

berserkJS

@貘吃馍香

2012-07-07 于杭州 D2

D2



- 工具由来（需求）
- 它能做什么
- 现有工具对比
- 一些特性例子（相对 PhantomJS ）
- 如何得到它
- 怎么使用它
 - 使用预制的模块配置功能
 - 使用自定义的实现方式
- 实现原理

人生不如意事 十之八九 T_T

尼玛！ 手工收集 HAR 的事儿
(HTTP Archive)

太 (粗口) 枯燥了
(粗口) 旧港

D2

- 收集数据方法之 Chrome / Safari :

The screenshot shows the Chrome DevTools Network tab. On the left, a list of resources is shown with icons for HTML, CSS, and JS files. A context menu is open over one of the resources, with the "Save all as HAR" option highlighted by a red box. A large red arrow points from the text "可保存HAR数据" (Can save HAR data) to this highlighted option. On the right, the HAR (HTTP Archive) data is displayed as a JSON object, showing the page's metadata and a list of network requests.

可保存HAR数据

```
1 {  
2   "log": {  
3     "version": "1.2",  
4     "creator": {  
5       "name": "WebInspector",  
6       "version": "536.4"  
7     },  
8     "pages": [  
9       {  
10        "startedDateTime": "2012-03-27T06:34:06.951Z",  
11        "id": "page_1",  
12        "title": "http://weibo.com/itapir#1332822445734",  
13        "pageTimings": {  
14          "onContentLoad": 1457,  
15          "onLoad": 1491  
16        }  
17      }  
18    ],  
19    "entries": [  
20      {  
21        "startedDateTime": "2012-03-27T06:34:06.951Z",  
22        "time": 1416,  
23        "request": {  
24          "method": "GET",  
25          "url": "http://weibo.com/itapir",  
26          "httpVersion": "HTTP/1.1",  
27          "status": 200,  
28          "statusText": "OK",  
29          "contentLength": 26810  
30        }  
31      }  
32    ]  
33  }  
34 }
```

- 收集数据方法之 IE :

The screenshot shows the Fiddler application window. At the top, the menu bar includes: 文件(F), 查找(N), 禁用(S), 查看(V), 图像(I), 缓存(C), 工具(T), 验证(A), 浏览器模式(B): IE9, 文档模式: IE7 标准(M). Below the menu is a toolbar with buttons for HTML, CSS, 控制台, 脚本, 探查器, and 网络. The "网络" button is highlighted with a blue border. A red box highlights the "开始捕获" (Start Capturing) button, which is also highlighted with a blue border. To its right is a "转到详细视图" (Switch to Detailed View) button. The main area displays a table of network requests:

URL	方法	结果	类型	已接收	已花费	发起程序	计时
http://weibo.com/?retcode=6102	GET	200	text/html	<?xml version="1.0" encoding="UTF-8"?>			
http://img.t.sinajs.cn/t35/style/...	GET	304	text/css	<log>			
http://js.t.sinajs.cn/t35/miniblo...	GET	304	application	→<version>1.1</version>			
http://tp3.sinaimg.cn/1195354434/...	GET	304	image/jpeg	→<creator>			
http://tp3.sinaimg.cn/1361024910/...	GET	200	image/jpeg	→→<name>Internet Explorer 网络检查器</name>			
http://tp3.sinaimg.cn/1574525494/...	GET	304	image/jpeg	→→<version>9.0.8112.16421</version>			
http://tp2.sinaimg.cn/1682804733/...	GET	200	image/jpeg	→</creator>			
http://tp4.sinaimg.cn/1563815015/...	GET	304	image/jpeg	<browser>			
http://tp2.sinaimg.cn/1755370981/...	GET	200	image/jpeg	→<name>Internet Explorer</name>			
http://tp1.sinaimg.cn/2150743924/...	GET	304	image/jpeg	→<version>9.0.8112.16421</version>			
http://tp1.sinaimg.cn/1191220232/...	GET	304	image/jpeg	→</browser>			
http://tp2.sinaimg.cn/2672831597/...	GET	304	image/jpeg	<pages>			
http://rs.sinajs.cn/mini.gif?_=w1...	GET	200	image/gif	→<page>			

A red arrow points from the "开始捕获" button to the XML code on the right, with the text "可保存HAR(XML)数据" (Can save HAR(XML) data) written over it.

```
<?xml version="1.0" encoding="UTF-8"?>
<log>
  <version>1.1</version>
  <creator>
    <name>Internet Explorer 网络检查器</name>
    <version>9.0.8112.16421</version>
  </creator>
  <browser>
    <name>Internet Explorer</name>
    <version>9.0.8112.16421</version>
  </browser>
  <pages>
    <page>
      <startedDateTime>2012-03-27T06:41:58.59+08:00</startedDateTime>
      <id>0</id>
      <title/>
      <pageTimings>
        <onContentLoad>872</onContentLoad>
        <onLoad>909</onLoad>
      </pageTimings>
    </page>
  </pages>
  <entries>
    <entry>
      <pageref>0</pageref>
      <startedDateTime>2012-03-27T06:41:58.59+08:00</startedDateTime>
      <time>296</time>
    </entry>
  </entries>
</log>
```

• 收集数据方法之 Firebug :

The screenshot shows the Firebug Network panel with the following details:

- Panel Tabs:** Control台, HTML, CSS, 脚本, DOM, 网络 (selected), YSlow.
- Sub-Panel Tabs:** XHR, 清除, 保持, 所有 (selected), HTML, CSS, JS, XHR, 图片, Flash, 媒体.
- Table Headers:** URL, 状态, 域, 大小, 远程 IP, 时间线.
- Message Bar:** 网络面板已激活. 在面板未激活时的任何请求都不会被显示.
- Request List:** GET weibo.com (200 OK, weibo.com, 10.2 KB, 123.125.104.197:80, 139ms), GET tlogin_1.css?version=201203 (200 OK, img.t.sinajs.cn, 47.7 KB, 237ms). Below these are several requests from Google (www.google.cz) with various response codes (200, 204 No Content) and sizes (e.g., 2 KB, 7 KB, 5 KB, 6 KB, 22 KB total).
- Toolbar Buttons:** Clear, Export (circled in red with an arrow pointing to it), All, HTML, CSS, JS, XHR, Images, Flash.
- Message Bar (Bottom):** Net panel activated. Any requests while the net panel is inactive are not shown.

*加装 NetExport 来收集 HAR 数据



坑爹呢~这是!!



FlashSoft

能抓自动页面请求数据么



FlashSoft

能在浏览器里操作抓部分数据么



FlashSoft

并且还能远程控制调用的



FlashSoft

能程序控制点按钮模拟操作么



教主

我的 Mac 能用么，可用 JS 编程控制的

咱可以用代理抓数据

偶



做个浏览器 plugin

偶



命令行化调用浏览器 + plugin.....

偶



= =||| 再辅助挂个按键精灵没准成

偶



你妹.....



berserkJS

这货能做什么？

D2

- berserkJS 是基于 Qt (C++ 跨平台库) 开发的前端网络 (性能) 监测工具。
- 它的核心功能是通过内置 webkit 收集由页面实际网络请求相关数据。
- 偏重于**页面上线前检测与评估**。

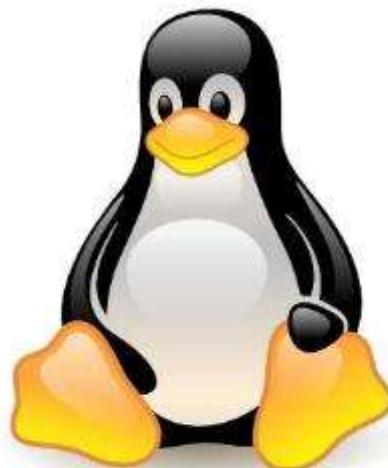
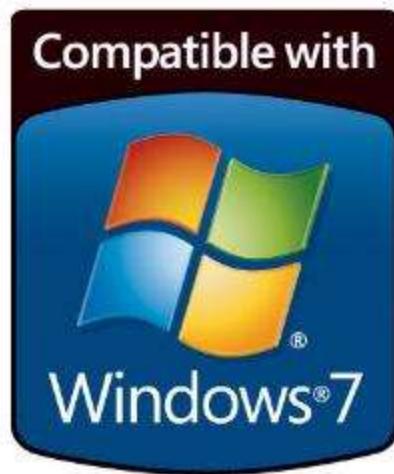
大致可以实现以下功能：

- 监测页面的网络请求，收集目标数据
- 首次渲染时间与首屏渲染时间监控
- 操作页面运行沙箱内 DOM 对象与 JS
- 模拟用户鼠标操作
- 操作内置 WebKit 浏览器
- 页面截图与文件读写
- HTTP 请求与启动外部进程操作等

同时它具有跨平台特性：

基于 Qt 可跨 windows 、 linux、 Mac OS 平台运行。

* 一份源码随处编译，无需修改。



我们常用的一些性能检测工具：



Boomerang

(开源)



Performance API

(规范)



PhantomJS

(开源)

Yahoo! boomerang :

- boomerang 项目 : <http://yahoo.github.com/boomerang/>
- 支持 IE8+ 以及其他浏览器

Performance API :

- Performance API 现处于草案阶段 <http://www.w3.org/TR/performance-timeline/>
- 其中 NavigationTiming API 部分，已经被 Chrome7+、IE9+、FF7+ 支持
<http://w3c-test.org/webperf/specs/NavigationTiming/>

*PS：如果要检测页面内所有资源，已经在制定 ResouceTiming API。

URL : <http://w3c-test.org/webperf/specs/ResourceTiming/> 但是，现在还没有浏览器实现。

Performance API	boomerang
仅检测被访问的页面文件自身情况	仅检测被访问的页面文件自身情况
各个时间需用 connectEnd-connectStart 之类的方法得到	依赖 BOM DOM事件估算请求时间，数据由脚本自动计算获得
单页性能数据非常详细	单页性能数据一般
需要将检测脚本与回传脚本放入生产环境	需要将检测脚本放入生产环境
注重上线后页面性能数据收集与分析	注重上线后页面性能数据收集与分析

aoao这货说的

成银这货说的

据说 淘宝 与 百度 在使用
PhantomJS

用 berserkJS 的好多坨例子做对比

与 PhantomJS 的区别：

- PhantomJS 项目 : <http://www.phantomjs.org/>
- 非客户端 API , 工具本身跨平台。

berserkJS	PhantomJS
使用 JS 控制 webkit	使用 JS 控制 webkit
可以操作 webkit 内当前页面内 DOM/JS 等	可以操作 webkit 内当前页面内 DOM/JS 等
可模拟用户操作 (Mouse Event API)	可模拟用户操作 (Mouse Event API)
File System API	File System Module
无	WebServer Module
内置实现 , 只需获取数据 , 使用更简单。	Network Event callback (稍复杂)
页面渲染以及布局事件监听	无 (暂时)
页面截图、区域截图、截图base64转换	全页面截图 (1.6 支持base64转换)
获取 CPU 与 内存 占用率	无
GUI 模式 与 模拟的命令行模式	命令行模式
更直接的 API	commonJS 规范
检测代码无需上线 任意时间可评估页面性能	检测代码无需上线 任意时间可评估页面性能



相比 PhantomJS
数据收集方法更简单

- PhantomJS 收集数据方法：

```
var page = require('webpage').create(), fs = require('fs'), content = '';
page.onLoadStarted = function () {
    page.onResourceRequested = function (request) {
        content += 'Request ' + JSON.stringify(request, undefined, 2);
    };
    page.onResourceReceived = function (response) {
        content += 'Receive ' + JSON.stringify(response, undefined, 2);
    };
}
page.onLoadFinished = function () {
    fs.write('c:\\a.txt', content, 'w')
};
page.open('http://www.taobao.com');
// 还有下载时间没计算呢，写不下不写了.....
// 额，好像 DNS 、 Waitting 时间啥的没法计算 = =|||
// phantomJS 的例子文件 netsniff.js 里 DNS 啥的时间写的都是 -1 .....
// 不过写了 130 行里有一半是为了拼 HAR 格式数据
```

- berserkJS 收集数据方法：

命令行：berserkJS --script=demo1.js --command=true

```
1 // 开启监听
2 App.netListener(true);
3 // 访问页面
4 App.webview.open("http://www.taobao.com");
5 // 页面加载完成后获取数据
6 App.webview.addEventListener('load', function() {
7     // 获取数据并序列化
8     var data = JSON.stringify(App.networkData(), undefined, 2);
9     // 写入文件
10    App.writeFile(App.path + "demo1.txt", data);
11    // 关闭监听
12    App.netListener(false);
13    // 退出应用
14    App.close();
15});
```

- berserkJS 收集数据方法：

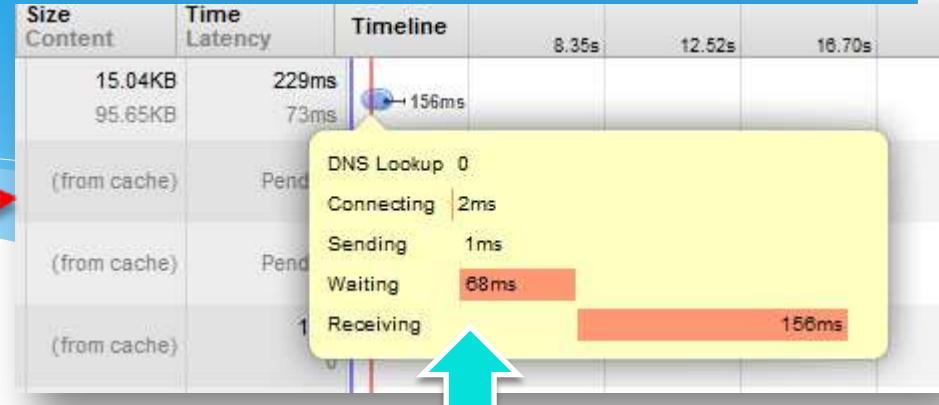
```
1 [  
2 {  
3     "StatusCode": 200,  
4     "ReasonPhrase": "OK",  
5     "FromCache": false,  
6     "url": "http://4823.kf92cbgj8xwcaydenlesmae.2031255175.dns.greencomputing.com",  
7     "RequestStartTime": 1341133203001,  
8     "RequestEndTime": 1341133206909,  
9     "ResponseSize": 29555319235608692,  
10    "ResponseDuration": 3908,  
11    "ResponseWaitingDuration": 3906,  
12    "ResponseDownloadDuration": 2,  
13    "ResponseDNSLookupDuration": 0,  
14    "RequestMethod": "GET",  
15    "UserAgent": "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/534.34 (KHTML, like Gecko) Chrome/12.0.742.112 Safari/534.34",  
16    "Accept": "*/*",  
17    "Referer": "http://www.taobao.com/",  
18    "AcceptRanges": "bytes",  
19    "Age": "",  
20    "AccessControlAllowOrigin": "",  
21    "CacheControl": "",  
22    "Connection": "close",  
23    "ContentType": "application/octet-stream",  
24    "ContentLength": 0,  
25    "ContentEncoding": ""},
```

* 自定义格式，扁平的 JSON 结构，就是个大数组对象。

• berserkJS 的 network 数据项：

App.networkData()[0] (单请求数据) :

- "url": < string >
- "ResponseSize": <number>
- "RequestStartTime": <number>
- "ResponseDuration": <number>
- "ResponseDNSLookupDuration": <number>
- "ResponseWaitingDuration": <number>
- "ResponseDownloadDuration": <number>
- "RequestMethod": < string >
-
- "StatusCode": <string>
- "Accept": <string>
- "Cookie": <string>
-



它们分别对应浏览器输出数据

Request URL: http://weibo.com/comment/inbox?f=1&topnav=1&wvr=4
Request Method: GET
Status Code: 200 OK
▼ Request Headers [view source](#)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*
Accept-Charset: GBK,utf-8;q=0.7,*;q=0.3
Accept-Encoding: gzip,deflate,sdch
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.6,en;q=0.4
Connection: keep-alive
Cookie: NSC_wjq_xfjcp.dpn_ipnfqbhf=fffffffff094113a345525d5f4f584550194; ULV=1332936550198:1:1:1:9342593455221.504.1332936550194;%26rs%3DoCxDpyGOHhKGiO%252F80Pw75bQ2bM6qU4HRUrHfxo1nYlnr3Uxmp19qSGmURqPPQDjJaUJgQNdHdtufwYCBFLi1Sty6R1M%252BE%253D%26rv%3D0; S0%26ac%3D0%26uid%3D1660579792%26user%3Ditemadetemade.%252A%252A%202021%253A25%253A20; SUS=SID-1660579792-1332936584-XD-6a9iu-495a8@sina.com; NSC_wjq_xfjcp.dpn_w3.6_w4=fffffffff0941135e45525d5f4f5912.1332998927.1332998927.1332998927.1; __utmc=182865017; __utmz ads_ck=0; SINAGLOBAL=9342593455221.504.1332936550194

默认一条请求包含 72 项常见数据

如果有其他头信息则还会更长



相比 PhantomJS
内置 selector 方法集
更易做数据筛选

- berserkJS 的 network 数据选择工具

选择数据集后使用 App.selector.get() 方法可以返回指定数据集

- App.selector.img()
- App.selector.png()
- App.selector.gif()
- App.selector.ico()
- App.selector.jpg()
- App.selector.svg()
- App.selector.doc()
- App.selector.css()
- App.selector.js()
- App.selector.cookie()
- App.selector.nonegzip()
- App.selector.nonecache()
- App.selector.nonecdn()
- App.selector.totaltimeout(duration <number>)
- App.selector.waittimeout(duration <number>)
- App.selector.downloadtimeout(duration <number>)
- App.selector.dnstimeout(duration <number>)
- App.selector.sizeout(size <number>)
- App.selector.http200()
- App.selector.http301()
- App.selector.http302()
- App.selector.http304()
- App.selector.http404()
- App.selector.fromcdn() //仅判断了 sina 的 CDN

它们实现了浏览器内类似筛选功能



- berserkJS 的 network 数据选择工具

命令行 : berserkJS --script=demo2.js

```
1 // 开启监听
2 App.netListener(true);
3 // 访问页面
4 App.webview.open('http://www.taobao.com');
5 // 页面加载完成后获取数据
6 App.webview.addEventListener('load', function() {
7     // 选择 png 格式图片
8     App.selector.png();
9     // 并且大于 30k 的
10    App.selector.sizeout(1024*30);
11    // 获取数据并序列化
12    var data = JSON.stringify(App.selector.get(), undefined, 2);
13    // 写入文件
14    App.writeFile(App.path + 'demo2.txt', data);
15    // 关闭监听
16    App.netListener(false);
17    // 退出应用
18    App.close();
19});
```

* 可连续调用 : App.selector.png().sizeout(1024 * 30).get();

- berserkJS 的 network 数据选择工具

```
1 [  
2 {  
3     "StatusCode": 200,  
4     "ReasonPhrase": "OK",  
5     "FromCache": true,  
6     "url": "http://img04.taobaocdn.com/tps/i4/T1l46rXX4jXXa6XtY7-490-170.png",  
7     "RequestStartTime": 1341597132248,  
8     "RequestEndTime": 1341597132362,  
9     "ResponseSize": 30971,  
10    "ResponseDuration": 114,  
11    "ResponseWaitingDuration": 43,  
12    "ResponseDownloadDuration": 69,  
13    "ResponseDNSLookupDuration": 2,  
14    "RequestMethod": "GET",  
15    "UserAgent": "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/534.34 (KHTML, like Gecko)",  
16    "Accept": "*/*",  
17    "Referer": "http://www.taobao.com/",  
18    "AcceptRanges": "",  
19    "Age": "52466",  
20    "AccessControlAllowOrigin": "",  
21    "CacheControl": "max-age=315360000",
```



相比 PhantomJS
更全面的脚本变量出入页面沙箱功能

- PhantomJS 的 page.evaluate 方法暂时只能出沙箱不能入沙箱

```
console.log('Page title is ' + page.evaluate(function () {  
    return document.title;  
}));
```

- berserkJS 的 webview.execScript 可以使用 JSON 出入页面脚本沙箱

```
var site = {topTen: 5, url: 'taobao'}  
sit = App.webview.execScript(function (obj) {  
    return {topTen: obj.topTen - 2, url: 'http://www.' + obj.url + '.com'};  
}, site);  
console.log(JSON.stringify(sit));
```

输出 : { "topTen": 3, "url": "http://www.taobao.com" }

- 基于 `_pageExtension.postMessage` 与 `message` 事件的异步出沙箱

```
// 在工具内监听 message 事件
App.webview.addEventListener('message', function(w, l) {
    if ('page' == l) {
        // 显示 "this is a message" 信息。
        alert(w.txt);
    }
});

App.webview.execScript(function(s) {
    // 异步触发事件，传送数据出页面沙箱
    setTimeout(function() {
        _pageExtension.postMessage({txt: "this is a message"}, "page");
    }, 1000);
});
```

* 内部使用 `JSON.stringify` 和 `JSON.parse` 来转换进出沙箱的 object 。所以，你懂的.....

- 基于其它方法的异步数据出沙箱
- consoleMessage 事件 或 alert / confirm 等事件

```
App.webview.addEventListener('consoleMessage', function(msg, lineNumber, sID) {  
    alert(JSON.parse(msg).txt); // 通过监听页面控制台输出达到目的  
});
```

```
App.webview.addEventListener('alert', function(msg, lineNumber, sID) {  
    alert(JSON.parse(msg).txt); // 通过监听页面 alert 方法输出达到目的  
});
```

```
App.webview.addEventListener('confirm', function(msg) {  
    alert(JSON.parse(msg).txt); // 通过监听页面 confirm 方法输出达到目的  
});
```

```
App.webview.execScript(function(s) {  
    // 异步触发事件，传送数据出页面沙箱  
    setTimeout(function() {  
        var jsonString = JSON.stringify({txt: "this is a message"});  
        alert(jsonString);  
        confirm(jsonString);  
        console.log(jsonString);  
    }, 1000);  
});
```



相比 PhantomJS

可使用区域截图与截图并base64功能

- 区域截图：

phantomJS 现有版本暂时没有此功能。

命令行：berserkJS --script=demo3.js --command=true

```
1 // 访问页面
2 App.webview.open("http://www.taobao.com");
3 // 监听文档加载完成
4 App.webview.addEventListener("load", function(rect) {
5     // 截取登录表单区域保存为图片
6     App.webview.saveImage(App.path + 'demo3.png', 'png', 100,
7     App.webview.elementRects("#J_CategoryHover")[0]);
8     App.close();
9 });
10
```

- 区域截图：

虚拟

磨拳擦掌抢公仔 竞猜2012欧洲杯

话费充值 移动 联通 电信 宽带

网上营业厅 3G上网 号码 合约机

游戏 dnf 魔兽 桃园 dn 火影

点卡 魔兽 梦幻西游 CF 征途 QQ

彩票 双色球 3D 15选5 七乐彩

机票 酒店 客栈 旅游 门票 国际票

服装

女装新版首页全新上线

女装 连衣裙 T恤 雪纺 衬衫 半裙

夏装 裤子 牛仔 针织 背心 胖MM

男装 T恤 衬衫 polo 休闲裤 短裤

商务 夏装 卫衣 工装裤 牛仔裤

内衣 文胸 睡衣 内裤 丝袜 男士

童装 连衣裙 T恤 套装 裤子 童鞋

鞋包配饰

鞋类品牌直销，全场3折起！

女鞋 新品 凉鞋 凉拖 单鞋 帆布鞋

女包 新品 真皮 单肩 钱包 手提

男鞋 休闲 时尚 帆布 凉鞋 拖鞋

男包 单肩 钱包 手提 休闲 真皮

配饰 丝巾 帽子 腰带 防晒 遮阳

旅行箱包 双肩 旅行箱 包 大牌

运动户外

2012泳装新款，全场包邮！

运动鞋 板鞋 篮球 跑步 帆布 包

运动服 卫衣 套装 新品 T恤 裤

装备 跑步机 健身 游泳 骑行 羽球

户外 洞洞鞋 速干 背包 强光 帐篷

凯乐石 耐克 李宁 捷安特 adivon

垂钓 军迷 热舞 暴走 瑜伽 海钓行

珠宝手表

夏日清凉美饰，戴上专属你的美丽

珠宝 钻石 黄金 铂金 施华洛

品牌手表 卡西欧 天梭 浪琴

翡翠 珍珠 琥珀 珊瑚 宝石 碧玺

时装表 果冻表 水钻表 复古表

饰品 项链 手链 发饰 耳饰 手镯

眼镜 太阳镜 眼镜架 zippo 烟具

- 区域截图的 base64 化：

命令行：berserkJS --script=demo4.js

```
1 // 访问页面
2 App.webview.open('http://www.taobao.com/');
3 // 监听文档加载完成
4 App.webview.addEventListener('load', function(rect) {
5     // 截取区域返回 base64
6     var base64 = App.webview.dataURIFromRect({
7         x:500,
8         y:200,
9         width:200,
10        height:200
11    });
12    // 将此图片加入文档中显示
13    App.webview.execScript(function(base64){
14        var img = document.createElement('img');
15        img.src = base64;
16        document.getElementById('site-nav').appendChild(img);
17    }, base64); // 数据入页面沙箱
18 });
19
```

- 区域截图的 base64 化：





相比 PhantomJS 更多的页面性能相关事件与方法

- 页面渲染
- 页面首次渲染时间
- 页面首屏时间
- CPU与内存占用

- 页面渲染监控，显示 repaint 次数：

命令行：berserkJS --script=demo5.js

```
1 // 访问页面
2 App.webview.open("http://www.taobao.com");
3 // 渲染计数
4 var repaintCount = 0;
5 // 监听文档渲染
6 App.webview.addEventListener("repaint", function(rect) {
7   rect.repaintCount = repaintCount++;
8   // 进入页面沙箱执行页面脚本
9   App.webview.execScript(function(rect) {
10     // 控制台输出，页面输出会触发 repaint，然后..... 你懂的.....
11     console.log("repaint count:" + rect.repaintCount,
12       "repaint rect:",
13       "x: " + rect.x,
14       "y: " + rect.y,
15       "width: " + rect.width,
16       "height: " + rect.height
17       );
18   }, rect);
19 });
20 }
```

- 页面渲染性能，显示 repaint 次数：

The screenshot shows the Taobao homepage with a browser developer tools window overlaid. The tools window is titled "Web Inspector - http://www.taobao.com/" and displays the DOM tree for the page. A specific element, the search bar, is selected in the tree view. The bottom pane of the inspector shows a list of repaint events. The text in the bottom pane is as follows:

```
repaint count:900 repaint rect: x: 201 y: 234 width: 488 height: 168
repaint count:901 repaint rect: x: 201 y: 234 width: 488 height: 168
repaint count:902 repaint rect: x: 201 y: 234 width: 488 height: 168
repaint count:903 repaint rect: x: 201 y: 234 width: 488 height: 168
repaint count:904 repaint rect: x: 201 y: 234 width: 488 height: 168
repaint count:905 repaint rect: x: 700 y: 233 width: 108 height: 50
repaint count:906 repaint rect: x: 700 y: 233 width: 108 height: 50
repaint count:907 repaint rect: x: 700 y: 233 width: 108 height: 50
repaint count:908 repaint rect: x: 201 y: 234 width: 488 height: 168
repaint count:909 repaint rect: x: 201 y: 234 width: 488 height: 168
```

- 页面首次渲染时间的获得

命令行：berserkJS --script=demo6.js

```
1 // 打开目标页
2 App.webview.open('http://www.taobao.com');
3 // 监听首次渲染事件
4 App.webview.addEventListener('firstPaintFinished', function(timeout, url) {
5   this.execScript(function (o) {
6     var node = document.createElement('div');
7     var cssText = 'font-size: 14px; background:yellow; border: 1px solid #000;';
8     cssText += 'position: fixed; top: 0; left:0; z-index:99999;';
9     node.style.cssText = cssText;
10    node.innerHTML = '<ul><li>' + o.timeout + ' ms</li>' +
11      '<li> ' + o.url + '</li></ul>';
12    document.body.appendChild(node);
13  }, {timeout: timeout, url: url});
14 });
15 }
```

- 页面首次渲染时间的获得



- 页面首屏（当前视口）渲染时间的获得

命令行：berserkJS --script=demo7.js

```
1 // 打开目标页
2 App.webview.open('http://www.taobao.com');
3 // 监听首屏渲染事件
4 App.webview.addEventListener('firstScreenFinished', function(timeout, url) {
5   this.execScript(function (o) {
6     var node = document.createElement('div');
7     var cssText = 'font-size: 14px; background:yellow; border: 1px solid #000;';
8     cssText += 'position: fixed; top: 0; left:0; z-index:99999;';
9     node.style.cssText = cssText;
10    node.innerHTML = '<ul><li>' + o.timeout + ' ms</li>' +
11      '<li> ' + o.url + '</li></ul>';
12    document.body.appendChild(node);
13  }, {timeout: timeout, url: url});
14 });
15 }
```

- 页面首屏（当前视口）渲染时间的获得

3356 ms ! 请登录 免费注册 新会员专享 我要买 | 我的淘宝 | 卖家中心 | 服务中心 | 购物
http://www.taobao.com/ 电器城 狂暑季 震撼上线 | 满5000返300 / 满8000返600 / 满2万返2千

宝贝 天猫(淘宝商城) 店铺

淘宝网

男衬衫 连体裤 百搭T恤 夏季长裙 新款凉鞋 4S手机壳 男POLO衫 雪纺衫 更多

首页 天猫(淘宝商城) 聚划算 电器城 一淘网 Hitao妆扮 本地生活 淘宝旅行

淘宝服务 更多 >

购物 拍卖会 跳蚤街 淘金币 试用中心 电子书 全球购

生活 彩票 电影票 保险 视频 水电煤

吉列 哈密 3合1 革命性新品 ￥99包邮 润肤 润滑 剃须 1 2 3 4 5 免费注册 公告 规则 高考“神器”淘宝 手机淘宝增长率 便民服务

This screenshot shows the Taobao homepage. A yellow performance bar in the top-left corner displays the rendering time as 3356 ms. The page features a prominent red banner for the 'Summer Shock Launch' with offers like '满5000返300' (Refund 300 for every 5000 spent). Below the banner, there's a search bar and navigation links for categories like Home, Taobao Mall, and Local Life. On the left, a sidebar provides links for various services and生活 (Life) categories like 彩票 (Lottery) and 电影票 (Movie Tickets). A large central advertisement for Gillette razors highlights its '3-in-1' feature and includes a price of 99 yuan with free shipping.

- 页面首屏（当前视口）渲染时间获得的两种计算方法

1. 默认检测方法：

1. 从 urlChanged 事件触发开始计时；
2. 按照当前视口区域平均分布 14400 个像素监控点；
3. 每 250 ms 检测一次所有监控点 RGB 值变化；
4. 如果连续 12 次大于 12000 个像素点无变化，则结束计时，减去检测耗时。

2. 自定义监控点检测方法（App.webview.setDetectionRects）：

1. 从 urlChanged 事件触发开始计时；
2. 按照 setDetectionRects 方法设置的重点检测区块内分布像素级检测点；
3. 每 250 ms 检测一次所有监控点 RGB 值变化；
4. 如果连续 12 次检测区像素阈值无变化，则结束计时，减去检测耗时。

- 获取当前 CPU 占用率与 瞬时内存占用
 - App.cpu() 和 App.memory()
_pageExtension.cpu() 和 _pageExtension.memory() 方法

```
1 var msg = [];
2 var oldMem = App.memory();
3 // 页面脚本刚刚被创建时候检测
4 App.webview.addEventListener('pageScriptCreated', function() {
5   msg[0] = App.cpu() + '%';
6   msg[1] = App.memory() - oldMem + 'KB';
7 });
8 // 页面初始布局完成后检测
9 App.webview.addEventListener("initLayoutCompleted", function() {
10   msg[2] = App.cpu() + '%';
11   msg[3] = App.memory() - oldMem + 'KB';
12 });
13 // 页面加载完成后检测
14 App.webview.addEventListener("load", function() {
15   msg[4] = App.cpu() + '%';
16   msg[5] = App.memory() - oldMem + 'KB';
17   alert(['CPU占用分别是:', msg[0], msg[2], msg[4]].join() + '\n' +
18         ['内存占用分别是:', msg[1], msg[3], msg[5]].join());
19 });
20 App.webview.open('http://www.taobao.com');
```

- 获取当前 CPU 占用率 与 瞬时内存占用

可作为诸多页面性能参照指标之一

命令行 : berserkJS --script=demo8.js

The screenshot shows the Taobao homepage with a red banner at the top for '电器城暑季大促' (Appliance City Summer Promotion) featuring brands like Canon, Apple, lenovo 联想, and SHARP. Below the banner, there's a search bar with tabs for '宝贝' (Item), '天猫(淘宝商城)' (Taobao Mall), and '店铺' (Shop). A navigation bar includes links for '男T恤' (Men's T-shirts), '凉鞋' (Sandals), '连衣裙' (Dresses), '新款T恤' (New T-shirts), '蚊帐' (Mosquito nets), '凉席' (Cooling mats), '父亲节' (Father's Day), '男士短裤' (Men's shorts), and a '更多' (More) link. On the left, a sidebar lists '首页' (Home), '天猫(淘宝商城)' (Taobao Mall), '聚' (Juhuasuan), '淘宝服务' (Taobao Services) with links to '拍卖会' (Auction), '跳蚤街' (Flea Market), '淘金币' (Taobao Gold Coins), '试用中心' (Trial Center), '电子书' (E-books), and '全球购' (Global Purchase); it also includes sections for '购物' (Shopping) with '拍卖会' (Auction), '跳蚤街' (Flea Market), '淘金币' (Taobao Gold Coins), '试用中心' (Trial Center), '电子书' (E-books), and '全球购' (Global Purchase); and '生活' (Life) with '彩票' (Lottery), '电影票' (Movie Tickets), '保险' (Insurance), '视频' (Videos), and '水电煤' (Water, Electricity, Gas). A central promotional banner for '夏日潮流' (Summer Trend) features a woman wearing large blue sunglasses and text for '白款物销' (White Goods Sales), '夏日潮品' (Summer Trend Products), '2012SUMMER HOT ITEMS 100', and '推翻潮流无限可能' (Pushing the boundaries of fashion无限可能). Overlaid on the page is a semi-transparent 'App Message' dialog box with an information icon. The message content is: 'CPU 占用分别是: 1.2%, 18%, 9.5% 内存占用分别是: 3712KB, 8164KB, 39800KB'. There is a 'Yes' button at the bottom right of the dialog.



相比 PhantomJS 加入了文件变更嗅探功能.....

- App.watchFile(file , callback)
- App.unWatcher(file | callbackHandle)
- App.watchedFiles()
- App.watcherClose()
-

这货可以干点蛋疼的事
情..... = = |||

BerserkJS! => <=

说明: 当前工具使用的 User Agent 字符串。

你好~~

App.webview.defaultUserAgent()

参数: 无

返回值: <string> 工具原始的 User Agent 字符串。

说明: 用来获取工具原始的不可被变更 User Agent 字符串, 它不是

App.webview.getUrl()

参数: 无

返回值: <string> 当前浏览页面的 URL。

说明: 当前浏览页面的 URL。

```
App.watchFile('C:\\\\Users\\\\Administrator\\\\Desktop\\\\aaa.html', function(){
  App.webview.reload();
})
```

文件修改同步刷新页面神马的.....
PhantomJS 没 GUI 也就没法干这菊紧的勾当.....

他娘的, 还是小看菊
花残的威力了.....

aa.html -

文件(F) 编辑(E) 搜索(S) 视图(V) 格式(M)

aaa.html

pan id="App.webview.u

1067 **参数:** 无

1068 **返回值:** <s

1069 说明: 当前工

1070

1071 你好~~

1072 <span id="App.webview.d

*实际上它是用来，实现监控文件变化以便发送新的检测报告、执行指定外部程序或运行指定测试模块等需求。



相比 PhantomJS
该有的东西咱还是有的.....

- 模拟鼠标点击事件
- 启动外部进程 获取标准输出流
- 读写文本文件
- 发送 HTTP 请求
- 设置代理
-

- 页面交互，模拟用户登录操作：

```
1 // 访问页面
2 App.webview.open("http://www.weibo.com");
3 // 监听文档加载完成
4 App.webview.addEventListener("load", function() {
5     // 登录表单是惰加载的，等1秒再操作
6     setTimeout(function() {
7         // 进入页面沙箱执行页面脚本
8         App.webview.execScript(function() {
9             // 输入测试账号密码
10            document.querySelectorAll('input[node-type=loginname]')[0]
11            .value = "用户名";
12            document.querySelectorAll('input[node-type=password]')[0]
13            .value = "密码";
14        });
15        // 点下登录按键
16        App.webview.sendMouseEvent(
17            App.webview.elementRects("a[node-type=submit]")[0]
18        );
19    }, 1000);
20});
21
```

* 这个 Demo 就不能给出了，否则偶的用户名密码.....

- 页面交互，完成登录操作：



* 这只是个演示，如果微博登录策略变化（强制登录输入验证码之类的），就不能这么做了。
还是别期望用它干坏事儿为好.....

- 启动外部进程，获取标准输出流，写文件：

命令行：berserkJS --script=demo10.js --command=true

```
1 // 输出当前目录
2 var msg = App.process('cmd.exe', [
3     '/c',
4     'dir',
5     App.path.replace(/\//g, '\\')
6 ]);
7 // 输出文件位置
8 var log = App.path + 'demo9.txt';
9 // 写入文件 UTF-8 编码
10 App.writeFile(log, msg, 'utf-8');
11 // 输出 c 盘目录
12 msg = App.process('cmd.exe', ['/c', 'dir', "c:\\"]);
13 // 以追加方式写入文件
14 App.writeFile(log, '\n\n' + msg, 'utf-8', true);
15 // 关闭
16 App.close();
```

* App.readFile(path [, charset])

- 启动外部进程，获取标准输出流：

```
19 2012/03/22 15:21 ..... 11,362 mingwm10.dll
20 2012/03/27 19:08 <DIR> ..... module
21 2012/03/22 15:22 ..... 2,843,136 QtCore4.dll
22 2012/03/22 15:22 ..... 10,135,040 QtGui4.dll
23 2012/03/22 15:22 ..... 1,289,728 QtNetwork4.dll
24 2012/03/22 15:22 ..... 2,179,584 QtScript4.dll
25 2012/03/22 15:22 ..... 20,170,240 QtWebKit4.dll
26 2012/03/27 19:10 <DIR> ..... test
27 ..... 15 个文件 ..... 37,029,648 字节
28 ..... 6 个目录 15,376,007,168 可用字节
29
30
31 驱动器 C 中的卷是 win
32 卷的序列号是 E0D9-FE0A
33
34 c:\ 的目录
35
36 2011/11/02 18:27 ..... 1,024 .rnd
37 2012/03/15 15:23 ..... 6,989 1.txt
38 2009/06/11 05:42 ..... 24 autoexec.bat
39 2009/06/11 05:42 ..... 10 config.sys
```

- 发送 HTTP 请求：

命令行：berserkJS --script=demo11.js

```
1 // 打开目标页
2 App.webview.open('http://www.baidu.com')
3 // 监听页面加载完成
4 App.webview.addEventListener('load', function() {
5     // 跨域同步取得请求结果
6     var html = App.httpRequest('get', 'http://www.taobao.com');
7     // 将结果交给浏览器 JS 处理
8     this.execScript(function(html) {
9         document.documentElement.innerHTML = html;
10    }, html);
11    // 看下当前 URL
12    alert(App.webview.getUrl());
13 });
14 }
```

- 发送 HTTP 请求：



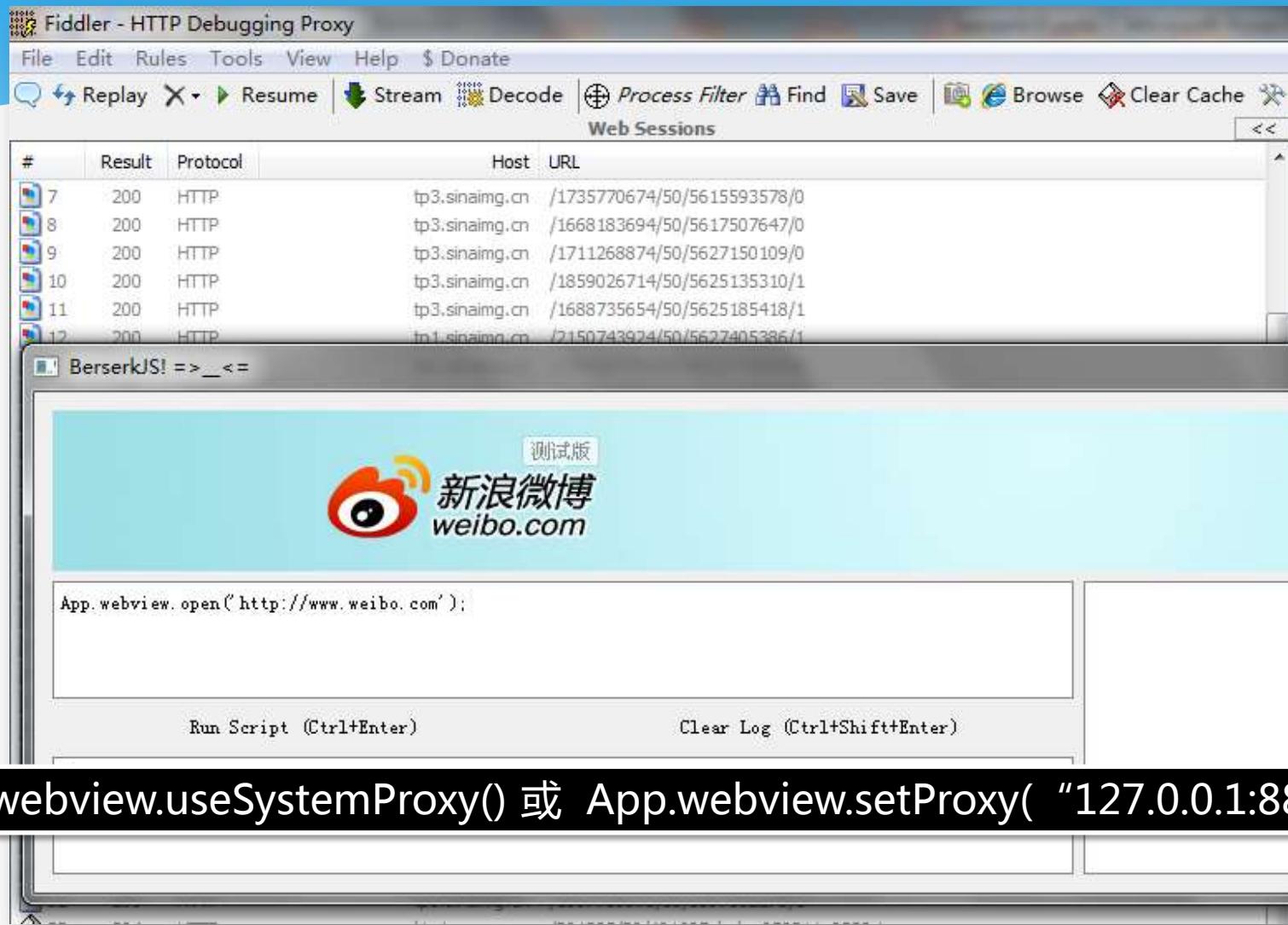
* 它不是浏览器内JS发起的XHR请求，可以完全无视跨域问题。

- 使用代理与自定UA：
 - `useSystemProxy([index])`
 - `App.webview.setProxy(host[, type, userName, password])`
 - `App.webview.clearProxy()`
 - `App.webview.setUserAgent(userAgent <string>)`

*berserkJS 默认使用系统代理。

- 由于 berserkJS 可使用脚本操作浏览器。使用代理 API 将可以在自动化操作浏览器基础上，在代理服务器端统计各项数据。
- 这些数据可以作为 `networkData` 方法提供的数据内容补充，或者完全代替它。
- 当然你也可以用收集墙外页面性能数据（貌似没必要.....）
- 可自定UA，方便服务端过滤与统计。

- 使用代理：



App.webview.useSystemProxy() 或 App.webview.setProxy("127.0.0.1:8888")

berserkJS

哪里有卖的？

哪里有卖的？

D2

Code Network Pull Requests 0 Issues 0 Wiki Graphs

基于 qt webkit，使用 JS 语法开发检测逻辑的 webkit netwrok request 检测工具。 — [Read more](#)

[Clone in Windows](#) [ZIP](#) [SSH](#) [HTTP](#) [Git Read-Only](#) git@github.com:tapir-dream/berserkJS.git [Read+Write access](#)

branch: [master](#) [Files](#) Commits Branches 1 Tags Downloads

Latest commit to the **master** branch

add plugin dll for window system.
unknown authored 2 days ago [commit 49039003af](#)

berserkJS /

name	age	message	history
api	2 days ago	add plugin dll for window system. [unknown]	
build	2 days ago	add plugin dll for window system. [unknown]	
demo	6 days ago	update [unknown]	
src	2 days ago	add plugin dll for window system. [unknown]	

— → Window 系统可以直接用 build 目录下有编译好的

获取源码：

<https://github.com/tapir-dream/berserkJS>

阅读 API 文档

<http://tapir-dream.github.com/berserkJS/> 或工程目录/api/index.html

berserkJS

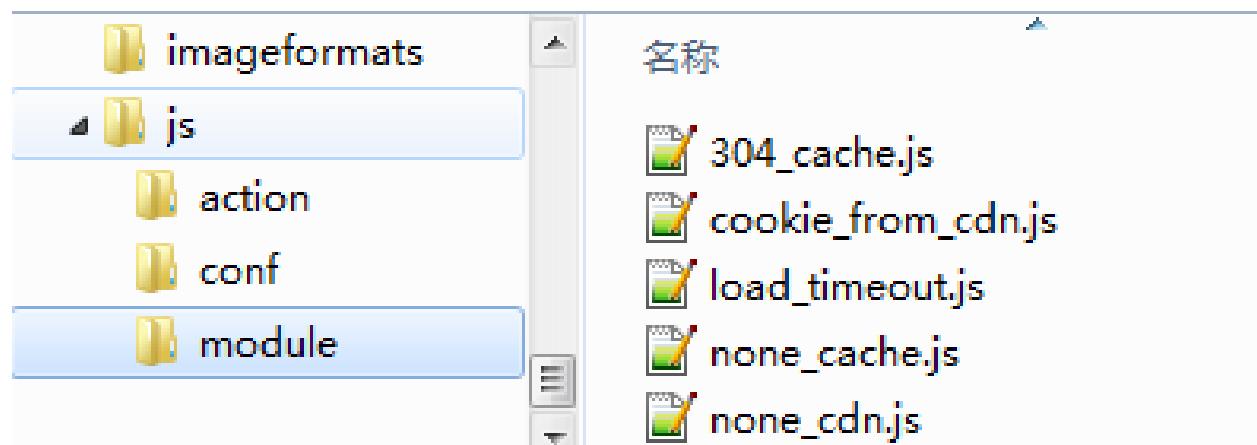
这货的两种使用方法

D2

使用预制的模块配置功能：

使用命令行参数： --start=true

- 它自动运行berserkJS 所在目录中的 js/conf/init.js 文件
- init.js 文件会根据 conf.js 文件内容执行相应模块中代码。



- 配置文件 config.js :

```
(function() {
    .....
    return {
        global: [
            namespace( 'action.helper' ) // 最初就需要执行的内容
        ],
        // 自动化交互脚本位置
        automation: namespace('action.autoscript'),
        // 交互完成后要执行的模块列表
        module: [
            {
                path: namespace('module.none_gzip_doc'),
                args: []
            },
            ...
        ],
        completed: [
            namespace( 'action.report' ) // 所有模块执行完后动作
        ]
    };
});
```

- 模块文件内容：

```
(function (data, max) {  
    var supplant = App.helper.supplant;  
  
    var duration = {};  
    for (var i = 0, c = data.length; i < c; ++i) {  
        if (data[i].ResponseDuration > max) {  
            duration[data[i].url] = data[i].ResponseDuration;  
        }  
    }  
  
    var urls = Object.keys(duration);  
    var count = urls.length;  
    var message = "如下 URL 加载时间大于 ${max} ms: \n";  
    for (var i = 0; i < count; ++i) {  
        message += supplant("URL: ${url}, Duration: ${time} ms \n", {  
            url: urls[i],  
            time: duration[urls[i]]  
        });  
    }  
    message = supplant(message, {max: max});  
    return message;  
});
```

- 所有模块执行后的输出内容：

```
1 如下 URL 没有设置 Expires 头, 共 14 个,建议设置:  
2 url: http://3.180.web0.im.weibo.com/im?jsonp=parent.org.cometd.script._callback2&message=[{  
3 url: http://3.180.web0.im.weibo.com/im?jsonp=parent.org.cometd.script._callback3&message=[{  
4 url: http://3.180.web0.im.weibo.com/im?jsonp=parent.org.cometd.script._callback4&message=[{  
5 url: http://3.180.web0.im.weibo.com/im?jsonp=parent.org.cometd.script._callback5&message=[{  
6 url: http://3.180.web0.im.weibo.com/im?jsonp=parent.org.cometd.script._callback6&message=[{  
7 url: http://beacon.sina.com.cn/a.gif?V=2&CI=sz:1366x768|dp:32|ac:Mozilla|an:Netscape|cpu:un  
8 url: http://beacon.sina.com.cn/a.gif?V=2&CI=sz:1366x768|dp:32|ac:Mozilla|an:Netscape|cpu:un  
9 url: http://beacon.sina.com.cn/d.gif?&gUid_1332846542390  
10 url: http://kandian.com/logon/do_crossdomain.php?action=login&savestate=1333451344&callback=  
11 url: http://login.sina.com.cn/sso/login.php?client=ssologin.js(v1.3.19)  
12 url: http://login.sina.com.cn/sso/prelogin.php?entry=weibo&callback=sinaSSOController.prelogi  
13 url: http://login.t.cn/sinaurl/sso.json?action=login&uid=1816235717&callback=sinaSSOController.  
14 url: http://nas.im.api.weibo.com/im/emotions/emotions.jsonp?v=v1.0&callback=STK_13328465431  
15 url: http://nas.im.api.weibo.com/im/webim.jsp?v=1.1&returntype=json&uid=1816235717&callback=  
16  
17 如下 JS/CSS/HTML URL 没有被GZip, 共 17 个,建议启用:  
18 url: http://3.180.web0.im.weibo.com/im?jsonp=parent.org.cometd.script._callback1&message=[{  
19 url: http://3.180.web0.im.weibo.com/im?jsonp=parent.org.cometd.script._callback2&message=[{  
20 url: http://3.180.web0.im.weibo.com/im?jsonp=parent.org.cometd.script._callback3&message=[{  
21 url: http://3.180.web0.im.weibo.com/im?jsonp=parent.org.cometd.script._callback4&message=[{  
22 url: http://3.180.web0.im.weibo.com/im?jsonp=parent.org.cometd.script._callback5&message=[{  
23 url: http://3.180.web0.im.weibo.com/im?jsonp=parent.org.cometd.script._callback6&message=[{  
24 url: http://biz.weibo.com/adfront/deliver?psid=PDPS000000037693&wbVersion=v4&uid=1816235717  
25 url: http://biz.weibo.com/adfront/deliver?psid=PDPS000000037694&wbVersion=v4&uid=1816235717  
26 url: http://biz.weibo.com/adfront/deliver?psid=PDPS000000037695&wbVersion=v4&uid=1816235717
```

使用自定义的实现方式：

在之前的 Demo 中已经演示过一部分了。

使用命令行参数：--script 指定启动时执行的脚本



使用 App.loadScript 方法在运行期载入脚本



使用 App.args 方法，在运行期获得命令行所有参数

- loadScript 方法与 App.args :

命令行 : berserkJS --script=demo12.js weibo

```
1 // 输出命令行参数
2 console.log('所有命令行参数是: ' + App.args)
3
4 // 检测指定参数内容
5 if (App.args[1] != 'weibo') {
6     App.close();
7 }
8
9 // 加载脚本文件
10 App.loadScript(App.path + 'js/conf/init.js', function(err, func) {
11     // 检查脚本是否可用
12     if (err) {
13         alert('Hi~ Load Script Error!')
14     }
15     // 执行脚本
16     func(App, App.webview);
17 });
18
```

*与前例执行结果一致

berserkJS

对比•总结

D2

普通的收集数据步骤：

1. 手动开启浏览器
2. 打开开发者工具或其他辅助软件
3. 输入网址或刷新
4. 等待数据收集完毕
5. 导出数据
6. 关闭浏览器
7. 编写（或使用开源的）数据处理程序
8. 分析出所需数据
9. 执行以上步骤若干次
10. 汇总制表或提交数据

缺陷：

- 无法自动化
- 无法远程请求数据收集

其它
自动
化替
代工
具

berserkJS 收集数据步骤：

1. 编写数据处理程序
(含汇总与提交数据处理)
2. 执行此程序
3. 汇总制表

对比

优势：

- 自动化
- 可命令行调用
- 被Web服务调用
-



横向比较缺陷：

- 非用户数据来路，数据来路单一
- 非多UA数据，数据丰富不够
- 欠缺详细的页面 JS 运行性能监控
- 现阶段调试起来还不是很方便

横向比较特性：

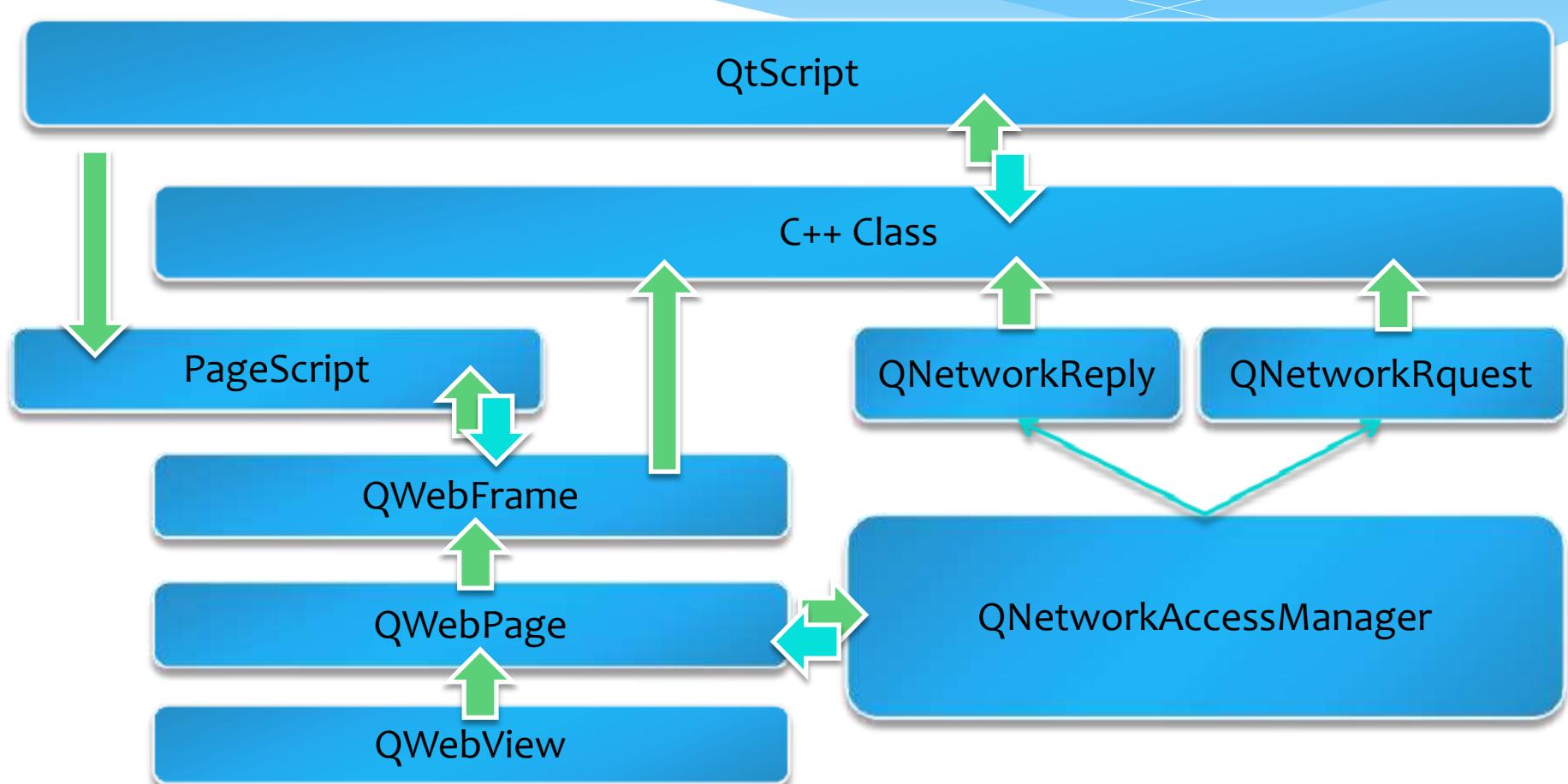
- 侧重代码上线前评估
- JS脚本化
- 相对功能较多
- 跨平台性
- 为了简化工作而定制的工具

berserkJS

实现原理是什么？

D2

- Qt 内的 QWebView、QWebPage、QWebFrame 等类就是 Webkit 内核的关键类。
- QtScript 是 Webkit 项目内 JS 引擎 JavaScriptCore 的实现。
- 继承他们，从它们提供的数据中抽取需要的内容。
- 包装给 QtScript 调用。





谨以此页献给偶心中伟大的
计算机之子 @寒冬winter 老师

Q & A

session

拼人品的时刻到了>_<

D₂

求 winter hax
别拍偶砖





完事儿 谢谢

D₂

