

微信原生开发与 uni-app 比较



本文用于记录微信小程序中原生开发与 uni-app 比较过程，文中内容来自社区精读整理。

[示例源码](#)

一、性能对比

结合公司业务场景，本次测评说明如下。

测评对象：

- 微信原生开发
- uni-app

测评场景：

- 长列表加载
- 组件批量更新

避免额外影响：

- 服务端接口响应时长
- 代码质量引起的性能问题

1.1 测评示例

为真实反映原生与 uni-app 性能差异，这里沿用社区提供的测评方式。

测试模型（仿微博小程序）：



测试用例：

- 自动加载 10 页共 200 条数据，查看渲染平均耗时
- 自动点击点赞按钮 20 次，查看渲染平均耗时

1.2 测评原理

测试涉及的关键点：

- **Performance** (用于性能统计)
- 复写微信小程序 Page、Component、setData 方法

由于 setData 中 callback 参数表示渲染完成，统计方式为：

- 渲染开始：调用 setData 之前

- 渲染结束：setData 调用结束的回调

伪代码如下：

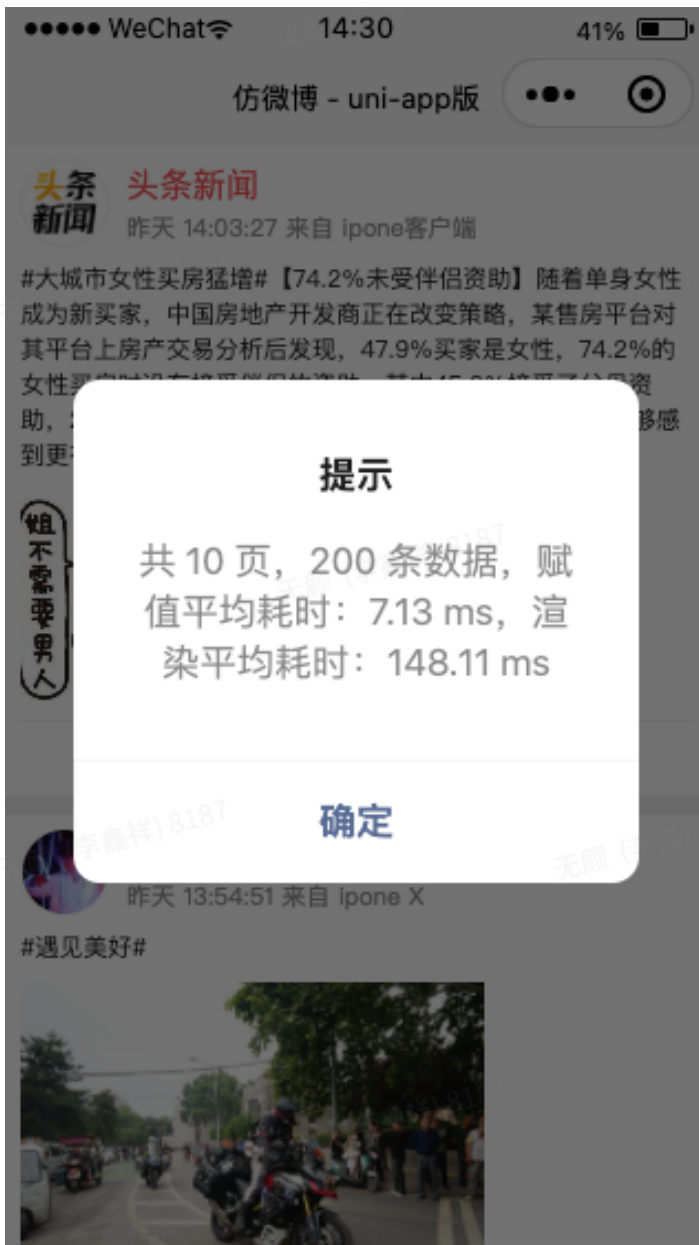
JavaScript

```
1 function wrapperSetData(namespace, shouldMeasureFn, contentFn, getAutoFn, oldSet
  Data) {
2   oldSetData = oldSetData || this.setData;
3
4   this.setData = function setData(data, callback) {
5     // 通过 performance api 标记 渲染开始
6     this.$perf.measure('setData.before');
7
8     oldSetData.call(this, data, function () {
9       // 通过 performance api 标记 渲染结束
10      this.$perf.measure('setData.end');
11
12      callback && callback();
13    });
14  };
15 }
```

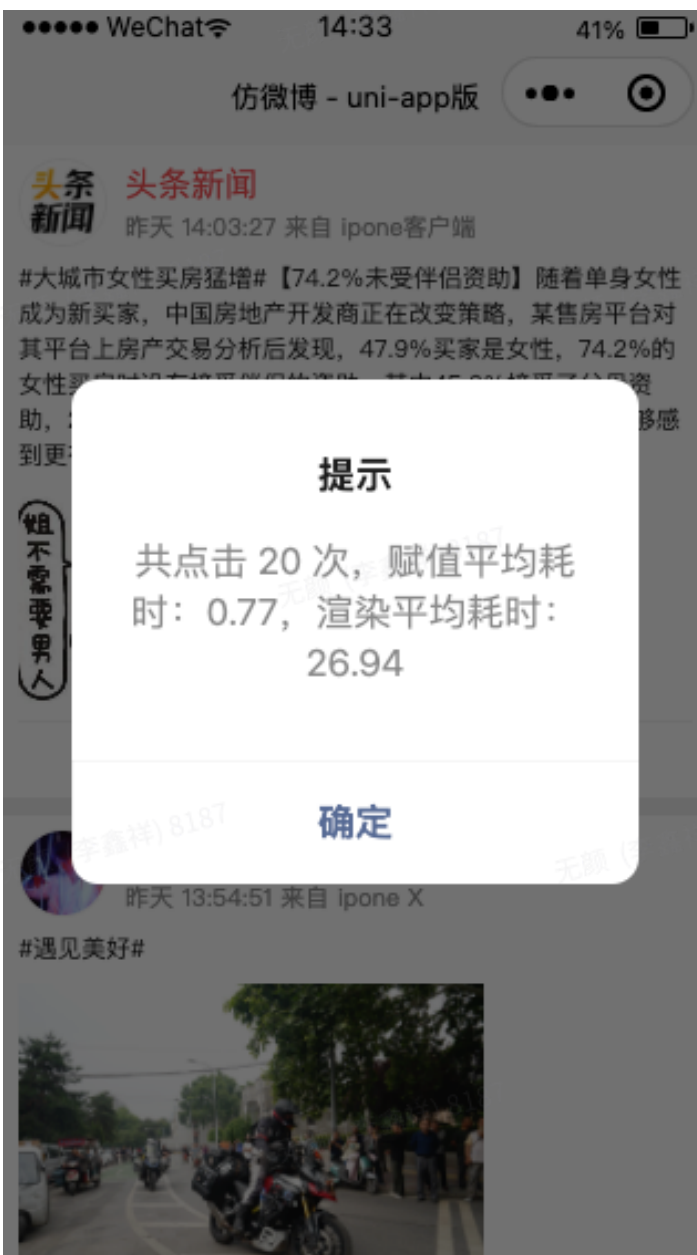
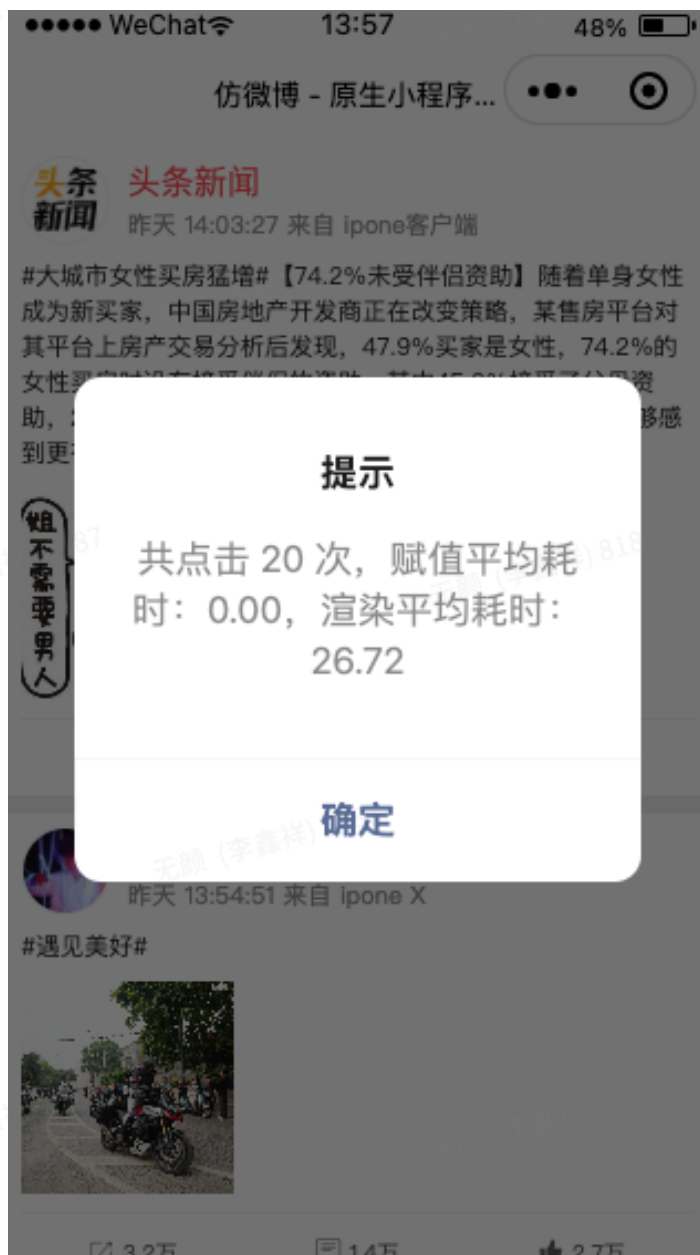
1.3 测评结果

测评结果分为原生开发、uni-app 开发。

渲染列表对比结果：



点击对比结果:



原生开发测试详情：

测试详情：	
第 1 次，	赋值耗时：2，渲染耗时：243
第 2 次，	赋值耗时：1，渲染耗时：102
第 3 次，	赋值耗时：1，渲染耗时：123
第 4 次，	赋值耗时：0，渲染耗时：117
第 5 次，	赋值耗时：1，渲染耗时：121
第 6 次，	赋值耗时：0，渲染耗时：71
第 7 次，	赋值耗时：1，渲染耗时：133
第 8 次，	赋值耗时：0，渲染耗时：153
第 9 次，	赋值耗时：1，渲染耗时：102
第 10 次，	赋值耗时：1，渲染耗时：104
共 10 页，200 条数据，赋值平均耗时：1.00 ms，渲染平均耗时：119.38 ms	

测试详情：	
第 1 次，	赋值耗时：1，渲染耗时：34
第 2 次，	赋值耗时：0，渲染耗时：31
第 3 次，	赋值耗时：0，渲染耗时：24
第 4 次，	赋值耗时：0，渲染耗时：27
第 5 次，	赋值耗时：0，渲染耗时：31
第 6 次，	赋值耗时：0，渲染耗时：21
第 7 次，	赋值耗时：0，渲染耗时：24
第 8 次，	赋值耗时：0，渲染耗时：25
第 9 次，	赋值耗时：0，渲染耗时：24
第 10 次，	赋值耗时：0，渲染耗时：22
第 11 次，	赋值耗时：0，渲染耗时：24
第 12 次，	赋值耗时：0，渲染耗时：24
第 13 次，	赋值耗时：0，渲染耗时：23
第 14 次，	赋值耗时：1，渲染耗时：30
第 15 次，	赋值耗时：0，渲染耗时：41
第 16 次，	赋值耗时：0，渲染耗时：33
第 17 次，	赋值耗时：0，渲染耗时：37
第 18 次，	赋值耗时：0，渲染耗时：24
第 19 次，	赋值耗时：0，渲染耗时：23
第 20 次，	赋值耗时：0，渲染耗时：19
共点击 20 次，赋值平均耗时：0.00，渲染平均耗时：26.72	

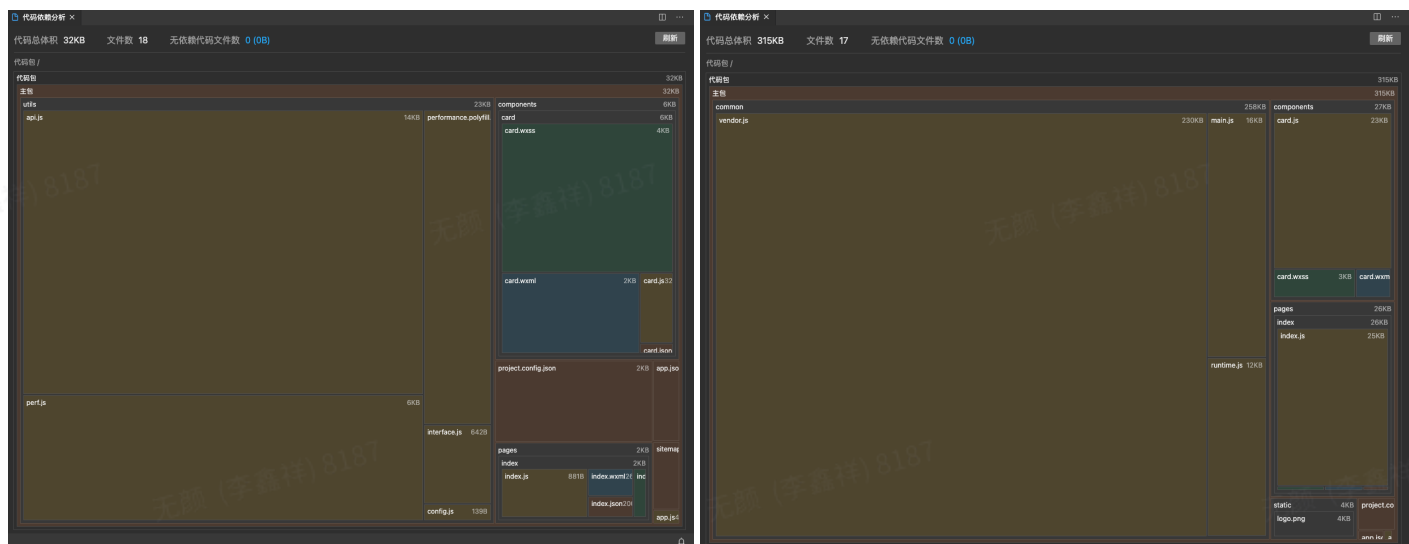
uni-app 测试详情：

测试详:	测试详情:
第 1 次, 赋值耗时: 4 ms, 渲染耗时: 133 ms	第 1 次, 赋值耗时: 3 ms, 渲染耗时: 33 ms
第 2 次, 赋值耗时: 2 ms, 渲染耗时: 138 ms	第 2 次, 赋值耗时: 2 ms, 渲染耗时: 23 ms
第 3 次, 赋值耗时: 6 ms, 渲染耗时: 151 ms	第 3 次, 赋值耗时: 0 ms, 渲染耗时: 22 ms
第 4 次, 赋值耗时: 7 ms, 渲染耗时: 168 ms	第 4 次, 赋值耗时: 1 ms, 渲染耗时: 25 ms
第 5 次, 赋值耗时: 5 ms, 渲染耗时: 149 ms	第 5 次, 赋值耗时: 1 ms, 渲染耗时: 29 ms
第 6 次, 赋值耗时: 6 ms, 渲染耗时: 124 ms	第 6 次, 赋值耗时: 0 ms, 渲染耗时: 16 ms
第 7 次, 赋值耗时: 12 ms, 渲染耗时: 178 ms	第 7 次, 赋值耗时: 0 ms, 渲染耗时: 28 ms
第 8 次, 赋值耗时: 15 ms, 渲染耗时: 135 ms	第 8 次, 赋值耗时: 1 ms, 渲染耗时: 42 ms
第 9 次, 赋值耗时: 7 ms, 渲染耗时: 134 ms	第 9 次, 赋值耗时: 1 ms, 渲染耗时: 31 ms
第 10 次, 赋值耗时: 10 ms, 渲染耗时: 147 ms	第 10 次, 赋值耗时: 1 ms, 渲染耗时: 29 ms
共 10 页, 200 条数据, 赋值平均耗时: 7.13 ms, 渲染平均耗时: 148.11 ms	第 11 次, 赋值耗时: 1 ms, 渲染耗时: 37 ms
	第 12 次, 赋值耗时: 0 ms, 渲染耗时: 30 ms
	第 13 次, 赋值耗时: 0 ms, 渲染耗时: 24 ms
	第 14 次, 赋值耗时: 0 ms, 渲染耗时: 22 ms
	第 15 次, 赋值耗时: 0 ms, 渲染耗时: 24 ms
	第 16 次, 赋值耗时: 0 ms, 渲染耗时: 28 ms
	第 17 次, 赋值耗时: 0 ms, 渲染耗时: 18 ms
	第 18 次, 赋值耗时: 1 ms, 渲染耗时: 41 ms
	第 19 次, 赋值耗时: 1 ms, 渲染耗时: 23 ms
	第 20 次, 赋值耗时: 0 ms, 渲染耗时: 18 ms
	共点击 20 次, 赋值平均耗时: 0.77, 渲染平均耗时: 26.94



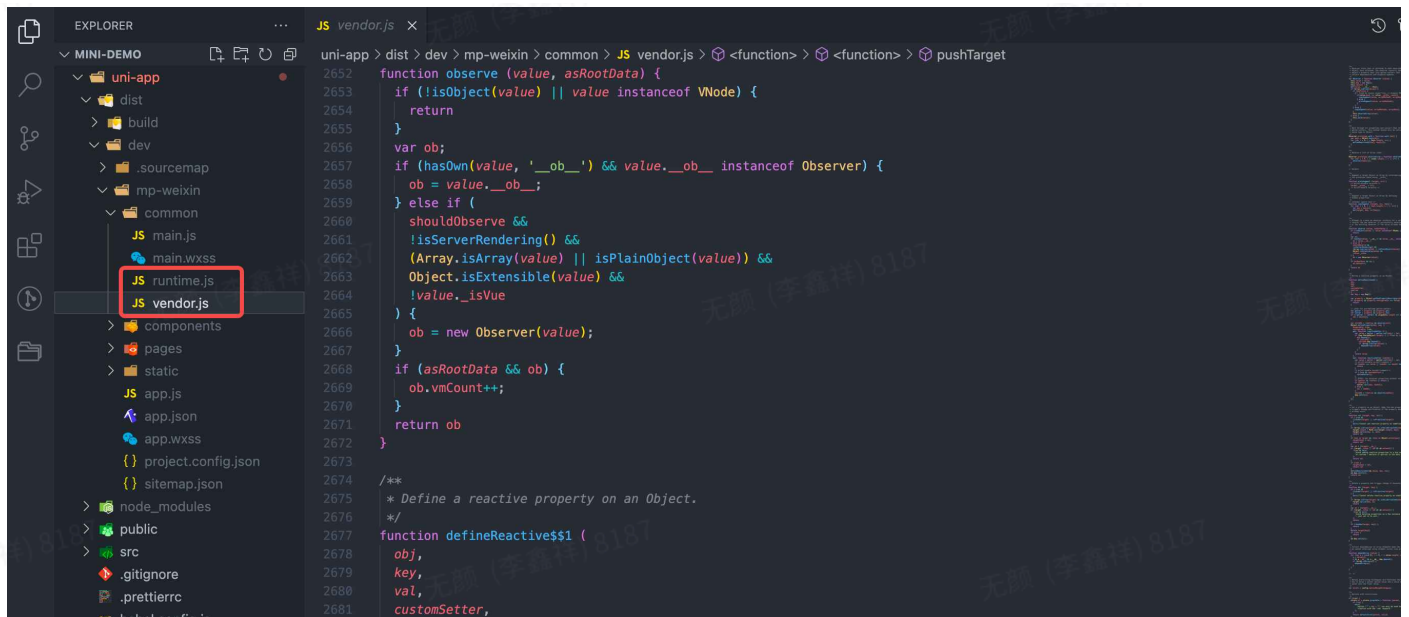
结论: 原生开发性能表现优于 uni-app, 但差异不大

代码包对比 (左原生、右 uni-app):



uni-app 多出的部分:

- vendor (Vue、Vuex)
- runtime



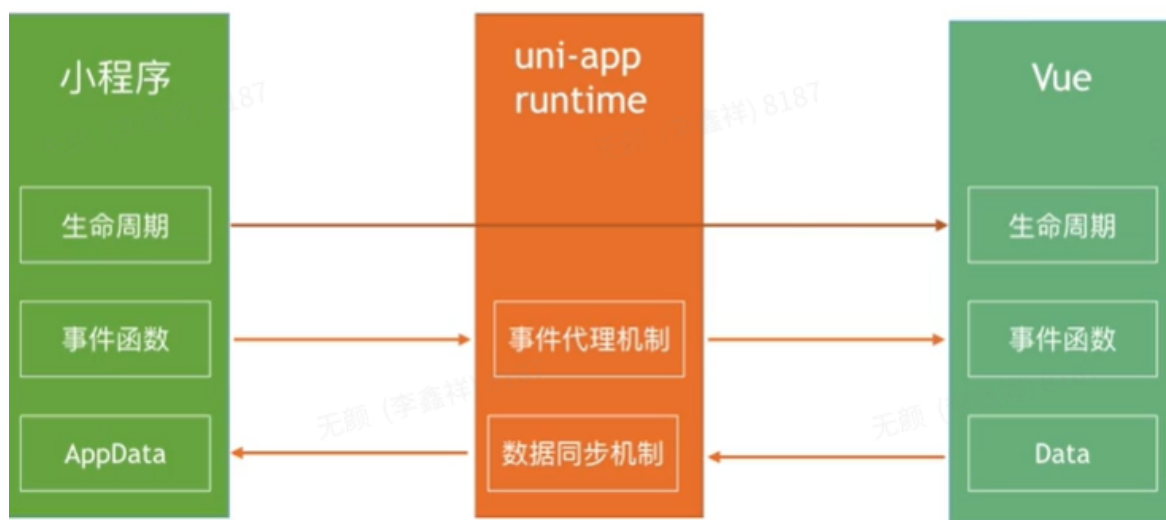
结论：uni-app 最终打包文件会比原生多出 70kb 左右，用于 vue、vuex 等支持

1.4 其他说明

uni-app 基于 `vue.js` runtime，在运行时补充规范实现跨端，与原生开发比较明显的差异存在于此。

发布到微信小程序时主要工作：

- 编译器：将 `.vue` 文件拆分成 `wxml/wxss/js/json` 4个原生页面文件
- 运行时：实现 `vue.js` 到小程序的数据同步，及小程序到 `vue.js` 的事件代理



与微信小程序跨端相关依赖：

- @dcloudio/uni-mp-vue
- @dcloudio/uni-mp-weixin
- @dcloudio/uni-ui
- @dcloudio/uni-template-compiler
- @dcloudio/webpack-uni-mp-loader
- @dcloudio/webpack-uni-pages-loader

1.5 总结

- 原生开发性能表现优于 uni-app，但差异不大
- uni-app 最终打包文件会比原生多出 70kb 左右，用于 vue、vuex 等支持

二、开发对比



本节目的：主要是让开发者在选择 uni-app 开发后，需要关注哪些差异以及注意事项

官方文档：

- [原生开发文档](#)
- [uni-app 开发文档](#)

2.2 uni-app 与 原生

微信原生为自创语法，与前端生态流行的 DSL (vue、React) 存在较大差异。建议直接查看文档进行对比。

2.2 uni-app 与 vue

`uni-app` 发布到微信小程序时，由于平台限制，无法实现全部vue语法。

Vue.js 在 `uni-app` 中使用差异主要集中在两个方面：

- 新增：`uni-app` 除了支持Vue实例的生命周期，还支持 [应用生命周期](#) 以及 [页面生命周期](#)
- 受限：在微信小程序端部分功能受限，[具体见](#)

注意：`uni-app` 中不支持的语法，编译器会报错

其他：

- [vue 在 uni-app 中的差异](#)
- [使用 Vue.js 的注意](#)
- [微信小程序开发注意](#)