**概述**

1. PC优化手段在Mobile侧同样适用  
2. 在Mobile侧我们提出三秒种渲染完成首屏指标  
3. 基于第二点，首屏加载3秒完成或使用Loading  
4. 基于联通3G网络平均338KB/s(2.71Mb/s)，所以首屏资源不应超过1014KB  
5. Mobile侧因手机配置原因，除加载外渲染速度也是优化重点  
6. 基于第五点，要合理处理代码减少渲染损耗  
7. 基于第二、第五点，所有影响首屏加载和渲染的代码应在处理逻辑中后置  
8. 加载完成后用户交互使用时也需注意性能

**优化指南**

**加载优化**

加载过程是最为耗时的过程，可能会占到总耗时的80%时间，因此是优化的重点

**减少HTTP请求**

因为手机浏览器同时响应请求为4个请求（Android支持4个，iOS 5后可支持6个），所以要尽量减少页面的请求数，首次加载同时请求数不能超过4个  
a) 合并CSS、JavaScript  
b) 合并小图片，使用雪碧图

**缓存**

使用缓存可以减少向服务器的请求数，节省加载时间，所以所有静态资源都要在服务器端设置缓存，并且尽量使用长Cache（长Cache资源的更新可使用时间戳）  
a) 缓存一切可缓存的资源  
b) 使用长Cache（使用时间戳更新Cache）  
c) 使用外联式引用CSS、JavaScript

**压缩HTML、CSS、JavaScript**

减少资源大小可以加快网页显示速度，所以要对HTML、CSS、JavaScript等进行代码压缩，并在服务器端设置GZip  
a) 压缩（例如，多余的空格、换行符和缩进）  
b) 启用GZip

**无阻塞**

写在HTML头部的JavaScript（无异步），和写在HTML标签中的Style会阻塞页面的渲染，因此CSS放在页面头部并使用Link方式引入，避免在HTML标签中写Style，JavaScript放在页面尾部或使用异步方式加载

**使用首屏加载**

首屏的快速显示，可以大大提升用户对页面速度的感知，因此应尽量针对首屏的快速显示做优化

**按需加载**

将不影响首屏的资源和当前屏幕资源不用的资源放到用户需要时才加载，可以大大提升重要资源的显示速度和降低总体流量  
PS：按需加载会导致大量重绘，影响渲染性能  
a) LazyLoad  
b) 滚屏加载  
c) 通过Media Query加载

**预加载**

大型重资源页面（如游戏）可使用增加Loading的方法，资源加载完成后再显示页面。但Loading时间过长，会造成用户流失  
对用户行为分析，可以在当前页加载下一页资源，提升速度  
a) 可感知Loading(如进入空间游戏的Loading)  
b) 不可感知的Loading（如提前加载下一页）

**压缩图片**

图片是最占流量的资源，因此尽量避免使用他，使用时选择最合适的格式（实现需求的前提下，以大小判断），合适的大小，然后使用智图压缩，同时在代码中用Srcset来按需显示  
PS：过度压缩图片大小影响图片显示效果  
a) 使用智图（ http://zhitu.tencent.com/ ）  
b) 使用其它方式代替图片(1. 使用CSS3 2. 使用SVG 3. 使用IconFont)  
c) 使用Srcset  
d) 选择合适的图片(1. webP优于JPG 2. PNG8优于GIF)  
e) 选择合适的大小（1. 首次加载不大于1014KB 2. 不宽于640（基于手机屏幕一般宽度））

**减少Cookie**

Cookie会影响加载速度，所以静态资源域名不使用Cookie

**避免重定向**

重定向会影响加载速度，所以在服务器正确设置避免重定向

**异步加载第三方资源**

第三方资源不可控会影响页面的加载和显示，因此要异步加载第三方资源

**脚本执行优化**

脚本处理不当会阻塞页面加载、渲染，因此在使用时需当注意

CSS写在头部，JavaScript写在尾部或异步

避免图片和iFrame等的空Src空Src会重新加载当前页面，影响速度和效率

**尽量避免重设图片大小**

重设图片大小是指在页面、CSS、JavaScript等中多次重置图片大小，多次重设图片大小会引发图片的多次重绘，影响性能

**图片尽量避免使用DataURL**

DataURL图片没有使用图片的压缩算法文件会变大，并且要解码后再渲染，加载慢耗时长

**CSS优化**

**尽量避免写在HTML标签中写Style属性**

**避免CSS表达式**

CSS表达式的执行需跳出CSS树的渲染，因此请避免CSS表达式

**移除空的CSS规则**

空的CSS规则增加了CSS文件的大小，且影响CSS树的执行，所以需移除空的CSS规则

**移除空的CSS规则**

空的CSS规则增加了CSS文件的大小，且影响CSS树的执行，所以需移除空的CSS规则

**正确使用Display的属性**

Display属性会影响页面的渲染，因此请合理使用  
a) display:inline后不应该再使用width、height、margin、padding以及float  
b) display:inline-block后不应该再使用float  
c) display:block后不应该再使用vertical-align  
d) display:table-\*后不应该再使用margin或者float

**不滥用Float**

Float在渲染时计算量比较大，尽量减少使用

**不滥用Web字体**

Web字体需要下载，解析，重绘当前页面，尽量减少使用

**不声明过多的Font-size**

过多的Font-size引发CSS树的效率

**值为0时不需要任何单位**

为了浏览器的兼容性和性能，值为0时不要带单位

**标准化各种浏览器前缀**

a) 无前缀应放在最后  
b) CSS动画只用 （-webkit- 无前缀）两种即可  
c) 其它前缀为 -webkit- -moz- -ms- 无前缀 四种，（-o-Opera浏览器改用blink内核，所以淘汰）

**避免让选择符看起来像正则表达式**

高级选择器执行耗时长且不易读懂，避免使用

**JavaScript执行优化**

**减少重绘和回流**

a) 避免不必要的Dom操作  
b) 尽量改变Class而不是Style，使用classList代替className  
c) 避免使用document.write  
d) 减少drawImage

**缓存Dom选择与计算**

每次Dom选择都要计算，缓存他

**缓存列表.length**

每次.length都要计算，用一个变量保存这个值

**尽量使用事件代理，避免批量绑定事件**

**尽量使用ID选择器**

ID选择器是最快的

**TOUCH事件优化**

使用touchstart、touchend代替click，因快影响速度快。但应注意Touch响应过快，易引发误操作

**渲染优化**

**HTML使用Viewport**

Viewport可以加速页面的渲染，请使用以下代码  
<meta name=”viewport” content=”width=device-width, initial-scale=1″>

**减少Dom节点**

Dom节点太多影响页面的渲染，应尽量减少Dom节点

**动画优化**

a) 尽量使用CSS3动画  
b) 合理使用requestAnimationFrame动画代替setTimeout  
c) 适当使用Canvas动画 5个元素以内使用css动画，5个以上使用Canvas动画（iOS8可使用webGL）

**高频事件优化**

Touchmove、Scroll 事件可导致多次渲染  
a) 使用requestAnimationFrame监听帧变化，使得在正确的时间进行渲染  
b) 增加响应变化的时间间隔，减少重绘次数

**GPU加速**

CSS中以下属性（CSS3 transitions、CSS3 3D transforms、Opacity、Canvas、WebGL、Video）来触发GPU渲染，请合理使用  
PS：过渡使用会引发手机过耗电增加