淘宝网
Taobao.com

第三方广告代码稳定性和性能优化实战

淘宝 广告技术部 李穆

2010年12月7日

第三方内容对您网站的影响

第三方内容对您网站的影响

第三方内容对网站速度的影响

- 网络是缓慢的:平均加载时间:4.9s
- 页面是复杂的:44个请求来自7个域名,平均大小320KB
- 很多加载的内容来自第三方

Velocity 2010：Google -- Don’t Let Third Parties Slow You Down

第三方内容对网站速度的影响

第三方内容对页面加载时间的影响:

Third-party	Publisher site	% Impact
Digg	services.newsweek.com	14
Digg	realtalkny.uproxx.com	9
FriendConnect	www.artinstructionblog.com	10
FriendConnect	friendconnectdirectory.com/Food	30
FacebookConnect	truveo.com	17
FacebookConnect	www.huffingtonpost.com	12
TribalFusion	www.xe.com	53
TribalFusion	www.wareseeker.com	31

Google Adsence	top 100 publishers	12.8
Google Analytics	top 100 publishers	<5
Google Doubleclick	top 100 publishers	11.5

第三方内容对网站速度的影响

P3PC -- 第三方内容性能分析项目:

P3PC 项目专注于第三方内容的性能分析.目标是找到让第三方内容更快的关键点,促进第三方的改进性能.

Performance of 3rd Party Content (P3PC) Project by Steve Souders

第三方内容对网站速度的影响

snippet blog post	impact on page	Page Speed	YSlow	doc. write	total reqs	total xfer size	JS ungzip	DOM elems	median Δ load time
Digg widget	big	88	84	y	8	52 kB	107 kB	84	699 ms
Facebook Sharer	small	90	92	n	5	8 kB	7 kB	15	137 ms
Google Analytics	small	91	99	n	2	18 kB	24 kB	2	29 ms
BuySellAds*	small	81	92	n	3	7 kB	14 kB	9	26 ms
Google AdSense*	big	87	84	y	8	41 kB	76 kB	9	114 ms
ValueClick*	med	89	98	y	3	2 kB	1 kB	1	na**
Quantcast	small	93	98	n	2	3 kB	3 kB	2	108 ms
Collective Media*	big	86	90	y	6	8 kB	9 kB	7	na**
Glam Media*	big	89	83	y	11	68 kB	63 kB	7	na**
品牌广告	---	**92**	**97**	**y**	**3**	**13 kB**	**39 kB**	**8**	**270 ms**

总体评分

- Page Speed
- YSlow

传输方面

- 总请求数
- 总数据量

运行方面

- JS代码量
- Dom节点量

依赖与影响

- 是否使用doc.write
- 对加载时间的影响

第三方内容可能影响网站稳定性

广告埋点是怎样结构?如何运行?

```
<script>
    alimama_pid="mm_1_2_3";alimama_type=2;
    alimama_width=270;alimama_height=390;
</script>
<script src="http://a.alimama.cn/inf.js"></
script>
```

document.write:

```
<iframe(script) src="http://t.alimama.com/a?i=mm_1_1_1
&fv=10.1&rd=xyz&u=a.com%2Fa.html"></iframe(script)>
```

第三方内容可能影响网站稳定性

为什么选择这种埋点结构?

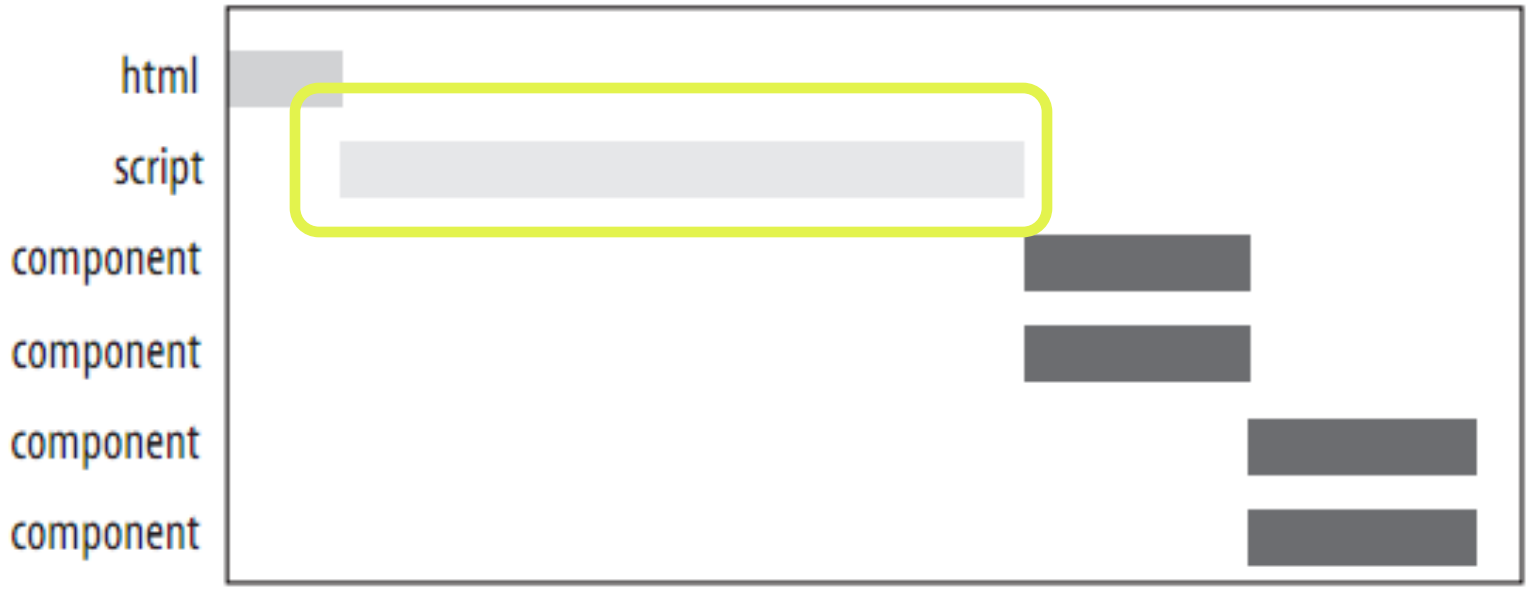
- **接口简单**:接口代码即功能强大又简单易用.
- **位置明确**:广告出现在埋点所在处.
- **展现迅速**:页面解析到埋点处立即触发广告展现.

依靠哪些技术要点?

引入外部脚本的<script>节点 + doc.write

第三方内容可能影响网站稳定性

脚本阻滞:HTML中<script>节点对应的脚本下载且执行结束之前,浏览器不会发起任何新的HTTP请求,也不会对该节点以下的任何内容进行渲染.



第三方内容可能影响网站稳定性

多数第三方代码会调用doc.write即刻输出展现,这限制了接口JS代码位置不能按照一般的优化方法移至页面底部,从而为系统引入不可控的**单一故障点**.

单一故障点:Single Point of Failure(SPoF)是指一个系统的这样一个部件，如果它失效或停止运转，将会导致整个系统不能工作。

Steve Souders :Frontend SPoF

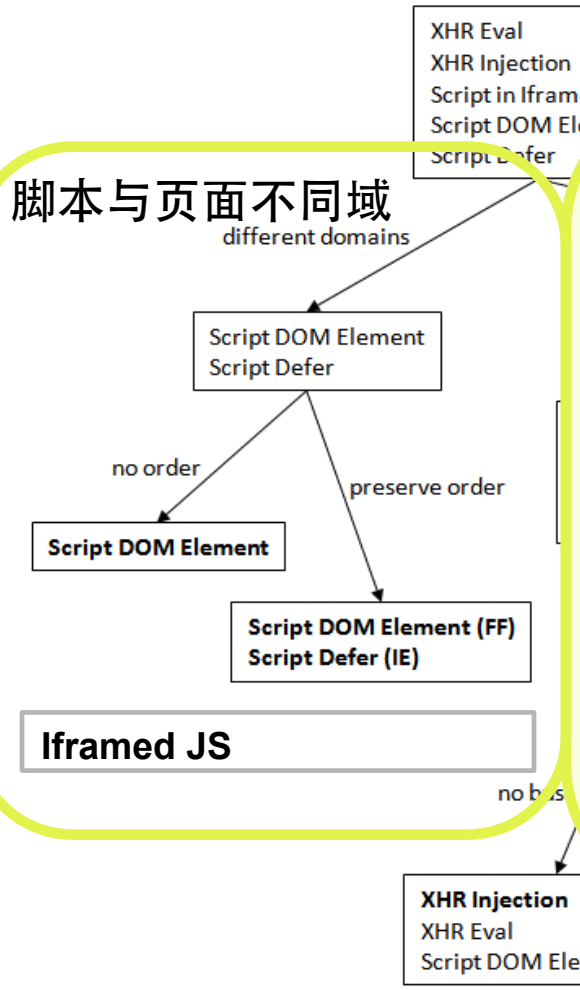
稳定性与性能优化思路

- 降低故障几率直至去除单一故障点.
- 尽量不抛弃原有埋点代的码接口简单,位置明确,展现迅速等方面的优势.
- 解决脚本阻滞问题
  - 无阻滞脚本异步加载方案
  - 使用doc.write的替代方案

淘宝网
Taobao.com

无阻滞脚本下载方案

无阻滞脚本下载方案



脚本不在所处页面的同域名下时,无阻滞加载备选方案:

Script Defer

Script DOM Element

Iframed JS

Steve Souders: Loading Scripts Without Blocking

载入Defer脚本速度不符合要求

IE中带defer属性的<script>执行时机即为domReady时.
国内若干热门页面domReady时间的一次随机采样数据:

网站	频道	domReady↑	onload	domReady/onload
qq	qzone	0.525s	0.83s	63.25%
baidu	tieba	0.599s	0.753s	79.55%
youku	video	1.54s	1.86s	82.80%
sohu	news	3.185s	3.7s	86.08%
sina	sports	5.11s	7.2s	70.97%
163	news	8.38s	12.32s	68.02%

存在大量页面domReady时间在1.5s以上,在onload时间的50%之后.
Script Defer加载广告脚本会严重拖后广告展现,不可行!

Script Dom Element 方法详情

```
function scriptDomElement(u) {
  var s = document.createElement('script'),
  h = document.getElementsByTagName('head')
[0];
  s.src = u;
  s.async = true;
  if(h)h.insertBefore(s,h.firstChild);
}
```

通过Dom方法创建一个script节点,并插入到文档当中.

在各种浏览器中这种方式都能保证脚本与其他资源并行下载.

Script Dom Element 兼容性

外部文件test.js,内容: **var g = 1;**页面上脚本如下:

```
<script>
  function scriptDomElement(u) {
    var s = document.createElement('script'),
    h = document.getElementsByTagName('head')[0];
    s.src = u; s.async = true; //不能忽略
    if(h)h.insertBefore(s,h.firstChild);
  }
  scriptDomElement('test.js');
</script>
<script>
  alert(typeof g); // 预期 undefined, Firefox 3.6 弹出 number
</script>
```

阻滞并不仅仅是平行下载的问题,还要区分下载阻滞和运行阻滞.
如果test.js较慢,Firefox会在下一个script节点处导致运行阻滞.
在给script增加了async属性后,除了Opera浏览器都做到了真无阻.

玉伯:Script 元素的异步加载属性

Iframed JS 方法详情

```
function iframedJS(s){
  document.write("<iframe id= 'i'></iframe>");
  var d = document.getElementById("i").contentWindow.document;
  d.write('<!doctype html><html><body><scr' + 'ipt src="'+s
+'"></scr' + 'ipt></body></html>');
  window.setTimeout((function(){d.close();}),0);
}
```

没有src的iframe的location和父页面相同,所以不存在跨域问题.

iframe内的脚本下载对父页面其他内容而言是无阻滞的.

Iframed JS 兼容性

- 父页面进行了域名升级:document.domain= "a.com",IE报错在iframe的src,通过javascript协议执行同样域名升级语句

```
<iframe id= "testiframe" src= "javascript:void((function()
{var d=document;d.open(); d.domain='a.com';d.write
('');d.close()})())"
></iframe>
```

- 刷新原页面, javascript:协议的语句可能未执行.
- Firefox doc.write iframe至页面,可能不能马上取到其引用.
- 对同一个iframe多次进行doc.open+write+close,会增加浏览器历史记录.

关于稳定性和速度的疑问

脚本加载由阻滞改为并行,势必引发多个资源间的竞争.

在复杂的网页上下文环境中,浏览器是否会及时且有效的启动异步脚本的加载和执行?即速度与稳定性如何?

我们选取inf.js部分流量进行了测试:
准备1.js,2.js,3.js..(Gzip后7k)主要运行一句话:

```
alimama_test.callback("1"); //2, 3…
```

根据callback的情况,发送统计请求到我们的量子统计进行计数.

无阻方案稳定性和速度测试1

CASE1:Dom Script Element 稳定性和速度测试.

方法一:模拟直接script埋点加载脚本

```
document.write("<script src='1.js'></script>");
```

方法二:直接通过Script Dom Element 加载脚本

```
scriptDomElement("2.js");
```

方法三:在setTimeout 0ms 后调用 Script Dom Element

```
setTimeout((function(){
        scriptDomElement("3.js");
 }),0);
```

在随机选中的PV中让这三个方法按照随机顺序执行分别执行

淘宝网 Taobao.com

Script Dom Element的稳定性

	doc.write (PV)	script dom(PV)	script dom /doc.write(%)	timeout0+ script dom(PV)	timeout0+script dom /doc.write(%)
IE6	15435	15469	100.22%	15265	98.90%
IE7	4613	4620	100.15%	4549	98.61%
IE8	2021	2019	99.90%	2002	99.06%
Others	46284	47940	103.58%	48388	104.55%

三种方法统计到的PV基本一致,说明Script Dom Element基本可靠

Script Dom Element的速度

计算两种Dom方法与doc.write的载入js的执行callback时间差

		总PV	<0 (ms)	0~100 (ms)	100~300 (ms)	>300 (ms)	>300(ms) /总PV(%)
[script dom] - [doc.write]	IE6	14813	6821	6003	859	1130	7.63%
	IE7	4427	2058	1712	320	337	7.61%
	IE8	1958	969	723	115	151	7.71%
	Others	47819	17324	21443	4452	4600	9.62%
[timeout0+script dom] - [doc.write]	IE6	14764	2660	4405	2971	4728	32.02%
	IE7	4388	677	1082	941	1688	38.47%
	IE8	1955	336	409	474	736	37.65%
	Others	47735	5943	23377	8147	10268	21.51%

综述:script dom相比doc.write有近8%的请求慢300ms以上,待定.
timeout0+script dom则有35%左右的请求慢300ms以上.淘汰之.

无阻方案稳定性和速度测试2

CASE2:Dom Script Element Vs. Iframed JS 速度测试.

方法一:模拟直接script埋点加载脚本

```
document.write("<script src='1.js'></script>");
```

方法二:通过Script Dom Element 加载脚本

```
scriptDomElement("2.js");
```

方法三:通过Iframed JS 加载脚本

```
iframedJS("3.js")
```

在随机选中的PV中让这三个方法按照随机顺序执行分别执行

Iframed JS 速度测试结论

		All	<0 (ms)	0~100 (ms)	100~300 (ms)	300~500 (ms)	500~1000 (ms)	>1000 (ms)	>300 /All(%)
[script dom] - [doc.write]	IE6	6045	5547			108	123	267	8.24%
	IE7	1827	1639			45	53	90	10.29%
	IE8	872	788			19	27	38	9.63%
	Others	15368	14074			312	304	678	8.42%
[iframed JS] - [doc.write]	IE6	6029	2596	2448	476	128	123	258	8.44%
	IE7	1819	888	588	157	45	44	97	10.23%
	IE8	867	469	230	88	23	28	29	9.23%
	Others	15332	11888	1956	524	225	216	523	6.29%

综述:Iframed JS 相比doc.write同样有9%的请求慢300ms以上,也加入备选方案

淘宝网 Taobao.com

Dom Script Vs.Iframed JS

- 两种技术都能做到无阻滞脚本加载,不引入SPoF(单一故障点).
- 都能保障JS文件的正确加载执行.
- 都略有延迟,8%左右的反馈慢于直接doc.write引入的JS300ms.
- 在直接较量中,测试收到的25013次反馈中,有14404次Iframed JS先返回.略占优.
- 使用Iframed JS重构原有代码改动量要大一些

Stable by Default:新广告埋点

如何改造埋点?

如果再发布一版广告埋点代码我们会怎么做?

毫无疑问会使用稳定且不慢无阻脚本加载方案!

方案会引入其他问题么?肯定有,那必须解决掉!

解决广告所在位置问题

问题:异步加载脚本中无法使用doc.write,也就无法通过doc.write确定广告位置.

在发布代码的in-line脚本中通过document.write一个锚点做参照.

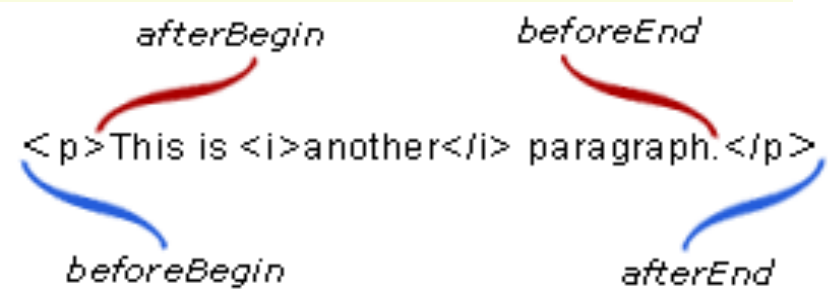
```
<script>
  document.write("<a style='display:none !important' id='a1'></a>");
  insertAsyncScript("http://adhost/?aid=a1");
</script>
```

也可以由用户指定广告展现的容器元素

解决广告内容插入问题

问题:如何方便的将广告HTML片段插入指定的位置?
使用 insertAdjacentHTML 方法.

```
function show(sHTML,anchor,container){
    if(anchor){
        anchor.insertAdjacentHTML("beforeBegin",sHTML);
    }else if(container){
        container. insertAdjacentHTML("afterBegin",sHTML);
    }else{
        document.write(sHTML)
    }
}
```



FireFox不支持insertAdjacentHTML! 通过扩展HTMLElement对象实现这个方法.
insertAdjacentHTML@W3C HTML5 insertAdjacentHTML@MSDN

如何避免IE进程崩溃的可能

问题:页面Loading过程中某些Dom操作会导致IE进程崩溃.

“无法打开Internet站点..已终止操作.”

@MSDN

借鉴YUI针对节点的onContentReady事件,操作已有nextSibling属性的节点:

Executes the callback as soon as the specified element is detected in the DOM with a nextSibling property (indicating that the element's children are available, determine if the content of the available element is safe to modify.)

YUI2 YUI3

如何避免客户页面CSS干扰

问题:输出到客户页面的内容易受到原页面CSS干扰.

- 使用不常见的Tag.
- 使用脚本输出自定义Tag.
- 使用IframedJS同样的办法,将广告展现在动态生成的src为空的iframe中.

Stable by Default的广告埋点

1.普通埋点

```
<script src="http://{host}/{path}?i={pid}"></script>
```

2.无阻埋点

```
<script>
 document.write('<a style="display:none !important" id="t-a-{id}"></a>');
 t_h = document.getElementsByTagName('head')[0];
 t_s = document.createElement('script');
 t_s.async = true;
 t_s.src = 'http://{host}/{path}?i={pid}';
 if(t_h)t_h.insertBefore(t_s,t_h.firstChild);
</script>
```

淘宝网
Taobao.com

Fast by Default:原有埋点优化

原埋点优化方案 -- 第一阶段

保留主体框架,对于普通Banner广告不引入额外的请求.

复杂广告展现形式所需的代码,按需加载.

预期:

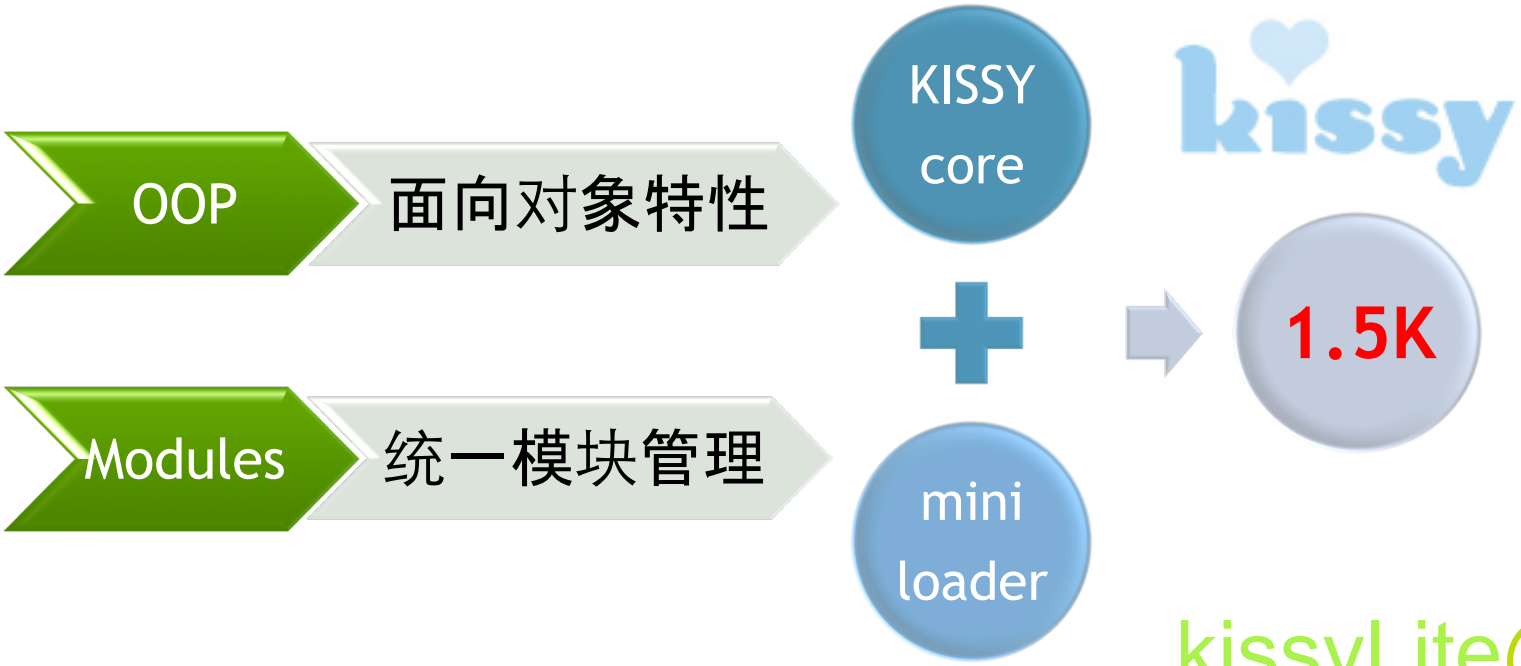
- inf.js大小Gzip后11k降至Gzip后5k左右.
- 对避免长时间阻滞的帮助不大

关于广告前端代码组织

第三方代码的要求: 小.原生.基本上所有类库都用不上.....
类库提供什么:

- 对JS对DOM功能的补强(oo,modules,domready,jsonload…)
- 对写法的简化和优化(lang,selector…)

补强部分是第三方代码亟需的:



kissyLite@kissy

原埋点改进方案 -- 第二阶段

inf.js只包含“alimama_”前缀的变量收集,输出占位锚点和kissyLite.

预期:

- inf.js文件(可能产生阻滞的部分)降至最低(约2k).
- 首个广告请求加载内容从11k下降至6k左右,但多一个脚本请求.
- 在用户不修改发布代码前提下,页面第二个广告的加载量仅2k.

固化JS - 去除SPoF的迂回路线

最小化的inf.js将会成为淘宝广告投放平台的固定API. 有明确的版本号和固定的内容,这样可以发布在合作伙伴的服务器上,或直接写成网页中的in-line脚本内.

这样在任何位置加入淘宝代码,整体页面的稳定性都不再依赖淘宝的CDN, 达到了去掉SPoF(单一故障点)的目的,对稳定性要求高的合作伙伴完全可以这样做.

淘宝网
Taobao.com

Mashuper团结起来,消灭SPoF.

做无SPoF的第三方内容开发

希望能够去除第三方引入SPoF有效解决方案,在第三方开发者当中有统一的规范.

首要是去除依赖直接<script>节点的doc.write使用

考虑 show(sHTML,anchor,container) 么 ☺

淘宝网
Taobao.com

关于引入第三方内容的建议.

关于引入第三方内容的建议

当前阶段网站开发者怎么做来保障引入的第三方内容的稳定性?

只引入信任的代码!

如果展现为固定尺寸的块状区域,尽量将其放入IFRAME!

保障IFRAME和父页面间无障碍通信,不会损失广告效果.

回顾.未来..

回顾:稳定问题解决进行时

安全和稳定是第三方内容开发者追求速度的前提.

阻滞的脚本引入方式为系统增加了不可控的单一故障点,解决办法是什么.

介绍更稳定的新埋点模式,以及相关问题的解决方案.

介绍对原有埋点进行性能优化,同时提供需要合作伙伴配合修改以彻底解决问题的方法.

未来:安全问题不能回避

关于第三方代码可能引入的安全问题.解决起来复杂的多.各种解决方案伴随着各种限制.

Douglas Crockford:ADsafe

Google:google-caja

Facebook:FBML

UIC:ADJail

Nicholas C. Zakas的悬赏

未来:我们的计划

在第三方代码不去修改的前提下,尝试只去解决部分问题,主要包括由doc.write引入稳定性隐患和由doc.cookie引入的隐式的cookie盗用隐患.

淘宝网
Taobao.com

Thanks All!

limu@taobao.com
http://twitter.com/lenel_li
http://limu.javaeye.com/