

# D3.js – Interaction

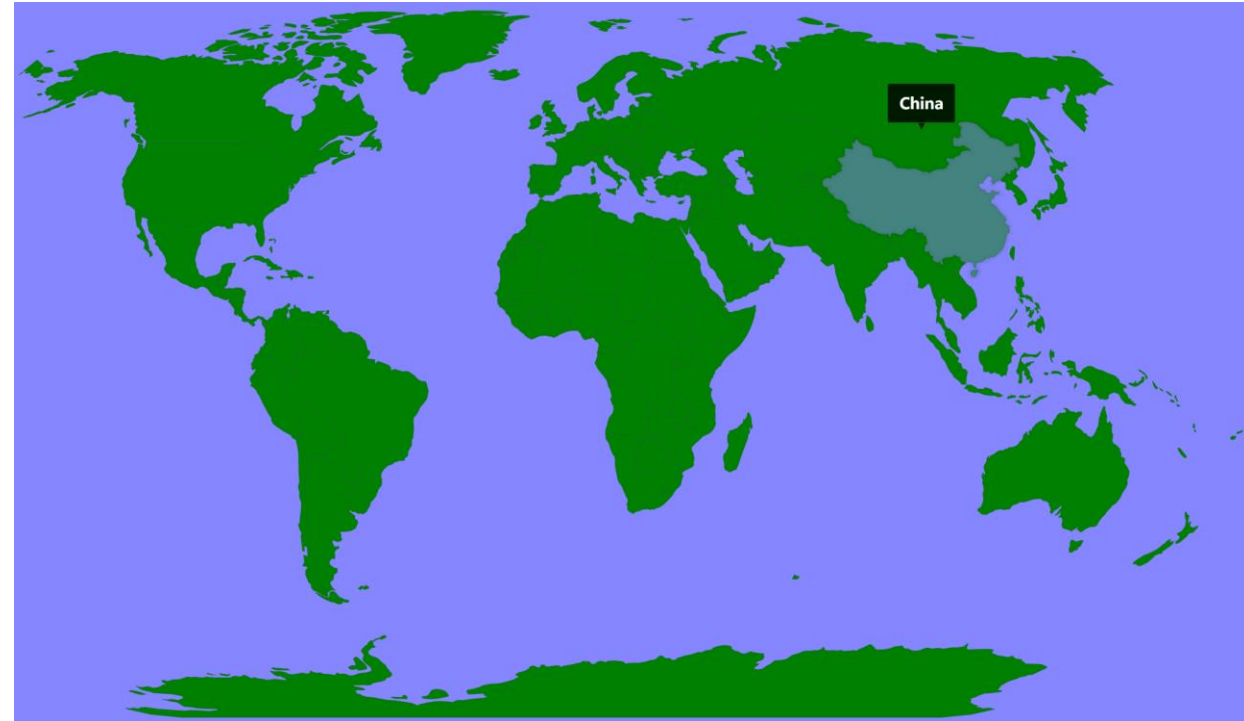
张松海、张少魁、周文洋、蔡韵

数据可视化 – D3.js

清华大学 可视媒体研究中心

# Interaction & Map

- 地图数据的可视化与D3.js的交互
- 如何使用D3绘制一张地图
  - Json数据
  - TopoJson & GeoJson
- D3中如何实现交互
  - 什么是事件
  - 事件的监听与处理
- D3-Tip
- 层叠式样式表 - CSS
- Code: <https://github.com/Shao-Kui/D3.js-Demos/blob/master/static/renderearth.html>
  - URL: <http://127.0.0.1:11666/renderearth>



# Json

- JavaScript Object Notation (JSON)
- 从数据格式上（本质上）是JavaScript的对象
- 保存为文件后是文本
- 文本与JavaScript的对象可以‘对等转换’

```
▼ {type: "Topology", objects: {...}, arcs: Array(595), bbox: Array(4), transform: {...}}
  type: "Topology"
  ▼ objects:
    ▼ countries:
      type: "GeometryCollection"
      ▼ geometries: Array(177)
        ▼ [0 ... 99]
          ▼ 0:
            type: "MultiPolygon"
            ► arcs: (2) [Array(1), Array(1)]
            id: "242"
            ▼ properties:
              name: "Fiji"
              ► __proto__: Object
            ► __proto__: Object
          ► 1: {type: "Polygon", arcs: Array(1), id: "834", properties: {...}}
          ► 2: {type: "Polygon", arcs: Array(1), id: "732", properties: {...}}
          ► 3: {type: "MultiPolygon", arcs: Array(30), id: "124", properties: {...}}
          ► 4: {type: "MultiPolygon", arcs: Array(10), id: "840", properties: {...}}
```

```
{
  "type": "Topology",
  "objects": {
    "countries": {
      "type": "GeometryCollection",
      "geometries": [
        {
          "type": "MultiPolygon",
          "arcs": [
            [
              [
                0
              ]
            ],
            [
              [
                1
              ]
            ]
          ],
          "id": "242",
          "properties": {
            "name": "Fiji"
          }
        }
      ]
    }
  }
}
```

# 地图数据的表达 – TopoJson & GeoJson

## • TopoJson

```
{
  "type": "Topology",
  "objects": {
    "countries": {
      "type": "GeometryCollection",
      "geometries": [
        {
          "type": "MultiPolygon",
          "arcs": [
            [
              [
                0
              ]
            ],
            [
              [
                1
              ]
            ]
          ],
          "id": "242",
          "properties": {
            "name": "Fiji"
          }
        }
      ]
    }
  }
}
```

## GeoJson

```
▼ {type: "FeatureCollection", features: Array(177)} ⓘ
  type: "FeatureCollection"
  ▼ features: Array(177)
    ▼ [0 ... 99]
      ► 0: {type: "Feature", id: "242", properties: {...}, geometry: {...}}
      ► 1: {type: "Feature", id: "834", properties: {...}, geometry: {...}}
      ► 2: {type: "Feature", id: "732", properties: {...}, geometry: {...}}
      ▼ 3:
        type: "Feature"
        id: "124"
        ▼ properties:
          name: "Canada"
          ► __proto__: Object
          ► geometry: {type: "MultiPolygon", coordinates: Array(30)}
          ► __proto__: Object
      ► 4: {type: "Feature", id: "840", properties: {...}, geometry: {...}}
      ► 5: {type: "Feature", id: "398", properties: {...}, geometry: {...}}
      ► 6: {type: "Feature", id: "860", properties: {...}, geometry: {...}}
      ► 7: {type: "Feature", id: "598", properties: {...}, geometry: {...}}
      ► 8: {type: "Feature", id: "360", properties: {...}, geometry: {...}}
```

# 地图数据的表达 – TopoJson & GeoJson

## TopoJson

- 本质上是JSON格式
- 对处理了GeoJson数据冗余的特点，节约存储空间
- 由D3的作者Mike Bostock制定

## GeoJson

- 本质上是JSON格式
- 官方：GeoJSON is a format for encoding a variety of geographic data structures.
- D3.js的geoPath使用GeoJson格式的地图数据
- <https://www.jianshu.com/p/465702337744>

# Json数据的读取

- Json数据的读取与CSV数据的读取非常相似

```
d3.json('/static/data/countries-110m.json').then(  
  function(data){ ...  
}  
);
```

- 读取后就会直接转换成JavaScript的对象
- （CSV数据读取后会转换成一个对象的数组）

**注意！** 转换后的GeoJson  
不可直接使用，需要根据  
比例尺进一步转换！

D3的geoPath使用GeoJson的格式，因此需要转换：

```
// convert topo-json to geo-json;  
worldmeta = topojson.feature(data, data.objects.countries);
```

# 地图数据的可视化 - geoPath

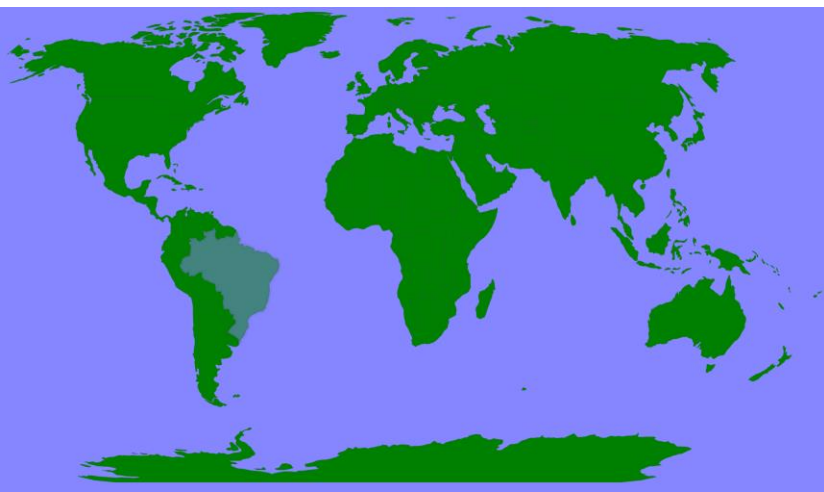
- 传入path的'd'属性的函数
- 用于将GeoJson转换成'path'标签接受的参数
- `const pathGenerator = d3.geoPath();`
- 需要进一步设置如何将GeoJson映射成参数
  - Projection

```
const paths = g.selectAll('path')
  .data(worldmeta.features, d => d.properties.name)
  .enter().append('path')
  .attr('d', pathGenerator)
  .attr('stroke', 'black')
  .attr('stroke-width', 1)
```

# 地图数据的可视化 – geoPath与投影

- 如何将地形的数据映射到画布上？
  - `const projection = d3.geoNaturalEarth1();`
  - `const pathGenerator = d3.geoPath().projection(projection);`
- 类似“比例尺”，地图要画在多大的画布上？
  - `projection.fitSize([innerWidth, innerHeight], worldmeta);`

`d3.geoNaturalEarth1()`



`d3.geoMercator()`



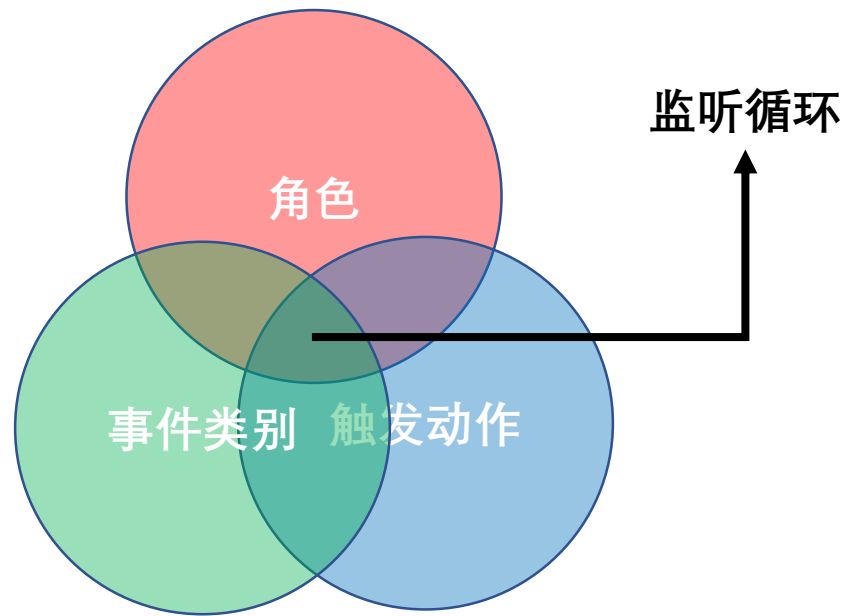
`d3.geoStereographic()`





# 事件

- 什么是事件？
- 思考：在第一人称射击（FPS）游戏中，为什么玩家操控的**角色**  
**点击鼠标左键**会**开枪**、**甚至击中敌人**？



# D3与Web中的事件机制

- 事件的设置对应于D3中的.on('eventName', callback)
  - d3.select('#someprimitive').on('click', myCallback)
  - **图元**.on(**事件类型**, **触发动作**)
- DOM Events (猜一猜下面都是哪些事件类别? )
  - click
  - mouseover
  - mouseout
  - keydown
  - [https://developer.mozilla.org/en-US/docs/Web/Events#Standard\\_events](https://developer.mozilla.org/en-US/docs/Web/Events#Standard_events)

# 为图元设置事件触发

- 确定是什么事件的触发？（**事件类型**）
- 确定函数要如何配置（如传入的数据）？（**触发动作**）
- 根据配置将事件流程预先写好，并把函数作为变量传给图元

```
const paths = g.selectAll('path')
  .data(worldmeta.features, d => d.properties.name)
  .enter().append('path')
  .attr('d', pathGenerator)
  .attr('stroke', 'black')
  .attr('stroke-width', 1)
  .on('mouseover', function(d) {
    d3.select(this)
      .attr("opacity", 0.5)
      .attr("stroke", "white")
      .attr("stroke-width", 6);
  })
```

```
.on('mouseout', function(d) {
  d3.select(this)
    .attr("opacity", 1)
    .attr("stroke", "black")
    .attr("stroke-width", 1);
})
.on('click', function(d) {
  if(lastid !== d.properties.name) {
    tip.show(d)
    lastid = d.properties.name;
  } else {
    tip.hide(d)
  }
})
```

# D3之外的库？

- D3-Tip
- 自动在‘合适’的位置显示对话框
- 不作为D3.js的本体，由community的爱好者们开发的用以辅助D3的库
- 初始化D3-Tip

```
const tip = d3.tip()  
.attr('class', 'd3-tip').html(function(d) { return d.properties.name });  
svg.call(tip);
```

# D3-Tip

- 初始化

```
const tip = d3.tip()  
.attr('class', 'd3-tip').html(function(d) { return d.properties.name });  
svg.call(tip);
```

- 在图元中直接调用
- tip.show(d)
- tip.hide(d)
- 调用接受图元绑定的数据d，然后根据发出‘调用命令’的图元决定位置

# 层叠式样式表 - CSS

- Web三要素的关系
  - Html – 骨架
  - CSS – 皮肤 (**Make your skin look better**)
  - JS – 控制骨架动作
- 插入样式表CSS ( Cascading Style Sheets ) 的三种方式
  - 外部样式表
  - 内部样式表
  - 内联样式
- Code: <https://github.com/Shao-Kui/D3.js-Demos/blob/master/static/html-tutorial/hello-css.html>

# 外部样式表

- 当样式需要应用于很多页面时，外部样式表将是理想的选择。
- 在使用外部样式表的情况下，你可以通过改变一个文件来改变整个站点的外观。
- 每个页面使用 `<link>` 标签在页面头部链接到样式表。

文件头部链接  
mystyle.css文件

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

创建mystyle.css  
文件

```
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("/images/back40.gif");}
```

# 内部样式表

- 当单个文档需要特殊的样式时，就应该使用内部样式表。
- 你可以使用 `<style>` 标签在文档头部定义内部样式表，就像这样：

```
<head>
<style>
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```



# 内联样式

- 当样式仅需要在元素上应用一次时，要使用内联样式。
- 你需要在相关的标签内使用样式（style）属性，Style 属性可以包含任何 CSS 属性。本例展示如何改变段落的颜色和左外边距。

```
<p style="color:sienna;margin-left:20px">这是一个段落。</p>
```

# 多重样式






- 优先级： 内联样式 > 内部样式 > 外部样式

# 为地图设置外部样式表

- 注意，不要忘记在代码主体引入该样式：

```
<!DOCTYPE html>
<html>
  <head>
    <title>Render the Earth</title>
    <link rel="stylesheet" href="/static/css/earth.css">
    <link rel="stylesheet" href="/static/css/d3tip.css">
    <script src="/static/js/d3.min.js"></script>
    <script src="/static/js/topojson.js"></script>
    <script src="/static/js/library/d3-tip.js"></script>
  </head>
```

```
body {
  background-color:  #8686ff;
}

path {
  fill:  green;
  stroke:  rgb(29, 25, 25);
  stroke-opacity: 0.1;
}
```

# END

- 请大家思考并练习?
  - 键盘事件要如何推广?
  - 我们之前介绍过的例子是否可用‘事件处理’升级为交互形式?
- 下次课的内容:
  - 堆叠数据 – ‘Stack’
  - 数据预处理 – 将数据转换成堆叠数据
  - 堆叠数据的可视化

