概述

最近在做一个上报相关的需求，该需求指定上班的内容中包含页面记载的事件、请求花费的事件、以及DOM渲染所要完成的时间等。  
为此查阅了大量的文档，收集了很多资料，所以趁热打铁，把自己所了解的记录下来，方便以后查询。

performance对象介绍

浏览器暴露给js的一个接口，可以通过这个接口查看用户访问网站的连接建立时间、dns时间等信息。使用该api时需要在页面完全加载完成之后才能使用，最简单的办法是在window.onload事件中读取各种数据，因为很多值必须在页面完全加载之后才能得出。

浏览器支持情况

IE9和chrome6以上的版本都支持：

pc端

1. window.performance : ie9
2. window.webkitPerformance : chrome6-9
3. window.performance : chrome10+

移动端

1. android4.0

performance是ECMAScript5中新增的一个特性，对于该特性，所支持的浏览器并不多。

属性和方法

属性

* performance.timing ：performance对象的timing属性指向一个对象，它包含了各种与浏览器性能有关的时间数据，提供浏览器处理网页各个阶段的耗时，这也是本文介绍的重点。
* performance.navigation ：网页导航的相关对象。
* performance.memory ：浏览器内存情况相关对象。

方法

* performance.getEntries ：浏览器获取网页时，会对网页中每一个对象（脚本文件、样式表、图片文件等等）发出一个HTTP请求。
* performance.mark：mark方法用于为相应的视点做标记。
* performance.now：performance.now方法返回当前网页自从performance.timing.navigationStart到当前时间之间的微秒数（毫秒的千分之一）

上报相关内容

上报相关的内容，大部分在performance.timing里面，performance.timing中包含的属性有：

* navigationStart：准备加载新页面的起始时间
* redirectStart：如果发生了HTTP重定向，并且从导航开始，中间的每次重定向，都和当前文档同域的话，就返回开始重定向的timing.fetchStart的值。其他情况，则返回0
* redirectEnd：如果发生了HTTP重定向，并且从导航开始，中间的每次重定向，都和当前文档同域的话，就返回最后一次重定向，接收到最后一个字节数据后的那个时间.其他情况则返回0
* fetchStart：如果一个新的资源获取被发起，则 fetchStart必须返回用户代理开始检查其相关缓存的那个时间，其他情况则返回开始获取该资源的时间
* domainLookupStart：返回用户代理对当前文档所属域进行DNS查询开始的时间。如果此请求没有DNS查询过程，如长连接，资源cache,甚至是本地资源等。 那么就返回 fetchStart的值
* domainLookupEnd：返回用户代理对结束对当前文档所属域进行DNS查询的时间。如果此请求没有DNS查询过程，如长连接，资源cache，甚至是本地资源等。那么就返回 fetchStart的值
* connectStart：返回用户代理向服务器服务器请求文档，开始建立连接的那个时间，如果此连接是一个长连接，又或者直接从缓存中获取资源（即没有与服务器建立连接）。则返回domainLookupEnd的值  
  -(secureConnectionStart)：可选特性。用户代理如果没有对应的东东，就要把这个设置为undefined。如果有这个东东，并且是HTTPS协议，那么就要返回开始SSL握手的那个时间。 如果不是HTTPS， 那么就返回0
* connectEnd：返回用户代理向服务器服务器请求文档，建立连接成功后的那个时间，如果此连接是一个长连接，又或者直接从缓存中获取资源（即没有与服务器建立连接）。则返回domainLookupEnd的值
* requestStart：返回从服务器、缓存、本地资源等，开始请求文档的时间
* responseStart：返回用户代理从服务器、缓存、本地资源中，接收到第一个字节数据的时间
* responseEnd：返回用户代理接收到最后一个字符的时间，和当前连接被关闭的时间中，更早的那个。同样，文档可能来自服务器、缓存、或本地资源
* domLoading：返回用户代理把其文档的 "current document readiness" 设置为 "loading"的时候
* domInteractive：返回用户代理把其文档的 "current document readiness" 设置为 "interactive"的时候.
* domContentLoadedEventStart：返回文档发生 DOMContentLoaded事件的时间
* domContentLoadedEventEnd：文档的DOMContentLoaded 事件的结束时间
* domComplete：返回用户代理把其文档的 "current document readiness" 设置为 "complete"的时候
* loadEventStart：文档触发load事件的时间。如果load事件没有触发，那么该接口就返回0
* loadEventEnd：文档触发load事件结束后的时间。如果load事件没有触发，那么该接口就返回0

上报的内容

上报的内容是通过上面的performance.timing各个属性的差值组成的，常用的有：

* DNS查询耗时 ：domainLookupEnd - domainLookupStart
* TCP链接耗时 ：connectEnd - connectStart
* request请求耗时 ：responseEnd - responseStart
* 解析dom树耗时 ： domComplete - domInteractive
* 白屏时间 ：responseStart - navigationStart
* domready时间 ：domContentLoadedEventEnd - navigationStart
* onload时间 ：loadEventEnd - navigationStart

http://javascript.ruanyifeng.com/bom/performance.html