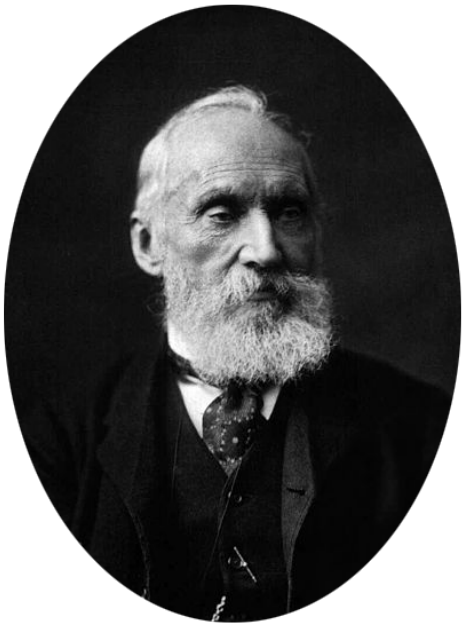
FEX Web 前端研发部

前端数据之美

张军-@大人物oo7

目录

Part 1 前端监控模式



“If you cannot measure it, you cannot improve it”

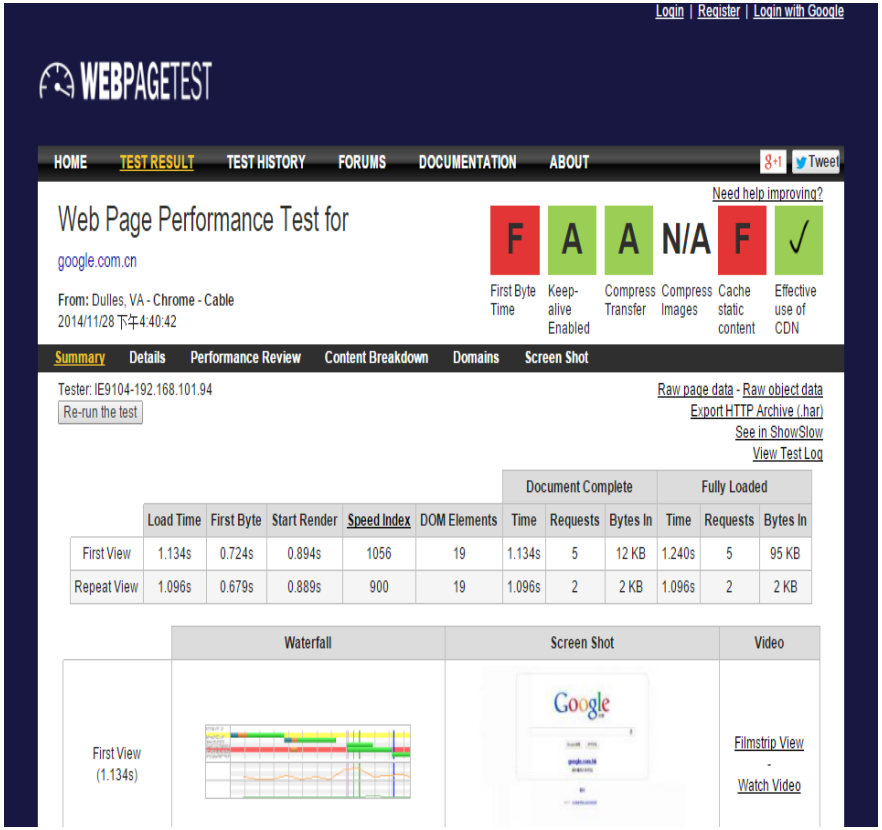
——William Thomson

Data!!!

监控方式：非侵入式

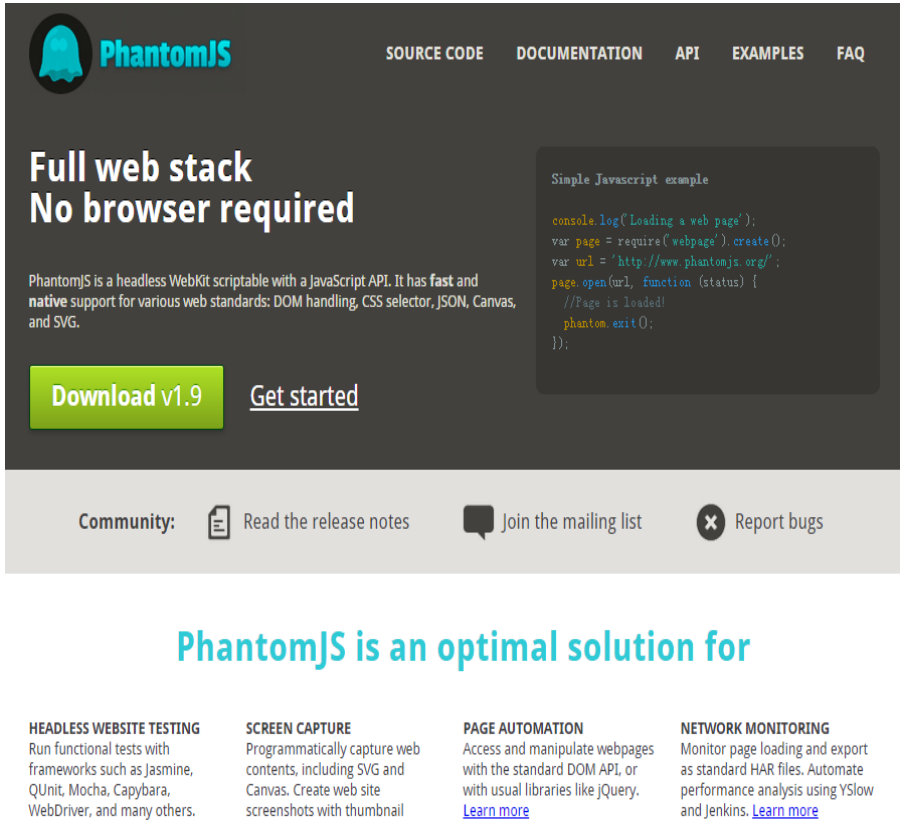
WebPageTest

输入url获取各地测试数据



PhantomJS

命令行工具



监控方式：侵入式

JS监测

页面中嵌入监控脚本

监控方式对比

类型	优点	缺点	示例
非侵入式	• 指标齐全 • 客户端主动监测 • 竞品监控	• 无法知道影响用户数 • 采样少容易失真 • 无法监控复杂应用与细分功能	Yslow， WebPageTest， PhantomJS， 百度UAQ
侵入式	• 真实海量用户数据 • 能监控复杂应用与业务功能 • 用户交互、异步操作	• 需插入脚本统计 • 浏览器兼容性 • 无法监控竞品	Google统计， 各种用户行为监控

我们的做法

- 使用JS监测线上用户真实访问性能 **为主**
- 使用PhantomJS等工具线下分析 **为辅**

Part 2 如何采集前端数据

一、监控原理

二、监控框架

1. 统一发送数据
2. 配置抽样命中监控模块
3. 异步加载监控模块
4. 提供公共的on、un、fire等事件操作

Alog： https://github.com/fex-team/alog

三、监控模块

① 访问痕迹

② 性能数据

③ JS异常

④ 其他：点击分析、跨域资源统计等

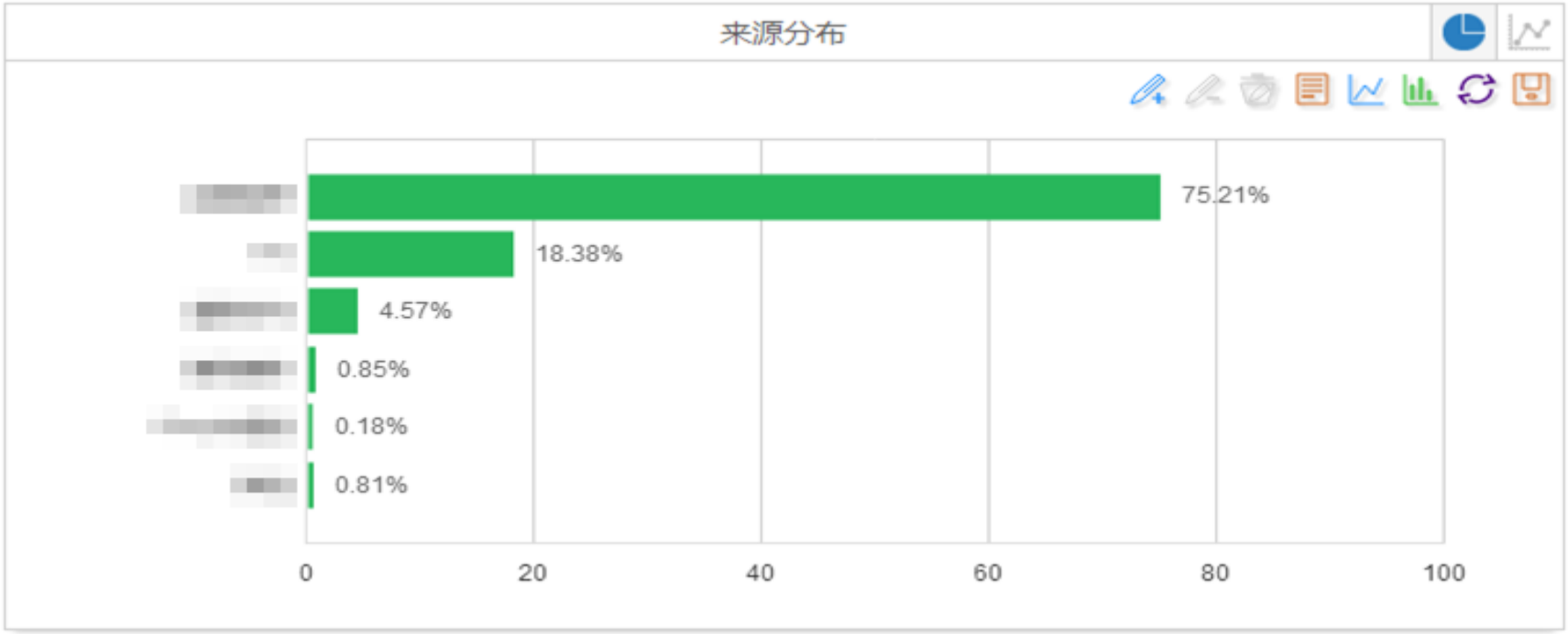
① 访问痕迹

```
// 创建pv监控模块
alog('pv.create', {
    postUrl: 'http://localhost:8080/pv.gif',
    pageId: 'tieba-index',
    refer: document.referrer
});
// 发送数据
alog('pv.send', 'pageview');
```

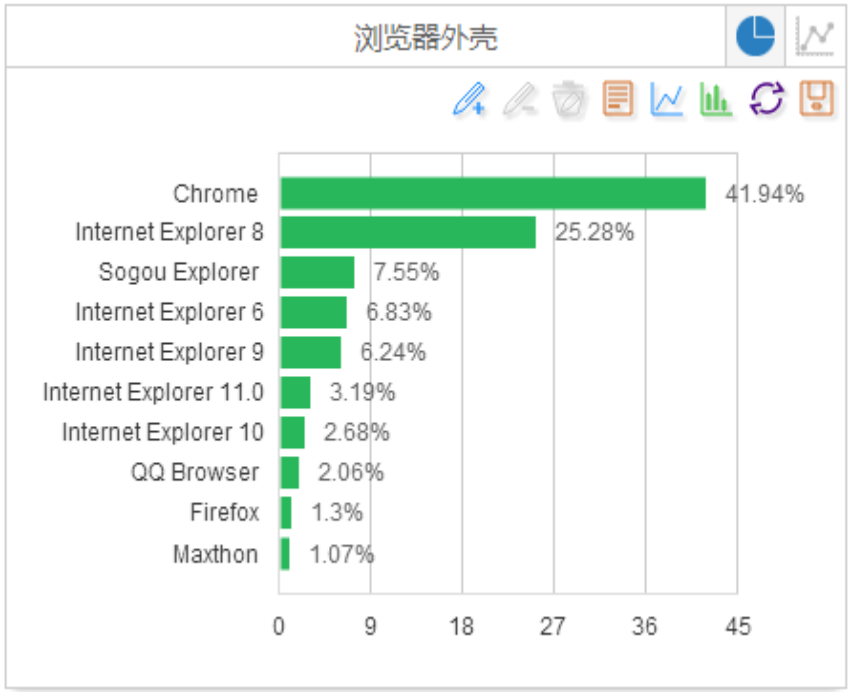
http://localhost:8080/pv.gif?pageId=tieba-index&type=pageview&refer=baidu.com

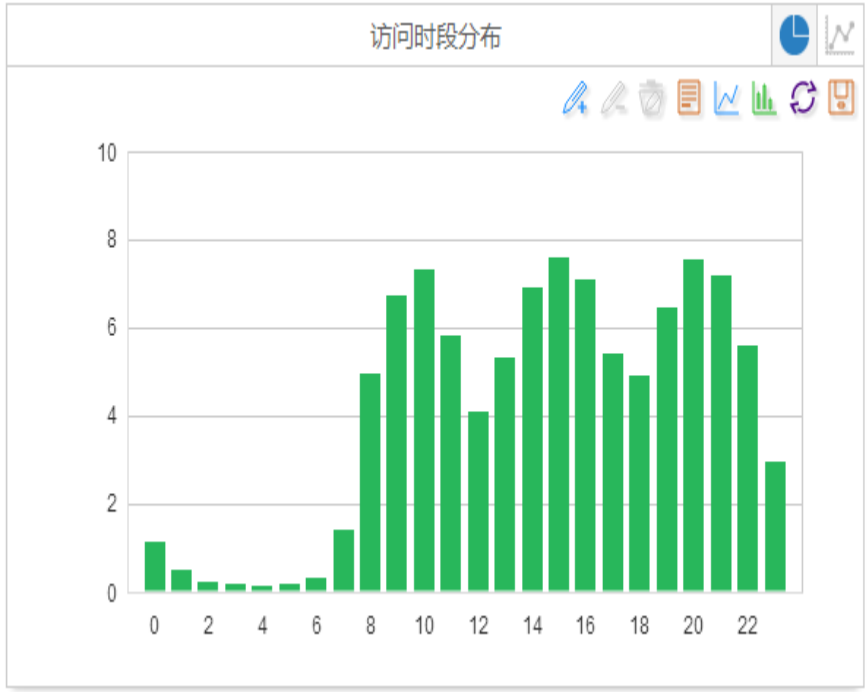
数据展示

PV	750,159,367	PV波动	UV	643,459,874	UV波动
同比变化	+9.37% (2014-03-13)	(2014-03-06~2014-03-20)	同比变化	+8.10% (2014-03-13)	(2014-03-06~2014-03-20)

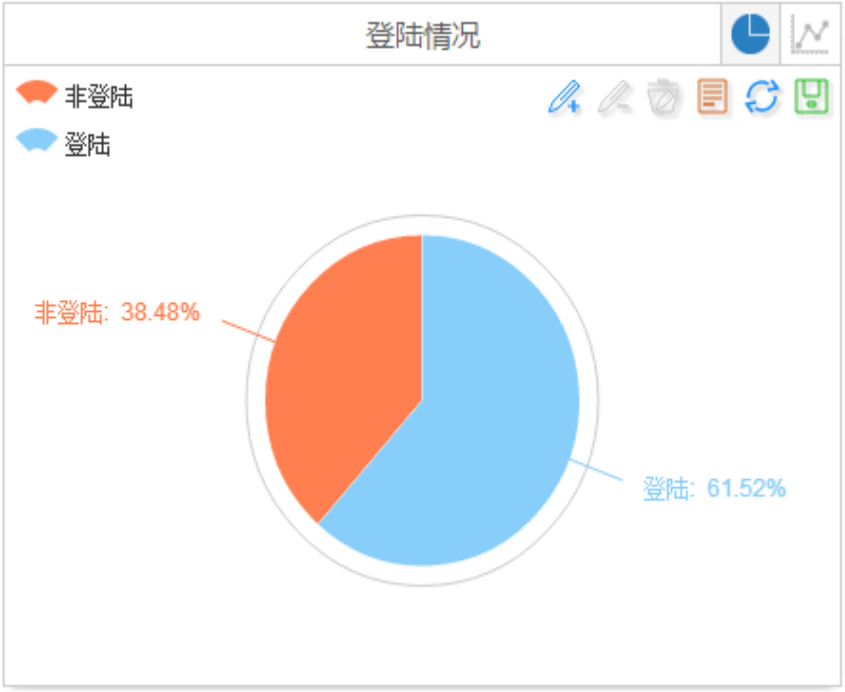


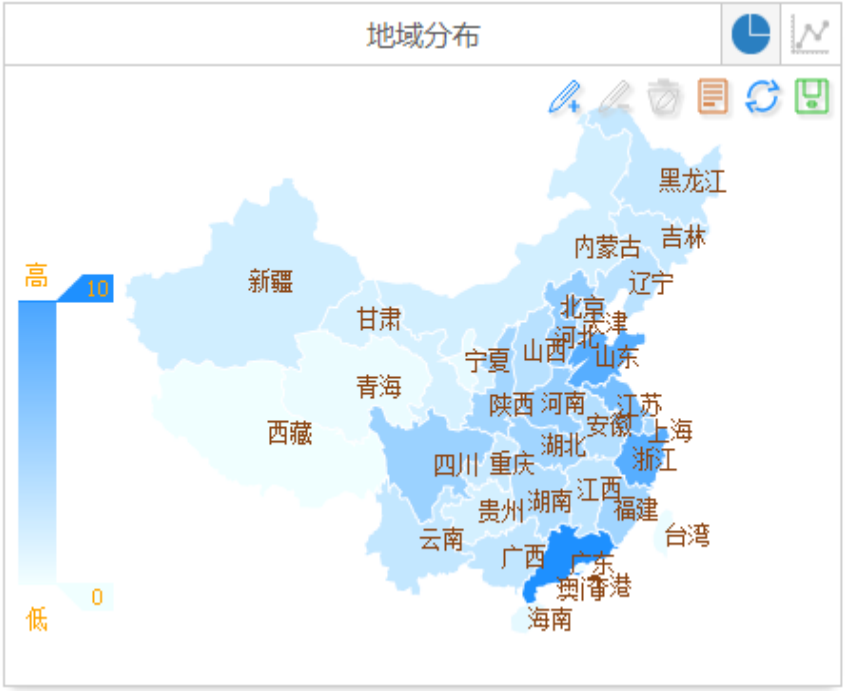
数据展示


浏览器外壳
Chrome
Internet Explorer 8
Sogou Explorer
Internet Explorer 6
Internet Explorer 9
Internet Explorer 11.0
Internet Explorer 10
QQ Browser
Firefox
Maxthon
41.94%
25.28%
7.55%
6.83%
6.24%
3.19%
2.68%
2.06%
1.3%
1.07%
0
9
18
27
36
45


访问时段分布
10
8
6
4
2
0
0
2
4
6
8
10
12
14
16
18
20
22

数据展示


登陆情况
非登陆
登陆
非登陆: 38.48%
登陆: 61.52%


地域分布
高
10
0
低
黑龙江
吉林
内蒙古
辽宁
新疆
甘肃
北京
天津
河北
山西
山东
宁夏
青海
陕西
河南
江苏
安徽
上海
西藏
湖北
浙江
四川
重庆
江西
贵州
湖南
福建
台湾
云南
广西
广东
香港
澳门
海南

② 性能数据

确定核心指标

- 为什么页面打不开？
- 为什么半天页面都没显示出来？
- 为什么按钮点击不了、搜索不了？
- 为什么图片显示这么慢？

时间起点

用户点击一个超链接或者输入url确认开始统计

方式一

使用cookie/hash记录用户点击超链接的时间戳

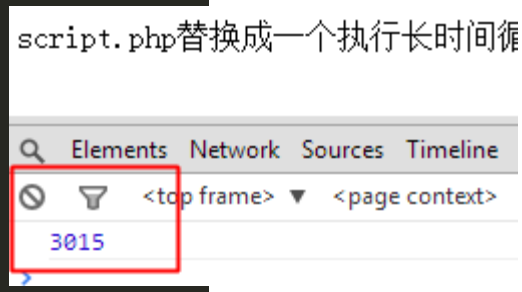
方式二

使用 Navigation Timing API

白屏时间

头部资源下载完成 ≈ 白屏时间

```
        <meta charset="UTF-8"/>
        <script>
            var start_time = +new Date; //测试时间起点，实际统计起点为DNS查询
        </script>
        <!-- 3s后这个js才会返回 -->
        <script src="script.php"></script>
        <script>
            var end_time = +new Date; //时间终点
            var headtime = end_time - start_time; //头部资源加载时间
            console.log(headtime);
        </script>
    </head>
    <body>
        <p>在头部资源加载完之前页面将是白屏</p>
        <p>script.php被模拟设置3s后返回，head底部内嵌JS等待前面js返回后才执行</p>
        <p>script.php替换成一个执行长时间循环的js效果也一样</p>
    </body>
```



head底部内嵌JS来统计头部资源加载 -> 白屏时间

首屏时间

图片是制约首屏的主要因素

获取首屏内图片的加载耗时即可获取首屏时间

首屏时间—统计流程

首屏时间—陷阱

1. 图片加载完成、出错，gif图片重复触发加载事件的处理
2. iframe的处理：同图片
3. **异步渲染**的处理：异步数据插入后再计算首屏
4. css**背景图片**的处理：首屏重要css背景通过JS发起图片请求判断是否已加载
5. 没有图片则以文字出现时间为准，可认为此统计首屏内JS执行时间

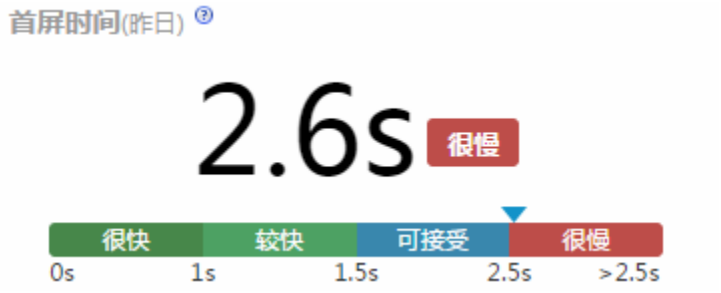
用户可操作时间

DomReady or **核心模块JS加载完毕**(JS中打点标记)

总下载时间

onload or 异步渲染完成

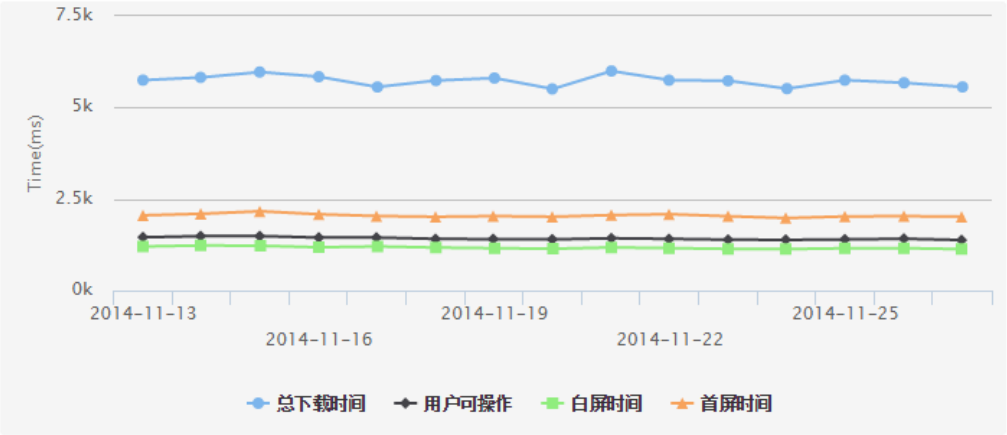
数据展示

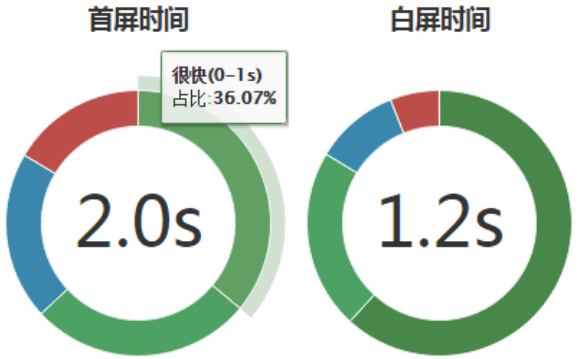

首屏时间(昨日)
2.6s
很慢
很快
较快
可接受
很慢
0s
1s
1.5s
2.5s
>2.5s


页面综合指数(昨日)
6027
用户可接受
↓1.4% 正常波动
很快
较快
可接受
很慢
0
1500
3100
6750
>6750

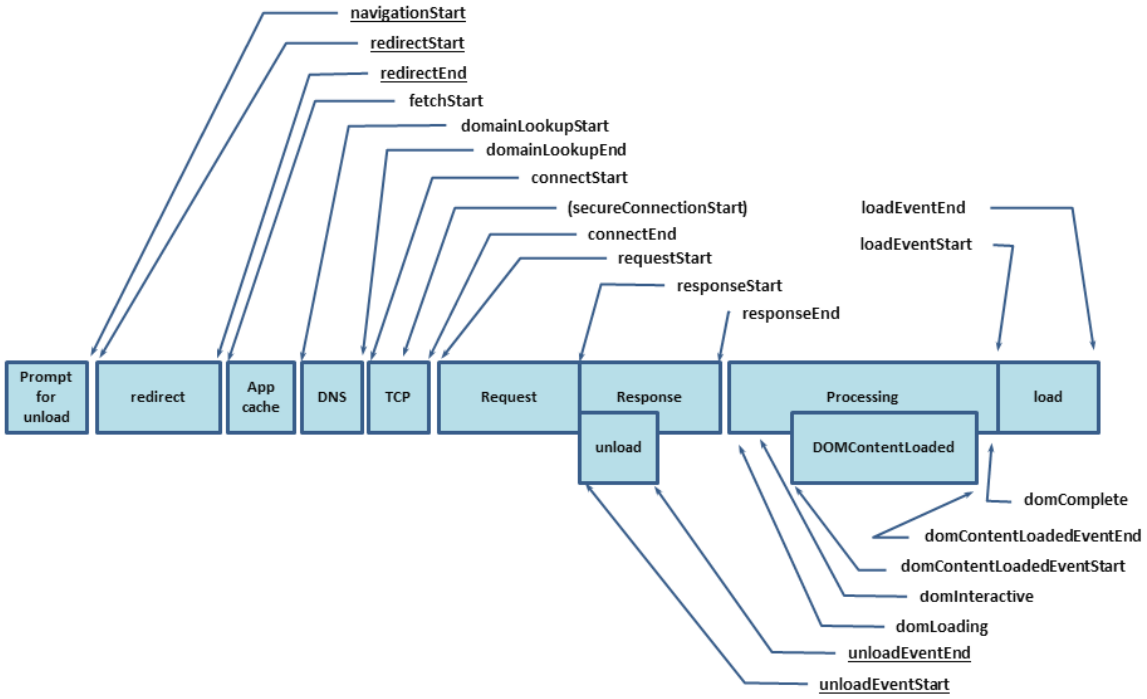

0
1592ms
2574ms
3787ms
11036ms
起点
白屏时间
首屏时间
用户可操作
总下载时间

数据展示


7.5k
5k
2.5k
0k
Time(ms)
2014-11-13
2014-11-16
2014-11-19
2014-11-22
2014-11-25
总下载时间
用户可操作
白屏时间
首屏时间


首屏时间
白屏时间
很快(0-1s)
占比:36.07%
2.0s
1.2s

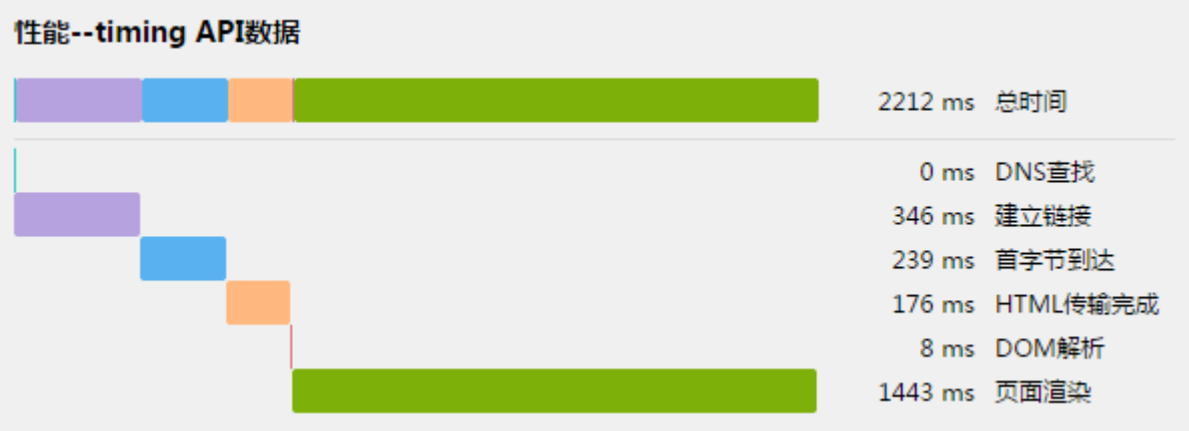
Navigation Timing API




```
> window.performance.timing
▼ PerformanceTiming {loadEventEnd: 13884033503
  1388403350128…}
    connectEnd: 1388403349694
    connectStart: 1388403349667
    domComplete: 1388403350296
    domContentLoadedEventEnd: 1388403350139
    domContentLoadedEventStart: 1388403350128
    domInteractive: 1388403350128
    domLoading: 1388403349759
    domainLookupEnd: 1388403349667
    domainLookupStart: 1388403349663
    fetchStart: 1388403349658
    loadEventEnd: 1388403350311
    loadEventStart: 1388403350310
    navigationStart: 1388403349617
    redirectEnd: 0
    redirectStart: 0
    requestStart: 1388403349694
    responseEnd: 1388403349765
    responseStart: 1388403349744
    secureConnectionStart: 0
    unloadEventEnd: 0
    unloadEventStart: 0
  ▶ proto : PerformanceTiming
```

http://www.w3.org/TR/navigation-timing/

数据展示



Chrome插件：
https://chrome.google.com/webstore/detail/web-performance/dmndepgifbfcjjmkjfgoecldegiodnjc

③ JS异常

初期： window.onerror

优点

- 可以监控到几乎所有JS异常
- 产品线的JS代码无需任何修改

缺点

- 线上JS是混淆压缩的，无行号
- 跨域的JS异常，错误信息是“Script error”
- 跨域的JS异常，获取不到JS文件名

产品线JS的CDN基本都跨域

Script error

PV异常率：0.46%(648,147)　UV异常率：0.88%(544,099)　异常发生次数：5,650,925

异常发生的文件列表：

(1) 未知文件　PV异常率：0.43%(636,688)　UV异常率：0.17%(533,301)　异常发生次数：437,837

③ JS异常

Then： try/catch

优点

- 不怕跨域
- 无惧JS压缩和混淆，捕获压缩前的行号
- 精确定位到模块、方法名

难道要手动给所有JS加try/catch？

a.js

```
var a = {
    init: function(){
        try{
            hello();
        }catch(e){
            alog('exception.send',{
                msg: e.message,
                line: 2
                module: 'a.js',
                method: 'init'
            });
        }
    }
}
```

③ JS异常

我们的做法

- 前端使用FIS开发（fis.baidu.com）
- 使用我们提供的FIS插件（https://github.com/zhangjunah/fis-preprocessor-js-exception-catch）
- JS文件打包时，自动在function内部加上try/catch

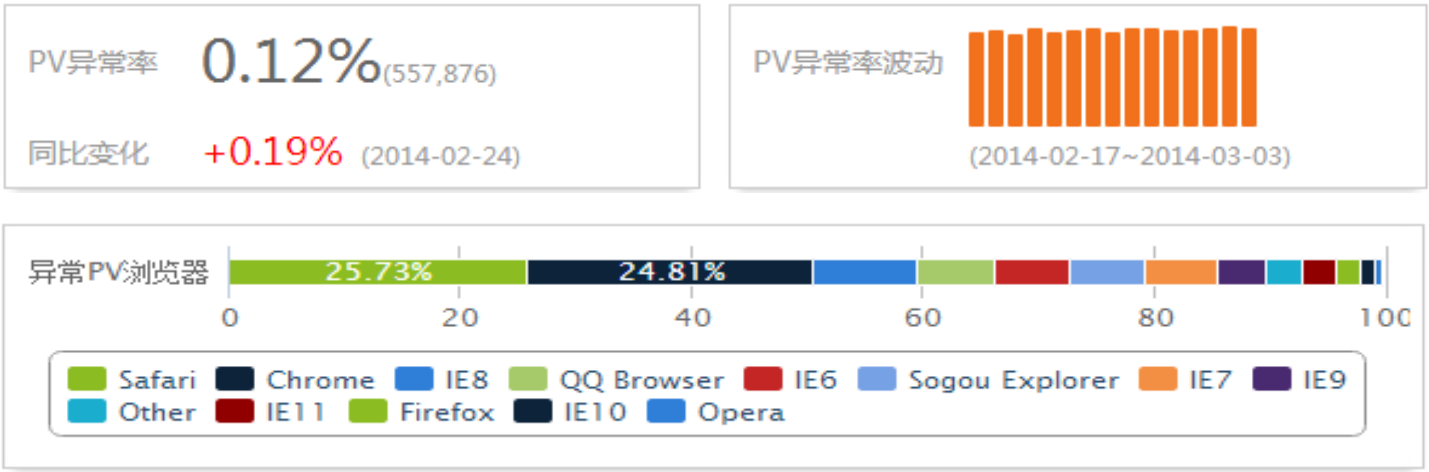
```
//编译前
var slide = {
    init: function(name){
        this.name = name;
    },
    name: ''
}

//编译后
var slide = {
    init: function(name){try{
        this.name = name;
    }catch(e){............}},
    name: ''
}
```

```
//编译前
$('#search').on('click', function(){
    search();
});

//编译后
$('#search').on('click', function(){try{
    search();
}catch(e){............}});
```

数据展示



error test

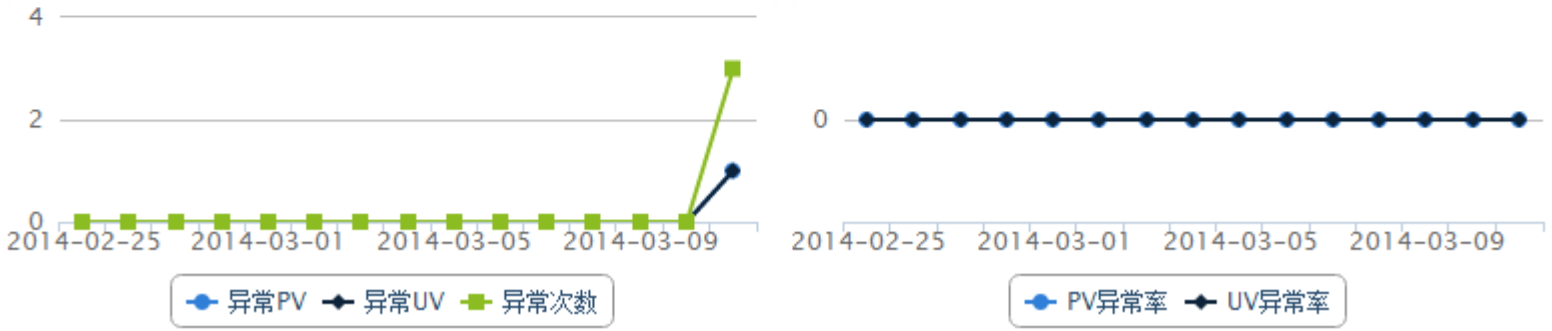
PV异常率：0.00%(1)　UV异常率：0.00%(1)　异常发生次数：3

异常发生的位置列表：

(1) common/export.js　异常method：call　所在行：7　PV异常率：0.00%(1)　UV异常率：0.00%(1)　异常发生次数：3

异常发生的浏览器：Chrome(100%)

最近半月趋势：



④ 其他监控模块

点击数据

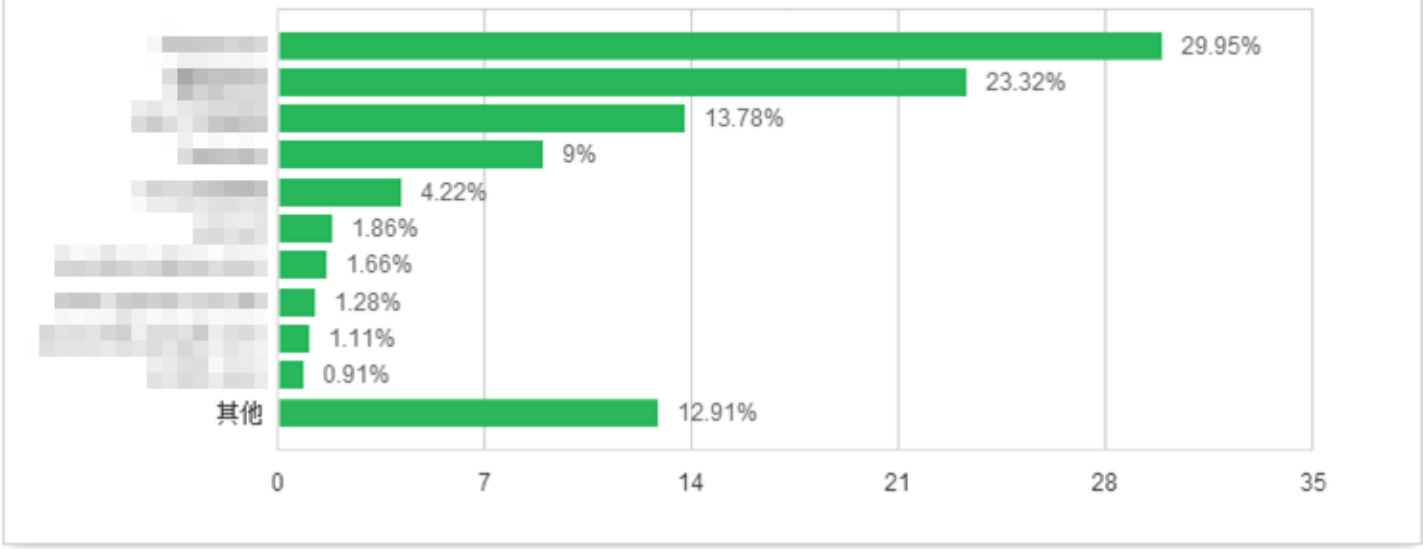
人均点击 10.93

同比变化 +7.82% (2014-03-13)

人均点击波动

(2014-03-06~2014-03-20)

流出url分布



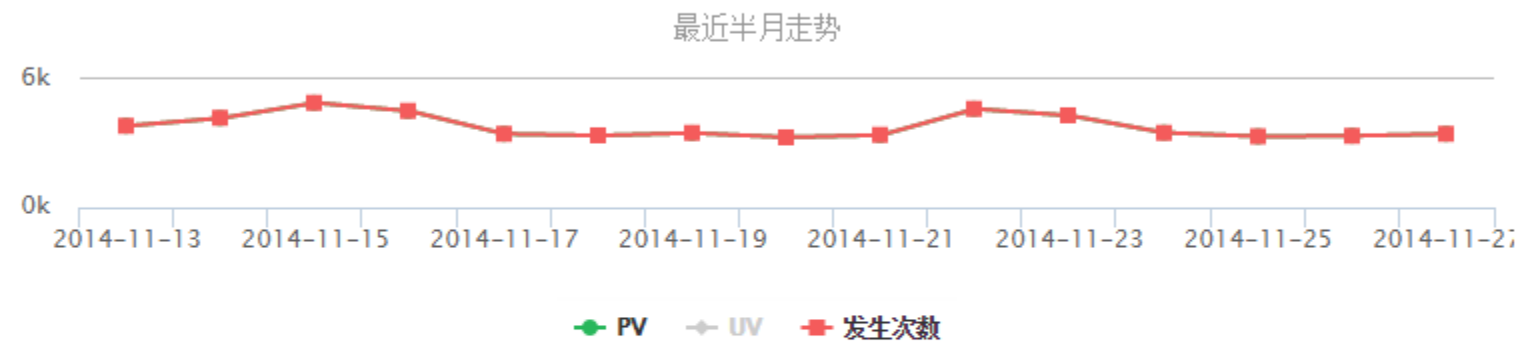
④ 其他监控模块

跨站资源

http://re.taotaosou.com/js/_tts_browser_center.js 淘淘搜比价 修改备注 --

PV：3,444 UV：3,258 发生次数：3,446

操作：查看详细日志



http://202.102.100.100/35ff706fd57d11c141cdefcd58d6562b.js 添加备注 +

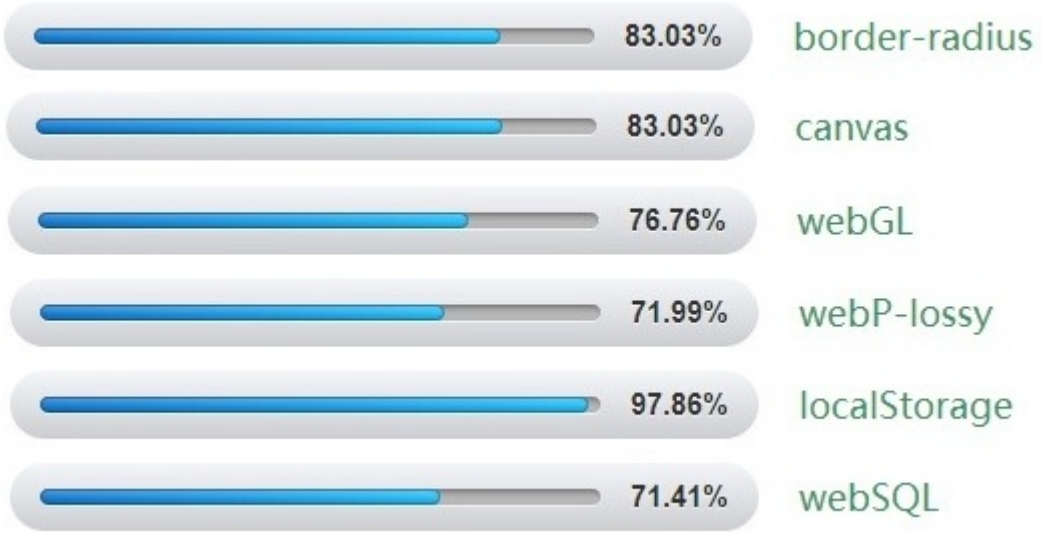
PV：3,036 UV：2,892 发生次数：3,036

http://re.taotaosou.com/browser-static/tmt/sea.js 淘淘搜比价 修改备注 +

PV：2,581 UV：2,443 发生次数：2,581

④ 其他监控模块

HTML5、CSS3支持程度



http://modernizr.com/

本节回顾

1. 统计框架 Alog
2. 监控访问痕迹
3. 性能数据：白屏、首屏、用户可操作、总下载、timing API
4. JS异常捕获：onerror、try/catch、FIS方案
5. 其他监控模块：点击、跨站资源、H5/CSS3支持度

Part 3 FE也能玩大数据

- map/reduce
- http://hadoop.apache.org

- 类SQL语句来执行hadoop任务
- http://hive.apache.org

nodeJS + hadoop

Hadoop支持标准输入输出，nodeJS也同样支持

```
hadoop-szwg-stoff/bin/hadoop jar hadoop-szwg-stoff/contrib/streaming/*streaming*.jar \
-D mapred.map.tasks=10 \
-D mapred.reduce.tasks=1 \
-D mapred.job.name=test001 \
-cacheArchive 'hdfs://szwg-stoff-hdfs.dmop.baidu.com:54310/node-v0.8.4-linux-x64.tar.gz#nodejs' \
-file map.js \
-file reduce.js \
-mapper map.js \
-reducer reduce.js \
-input test001.log \
-output result
```

nodeJS + hadoop

在map.js、reduce.js中

```
// 表明运行环境为nodejs
#!nodejs/bin/node

// 标准输入输出
var stdin = process.stdin;
var stdout = process.stdout;

// 处理每行数据
function eachLine(line) {
    // 这里以pv数据为例
    if (line.type == 'pageview') {
        // tieba-index \t pageview
        stdout.write(line.pageId + '\t' + 'pageview' + '\n');
    }
}

// 读数据流
stdin.on('data', function(chunk) {
    // 将chunk拆为每行
    for ( ......) {
        eachLine(line);
    }
});

// 读数据流结束
stdin.on('end', function() {
    console.log('hi hi,  read end!!!');
});
```

Hive

创建数据表、导入hadoop格式化后的数据

```
CREATE TABLE data_all (
    pageId STRING,
    type STRING
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;

LOAD DATA INPATH '/result' OVERWRITE INTO TABLE data_all;
```

查询所有页面的pv

```
SELECT pageId, count(*) AS pv
FROM data_all
GROUP BY pageId
```

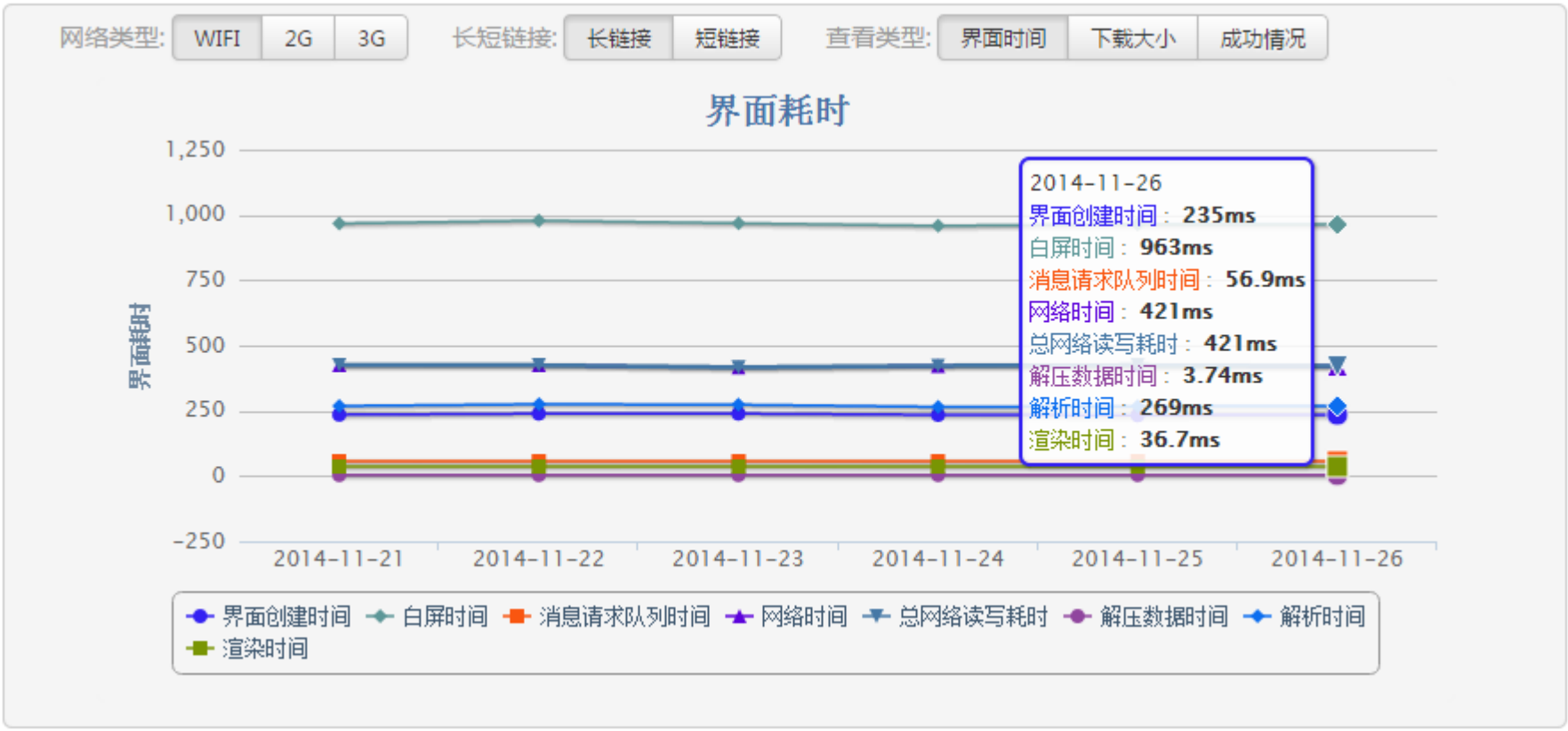
Part 4 我们还能做什么

非侵入式监控

竞品对比分析、全互联网网页性能排名、命令行工具 等

http://siteArchive.baidu.com

native监控


网络类型:
WIFI
2G
3G
长短链接:
长链接
短链接
查看类型:
界面时间
下载大小
成功情况
界面耗时
1,250
1,000
750
500
250
0
-250
界面耗时
2014-11-21
2014-11-22
2014-11-23
2014-11-24
2014-11-25
2014-11-26
2014-11-26
界面创建时间 : 235ms
白屏时间 : 963ms
消息请求队列时间 : 56.9ms
网络时间 : 421ms
总网络读写耗时 : 421ms
解压数据时间 : 3.74ms
解析时间 : 269ms
渲染时间 : 36.7ms
界面创建时间
白屏时间
消息请求队列时间
网络时间
总网络读写耗时
解压数据时间
解析时间
渲染时间

前后端监控结合

1. 前、后端性能统一监控，形成完整的time line
2. 前、后端异常统一监控，快速定位bug

Part 5 总结

1. 监控方式：非侵入式、侵入式
2. 数据采集：Alog框架、访问、性能、JS异常等
3. 玩转大数据：nodeJS + hadoop 、hive
4. 下一步方向：非侵入式工具、native APP、前后端监控结合

Q&A

THANKS