

D3.js

交互

张松海 张少魁

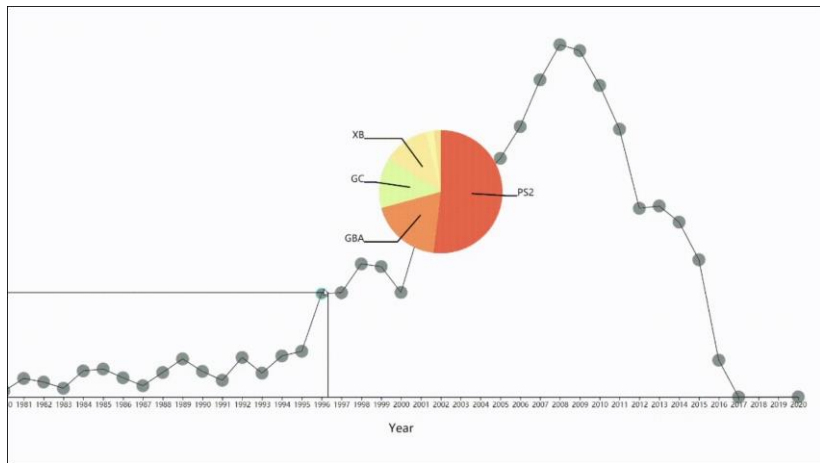
清华大学 可视媒体研究中心

概览

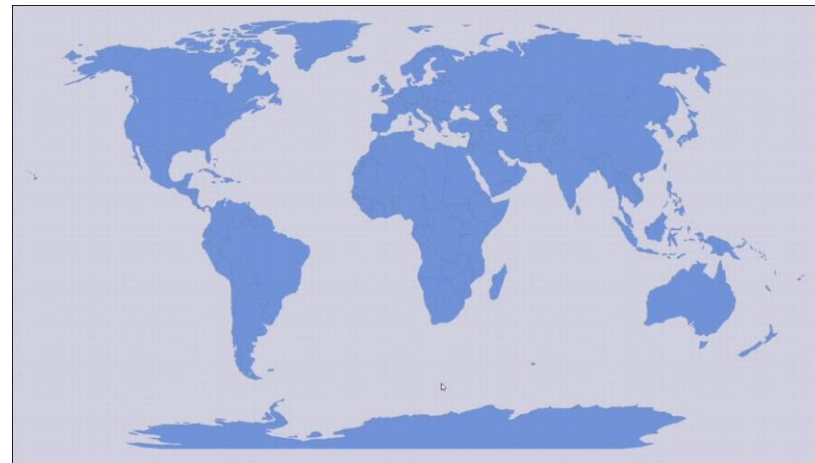
- 交互要素与流程。
- 使用D3.js处理交互事件：
 - 鼠标左键单击。
 - 鼠标移入。
 - 鼠标移出。
- 基于交互的地图可视化。
 - D3-Tip
- 前端交互的事件传递：捕获(capturing)、定向(targeting)与冒泡(bubbling)。
- 更多的交互事件：
 - 鼠标右键单击。
 - 键盘。
- D3.js的版本问题。

交互的要素

- 某一个**实体**(如图元)针对, **用户的某一种事件**, 做出了某一**动作**:
 - 需要编程者给出以上三个方面。



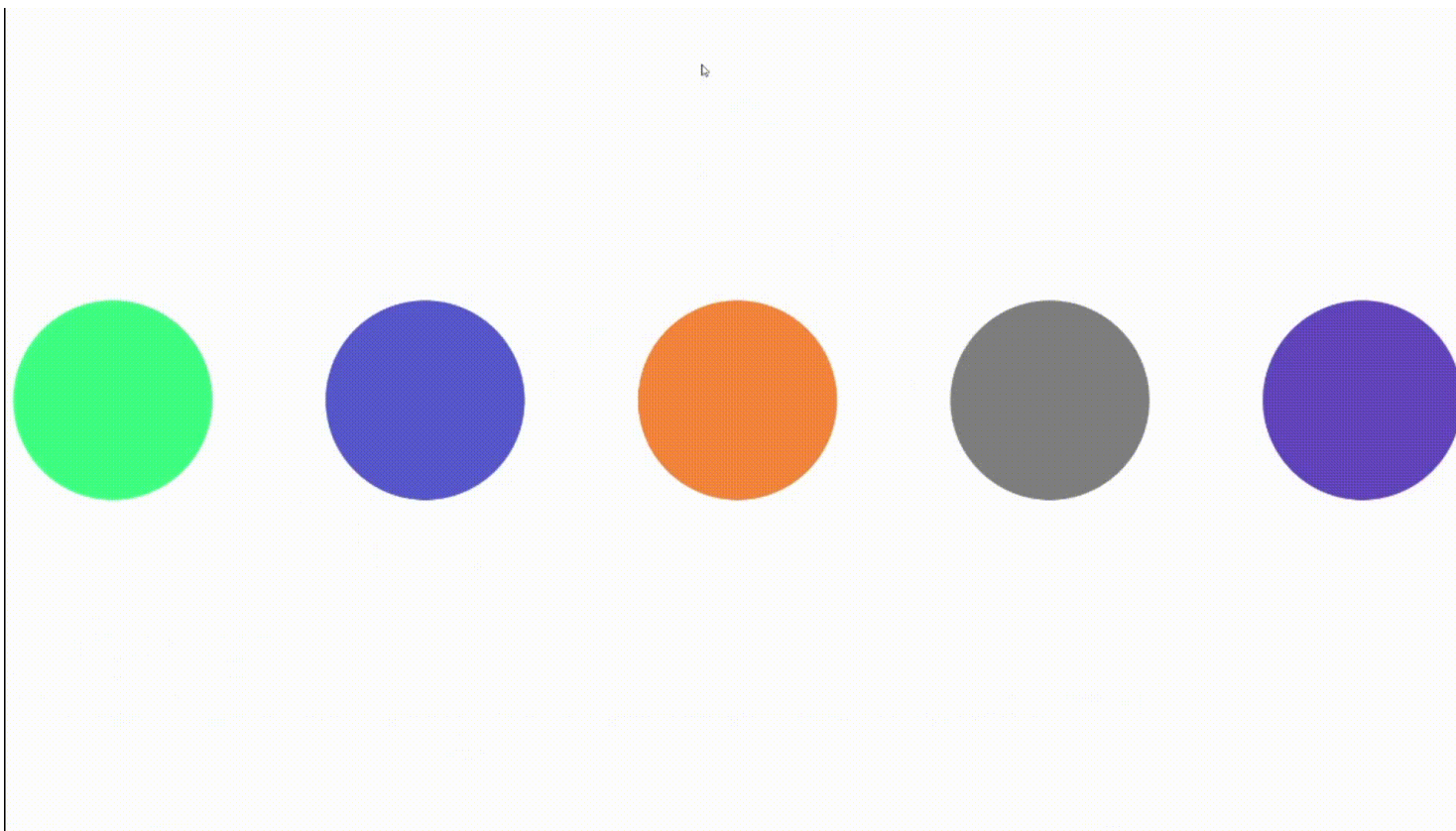
折线图节点-鼠标左键单击-展开饼图



地图的区域(<path>)-鼠标移入-高亮

交互的要素

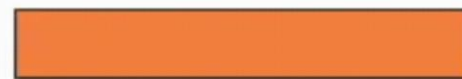
- 对于相同的**图元**，应用不同的**事件**：
 - 左至右：鼠标左键单击、鼠标移入、鼠标移出、键盘、鼠标右键单击。



使用D3.js处理交互事件

- D3.js的事件设置通用语法: `selection.on('eventName', (event, d) => {触发动作})`
 - `图元.on(事件类型, 触发动作)`
 - 'event': 事件对象, 包含这个事件的信息, 如鼠标触发事件的位置、对应的图元等。
 - 'event.currentTarget': 得到被事件触发的图元(Html)。
 - `d3.select(event.currentTarget)`: 基于Html选择并得到图元。
 - 'd': 当前图元绑定的数据, 在回调函数第二个位置。
 - 函数体包含了图元对事件的相应。
 - 下述代码在矩形收到鼠标左键单击后, 为矩形的填充变色。
 - e.g.,

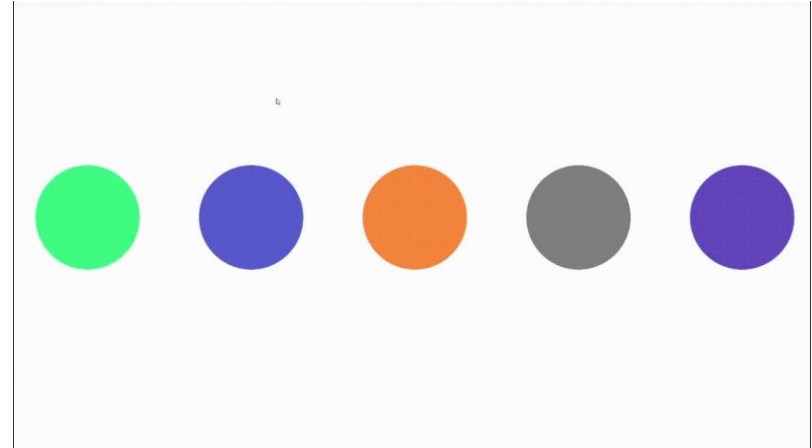
```
d3.select("#my_rect").on('click', (event, d) => {  
    d3.select(event.currentTarget).attr('fill', d3.interpolateRainbow(Math.random()));  
});
```
 - 注意: 语法仅适用于D3.js第6发行版, D3.js的第5发行版及更早版本并不适用。



使用D3.js处理交互事件

- DOM Events:
 - click: 鼠标左键单击。
 - mouseover: 鼠标移‘入’某个图元。
 - mouseout: 鼠标移‘出’某个图元。
 - keydown: 某一个键盘的‘键’被按下。
 - contextmenu: 鼠标右键单击。
 - mousemove: 鼠标移动。
 -
 - <https://developer.mozilla.org/en-US/docs/Web/Events>

鼠标左键单击



- 对应‘click’**事件类别**。

- 编程实例：

- 定义一个统一的函数用来相应点击事件，即随机改变颜色。

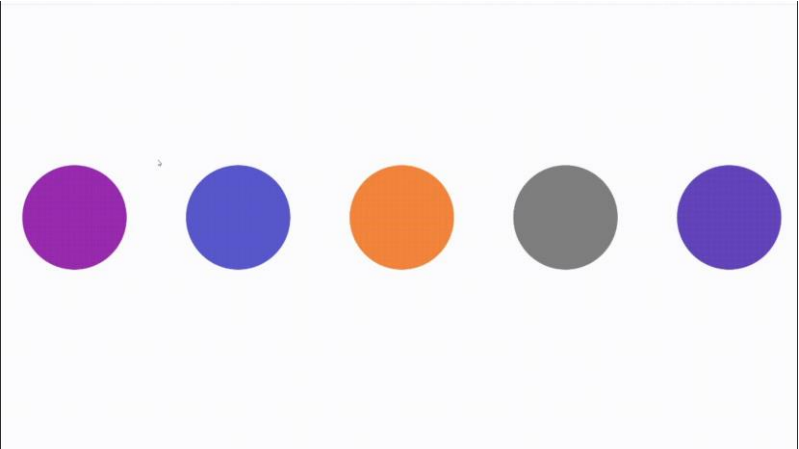
```
const changeColor = (event) => {  
  d3.select(event.currentTarget)  
    .attr('fill', d3.interpolateRainbow(Math.random()));  
};
```

得到事件触发的**图元**。

- 配置鼠标单击事件。

```
svg.append('circle')  
  .on('click', changeColor);
```

鼠标移入



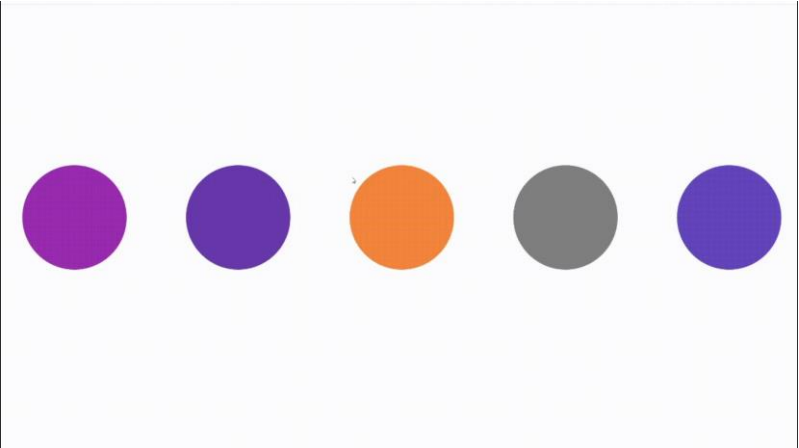
- 对应‘mouseover’事件类别：
 - 鼠标的移入与鼠标的移出是两个不同的事件。
 - 一些图元，如<path>，当填充为‘none’时，只有边缘会触发该事件。
- 编程实例：
 - 定义一个统一的函数用来相应点击事件，即随机改变颜色。

```
const changeColor = (event) => {  
  d3.select(event.currentTarget)  
    .attr('fill', d3.interpolateRainbow(Math.random()));  
};
```

- 配置鼠标移入事件。

```
svg.append('circle')  
  .on('mouseover', changeColor);
```


鼠标移出



- 对应'mouseout'事件类别。
- 编程实例：
 - 定义一个统一的函数用来相应点击事件，即随机改变颜色。

```
const changeColor = (event) => {  
  d3.select(event.currentTarget)  
    .attr('fill', d3.interpolateRainbow(Math.random()));  
};
```

- 配置鼠标移出事件。

```
svg.append('circle')  
  .on('mouseout', changeColor);
```

使用D3.js处理交互事件

- 编程实例:

```
<body>
  <svg width="960" height="400" id="mainsvg"
    class="svgs" style='display: block; margin: 0 auto;'>
    <rect id="my_rect"
      x="10" y="200" width="200" height="30"
      stroke="black" fill="#69b3a2" stroke-width="1"
    />
  </svg>
  <script>
    d3.select("#my_rect").on('click', (event, d) => {
      d3.select(event.currentTarget).attr('fill', d3.interpolateRainbow(Math.random()));
    });
  </script>
</body>
```

基于交互的地图可视化

- 任务：北京市及其各个区的地图数据可视化。
- 数据来源：
 - <http://datav.aliyun.com/tools/atlas/>



D3-Tip

- D3-Tip用于快速生成、添加或隐藏标签：
 - 自动在合适的地方生成标签框。
- D3-Tip不是D3.js自带的接口，但完全基于D3：
 - D3-Tip的导入需要额外的JavaScript脚本。
- 源代码来源：
 - <https://github.com/bumbeishvili/d3-v6-tip>
- 不作为D3.js的本体，由D3社区的爱好者们开发的用以辅助D3的库：
 - 库的开发本身也依赖于D3。
 - 库的使用遵循的D3的语法习惯，即(event, d) => {…}。

D3-Tip

- `const tip = d3.tip().html((event, d) => d.properties.name):`
 - 初始化一个Tip定义。
 - 告知Tip要如何根据输入的数据显示标签, `(event, d) => d.properties.name`
- `svg.call(tip):`
 - 类似坐标轴的添加, 正式渲染出Tip。
- `tip.show(event, d):`
 - 基于事件与图元绑定的数据, 显示Tip。
- `tip.hide(event, d):`
 - 基于事件与图元绑定的数据, 隐藏Tip。

基于交互的地图可视化

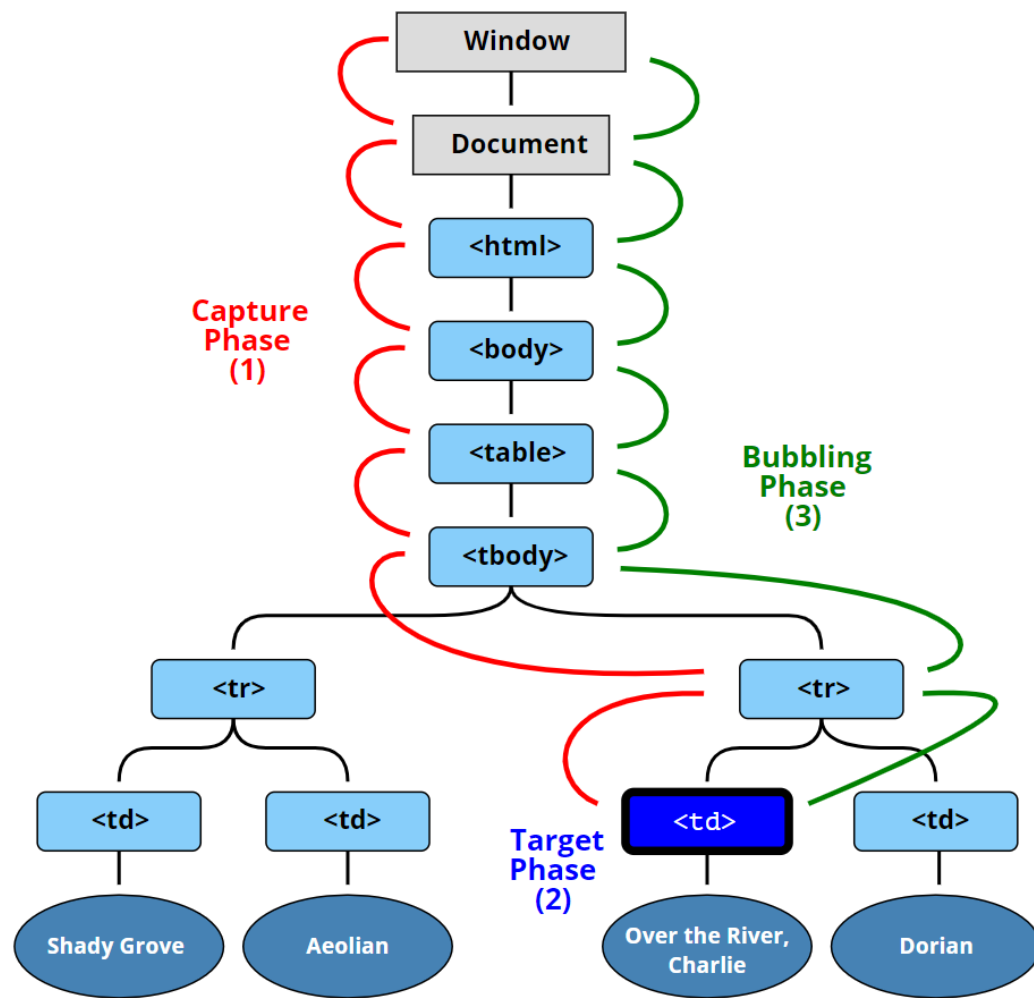
- 编程实例:

```
const tip = d3.tip().attr('class', 'd3-tip').html((EVENT, d) => d.properties.name);
svg.call(tip);

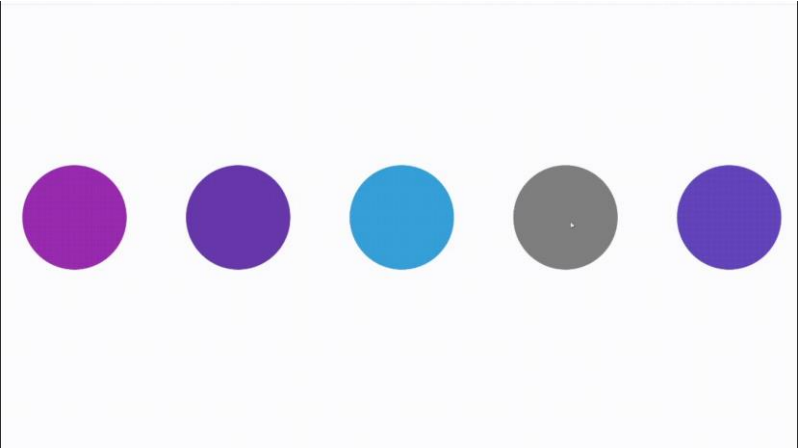
d3.json('beijing.json').then(async data => {
  const projection = d3.geoNaturalEarth1();
  projection.fitSize([innerWidth, innerHeight], data);
  const path = d3.geoPath().projection(projection);
  console.log(path(data.features))
  mainGroup.selectAll('path').data(data.features).join('path')
    .attr('stroke', 'black').attr('fill', 'white')
    .attr('d', path)
    .attr('id', d => d.properties.name)
    .on('mouseover', tip.show)
    .on('mouseout', tip.hide);
});
```

前端交互的事件传递（不做要求）

- Event Flow:
 - 事件的传递经历三个阶段。
 - **Capturing**: 捕获，从根节点(Window)，一路传递到目标节点(图元)。
 - **Targeting**: 事件在目标节点(图元)。
 - **Bubbling**: 事件从目标节点一路传回根节点(Window)。
- Event Flow的过程中，任何一个节点都可以响应事件。
 - 事件也可以在任何一个节点处终止继续传播。
 - `event.stopPropagation()`;
- 除Targeting外，任何事件的相应均可设置在Capturing阶段或Bubbling阶段，甚至同时设置在两边。
 - 可用来决定事件执行的顺序。
- 大多数浏览器都会有分配的默认事件：
 - 如右键单击出现菜单栏。
 - 取消浏览器的默认事件要调用如下接口：
 - `event.preventDefault()`;



键盘



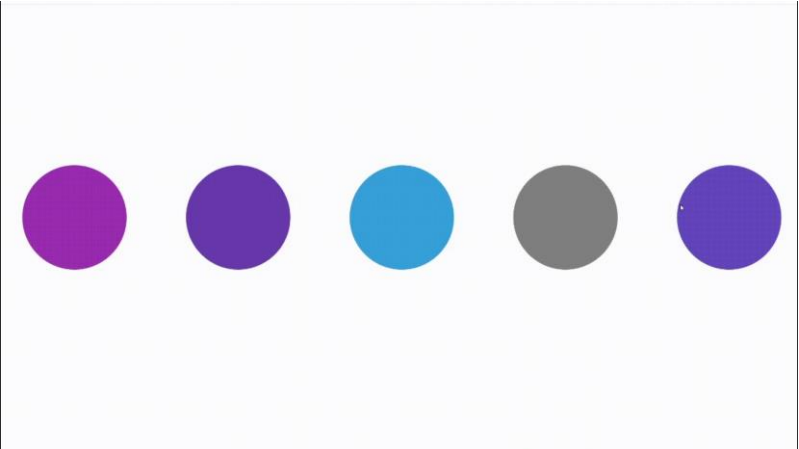
- 对应‘keydown’事件类别。
- 键盘事件对于一般图元通常不适用。
 - （不做要求）
 - 只有可以‘focus’的图元才能响应键盘事件。
 - 键盘事件会被‘focus’的元素获取，如网页常见的文本输入框。
 - SVG的图元在主流浏览器中几乎不响应键盘事件。
 - 在没有其他‘focus’的图元时，需要对<body>标签设置键盘事件。
- d3.select(‘body’).on(‘keydown’, f):
 - 为<body>标签添加键盘事件。
 - 其中，f函数中可以通过任何方式调整任何图元。
 - ‘任何图元’需要编程者提供，因event.currentTarget此时返回<body>。
 - 可直接通过ID制定：d3.select(‘#keyele’).attr(…)
 - 可引入临时变量，并配合‘click’、‘mouseover’事件来制定当前与鼠标交互的图元。
- 不同键盘的输入要通过event.keyCode获得：
 - e.g., event.keyCode === 87 // 判断是否点击了W键。
 - keyCode与实际键盘ID的对应：<https://keycode.info/>。

键盘

- 编程实例：
 - 通过
W/A/S/D
控制图元
上下左右
移动。

```
let isTransitioning = false;
const step = 60;
const duration = 1000;
const keyTranslate = (at, s) => {
  isTransitioning = true;
  let transition = d3.transition().duration(duration).on('end', () => isTransitioning = false);
  d3.select('#keyele').transition(transition).attr(at, function(){
    return +d3.select(this).attr(at) + s;
  });
}
d3.select('body').on('keydown', async function(){
  if(isTransitioning){
    return;
  }
  if(event.keyCode === 87){ // W
    keyTranslate('cy', -step);
  }else if(event.keyCode === 83){ // S
    keyTranslate('cy', step);
  }else if(event.keyCode === 65){ // A
    keyTranslate('cx', -step);
  }else if(event.keyCode === 68){ // D
    keyTranslate('cx', step);
  }
});
```

鼠标右键单击



- 对应‘contextmenu’事件类别。
- 鼠标右键在主流浏览器中均有默认行为：
 - 右键打开菜单(menu)。
 - `event.preventDefault()`用来阻止浏览器默认事件。
 - 如不调用`event.preventDefault()`， 点击鼠标右键时， 会出现默认的菜单栏。
- 编程实例：

```
svg.append('circle')
.on('contextmenu', (event) => {
  event.preventDefault();
  d3.select(event.currentTarget).attr('fill', color(Math.random()));
});
```

Tip: D3.js的版本问题

- 当前D3.js的版本为v6.5.0, 即D3-V6:
 - 于2020年8月27日正式公布D3.js: v6.0.0, 目前为D3 v6.5.0。
 - 截至【2021年3月8日-14点41分】。
- 基于D3.js的代码(可视化作品), 并不保证在不同D3.js的版本兼容:
 - e.g., D3-Tip: 原版的D3-Tip已不再兼容D3-V6, <https://github.com/caged/d3-tip>。
 - e.g., D3-Geo-Voronoi: 将GeoJson转换为Voronoi图不兼容D3-V6前的版本。
- D3-V6中对D3.js的核心机制有所改变, 以交互为例:
 - <https://observablehq.com/@d3/d3v6-migration-guide>
 - d3.event被取消, 由selection.on("mousemove", function(event, d) {...})代替。
- 课程中的所有代码一律以D3-V6为准。