uniapp 项目开发经验总结

前言

总结 uniapp 多端项目三个月开发维护的经验，遇到并解决了什么困难，收获了什么。多端指：

- Andorid: H5、微信 H5、App
- iOS: H5、微信 H5、App

issues

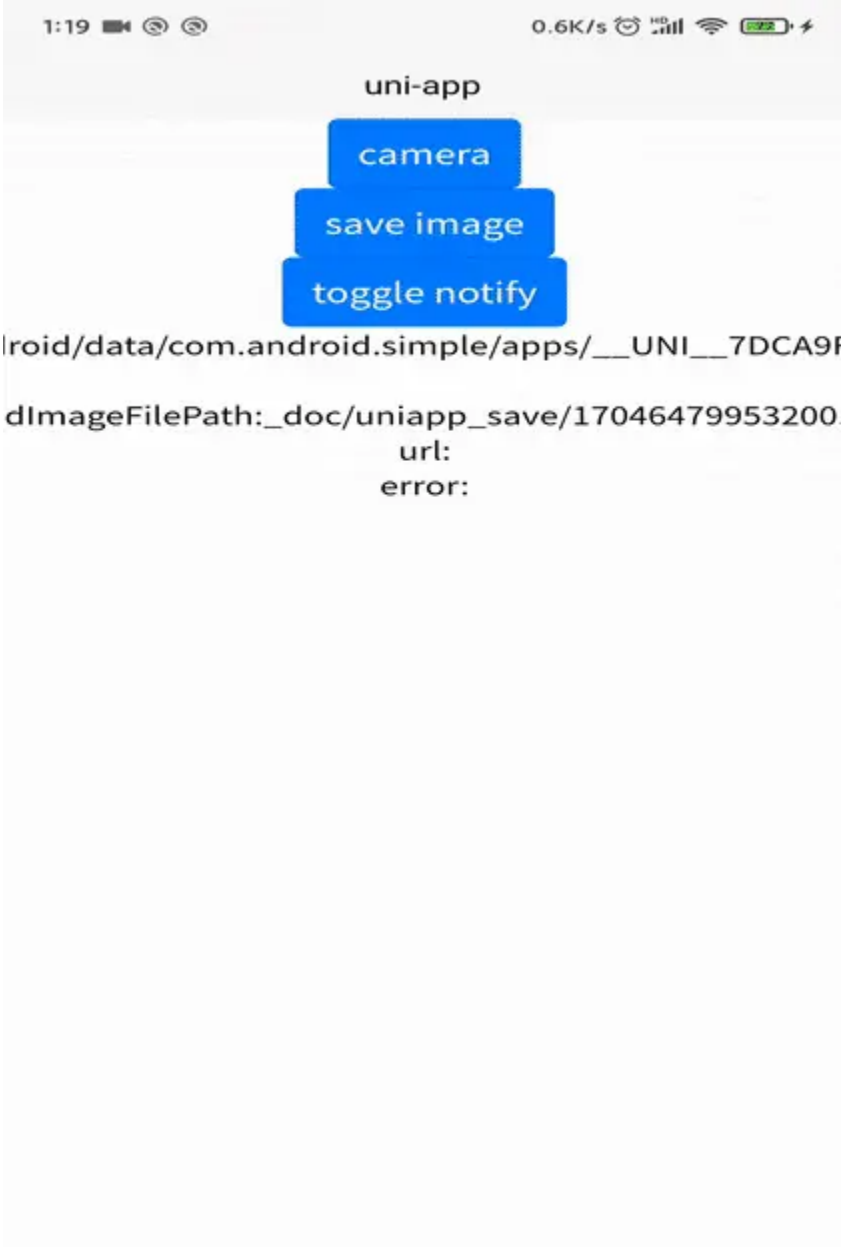
Android 拍照闪退问题

部分机型中调用 `uniapp` 提供的 Api 进行拍照会偶现应用闪退，已经确认是系统回收资源，结束了应用进程导致的（调用系统相机进行拍摄，应用处于后台）。

解决方案：

- 如果项目有需要保活的相关功能，可以做保活来提升应用优先级，避免拍照时进入后台被系统回收资源。
- 如果项目没有保活的业务，做了保活很大概率无法通过应用商店的审核，此时可以考虑通过自定义应用相机来实现。

目前 uniapp 插件市场有很多自定义相机可以使用，或者可以基于 uniapp 官方的 live-pusher 组件和 nvue 结合自定义自己的相机；这里我是写了个仿微信应用内相机的原生插件：



主要使用了三个依赖库来完成拍照、编辑、裁切功能：

- CameraView
- PhotoEditor
- uCrop

感兴趣的可以去 u-android-native-plugin-camera 看看，或者下载 demo 体验。

搞这个还是花了不少时间的，毕竟不是专业的 Android 开发，后续可能会就实现这个功能写一篇文章作为经验分享。

应用内通知

项目是 C 端项目，有 C 到 C 的通讯需求，也接入了各大平台的推送通知功能；系统自带的通知在应用处于前台时是不会弹出提示的，而项目要尽量跟微信对齐，需要能够在应用内弹出通知，也就是这种效果：

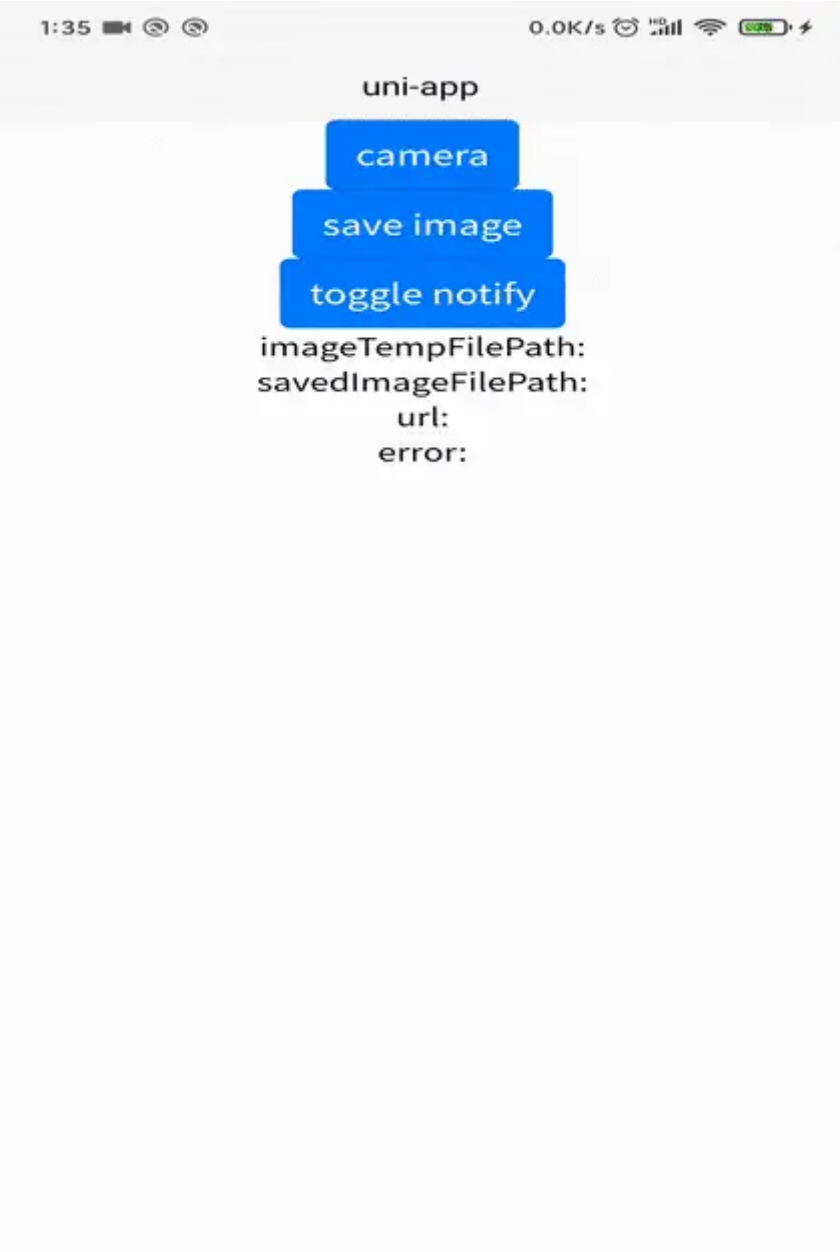
这里我对比了微信 Android 端和 iOS 端的应用内通知，区别如下：

- Android 端使用的是系统通知接口，与推送通知是相同的（需要通知权限）。
- iOS 端，微信是自定义了一个通知视图，与系统推送通知不同，猜测可能是 iOS 没有提供相关接口，无法创建系统通知。

这个需求在项目中是使用自定义组件来完成的，这种解决方式有两个很明显的缺点：

1. App 端无法使用全局组件，自定义组件必须在所有页面中引入一遍。
2. App 端，切换页面时，即使在新的页面继续弹出通知，也会出现闪烁的情况。

我尝试使用原生插件的方式来解决，参考了 NotificationToast 的实现方式，使用 Andorid 中的 `Toast` 类并加上一些黑科技来模拟系统通知，如下：



因为只是一个 `Toast` 轻提示, 所以不需要通知权限，可以做到的功能大概如下（PS：不建议使用这种方式代替系统通知，当做应用内通知使用还是可以的）：

- 自定义视图
- 自定义触摸、点击事件，以此可以完成清除通知、跳转页面等功能

- 应用内、应用外弹出通知

想做到但没有做到的功能：自定义弹出动画，尝试了网上的一些文章都不能很好的完成效果。

iOS 端，因为没有 iOS 的开发经验，所以不知道有什么方式可以解决，不过也就是添加一个视图，肯定也是能够实现的。

感兴趣的可以去 u-android-native-plugin-notify 看看，或者下载 demo 体验。

iOS 侧滑返回

iOS 侧滑返回是无法阻止的，如果我们在导航守卫中阻止用户离开，而用户使用了侧滑进行了强制返回，此时会出现这种情况：页面卡顿，一段时间后重新回到那个阻止离开的页面，如下图：

用户通过侧滑强制返回时，不能阻止，所以需要知道当前是否是侧滑；经过查找资料和测试，侧滑返回可以靠以下特征来标识：

1. touchstart 的触摸坐标 x 点在指定区域内（比如 0-50px 的范围）
2. touchmove 时，触摸坐标 x 点变为负值

基于这些特征我写了个判断是否是侧滑返回的工具：

```
/*
 * @Author: yuanyxh
 * @Date: 2023-12-27 12:04:11
 * @Last Modified by:   yuanyxh
 * @Last Modified time: 2023-12-27 12:04:11
 */

import Vue from "vue";

const ZERO = 0;

/** 最小左侧触摸开始位置 */
const MINIMUM_LEFT_DISTANCE = 100;
/** 最小移动距离 */
const MINIMUM_MOVE_DISTANCE = 50;
/** 最小间隔事件 */
const MINIMUM_TIME_INTERVAL = 100;

/**
 *
 * @callback OnSideSlipListener
 * @param {number} sideTime 上次侧滑事件
*/

/**
 *
```

```
 * @description 侦听 ios 侧滑事件，通过此工具配合 beforeRouteLeave 导航守卫判断是否侧滑返回
 */
const sideSlip = (function sideSlipListener(global) {
  let startX = 0,
    endX = 0;
  let sideTime = 0;

  const vm = Vue.prototype;

  /** @type {OnSideSlipListener[]} */
  const callbackList = [];

  function start(e) {
    const point = e.touches?.length ? e.touches[0] : e;

    startX = point.pageX;
  }

  function end(e) {
    const point = e.changedTouched?.length ? e.changedTouched[0] : e;

    endX = point.pageX;

    if (
      startX >= ZERO &&
      startX <= MINIMUM_LEFT_DISTANCE &&
      endX < ZERO &&
      vm.windowWidth - Math.abs(endX) > MINIMUM_MOVE_DISTANCE
    ) {
      sideTime = Date.now();

      callbackList.forEach((callback) => callback(sideTime));
    }
  }

  function isWithinValidityPeriod(time) {
    const interval = Date.now() - time;

    return interval >= ZERO && interval < MINIMUM_TIME_INTERVAL;
  }

  if (vm.isIos) {
    global.addEventListener("touchstart", start);
    global.addEventListener("touchend", end);
    global.addEventListener("touchcancel", end);
  }
```

```
73
74    return {
75      /**
76       * @description 是否左滑
77       * @readonly
78       */
79      get isSideSlip() {
80        if (vm.isIos) {
81          return isWithinValidityPeriod(sideTime);
82        }
83
84        return false;
85      },
86      /**
87       * @description 判断侧滑时间是否在有效期内
88       * @param {number} time 侧滑时间
89       * @returns {boolean} 是否有效
90       */
91      isWithinValidityPeriod: isWithinValidityPeriod,
92      /**
93       *
94       * @description 侦听侧滑
95       * @param {OnSideSlipListener} callback
96       */
97      on(callback) {
98        if (typeof callback === 'function') {
99          callbackList.push(callback);
100       }
101     },
102     /**
103      * @description 取消事件侦听
104      * @param {OnSideSlipListener} callback
105      */
106     off(callback) {
107       const i = callbackList.indexOf(callback);
108
109       if (i === -1) return;
110
111       callbackList.splice(i, 1);
112     },
113   };
114 })(
115   (function() {
116     let global = {};
117
118     if (typeof window !== "undefined") {
119       return (global = window);
```

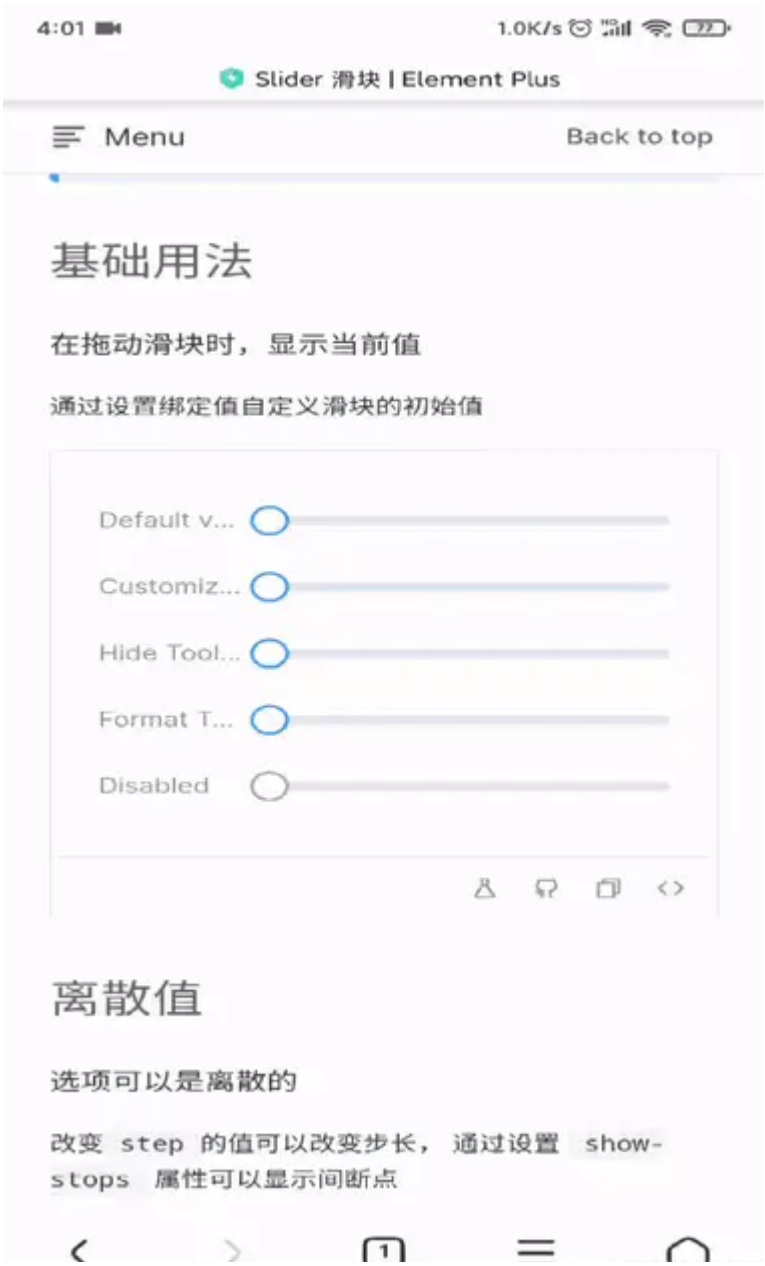
```
    }

    global = {
      addEventListener() {},
      removeEventListener() {},
    };

    return global;
  })()
);

export default sideSlip;
```

具体逻辑为：判断是否侧滑返回，touchend 时记录当前时间，在指定时间间隔内触发了诸如 `onUnload`、`beforeRouteLeave` 等钩子时就可以认为是侧滑返回。

小米浏览器滑动冲突

不知道从什么版本开始，小米浏览器加了一个让人恼火的功能：左右滑屏手势。通过这个功能可以让左右滑动手势控制网页的前进后退，如下：

看起来很美好，但却与很多项目中的滑动手势冲突，比如 Element 中的滑块功能：



这时候一般就会想到阻止冒泡、阻止默认事件来禁用左右滑动手势了，但恼火的点就在于没有效果，由此可以猜测这个功能的实现没有在浏览器内核中或者没有适配我们熟知的那一套 js 事件体系。

目前还没有研究出解决方案，有知道如何解决的大佬欢迎讨论。

后台定时器

因为内核的限制，网页和基于 webview 实现的框架下使用定时器时，进入后台一段时间后定时器会被挂起，回到前台时才继续执行，导致程序的运行和我们预想的有偏差。

解决方案：

- 一种比较简单的解决方案是定时器 + 系统时间的结合，但是系统时间可以被更改，这种方式适用于不是那么重要的定时任务。
- H5 中也可以尝试使用定时器 + `performance.now()` 结合来完成，`performance.now()` 返回的是距 时间源 过去的时间，且不会被系统时间影响。

- 以上两种方式都不能在定时器被挂起时继续执行任务，如果需要在后台时继续执行任务，可以考虑使用 worker 的方式。

这里给出个人认为的最佳实践：

1. 如果使用了定时器来完成持续的动画效果或视图变化，在程序进入后台时主动停止，回到前台时恢复
2. 较简单的定时任务，可以使用定时器 + 系统时间的组合，能够在多端生效
3. 需要保证任务的执行逻辑，H5 可以使用定时器 + `performance.now()`，App 可以使用 worker 或原生插件
4. 对于需要持续执行，不影响视图的任务，使用 worker

软键盘高度的获取

App 端软键盘高度的获取还是比较简单的，H5 端就比较难受了，因为不同浏览器对于软键盘弹出时的页面模式是不能确认的，导致我们不能简单的通过 `window.onresize` 事件来判断软键盘是否弹出，好在目前主流浏览器都支持了 `Window.visualViewport` 接口，也能兼容到大部分的主流浏览器版本。

我们可以通过 `Window.visualViewport` 接口的 `resize` 事件，接收到可视窗口变化的事件，以此计算出软键盘的高度。

```
import Vue from "vue";

const DELAY_TIME = 300;

let callbackList = [];
let unimplementedChangeList = [];

function emit(target, payload) {
  if (target.length) {
    for (let i = 0; i < target.length; i++) {
      target[i](payload);
    }
  }
}

// 页面原始高度
let windowHeight = 0;
function onKeyboardHeightChangeWithH5() {
  let extraHeight = 0;
  let hasFocus = false;
  let originScrollY = 0;

  let cancheHeight = 0;
```

```
24
25  let keyboardChangeTimer = null;
26
27  function exec(height) {
28    const keyboardHeight = Math.max(0, windowHeight - height);
29
30    if (cancheHeight === keyboardHeight) {
31      return;
32    }
33
34    cancheHeight = keyboardHeight;
35
36    const { isIos } = Vue.prototype;
37
38    emit(callbackList, {
39      extra: isIos ? extraHeight : 0,
40      height: keyboardHeight,
41    });
42  }
43
44  window.addEventListener(
45    "focus",
46    (e) => {
47      if (
48        e instanceof FocusEvent &&
49        (e.target instanceof HTMLInputElement ||
50          e.target instanceof HTMLTextAreaElement ||
51          e.target.contenteditable)
52      ) {
53        hasFocus = true;
54
55        originScrollY = window.scrollY;
56
57        setTimeout(() => {
58          hasFocus = false;
59        }, 600);
60      }
61    },
62    { capture: true }
63  );
64
65  window.addEventListener(
66    "blur",
67    (e) => {
68      if (
69        e instanceof FocusEvent &&
70        (e.target instanceof HTMLInputElement ||
```

```
          e.target instanceof HTMLTextAreaElement ||
          e.target.contenteditable)
      ) {
        hasFocus = true;

        setTimeout(() => {
          hasFocus = false;
        }, DELAY_TIME);
      }
    },
    { capture: true }
  );

  window.addEventListener(
    "scroll",
    () => {
      if (hasFocus) {
        extraHeight = window.scrollY - originScrollY;
      }
    },
    { capture: true }
  );

  if (typeof window.visualViewport !== "undefined") {
    return window.visualViewport.addEventListener("resize", (e) => {
      if (hasFocus === false) {
        return emit(unimplementedChangeList);
      }

      if (keyboardChangeTimer) {
        clearTimeout(keyboardChangeTimer);
      }

      keyboardChangeTimer = setTimeout(() => {
        exec(e.target.height);
      }, DELAY_TIME);
    });
  }

  window.addEventListener("resize", () => {
    if (hasFocus === false) {
      return emit(unimplementedChangeList);
    }

    if (keyboardChangeTimer) {
      clearTimeout(keyboardChangeTimer);
    }
```

```
118
119     keyboardChangeTimer = setTimeout(() => {
120       exec(window.innerHeight);
121     }, DELAY_TIME);
122   });
123 }
124
125 let isInited = false;
126
127 function initKeyboardHeightChangeListener() {
128   // #ifdef APP-PLUS
129   uni.onKeyboardHeightChange((res) => {
130     emit(callbackList, { extra: 0, height: res.height });
131   });
132   // #endif
133
134   // #ifdef H5
135
136   windowHeight = Vue.prototype.windowHeight;
137
138   onKeyboardHeightChangeWithH5();
139   // #endif
140 }
141
142 /**
143  * @callback OnKeyboardHeightChangeCallback
144  * @param {{ extra: number; height: number }} 键盘高度(ios 中还有底部块的高度)
145  */
146 /**
147  * @typedef Options
148  * @type {Object}
149  * @property {boolean} reset app 端对 uni.onKeyboardHeightChange 重置侦听
150  */
151 /**
152  *
153  * @description 侦听键盘高度变化事件，对多端做兼容处理
154  * @param {OnKeyboardHeightChangeCallback} callback 键盘高度变化事件回调
155  * @param {Options} options 额外参数
156  */
157 function onKeyboardHeightChange(callback, options = {}) {
158   const { isPc } = Vue.prototype;
159
160   if (isPc) {
161     return console.error(
162       "PC devices do not need to listen for soft keyboard height"
163     );
164   }
```

```
165
166   // #ifdef APP-PLUS
167   if (options.reset) {
168     isInited = false;
169
170     callbackList = [];
171   }
172   // #endif
173
174   if (typeof callback === "function") {
175     callbackList.push(callback);
176   }
177
178   if (isInited === false) {
179     isInited = true;
180
181     initKeyboardHeightChangeListener();
182   }
183 }
184
185 /**
186  *
187  * @description 移除事件侦听函数
188  * @param {OnKeyboardHeightChangeCallback} callback 需要移除的已注册函数
189  */
190 function offKeyboardHeightChange(callback) {
191   if (typeof callback === "function") {
192     const index = callbackList.indexOf(callback);
193
194     if (index >= 0) callbackList.splice(index, 1);
195   }
196 }
197
198 /**
199  *
200  * @description H5 页面高度变化但未执行键盘高度变化事件时触发
201  * @param {OnKeyboardHeightChangeCallback} callback 需要添加的注册函数
202  */
203 function onUnimplementedChange(callback) {
204   const { isPc } = Vue.prototype;
205
206   if (isPc) {
207     return console.error(
208       "PC devices do not need to listen for soft keyboard height"
209     );
210   }
211
```

```
212   if (typeof callback === "function") {
213     unimplementedChangeList.push(callback);
214   }
215
216   if (isInited === false) {
217     isInited = true;
218
219     initKeyboardHeightChangeListener();
220   }
221 }
222
223 /**
224  *
225  * @description H5 页面高度变化但未执行键盘高度变化事件时触发
226  * @param {OnKeyboardHeightChangeCallback} callback 需要添加的注册函数
227  */
228 function offUnimplementedChange(callback) {
229   if (typeof callback === "function") {
230     const index = unimplementedChangeList.indexOf(callback);
231
232     if (index >= 0) unimplementedChangeList.splice(index, 1);
233   }
234 }
235
236 export default {
237   onKeyboardHeightChange,
238   offKeyboardHeightChange,
239   onUnimplementedChange,
240   offUnimplementedChange,
241 };
```

上述代码对于软键盘高度的获取做了统一的封装，对于 App 端使用 `uni.onKeyboardHeightChange` 接口；H5 端利用 `Window.visualViewport`，同时使用 `window.onresize` 兜底。

思路与看法

全局弹窗

用过 uniapp 的人应该都对全局这个词很敏感，因为很实用但却无法实现（除 H5 外）。接触这个项目后我也感到了局部组件带来的一些问题：

- 书写相同的结构代码，做同一件事

- 一个页面多个实例，状态不能统一（如一个通用的弹窗组件，在一个页面引入了多次，代码逻辑同时弹出了其中两个弹窗，弹窗重叠在一起；如果使用全局弹窗就能在内部维护，关闭第一个后再弹出第二个）
- 。。。

这些问题看起来很小，但对项目的影响巨大，所以我也在找能够实现全局弹窗的办法。

看过插件市场的一些全局弹窗实现，基本思路都是使用一个页面作为全局弹窗的载体，需要弹出时跳转至这个页面即可。

App 端中一个页面就是一个 webview，其实我们可以通过 H5+ Api 的方式创建 webview 来作为全局弹窗的载体，会比页面的形式更好一点，查看 uni-app-picker 组件的源码也可以发现，App 端的 `picker` 就是依赖于 H5+ webview 接口实现的，且经过实验，创建的 webview 是可以做到全局重用的。

我们还需要考虑这个全局弹窗的可扩展、可重用性：

- 考虑结构的可扩展性，是否可以传递组件实例在 webview 中创建
- 考虑事件的传递
- 考虑对外接口的封装

对项目的看法

这个项目是一个比较老的项目了，给我的感觉就是乱，很乱，没有一个统一的规范，你想写什么样的代码都可以，导致经过多个开发人员的手之后，各种风格的代码夹杂其中，给后续的开发维护造成很大的影响。

项目的初期没有规划好，在开发过程中也没有考虑封装统一，比如：

- 输入框的封装
  - 目前关于输入框和软键盘的问题还有很多，如果初期考虑了封装的话能够统一解决，会轻松很多
- 弹窗的封装
- 录音功能的封装
  - 多端中录音实现都不同，H5 使用 Recorder、微信 H5 使用 wx-jssdk、App 使用 uniapp 提供的接口；直接在页面中内嵌代码，而没有考虑封装为一个模块，内部区分，对外提供一致接口
- 全局事件的封装
  - 在 H5 中经常使用 `window.onresize` 之类的接口完成功能，可能一个组件就添加一个事件，应该封装为模块，内部注册一次事件，提供对外接口，事件触发时再分发出去

重构

一开始接手这个项目时就感觉项目问题很大，天天喊着需要重构重构，需要几个月的时间对项目进行大的重构，后来需求要将项目中使用 `scroll-view` 的列表替换为三方组件，在完成这个需求时就

发现了所谓大的重构是不切实际的。

在进行这个需求时出现的比较明显的问题：

1. 组件与 uniapp `scroll-view` 的属性、逻辑不兼容，替换需要格外小心，耗时是很漫长的
2. 替换过程中，可能遗漏了部分逻辑或结构出现错误；如果列表功能复杂，可能很长时间内不会发现这个问题

从这个需求中也更能体会到 《重构2》 中作者所践行的 ***一步一测试*** 的必要性了，也更深刻的理解了作者表达的：***当你有时间且你认为这段代码需要重构时才进行重构，对于那些永远不会碰到的，不要去碰它（让营地比你来时更干净）***