富文本编辑器的技术演进

罗龙浩

蚂蚁金服高级前端技术专家，语雀文档编辑器负责人

自我介绍

2008～2014：业余时间研发 KindEditor，经历 3 个版本的重写

2012～2014：土豆网前端架构师、前端负责人

2014～2018：支付宝行业前端负责人、口碑前端负责人

2018～至今：语雀文档编辑器负责人

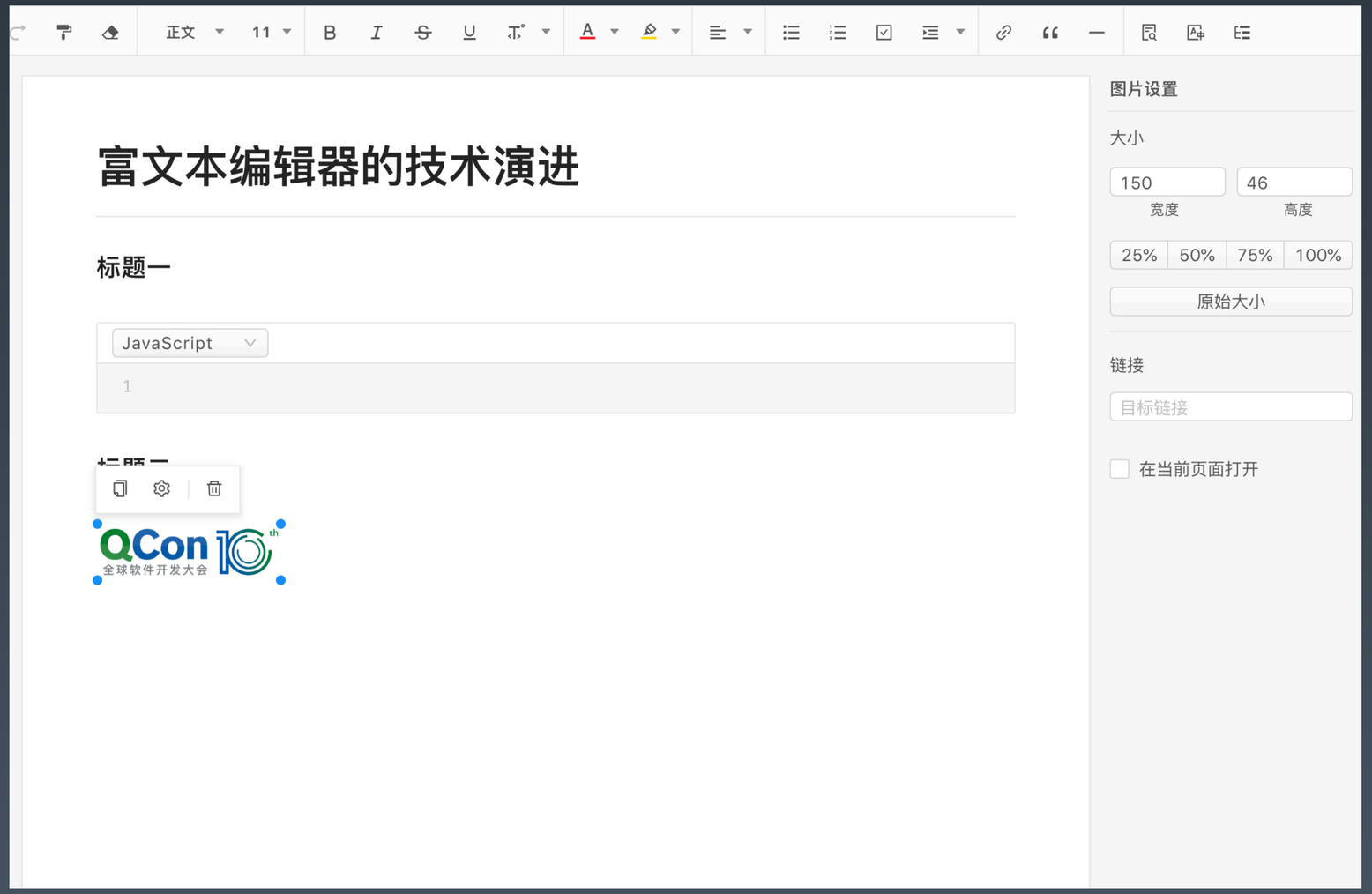
目录

一、富文本编辑器介绍

二、语雀文档编辑器面临的问题与解决思路

三、多人实时协同的解决思路

富文本编辑器 - 常见交互



富文本输入框

- 输入内容
- 选中 & 操作

操作栏

- 顶部工具栏
- 侧边栏
- 内嵌工具栏

富文本编辑器 - 浏览器特性

富文本输入框

```
<div contenteditable=“true” >这里可以编辑</div>
```

对内容进行操作

```
document.execCommand( ‘bold’ );
```

富文本编辑器 - 技术类型

类型	描述	典型产品
L0	1、基于 contenteditable 2、使用 document.execCommand 3、几千～几万行代码	早期的轻量级编辑器
L1	1、基于 contenteditable 2、不用 document.execCommand，自主实现 3、几万行～几十万行代码	CKEditor、TinyMCE Draft.js、Slate 石墨文档、腾讯文档
L2	1、不用 contenteditable，自主实现 2、不用 document.execCommand，自主实现 3、几十万行～几百万行代码	Google Docs Office Word Online iCloud Pages WPS 文字在线版

富文本编辑器 - 不同类型的优劣

类型	优势	劣势
L0	技术门槛低，短时间内快速研发	可定制的空间非常有限
L1	站在浏览器肩膀上，能够满足 99% 业务场景	无法突破浏览器本身的排版效果
L2	技术都掌控在自己手中，支持个性化排版	技术难度相当于自研浏览器、数据库

富文本编辑器 - L1 编辑器

传统模式

DOM 树等于数据，调用各种 DOM API 进行操作

典型产品：CKEditor 4、TinyMCE、UEditor

MVC 模式

数据和渲染分离，实现一套操作数据模型的方法，数据变更带动渲染

典型产品：CKEditor 5、Draft.js、Slate

富文本编辑器 - L1 编辑器两种模式优劣

传统模式

优势：20 年的历史，代码简单直接，可维护性好，充分利用 contenteditable 特性

劣势：代码写法不符合潮流，都是 10 几年前的技术

MVC 模式

优势：代码写法符合潮流

劣势：引起数据和渲染不同步的问题，因为这个机制需要有完全控制用户输入的前提，实际上基于 contenteditable 没办法控制用户的所有输入，第三方输入法、壳浏览器会让用户输入不可控

富文本编辑器 - L2 编辑器

自主实现富文本输入框，包含用户输入和排版引擎，可用 DOM、SVG 技术

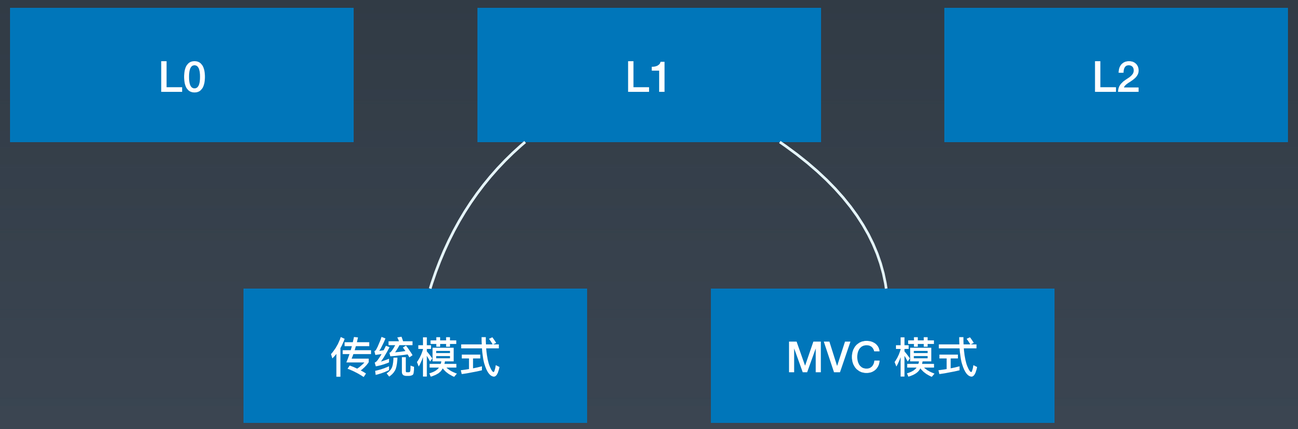
用户输入：

光标、选区自主实现，光标位置放隐藏 textarea 接受键盘输入，输入完成之后变更数据、渲染视图

排版引擎：

实现各种个性化的文字排版、图文布局，突破浏览器排版限制

富文本编辑器 - 总结



技术类型

如何技术选型？

没有编辑器研发团队：推荐基于 CKEditor 4、TinyMCE 二次开发

有几人编辑器研发团队：推荐自研 L1 传统模式编辑器

有几十人编辑器研发团队 & 需要个性化排版：推荐自研 L2 编辑器

目录

一、富文本编辑器介绍

二、语雀文档编辑器面临的问题与解决思路

三、多人实时协同的解决思路

语雀编辑器 - 面临的问题

疑难杂症多

问题难以修复，页面崩溃、光标错乱、粘贴卡死等

排查链路长

语雀编辑器、Slate、React 一层层往下查

新增功能难

很多个性化需求，在 Slate 架构上实现成本较高

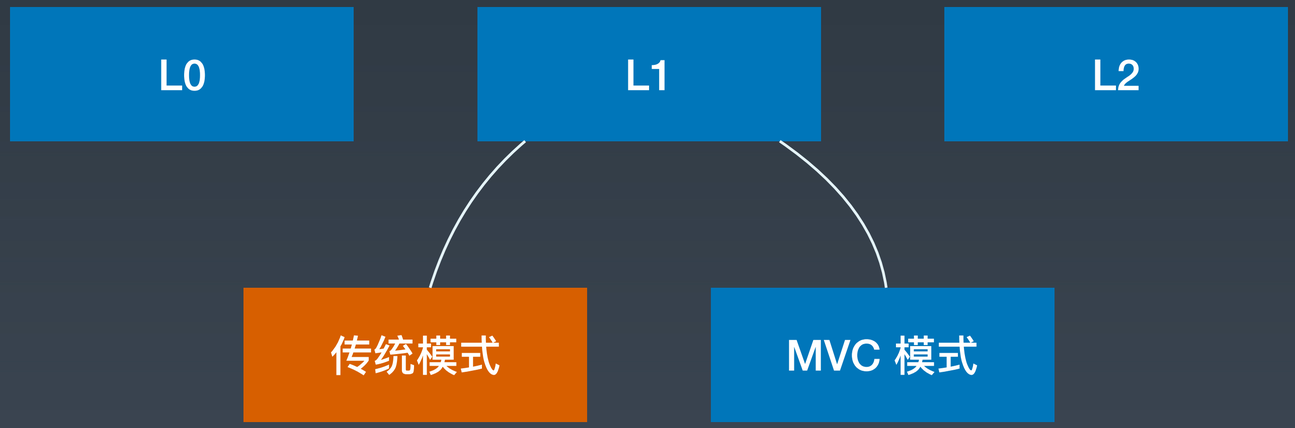
语雀编辑器 - 根本问题

技术选型问题

1）基于 Slate，是 L1 MVC 模式

2）基于 React 渲染，但 React 是 UI 构建库

语雀编辑器 - 技术选型



更换技术选型，用 L1 传统模式重写编辑器

为什么没有基于开源编辑器？

第一是 license 问题，第二是我正好具备多年 L1 编辑器研发经验 ：）

语雀编辑器 - 技术目标

高健壮性

采用一切手段保证功能的稳定，努力做到业内问题最少的编辑器

可维护性

编辑器本身代码量很大，后期可维护性是关键，能用简单方式解决问题，尽量简化

可扩展性

具备良好的扩展性，不能因为架构问题，满足不了业务需求

语雀编辑器 - 开发思路

数据格式：在 HTML 基础上扩展

卡片机制：承接组件的扩展，在编辑器里独立的一块区域

开发模式：Hybrid 混合开发，编辑区域用原生 JS，UI 层用 React

技术原理：基于 contenteditable，通过 Range API 对选中的内容进行操作

语雀编辑器 - 数据格式

```
<h3>heading</h3>
<p>
  <strong><anchor />bold<focus /></strong>
  <em>italic</em>
  <u>underline</u>
  <span style="color: #FFFFFF;">fontcolor</span>
  <span style="background-color: #000000;">backcolor</span>
  <card type="inline" name="image" value="JSON string"></card>
</p>
<p style="text-align: center;">alignment</p>
<ol>
  <li>orderedlist</li>
</ol>
<card type=“inline" name="file" value="JSON string"></card>
<card type="block" name="codeblock" value="JSON string"></card>
```

光标

<cursor />

选区

<anchor />HTML<focus />

卡片组件

<card type= “”

name= “” ></card>

语雀编辑器 - 卡片机制


卡片工具栏
卡片内容区域（contenteditable = false）
Left Cursor
Right Cursor

语雀编辑器 - 卡片类型

Inline Card

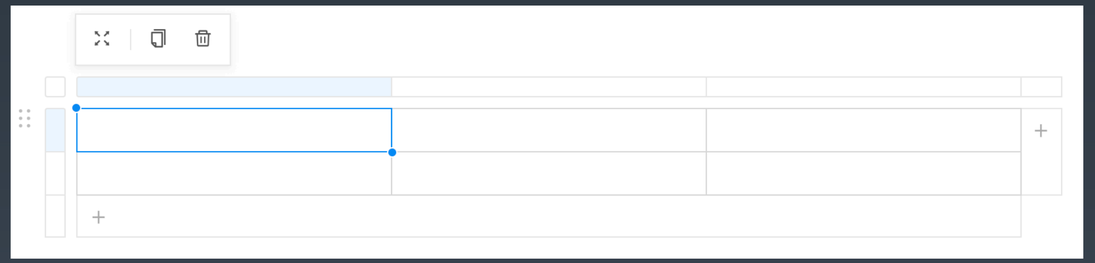
左侧文字  右侧文字

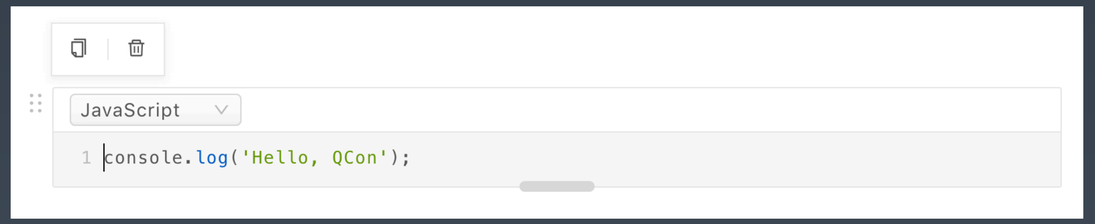
左侧文字 《前端路上的思考》- 圆心.pdf (34.9 MB) 右侧文字

左侧文字 开发中 右侧文字

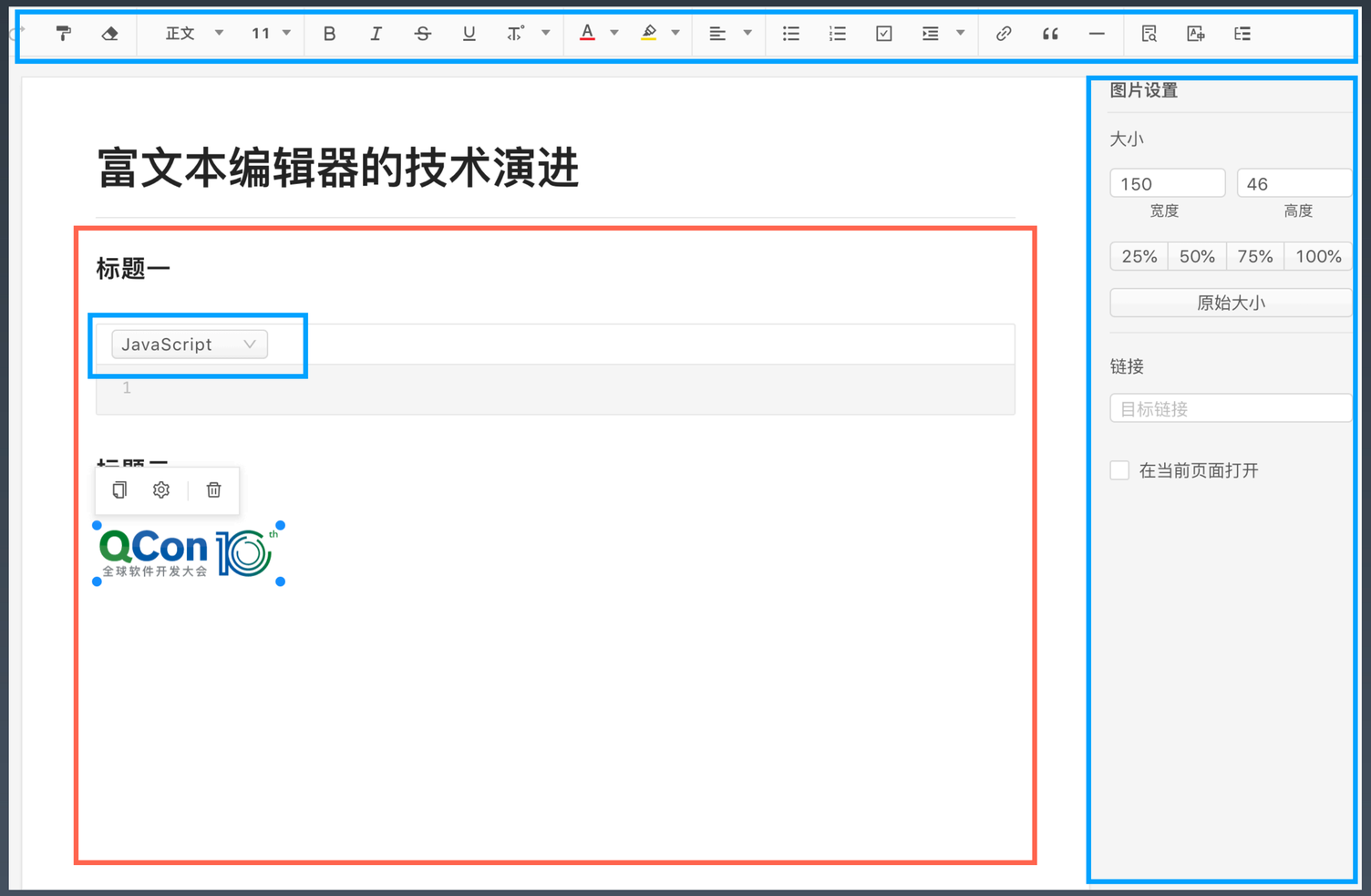
左侧文字 👍 右侧文字

Block Card





语雀编辑器 - 混合开发模式



红色区域：原生 JS

蓝色区域：React

语雀编辑器 - 为什么用混合开发模式?

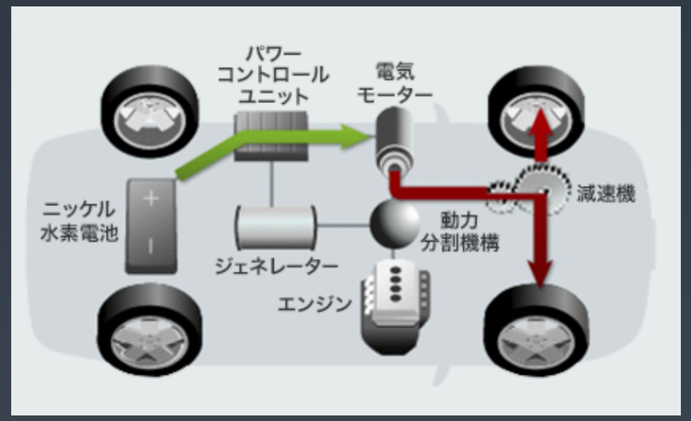
统一不一定是最佳选择，还是要看带来的业务价值

有两个成功案例：

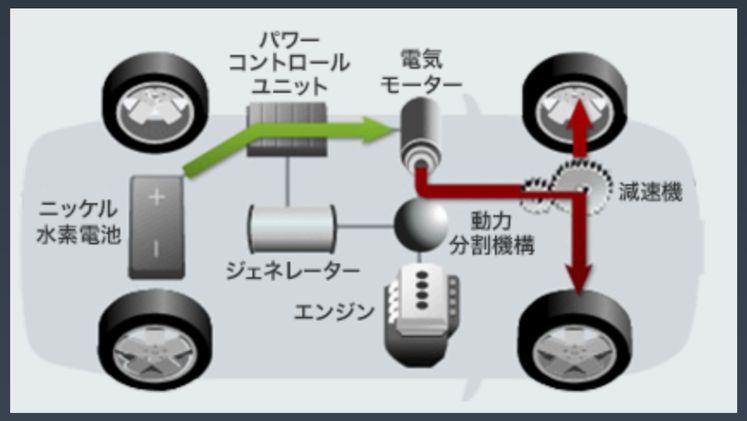
1）移动端 Hybrid 开发（Native + H5）

2）丰田、本田的 Hybrid 汽车（电机 + 内燃机）

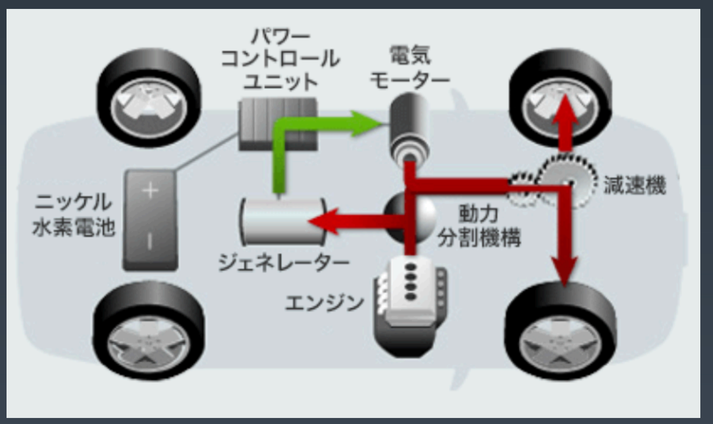
语雀编辑器 - 丰田混动系统



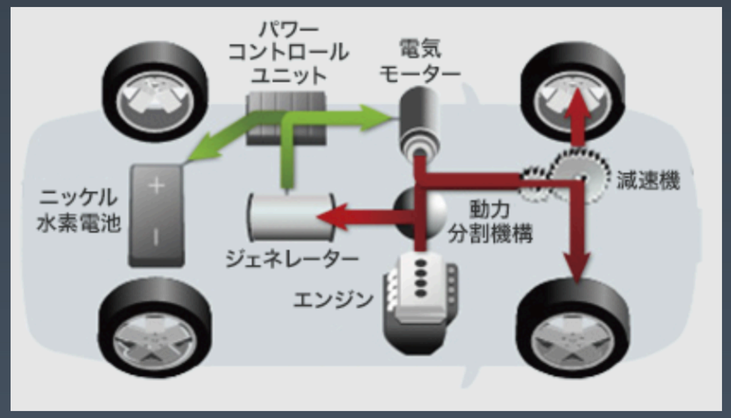
起步



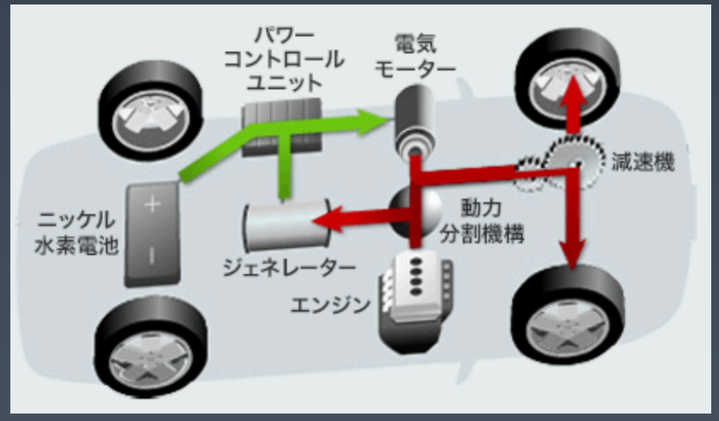
低速行驶



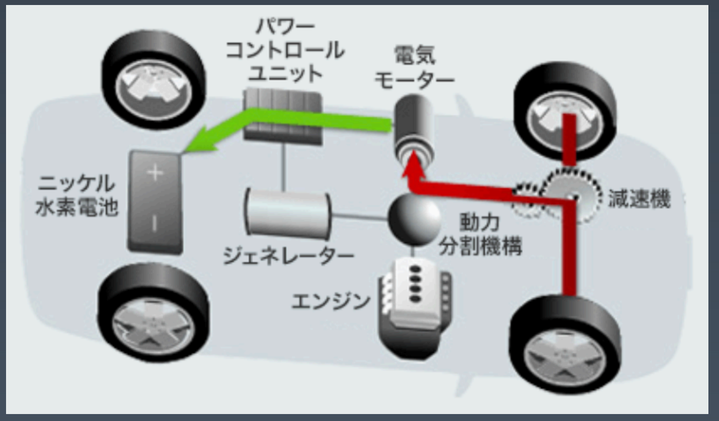
正常行驶，无剩余能量



正常行驶，有剩余能量

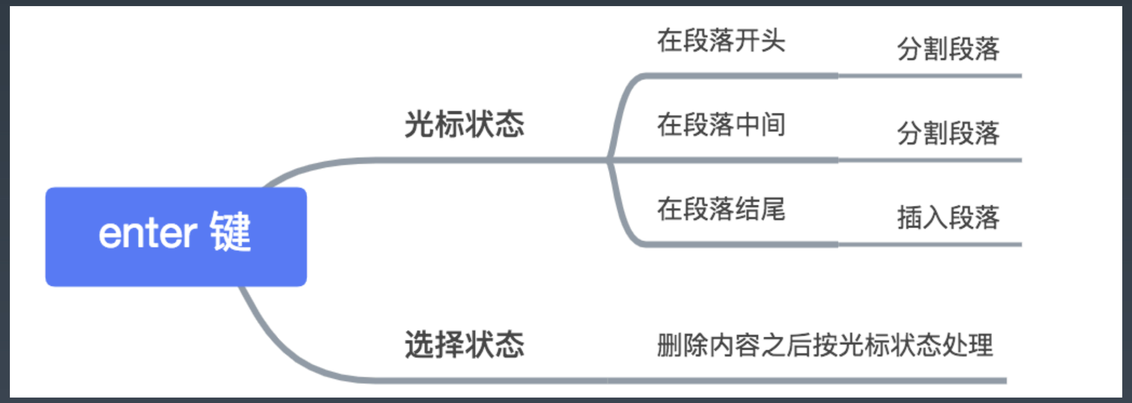


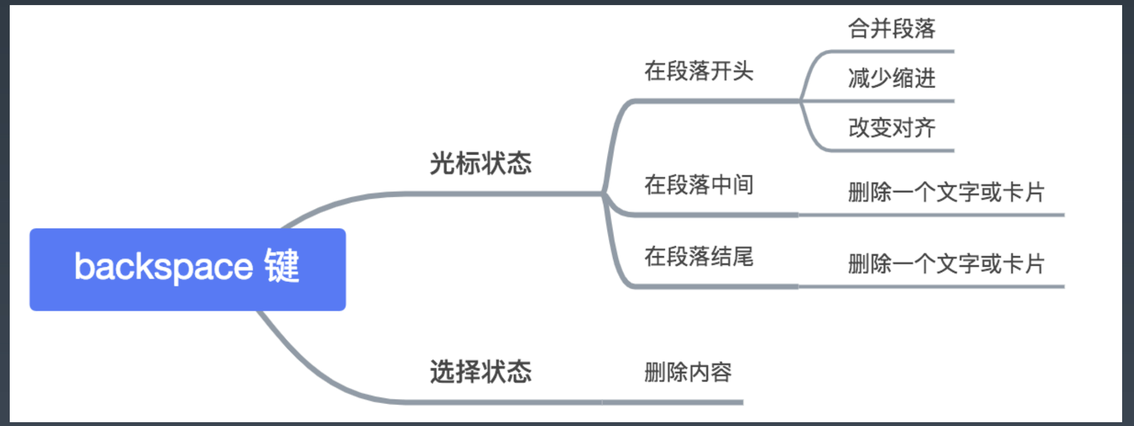
急加速

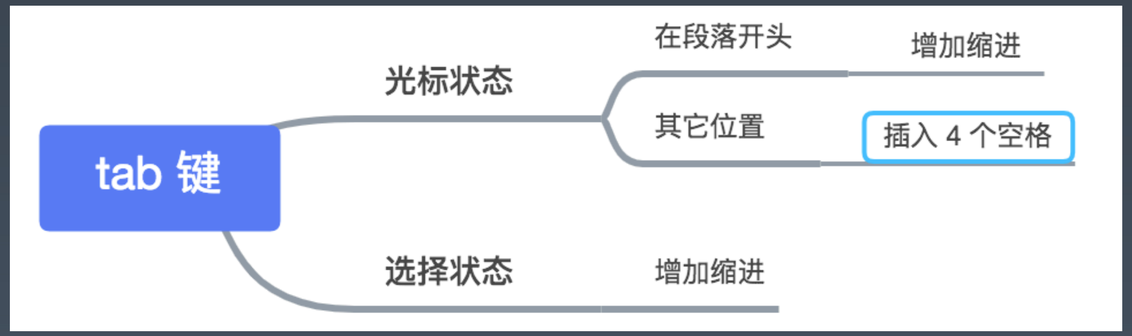


减速，充电

语雀编辑器 - 键盘输入定制







语雀编辑器 - contenteditable 问题

光标无法移动到空标签里： | 、|

光标漂移到 inline-block 右侧： |

光标无法精确控制： link||

无法输入中文： emoji|

语雀编辑器 - contenteditable 解决方案

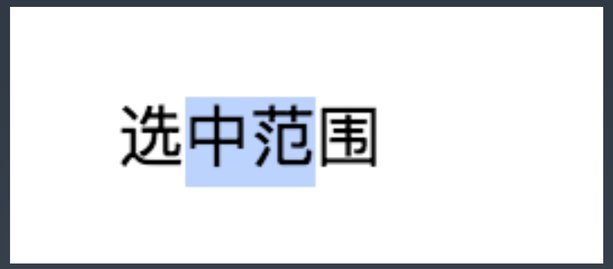
光标无法移动到空标签里：
| 、|

光标漂移到 inline-block 右侧： |

光标无法精确控制： link|

无法输入中文： emoji|

语雀编辑器 - Range 介绍



1、开始位置和结束位置通过 container 和 offset 标记位置

2、在文本之间：container 为 TextNode，offset 为从第一个字符到当前位置的偏移量（第几个字符）

3、在节点之间：container 为父节点，offset 为从第一个子节点到当前位置的偏移量（第几个子节点）

4、开始位置等于结束位置， range.collapsed 为 true，也就是光标状态

5、开始位置不等于结束位置，range.collapsed 为 false，也就是选择一段内容的状态

语雀编辑器 - Range 示例

```
<p>a<cursor />bc</p>

range.startContainer = abc;
range.startOffset = 1;
range.endContainer = abc;
range.endOffset = 1;
range.collapsed = true;
```

```
<p><cursor /></p>

range.startContainer = p;
range.startOffset = 0;
range.endContainer = p;
range.endOffset = 0;
range.collapsed = true;
```

```
<p><anchor />abc<focus /></p>

range.startContainer = abc;
range.startOffset = 0;
range.endContainer = abc;
range.endOffset = 3;
range.collapsed = fasle;
```

```
<p><anchor />abc<focus /></p>

range.startContainer = p;
range.startOffset = 0;
range.endContainer = p;
range.endOffset = 1;
range.collapsed = false;
```

语雀编辑器 - 性能对比

	语雀文档	Google Docs	腾讯文档	石墨文档
加载时间	2 秒	3 秒	3 秒	8 秒
粘贴时间	4 秒	7 秒	6 秒	14 秒
操作响应	有点卡	顺畅	比较卡	比较卡

测试设备：2015 款 MacBook Pro 15，Chrome 77.0.3865
测试数据：https://shimo.im/docs/keW3LxVd2vQHxUHD/read

声明：由于每个产品的定位、功能复杂度有差异，测试结果好，不代表编辑器整体领先，只能说明某一方面有优势。

语雀编辑器 - 时间节点

2018.08：技术选型，开始研发

2018.09：基础编辑 demo 演示

2018.11：讨论区、评论小型编辑器上线

2019.01：文档编辑器上线

2019.03：旧版编辑器全部替换完成，整体运行平稳

语雀编辑器 - 总结

一、根据当前主要问题和后续产品方向，选择合适的技术方案

二、对于绝大多数业务，L1 传统模式编辑器是合适的选择

三、利用好现有的资源，可以用 React、Vue 成熟的组件搭建外围的 UI 层

目录

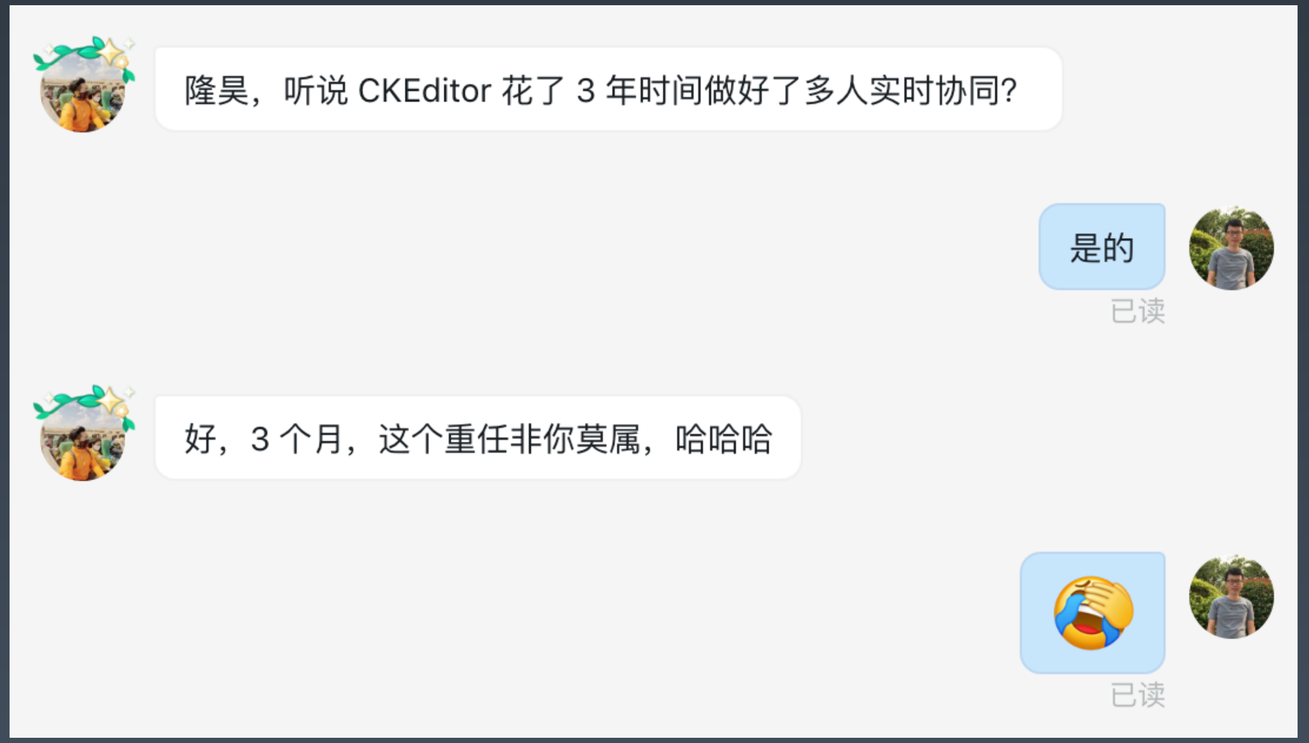
一、富文本编辑器介绍

二、语雀文档编辑器面临的问题与解决思路

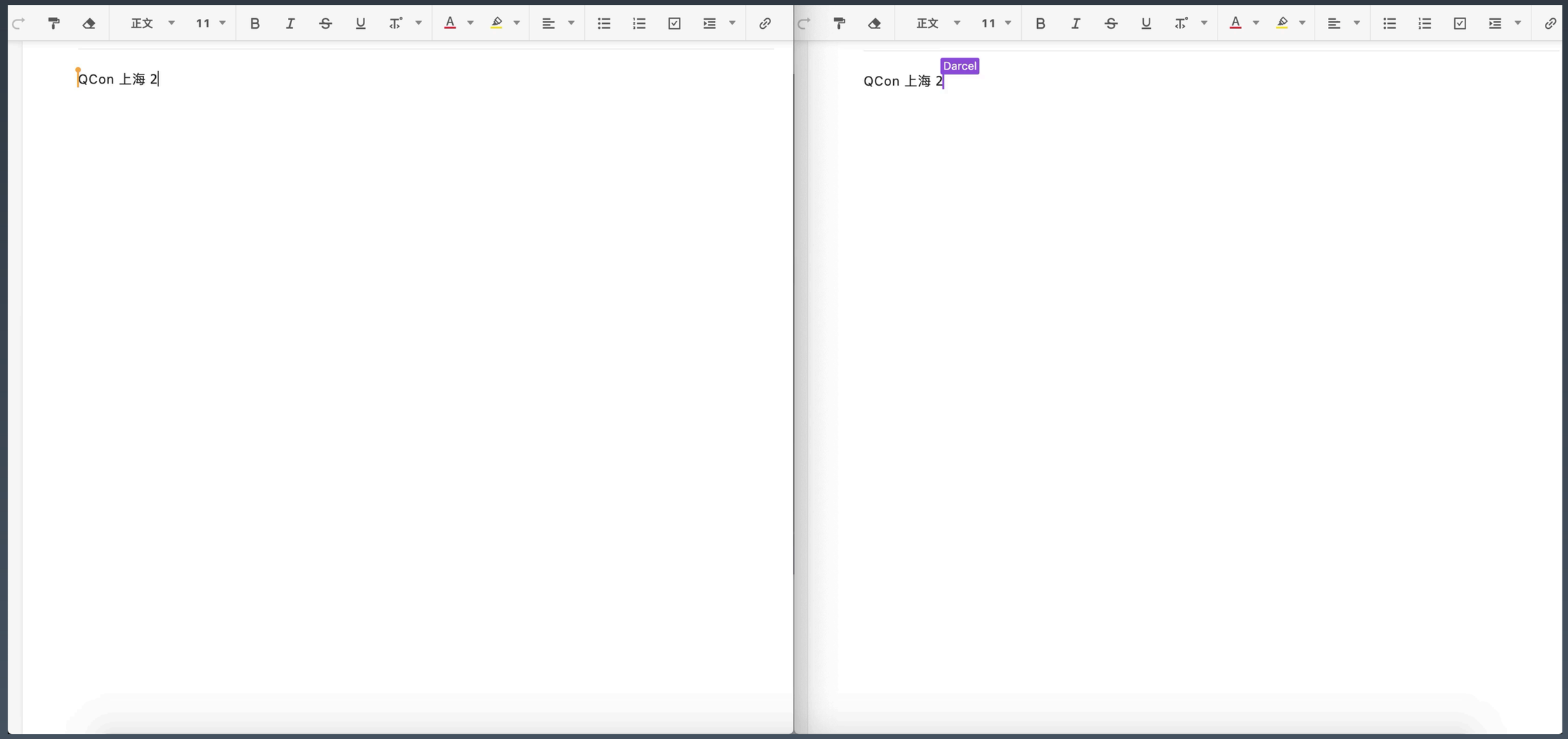
三、多人实时协同的解决思路

多人实时协同 - 新的挑战

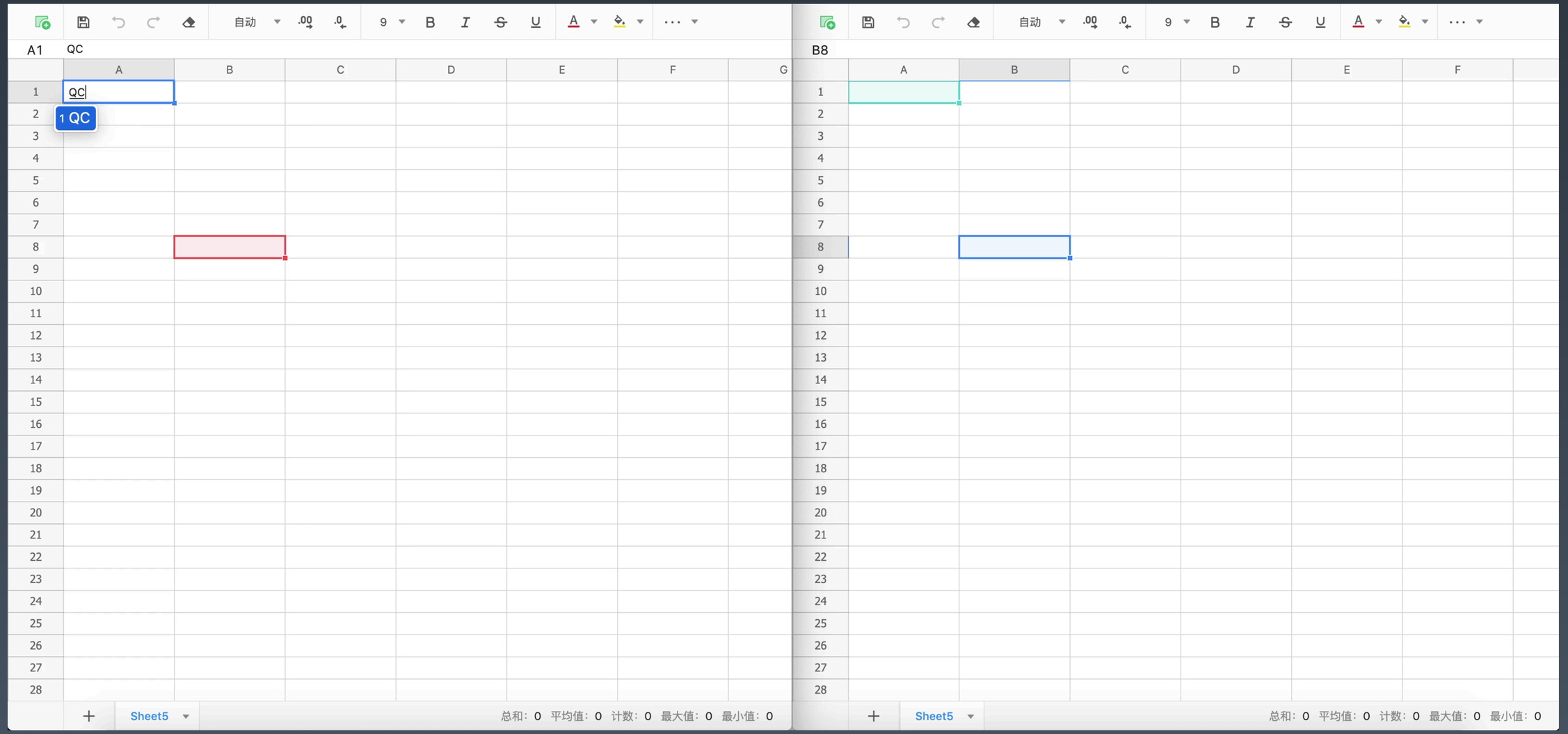
今年 3 月份，我们 PD 找我说



多人实时协同 - 语雀文档



多人实时协同 - 语雀表格



多人实时协同 - 分析竞品

调研对象：Google Docs、Etherpad、 CKEditor 5、Slate、Quill

结论：都用 OT（Operational Transformation）或类似的技术，将操作转化成 OP（operations），发送到协作服务，再转发给其它在线用户。所以都具备原子化的操作 API，所有的高级操作都通过原子化 API 完成，实时协同只需要将这些原子化 API 的调用信息转化成 OP 即可

多人实时协同 - 开源编辑器的原子化 API

Quill： insert、delete、retain、format

Slate： insert_text、remove_text、insert_node、merge_node、remove_node、move_node、set_node、split_node

CKEditor 5： insert、move、detach、merge、split、attribute

多人实时协同 - 想法一

改成 MVC 模式

引入 DataModel、抽象原子化 API，但这个意味着重新开发一套编辑器，工作量巨大，很可能重回 Slate 老路，丢失我们自己的优势，稳定性、易维护、粘贴性能等

多人实时协同 - 想法二

封装原子化 API

能不能封装 insertNode、removeNode、mergeNode、splitNode 等原子化 API，所有上层操作都基于这些 API，是否可行？

但几乎所有代码都要修改，影响面完全不可控

多人实时协同 - 想法三

DOM diff 方案

能不能每次操作之后直接对比变更前后的 2 个 DOM 树，生成 JSON 格式的 diff，是否可行？

最大问题是性能，虽然能通过局部 diff 提升性能，但每次操作都要 diff 有点夸张。

多人实时协同 - 想法四

全量 command 机制

引入新的 command 机制，所有的变更都通过 command 完成，变更之后产生对应的 OP，包含 backward 逆向操作

看起来可行

开始 demo 开发，发现编写 command 是非常复杂的事情，写 backward 逻辑成本太高

多人实时协同 - 想法五

在 DOM 底层实现

我们的目标是增加实时协同能力，功能的稳定性比较重要，现在的优势不能丢失，日常迭代和新功能开发还是要持续进行。所以只能在现有代码上进行改进和扩展，不能推翻重来

只有一条路，在 DOM 底层做文章

其实 DOM 树相当于 DataModel，DOM API 相当于原子化 API

多人实时协同 - 生成 OP

通过浏览器的 MutationObserver API，获取 DOM 树的变更信息

```
// Select the node that will be observed for mutations
const targetNode = document.getElementById('some-id');

// Options for the observer (which mutations to observe)
const config = { attributes: true, childList: true, subtree: true };

// Callback function to execute when mutations are observed
const callback = function(mutationsList, observer) {
    for(let mutation of mutationsList) {
        if (mutation.type === 'childList') {
            console.log('A child node has been added or removed.');
        }
        else if (mutation.type === 'attributes') {
            console.log('The ' + mutation.attributeName + ' attribute was modified.');
        }
    }
};

// Create an observer instance linked to the callback function
const observer = new MutationObserver(callback);

// Start observing the target node for configured mutations
observer.observe(targetNode, config);

// Later, you can stop observing
observer.disconnect();
```

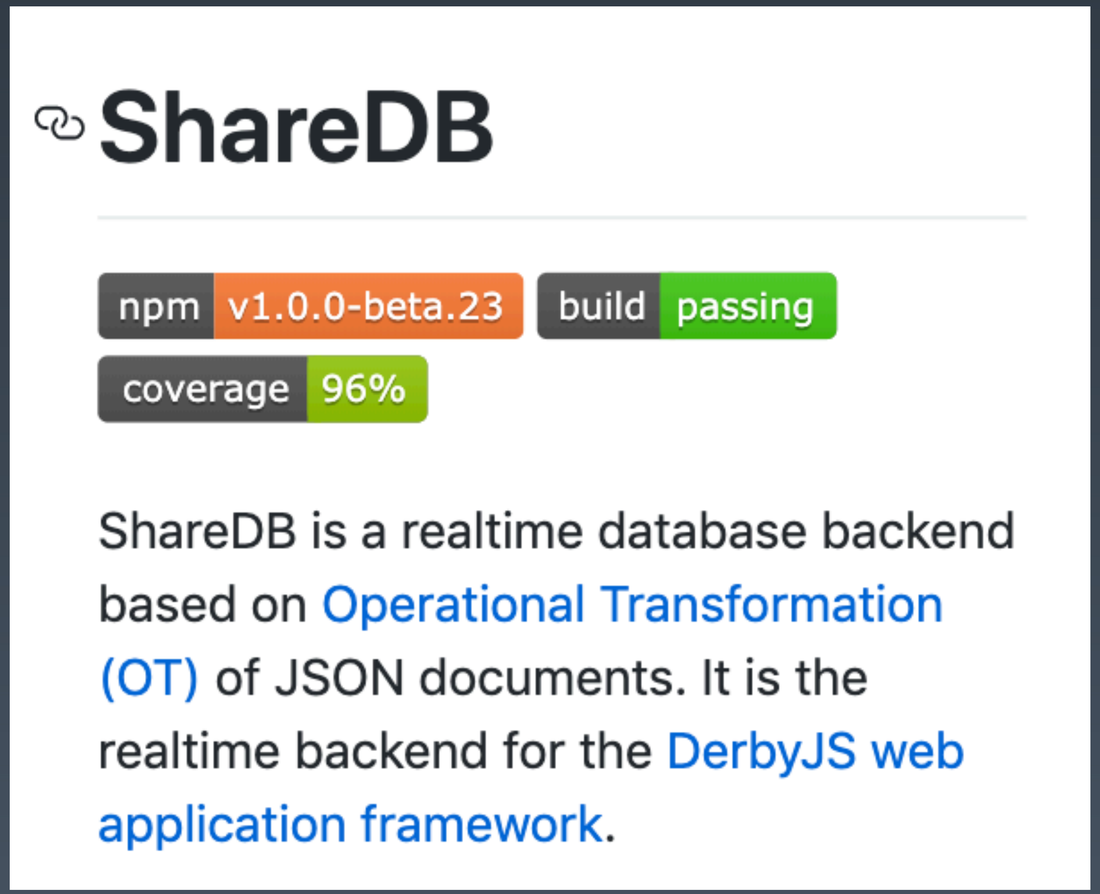
Example

Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Firefox for Android	Opera for Android	Safari on iOS
26	Yes	14	11	15	7	Yes	26	14	14	7
26	Yes	14	11	15	7	Yes	26	14	14	7
18	12	14	11	15	6	Yes	18	14	14	6
18	12	14	11	15	6	Yes	18	14	14	6
18	12	14	11	15	6	Yes	18	14	14	6

Compatibility

多人实时协同 - OT 服务

采用 ShareDB，实现了 OT 算法，提供一个基于 JSON 的 OT 通用能力




```
var sharedb = require('sharedb/lib/client');

// Open WebSocket connection to ShareDB server
var socket = new WebSocket('ws://' + window.location.host);
var connection = new sharedb.Connection(socket);

// Create local Doc instance mapped to 'examples' collection document with id 'counter'
var doc = connection.get('examples', 'counter');

// Get initial value of document and subscribe to changes
doc.subscribe(showNumbers);
// When document changes (by this client or any other, or the server),
// update the number on the page
doc.on('op', showNumbers);

function showNumbers() {
  document.querySelector('#num-clicks').textContent = doc.data.numClicks;
};

// When clicking on the '+1' button, change the number in the local
// document and sync the change to the server and other connected
// clients
function increment() {
  // Increment `doc.data.numClicks`. See
  // https://github.com/ottypes/json0 for list of valid operations.
  doc.submitOp([{p: ['numClicks'], na: 1}]);
}

// Expose to index.html
global.increment = increment;
```

多人实时协同 - 解决方案

OT 服务：基于 ShareDB

数据格式：JSONML

技术原理：通过 MutationObserver API 监听编辑器的 DOM 树变更，生成 JSON 格式的 OP，发送到 ShareDB，更新 JSONML 数据。同时将 OP 发送到其它用户，将 OP 转化成 DOM 操作方法之后执行

多人实时协同 - OP 格式

OP 格式	JSON	DOM
{p:PATH, li:NEWVALUE}	List Insert	插入 Node
{p:PATH, ld:OLDVALUE}	List Delete	删除 Node
{p:PATH, oi:NEWVALUE}	Object Insert	增加 Element 属性
{p:PATH, od:OLDVALUE}	Object Delete	删除 Element 属性
{p:PATH, si:TEXT}	String Insert	插入 Text
{p:PATH, sd:TEXT}	String Delete	删除 Text

多人实时协同 - 时间节点

2019.04：技术选型，开始研发

2019.08：文档编辑器的多人协同上线

2019.10：表格编辑器的多人协同上线（计划）

多人实时协同 - 总结

一、L1 传统模式编辑器也可以实现多人实时协同

二、如果其它业务中需要多人实时协同的场景，推荐 ShareDB

三、仅仅完成功能，其实不难，是从 0 到 1 的过程

四、要做成完美，非常难，是从 1 到 100 的过程

Geekbang 极客邦科技 | InfoQ 极客传媒

InfoQ官网 全新改版上线

促进软件开发领域知识与创新的传播

关注InfoQ网站
第一时间浏览原创IT新闻资讯

免费下载迷你书
阅读一线开发者的技术干货

THANKS!

QCon 10th