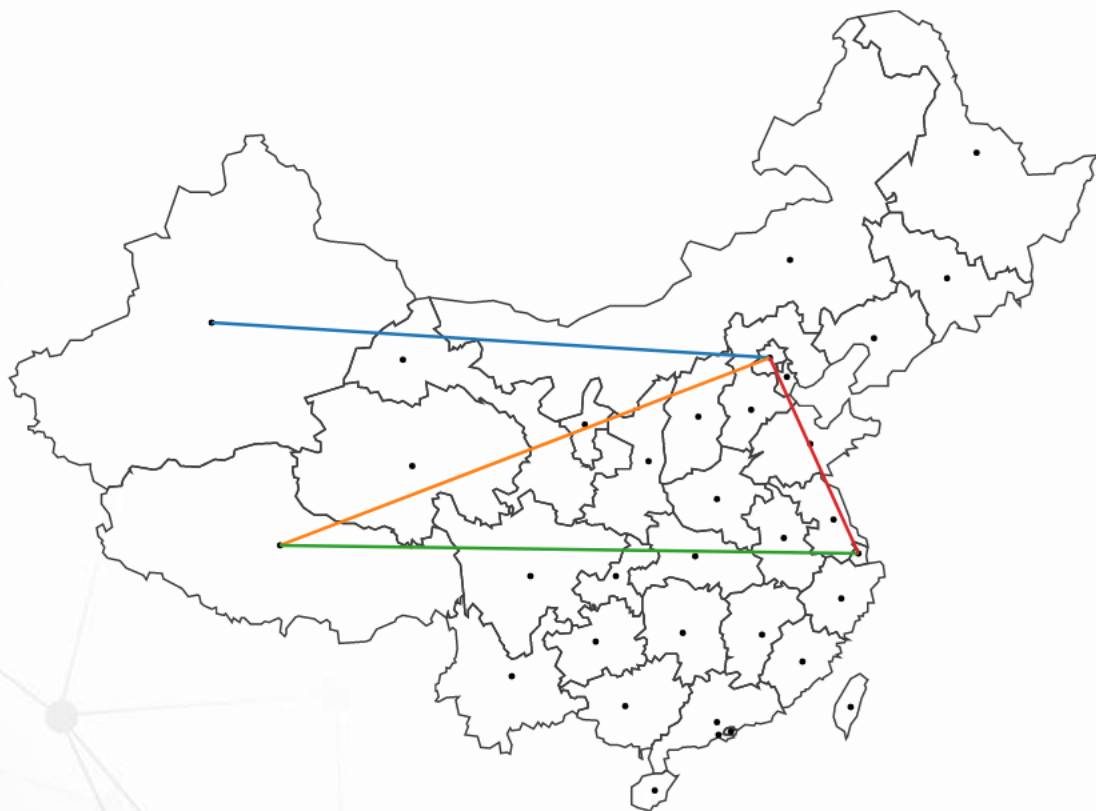
```
items = [100, 200, 150, 400, 100, 200, 150,];
```

```
colorOf(index: number): string {  
  return schemeCategory10[index];  
}
```

```
heightOf(item: number): number {  
  const scale = scaleLinear().domain([0, max(this.items)]).range([0, 1000]);  
  return scale(item);  
}
```

```
offsetOf(index: number): number {  
  const scale = scaleBand().domain(this.items.map((v, i) => i.toString())).paddingOuter(10).range([0, 1000]);  
  return scale(index.toString());  
}
```

迁徙图 - 效果图



迁徙图 - 模板

```
<svg [uiGeoBox]="map">  
  <g>  
    <g *ngFor="let feature of map.features">  
      <path stroke="rgba(61, 61, 61, 1)" stroke-width="0.3" fill="none" [uiGeoPath]="feature"></path>  
      <circle [uiGeoGps]="feature.properties.cp" nameX="cx" nameY="cy" r="0.5" fill="#000"></circle>  
    </g>  
  </g>  
  <g class="overlay">  
    <g *ngFor="let route of routes;let index = index">  
      <line [style.stroke]="index | d3Colors" [attr.stroke-width]="0.5" [uiGeoLine]="route.vector"></line>  
    </g>  
  </g>  
</svg>
```

迁徙图 - 模型代码

provinces和routes的常量定义：略.....

```
map: FeatureCollection<GeometryObject> = provinces;  
routes = routes.map(({from, to, weight}) => {  
  const fromLoc = _.find(this.map.features, {id: from}).properties['cp'];  
  const toLoc = _.find(this.map.features, {id: to}).properties['cp'];  
  return {  
    weight: weight,  
    vector: [fromLoc, toLoc],  
  };  
});
```

迁徙图 - 可复用的基础设施

uiGeoBox

- 计算一个地图中所有Feature的总边界
- 将地图的总边界设置为svg的viewBox

uiGeoPath

- 生成特定Feature的path字符串

uiGeoLine

- 设置line元素的起终点svg坐标

uiGeoGps

- 把一个gps坐标投影为平面坐标

动心了？然而.....

还有一些待解决问题

D3 的部分特性（比如坐标轴的绘制）依赖 UI 层

- 需要抽象出一个独立的坐标轴模型

现有资源可能需要迁移才能用

- 但未必能产生相应的业务价值

超大数据量下会形成性能瓶颈

- 一万个以上 SVG 节点可能会形成性能瓶颈
- 最好能通过模型层处理隐藏掉当前不可见的部分
- 也可以考虑用 D3 原生API

你未必真的需要这种方式

定制化的需求高吗？

- 定制化的业务价值在哪里？
- 它能节省成本或降低风险吗？

现有资源如何？

- 已经有了 D3 高手和大量库？那就没必要迁移
- 组织架构无法/不会支持？还是维持现状吧。

Q & A



CDA 数据分析师
www.cda.cn

THANKS

跨界互联 数聚未来

第四届中国数据分析师行业峰会
CHINA DATA ANALYST SUMMIT