**PhantomJS**

*参考：*[*http://javascript.ruanyifeng.com/tool/phantomjs.html*](http://javascript.ruanyifeng.com/tool/phantomjs.html)

*来自[《JavaScript 标准参考教程（alpha）》](http://javascript.ruanyifeng.com/)，by 阮一峰*

**目录**

* [**概述**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc0)
* [**REPL环境**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc1)
* [**webpage模块**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc2)
* [**open()**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc3)
* [**evaluate()**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc4)
* [**includeJs()**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc5)
* [**render()**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc6)
* [**viewportSize，zoomFactor**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc7)
* [**onResourceRequested**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc8)
* [**onResourceReceived**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc9)
* [**system模块**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc10)
* [**应用**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc11)
* [**过滤资源**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc12)
* [**截图**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc13)
* [**抓取图片**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc14)
* [**生成网页**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc15)
* [**参考链接**](http://javascript.ruanyifeng.com/tool/phantomjs.html#toc16)

**概述**

有时，我们需要浏览器处理网页，但并不需要浏览，比如生成网页的截图、抓取网页数据等操作。[PhantomJS](http://phantomjs.org/)的功能，就是提供一个浏览器环境的命令行接口，你可以把它看作一个“虚拟浏览器”，除了不能浏览，其他与正常浏览器一样。它的内核是WebKit引擎，不提供图形界面，只能在命令行下使用，我们可以用它完成一些特殊的用途。

PhantomJS是二进制程序，需要[安装](http://phantomjs.org/download.html)后使用。

$ npm install phantomjs -g

使用下面的命令，查看是否安装成功。

$ phantomjs --version

**REPL环境**

phantomjs提供了一个完整的REPL（交互式解释器环境）环境，允许用户通过命令行与PhantomJS互动。键入phantomjs，就进入了该环境。

$ phantomjs

这时会跳出一个phantom提示符，就可以输入Javascript命令了。

phantomjs> 1+2

3

phantomjs> function add(a,b) { return a+b; }

undefined

phantomjs> add(1,2)

3

按ctrl+c可以退出该环境。

下面，我们把上面的add()函数写成一个文件add.js文件。

// add.js

function add(a,b){ return a+b; }

console.log(add(1,2));

phantom.exit();

上面的代码中，console.log()的作用是在终端窗口显示，phantom.exit()则表示退出phantomjs环境。一般来说，不管什么样的程序，exit这一行都不能少。

现在，运行该程序。

$ phantomjs add.js

终端窗口就会显示结果为3。

下面是更多的例子。

phantomjs> phantom.version

{

"major": 1,

"minor": 5,

"patch": 0

}

phantomjs> console.log("phantom is awesome")

phantom is awesome

phantomjs> window.navigator

{

"cookieEnabled": true,

"language": "en-GB",

"productSub": "20030107",

"product": "Gecko",

// ...

}

**webpage模块**

webpage模块是PhantomJS的核心模块，用于网页操作。

var webPage = require('webpage');

var page = webPage.create();

上面代码表示加载PhantomJS的webpage模块，并创建一个实例。

下面是webpage实例的属性和方法介绍。

**open()**

open方法用于打开具体的网页。

var page = require('webpage').create();

page.open('http://slashdot.org', function (s) {

console.log(s);

phantom.exit();

});

上面代码中，open()方法，用于打开具体的网页。它接受两个参数。第一个参数是网页的网址，这里打开的是著名新闻网站[Slashdot](http://slashdot.org/)，第二个参数是回调函数，网页打开后该函数将会运行，它的参数是一个表示状态的字符串，如果打开成功就是success，否则就是fail。

注意，只要接收到服务器返回的结果，PhantomJS就会报告网页打开成功，而不管服务器是否返回404或500错误。

open方法默认使用GET方法，与服务器通信，但是也可以使用其他方法。

var webPage = require('webpage');

var page = webPage.create();

var postBody = 'user=username&password=password';

page.open('http://www.google.com/', 'POST', postBody, function(status) {

console.log('Status: ' + status);

// Do other things here...

});

上面代码中，使用POST方法向服务器发送数据。open方法的第二个参数用来指定HTTP方法，第三个参数用来指定该方法所要使用的数据。

open方法还允许提供配置对象，对HTTP请求进行更详细的配置。

var webPage = require('webpage');

var page = webPage.create();

var settings = {

operation: "POST",

encoding: "utf8",

headers: {

"Content-Type": "application/json"

},

data: JSON.stringify({

some: "data",

another: ["custom", "data"]

})

};

page.open('http://your.custom.api', settings, function(status) {

console.log('Status: ' + status);

// Do other things here...

});

**evaluate()**

evaluate方法用于打开网页以后，在页面中执行JavaScript代码。

var page = require('webpage').create();

page.open(url, function(status) {

var title = page.evaluate(function() {

return document.title;

});

console.log('Page title is ' + title);

phantom.exit();

});

网页内部的console语句，以及evaluate方法内部的console语句，默认不会显示在命令行。这时可以采用onConsoleMessage回调函数，上面的例子可以改写如下。

var page = require('webpage').create();

page.onConsoleMessage = function(msg) {

console.log('Page title is ' + msg);

};

page.open(url, function(status) {

page.evaluate(function() {

console.log(document.title);

});

phantom.exit();

});

上面代码中，evaluate方法内部有console语句，默认不会输出在命令行。这时，可以用onConsoleMessage方法监听这个事件，进行处理。

**includeJs()**

includeJs方法用于页面加载外部脚本，加载结束后就调用指定的回调函数。

var page = require('webpage').create();

page.open('http://www.sample.com', function() {

page.includeJs("http://path/to/jquery.min.js", function() {

page.evaluate(function() {

$("button").click();

});

phantom.exit()

});

});

上面的例子在页面中注入jQuery脚本，然后点击所有的按钮。需要注意的是，由于是异步加载，所以phantom.exit()语句要放在page.includeJs()方法的回调函数之中，否则页面会过早退出。

**render()**

render方法用于将网页保存成图片，参数就是指定的文件名。该方法根据后缀名，将网页保存成不同的格式，目前支持PNG、GIF、JPEG和PDF。

var webPage = require('webpage');

var page = webPage.create();

page.viewportSize = { width: 1920, height: 1080 };

page.open("http://www.google.com", function start(status) {

page.render('google\_home.jpeg', {format: 'jpeg', quality: '100'});

phantom.exit();

});

该方法还可以接受一个配置对象，format字段用于指定图片格式，quality字段用于指定图片质量，最小为0，最大为100。

**viewportSize，zoomFactor**

viewportSize属性指定浏览器视口的大小，即网页加载的初始浏览器窗口大小。

var webPage = require('webpage');

var page = webPage.create();

page.viewportSize = {

width: 480,

height: 800

};

viewportSize的Height字段必须指定，不可省略。

zoomFactor属性用来指定渲染时（render方法和renderBase64方法）页面的放大系数，默认是1（即100%）。

var webPage = require('webpage');

var page = webPage.create();

page.zoomFactor = 0.25;

page.render('capture.png');

**onResourceRequested**

onResourceRequested属性用来指定一个回调函数，当页面请求一个资源时，会触发这个回调函数。它的第一个参数是HTTP请求的元数据对象，第二个参数是发出的网络请求对象。

HTTP请求包括以下字段。

* id：所请求资源的编号
* method：使用的HTTP方法
* url：所请求的资源 URL
* time：一个包含请求时间的Date对象
* headers：HTTP头信息数组

网络请求对象包含以下方法。

* abort()：终止当前的网络请求，这会导致调用onResourceError回调函数。
* changeUrl(newUrl)：改变当前网络请求的URL。
* setHeader(key, value)：设置HTTP头信息。

var webPage = require('webpage');

var page = webPage.create();

page.onResourceRequested = function(requestData, networkRequest) {

console.log('Request (#' + requestData.id + '): ' + JSON.stringify(requestData));

};

**onResourceReceived**

onResourceReceived属性用于指定一个回调函数，当网页收到所请求的资源时，就会执行该回调函数。它的参数就是服务器发来的HTTP回应的元数据对象，包括以下字段。

* id：所请求的资源编号
* url：所请求的资源的URL r- time：包含HTTP回应时间的Date对象
* headers：HTTP头信息数组
* bodySize：解压缩后的收到的内容大小
* contentType：接到的内容种类
* redirectURL：重定向URL（如果有的话）
* stage：对于多数据块的HTTP回应，头一个数据块为start，最后一个数据块为end。
* status：HTTP状态码，成功时为200。
* statusText：HTTP状态信息，比如OK。

如果HTTP回应非常大，分成多个数据块发送，onResourceReceived会在收到每个数据块时触发回调函数。

var webPage = require('webpage');

var page = webPage.create();

page.onResourceReceived = function(response) {

console.log('Response (#' + response.id + ', stage "' + response.stage + '"): ' + JSON.stringify(response));

};

**system模块**

system模块可以加载操作系统变量，system.args就是参数数组。

var page = require('webpage').create(),

system = require('system'),

t, address;

// 如果命令行没有给出网址

if (system.args.length === 1) {

console.log('Usage: page.js <some URL>');

phantom.exit();

}

t = Date.now();

address = system.args[1];

page.open(address, function (status) {

if (status !== 'success') {

console.log('FAIL to load the address');

} else {

t = Date.now() - t;

console.log('Loading time ' + t + ' ms');

}

phantom.exit();

});

使用方法如下：

$ phantomjs page.js http://www.google.com

**应用**

Phantomjs可以实现多种应用。

**过滤资源**

处理页面的时候，有时不希望加载某些特定资源。这时，可以对URL进行匹配，一旦符合规则，就中断对资源的连接。

page.onResourceRequested = function(requestData, request) {

if ((/http:\/\/.+?\.css$/gi).test(requestData['url'])) {

console.log('Skipping', requestData['url']);

request.abort();

}

};

上面代码一旦发现加载的资源是CSS文件，就会使用request.abort方法中断连接。

**截图**

最简单的生成网页截图的方法如下。

var page = require('webpage').create();

page.open('http://google.com', function () {

page.render('google.png');

phantom.exit();

});

page对象代表一个网页实例；open方法表示打开某个网址，它的第一个参数是目标网址，第二个参数是网页载入成功后，运行的回调函数;render方法则是渲染页面，然后以图片格式输出，该方法的参数就是输出的图片文件名。

除了简单截图以外，还可以设置各种截图参数。

var page = require('webpage').create();

page.open('http://google.com', function () {

page.zoomFactor = 0.25;

console.log(page.renderBase64());

phantom.exit();

});

zoomFactor表示将截图缩小至原图的25%大小；renderBase64方法则是表示将截图（PNG格式）编码成Base64格式的字符串输出。

下面的例子则是使用了更多参数。

// page.js

var page = require('webpage').create();

page.settings.userAgent = 'WebKit/534.46 Mobile/9A405 Safari/7534.48.3';

page.settings.viewportSize = { width: 400, height: 600 };

page.open('http://slashdot.org', function (status) {

if (status !== 'success') {

console.log('Unable to load!');

phantom.exit();

} else {

var title = page.evaluate(function () {

var posts = document.getElementsByClassName("article");

posts[0].style.backgroundColor = "#FFF";

return document.title;

});

window.setTimeout(function () {

page.clipRect = { top: 0, left: 0, width: 600, height: 700 };

page.render(title + "1.png");

page.clipRect = { left: 0, top: 600, width: 400, height: 600 };

page.render(title + '2.png');

phantom.exit();

}, 1000);

}

});

上面代码中的几个属性和方法解释如下：

* settings.userAgent：指定HTTP请求的userAgent头信息，上面例子是手机浏览器的userAgent。
* settings.viewportSize：指定浏览器窗口的大小，这里是400x600。
* evaluate()：用来在网页上运行Javascript代码。在这里，我们抓取第一条新闻，然后修改背景颜色，并返回该条新闻的标题。
* clipRect：用来指定网页截图的大小，这里的截图左上角从网页的(0. 0)坐标开始，宽600像素，高700像素。如果不指定这个值，就表示对整张网页截图。
* render()：根据clipRect的范围，在当前目录下生成以第一条新闻的名字命名的截图。

**抓取图片**

使用官方网站提供的[rasterize.js](https://github.com/ariya/phantomjs/blob/master/examples/rasterize.js)，可以抓取网络上的图片，将起保存在本地。

phantomjs rasterize.js http://ariya.github.com/svg/tiger.svg tiger.png

使用[rasterize.js](https://github.com/ariya/phantomjs/blob/master/examples/rasterize.js)，还可以将网页保存为pdf文件。

phantomjs rasterize.js 'http://en.wikipedia.org/w/index.php?title=Jakarta&printable=yes' jakarta.pdf

**生成网页**

phantomjs可以生成网页，使用content方法指定网页的HTML代码。

var page = require('webpage').create();

page.viewportSize = { width: 400, height : 400 };

page.content = '<html><body><canvas id="surface"></canvas></body></html>';

phantom.exit();

官方网站有一个[例子](https://github.com/ariya/phantomjs/blob/master/examples/colorwheel.js)，通过创造svg图片，然后截图保存成png文件。

**参考链接**

* [Testing JavaScript with PhantomJS](http://net.tutsplus.com/tutorials/javascript-ajax/testing-javascript-with-phantomjs/)
* [Phantom Quick Start](https://github.com/ariya/phantomjs/wiki/Quick-Start)
* Ariya Hidayat, [Web Page Clipping with PhantomJS](http://ariya.ofilabs.com/2013/04/web-page-clipping-with-phantomjs.html)
* BenjaminBenBen, [Using PhantomJS WebServer](http://benjaminbenben.com/2013/07/28/phantomjs-webserver/)
* Ariya Hidayat, [Capturing Web Page Without Stylesheets](http://ariya.ofilabs.com/2013/06/capturing-web-page-without-stylesheets.html): 过滤CSS文件

**留言**

[comments powered by Disqus](http://disqus.com/)

s