Who runs china 实现方式简介

这个说明是专门为程序员准备的，如果想要软件生成现在的效果，可以参考

https://flourish.studio/visualisations/survey-data/

一、展现方式：

分为动画层和交互层。

动画层主要用canvas来渲染粒子的运动。性能较svg优异。

交互层用d3和svg来绘制粒子静止时的图表，绑定交互事件方便，也可以用css来设置交互样式和交互动画。编码方便。

二、流程控制：

主要有两个：页面滚动事件控制以及fsm渲染控制。

1.页面滚动事件控制

页面滚动用到了storytelling神器 scrollama.js 库。
scrollama会触发页面滚动到不同区域时的事件（跨越不同step触发事件， 单个step内部滚动触发事件 ）
本项目可看startScroll.js

2.自定义的Fsm渲染控制：

fsm有限状态机简单概念可以参考这篇文章 。
在项目中，页面被分为不同的step, 如性别，年龄等，页面滚动到某个step，fsm就进入一种状态。每个状态包含四个函数：
1). enter 进入某个状态。初始化每个粒子的大小，颜色，速度等
2）update 更新某个状态。如果还在移动阶段，更新每个粒子的位置。如果已经结束，直接返回。
3）endTrans 如果粒子移动已经结束，触发该函数。通常会绘制svg交互层，添加交互层的交互事件等等。另外搜索高亮某个粒子时，也会调用该函数。
4). Leave 退出某个状态。通常会清空svg交互层。
Enter某个状态前，会执行前一个状态的leave函数。update会不断调用直致状态结束。

```
// 循环更新fsm
    function renderScene(timestamp) {
            fsm.update(timestamp);
        requestAnimationFrame(renderScene);
    }
    renderScene();

// fsm 主流程
    fsm = {
        current: null,
        next: null,
        update: function (timestamp) {
            if (fsm.next) {
                var change = function () {
                    fsm.current = fsm.next;
                    fsm.next = null;
                    stages[fsm.current].enter();
                    stages[fsm.current].update(timestamp);
                };
                if (fsm.current) {
                    stages[fsm.current].leave(change);
                } else {
                    change();
                }
            } else if (fsm.current) {
                stages[fsm.current].update(timestamp);
            }
        },
        trans: function (next) {
            if (fsm.current !== next) {
                fsm.next = next;
            }
        },
        endTrans: function () {
            var endTrans = stages[fsm.current].endTrans;
            if (endTrans) {
                endTrans();
            }
        }
    };

// fsm状态，以下为random状态示例
    var stages = window.stages = {
        'random': {
            stage: 'random',
            enter: function () {
                this.isFinished = false;
                this.start = null;
                this.duration = durationTime;
                var stage = this.stage;
                var duration = this.duration;

                var r = 3;
                function getRandomColor() {
                  var letters = '0123456789ABCDEF';
                  var color = '#';
                  for (var i = 0; i < 6; i++) {
                    color += letters[Math.floor(Math.random() * 16)];
                  }
                  return color;
                }
                // compute v
                data.forEach(function (d, i) {
                    var stagePosition = positions[stage];
                    d.position = {
                        startX: d.position.x,
                        startY: d.position.y,
                        endX: stagePosition[i].x,
                        endY: stagePosition[i].y,
                        x: d.position.x,
                        y: d.position.y,
                        vx: (stagePosition[i].x - d.position.x) / duration,
                        vy: (stagePosition[i].y - d.position.y) / duration,
                        r: r,
                        color: d.randomColor || d.position.color || getRandomColor(),
                        index: i
                    };
                    d.randomColor = d.randomColor || d.position.color;
                });
            },
            update: function (timestamp) {
                if (this.isFinished) {
                    return;
                }

                if (!this.start) this.start = timestamp;
                var progress = timestamp - this.start;
                var duration = this.duration;
                // console.log(progress, timestamp, duration, durationTime);
                if (progress >= duration) {
                    this.isFinished = true;
                }

                var setData = function () {
                    data.forEach(function (item, i) {
                        var d = item.position;
                        if (progress >= duration) {
                            d.x = d.endX;
                            d.y = d.endY;
                        } else {
                            d.x = d.startX + d.vx * progress;
                            d.y = d.startY + d.vy * progress;
                        }
                    });
                };

                var draw = function () {
                    ctx.clearRect(0, 0, canvas.width, canvas.height);
                    data.forEach(function (item, i) {
                      var d = item.position;
                      ctx.beginPath();
                      // arc(x, y, radius, startAngle, endAngle, anticlockwise)
                      ctx.arc(d.x, d.y, d.r, 0, 2 * Math.PI, false);
                      ctx.fillStyle = d.color;
                      ctx.fill();
                      ctx.closePath();
                    });
                    if (progress >= duration && (!isMobile.any || recentDeputy !== null)) {
                        stages[fsm.current].endTrans();
                    }
                };

                setData();
                draw();
            },
            endTrans: function () {
                if (!stages[fsm.current].isFinished) {
                    return;
                }
                setTimeout(function () {
                    $mysvg.show();
                    mysvgCircle
                        .attr('cx', function (d, i) { return d.position.x; })
                        .attr('cy', function (d, i) { return d.position.y; })
                        .attr('r', function (d, i) {
                            var r = recentDeputy === d.index ? 5 : d.position.r;

                            if (recentDeputy === d.index) {
                                console.log(i, d);
                                console.log(r);
                            }
                            return r;
                        })
                        .attr('fill',  function (d, i) { return recentDeputy === d.index ?
highlightColor : d.position.color; })
                        .classed('active', function (d) { return recentDeputy === d.index; })
                    if (!isMobile.any) {
                        mysvgCircle.on('mouseover', null);
                        mysvgCircle.on('mouseover', function (d, i) {
                            showPerson(d);
                        });
                        mysvgCircle.on('mouseout', hidePerson);
                    }
                }, 0);

            },
            leave: function (cb) {
                $mysvg.hide();
                cb && cb();
            },
            isFinished: false
        },
        ....
    }
```

三、布局算法

1. 屏幕适配。项目采用了scaleToWindow

这个库可以把一个div缩放，撑满屏幕，效果类似于background 图片的position设置为contain。所以我们只做了PC、移动2种固定的分辨率，然后缩放。极大地简化了布局算法的编写。

2. 文字粒子的定位

我们的设计师很认真地画出了文字粒子的AI图。AI图导出成svg, 就有了各个圆的位置、大小和颜色。通过解析svg文本文件，就得到了文字粒子的定位数据。

3. 常规图表的定位

这个是自己编写的。设计稿中有图例，数字，根据这些图例和数据，手工调整了每一群粒子的位置。

四、其他

这个项目的交互和视觉设计是出彩的。从技术上来说，没啥新东西，canvas和svg 8、9年前都有了，那时候也可以做出来了。但是如果放3年前，这个项目不太会得到这样的传播。让页面在微信浏览器里可以播放，是传播开来的一个重要前提。三年来手机性能的提升，网络速度的提升，流量资费的下降，浏览器兼容性的提升等等各种环境因素，才使这样的可视化传播成为可能。5G到来之后会怎样，画面太美不敢想。

项目的源码没有压缩，可以直接下载看到。这些年来好多可视化的前端页面代码压缩了，增加了学习的困难程度。旧时王谢堂前燕，飞入寻常百姓家。几年前再牛逼的技术，到今天也是稀松平常了，所以就不压缩了，方便别人学习。毕竟可视化开发相对于其他前端开发，更依赖创意、设计和数据。即使这个简单的可视化，其实也有十几人奋战半月，我们几个开发的人只是吃饼的，种麦的、做饼的是产品、设计、编辑、审稿人、耐烦的部署工程师、迭代了十几版本的数据工程师们、测试等等。有幸参与这个有意思的项目也算荣幸。

作为一个工程能力强，演讲能力几乎为0的工程师，我觉得我已经尽力去把这个技术层面的东西讲清楚了。有问题再问吧。

最后无耻地打小广告，欢迎关注数可视微信号: VDataWorks

团队定制各种数据可视化，作品获奖。

感谢下面的示例，给了不少启发：

https://www.cnblogs.com/hongru/archive/2012/03/28/2420415.html

https://pudding.cool/2017/03/hamilton/

https://flourish.studio/visualisations/survey-data/