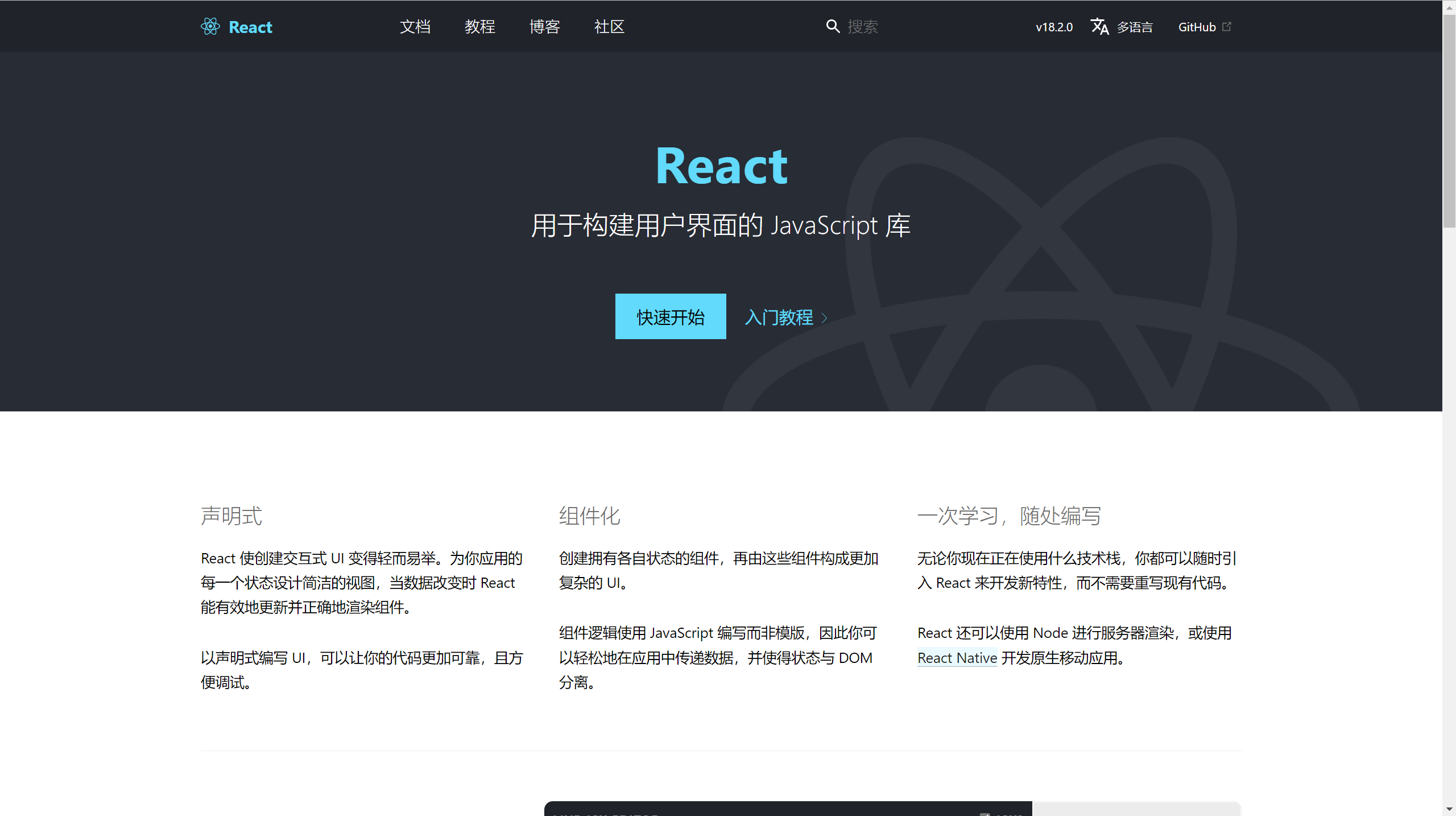
**数据可视化课程实验指导书**

**实验二 用umi+react框架可视化并实现交互**

**React简介**



React 是一个用于构建用户界面的 JavaScript 库，主要用于构建 UI，而不是一个 MVC 框架，但可以使用 React 作为 MVC 架构的 View 层轻易的在已有项目中使用，它是一个用于构建用户界面的 JavaScript 库，起源于 Facebook 的内部项目，用来架设 Instagram 的网站，于 2013 年 5 月开源。React 拥有较高的性能，代码逻辑非常简单，越来越多的人已开始关注和使用它。

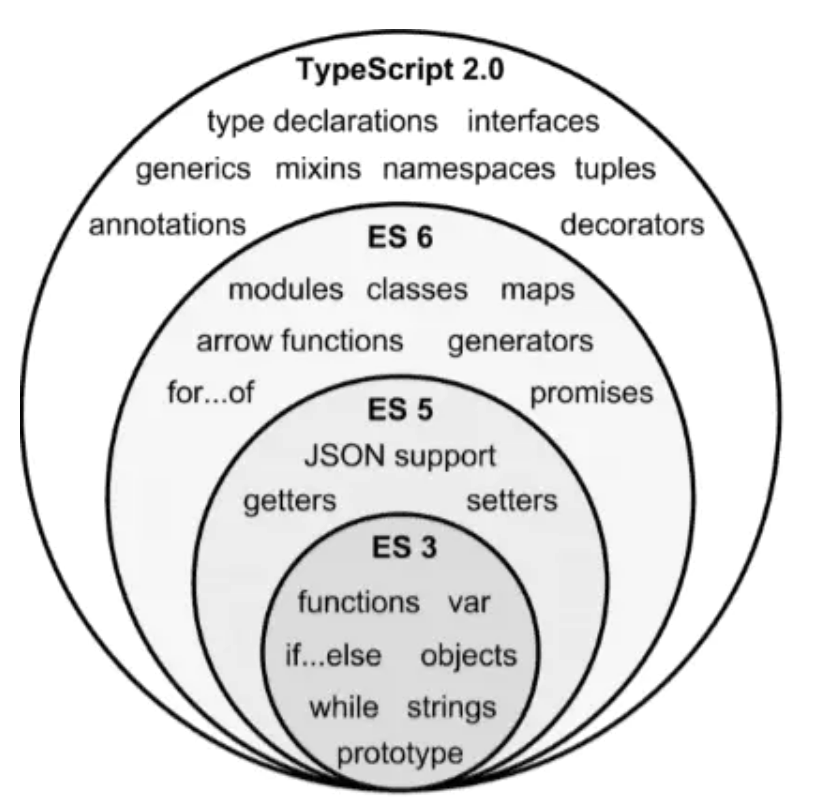
**Umijs简介**



umi，是一个可插拔的企业级 react 应用框架。umi以路由为基础的，支持类 next.js 的约定式路由，以及各种进阶的路由功能，并以此进行功能扩展，比如支持路由级的按需加载。然后配以完善的插件体系，覆盖从源码到构建产物的每个生命周期，支持各种功能扩展和业务需求，目前内外部加起来已有 50+的插件。

umi 是蚂蚁金服的底层前端框架，已直接或间接地服务了 600+ 应用，包括 java、node、H5 无线、离线（Hybrid）应用、纯前端 assets 应用、CMS 应用等。他已经很好地服务了我们的内部用户，同时希望他也能服务好外部用户。

**Typescript简介**



TypeScript 是一种由微软开发的自由和开源的编程语言。它是 JavaScript 的一个超集，而且本质上向这个语言添加了可选的静态类型和基于类的面向对象编程。

TypeScript 扩展了 JavaScript 的句法，所以任何现有的 JavaScript 程序可以不加改变的在 TypeScript 下工作。TypeScript 是为大型应用之开发而设计，而编译时它产生 JavaScript 以确保兼容性。

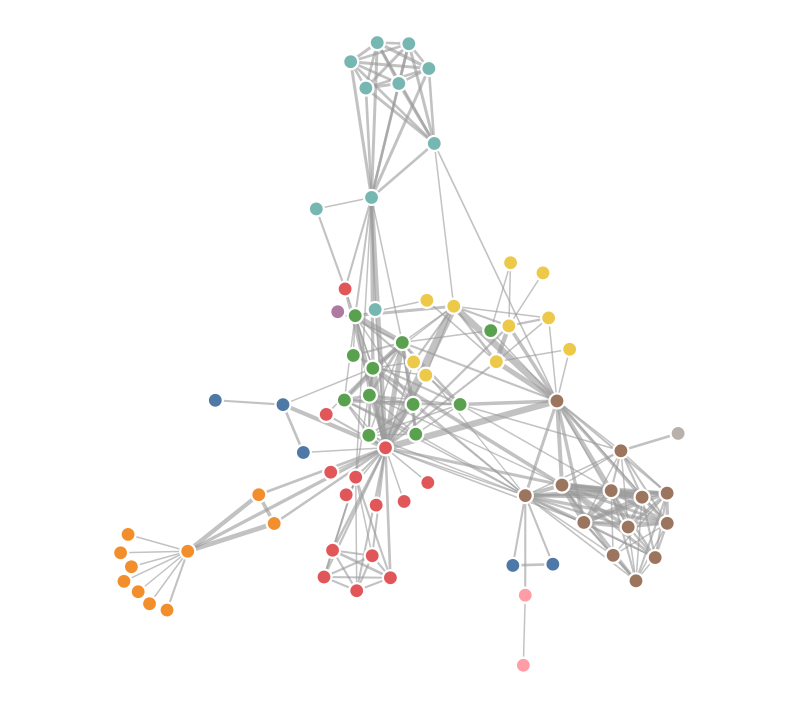
**d3简介**



D3，全称Data-Driven Documents（数据驱动文档），是一个用来使用Web标准做数据可视化的JavaScript库。 它将可视化、交互和DOM操作方法完美结合，帮助我们使用SVG, Canvas 和 HTML技术让数据变得生动有趣。它比Processing这样的底层绘图库更简单，比Echarts这样高度封装的图表库更自由。

学习和使用d3之前需要了解基本的前端知识，包括JavaScript、HTML5、CSS3、SVG（或者Canvas）等。本实验使用svg来绘图，相比于canvas，它更简单、直观、对交互的支持更友好。

**力导向图简介**



力导向图形绘制算法是以美观的方式绘制图形的一类算法。它们的目的是将一个图的节点定位在二维或二维三维空间中，这样所有的边或多或少都是等长的，交叉的边越少越好。方法是根据边和节点的相对位置在边和节点的集合中分配力，然后利用这些力模拟边和节点的运动。

**模拟仿真系统：**仿真模拟系统中存在多个节点和多种类型的力，通过力控制节点的运动，每个节点都在多个力的作用下不断发生移动，直到系统趋于平衡。中间会发生多次tick事件，每次tick，仿真系统都会更新节点的位置，且系统的能量(alpha)也会逐渐降低，直到达到某个数值(alphaMin)，整个图表就停止运动了。

**节点：**节点是一个对象数组，对象的属性没有限制，你可以添加多种信息来控制图表的渲染(例如颜色大小等)，每次tick都会更新节点的位置(x,y)和速度velocity(vx,vy)。

**力：**力驱动着整个系统运动，你可以给系统添加力，控制节点的运动。

**Links：**定义了节点之间的关系，通过节点间的连线定义。同时links也是创建连接力(forceLink)必不可少的东西。例如：



**数据集：**



Nodes代表节点信息，links代表边，source代表源节点，target代表目的节点，links根据id构建网络，交互产生的直方图根据节点的group信息绘制分布情况。

**实验要求**：

请同学们提前配置好nodejs并启用yarn。使用yarn add @types/d3导入d3包。

代码文件夹中已提供了umij的基本架构（说明见<https://v3.umijs.org/zh-CN/docs/directory-structure>）。miserables.json为数据集，按照要求实现如下交互性可视化图表。构建如图1所示的力导向图。为力导向图添加矩形选择框，并根据选择框所选择的节点绘制直方图，如图2所示。

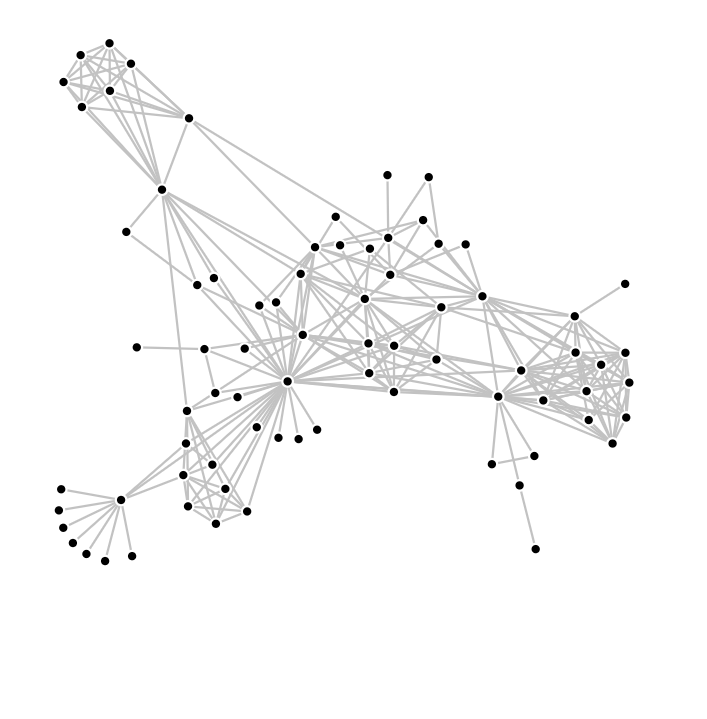


图1：选择框未选择的状态

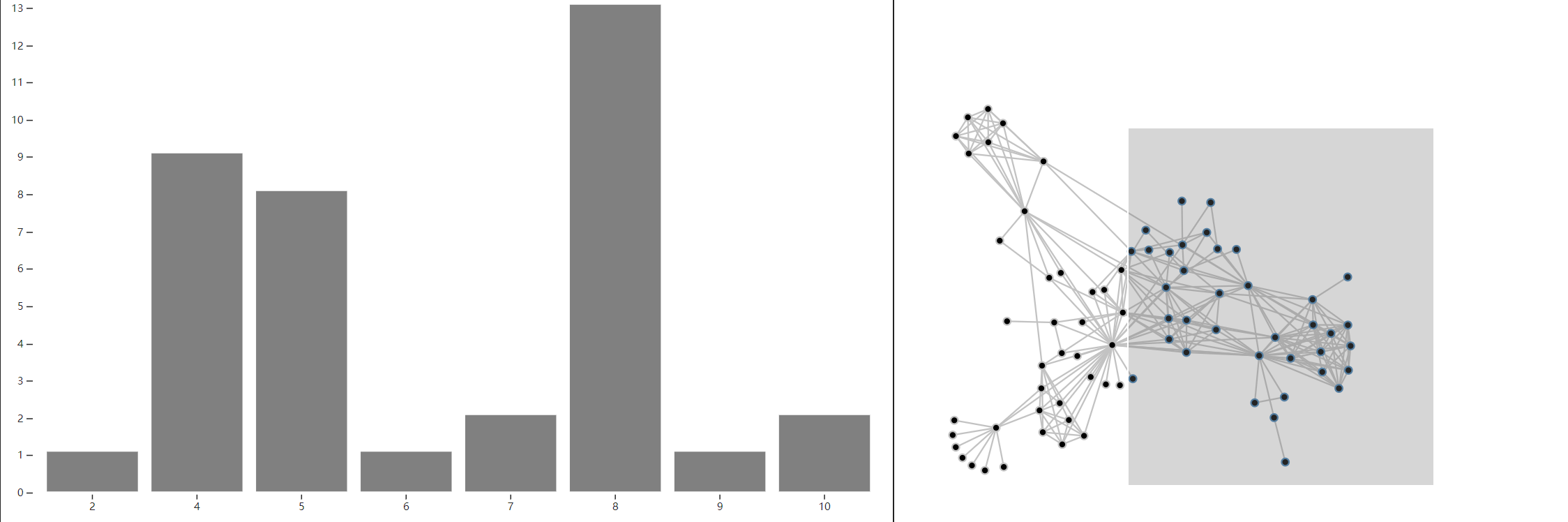
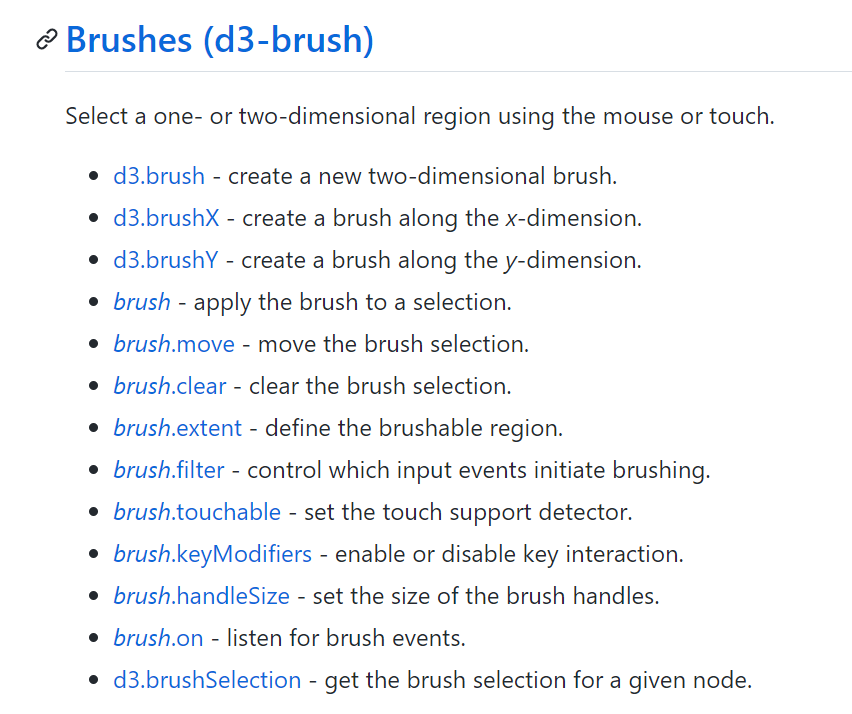
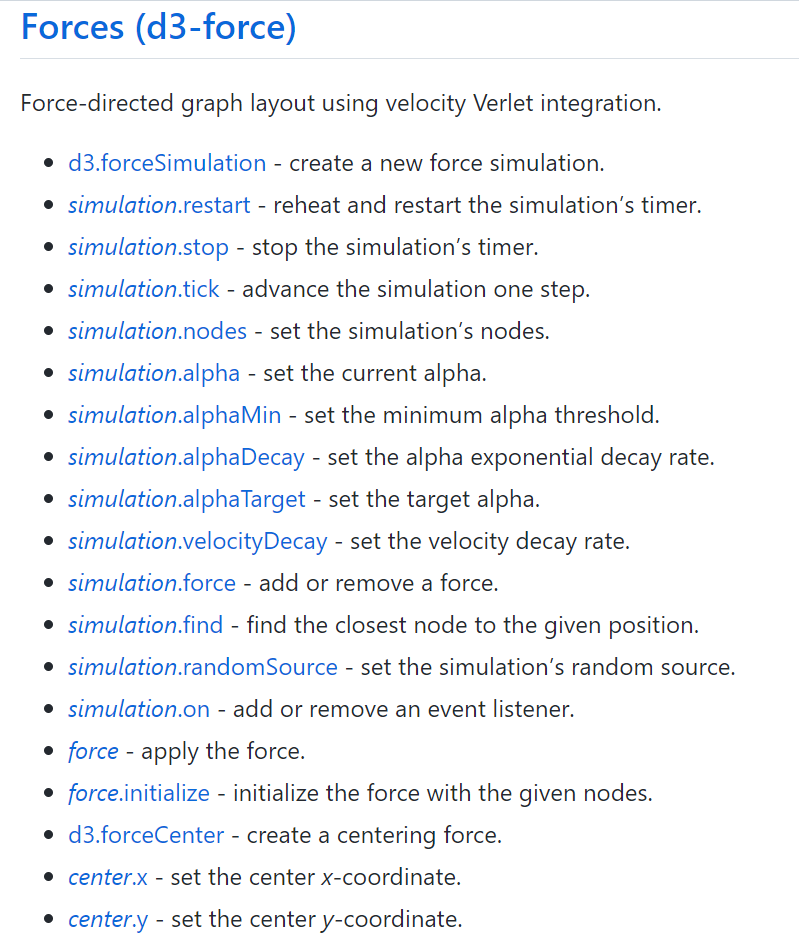


图2：选择框框选节点，显示所选中节点的分布直方图

**步骤一：绘制力导向图。**

根据数据集中的nodes和links构建网络，links中的source代表源节点，target代表目标节点。首先要读取graph.json文件分析出绘图需要的节点和边。在Scatter2中的index.tsx补全代码，绘制力导向图并添加矩形选择框。可能会用到的API：



注：

1. 被矩形框选中后的节点需要高亮显示。

2. 节点、边的样式可以自定义，不一定与图中完全一致，高亮效果也可以自己设置，图中是被选中的节点边框变蓝色。

3. ts语法和js语法类似，可以暂时将数据都定义为any类型就不会报错。

4. 熟悉react.js的jsx语法，以及怎样传出数据。

**步骤二：绘制直方图。**

由框选的数据作为参数传入Scatter1中，补全index.tsx中的代码。要求根据框选的数据获取数据的分组，并根据分组情况绘制分布直方图，x轴代表分组名，y轴代表每个分组出现的个数。直方图根据力导向图的选择框实时更新。

注：

1. 根据矩形框获取的值自动调整所有轴。不要硬编码任何值。
2. 注意y轴比例尺的设置与x轴有所不同，网页中的坐标原点是在左上角的，x轴向右，y轴向下。
3. 绘制直方图，添加矩形，设置动画平滑过渡。
4. 横纵坐标均使用线性刻度。
5. 注意每次绘图前先清空画布，不然会导致每次框选得到的直方图叠加在一起。

提示：提供的代码是不完整的，请同学们补全所有代码后在终端使用yarn start启动项目。重点关注src文件下的内容，其他根目录下文件是已经写好的配置文件，可以查看，谨慎修改。

**可能有用的链接：**

React：https://react.docschina.org/

Umij：https://v3.umijs.org/zh-CN

力导向图：<https://observablehq.com/@d3/force-directed-graph>

Brush选择框：<https://observablehq.com/@alejandrokennedy/basic-brush-example>

https://github.com/mbostock/d3/wiki/SVG-Controls#brush

Linear scale：<https://github.com/d3/d3-scale#linear-scales>

在 D3 中处理嵌套数据的有用资源：http://bl.ocks.org/phoebebright/raw/3176159/