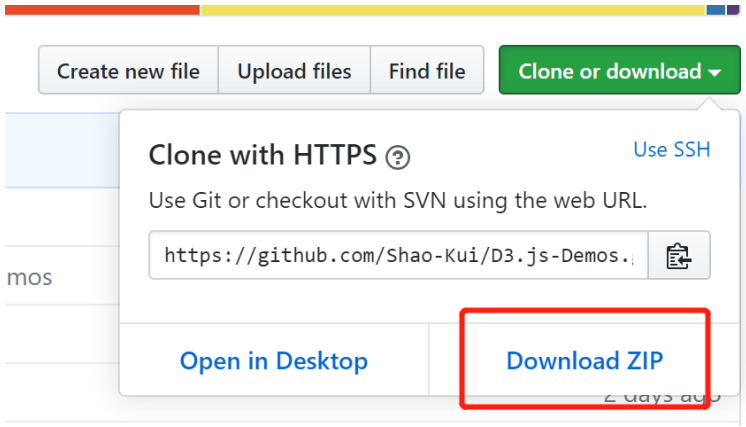
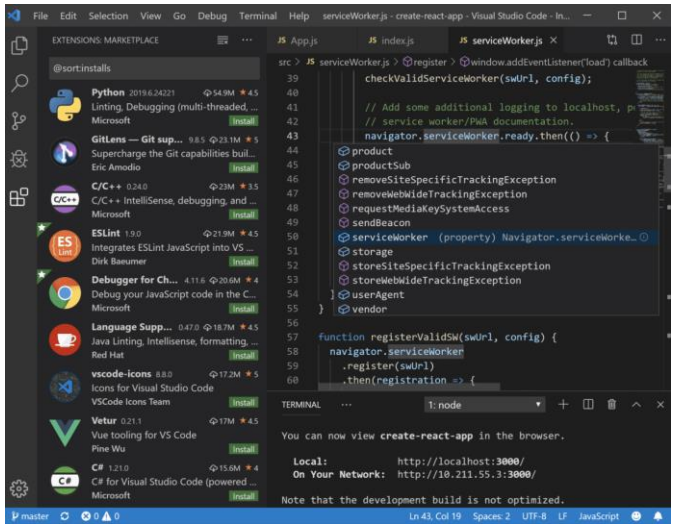
D3.js – 操控SVG

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

Previously…

- 无法打开代码仓库的某个.html文件
  - 大部分.html文件中的链接是相对于服务器的链接，必须通过启动服务器后在浏览器输入url打开；
- VSCODE
  - https://code.visualstudio.com/
- 代码库的下载
- 服务器的访问要用
  - 127.0.0.1
  - 0.0.0.0 ✕





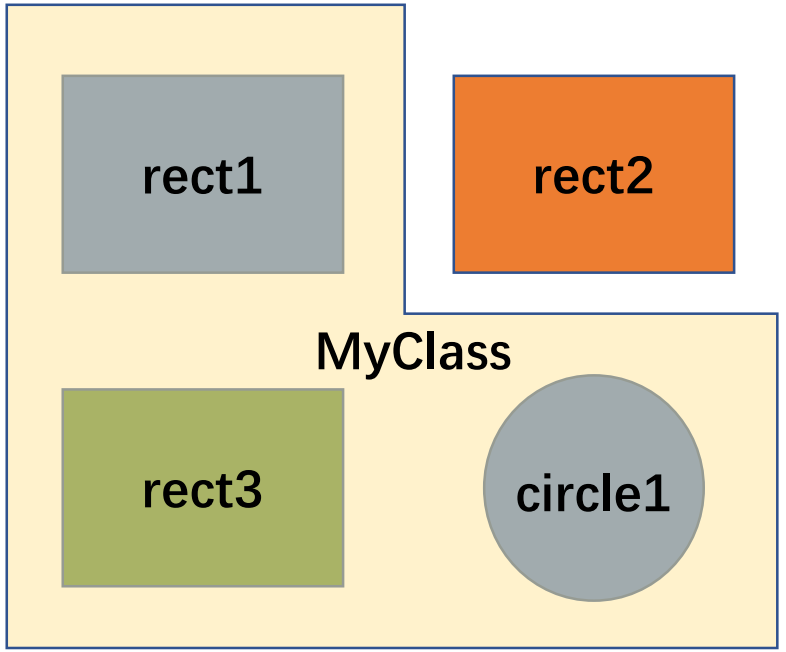
Hello D3

- 使用D3获取、修改、删除节点（图元）
  - *D3的基础
- 比例尺
  - 线性比例尺
  - “条带”比例尺（Band Scale）
- 实例：使用D3绘制简单柱状图
- 引入坐标轴
  - *配置坐标轴根据剩余时间

元素（标签）的标识

- 当我们在一个同学群体中（比如微信群）对某些同学发出通知时:
  - 请**文洋**和**相利**在东主楼集合;
  - 请**CSCG研一**的同学在东主楼集合;
- 在一个群体中，索引个体:
  - 通过唯一的名称索引
  - 通过共同点索引

元素（标签）的标识

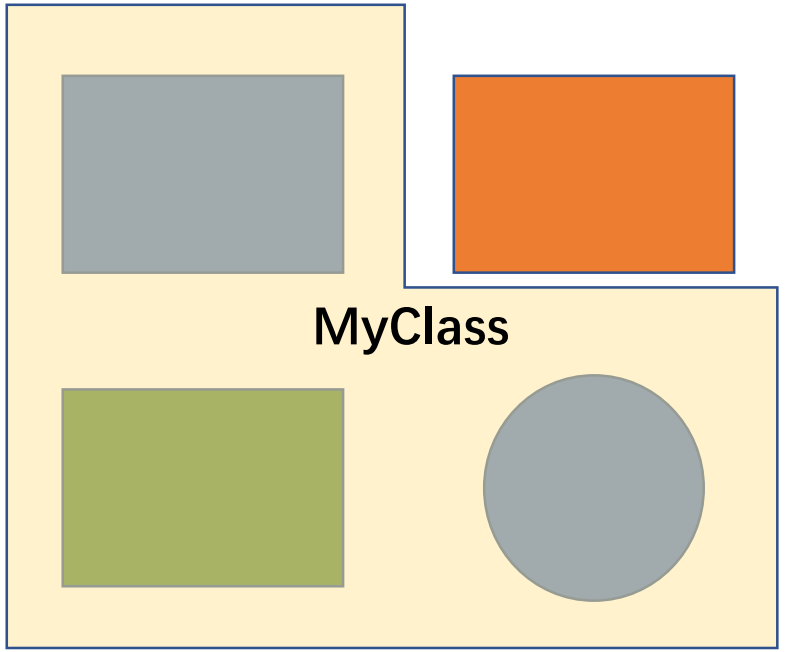


- 操作元素首先需要知道元素的标识
  - 即要得到已有或已经创建的元素
- 元素的ID
  - 可以唯一找到元素的标识符
- 元素的Class
  - 人为赋予的“类别”可以标记元素的集合，**其中的元素标签可以不相同**！
- 元素的标签
  - HTML自带的标签名称，可以找到一批同类别的物体，如所有的“矩形”
  - 使用自带的标签往往难以直接索引到目标元素
  - <rect>, <circle>, <svg>等

元素（标签）的标识

```
<g id='maingroup' transform='translate(100, 100)'>
    <circle id='circle1'
    stroke='black' r='66' fill='#4B8E6F' cx='0'></circle>
    <rect id='rect1' class='class2'
    stroke='black' height='200' width='66' fill='#ff8603' x='100' y='-100'></rect>
    <rect id='rect2' class='class1'
    stroke='black' height='200' width='66' fill='#ffde1d' x='200' y='-100'></rect>
    <rect id='rect3' class='class1'
    stroke='black' height='200' width='66' fill='#7289AB' x='300' y='-100'></rect>
    <text id='myText' class='class2'
    stroke='yellow' font-size='2em' x='400' fill='#1e9d95'>Hey D3! </text>
</g>
```

使用D3查询SVG



- d3.select(…)
  - d3.select(‘#rect1’)
  - 查询ID为’rect1’的元素
  - #表示后面的字符串是一个ID
- d3.selectAll(…)
  - d3.selectAll(‘.class1’)
  - 查询所有class是’class1’的元素
  - d3.selectAll(‘rect’)
  - 查询所有标签是’rect’的元素（rect为SVG中的矩形标签）

使用D3查询SVG

- 基于层级的查询：
  - d3.select(‘#maingroup rect’)
  - d3.selectAll('.tick text’)
  - d3.selectAll(‘#secondgroup rect’)
- 如：’#secondgroup rect’
  - 首先会找到id为secondgroup的标签
  - 进一步找到secondgroup的子标签中是rect的
  - **仍然是对rect做查询，只是结果通过父标签做了筛选**！

  - 注意：这种形式的查询，经常会在配置坐标轴的代码中使用，请至少熟悉其形式！

使用D3查询SVG

- d3.select(…)也可用于查询类别，如
  - d3.select('.class1')
  - 但只会返回找到的第一个元素
- 因此对于class、标签名称的查询建议使用d3.selectAll
- 对于特定某一个元素的查询建议使用d3.select

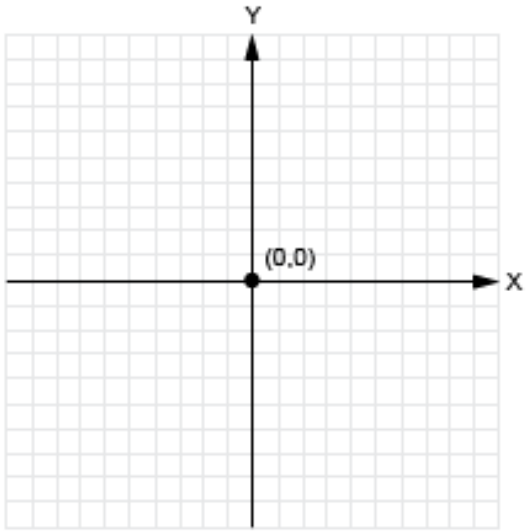
- CSCG研一的同学如果只有一名，则同样可以唯一索引
- 对于ID与Class的使用是编程上的"Design Decision"…

使用D3设置SVG中的属性

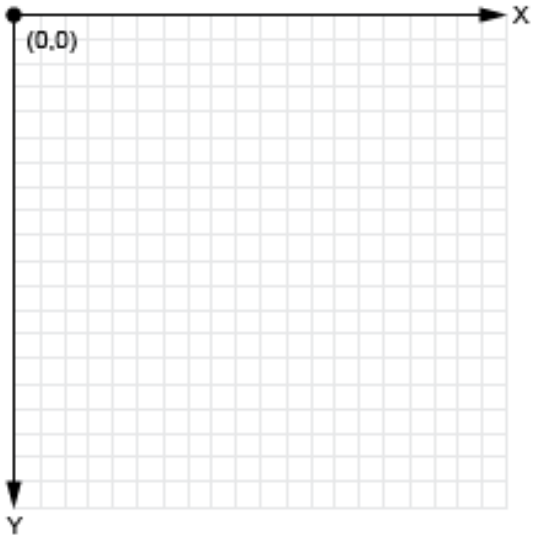
- 常见的属性
  - id, class（特殊的属性，可以使用.attr设置）
  - x, y, cx, cy （注意屏幕的坐标系！见下页）
  - fill, stroke
  - height, width, r (圆的半径)
  - transform -> translate, rotate, scale
- SVG的属性非常多，且属性的取值 **范围&类型** 各不同
  - tip1: 尽可能记住一些常见的属性，以提高编程速度
  - tip2: 遇到不认识or想要设置某个属性，一定要查阅
    - https://developer.mozilla.org/zh-CN/docs/Web/SVG/Attribute
- <rect id='rect3' class='class1'
- stroke='black' height='200' width='66' fill='#7289AB' x='300' y='-100'>

使用D3设置SVG中的属性

- 屏幕空间的坐标系与常见坐标系不同
  - 左上方为原点
  - Y、X分别垂直向下、水平向右



1. Cartesian coordinate space



2. Canvas coordinate space

https://dab1nmslvvntp.cloudfront.net/wp-content/uploads/2014/03/1393985491canvas-coordinate-space.png

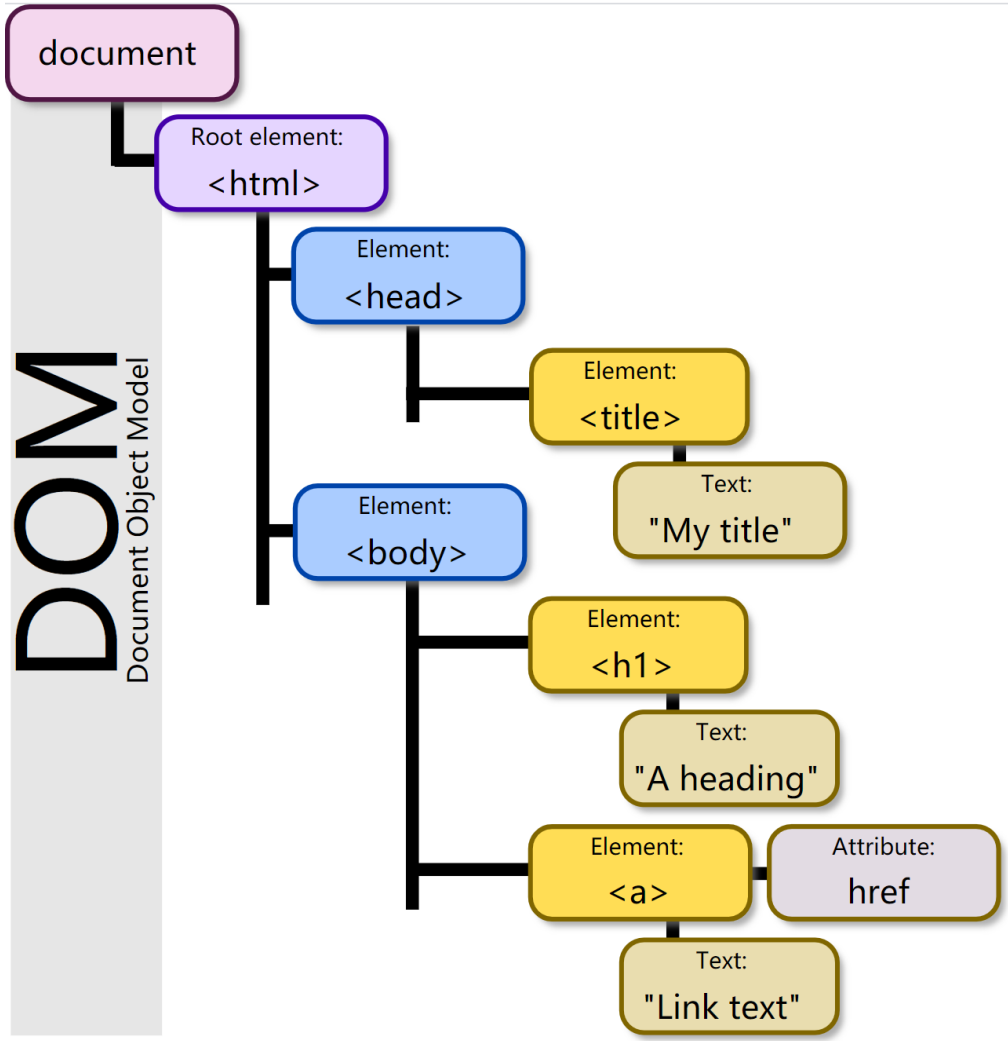
element.attr(…)

- 设置元素的属性: element.attr(‘attr_name’, ‘attr_value’)
  - rect1.attr(‘y’, ‘100’)
  - d3.select(‘#rect1’).attr(‘y’, ‘100’)
- 获取元素的属性: element.attr(‘attr_name’)
- 注意：数值类型的数据扔在HTML中仍由字符串存储
  - 要使用+(strValue)及时转换，如 let value = +(‘233.666’);
  - 活用模板字符串，如
    - let width = 666;
    - .attr(‘transform’, `translate(0, ${width + 100})`)
    - 模板字符串用`…`表示（本质上还是一个字符串）
    - ${ … }可以在字符串中嵌入程序表达式

使用D3设置SVG中的属性

- DOM
  - 父节点的属性会影响子节点
  - 子节点的属性会相对于父节点
- 活用<g>节点可以省掉很多冗余代码!
  - d3.select('#maingroup')
  - .attr('transform', 'translate(200, 100)')

```
<g id='maingroup' transform='translate(100, 100)'>
    <circle id='circle1'
    stroke='black' r='66' fill='#4B8E6F' cx='0'></circle>
    <rect id='rect1' class='class2'
    stroke='black' height='200' width='66' fill='#ff8603' x='100' y='-100'></rect>
    <rect id='rect2' class='class1'
    stroke='black' height='200' width='66' fill='#ffde1d' x='200' y='-100'></rect>
    <rect id='rect3' class='class1'
    stroke='black' height='200' width='66' fill='#7289AB' x='300' y='-100'></rect>
    <text id='myText' class='class2'
    stroke='yellow' font-size='2em' x='400' fill='#1e9d95'>Hey D3! </text>
</g>
```

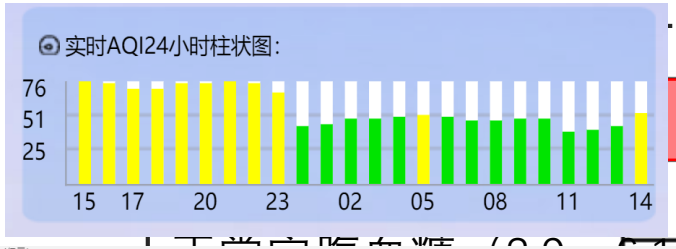


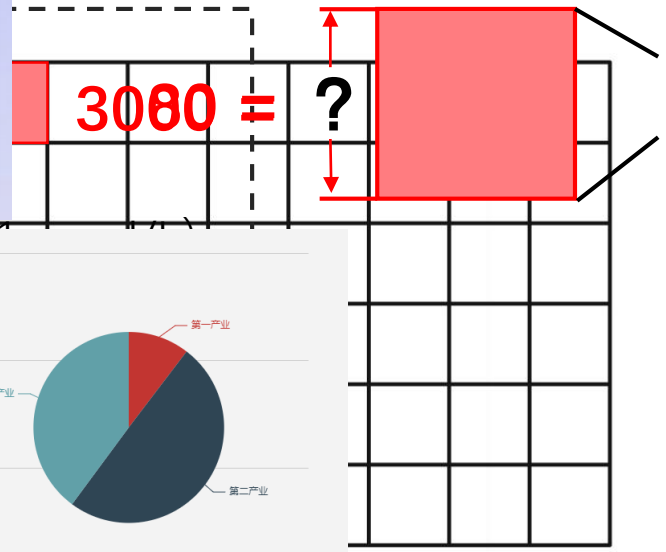
使用D3 **添加&删除** SVG元素

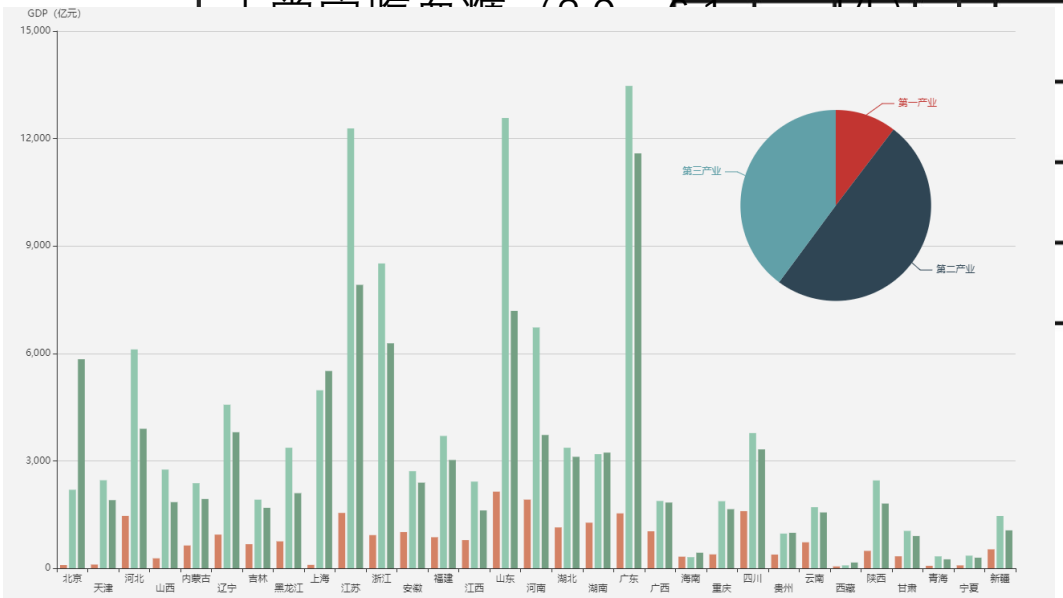
- element.append()
  - const myRect = svg.append(‘rect’);
  - const myRect = d3.select(‘#mainsvg’).append(‘rect’)
  - const myRect = d3.select(‘#mainsvg’).append(‘rect’).attr(‘x’, ‘100’)
- 链式添加
  - const myRect = d3.select(‘#mainsvg’).append(‘g’).attr(‘id’, ‘maingroup’)
  - .append(‘rect’).attr(‘fill’, ‘yellow’)
- **链式调用**：习惯这种编程习惯有助于大家使用D3和浏览D3的代码
- element.remove()
  - 请小心使用
  - 会移除整个标签
- 在debug的过程中可以考虑使用’opacity’属性hack出移除的效果
  - element.attr(‘opacity’, ‘0’)

比例尺

- 比例尺用于把实际的数据空间映射到屏幕的空间
- 比例尺非常重要，会经常同时传给坐标轴与数据!

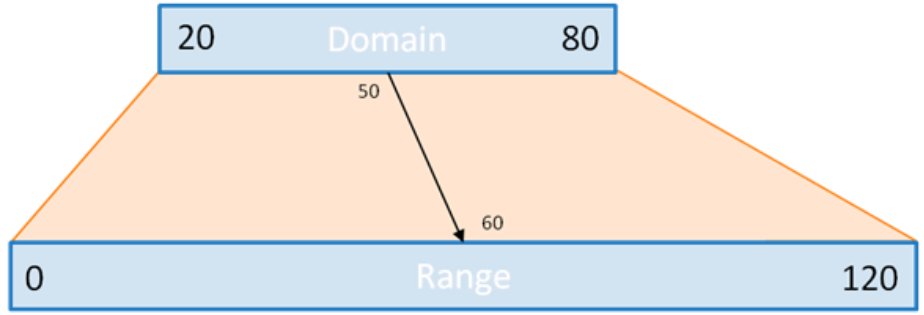






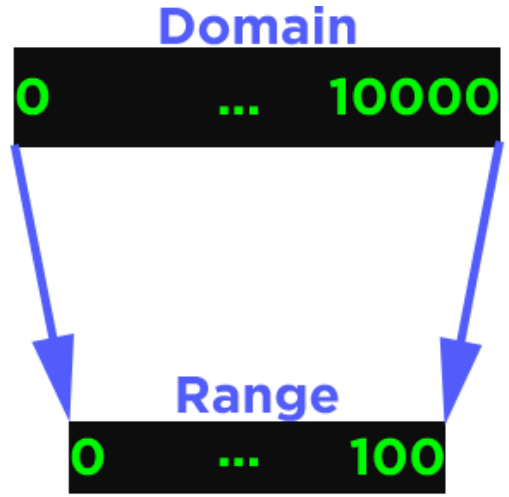
Scale - Linear

```
const xScale = d3.scaleLinear().
.domain([min_d, max_d]).
.range([min, max]);
```



(a)

```
const xScale = d3.scaleLinear()
.domain([0, d3.max(data, datum=>datum.value)])
.range([0, innerWidth]);
```



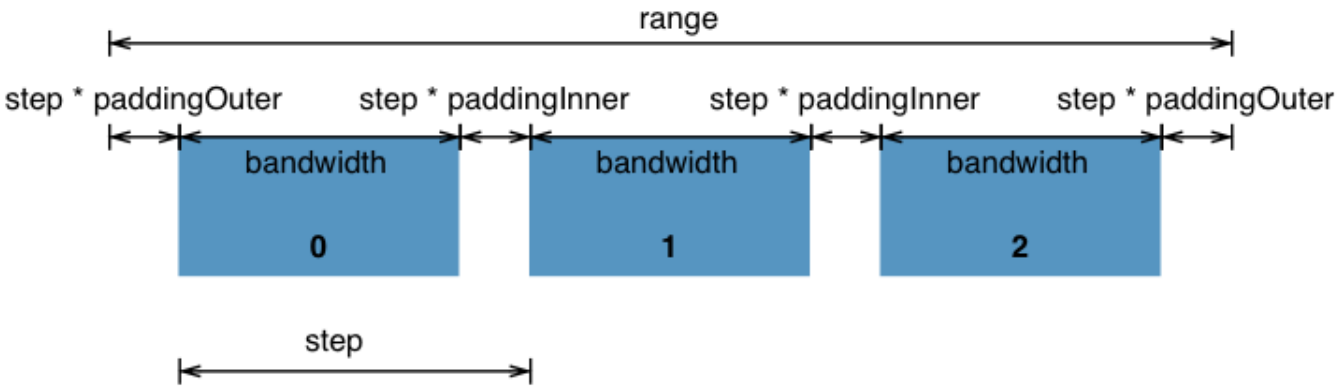
(b)

d3.max(数据，回调：如何提取数据的值)
d3.max: 求出数据某一属性的最大值，比如年龄的最大值

(a) http://www.jeromecukier.net/wp-content/uploads/2011/08/d3scale1.png
(b) https://s3.amazonaws.com/dashingd3js/images/d3.js_scales_scale_domain_down_to_range_300x300.png

Scale - Band

```
const yScale = d3.scaleBand()
.domain(list)
.range([min, max])
.padding(p);
```




```
const yScale = d3.scaleBand()
.domain(data.map(datum => datum.name))
.range([0, innerHeight])
.padding(0.1);
```

https://raw.githubusercontent.com/d3/d3-scale/master/img/band.png

Scale是函数

- 通过d3.scaleLinear或d3.scaleBand得到的返回值本质上是函数
  - 给出数据中的值（domain）
  - 返回映射后的值（range）
  - .domain(…)和.range(…)可以理解为**配置**这个函数（**功能**）的过程
- 比例尺的定义仍适用链式调用

```
const xScale = d3.scaleLinear().
domain([0, d3.max(data, datum=>datum.value)]).
range([0, innerWidth]);
const mapped = xScale(100)
```

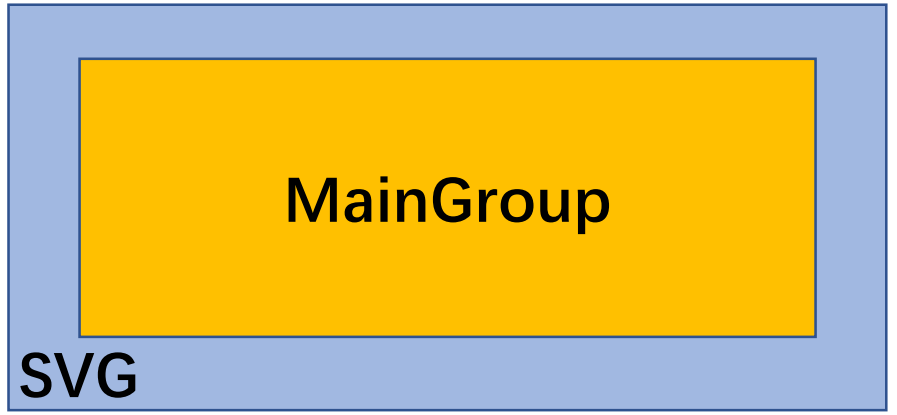
Bar Chart

- 使用select, selectAll, append, attr来实现柱状图!
  - 数据暂时通过前端给定（Fake Data，下节课正式进入Data-Join）

code: https://github.com/Shao-Kui/D3.js-Demos/blob/master/static/d3-tutorial/barchart.html

link: http://127.0.0.1:11666/barchart-tutorial

定义Margin



- SVG对于D3.js是一个“画布”
- SVG范围外的任何内容属于画布之外，浏览器将不予显示

- 定义Margin:
  - const margin = {top: 60, right: 30, bottom: 60, left: 200}
- 计算实际操作的 inner 长/宽
  - const innerWidth = width - margin.left - margin.right;
  - const innerHeight = height - margin.top - margin.bottom;
- 在SVG下额外定义一个组作为新的根节点
  - const g = svg.append('g').attr('id', 'maingroup')
  - .attr('transform', `translate(${margin.left}, ${margin.top})`);

引入坐标轴

- 一个坐标轴为一个group（<g>），通常我们需要两个坐标轴
- 坐标轴中包含：
  - 一个<path>用于横跨坐标轴的覆盖范围
  - 若干个刻度
    - 每个刻度也是一个group
  - 每个刻度下属还会包含一个<line>和一个<text>
    - <line>用于展示坐标轴的轴线，如左到右或上到下
    - <text>用于展示坐标轴的刻度值，如实数、姓名、日期
  - （可选）一个标签用以描述坐标轴

引入坐标轴

- 定义坐标轴（获得结果是**函数**）：
  - const yAxis = d3.axisLeft(yScale);
  - const xAxis = d3.axisBottom(xScale);
  - axisLeft: 左侧坐标轴
  - axisBotton: 右侧坐标轴
- **实际**配置坐标轴：
  - const yAxisGroup = g.append('g').call(yAxis);
  - const xAxisGroup = g.append('g').call(xAxis)
- 实际配置后会发现<g>中增添了与坐标轴相关的元素
- 任何坐标轴在初始化之后会默认放置在坐标原点，需要进一步的平移

关于 selection.call(…)

- （不做要求，希望感兴趣的同学可以理解）
- 函数的输入为**另一个函数**
- **另一个函数**以selection的图元作为输入
- **另一个函数**中会根据函数体的内容修改selection对应的图元
- 定义一个空白的<g>，D3会帮助我们定义好**另一个函数**，我们通过.call(…)让<g>得以在**另一个函数**中修改
  - const yAxis = d3.axisLeft(yScale);
  - const yAxisGroup = g.append('g').call(yAxis);
- 对函数式编程敏感有助于学习D3 & 理解D3的代码!

配置坐标轴

- 可以对坐标轴的风格进行修改：d3.selectAll('.tick text').attr('font-size', '2em’);
- .tickSize来设置网格线
- 坐标轴的标签加入不在D3接口的负责范围内：
  - 通过对应组 .append(‘text’)来人为实现
  - （左）纵轴坐标需要 .attr(‘transform’, ‘rotate(-90)’) 来旋转
  - 纵轴坐标旋转后，x / y 会颠倒甚至取值范围相反
  - 回忆DOM：父节点的属性会影响子节点，而坐标轴默认的’fill’属性是‘none’，因此请一定手动设置文字颜色 .attr('fill', 'black')

操控SVG - End

- 下节课会正式开始:
  - 读取数据
  - Data-Join
  - D3的动态可视化（动画）！
- 请大家使用课上的Demo稍加熟悉D3的增删改查
  - code: https://github.com/Shao-Kui/D3.js-Demos/blob/master/static/d3-tutorial/manipulation.html
  - 鼓励大家用D3作画（比如表情 😄）