一般我们可以通过浏览器的调试工具-网络面板，或者代理工具查看网页加载过程中的各个阶段的耗时。而利用window.performance属性则可以获得更为精确的原始数据，以毫秒为单位，精确到微秒。

**performance属性介绍**

如图所示： timing-overview.png

属性说明：

* navigationStart：浏览器处理当前网页的启动时间
* fetchStart：浏览器发起http请求读取文档的毫秒时间戳。
* domainLookupStart：域名查询开始时的时间戳。
* domainLookupEnd：域名查询结束时的时间戳。
* connectStart：http请求开始向服务器发送的时间戳。
* connectEnd：浏览器与服务器连接建立（握手和认证过程结束）的毫秒时间戳。
* requestStart：浏览器向服务器发出http请求时的时间戳。或者开始读取本地缓存时。
* responseStart：浏览器从服务器（或读取本地缓存）收到第一个字节时的时间戳。
* responseEnd：浏览器从服务器收到最后一个字节时的毫秒时间戳。
* domLoading：浏览器开始解析网页DOM结构的时间。
* domInteractive：网页dom树创建完成，开始加载内嵌资源的时间。
* domContentLoadedEventStart：网页DOMContentLoaded事件发生时的时间戳。
* domContentLoadedEventEnd：网页所有需要执行的脚本执行完成时的时间，domReady的时间。
* domComplete：网页dom结构生成时的时间戳。
* loadEventStart：当前网页load事件的回调函数开始执行的时间戳。
* loadEventEnd：当前网页load事件的回调函数结束运行时的时间戳。

**性能数据分析**

1.DNS寻址时间：t.domainLookupEnd - t.domainLookupStart。

优化方法：检查页面是否添加了DNS预解析代码， <link rel="dns-prefetch" href="[//haitao.nosdn1.127.net](http://haitao.nosdn1.127.net/)">， 合理利用域名发散与域名收敛的策略。

2.TCP连接耗时：t.connectEnd - t.connectStart。

3 首包时间: t.responseStart - t.navigationStart。

优化方法：是否加cdn，数据可否静态化等。

4.request请求耗时：t.responseEnd - t.requestStart。

优化方法：返回内容是否已经压缩过，静态资源是否打包好等。

5.白屏时间。白屏时间是最影响用户体验的，时间越久，用户等待就越久。

6.解析DOM树结构的时间：t.domComplete - t.domLoading。

优化方法：检查dom节点是否过多，dom是否嵌套过深。

7.页面加载完成的时间：t.loadEventEnd - t.fetchStart。

优化方法：考虑延迟加载，懒加载，部分加载，减少首屏渲染时间。

**方法**

performance.getEntries()：每个资源请求的时间数据。

performance.now() 计算网页从performance.timing.navigationStart到当前时间的毫秒数。

精确计算某个操作，或某个方法执行的耗时。

**var** start = performance.now();

dosomething();

**var** end = performance.now();

**var** time = end - start;

console.log('耗时' + time + '毫秒。');

或者console.time('aa'),console.timeEnd('aa')计算。

performance.mark()给相应的视点做标记。结合performance.measure()使用也可以算出各个时间段的耗时。

**对比分析**

环境：chrome版本51.0，隐身模式，wifi环境下：

**function** **getTimes**(){

**var** t = performance.timing;

**var** timing = {

domainLookupTime: t.domainLookupEnd - t.domainLookupStart,

TCPConnectTime: t.connectEnd - t.connectStart,

ttfb: t.responseStart - t.navigationStart,

requestTime: t.responseEnd - t.requestStart,

waitTime: t.domLoading - t.navigationStart,

renderDomTime: t.domComplete - t.domLoading,

loadPageTime: t.loadEventEnd - t.fetchStart

}

console.log(timing);

}

getTimes();

##### 参考资料：

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

<https://developer.mozilla.org/zh-CN/docs/Web/API/Performance>

<https://www.w3.org/TR/navigation-timing/#sec-window.performance-attribute>

<http://www.alloyteam.com/2015/09/explore-performance/>