## Kity概要设计

1. 简介
2. 项目介绍

基于浏览器页面中运行的富图形编辑器，可以实现编辑图形，输入文字，图文混排等功能，实现文档编辑中的流程图，脑图等编辑功能。目前是作为UEditor的大型第三方插件为开发目的的，其产出可以落地百科。前端主要通过JavaScript+Html+Css实现，JavaScript的核心技术是SVG。

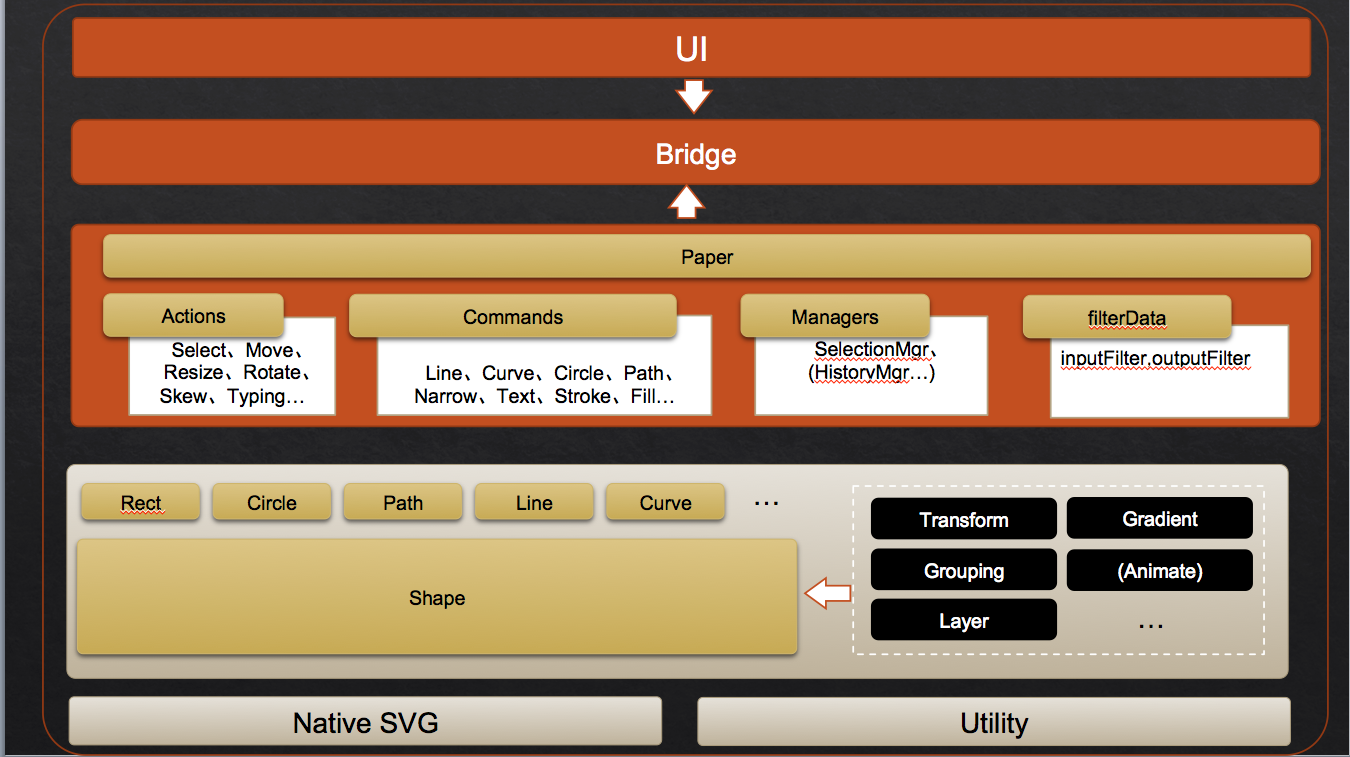
1. SVG/VML的选型

|  |  |
| --- | --- |
| Canvas | SVG/VML |
| 基于像素 | 基于对象模型 |
| 单个 HTML 元素 | 多个图形元素，是文档对象模型 (DOM) 的一部分 |
| 通过脚本以编程方式创建和修改视觉呈现 | 使用标记创建视觉呈现，并通过 CSS 或通过脚本以编程方式修改视觉呈现 |
| 事件模型/用户交互是粗粒度的—仅在 canvas 元素级别；必须通过鼠标坐标对交互进行手动编程设置 | 事件模型/用户交互是基于对象的，在最基本的图像元素级别进行——线条、矩形、路径 |
| API 不支持可访问性 | SVG/VML 标记和对象模型直接支持可访问性 |

主流的googleDoc,sohu也都是采用的 SVG技术

1. 设计说明
2. **架构图**

**整体架构采用结构化模式—分层模式**

****

1. **分层简介**

Kity主要分为4层：

* UI层： 表现层，即用户看到的整体界面，包括工具栏，提示框，菜单等。
* Bridge层：桥接层，提供UI层和编辑区域的交互绑定。只有该层，才能持有UI层和画布层的引用。
* Paper层：图形编辑业务层，整个编辑器的核心层。除了提供用户交互，还要为上层提供命令接口。
* Core层：对原生SVG接口的封装和类抽象，为上层画布提供统一的基于类的语意化接口。

1. **各层中主要模块**
   1. **UI**

参照UM的UI层设计

* 1. **Bridge**

参照UM的bridge层设计

* 1. **Paper**

**该层产生主类Paper，该类是对编辑区域的画布的抽象类，是将所有模块串联在一起的主干。它提供统一的事件注册和分发，还为各模块提供各种类型的接口，也是桥接层实现所需要持有的唯一引用。**

1. Action模块（用户直接操作画布的操作）
   * 解释：用户直接操作画布的各种动作，比如拖拽，点击某个图形等，我们都称其为一种交互的动作。
   * 类型：选择，放大缩小，移动，双击输入文字等
   * 每个动作模块是独立的，每个模块由检测机制和执行机制组成，画布对象提供统一的注册Action模块的接口。
2. 命令模块
   * 解释：为外部调用提供命令接口。比如用户点击工具栏上的插入矩形按钮，那就会在桥接层调用命令“insertRect”
   * 命令模块采用命令模式设计，由画布对象提供execCommand命令接口，通过传入不同的操作命令，对其画布产生不同的影响
   * 命令在执行期间，是独占的。Action模块的检测机制是被屏蔽的。
3. 数据模块
   * 针对写入编辑器的数据，编辑器的产出数据，粘贴进入编辑器的数据，进行过滤转换管理
4. undo/redo模块
5. 事件机制
   * 由Paper统一代理所有的事件，外部注册事件，无论是原生事件还是自定义事件都注册在Paper上，再由Paper统一分发。
   1. **core**
      1. **Shape是基类，负责产生一些通用的方法**
6. 整体流程

操作流程问题两类

触发工具栏流程



直接触发画布流程

