



Yunishi 石玉磊

HTML5在Qzone的实践



About yuni





HTML5是什么？

HTML



One Web **W3C** *for All*





广义上





JavaScript new API



OFFLINE



STORAGE



CONNECTIVITY



FILE ACCESS



3D/GRAPHICS



PERFORMANCE



NUTS & BOLTS





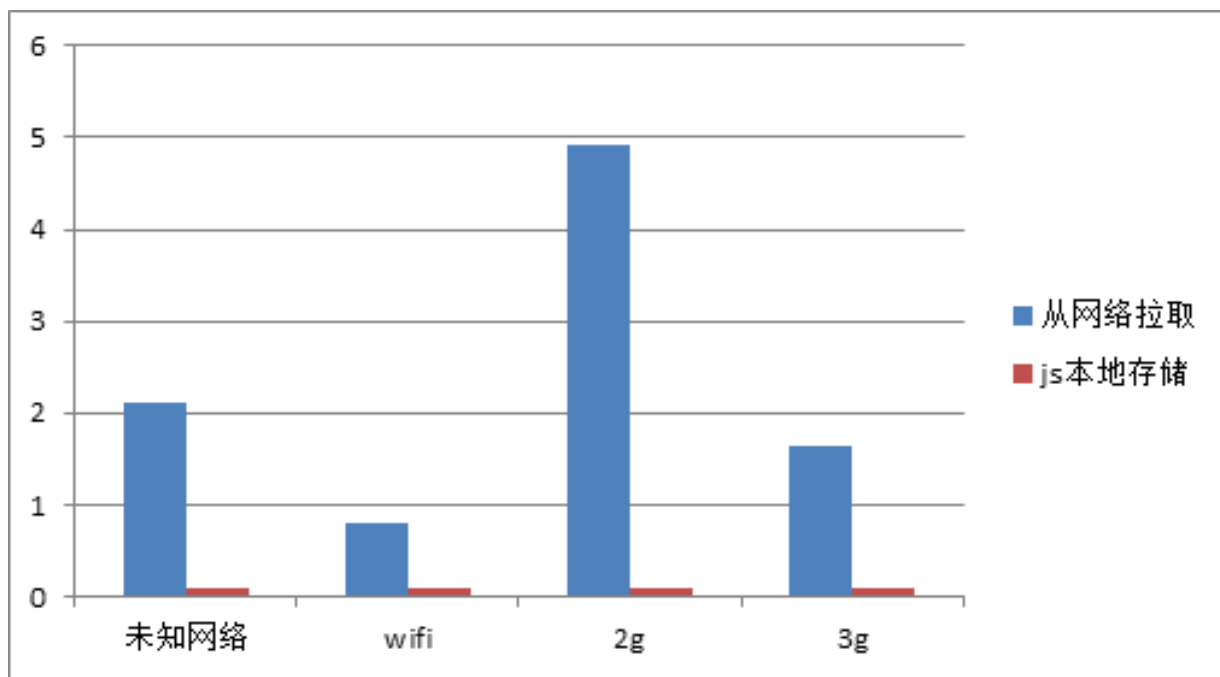
本地存储

- 静态资源
- 动态数据



资源本地存储

- 当用户再次访问页面的时候读取本地cache js
- 相对于从网络拉取，本地读取的时间可以忽略





资源本地存储

■ 原理





资源本地存储

■ 优点

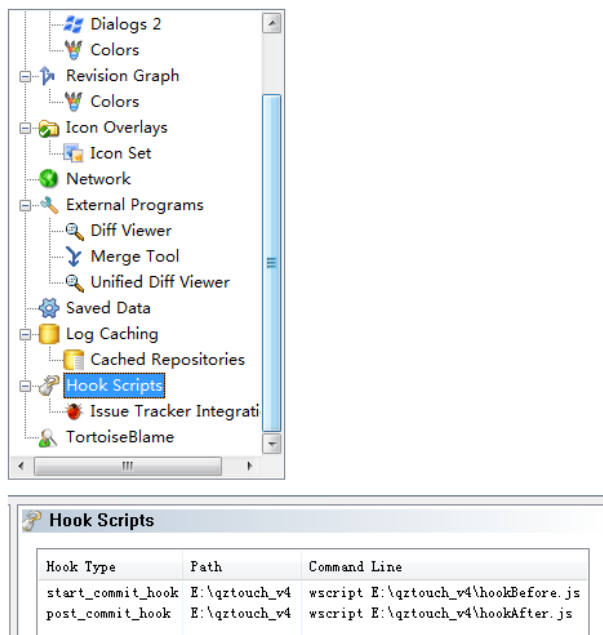
- ✓ 相比于html5官方的manifest cache：
 - 支持js单文件更新，更新后本次立即生效；
 - 支持动态直出页配置manifest cache但是不cache动态页本身；
 - 支持combo文件的cache；且下次combo请求自动排除本地cache已有的文件
- ✓ 在触屏项目实践中，项目针对对plugin storage插件做了以下优化：
 - 在不支持localstorage的平台，manifest.js配置文件直接应用为cache文件的版本号控制；
 - Js本地存、取前加合法性验证



资源本地存储

■ 自动化版本号管理

- ✓ 通过svn钩子实现前端js文件的版本号管理，配置该svn hook后，每次提交js文件都会自动更新manifest.js配置文件





http 304

■ conditional request

✓ Cache-control

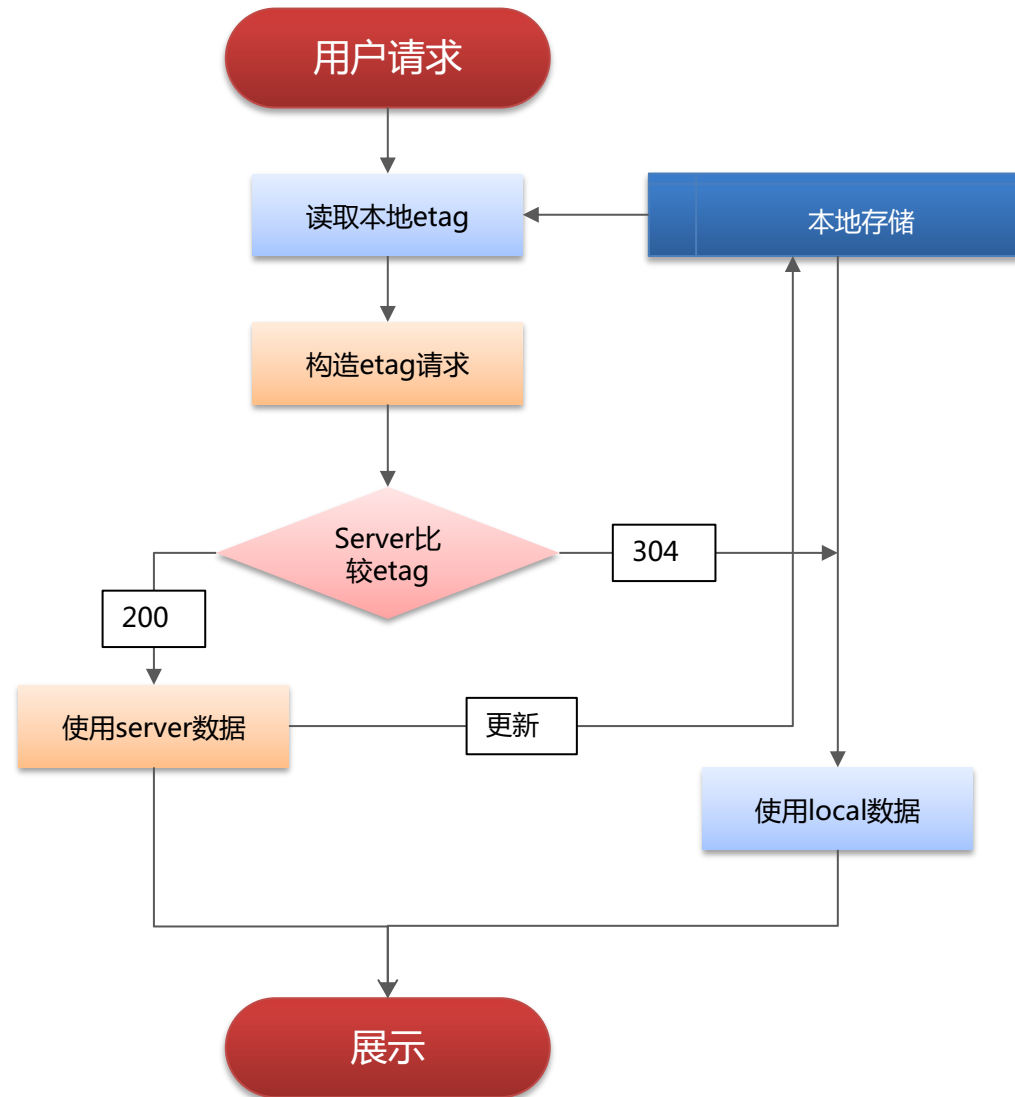
- If-Modified-Since

✓ Etag

- If-None-Match



Etag





Etag

■ Etag & 本地存储

同比大概节约了**40%**的流量



但是...

- 性能的提升不会显著增加用户，要倾听用户的心声，满足他们更多的需求

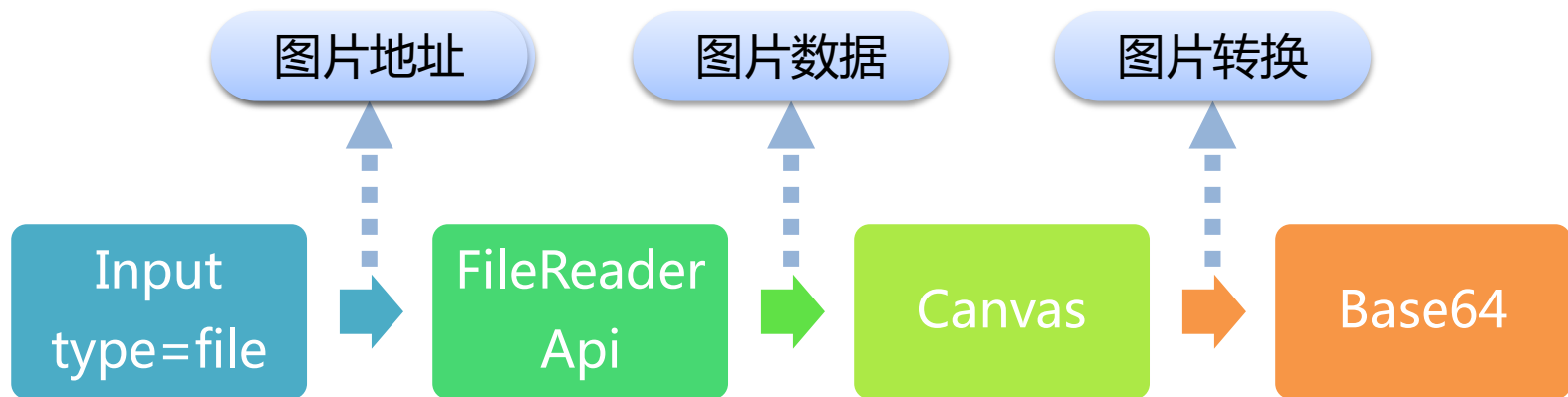
- ✓ 拍照上传



拍照压缩上传

■ 基本原理

- ✓ 通过input type=file选择本地图片
- ✓ 通过FileReader Api获取本地图片数据
- ✓ 将本地大尺寸图片渲染到尺寸更小的canvas
- ✓ 通过canvas生成被缩放后的小图的base64字符串
- ✓ base64字符串可以用来本地预览和ajax上传








图片本地压缩

■ api支持情况

- ✓ Input type=file & FileReader
- ✓ canvas

	Input type=file	FileReader
 Android QQBrowser 3.X	✓	×
 Android QQBrowser 4.X	✓	✓
 Android UcWeb	✓	✓

iOS Safari	Android Browser
	2.1
3.2	2.2
4.0-4.1	2.3
4.2-4.3	3.0
5.0-5.1	4.0
6.0	4.1

表1. FileReader

iOS Safari	Android Browser
	2.1
3.2	2.2
4.0-4.1	2.3
4.2-4.3	3.0
5.0-5.1	4.0
6.0	4.1

表2. canvas



图片本地压缩

■ iOS平台bug

✓ Subsample

- [官方文档](#)描述

- 大于2M的图片读到浏览器里的时候会做subsample处理
- 最大可以处理的jpg图片为32M
- 其他类型图片256M内存机器最大可以处理3M图片；大于256M内存可以处理5M图片

✓ 大图高度被压缩bug

- 图片高度只有原来的1/4

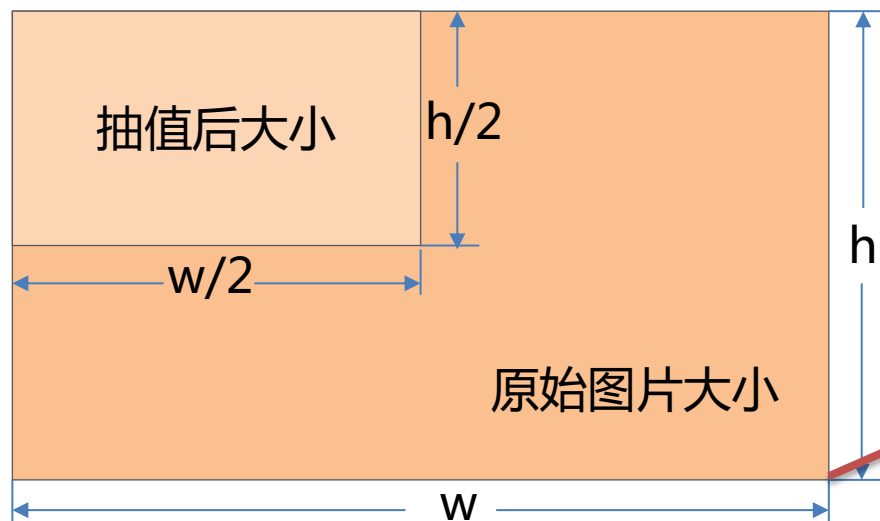
图片本地压缩

■ iOS平台实战

✓ Subsample

- 对大于1024x1024的图片检测是否有被抽值
- 检测抽值原理：取图片右下角的1x1px像素的aRGB数组判断alpha值

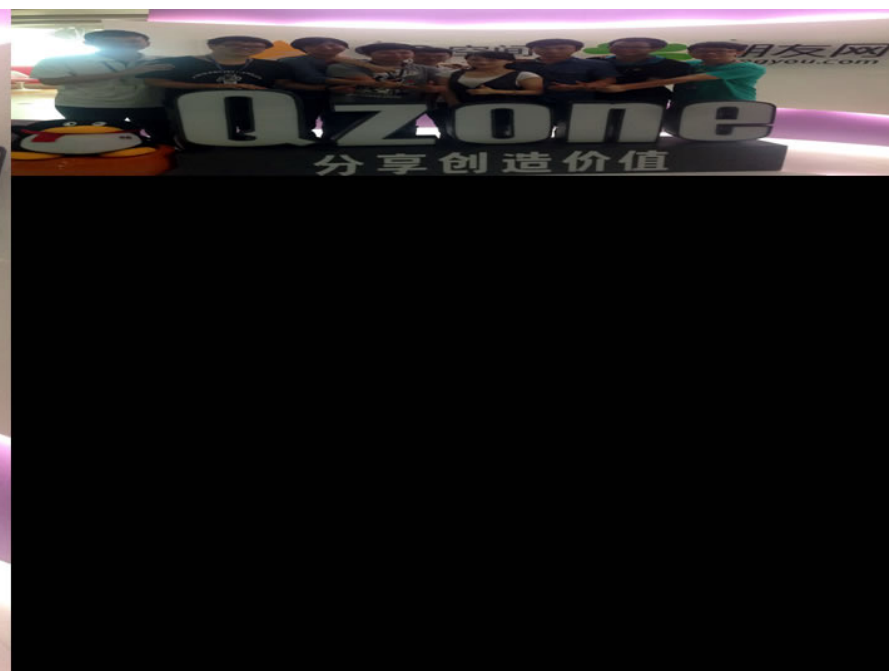
被抽值的
图片按1 : 2
(经验值)
还原



判断该点的
alpha值



图片本地压缩





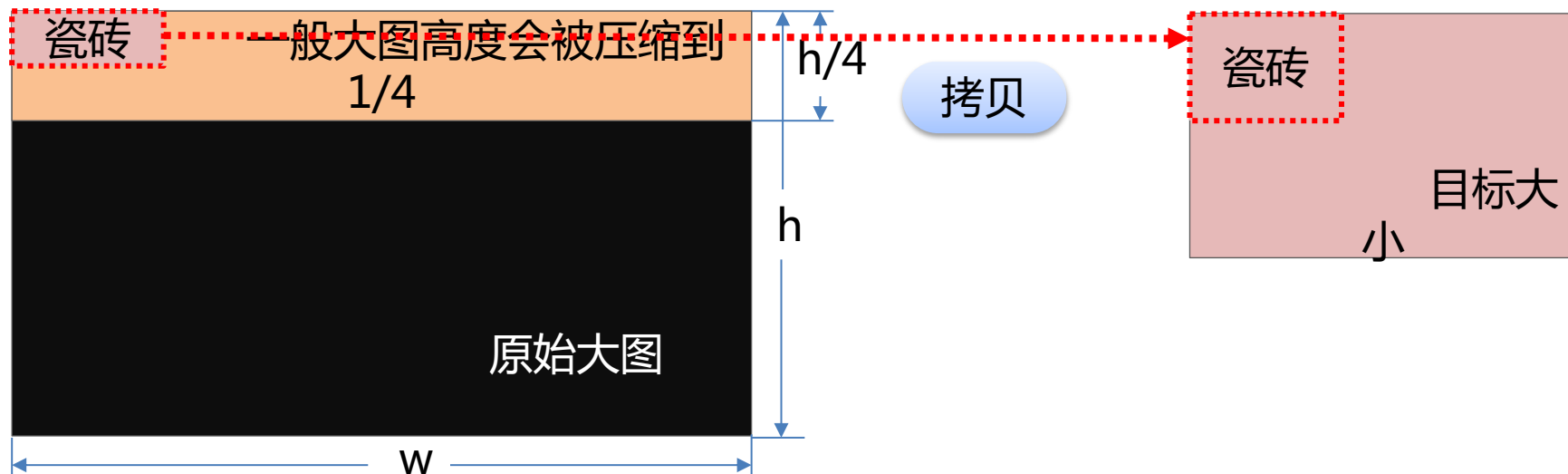
图片本地压缩

■ iOS平台实战

PUBLIC  stomita / ios-imagefile-megapixel

✓ 高度被压缩bug

- 计算图片高度被压缩比例，通过贴瓷砖的方法用固定大小的小canvas去分片读取大图到小canvas，拷贝过程中计算压缩比





图片本地压缩

■ Android平台`canvas.toDataURL()`输出图片格式限制

- ✓ w3c标准：image/png；浏览器可选择实现其他格式
- ✓ Ios支持image/jpg格式输出，可调整压缩质量
 - `canvas.toDataURL('image/jpeg' , 0.8)`
- ✓ android只支持默认格式☹

Jpg图片大小只有Png格式的约**1/3**



图片本地压缩

■ Android平台实战

- ✓ 借助第三方工具库[jpegEncoder](#)
- ✓ 将canvas的argb颜色数组转化为为压缩比更高的jpg格式，同时支持设置压缩质量





图片本地压缩

■ 其实，还可以做的更好！

- ✓ 借助第三方工具库 [JpegMeta](#)
- ✓ 读取图片meta数据里照片拍摄方向后自动旋转图片



jsjpegmeta

Javascript library for parsing Jpeg file meta data

```
> jpg.tiff
▼ MetaGroup
  ▶ DateTime: MetaProp
  ▶ ExifIfdPointer: MetaProp
  ▶ Make: MetaProp
  ▶ Model: MetaProp
  ▼ Orientation: MetaProp
    description: "Orientation of image"
    fieldName: "Orientation"
    value: 1
```



图片本地压缩

■你担心性能问题吗？

小米2A整个过程约**1s**



图片本地压缩

■ 推荐压缩方案

✓ 根据业务特点

- 图片压缩到800x800以内
- 压缩质量0.8
- Android下2G网络压缩质量调整为0.5

一般2M的图片可以压缩到**150k**左右

适合移动网络下传输



百万上传量!

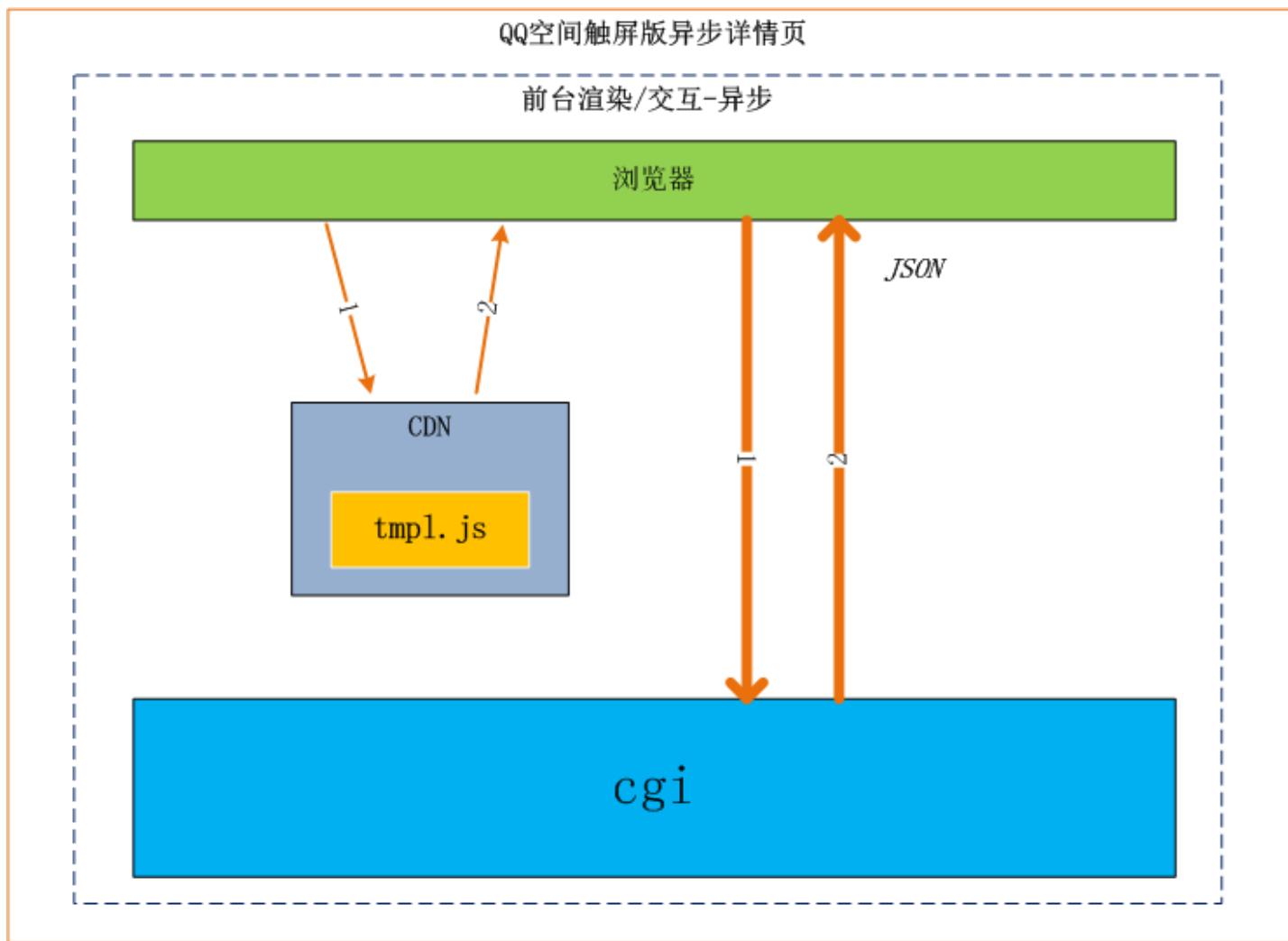
用户 很开心



nodejs

- 取代CGI ?
- 最小成本最大功效

动静分离





多普勒测速

■ Qzone touch多普勒测速数据

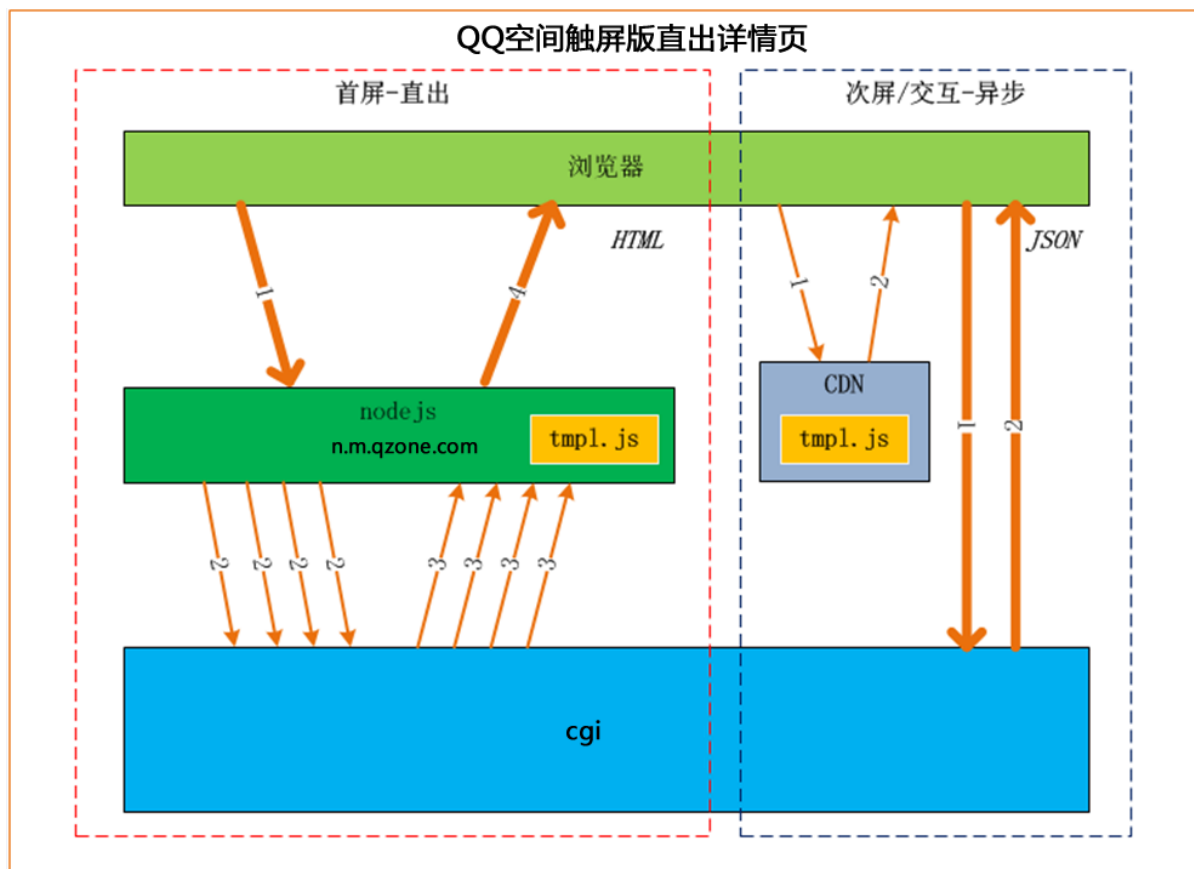
#	dns(s)	conn(s)	rtt(s)	tran(kb/s)
2g	3.85785	2.33482	2.57478	14.0374
3g	1.60643	0.743109	0.608047	60.1967
wifi	0.986921	0.550208	0.444332	70.8728



nodejs直出

■ 原理

- ✓ 基于nodejs
前后端公用
一套js模板
- ✓ 对前端更加
友好





nodejs直出

■ 容灾

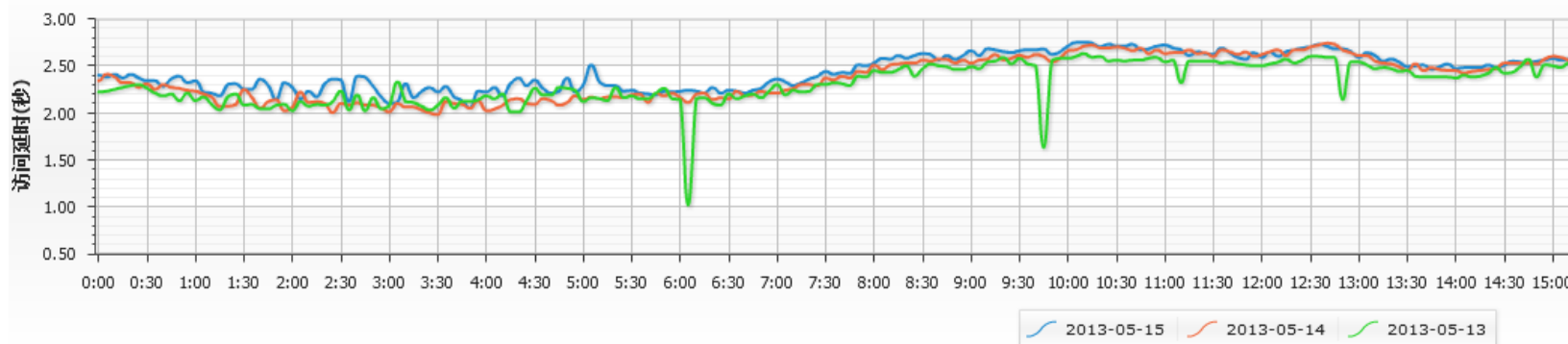
- ✓ 当nodejs请求非关键路径出错时，转化为异步渲染
 - 吐页面到浏览器，浏览器尝试异步再次请求
- ✓ 当nodejs请求关键路径出错时
 - 提示用户错误信息

nodejs直出

■ 测速

✓ 可查看时间点直出一般在2.5s；而异步渲染则大于5s

[[触屏版]] [domready] [中国]分钟访问延时趋势图



计算规则：下面数据是从288个5分钟点数据中求和再平均的结果。

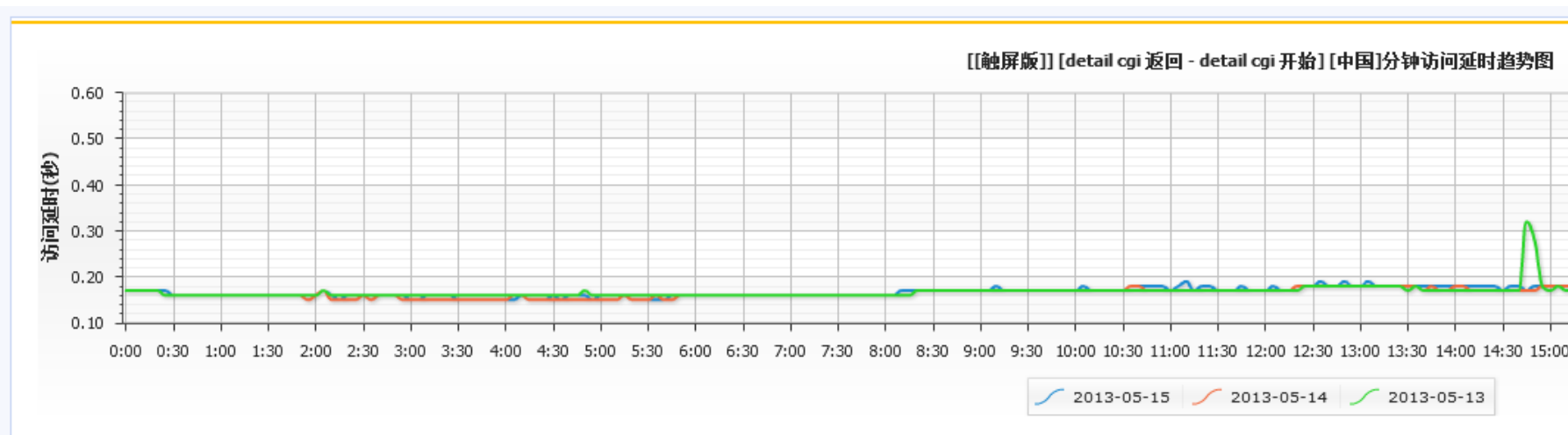
时间	访问次数	慢速用户比例	延时(秒)	对比一时间	访问次数	慢速用户比例	延时(秒)	变化比例
2013-05-15	20102508	11.93%	2.53	2013-05-14	19537741	11.71%	2.51	0.69%



nodejs直出

■ 测速

- ✓ Nodejs内网拉取后端cgi耗时 (180ms左右)



计算规则: 下面数据是从288个5分钟点数据中求和再平均的结果。

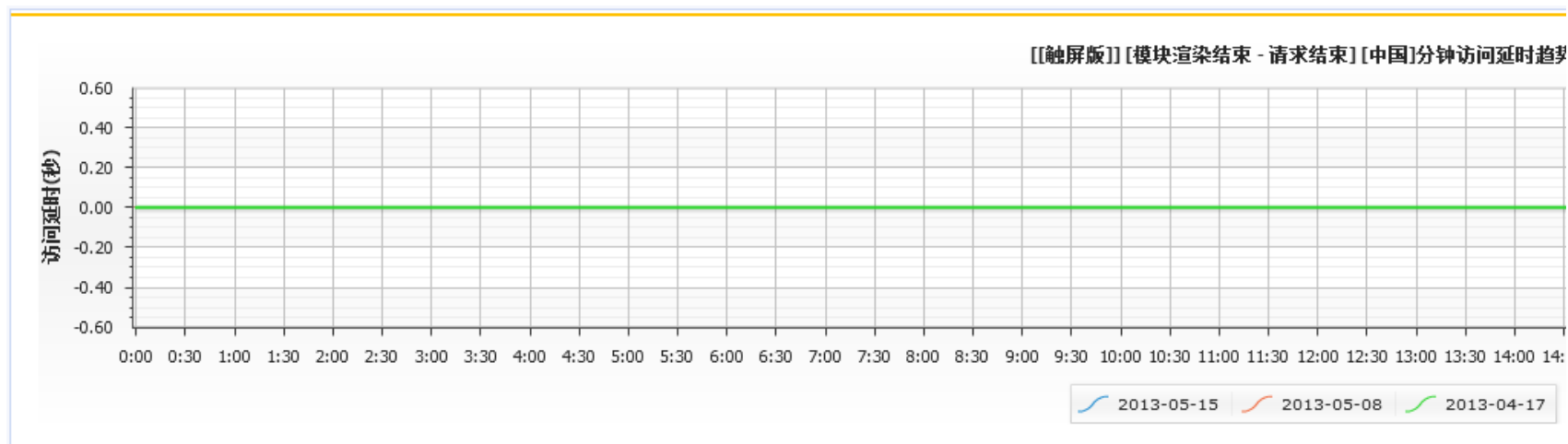
时间	访问次数	慢速用户比例	延时(秒)	对比一时间	访问次数	慢速用户比例	延时(秒)	变化比例
2013-05-15	7814380,20111645	-	0.18	2013-05-14	7656721,19548672	-	0.18	0.99%



nodejs直出

■ 测速

- ✓ Nodejs渲染模板耗时（可以忽略）



计算规则：下面数据是从288个5分钟点数据中求和再平均的结果。

时间	访问次数	慢速用户比例	延时(秒)	对比一时间	访问次数	慢速用户比例	延时(秒)
2013-05-15	20111645,20111646	-	0	2013-05-08	25400355,25400353	-	0



HTML5是



理想与现实

Expectation:



Reality:



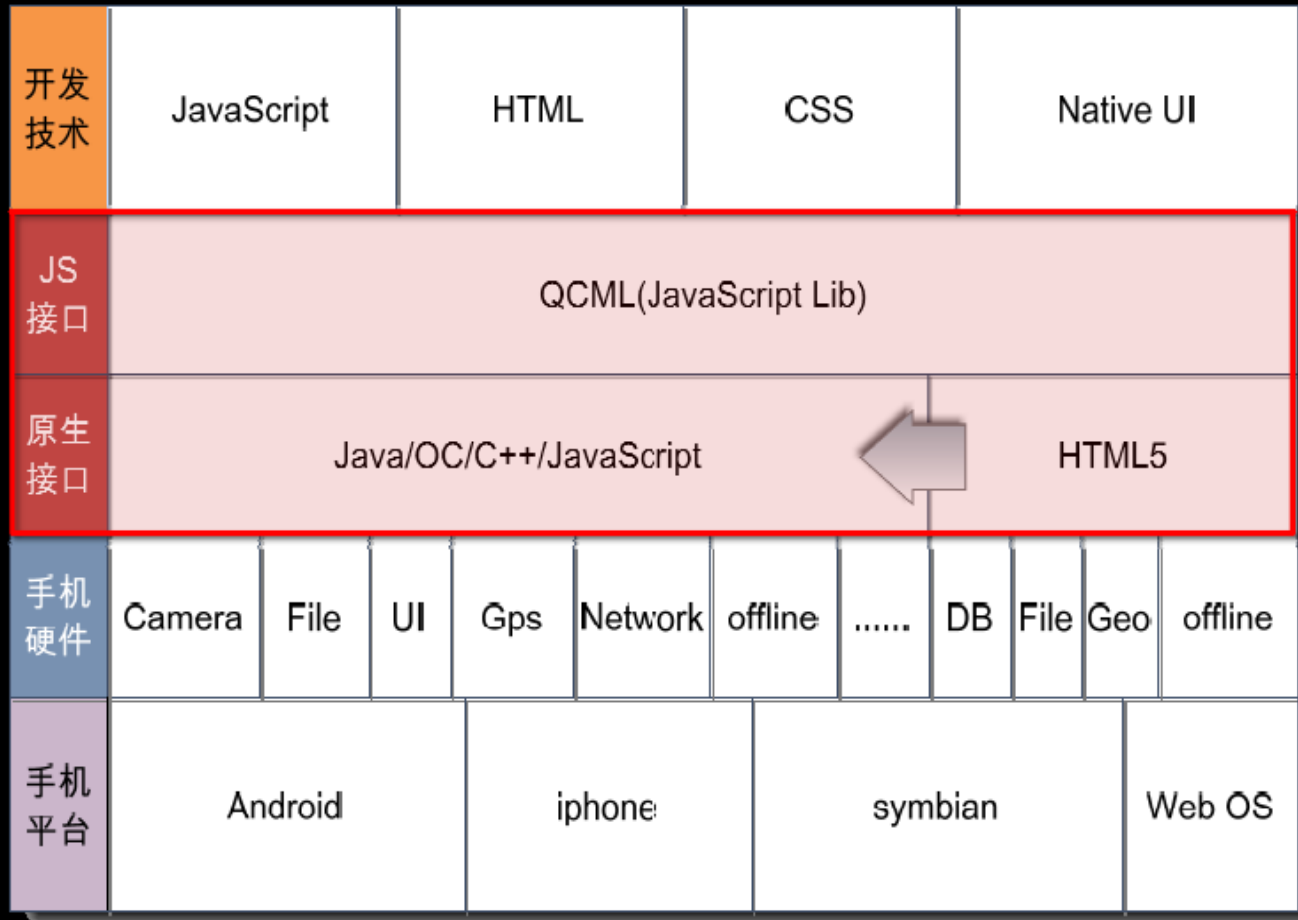


H5+

- Q+ (PC)
- Phonegap
- QCML
- SNG H5 hybrid SDK
- 微信JS SDK



QCML 架构





H5的未来



还是一个态度



敢吃螃蟹





提炼和分享





融合





健康生活



昵图网 www.nipic.com 87:5555

NO:20100228142203538797





为H5继续奋斗20年！



Q & A

<https://github.com/QzoneTouch>

微信：yunishi



参考资料

- HTML5. <http://www.w3.org/TR/2011/WD-html5-20110525/>. 2012.12.2
- W3schools. <http://www.w3schools.com/>. 2012.12.2
- Know iOS Resource Limits.
http://developer.apple.com/library/safari/#documentation/AppleApplications/Reference/SafariWebContent/CreatingContentforSafariiPhone/CreatingContentforSafariiPhone.html#//apple_ref/doc/uid/TP40006482-SW15. 2012.12.2
- Weinre. <http://people.apache.org/~pmuellr/weinre/docs/latest/>. 2012-12-2
- JpegEncoder.
http://www.bytestrom.eu/blog/2009/1120a_jpeg_encoder_for_java_script. 2012-12-2
- JpegMeta. <http://code.google.com/p/jsjpegmeta/>. 2012-12-2
- Seajs storage plugin. <http://ux.etao.com/posts/449>. 2012-12-2