Hybrid App之H5体验优化

王利华
@vczero

自我介绍

- 王利华，@vczero，高德、携程、宝宝树电商
- 《React Native入门与实战》作者
- Web Developer、专注H5框架和架构设计
- https://github.com/vczero

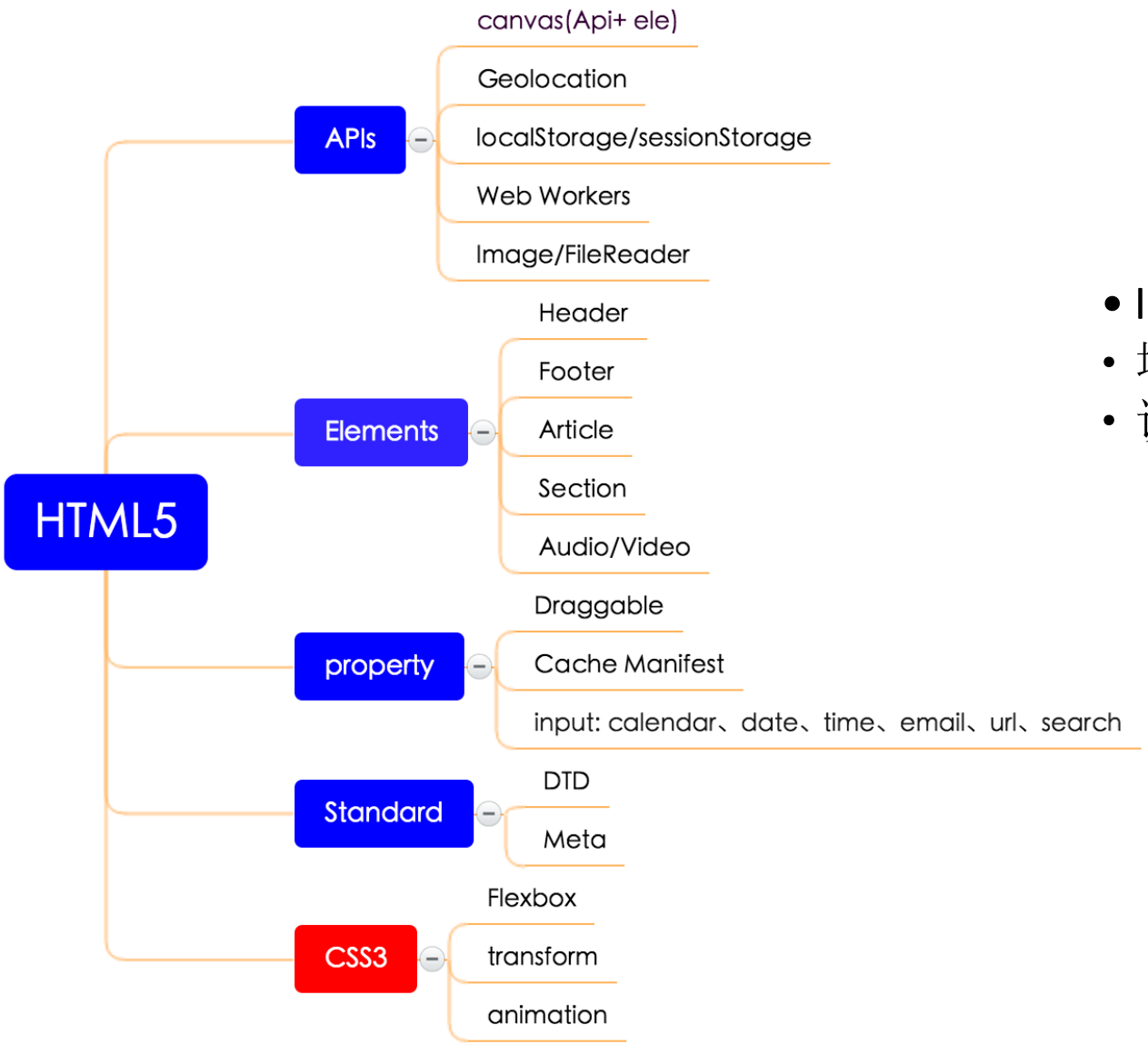
H5该如何定义

百科定义

HTML5草案的前身名为 Web Applications 1.0，于2004年被WHATWG提出，于2007年被W3C接纳，并成立了新的 HTML 工作团队。

HTML 5 的第一份正式草案已于2008年1月22日公布。HTML5 仍处于完善之中。然而，大部分现代浏览器已经具备了某些 HTML5 支持。

HTML5标准范畴



- In Browser Tech
- 增强功能
- 设定标准

这就是H5?

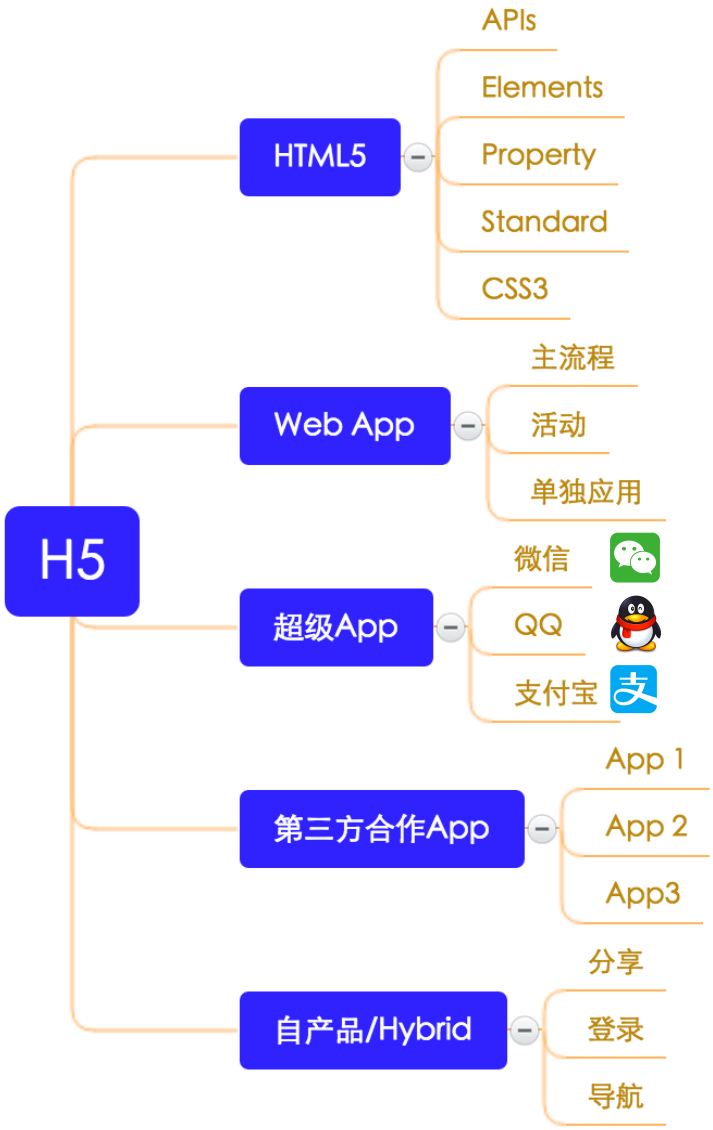
H5工程师的定位

“运营活动”工程师？

H5工程师在运营团队里面？

Too young, too naive

任何定义应该由工程来诠释

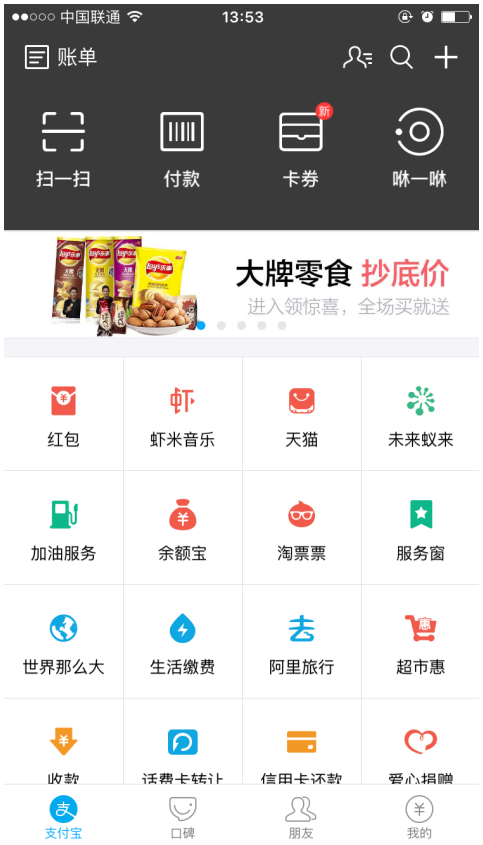


in App Container Tech

H5? HTML5 in App Technology：HTML5

H5能做什么

H5 Example


●●○○○ 中国联通 13:53
账单
扫一扫
付款
卡券
咻一咻
大牌零食 抄底价
进入领惊喜，全场买就送
红包
虾米音乐
天猫
未来蚁来
加油服务
余额宝
淘票票
服务窗
世界那么大
生活缴费
阿里旅行
超市惠
支付宝
口碑
朋友
我的


●●●○○ 中国联通 13:57
我
钱包
微信安全支付
付款
零钱
¥23.80
银行卡
腾讯服务
转账
手机充值
理财通
Q币充值
生活缴费
城市服务
信用卡还款
微信红包
AA收款
腾讯公益
第三方服务


●●●○○ 中国联通 14:00
奶粉
纸尿裤
奶粉
孕妈
婴童
童装
树·生活
热门品牌
强生
贝亲
新安怡
妙思乐
新贝
五羊
费雪
贝恩施
chbaby
嘉宝
如爱初见
良品铺子
热销榜单
首页
分类
全球购
购物车
我的订单

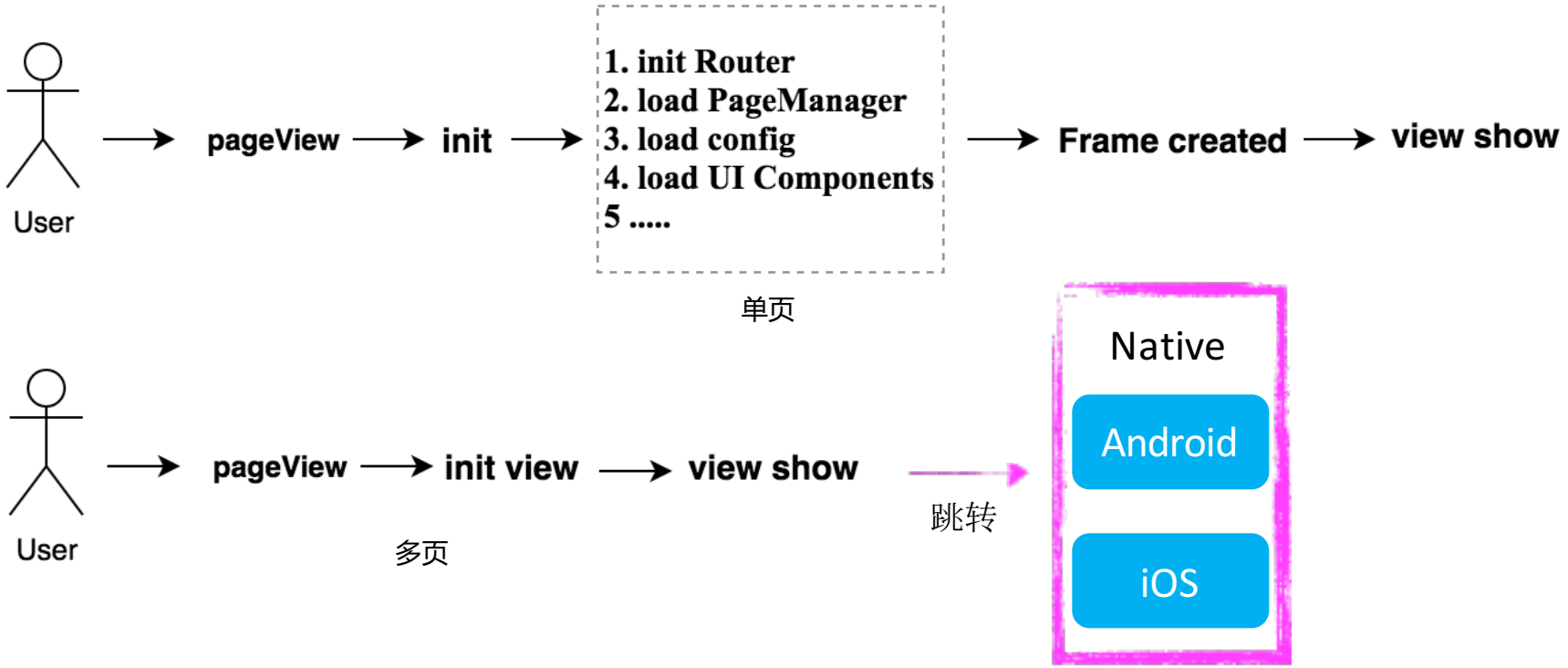
Betting on HTML5 Was a Mistake

暴露的问题

1. 没有相应的工具对内存进行跟踪
2. 网页长滚屏效果太差
3. 更加平滑的动画处理
4. 对缓存提出了更高的要求
5. 对及时响应提出了更高的要求
6. 返回页面状态的控制（单页/多页）
7. 手势支持不佳

H5 in App优化

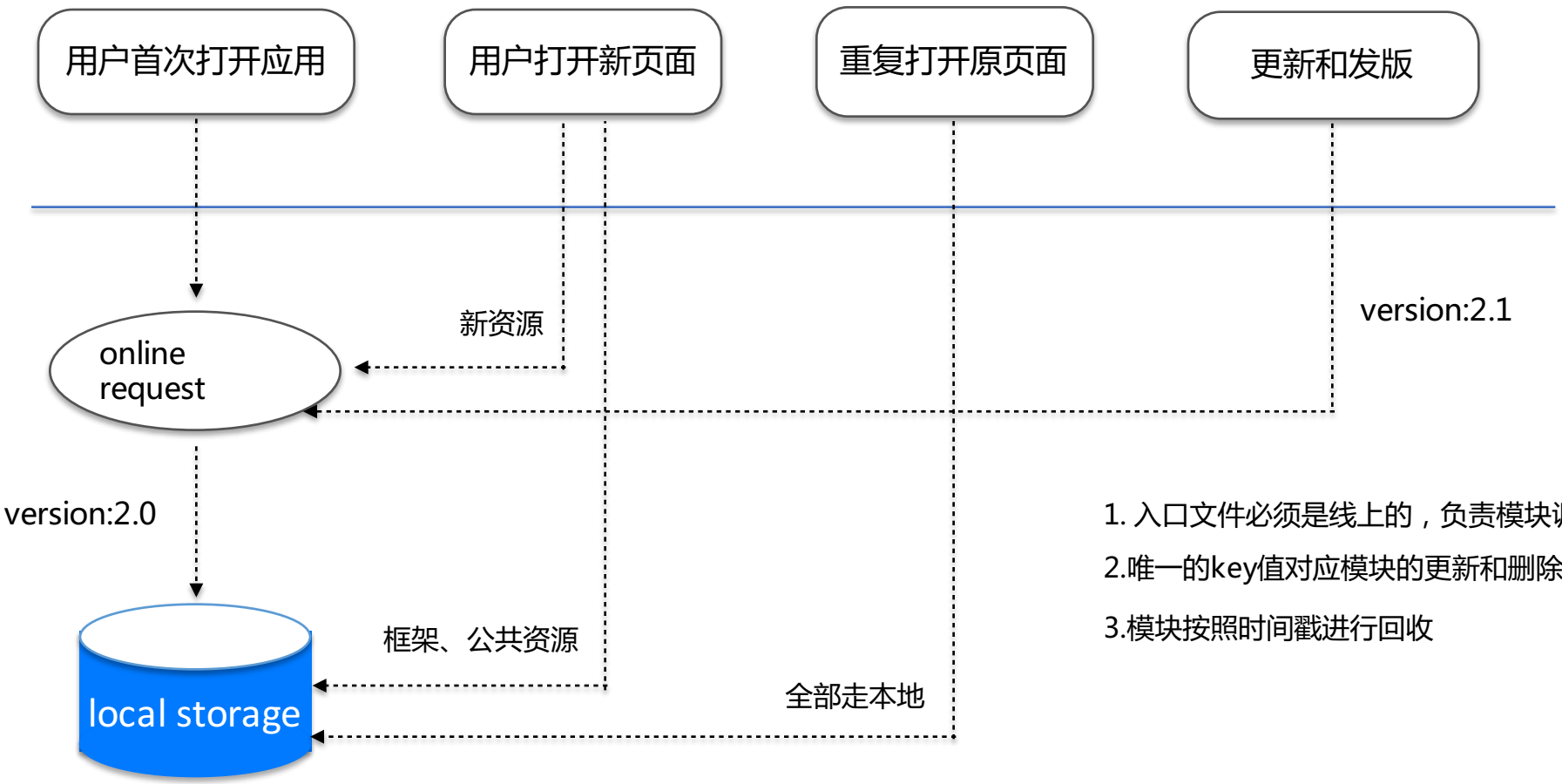
1 模式选择：单页 vs 多页



单页适合Web App

多页适合in App，作为App的某个功能组成部分

2 localStorage 本地存储



1. 入口文件必须是线上的，负责模块调度
2.唯一的key值对应模块的更新和删除
3.模块按照时间戳进行回收

口碑奶粉
环球好品质
GO >

纸尿裤馆
清爽好透气
GO >

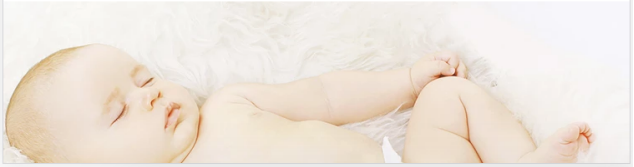
宝宝用品
温柔好呵护
GO >

辅食零食
健康好营养
GO >

美妆个护
美丽好气色
GO >

营养保健
塑造好身体
GO >

环球精选



index.js?v=1465202613773	200	xhr	fast.js:1	2.8KB	26ms
zepto.js?v=1465202613773	200	xhr	fast.js:1	12.6KB	44ms
page.js?v=1465202613773	200	xhr	fast.js:1	906B	35ms
ajax.js?v=1465202613773	200	xhr	fast.js:1	994B	36ms
loading.js?v=1465202613773	200	xhr	fast.js:1	767B	34ms
storage.js?v=1465202613773	200	xhr	fast.js:1	892B	33ms
swipe.js?v=1465202613773	200	xhr	fast.js:1	2.8KB	48ms
imagelazyload.js?v=1465202613773	200	xhr	fast.js:1	1.2KB	50ms
ua.js?v=1465202613773	200	xhr	fast.js:1	1.2KB	50ms
utils.js?v=1465202613773	200	xhr	fast.js:1	921B	51ms
index.js?v=1465202613773	200	xhr	fast.js:1	1.2KB	51ms
toast.js?v=1465202613773	200	xhr	fast.js:1	1.1KB	60ms
class.js?v=1465202613773	200	xhr	fast.js:1	994B	31ms
viewport.tpl.js?v=1465202613773	200	xhr	fast.js:1	2.8KB	31ms
loading.tpl.js?v=1465202613773	200	xhr	fast.js:1	1.0KB	27ms
template.js?v=1465202613773	200	xhr	fast.js:1	1.1KB	19ms
fast.js	200	script	index.html:10	1.5KB	24ms
require.js	200	script	fast.js:1	6.7KB	18ms
config.js	200	script	require.js:1	1.0KB	16ms

3 / 20 requests | 9.3KB / 320KB transferred | Finish: 758ms | DOMContentLoaded: 131ms | Load: 150ms

9.3 KB 150ms

3 减少DOM深度，避免构建复杂的DOM结构

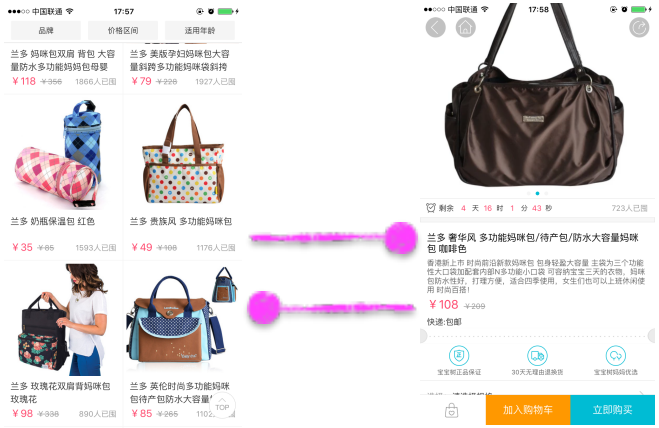
sessionStorage临时数据存储方案

需求

列表页不断下拉刷新，然后点击进入详情页，详情页回退到列表页

方案

1. sessionStorage存储列表数据和页码
2. 详情页返回时，滚动到页面底端



优势

1. 数据请求走缓存，减少网络304 / 200 OK (from cache)
2. 可以回到临时状态

建议

1. 注意存储的临界值，尽管存储空间大约5M，但是有些WebView和浏览器不标准
2. 在setItem时try catch，捕获quota的异常，剔除过期的数据
3. 存储的JSON对象统一encodeURIComponent编码

丝滑　　精致

动画处理之CSS和JavaScript的应用场景

```
div{
  transition: color 250ms;
}
div:hover{
  color: red;
}
```

```
$dom
  .on('mouseover', function(){
    $(this).animate({color: 'red'}, 250);
  })
  .on('mouseout', function(){
    $(this).animate({color: '#fff'}, 250);
  });
```

JavaScript费力的做了CSS该做的事，没有额外好处

```
var aniStyle = {
  fadeIn:{/*fadeIn的效果*/},
  fadeOut:{/*fadeOut的效果*/},
  back:{/*回到初始位置*/},
  width:{/*宽度变化*/}
};
$dom.animate(aniStyle.fadeIn, 30).animate(aniStyle.back, 10).animate(aniStyle.width, 60);
```

JavaScript可以处理复杂的效果，做到效果和逻辑分离

动画处理之CSS和JavaScript的应用场景

- 简单的动画交给CSS，例如：微交互，鼠标悬停效果；
- 复杂的动画逻辑交给JavaScript，做到样式和逻辑分离；
- CSS的关键帧可以很好的区分时间段的逻辑；
- 单一元素的效果应该由CSS完成；
- 其他的都交由JavaScript吧！！！

1. 减少在循环动画中的reflow、layout计算

平滑动画(min value) = 60 帧/秒 = 16.67 ms * 60 = 确保引起layout因子很少

2. 样式的获取和设置批量化

```
var ele = $('#container');
var top = ele.css('top');
var color = ele.css('color');
ele.css('top', 300);
ele.css('color', 'black');
```

```
var top = $('#container').css( 'top' );
$('#container').css('top', 300);
var color = $('#container').css('color');
$('#container').css('color', 'black');
```

3.批量构建内存DOM片段

```
var eles = [...];
  for(var i in eles){
    $(ele).appendTo($container);
  }
```

```
var eles = [...];
  var htmlStr = '';
  eles.each(function(i, ele){
    //TODO:其他操作
    htmlStr += ele;
  });
  $(htmlStr).appendTo($container);
```

4. 提高动画层级，巧妙使用transform属性

transform属性可以将目标元素独立一个层级，通过GPU加速，提升比较明显(show case)

5. 减少不必要的持续滚动、throttle & debounce

```
$(window).scroll(function(){
  //滚动的监测逻辑
  //用户滚动,持续响应
});

_.debounce = function(func, wait, immediate) {……}
```

6. 一些复杂的动画，可交由canvas实现

代表：egret Engine、AMap JavaScript API

动画测试：针对Android4.0+、iOS7测试性能的最低限制

优化，
关乎架构，

也，
关乎细节

H5 & Hybrid分工

H5 & Hybrid分工（一）

1. 上传功能(客户端，阻塞动作)

   uploadSize > 5M ? Native Component : H5 FileReader

2. 图片压缩和裁剪

   uploadSize > 4M ? Native Component : H5 canvas.toDataURL

3. Big List View(列表长度超过20000px)

   listViewHeight > 20000px ? Native Component : H5 List(优化/舍弃监听onScroll)

4. 主流程

   多页的情况，App容器内采用Native NavigatorBar控制页面的跳转，如微信

   单页的情况，App容器内采用Native提供API形式在H5端控制，例如h5.goBack()

5. H5容器的构建

Native同学做好H5容器，例如UIWebView & WKWebView
JSBridge & JavaScript Core

6. 单个页面混排

场景：列表 & 介绍 / 推荐

建议：Native的View在H5页面上方，否则容易出现闪烁

7. 手势的冲突

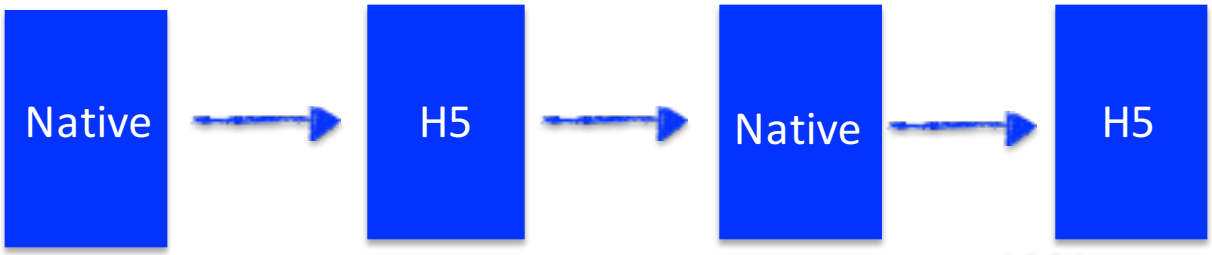
WebView内手势由H5控制；整个H5页面的流程手势由Native控制

8. 页面下拉刷新

Native负责整个页面的刷新更加高效；H5可以做局部刷新

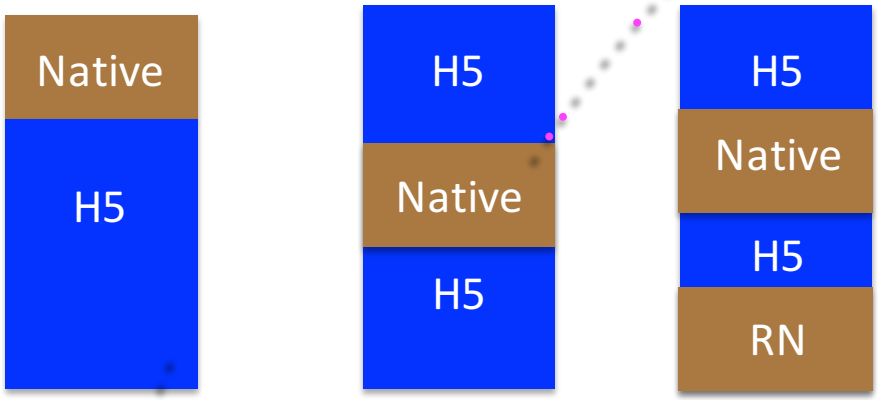
H5 & Hybrid分工总结

H5负责流程性的页面



Native负责高性能要求的功能

例如：分享、大文件上传



厕所在哪(iOS Native+RN+H5)

例如经常变的活动规则，全球购的规则信息

H5的未来

1. H5的分享、轻量、动态、快速开发是特别明显的优势

2. 混合可以是多融合：Native & Web & 类React Native

3. 超级App内嵌类React Native / Weex引擎

4. H5会越来越强大，越来越健壮；国内H5环境因为超级App的存在而更好

GMTC 全球移动技术大会
GLOBAL MOBILE TECH CONFERENCE

THANKS

聚焦前沿技术　传递实践经验

主办方 Geekbang. 极客邦科技　InfoQ