



WEEX Conf

企鹅电竞Weex实践和性能优化

渠宏伟

Contents

01.

接入和收益

02.

努力填坑

03.

性能优化



H5页面存在的问题



加载耗时长

WebView启动慢、资源加载慢



交互体验差

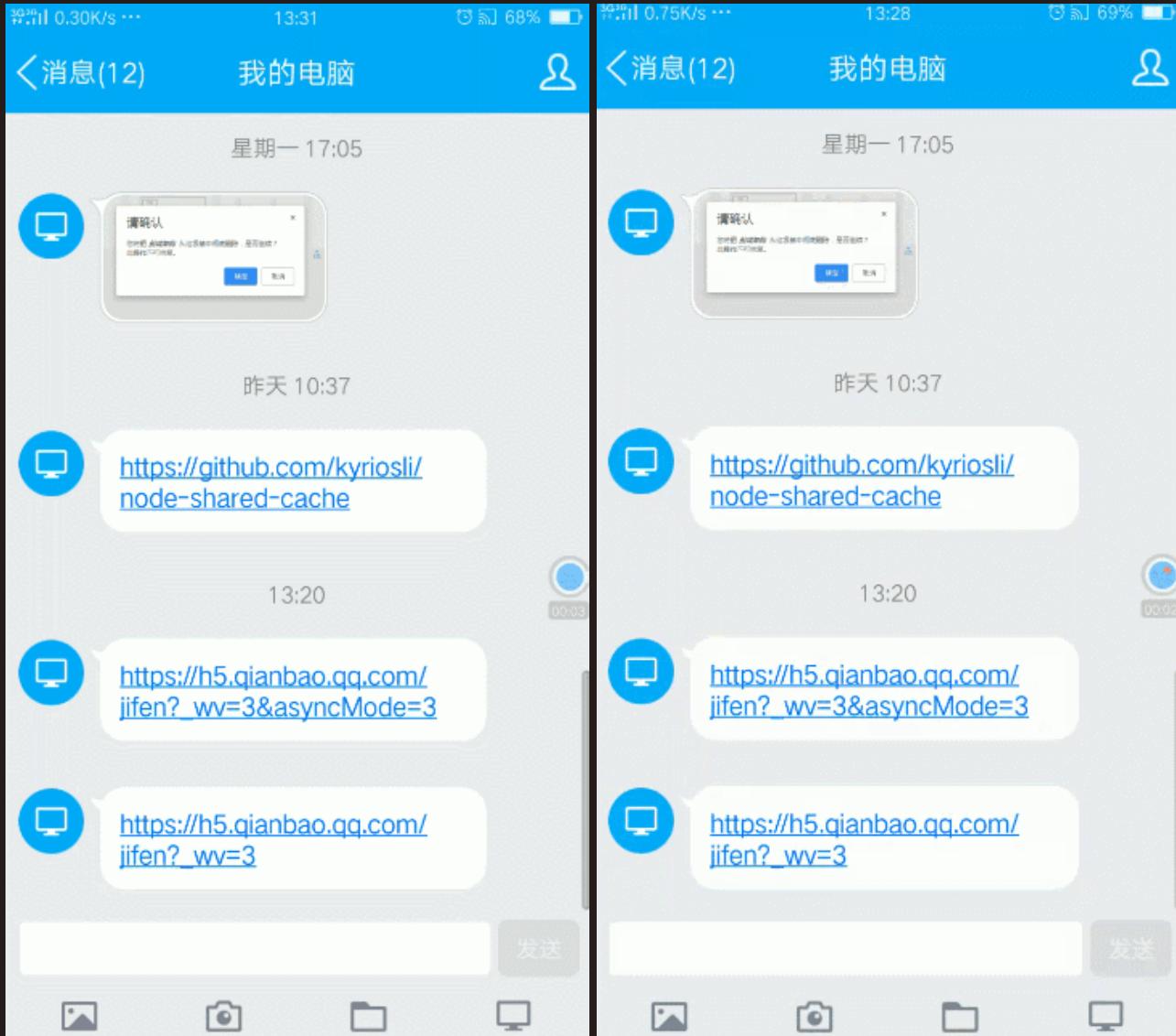
长列表滚动、测滑不流畅

H5极速加载方案 —— VasSonic

- WebView和页面并行加载
- 资源离线预推
- 页面局部刷新
- 动态缓存

H5页面首屏实现秒开

<https://github.com/Tencent/VasSonic>



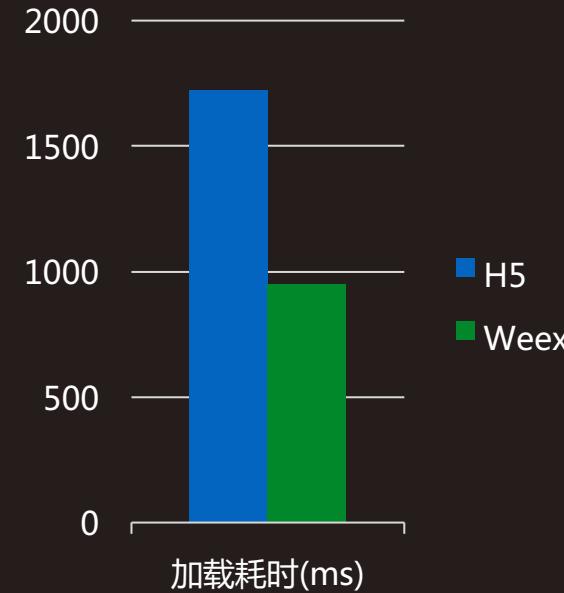
企鹅电竞Weex实践

前端页面原生渲染，增强前端页面体验

一次开发支持三端（Android、iOS、H5）
运行，提高开发效率



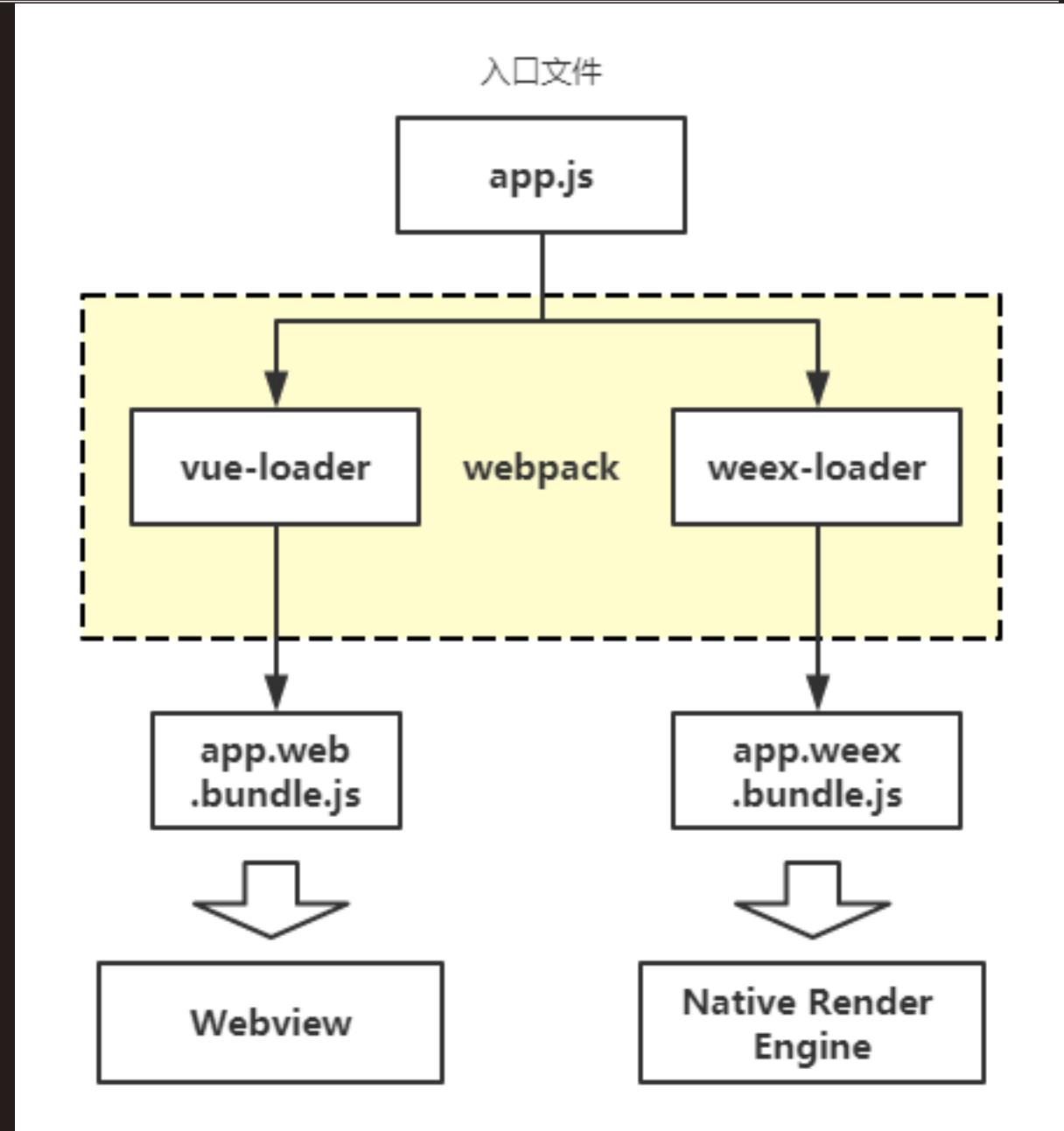
企鹅电竞Weex实践



- 内存降低 **45%**, 帧率提高 **15.7%**; 打开耗时下降 **44.9%**
- 开发效率比终端提升 **40%**, 有效释放终端开发人力
- 页面更新不依赖版本发布, 特性发布效率提升

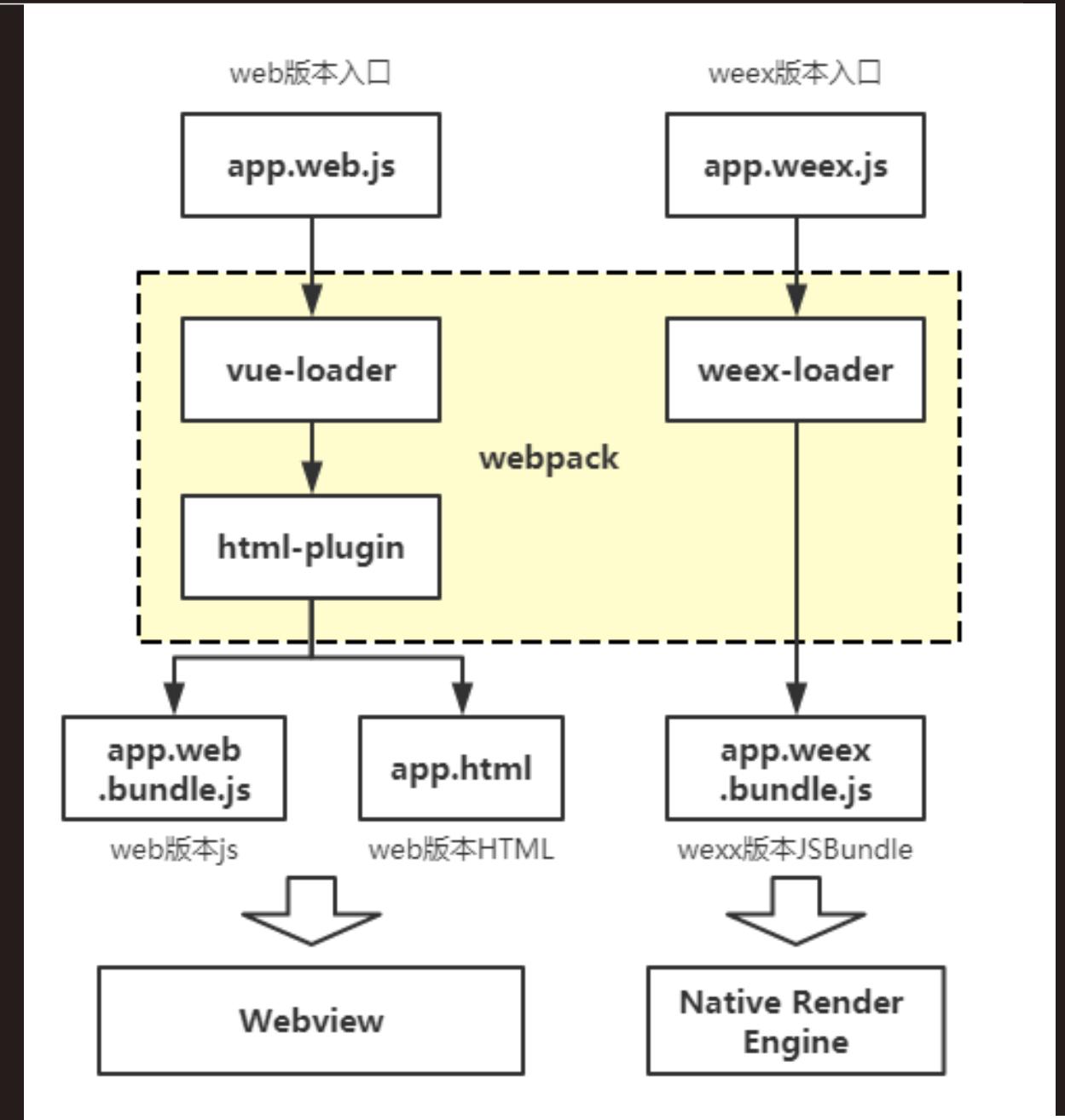
构建流程改造

实际业务比较复杂，时常需要在webview中增加一些特殊逻辑。将特殊逻辑、冗余代码打入到终端运行的JSBundle并不合理。



构建流程改造

分离入口再引用相同的app.js，可以让web版本和weex版本保持相同逻辑，也可以独立扩展，互不影响。



接头暗号

头部的{"framework": "Vue"}告诉解析器用Vue的语法解析JSBundle。如果删除了这行注释，将会白屏。

Uglify压缩会删除注释，导致”接头暗号”被删除。压缩必须放在插入framework之前

```
// { "framework": "Vue"}  
***** (function(modules) {  
.....  
***** })
```

plugins:[

```
//uglify必须在bannerPlugin前面  
new webpack.optimize.UglifyJsPlugin({compress:  
warnings: false}),
```

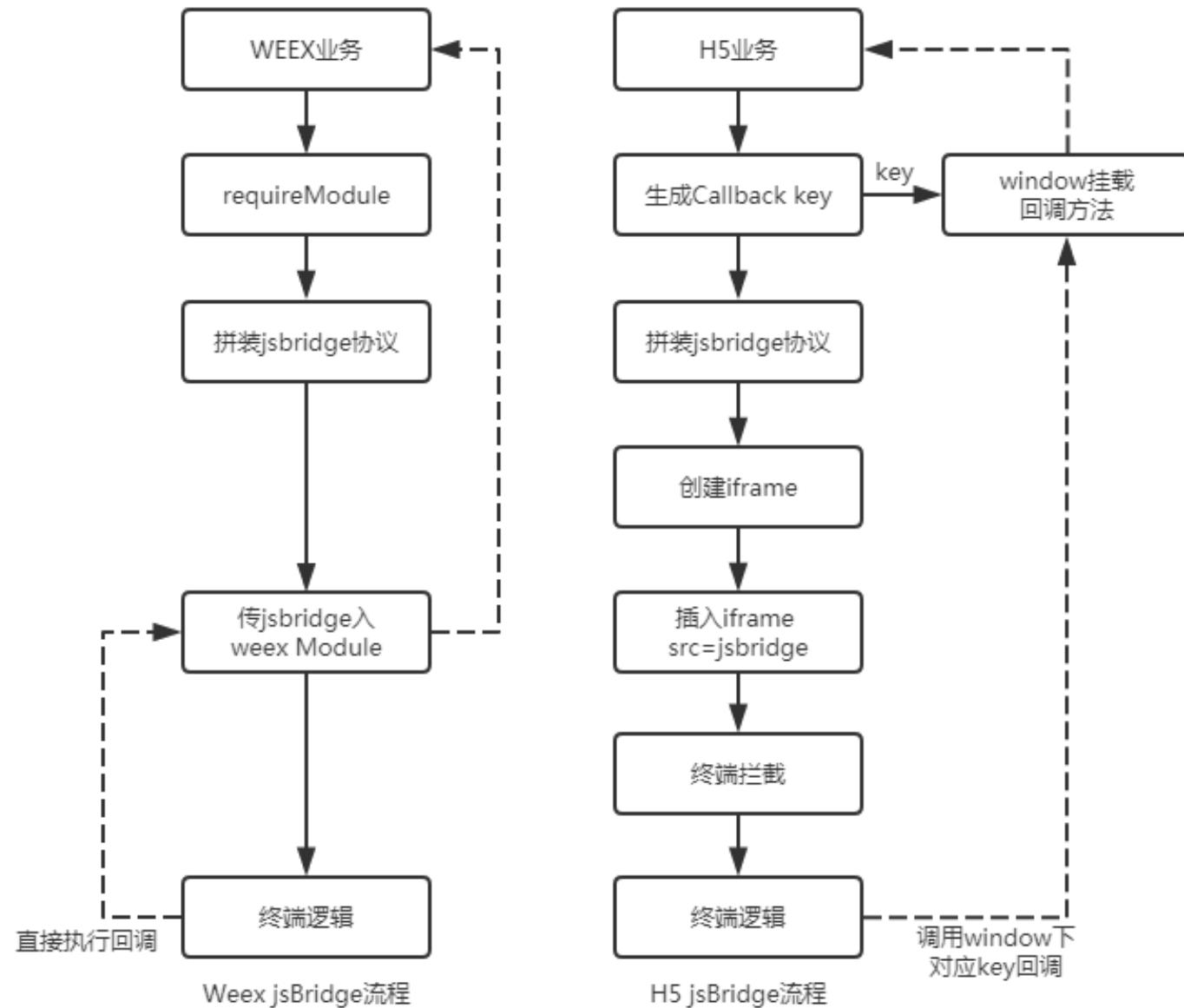
```
new webpack.BannerPlugin({  
banner: '// { "framework": "Vue" }\n',  
raw: true  
})  
]
```



调用终端接口

在webview中使用插入iframe或者img的方式传schema协议与终端通信。现有业务有大量的jsapi，如果全部改造成module工程量很大。

企鹅电竞扩展了一个公共的jsbridge的module用于转发jsapi，底层jsapi的解析和逻辑保持不变。



获取cookie

H5中我们使用document.cookie可以获取浏览器的cookie信息。

在weex中如何实现？

我们利用weex提供的扩展Module的能力，在终端中扩展一个获取cookie的方法

```
// 获取cookies
export async function getCookies() {
  if (typeof document === 'object' && typeof
  document.cookie !== 'undefined') {
    return document.cookie
  } else {
    return await getNativeCookie()
  }
}

// 从终端获取cookies
export function getNativeCookie() {
  let dataFetch = weex.requireModule('dataFetch')
  return new Promise((resolve, reject) => {
    dataFetch.fetch('cookie', function (map) {
      let result = map.result
      if (result === 0) {
        resolve(map.data)
      } else {
        reject(map.message)
      }
    })
  })
}
```



支持cookie

iOS基于UI webview共享cookie。
iOS天然支持请求带上cookie

android需要终端支持，也能实现请求自动带cookie

```
public String getCookie(String url) {  
    CookieManager cookieManager = CookieManager.getInstance();  
    cookieManager.setAcceptCookie(true);  
  
    // 先清除旧的cookie  
    String oldCookie = cookieManager.getCookie(url);  
    if (oldCookie == null) {  
        // 设置错误标志，并在10秒后过期，后台可通过此标志监控客户端异常情况  
        SimpleDateFormat dateFmt = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z", Locale.US);  
        dateFmt.setTimeZone(TimeZone.getTimeZone("UTC"));  
        Date expiresDate = new Date(System.currentTimeMillis() + 10 * 1000);  
        String expires = dateFmt.format(expiresDate);  
        return "login_key_set_failed=AlreadyLogout; EXPIRES=" + expires + ";";  
    } else {  
        return oldCookie;  
    }  
}
```

获取userAgent

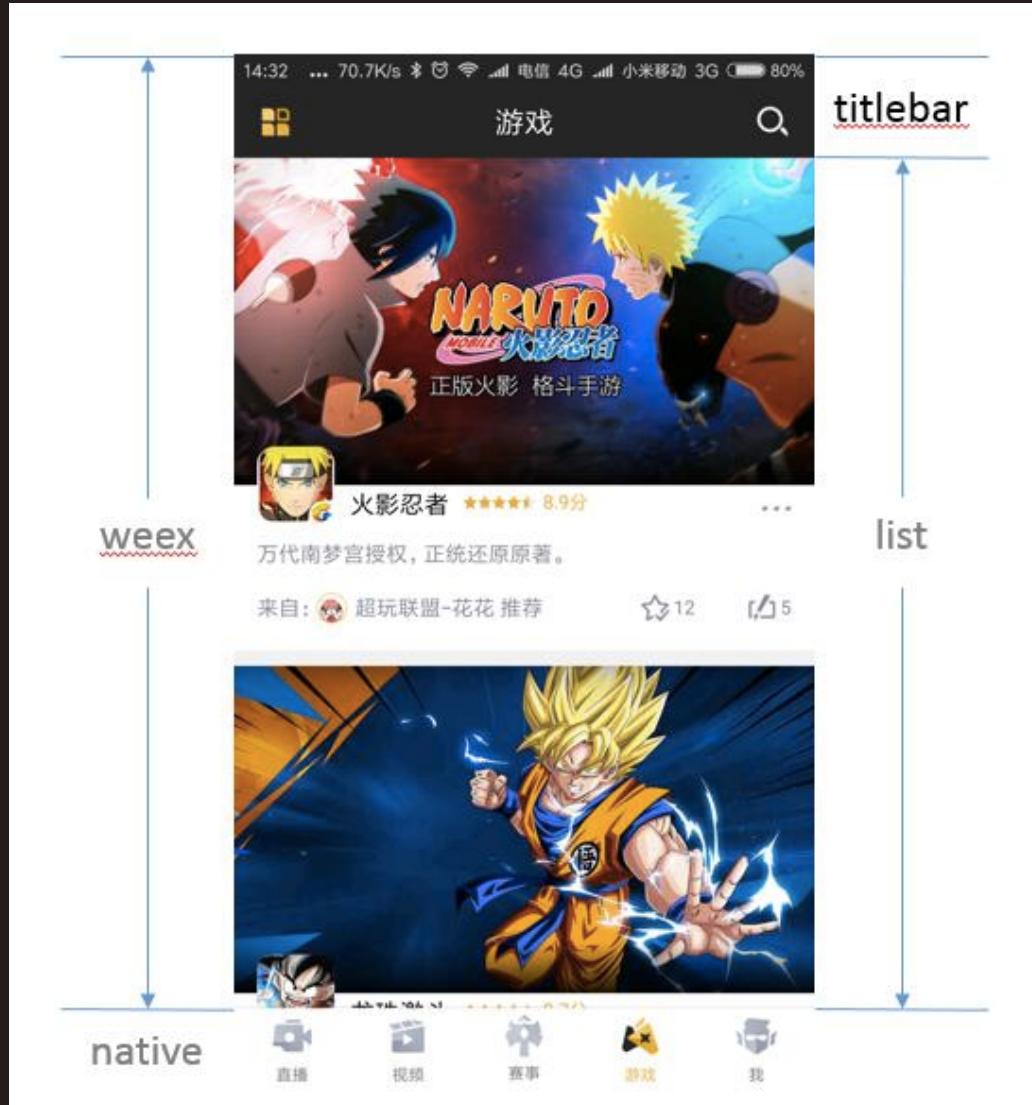
在weex.config.env添加userAgent属性。

需要注意的是，从weex实例中获取的userAgent是静态的。一旦weex实例创建就固定下来，终端修改UA也不会跟着变化

```
export default function () {
  let userAgent = ""
  if (typeof weex === 'object' && weex.config &&
  weex.config.env) {
    userAgent = weex.config.env.userAgent || ""
  } else if (typeof window === 'object' &&
  window.navigator) {
    userAgent = window.navigator.userAgent || ""
  }
  return userAgent
}
```



如何实现1px



```
let env = weex.config.env  
let deviceWidth = env.deviceWidth  
let scale = env.scale  
// 获取布局宽度  
export function getPx(pt) {  
    return pt * 750 / (deviceWidth / scale)  
}  
  
// 获取真实宽度  
export function getPt(px) {  
    return px * (deviceWidth / scale) / 750  
}
```

横屏适配



```
function getHWidth (w) {  
    return deviceWidth / w * 750  
}  
  
const meta = weex.requireModule('meta')  
// 配置 viewport 的宽度为  
meta.setViewport({  
    width: getHWidth(w) // w为设计稿宽度  
})
```

:class 语法限制

```
:class="[showBottomBorder ? 'ui-border-b' : "]" // 只支持数组语法  
:class="showBottomBorder ? 'ui-border-b' : "" // 不支持三元运算符  
:class="{ 'ui-border-b': showBottomBorder }" // 不支持对象语法
```



点击态

项目比较常见的点击态多半是透明度的变化，如按钮、列表、链接等，css的做法是添加伪类 (:active)，weex中也同样支持，但是weex需要在原样式中添加 opacity:1，否则点击后回不到初始状态；

此外，:active使用时，background-image在ios下会失效。

```
<template>
  <div class="ui-btn">
    <text class="ui-btn-text">下载</text>
  </div>
</template>
<style scoped>
  .ui-btn{
    opacity: 1; /*必须添加*/
  }
  .ui-btn:active{
    opacity: .5;
  }
</style>
```



文本截断

文本从限制1行到不限制可以用lines:0

城市赛战报,《王者荣耀》城市赛郑州站欢
乐落幕城市赛战报,《王者荣耀》城市赛郑
州站欢乐落幕城市赛战报,《王者荣耀》城
市赛郑州站欢乐落幕城市赛战报,《王者荣耀》
城市赛郑州站欢乐落幕

```
<template>
  <text class="info-text" @click="textClick" :style="textStyle">城市
    赛战报,《王者荣耀》城市赛郑州站欢乐落幕城市赛战报,《王者荣耀》城市赛郑州
    站欢乐落幕城市赛战报,《王者荣耀》城市赛郑州
    站欢乐落幕城市赛战报,《王者荣耀》城市赛郑州站欢乐落幕
</text>
</template>
<style scoped>
  .info-text{
    lines:1;
    text-overflow:ellipsis;
  }
</style>
<script>
  export default {
    data(){
      return { textStyle:{} };
    },
    methods:{
      textClick(){
        this.textStyle = { lines:0 }
      }
    }
  }
</script>
```



圆角抖动问题

问题存在于android下 weex sdk
配合在新版本weex sdk解决了这个问题

```

case Constants.Name.BORDER_TOP_RIGHT_RADIUS:
case Constants.Name.BORDER_BOTTOM_RIGHT_RADIUS:
case Constants.Name.BORDER_BOTTOM_LEFT_RADIUS:
    final Float radius = WXUtils.getFloat(param,null);
    final String finalKey = key;
    Float radius = WXUtils.getFloat(param,null);
    if (radius != null) {
        if (this instanceof WXDiv && mHost != null) {
            /* Hacked by moxun
             * Set border radius on ViewGroup will cause the Overlay to be cut and don't know why
             * Delay setting border radius can avoid the problem, and don't know why too, dog science.... */
            mHost.postDelayed(new Runnable() {
                @Override
                public void run() {
                    setBorderRadius(finalKey, radius);
                }
            }, 64);
        } else {
            setBorderRadius(finalKey, radius);
        }
        setBorderRadius(key, radius);
    }
    return true;
}

```





末剑

游戏评分: ★★★★★ 9.6分

开发者: 腾讯

5万人已安装

简介: 《末剑》是一款以武器操作为核心的横版过关角色扮演单机手游。荣获腾讯光子工作室群第一届独立...

#武侠 #冒险 #创意 #横版 #单机 #闯关

推荐理由 精彩视频 讨论区

评价 我要评价


齐天大圣--孙悟空
 初级游戏玩家
 很好玩！建议大家试玩一下！(*^__^*)
10分

终端crash问题

Android

- OOM
- PaintDrawable在API 21下偶现渲染crash
- Bitmap回收后继续使用导致crash
- instance被销毁后调用Toast的NPE问题
- WXThread.secure反射调用异常

iOS

- 未显示禁用estimatedRowHeight导致
- tableview在更新数据源时偶现crash
- 布局时node->get_child返回空指针导致crash
- text component渲染和测量高度时偶现crash

Android接入Weex crash率 **0.13% → 1.01%**



自动化测试方案

生成的终端view中的id是weex sdk自动生成的。自动化测试系统无法识别。

借助无障碍化ariaLabel属性作为原生view自动化测试标签。

```
<div :accessible="true"  
ariaLabel="pageEnd|index|div|  
游戏首页"></div>
```

The screenshot shows a mobile application interface. At the top, there's a navigation bar with 'Weex' and other icons. Below it is a search bar. The main content area displays several game cards, such as '生命倒计时' and '碧蓝航线'. On the right side, there are sections for '游戏专区' and '编辑推荐', featuring '碧蓝航线' and other games.

At the bottom, there are tabs for '直播', '视频', '赛事', '游戏', and '我'.

The central part of the screen shows a detailed view of the application's internal structure. A tree view on the left lists the following hierarchy:

- content
- tabhost
- None
- tabcontent
- real_content
- title_bar_root
- None
- None
- weexview
 - 0x1
 - 0x7
 - None
 - 0x2
 - None
 - None
 - None
 - None
- announce_text
- announce_close_btn
- common_title
 - common_title_inside
 - container_left
 - container_center
 - container_right
- shadow_border
- tabs
 - root_tab
 - root_tab
 - root_tab
 - root_tab

Below this tree view is a '控件属性' (Control Properties) table:

ID	0xf	HashCode	0x01A12136	ProcessName	com.tencent.qgame
Type	com.taobao.weex.ui.view.V	BoundingRect	(0, 1920, 1080, 0)	Description	pageEnd index div 游戏首页
Visible	True	Enabled	True		
Text		<input type="checkbox"/> 显示16进制			

At the very bottom, there are links for '查看帮助文档' and '问题反馈请点击'.

数据缓存优化

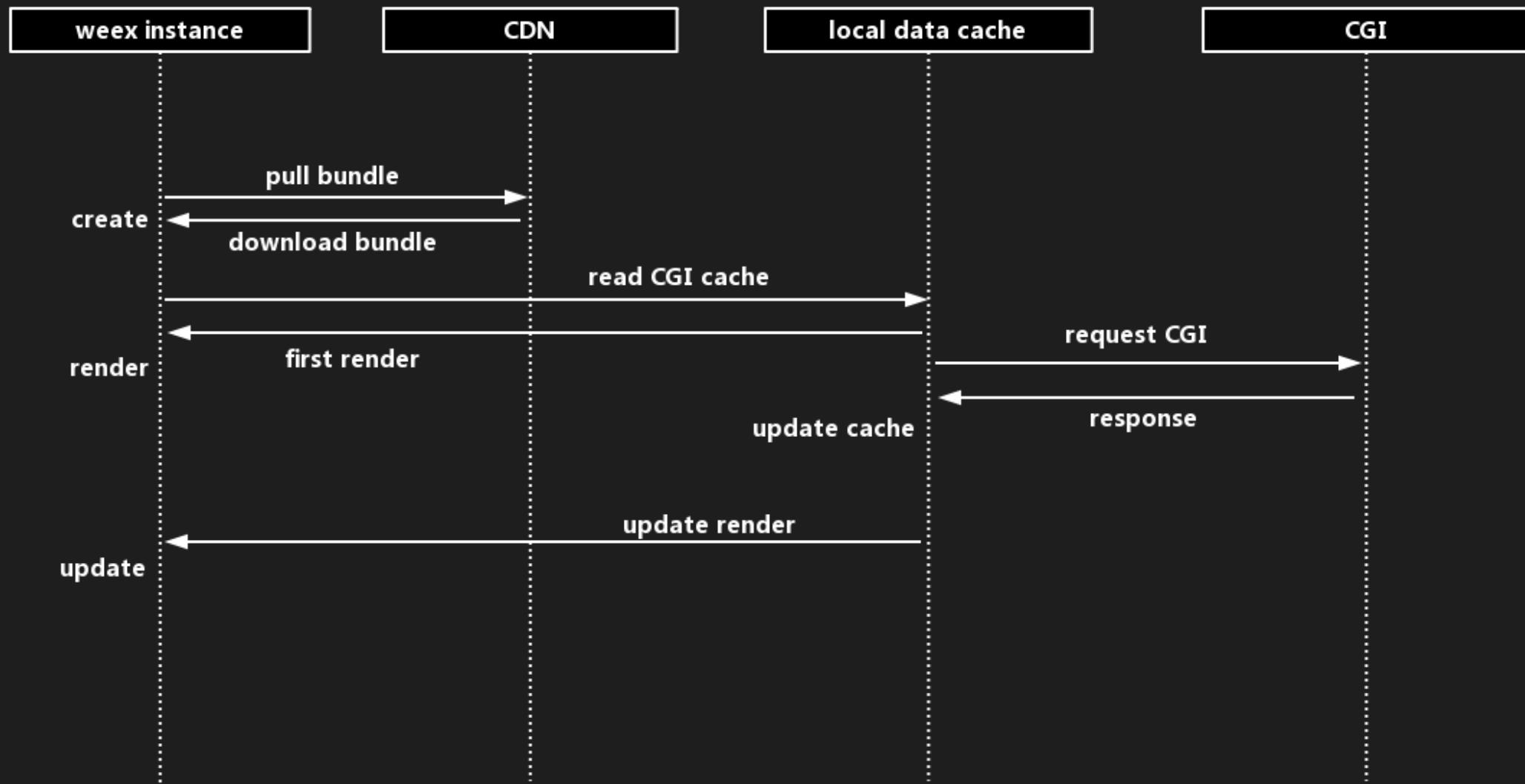
- 1、使用storage缓存接口请求数据
- 2、优先使用本地缓存数据，同步请求更新缓存数据和页面

```
/**
 * @desc 缓存辅助函数
 */
export function cacheHelper(key, fetchHelper, callback) {
  log('开始获取storage:' + Date.now())
  getStorageItem(key).then(result => {
    //命中缓存，优先使用缓存
    callback && callback(null, Object.assign({}, result, {'$_thisIsCache': true}), true)
    //命中缓存后，拉取数据更新
    fetchData(key, fetchHelper, callback)
  }).catch(err => {
    //未命中缓存，拉取数据
    fetchData(key, fetchHelper, callback)
  })
}

function fetchData(key, fetchHelper, callback) {
  fetchHelper && fetchHelper.then(data => {
    setStorageItem(key, data) //设置缓存
    return data
  }).then(data => callback && callback(null, data)).catch(err => callback(err))
}
```



数据缓存优化



jsbundle预缓存优化

1、在现有的 weex 页面配置文件 的基础上增加一个字段 preload， 当此字段值为1时候， 对jsbundle文件进行预缓存。

```
[  
  game_index:{  
    is_weex: 1,  
    weex: 'http://cdn.egame.qq.com/game-weex/weex/game-index/app.js?v=518',  
    web: 'http://cdn.egame.qq.com/game-weex/page/game-index.html',  
    preload:1  
  }  
]
```

2、提供js api :

biz/preload 提供给前端进行对下一个页面的预加载。

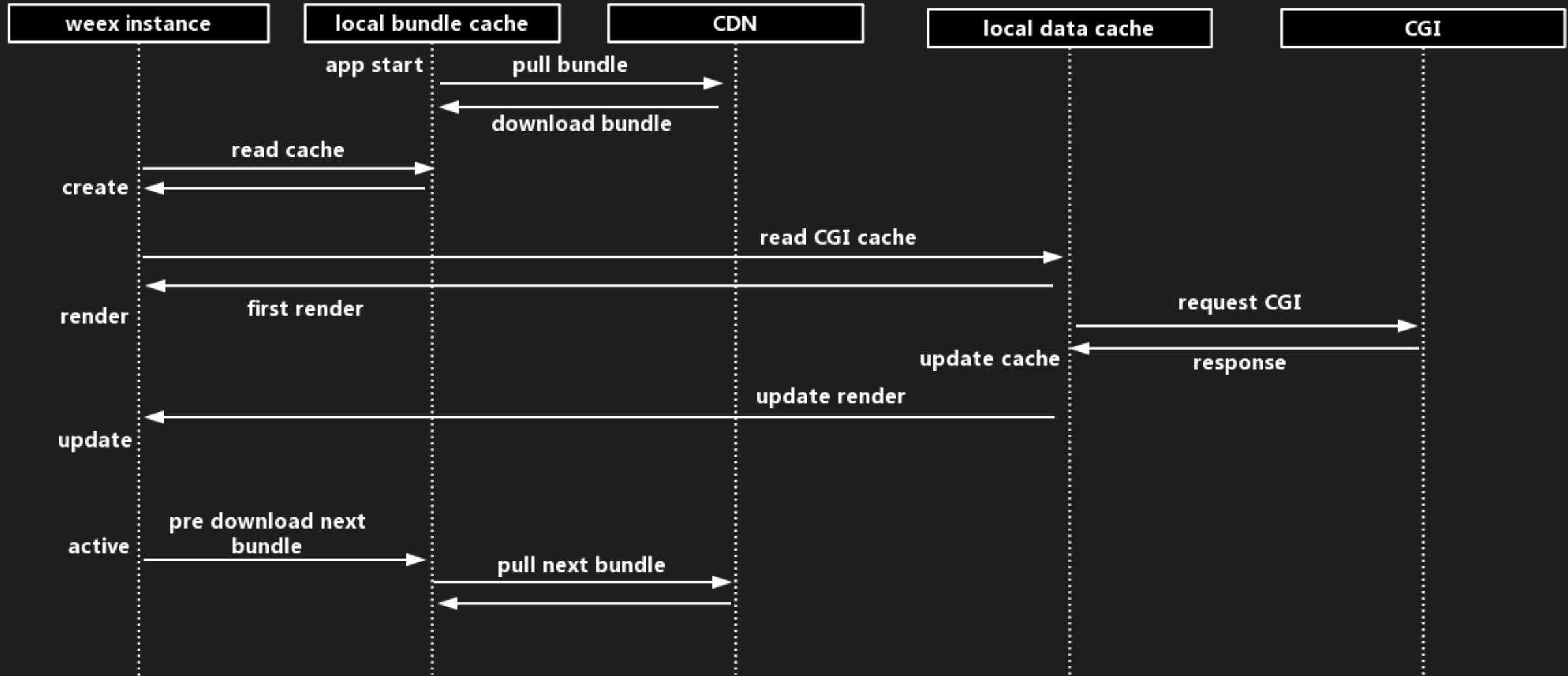
biz/clearProload 用于前端对于既有缓存内容的清空

3、对于缓存内容， 使用LRU的算法进行管理， 控制缓存总大小为20M

4、对于使用缓存的内容， 终端在ua上添加一个字段 PRELOAD/1 ; 1表示是预缓存内容， 0表示不是。



jsbundle预缓存优化



耗时打点统计

- 拦截请求开始、请求完成、RenderFinish计算网络耗时（请求结束 - 请求开始）、终端渲染耗时（RenderFinish - 请求结束）
- 终端通过js api提供前端创建instnace时间点
- 终端、前端两份上报，相互校验数据

统计日期	页面	手Q版本	上报量	客户端耗时部分		页面耗时部分		宏观指标				笔记
				WV响应	WV创建	首屏可见	可交互	webview耗时	网络耗时	页面耗时	耗时总计	
2018-01-09 (周二)	游戏社区首页(weex)	3.5.0.226	74	44	13	132	209	56	334	341	731	
2018-01-10 (周三)	游戏社区首页(weex)	3.5.0.226	2,310	39	13	120	225	52	335	345	731	
2018-01-11 (周四)	游戏社区首页(weex)	3.5.0.226	627	41	12	122	226	53	341	348	742	

统计日期	版本	上报次数	isX5	page	总耗时平均值	阶段耗时平均值	关键点耗时平均值	终端耗时平均值	state0	state1	state2	state3	state4	state5	state6	state7
2018-01-10 (周三)	3.5.0.226_430	2,329	weex缓存	game_index	610.25	610.25	566.45	50.24	30.95	5.77	13.52	559.57	0.45	54.39	460.44	9.25



终端接口调用耗时优化

- 1、Android 升级weex 0.16版本
module接口调用耗时大幅减少。
- 2、减少启动页面时并发调用
module接口
- 3、Android 调用时多次反射获取类
名导致Dom渲染耗时过长。

版本	初次调用耗时	二次调用	初次回调耗时	二次回调耗时
0.12.0	50–60	30–40	10	10
0.16.0	5–6	2–3	2	2

```
public static Runnable secure(Runnable runnable) {  
    if(runnable == null || runnable instanceof SafeRunnable) {  
        return runnable;  
    }  
    try {  
        String className = runnable.getClass().getCanonicalName();  
        if (className != null && className.startsWith(SYSTEM_ACTION_PREFIX)) {  
            return new SafeRunnable(runnable);  
        }  
    } catch (Exception e) {  
        Log.e("SafeRunnable", "Error getting canonical name for " +  
                "Runnable: " + e.getMessage());  
    }  
}
```



Android Bitmap 内存优化

版本	crash率	主要crash
3.1.0	0.13%	系统crash, 业务逻辑错误等
3.2.0	1.01%	OOM、SIGSEGV(SEGV_MAPERR)

使用Fresco管线加载图片，加载后不持有图片引用。

问题：

- a、会将用户当前不展示的图片也加载到内存中，实际占用内存变多，OOM问题严重。
- b、引用不被持有会导致Bitmap被回收后仍然被使用（3.2 – 3.4版本top crash—SIGSEGV (SEGV_MAPERR)）

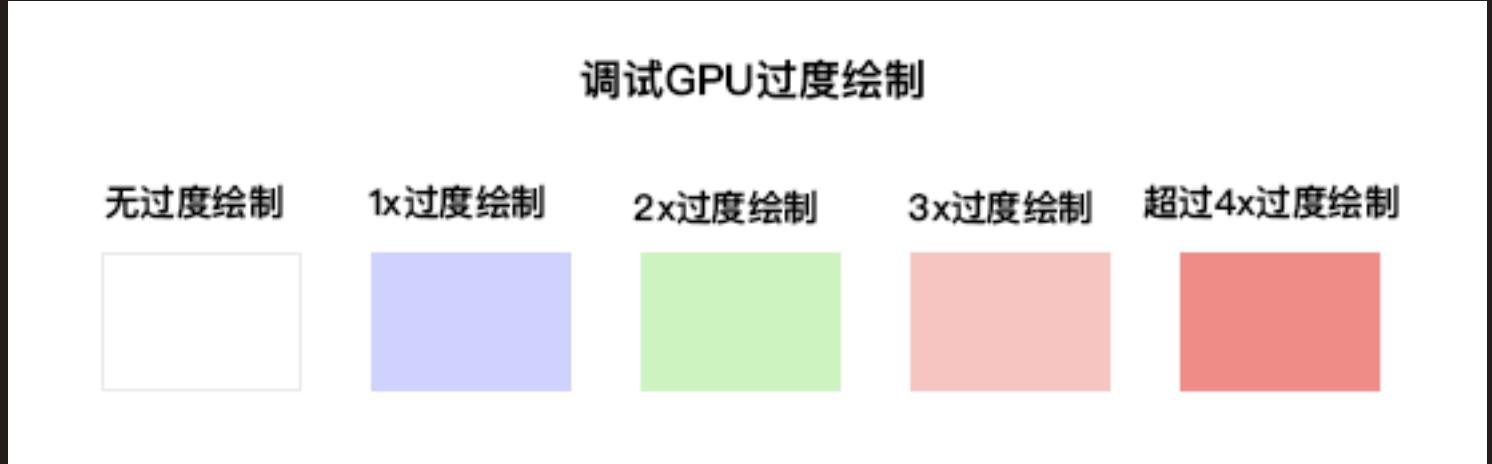
优化：

仿照Fresco的DraweeController管理WXImageView的bitmap引用。
显示时加载，隐藏时解除引用等待回收。

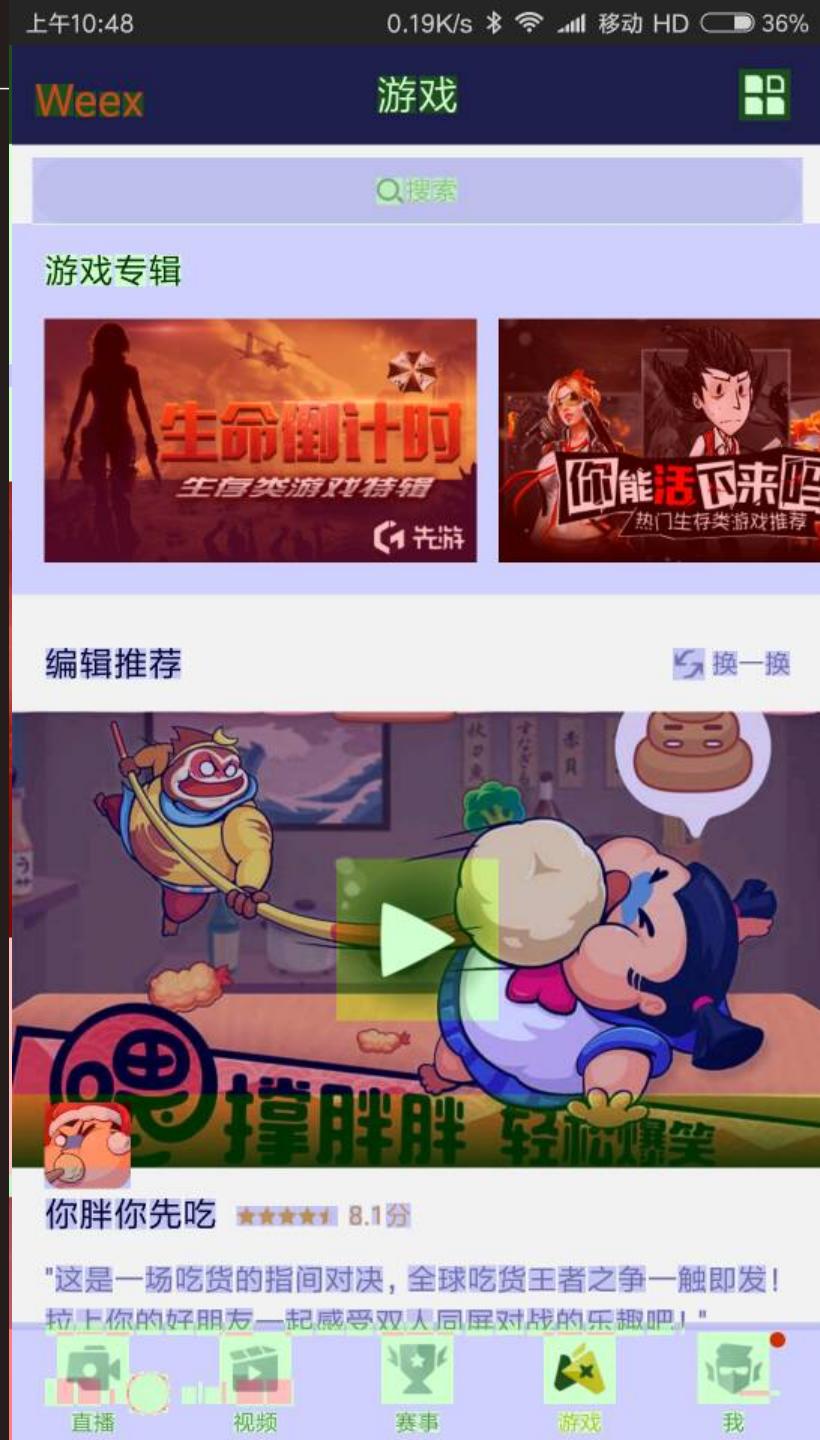
OOM问题明显缓解，bitmap被回收的crash问题也得到解决。



GPU过度绘制优化



- 1、尽量不要设置背景色
- 2、不要过度嵌套，结构尽量扁平化



Weex实践分享——内部影响力



企鹅电竞weex实践——从入门到吃鸡

donaldcen 2017年11月23日 10:49 浏览(1394) ❤️ 已收藏(84) 💬 评论(60) 分享



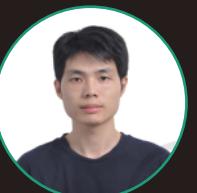
企鹅电竞weex实践——接入调试篇

marvinqi 2017年11月22日 12:12 浏览(731) ❤️ 已收藏(63) 💬 评论(34) 分享



企鹅电竞weex实践——框架原理剖析（上）

homkerliu 2017年12月07日 12:56 浏览(427) ❤️ 已收藏(44) 💬 评论(31) 分享



企鹅电竞weex实践——UI开发篇

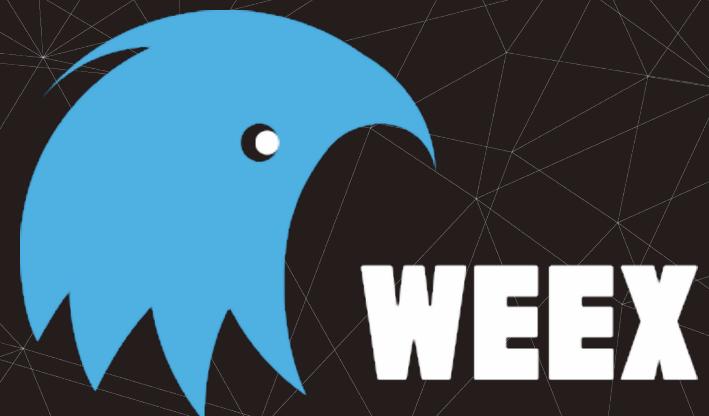
erickchen 2017年12月12日 10:18 浏览(505) ❤️ 已收藏(46) 💬 评论(28) 分享



希望优化的问题

- 横竖屏切换方案支持
- 支持armabi-v7a的so包
- 组件统一开源：例如 jscore内核、 debugger、 gcanvas
- 更好的性能细分统计
- 高效的终端和Weex共享动态数据方案
- 更好的容错： iOS JS call Native时，参数传错会造成终端crash





<https://weex.apache.org>