Chapter 9

Using Development Tools to Examine Webpages

Skills you will learn: For this tutorial, we will use the developer tools in Firefox. However, these are quite similar to the developer tools found in just about every modern web browser, including Google's Chrome and Microsoft's Edge.

We aren't covering everything Firefox's developer tools can do, but instead are focusing on aspects that are of value for web scraping and simple development.

Note that the prior version of this tutorial focused on the Firebug plugin, which has since been integrated into Firefox's toolset.

Summary: Hidden behind the public face of modern web browsers is a suite of tools that gives you extraordinary power to inspect the source code of web pages, and observe network requests and responses. These tools are useful both in web scraping, discussed in chapter 9, and web development, discussed in chapter 10.

Introduction:

All web browsers allow the user to look at the source HTML for the page, and this is perfectly useful for looking at the overall HTML structure of a page, for seeing tags in their full context, for searching for elements, and so on. But sometimes you want to drill down and look more closely at specific elements on the page, or you want to be able to see the network traffic being passed to and from the remote web server. For these tasks, development tools can be the perfect solution.

Using the tools

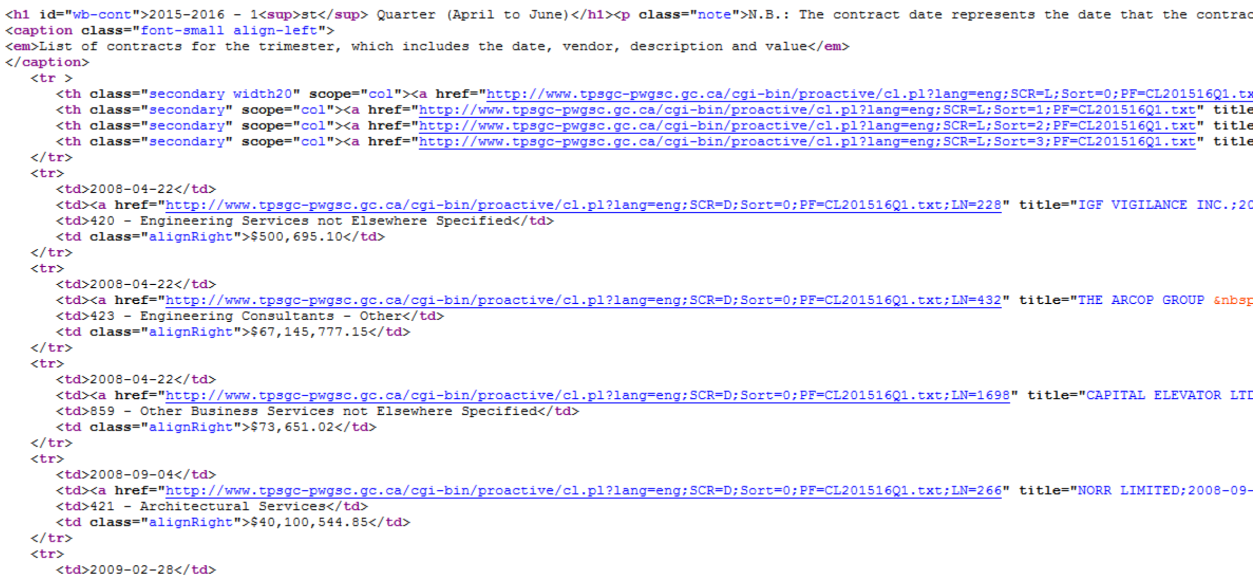
The developer tools are an integrated part of Firefox.

To start using them, got to Tools>Web Developer>Toggle Tools. We'll use the Mac version in this tutorial, but the functionality is essentially identical on a Windows PC.

Examining Elements in a webpage

We're going to have a look at the source code for the Public Works and Government Services Canada (now renamed Public Services and Procurement Canada) proactive disclosure page that we first saw in chapter 9 of *The Data Journalist*. While the government is moving the proactive disclosure information to its open data portal, the page you saw in Chapter 9 is archived at this address: https://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=L;Sort=0;PF=CL201516Q1.txt

If you simply open the page source (Tools>Web Developer>Page Source), you will see all of the HTML for the page.

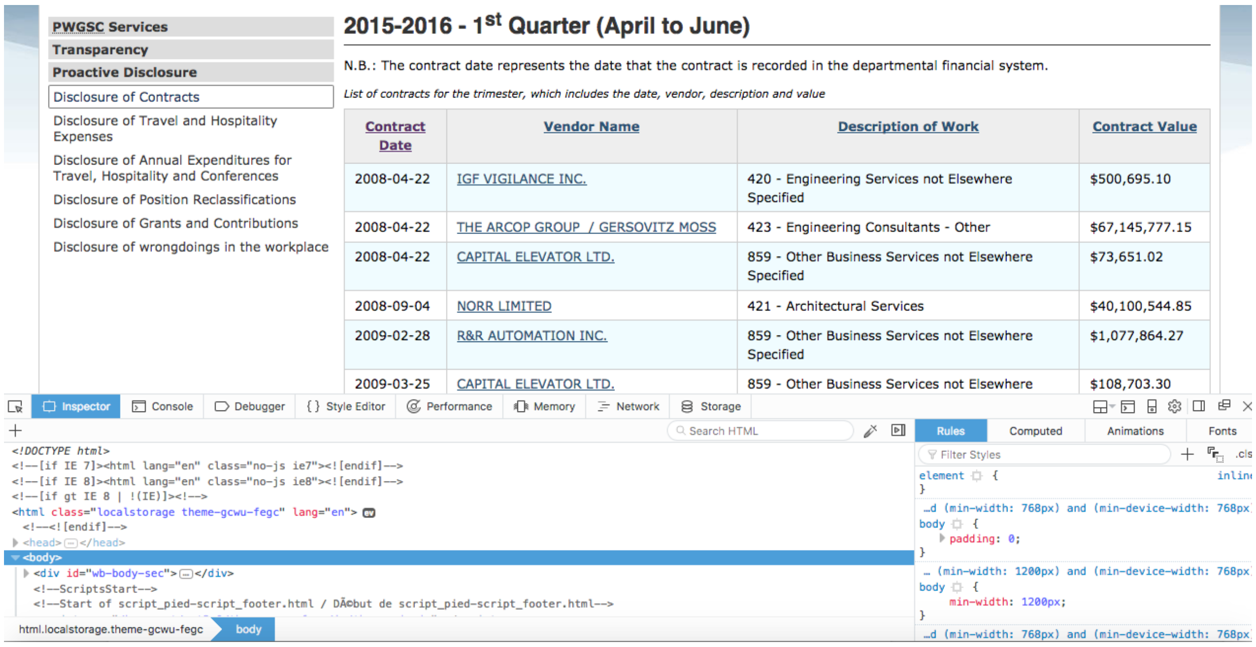


```
<h1 id="wb-cont">2015-2016 - 1<sup>st</sup> Quarter (April to June)</h1><p class="note">N.B.: The contract date represents the date that the contrac
<caption class="font-small align-left">
<em>List of contracts for the trimester, which includes the date, vendor, description and value</em>
</caption>
   <tr >
      <th class="secondary width20" scope="col"><a href="http://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=L;Sort=0;PF=CL201516Q1.tx
      <th class="secondary" scope="col"><a href="http://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=L;Sort=1;PF=CL201516Q1.txt" title
      <th class="secondary" scope="col"><a href="http://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=L;Sort=2;PF=CL201516Q1.txt" title
      <th class="secondary" scope="col"><a href="http://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=L;Sort=3;PF=CL201516Q1.txt" title
   </tr>
   <tr>
      <td>2008-04-22</td>
      <td><a href="http://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=D;Sort=0;PF=CL201516Q1.txt;LN=228" title="IGF VIGILANCE INC.;20
      <td>420 - Engineering Services not Elsewhere Specified</td>
      <td class="alignRight">$500,695.10</td>
   </tr>
   <tr>
      <td>2008-04-22</td>
      <td><a href="http://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=D;Sort=0;PF=CL201516Q1.txt;LN=432" title="THE ARCOP GROUP &nbsp
      <td>423 - Engineering Consultants - Other</td>
      <td class="alignRight">$67,145,777.15</td>
   </tr>
   <tr>
      <td>2008-04-22</td>
      <td><a href="http://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=D;Sort=0;PF=CL201516Q1.txt;LN=1698" title="CAPITAL ELEVATOR LTD
      <td>859 - Other Business Services not Elsewhere Specified</td>
      <td class="alignRight">$73,651.02</td>
   </tr>
   <tr>
      <td>2008-09-04</td>
      <td><a href="http://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=D;Sort=0;PF=CL201516Q1.txt;LN=266" title="NORR LIMITED;2008-09-
      <td>421 - Architectural Services</td>
      <td class="alignRight">$40,100,544.85</td>
   </tr>
   <tr>
      <td>2009-02-28</td>
```

Looking at the HTML code this way is useful way to put it all in context, especially when you already know what elements you want to look at or when you are trying to figure out how you will instruct a Python library such as Beautiful Soup to pinpoint the elements you want (Chapter 9 introduces Beautiful Soup).

But when you are first exploring a page to see how it works, the element inspector can make the job a lot easier.

If you open Developer Tools while the page is open, you see a panel like the one below appears at the bottom of the browser screen.


PWGSC Services
Transparency
Proactive Disclosure
Disclosure of Contracts
Disclosure of Travel and Hospitality Expenses
Disclosure of Annual Expenditures for Travel, Hospitality and Conferences
Disclosure of Position Reclassifications
Disclosure of Grants and Contributions
Disclosure of wrongdoings in the workplace

2015-2016 - 1st Quarter (April to June)

N.B.: The contract date represents the date that the contract is recorded in the departmental financial system.

List of contracts for the trimester, which includes the date, vendor, description and value

Contract Date	Vendor Name	Description of Work	Contract Value
2008-04-22	IGF VIGILANCE INC.	420 - Engineering Services not Elsewhere Specified	$500,695.10
2008-04-22	THE ARCOP GROUP / GERSOVITZ MOSS	423 - Engineering Consultants - Other	$67,145,777.15
2008-04-22	CAPITAL ELEVATOR LTD.	859 - Other Business Services not Elsewhere Specified	$73,651.02
2008-09-04	NORR LIMITED	421 - Architectural Services	$40,100,544.85
2009-02-28	R&R AUTOMATION INC.	859 - Other Business Services not Elsewhere Specified	$1,077,864.27
2009-03-25	CAPITAL ELEVATOR LTD.	859 - Other Business Services not Elsewhere	$108,703.30

```
<!DOCTYPE html>
<!--[if IE 7]><html lang="en" class="no-js ie7"><![endif]-->
<!--[if IE 8]><html lang="en" class="no-js ie8"><![endif]-->
<!--[if gt IE 8 | !(IE)]><!-->
<html class="localstorage theme-gcwu-fegc" lang="en"> ev
  <!--<![endif]-->
▶ <head>…</head>
▼ <body>
  ▶ <div id="wb-body-sec">…</div>
    <!--ScriptsStart-->
    <!--Start of script_pied-script_footer.html / DÃ©but de script_pied-script_footer.html-->
```

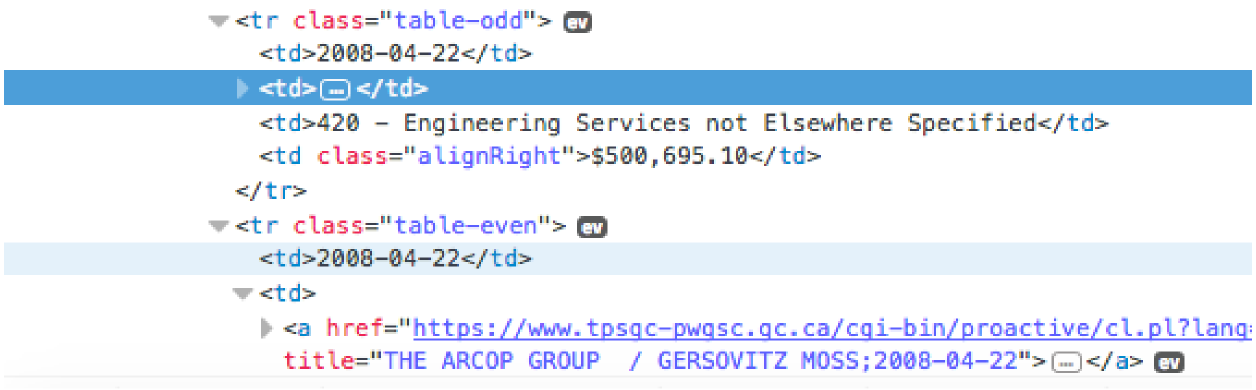
html.localstorage.theme-gcwu-fegc > body

```
element {                                   inline
}
…d (min-width: 768px) and (min-device-width: 768px
body {
  ▶ padding: 0;
}
… (min-width: 1200px) and (min-device-width: 768px
body {
    min-width: 1200px;
}
…d (min-width: 768px) and (min-device-width: 768px
```

If you look at the top left corner of the panel, you can see the element inspector icon.

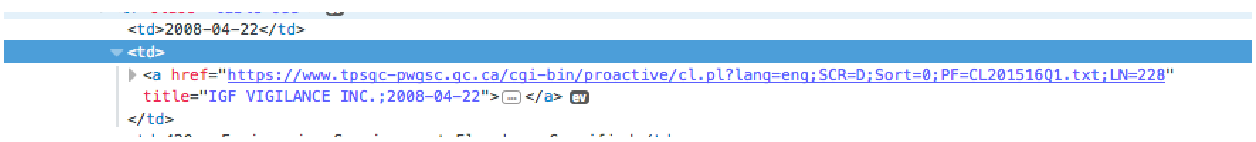
Clicking on the icon makes the inspector active. With it we can drill down into the HTML for any part of the page that we want to examine. Simply click on a part of the webpage in the upper window containing the web page as rendered by Firefox, and the inspector will show you the underlying HTML and CSS code. We'll use the inspector to click on the HTML table value IGF Vigilance Inc.

We then see the HTML for that part of the page, in the lower panel.



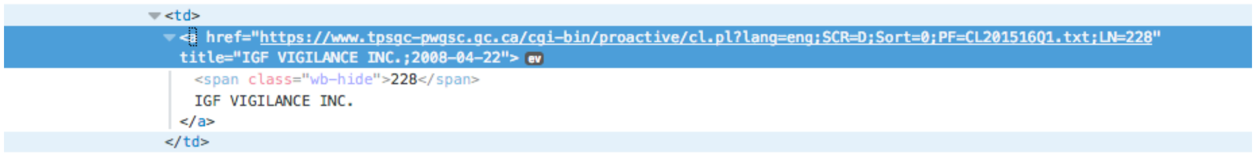
```
▼ <tr class="table-odd"> ev
    <td>2008-04-22</td>
  ▶ <td>…</td>
    <td>420 - Engineering Services not Elsewhere Specified</td>
    <td class="alignRight">$500,695.10</td>
  </tr>
▼ <tr class="table-even"> ev
    <td>2008-04-22</td>
  ▼ <td>
    ▶ <a href="https://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang
      title="THE ARCOP GROUP  / GERSOVITZ MOSS;2008-04-22">…</a> ev
```

The small expansion arrows to the left of individual HTML elements indicate that more detail is available to be viewed. If we click on that, we see more code, and yet another expansion arrow indicating we can drill down even further.



```
<td>2008-04-22</td>
<td>
  <a href="https://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=D;Sort=0;PF=CL20151601.txt;LN=228"
  title="IGF VIGILANCE INC.;2008-04-22">…</a> ev
</td>
```

So we'll click on that to reveal the last layer of detail.



```
<td>
  <a href="https://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=D;Sort=0;PF=CL20151601.txt;LN=228"
  title="IGF VIGILANCE INC.;2008-04-22"> ev
    <span class="wb-hide">228</span>
    IGF VIGILANCE INC.
  </a>
</td>
```

We can now see all of the HTML code associated with the IGF Vigilance entry. First off, it is contained in a <td> or table cell tag, which itself is enclosed in a <tr> or table row tag, which in turn is enclosed in a <tbody> or table body tag, and so in, in the hierarchy of tags that makes up the webpage.

Within the <td> tag is a <a> or hyperlink tag. The title attribute that follows causes its contents to appear as a tooltip when a user hovers the mouse over the link. This is then followed by the HREF attribute, which indicates the destination of the link, in this case another Public Words web page that provides detail on this particular contract. This is followed by a <span> element that has a CSS class that hides it on the page, and finally the actual text seen by the browser user. This is followed by the closing </a> tag for the hyperlink.

If we were scraping data from the page, we would probably be most interested in the text for the name of the contract vendor, IGF Vigilance Inc. By using the element inspector, we have identified that we would be looking for the visible display text within the hyperlink within the <td> tag. We can use a parser such as Beautiful Soup, or a regular expression, to isolate the desired text element, and grab it for further processing (Chapter 9 walks through how to scrape this page using a copy of the data saved so it will remain available).

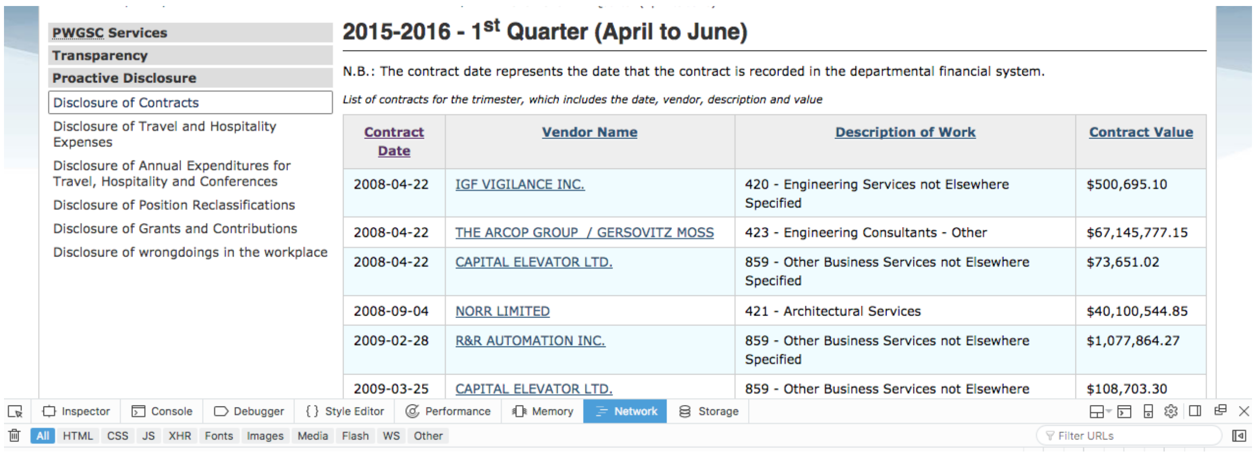
Using the Network monitor tab

As discussed in chapter 9, when you enter a web address in a browser, click on a link, or when a script running in the page requests an external resource, a request is sent to a remote server, somewhere else on the Internet, and a response comes back. You can use the Network panel in the developer tools to examine these requests and responses, so as to better understand what the browser is asking for, and what is being sent back in response.

We'll begin with a simple example, what happens when you click on one of the details links in the Proactive Disclosure page we've been looking at.

To start, open developer tools and click on the network tab.

Here's a look at the network panel:


PWGSC Services
Transparency
Proactive Disclosure
Disclosure of Contracts
Disclosure of Travel and Hospitality Expenses
Disclosure of Annual Expenditures for Travel, Hospitality and Conferences
Disclosure of Position Reclassifications
Disclosure of Grants and Contributions
Disclosure of wrongdoings in the workplace

2015-2016 - 1st Quarter (April to June)

N.B.: The contract date represents the date that the contract is recorded in the departmental financial system.

List of contracts for the trimester, which includes the date, vendor, description and value

Contract Date	Vendor Name	Description of Work	Contract Value
2008-04-22	IGF VIGILANCE INC.	420 - Engineering Services not Elsewhere Specified	$500,695.10
2008-04-22	THE ARCOP GROUP / GERSOVITZ MOSS	423 - Engineering Consultants - Other	$67,145,777.15
2008-04-22	CAPITAL ELEVATOR LTD.	859 - Other Business Services not Elsewhere Specified	$73,651.02
2008-09-04	NORR LIMITED	421 - Architectural Services	$40,100,544.85
2009-02-28	R&R AUTOMATION INC.	859 - Other Business Services not Elsewhere Specified	$1,077,864.27
2009-03-25	CAPITAL ELEVATOR LTD.	859 - Other Business Services not Elsewhere	$108,703.30

Within the panel there are a number of options are available.

Clicking on the small trash can icon will remove all request and response data from the panel.

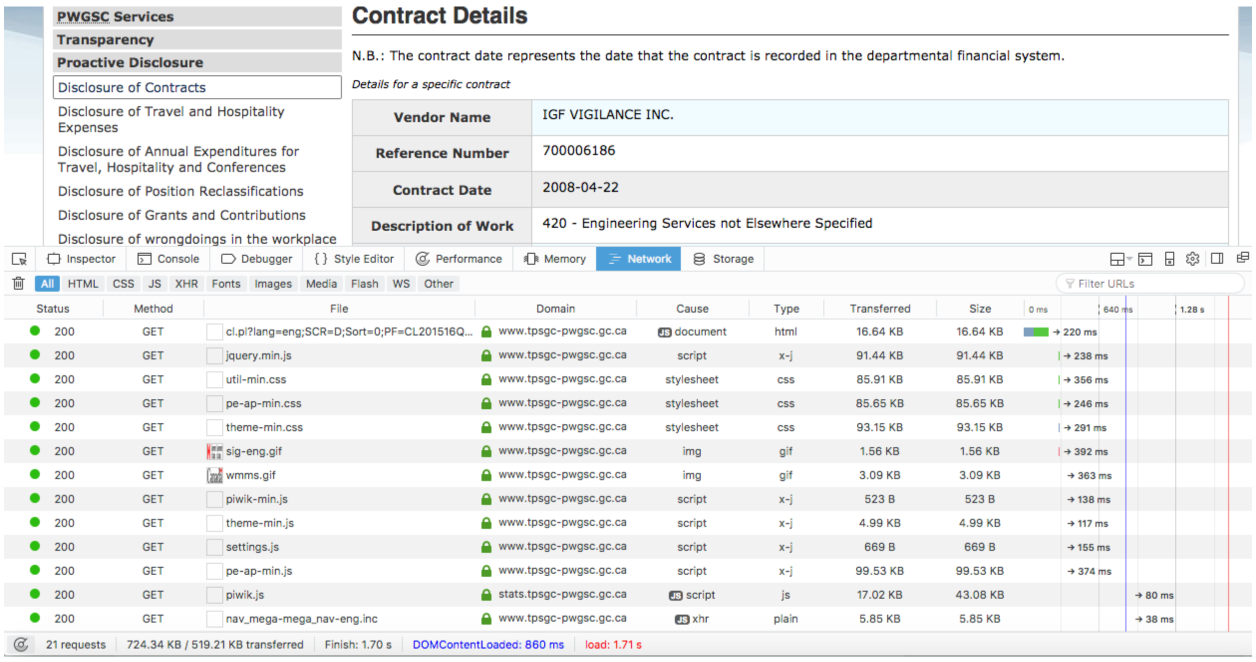
The remaining options filter the information displayed. For example, clicking on HTML limits the display to markup files such as HTML and XML. The CSS option limits the display to CSS files, the JS option to script files, the Images option to image files, and so on. The XHR option will show AJAX requests and responses.

The All option shows all request and response activity.

We'll look more closely at the HTML and XHR options.

Watching network traffic

The network panel is populated as soon as you take an action, or a script triggers an action, that makes a request to an external server. For example, if we click on the details link for IGF Vigilance Inc., seen above, the panel will display all information about requests and responses that follow the click.


PWGSC Services
Transparency
Proactive Disclosure
Disclosure of Contracts
Disclosure of Travel and Hospitality Expenses
Disclosure of Annual Expenditures for Travel, Hospitality and Conferences
Disclosure of Position Reclassifications
Disclosure of Grants and Contributions
Disclosure of wrongdoings in the workplace
Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
Storage
All
HTML
CSS
JS
XHR
Fonts
Images
Media
Flash
WS
Other
Filter URLs

Contract Details

N.B.: The contract date represents the date that the contract is recorded in the departmental financial system.

Details for a specific contract

Vendor Name	IGF VIGILANCE INC.
Reference Number	700006186
Contract Date	2008-04-22
Description of Work	420 - Engineering Services not Elsewhere Specified

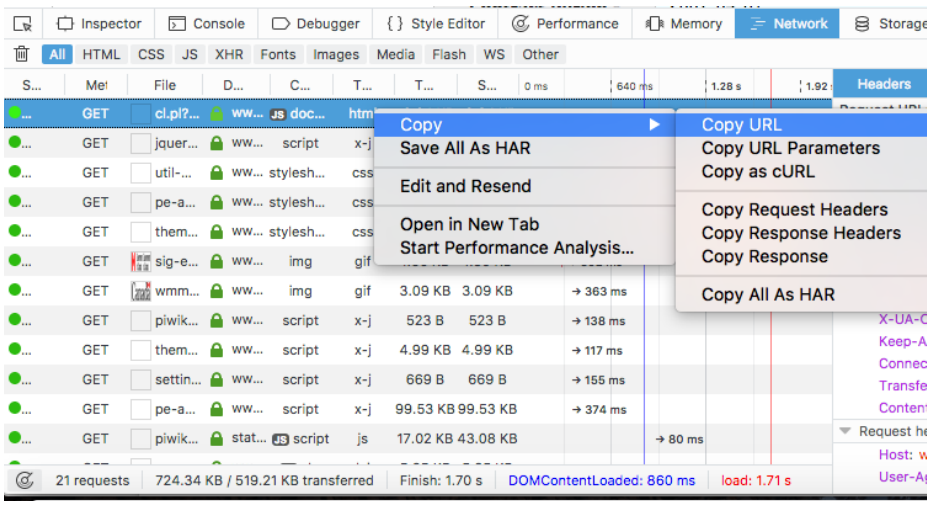
Status	Method	File	Domain	Cause	Type	Transferred	Size	0 ms 640 ms 1.28 s
● 200	GET	cl.pl?lang=eng;SCR=D;Sort=0;PF=CL201516Q...	www.tpsgc-pwgsc.gc.ca	document	html	16.64 KB	16.64 KB	→ 220 ms
● 200	GET	jquery.min.js	www.tpsgc-pwgsc.gc.ca	script	x-j	91.44 KB	91.44 KB	→ 238 ms
● 200	GET	util-min.css	www.tpsgc-pwgsc.gc.ca	stylesheet	css	85.91 KB	85.91 KB	→ 356 ms
● 200	GET	pe-ap-min.css	www.tpsgc-pwgsc.gc.ca	stylesheet	css	85.65 KB	85.65 KB	→ 246 ms
● 200	GET	theme-min.css	www.tpsgc-pwgsc.gc.ca	stylesheet	css	93.15 KB	93.15 KB	→ 291 ms
● 200	GET	sig-eng.gif	www.tpsgc-pwgsc.gc.ca	img	gif	1.56 KB	1.56 KB	→ 392 ms
● 200	GET	wmms.gif	www.tpsgc-pwgsc.gc.ca	img	gif	3.09 KB	3.09 KB	→ 363 ms
● 200	GET	piwik-min.js	www.tpsgc-pwgsc.gc.ca	script	x-j	523 B	523 B	→ 138 ms
● 200	GET	theme-min.js	www.tpsgc-pwgsc.gc.ca	script	x-j	4.99 KB	4.99 KB	→ 117 ms
● 200	GET	settings.js	www.tpsgc-pwgsc.gc.ca	script	x-j	669 B	669 B	→ 155 ms
● 200	GET	pe-ap-min.js	www.tpsgc-pwgsc.gc.ca	script	x-j	99.53 KB	99.53 KB	→ 374 ms
● 200	GET	piwik.js	stats.tpsgc-pwgsc.gc.ca	script	js	17.02 KB	43.08 KB	→ 80 ms
● 200	GET	nav_mega-mega_nav-eng.inc	www.tpsgc-pwgsc.gc.ca	xhr	plain	5.85 KB	5.85 KB	→ 38 ms

21 requests | 724.34 KB / 519.21 KB transferred | Finish: 1.70 s | DOMContentLoaded: 860 ms | load: 1.71 s

As you can see, the browser has gone to the page and the panel below has been populated with information about 21 requests made. Below, there is a summary of the requests which shows that the total size of the request and response traffic was about 724 KB. Of that, about 519 was transferred from the remote server. When you visit a website, your browser will cache some of the response files on your hard drive so that when you return to the site, the files can be loaded quickly without having to request them from the server again.

The File column shows the URL to which the request was made. It's a relative URL, but if you hover over it you will see the full URL. The Method column shows the request method, GET or POST.

If you <Ctrl> Click (right click on a PC) on the file cell for a request, a context sensitive menu appears that allows you to copy the URL to paste into a browser's address bar.



S...	Met	File	D...	C...	T...	T...	S...	0 ms 640 ms 1.28 s 1.92
●...	GET	cl.pl?...	ww...	doc...	htm			
●...	GET	jquer...	ww...	script	x-j			
●...	GET	util-...	ww...	stylesh...	css			
●...	GET	pe-a...	ww...	stylesh...	css			
●...	GET	them...	ww...	stylesh...	css			
●...	GET	sig-e...	ww...	img	gif			
●...	GET	wmm...	ww...	img	gif	3.09 KB	3.09 KB	→ 363 ms
●...	GET	piwik...	ww...	script	x-j	523 B	523 B	→ 138 ms
●...	GET	them...	ww...	script	x-j	4.99 KB	4.99 KB	→ 117 ms
●...	GET	settin...	ww...	script	x-j	669 B	669 B	→ 155 ms
●...	GET	pe-a...	ww...	script	x-j	99.53 KB	99.53 KB	→ 374 ms
●...	GET	piwik...	stat...	script	js	17.02 KB	43.08 KB	→ 80 ms

21 requests | 724.34 KB / 519.21 KB transferred | Finish: 1.70 s | DOMContentLoaded: 860 ms | load: 1.71 s

This feature is useful for examining files that are included as part of a response, such as JavaScript files, CSS style files, images, and other files.

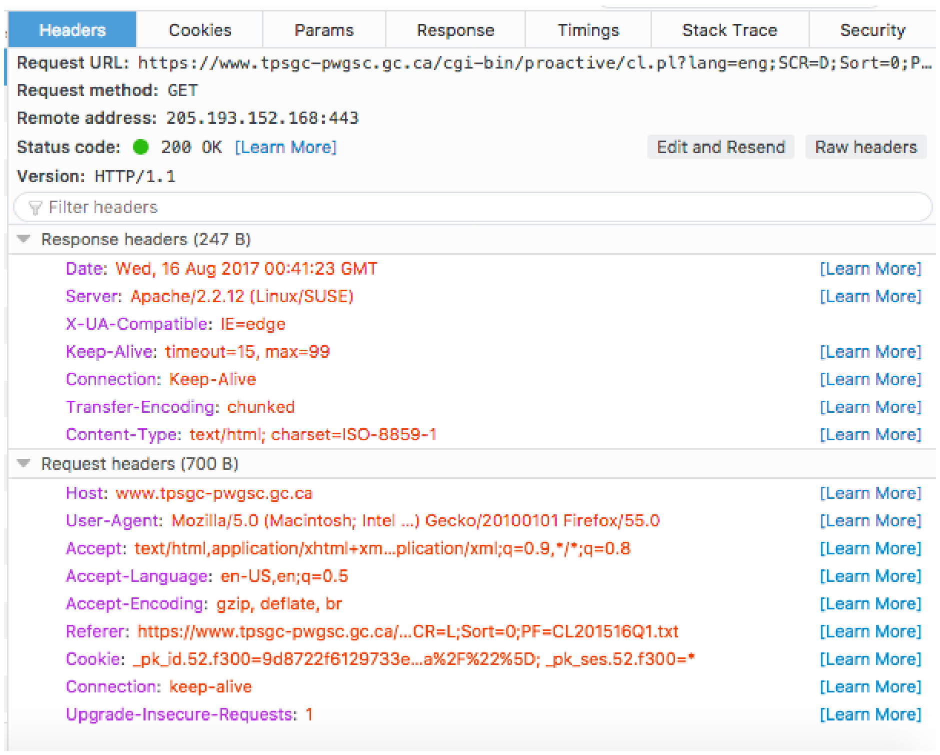
The Status column reports on the status of the HTTP request. The code 200 means the request was successful. 304 indicates that the file was fetched from the browser cache. 404 means the resource did not exist on the server. You can read about all of the different possible status codes here: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

The Domain column indicates which Internet domain responded to the request.

The Size column indicates the size of the request and the returned response, in Bytes.

The time scale on the far right tells you how long the request took in milliseconds.

Clicking on one of the request rows will open a request details panel to the right (it can also be opened with the icon. It has a number of tabs of its own. There's a lot of technical stuff here, but we'll make note of a few things.



```
Request URL: https://www.tpsgc-pwgsc.gc.ca/cgi-bin/proactive/cl.pl?lang=eng;SCR=D;Sort=0;P…
Request method: GET
Remote address: 205.193.152.168:443
Status code: ● 200 OK [Learn More]
Version: HTTP/1.1

Response headers (247 B)
    Date: Wed, 16 Aug 2017 00:41:23 GMT                                   [Learn More]
    Server: Apache/2.2.12 (Linux/SUSE)                                    [Learn More]
    X-UA-Compatible: IE=edge
    Keep-Alive: timeout=15, max=99                                        [Learn More]
    Connection: Keep-Alive                                                [Learn More]
    Transfer-Encoding: chunked                                            [Learn More]
    Content-Type: text/html; charset=ISO-8859-1                           [Learn More]
Request headers (700 B)
    Host: www.tpsgc-pwgsc.gc.ca                                           [Learn More]
    User-Agent: Mozilla/5.0 (Macintosh; Intel …) Gecko/20100101 Firefox/55.0   [Learn More]
    Accept: text/html,application/xhtml+xm…plication/xml;q=0.9,*/*;q=0.8  [Learn More]
    Accept-Language: en-US,en;q=0.5                                       [Learn More]
    Accept-Encoding: gzip, deflate, br                                    [Learn More]
    Referer: https://www.tpsgc-pwgsc.gc.ca/…CR=L;Sort=0;PF=CL201516Q1.txt [Learn More]
    Cookie: _pk_id.52.f300=9d8722f6129733e…a%2F%22%5D; _pk_ses.52.f300=*  [Learn More]
    Connection: keep-alive                                                [Learn More]
    Upgrade-Insecure-Requests: 1                                          [Learn More]
```

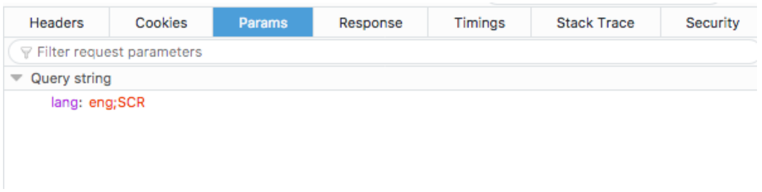
Under the Headers tab, the Remote Address entry indicates the IP address of the server that sent the response. This can also be copied and pasted into the address bar of a web browser. We can also see the HTML request and response headers. HTML headers contain information used by the client and server computers. They are hidden from users in the normal day-to-day use of a browser.

Let's look at the request headers first.

The Host is the domain of the server. The referrer is the page from which the request came (we clicked on a link). Most important for our purposes is the User Agent. This is a character string that represents the type of browser, including the version, that made the request. Some web servers will ignore requests that don't come from actual browsers. It is possible to set the header string sent by a script so it will appear to be a real browser. You can copy the user agent string from this panel, and paste it into a script. Of course, doing so raises potential ethical issues, and these are discussed in Chapter 9.

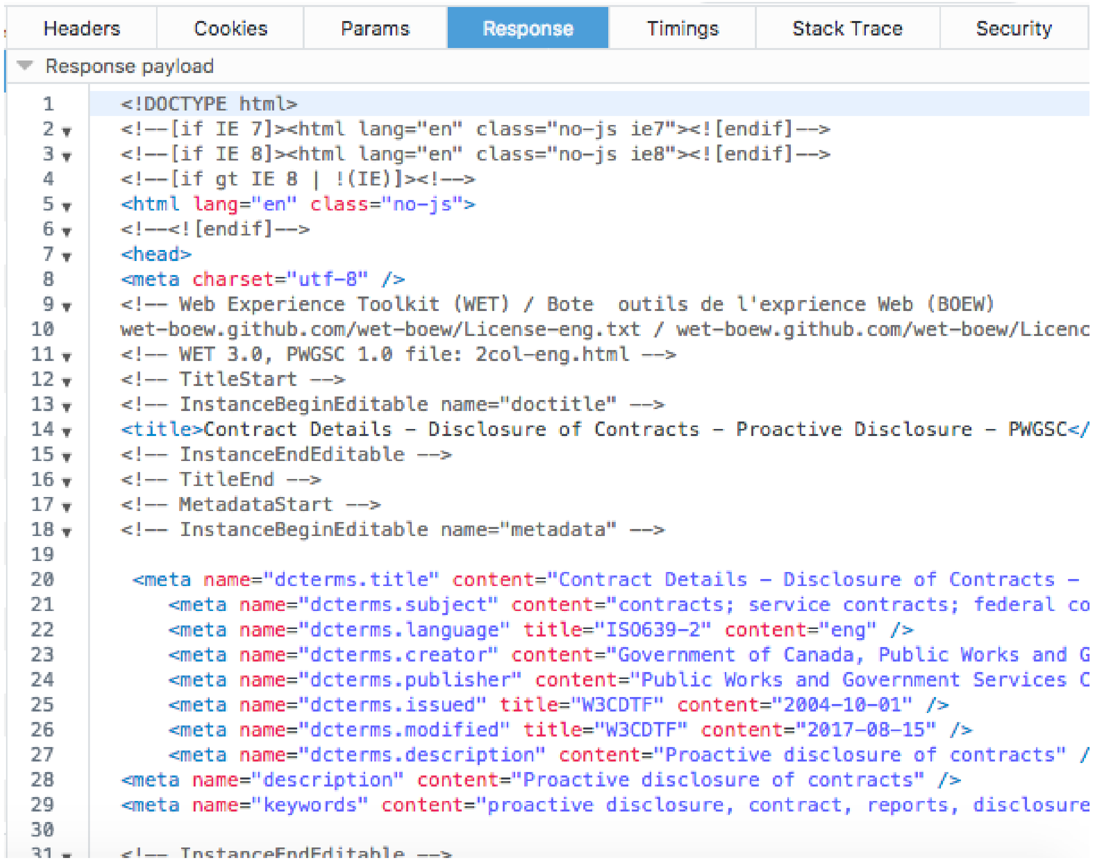
The response headers tell us the type of content in the response, in this case an HTML page, the character encoding, in this case ISO-859-1, the time and date of the request given in coordinated universal time, or GMT, and other technical details.

The Params tab contains the parameters that were passed to the external web server as part of the URL for the request. This can be extremely useful for understanding what you might need to send to the server in a scraping script.



Query string
lang: eng;SCR

The Response tab contains the actual content of the response, in this case the HTML page that was sent back.



Response payload

```
<!DOCTYPE html>
<!--[if IE 7]><html lang="en" class="no-js ie7"><![endif]-->
<!--[if IE 8]><html lang="en" class="no-js ie8"><![endif]-->
<!--[if gt IE 8 | !(IE)]><!-->
<html lang="en" class="no-js">
<!--<![endif]-->
<head>
<meta charset="utf-8" />
<!-- Web Experience Toolkit (WET) / Bote  outils de l'exprience Web (BOEW)
wet-boew.github.com/wet-boew/License-eng.txt / wet-boew.github.com/wet-boew/Licenc
<!-- WET 3.0, PWGSC 1.0 file: 2col-eng.html -->
<!-- TitleStart -->
<!-- InstanceBeginEditable name="doctitle" -->
<title>Contract Details - Disclosure of Contracts - Proactive Disclosure - PWGSC</
<!-- InstanceEndEditable -->
<!-- TitleEnd -->
<!-- MetadataStart -->
<!-- InstanceBeginEditable name="metadata" -->

 <meta name="dcterms.title" content="Contract Details - Disclosure of Contracts -
    <meta name="dcterms.subject" content="contracts; service contracts; federal co
    <meta name="dcterms.language" title="ISO639-2" content="eng" />
    <meta name="dcterms.creator" content="Government of Canada, Public Works and G
    <meta name="dcterms.publisher" content="Public Works and Government Services C
    <meta name="dcterms.issued" title="W3CDTF" content="2004-10-01" />
    <meta name="dcterms.modified" title="W3CDTF" content="2017-08-15" />
    <meta name="dcterms.description" content="Proactive disclosure of contracts" /
<meta name="description" content="Proactive disclosure of contracts" />
<meta name="keywords" content="proactive disclosure, contract, reports, disclosure

<!-- InstanceEndEditable -->
```

This may seem superfluous when the response is a plain HTML page, but sometimes you will want to see the response made to an XHR (XMLHttpRequest) request when AJAX is being used, the data sent back in response to a search request, or the contents of a JavaScript or CSS file.

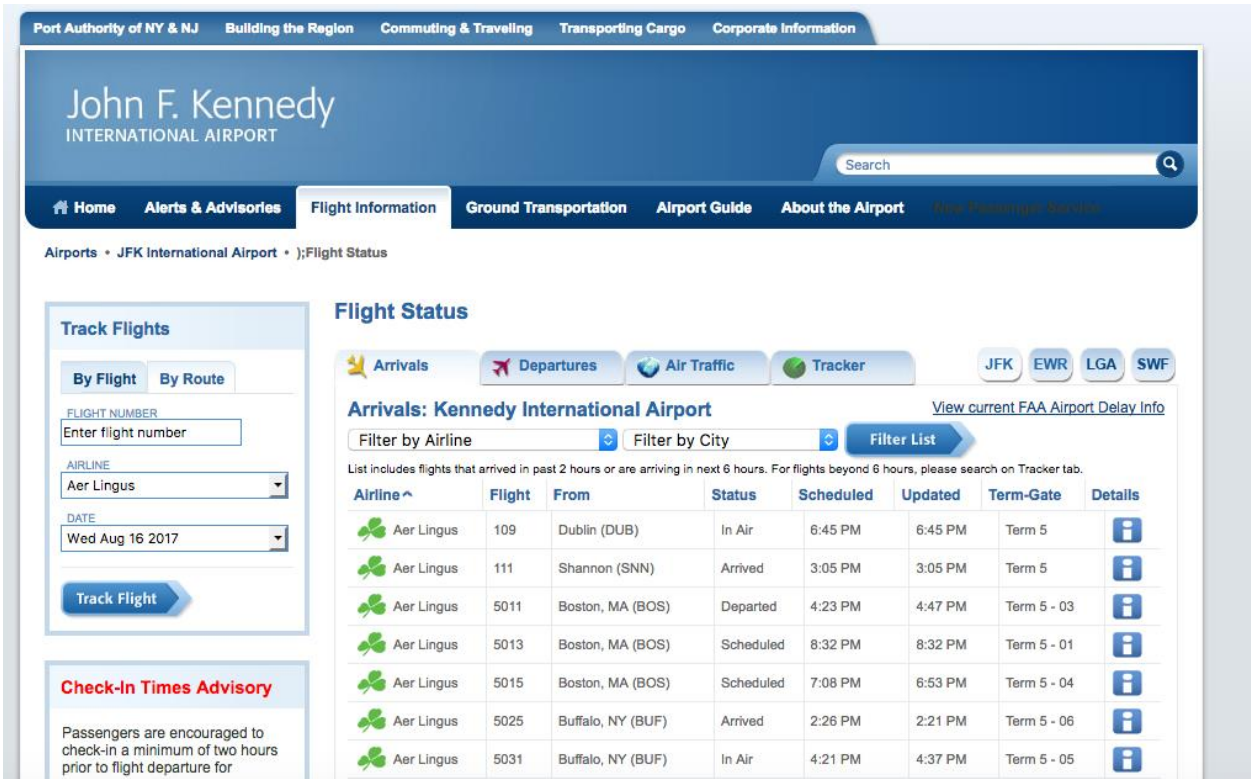
The Timings tab provides information on how long the request and response took.

When a page isn't completely reloaded

As discussed in Chapter 9, not all web sites request a whole new page each time a user interacts with the page. Instead, data may be requested and passed back to the browser without having to reload the page. This can be done in different ways. For example, a request may be made for an external html resource that is then placed in the main page by way of an iframe. Or a site may use a technology called AJAX. When these kinds of methods are being used, opening and looking at the HTML source for a page will only show you the original HTML that was sent by the server. Any changes to the DOM (document object model), which could be thought of as the current state of the web page the browser has stored in memory, will not be reflected. The network panel, however, can allow you to peek under the hood, and see everything that is happening.

Let's say you wanted to have a look at flight arrivals and departures and John F. Kennedy Airport in New York. You could go to the website maintained by the Port Authority of New York and New Jersey at https://www.panynj.gov/airports/flight-status.html

You may notice that the actual flight status information, which comes from an outside provider, is slightly delayed in loading compared to the rest of the page. This is because you are being provided with the latest information, and it is being added after the main page loads. Here is what the fully loaded page, with the embedded flight information looks like:



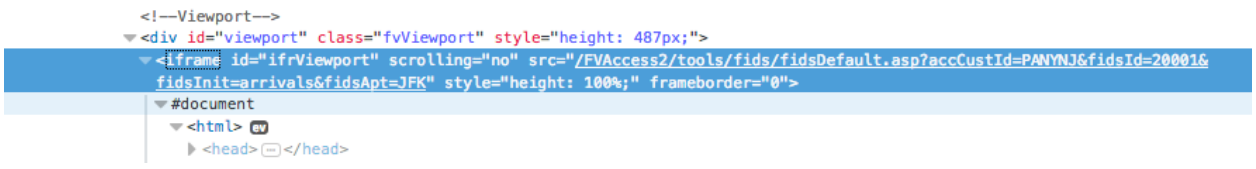
Passengers are encouraged to check-in a minimum of two hours prior to flight departure for

Arrivals: Kennedy International Airport

List includes flights that arrived in past 2 hours or are arriving in next 6 hours. For flights beyond 6 hours, please search on Tracker tab.

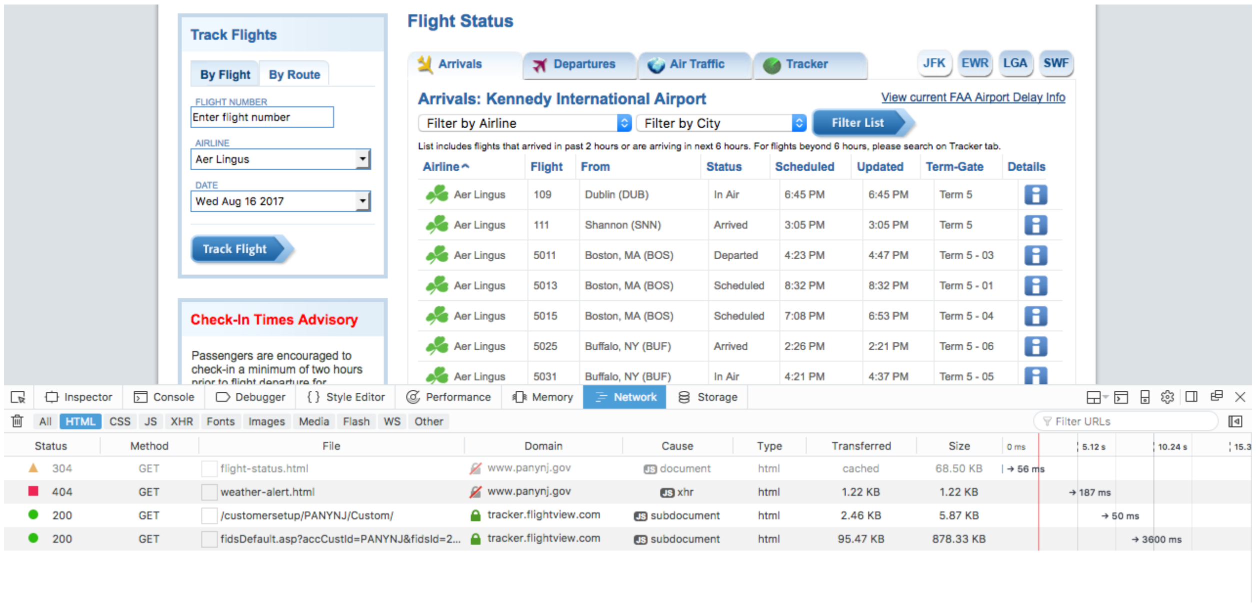
Airline ^	Flight	From	Status	Scheduled	Updated	Term-Gate	Details
Aer Lingus	109	Dublin (DUB)	In Air	6:45 PM	6:45 PM	Term 5	ℹ
Aer Lingus	111	Shannon (SNN)	Arrived	3:05 PM	3:05 PM	Term 5	ℹ
Aer Lingus	5011	Boston, MA (BOS)	Departed	4:23 PM	4:47 PM	Term 5 - 03	ℹ
Aer Lingus	5013	Boston, MA (BOS)	Scheduled	8:32 PM	8:32 PM	Term 5 - 01	ℹ
Aer Lingus	5015	Boston, MA (BOS)	Scheduled	7:08 PM	6:53 PM	Term 5 - 04	ℹ
Aer Lingus	5025	Buffalo, NY (BUF)	Arrived	2:26 PM	2:21 PM	Term 5 - 06	ℹ
Aer Lingus	5031	Buffalo, NY (BUF)	In Air	4:21 PM	4:37 PM	Term 5 - 05	ℹ

If we use the element inspector to examine the area of the page that contains the flight information, we can see that contained within the <div> tag with the id of "viewport" there is an iframe that has as its source, a different URL. It's a relative URL, so we can't see the domain, but we can see that by using the network monitor.



```
<!--Viewport-->
<div id="viewport" class="fvViewport" style="height: 487px;">
  <iframe id="ifrViewport" scrolling="no" src="/FVAccess2/tools/fids/fidsDefault.asp?accCustId=PANYNJ&fidsId=20001&
  fidsInit=arrivals&fidsApt=JFK" style="height: 100%;" frameborder="0">
    #document
      <html>
        <head></head>
```

We'll switch to the network tab in developer tools, then choose the HTML filter (if you go straight to the network tab after opening developer tools, you may need to reload the page in the browser before the network panel populates). We're using the HTML filter because an iframe is used to embed an HTML document.



Arrivals: Kennedy International Airport

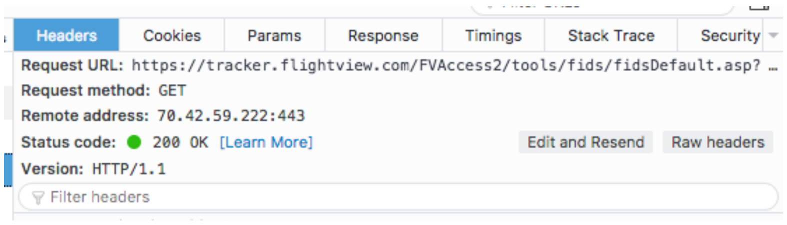
List includes flights that arrived in past 2 hours or are arriving in next 6 hours. For flights beyond 6 hours, please search on Tracker tab.

Airline	Flight	From	Status	Scheduled	Updated	Term-Gate	Details
Aer Lingus	109	Dublin (DUB)	In Air	6:45 PM	6:45 PM	Term 5	
Aer Lingus	111	Shannon (SNN)	Arrived	3:05 PM	3:05 PM	Term 5	
Aer Lingus	5011	Boston, MA (BOS)	Departed	4:23 PM	4:47 PM	Term 5 - 03	
Aer Lingus	5013	Boston, MA (BOS)	Scheduled	8:32 PM	8:32 PM	Term 5 - 01	
Aer Lingus	5015	Boston, MA (BOS)	Scheduled	7:08 PM	6:53 PM	Term 5 - 04	
Aer Lingus	5025	Buffalo, NY (BUF)	Arrived	2:26 PM	2:21 PM	Term 5 - 06	
Aer Lingus	5031	Buffalo, NY (BUF)	In Air	4:21 PM	4:37 PM	Term 5 - 05	

Status	Method	File	Domain	Cause	Type	Transferred	Size	0 ms 5.12 s 10.24 s 15.3
304	GET	flight-status.html	www.panynj.gov	document	html	cached	68.50 KB	→ 56 ms
404	GET	weather-alert.html	www.panynj.gov	xhr	html	1.22 KB	1.22 KB	→ 187 ms
200	GET	/customersetup/PANYNJ/Custom/	tracker.flightview.com	subdocument	html	2.46 KB	5.87 KB	→ 50 ms
200	GET	fidsDefault.asp?accCustId=PANYNJ&fidsId=2...	tracker.flightview.com	subdocument	html	95.47 KB	878.33 KB	→ 3600 ms

We can see that four files were transferred, including one that is almost a megabyte in size. We'll guess that might be the file we want, because there's quite a bit of arrivals data.

If we now open the request details panel by clicking on the row for the request, we can see that the request was made to a URL that does not appear in the address bar of our web browser.

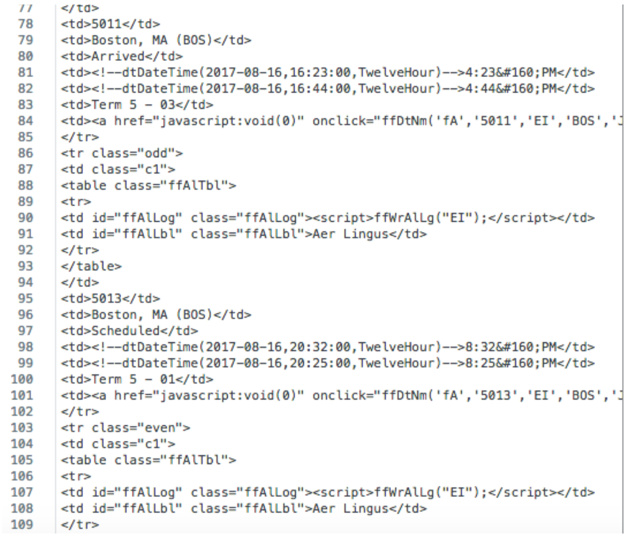


```
Request URL: https://tracker.flightview.com/FVAccess2/tools/fids/fidsDefault.asp? …
Request method: GET
Remote address: 70.42.59.222:443
Status code: ● 200 OK [Learn More]
Version: HTTP/1.1
```

This is the full URL:

https://tracker.flightview.com/FVAccess2/tools/fids/fidsDefault.asp?accCustId=PANYNJ&fidsId=20001&fidsInit=arrivals&fidsApt=JFK

Now, if we click on the Response tab, we can see the HTML filed that was transferred.



```
</td>
<td>5011</td>
<td>Boston, MA (BOS)</td>
<td>Arrived</td>
<td><!--dtDateTime(2017-08-16,16:23:00,TwelveHour)-->4:23&#160;PM</td>
<td><!--dtDateTime(2017-08-16,16:44:00,TwelveHour)-->4:44&#160;PM</td>
<td>Term 5 - 03</td>
<td><a href="javascript:void(0)" onclick="ffDtNm('fA','5011','EI','BOS','
</tr>
<tr class="odd">
<td class="c1">
<table class="ffAlTbl">
<tr>
<td id="ffAlLog" class="ffAlLog"><script>ffWrAlLg("EI");</script></td>
<td id="ffAlLbl" class="ffAlLbl">Aer Lingus</td>
</tr>
</table>
</td>
<td>5013</td>
<td>Boston, MA (BOS)</td>
<td>Scheduled</td>
<td><!--dtDateTime(2017-08-16,20:32:00,TwelveHour)-->8:32&#160;PM</td>
<td><!--dtDateTime(2017-08-16,20:25:00,TwelveHour)-->8:25&#160;PM</td>
<td>Term 5 - 01</td>
<td><a href="javascript:void(0)" onclick="ffDtNm('fA','5013','EI','BOS','
</tr>
<tr class="even">
<td class="c1">
<table class="ffAlTbl">
<tr>
<td id="ffAlLog" class="ffAlLog"><script>ffWrAlLg("EI");</script></td>
<td id="ffAlLbl" class="ffAlLbl">Aer Lingus</td>
</tr>
```

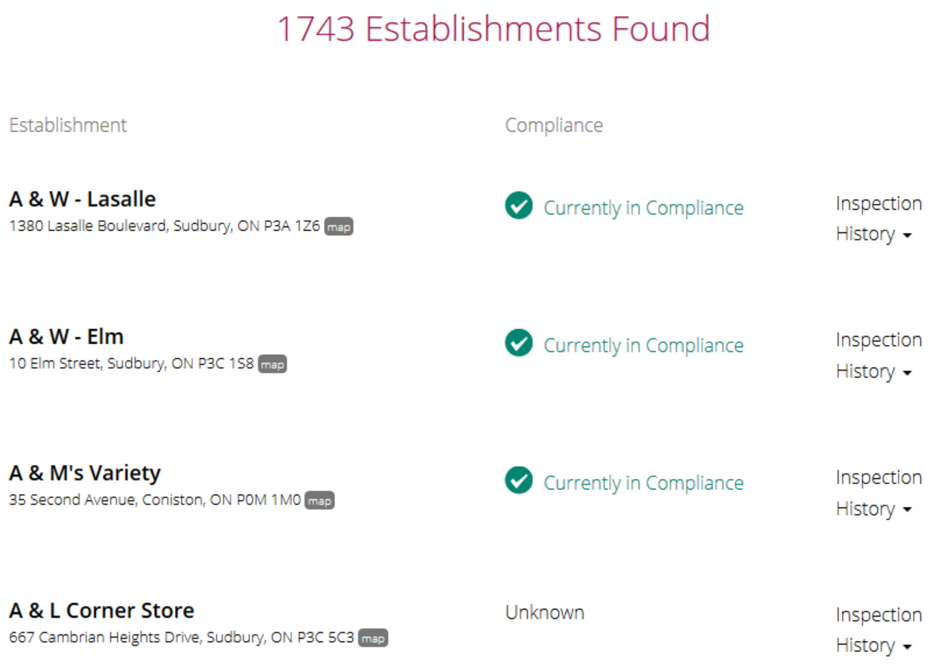
If we like, we can copy and paste the HTML into a separate file, if you wished to retain it for future reference.

Looking at AJAX

As we discussed above, another way used to insert information into an existing web page is the AJAX protocol.

We'll take a look at a site used by the Sudbury District Health Unit in the northern part of the Canadian province of Ontario to provide information on restaurant inspections to the public. The page now contains archived information, and will be replaced by a new site, according to the municipality.

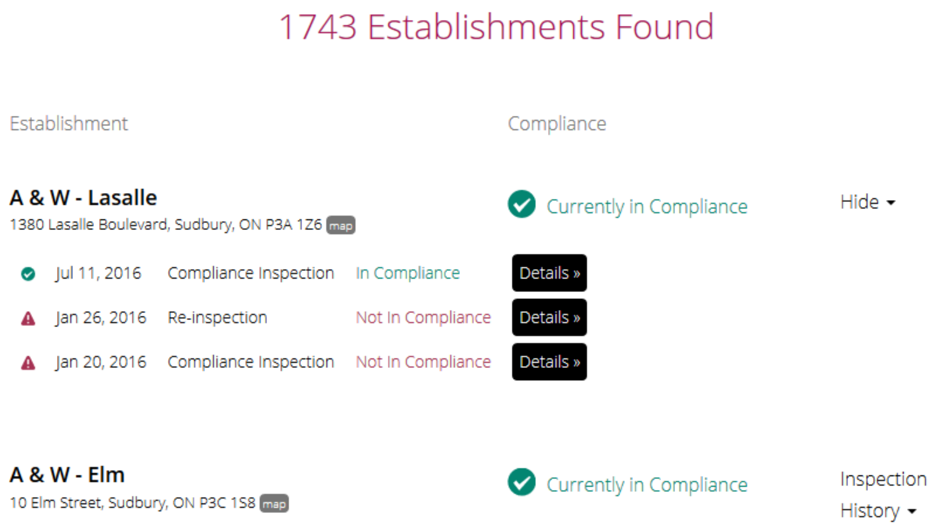
The site showed the restaurants in the city, and whether each is in compliance. You could also access a more detailed inspection history for each premise.



1743 Establishments Found

Establishment	Compliance	
A & W - Lasalle 1380 Lasalle Boulevard, Sudbury, ON P3A 1Z6 map	✓ Currently in Compliance	Inspection History
A & W - Elm 10 Elm Street, Sudbury, ON P3C 1S8 map	✓ Currently in Compliance	Inspection History
A & M's Variety 35 Second Avenue, Coniston, ON P0M 1M0 map	✓ Currently in Compliance	Inspection History
A & L Corner Store 667 Cambrian Heights Drive, Sudbury, ON P3C 5C3 map	Unknown	Inspection History

If you click on the Inspection History link for a premise, the page was populated with individual inspections for that facility, complete with more links to delve down to more precise detail.



1743 Establishments Found

Establishment	Compliance	
A & W - Lasalle 1380 Lasalle Boulevard, Sudbury, ON P3A 1Z6 map	✓ Currently in Compliance	Hide

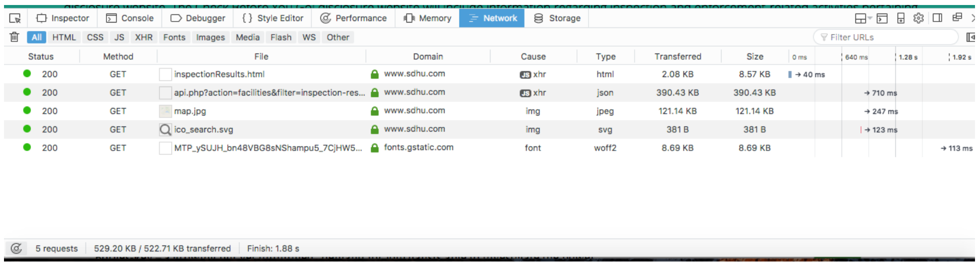
✓	Jul 11, 2016	Compliance Inspection	In Compliance	Details »
▲	Jan 26, 2016	Re-inspection	Not In Compliance	Details »
▲	Jan 20, 2016	Compliance Inspection	Not In Compliance	Details »

A & W - Elm 10 Elm Street, Sudbury, ON P3C 1S8 map	✓ Currently in Compliance	Inspection History

An examination of the page source, however, shows that the detail listed is not present. The DOM of the page is being manipulated using JavaScript that runs based on user interactions, such as clicking on an Inspection History link. New data is being added to the page dynamically.

We can have a closer look at what is going on using the Network monitor in Firefox's developer tools.

When we first load the page, we can see that five requests were made to the server, totalling 529 kb of data sent and received.



Status	Method	File	Domain	Cause	Type	Transferred	Size	0 ms / 640 ms / 1.28 s / 1.92 s
● 200	GET	inspectionResults.html	www.sdhu.com	xhr	html	2.08 KB	8.57 KB	→ 40 ms
● 200	GET	api.php?action=facilities&filter=inspection-res...	www.sdhu.com	xhr	json	390.43 KB	390.43 KB	→ 710 ms
● 200	GET	map.jpg	www.sdhu.com	img	jpeg	121.14 KB	121.14 KB	→ 247 ms
● 200	GET	ico_search.svg	www.sdhu.com	img	svg	381 B	381 B	→ 123 ms
● 200	GET	MTP_ySUJH_bn48VBG8sNShampu5_7CjHW5...	fonts.gstatic.com	font	woff2	8.69 KB	8.69 KB	→ 113 ms

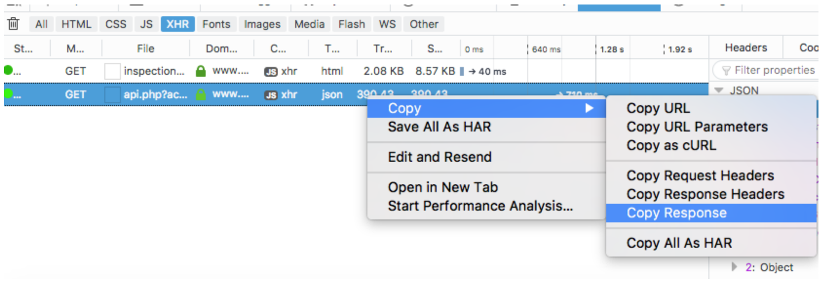
5 requests | 529.20 KB / 522.71 KB transferred | Finish: 1.88 s

The site used to serve up a great many more resources, but has been reduced in scope as it only archives date up till January 2017.

As you can see, two of the files transferred have XHR under type. XHR request is another way of saying AJAX request.

If we switch to the XHR tab, we can see just these two and we can also see that one of the files is an html file and other a JSON file. JSON stands for JavaScript Object Notation, which as discussed in Chapter 2 is a plain-text data format often used when the data is intended to be machine read.

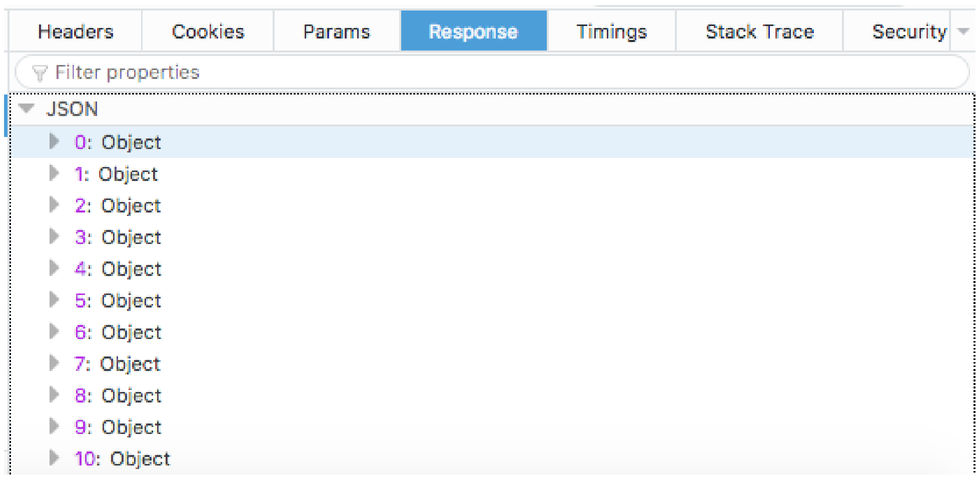
If we want to see the whole response, we can <ctrl> click (right click on a Windows computer) on the row for the JSON file in the network panel, and choose Copy>Response.



St...	M...	File	Dom...	C...	T...	Tr...	S...	0 ms / 640 ms / 1.28 s / 1.92 s
● ...	GET	inspection...	www....	xhr	html	2.08 KB	8.57 KB	→ 40 ms
● ...	GET	api.php?ac...	www....	xhr	json	390.43	390.43	→ 710 ms

We can then paste the resulting JSON into a separate file to examine further. You can then use a Python script or an online converter to convert the JSON to a CSV file for import into a spreadsheet or database program. In this way, you are using the web development tool as a rudimentary web scraping tool. For a few pages, it could be all you need.

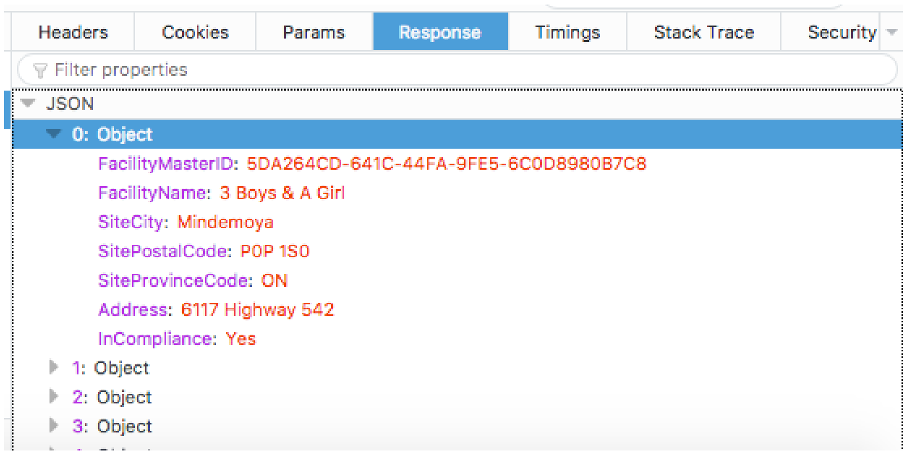
If we like, we can also look at the response in the request details window. This allows us to examine each individual object in the JSON file.



```
JSON
  0: Object
  1: Object
  2: Object
  3: Object
  4: Object
  5: Object
  6: Object
  7: Object
  8: Object
  9: Object
  10: Object
```

An object in this context is a data construct that holds all of the information for one row of data, that is, one facility that was inspected.

If we click on the expansion triangle for object 0, we can see the data contained in that object, neatly organized.



```
JSON
  0: Object
      FacilityMasterID: 5DA264CD-641C-44FA-9FE5-6C0D8980B7C8
      FacilityName: 3 Boys & A Girl
      SiteCity: Mindemoya
      SitePostalCode: P0P 1S0
      SiteProvinceCode: ON
      Address: 6117 Highway 542
      InCompliance: Yes
  1: Object
  2: Object
  3: Object
```

This makes it trivially easy to examine the data structure.

There is a great deal more that the Firefox developer tools, and those in other browsers, can do, including helping you develop and debug JavaScript, manipulate the HTML and CSS of the page to see, in real time, how the change would affect the page, and much more that we simply couldn't cover here. These are remarkable tools and belong in every data journalist's toolkit.