css优化，js优化以及web性能优化

Css优化总结

对于css的优化可以从网络性能和css语法优化两方面来考虑。

Css性能优化方法如下：

1、css压缩

Css 压缩虽然不是高端的知识，但是很有用。其原理也很简单，就是把我们css代码中没有用的空白符等删除，达到缩减字符个数的目的。

压缩css代码的工具：

A、YUI compressor，可以在线压缩css和js代码。

B、gulp自动化构建工具，中的gulp-minify-css。

2、gzip压缩

Gzip是一种流行的文件压缩算法，现在应用的十分广泛，尤其实在Linux这个平台上，这个不止是对css，

当应用Gzip压缩一个文本时，效果是非常明显的。大约可以减少70%以上的文件大小（这取决于文件中的内容）。

在没有 gzip压缩的情况下，web服务器直接把html页面、css脚本以及js脚本发送到浏览器，而支持gzip的浏览器在本地进行解压和解码，并显示原文件。

这样我们传输的文件字节数减少了，自然可以达到网络性能优化的目的。Gzip压缩需要服务器的支持，所以需要在服务器端进行配置。

3、合写css（通过少些css属性来达到减少css字节码的目的）

例子：

```
background:#000 url(image.jpg) top left no-repeat;
font:font-style  font-weight  font-size  font-familiy;
margin:5px 10px 20px 15px;
padding:5px;
border:2px 5px 10px 3px;
```

border-top:2px 5px 10px 3px;

4、利用继承

Css的继承机制可以帮我们在一定程度上缩减字节数，我们知道css很多属性可以继承，即在父容器设置了默认属性，子容器会默认也使用这些属性。

可继承的属性举例：

所有元素都可以继承的属性：visibility 、cursor

内联元素和块元素可以继承的属性：

Letter-spacing、word-spacing、white-space、line-height、color、font、font-family、font-size、
font-style、font-variant、font-weight、text-decoration、text-transform、direction。

块状元素可以继承的属性：

Text-indent、text-align；

列表元素可以继承的属性：

List-style、list-style-type、list-style-position、list-style-image

表格元素可以继承的属性：

Border-collapse

不可以继承的属性：

Display、margin、border、padding、background、height、min-height、max-height、width、min-width、max-width、overflow、position、left、right、
top、bottom、z-index、float、clear、table-layout、vertical-align、page-break-after、page-break-before、unicode-bidi

5、抽离、拆分css，不加载所有css

抽离原则：在很多时候，我们把页面通用的css写到了一个文件，这样加载一次后，就可以利用缓存，但这样做兵不适合所有的场景。所以抽

离拆分的时候要考虑好。

6、css放在head中，减少repaint和reflow

Css方法在页面的顶部，有利于优化的原因？？？

当浏览器从上到下一遍下载html生成dom tree，一边根据浏览器默认以及现有css生成render tree来渲染页面。当遇到新的css的时候下载并结合现有css

重新生成render tree。则刚才的渲染功能就全废了。当我们把所有css放在页面的顶部，就没有重新渲染的过程了。

脚本中应该尽量少用repaint和reflow：

Reflow：当dom元素出现隐藏/显示，尺寸变化。位置变化的时候，逗号让浏览器重新渲染页面，以前的渲染工作白费了。

Repaint：当元素的背景颜色，边框颜色变化的时候，不会引起reflow的变化，是会让浏览器重新渲染该元素。

7、避免使用通配符或隐式通配符：visible.

8、避免层级或过度限制css（css是从右向左解析的）

A、不要用标签或class来限制id。

#test .info　/div #test这都属于画蛇添足

Id已经可以唯一而且快速的定位到一个元素了。

B、不要用标签限制class

Div .info 不好

在css代码编写中，如果直接使用class不能达到目的，一般是class设计出现了问题，css需要重构。

C、尽量使用最具体的类别，避免使用后代选择器，在css选择器中，后代选择器非但没有加快css查找，反而后代选择器是css中耗费最贵的。

JavaScript优化总结

1、避免全局查找

在一个函数中尽量将全局对象存储为局部变量来查找，因为访问局部变量的数要更快一些。

```
function(){
alert(window.location.href+window.location.host);
}
```

修改为：

```
funciton(){
var location=window.location;
alert(location.href+location.host);
}
```

2、定时器

如果针对的是不断运行的代码，不应该使用setTimeout，而应该使用setInterval。

因为setTimeout每一次都会初始化一个定时器。而setInterval只会在开始的时候初始化一个定时器。

```
var timeoutTimes=0;
function timeout(){
    timeoutTimes++;
    if(timeoutTimes<10){
        setTimeout(timeout,10);
    }
}
```

修改为：

```
var intervalTimes=0;
function interval(){
    intervalTimes++;
    if(intervalTimes>=10){
```

```
        clearInterval(interval)''
    }
}
var interval = setInterval(interval,10);
```

3、字符串连接

如果需要连接多个字符串，应该少使用+=：

如

```
s+=a;s+=b;s+=c;
```

修改为：

```
s+=a+b+c;
```

而如果是收集字符串，比如多次对同一个字符进行+=操作的话，最好使用一个缓存，使用js数组来收集，最后join方法连接起来。

```
var buf=[];
for(var i=0;i<100;i++){
    buf.push(i.toString());
}
var all=buf.join("");
```

4、避免with语句

和函数类似，with语句会创建自己的作用域，因此会增加其中执行的代码的作用域链的长度。犹豫额外的作用域链的查找，

在with语句中执行的代码肯定会比外面执行的低吗要慢，在能不使用with语句的时候，尽量不要使用with语句。

```
with(a,b,c,d){
    property1=1;
    property2=2;
}
```

修改为：

var obj=a.b.c.d;

obj.property1=1;

obj.property2=2;

5、数字转为字符串

一般用“”+1来将数字转为字符串，虽然看起来比较丑一点，但事实上这个效率最高。

（“”+1）>String()>.toString()>new String()

6、浮点数转化为整型

很多人喜欢使用pareseInt(),其实parseInt()是用于将字符串转为数字，而不是浮点数和整型之间的转换。我们应该使用的是Math.floor()或Math.round();

7、多个类型声明

所有变量可以使用单个var语句来声明，这样就是组合在一起的语句，以减少整个脚本的执行时间。

```
var A=1,
    B=2,
    C=3;
```

8、使用直接量

```
var aTest=new Array();//var aTest=[]
var aTest=new Object();//var aTest={}
var reg=new RegExp();//var reg=/ /;
```

创建具有特殊性的一般对象，也可以使用字面量。

```
var oFruit=new O();
oFruit.color="red";
oFruit.name="apple";
```

应改为

```
varoFruit={color:"red",name:'apple'}
```

9、使用DocumentFragment优化多次的append

一旦需要跟新DOM，请考虑使用文档碎片来构建结构，然后再添加到现有的文档中。

```
for(var i=0;i<1000;i++){
    varel=document.createElement('p');
    el.innerHTML=i;
    document.body.append(el);
}
```

应该改为

```
varfrag=document.createDocumentFragment();
for(var i=0;i<1000,i++){
    varel=document.createElement('p');
    el.innerHTML=i;
    frag.append(el);
}
document.body.appendChild(frag);
```

10、使用一次innerHTML复制代替构建dom元素。

对于大的dom更改，使用innerHTML要比使用标准的dom方法创建同样的dom结构快的多。

```
varfrag=document.createDocumentFragment();
for(var i=0;i<1000;i++){
    varel=document.createElement('p');
    el.innerHTML=i;
    frag.appendChild(el);
}
document.body.appendChild(frag);
```

应该改为

```
var html=[];
for(var i=0;i<1000;i++){
    html.push('<p>'+i+"</p>");
}
document.body.innerHTML=html.join("");
```

11、使用firstChild和nextSibling代替childNodes遍历dom元素。

```
var nodes=element.childNodes;
for(var i=0,l=nodes.length;i<l;i++){
    varnode=nodes[i];
}
```

应该改为

```
var node=element.firstChild;
while(node){
    node.node.nextSibling;
}
```

12、删除dom节点

删除dom节点之前，一定要删除注册在该节点上的事件，不管是observe方式还是用attachEvent方式注册的事件。

否则将会产生无法回收的内存。另外，removeChild和innerHTML= ‘’ 两者之间尽量选择后者，因为在内存泄漏工具中监测的结果是用removeChild无法有效的释放dom节点。

13、简化终止条件

由于每次循环过程，都会计算终止条件。所以必须保证他尽可能的快。也就是避免属性查找或其他操作。

最好是将循环控制量保存到局部变量中，也就是说对数组或列表对象遍历的时候，提前将length保存到局部变量中，避免循环的每一步重复取值。

```
varlist=document.getElementsByTagName('p');
for(var i=0;i<list.length;i++){

}
```

应该改为

```
for(var i=0,l=list.length;i<l;i++){

}
```

14、使用后测试循环

在js中，我们使用for(;;),while(),for(in)三种循环。for(in)的效率极差。因为他需要查询散列键，只要可以，就应该尽量少用。

for(;;)和while循环，while优于for（ ； ； ）,可能for（ ； ； ）结构问题，需要经常的跳转。do..while更好。

15、尽量少用eval函数

使用eval函数相当于在运行时，再次调用解释引擎对内容进行运行，需要消耗大量时间，而且使用eval带来的安全性问题也是不容忽视的。

16、不要给setTimeout或setInterval传递字符串参数。

```
var num=0;
setTimeout(num++,10);
```

应该改为

```
var num=0;
function addNum(){
    num++;
}
setTimeout(addNum,10);
```

17、缩短否定检测

```
if(oTest!="#ff0000"){}
```

if(oTest!=null){}

if(oTest!=false){}

以上都不太好

if(!oTest){这个比较好}

18、可以用三目运算符替换条件分支，可以提高效率。

Web性能优化

1、避免坏请求

有时候页面中的html或css会向服务器请求一个不存在的资源，比如图片或者html文件，这会造成浏览器与服务器之间过多的往返请求。

类似于：

浏览器：“我需要这个图像”

服务器：“我没有这个图像”

浏览器：“你确定吗？这个文档说你有”

服务器：“真的没有”

这个降低页面加载速度。因此检查坏连接很有必要。可通过Google的PageSpeed工具，找到问题后，补充相当资源文件或者修改资源链接地址即可。

2、避免css@import

使用@import方法引用css文件可能会带来一些影响页面加载速度的问题。比如导致文件按顺序加载（一个加载完成后才会加载另一个），无法并行加载；

检查工具：css delivery

查到@import url(“style.css”)

就替换为：<link….>

3、避免使用document.write

在js中，可以使用document.write。在网页上显示内容或者调用外部资源，而通过此方法，浏览器采取一些多余的步骤（下载资源，读取资

源）。
运行js来了解需要做什么，调用其他资源时，需要重新在执行一次这个过程。由于浏览器之前不知道要显示什么，所以会降低页面加载的速度。
要知道，任何能够被document.write调用的资源，都可以通过html调用。这样速度会更快
document.write('<scriptsrc="another.js"></script>');
改为
<scriptsrc="another.js"></script>
4、合并多个外部css文件
网站中每使用一个css文件，都会让你的页面加载速度慢一点。可以css delivery工具，来检测页面代码中css文件。然后通过复制粘贴合并成一个。
5、合并多个外部js文件
可以用resource check来检测页面中所引用的js文件数，然后可以通过复制粘贴的方法将多个文件合并成一个。
6、通过css sprites来整合图像
若页面中有6个小图像，那么浏览器在显示时会分别下载，你可以通过css sprites将这些图像合并成为一个，可以减少页面加载所需要的时间。
Css sprites两个步骤：整合图像，定位图像
7、延迟js加载
浏览器在执行js代码时，会停止处理页面。当页面中很多js文件或者代码要加载时，将导致严重的延迟。
尽管可以使用defer，异步或将js代码放自爱页面底部来延迟js的加载。但这些都不是一个好的解决方案。
好方法

```
<script>
function downloadJSAtOnload(){
    varelement=document.createElement("script");
    element.src="defer.js";
    document.body.appendChild(element);
}
if(window.addEventListener){
    window.addEventListener('load',downloadJSAtOnload,false);
}else if(window.attachEvent){
    window.attachEvent('onload',downloadJSAtOnload);
}else{
    window.onload=downloadJSAtOnload;
}
</script>
```

8、启用压缩/Gzip

使用gzip对html和css文件进行压缩，通常可以大约节省50%到70%，这样加载页面只需要更少的带宽和更少的时间。

9、如果你的css和js较小，可以将css和js内嵌到html页面中，这样可以减少页面加载所需要的文件数，从而加快页面的加载。

10、用minify css压缩css代码

11、尽量减少dns查询次数

当浏览器和服务器建立链接时，它需要进行dns解析，将域名解析为ip地址，然而，一旦客户端需要执行dns lookup时，等待时间将会取决于域名服务器的有效响应速度。

虽然所有的isp的dns服务器都能缓存域名和ip地址映射表。但如果缓存的dns记录过期了而需要更新，则可能需要遍历多个dns节点，有时候需要通过全球范围内来找到可信任的域名服务器，

一旦域名服务器工作繁忙，请求解析时，就需要排队则进一步延时等待时间。
所有减少dns查询次数很重要，页面加载就尽量避免额外耗时，为了减少dns查询次数，最好的解决方法就是在页面中减少不同的域名请求的机会、
可通过request checker工具来检测页面中存在多少请求后，进行优化。
12、尽量减少重定向
有时候为了特定需求，需要在网页中使用重定向。重定向的意思是，用户的原始请求（如请求A）被重定向到其他的请求（如请求B）；
网页中使用重定向会造成网站性能和速度下降，因为浏览器访问网址是一连串的过程，如果访问到一半，而跳转到新的地址，就会重复发起一连串的过程，这将浪费很多时间。所有我们尽量避免重定向。Google建议
A、不要链接到一个包含重定向的页面
B、不要请求包含重定向的资源
13、优化样式表和脚步顺序
Style标签和样式表调用代码应该放置在js代码的前面，这样可以使页面的加载速度加快。
14、避免js阻塞渲染
浏览器在遇到一个引入外部js文件的<script>标签时，会停下所有工作下载并解析执行它。在这个过程中，页面渲染和用户交互完全被阻塞了。这是页面加载就会停止。
谷歌建议删除干扰页面第一屏内容加载 的js，第一屏指的是用户在屏幕中最初看到的页面，无论桌面浏览器，还是手机
15、指定图像尺寸

当浏览器加载页面的html时，有时候需要在图片下载完成前，对页面布局进行定位。如果hmtl里的图片没有指定尺寸，
或者代码描述的尺寸和实际的图片尺寸不符合时，浏览器需要在图片下载完成后在回溯到该图片，并重新显示，这将消耗额外的时间。
最好的页面中每一张图片都指定尺寸，不管在html里的img中，还是在css中。