为了确定 ENOW 在图形渲染的技术方案选型, 我们实施了图形渲染的性能测试. 第一步调研主要关于 canvas 和 svg 的选择:

svg 基于 dom 渲染, 接口易用并且可以使用原生事件, 但是不同浏览器的渲染差异造成在复杂特效的情境下表现差异较大, 调试困难.

为了避免后期面临更多的困境, 我们选择基于 canvas 的整体渲染的方案, 这是一个新颖具有挑战性的方案, 当然也有一些实现的例子 (Flipboard, <https://flipboard.com/explore>, 在移动端采用纯 canvas 来达到近乎原生应用的交互体验), 这里有对于这个方案较为简明地描述, <http://www.ruanyifeng.com/blog/2015/02/future-of-dom.html>

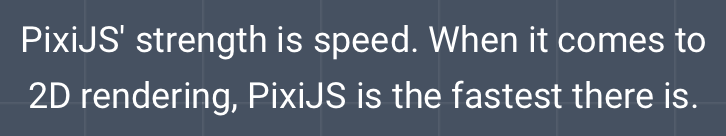
基于 canvas, 我们大致需要开发一套渲染的框架, 因此我们选择一个生态/性能较好的库作为底层. 为了考量性能, 我们选择了两个较为热门的 canvas库 以作比较: Fabric.js, PIXI.js

采用短时间绘制图形的数量作为性能的衡量, 考察了两个库在 Chrome, Safari, Firefox 和 Android 上的 Chrome 的表现. 测试连接在此:

https://shaojieliu.github.io/fabric\_vs\_pixi/

结果如下:

结果就是PIXI 全面优于 Fabric, 说明 PIXI 官网的豪言壮语还是有一定道理的:



附件一: 绘制多个基本图形用时测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **599** | **绘图性能 / 个/s** | **绘制多个基本图形(599圆圈) /ms** | | |
| **Chrome** | Native | 85571 | 11 | 6 | 4 |
| Fabric | 350 | 1736 | 1708 | 1694 |
| PIXI | 664 | 916 | 901 | 890 |
| **Safari** | Native | 138231 | 4 | 6 | 3 |
| Fabric | 186 | 3322 | 3191 | 3136 |
| PIXI | 872 | 709 | 685 | 667 |
| **Firefox** | Native | 81682 | 10 | 5 | 7 |
| Fabric | 66 | 9367 | 9203 | 8828 |
| PIXI | 915 | 708 | 632 | 624 |
| **Android** | Native | 23960 | 40 | 16 | 19 |
| Fabric | 58 | 10437 | 10353 | 10368 |
| PIXI | 130 | 4559 | 4616 | 4625 |