

Excel and the World Wide Web

Straight to the Point

Eduardo N Sanchez



Straight to the Point

The Straight to the Point e-books are designed to thoroughly cover one targeted aspect of Excel.

Excel and The World Wide Web Straight to the Point

Eduardo Sanchez

Holy Macro! Books
PO Box 541731, Merritt Island FL 32953

Excel and the World Wide Web Straight to the Point

© 2021 by Tickling Keys, Inc. and Eduardo N. Sanchez

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information or storage retrieval system without written permission from the publisher.

All terms known in this book known to be trademarks have been appropriately capitalized. Trademarks are the property of their respective owners and are not affiliated with Holy Macro! Books

Every effort has been made to make this book as complete and accurate as possible, but no warranty or fitness is implied. The information is provided on an “as is” basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

First Published: January 2021

Author: Eduardo Sanchez

Cover Design: Suat M. Ozgur

Cover Illustration: Shannon Mattiza, 6'4 Productions

Indexer: Nellie Jay

Compositor: Jill Cabot

Published by: Holy Macro! Books, PO Box 541731, Merritt Island, FL 32953

Distributed by Independent Publishers Group, Chicago, IL

ISBN 978-1-61547-158-4

TABLE OF CONTENTS

About the Author	1
Acknowledgments	1
Getting Started	2
What is HTML?.....	2
Performing Queries	4
A static query	5
A dynamic web query.....	6
Querying with VBA.....	8
Interacting with Sites without using a Browser	10
What is JSON?.....	10
Getting data from the Web programmatically.....	10
Getting data from Web page elements	14
Posting data over the Internet	15
Authenticated requests	16
Internet Explorer and VBA.....	18
Transferring a Web table to the worksheet.....	19
Working with events	21
Introducing Selenium	24
Installing Selenium	24
Internet Explorer and Selenium	26
Google Chrome and VBA	28
Running JavaScript on a page	28
Iframes and transferring tables without a loop	30
Using XPath to find elements.....	32
Microsoft Edge and VBA.....	35
Finding elements by CSS selectors.....	36
Downloading a file	37
Creating a PDF file from scratch	39
Wrapping up Selenium	40
Power Query and the Web.....	42
Connecting to a Web table – the direct way	42
Connecting to a Web table – the long way	45
Conclusion.....	51
Index.....	52

This page intentionally left blank. The Introduction starts on the next page.

About the Author

Eduardo N. Sanchez is a Brazilian Chemical Engineer with a long-time penchant for computers. Since college he has been programming using languages such as Fortran, Basic, Algol, Pascal, C and most recently VBA.

Eduardo enjoys answering Excel and Office questions at Internet forums, as this is a great way to get in touch with topics that could otherwise remain unnoticed. When he is not tackling a computational challenge with electronic music in the background, Eduardo likes to watch Star Trek episodes and play with the family's Dachshund.

Acknowledgments

I want to thank Bill Jelen for believing in this project when he knew nothing at all about me. Thanks also to all blog posters and Internet site owners who share valuable technical knowledge for free; even the most experienced user will benefit from reading well-constructed articles. After retirement I plan to build my own little Excel corner online...

Finally, thanks to my family for being always supportive on all my endeavors, and to my co-workers and Web posters who keep me on my toes by asking Excel questions of varying difficulty levels.

Code used in this book

The code examples are available for download at

<https://www.mrexcel.com/download-center/excel-and-the-world-wide-web-8051/>

GETTING STARTED

In the good old days, when personal computers had no hard drives and monitors were low resolution and monochromatic, there was no Internet. Then it started, shyly at first, and that was a time when people were viewing sites with Netscape and searching with Altavista.

Nowadays, on the other hand, the World Wide Web is everywhere, and it is hard to find an electronic device that is not connect to it in one way or another. The aim of this book is to provide you with tools that will automate repetitive tasks and boost your productivity when working with Excel and the Web.

Most of the techniques shown here can be found online, scattered all over the Web. Here you will find them logically organized and all in one place.

What is HTML?

HTML stands for Hyper Text Markup Language and is used to build websites. Have you ever wondered what hypertext is? It is a content that behaves in a non-linear way. Imagine a web site where each page has several links to other pages, both on the same site or not. The user will navigate by jumping from one page to another; this is hypertext behavior.

A conventional print book is the counter example, it is supposed to be read sequentially.

The markup part stems from the fact that information in HTML is marked with tags; we will see more on this shortly.

There are other languages used to help create websites, such as CSS and Java Script, but we are not going to deal with them in detail.

Note: CSS means Cascading Style Sheet and takes care of the graphical part of web pages. It works together with HTML, which will be responsible for the page contents. Cascading means that it is possible to use several CSS files to create the final visual style. This language will manipulate elements like font size, background images and the color palette.

JavaScript is a powerful programming language used to implement dynamic behavior on web sites. With it, developers can manipulate the page content, produce charts and interact with APIs (*Application Programming Interface*), among other tasks. Note that JS and Java are two different languages.

It is even said that in the future JS could replace VBA as the Office programming language.

One great thing about current browsers is that they expose their source code. That is correct, you can visualize the corresponding HTML that was written to create the page.

If you are using Google Chrome, just right click any page element and select inspect on the menu that appears; a pane similar to the one shown below is displayed.

```

<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <!-- Google Tag Manager (noscript) -->
    <noscript>...</noscript>
    <!-- End Google Tag Manager (noscript) -->
    <div class="d-none d-block d-lg-none" style="height: 37px;"></div>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top d-none d-block d-lg-none" style=...>
    </nav>
    <header class="header d-none d-md-block">...</header>
    <br>
    <div class="container-fluid">
      <div class="row">
        <aside id="left-sidebar" class="col-lg-3 order-10 order-lg-first">
          <div class="card mb-1 d-none d-lg-block ">
            <nav class="card-body p-2">
              <h4 class="card-title">...</h4>
              <p class="card-text">A wealth of Excel knowledge at your fingertips...</p>
              <hr>
              <ul class="navbar-nav">
                <li class="nav-item ">...</li>
                <li class="nav-item ">...</li>
                <li class="nav-item ">...</li>
                <li class="nav-item ">...</li>
                <li class="nav-item ">
                  <a class="nav-link" href="https://www.mrexcel.com/excel-tips/">...</a> == $0
                </li>
                <li class="nav-item">...</li>
                <li class="nav-item">...</li>
              </ul>
            </nav>
          </div>
        </aside>
      </div>
    </div>
  </body>
</html>

```

html body div div #left-sidebar div nav.card-body.p-2 ul.navbar-nav li.nav-item a.nav-link

A piece of HTML from a web page.

A few things to note:

- The keywords header, div, aside, nav, h4, li and others are all tags.
- When the little black arrows are pointing to the right, this means they can be clicked to expand and display more information.
- The keywords id, class and href are attributes.
- <li is an opening tag, whereas is a closing one.
- An element can belong to several different classes.

If you are feeling overwhelmed by all of this, do not panic; it is not necessary to be an HTML programmer to work with this kind of automation. Later we will see how to read source HTML and extract the necessary information to make our VBA code functional.

Although Web designers use professional editors, it is possible to create simple HTML files with standard Windows applications such as Notepad. This can be useful when you want to test some VBA code that will interact with a Web page but for some reason the real site is not available.

To generate a local HTML file, follow these steps:

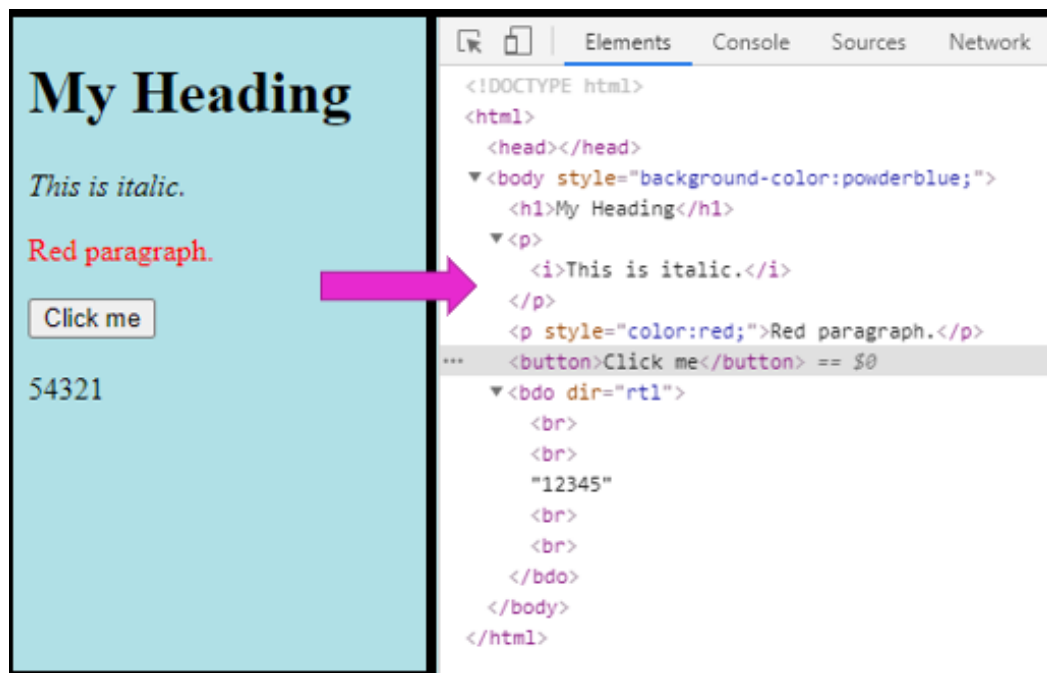
- Open Notepad and type your source code; an example is given below.
- Save it with a htm extension.

- Open the saved file with your favorite browser.
- To later edit the source code, rename the file to a txt extension.

This is the sample HTML:

```
<!DOCTYPE html>
<html>
<body style="background-color:powderblue;">
<h1>My Heading</h1>
<p><i>This is italic.</i></p>
<p style="color:red;">Red paragraph.</p>
<button>Click me</button>
<bdo dir="rtl"><br><br>12345<br><br></bdo>
</body>
</html>
```

Now this is the finished product, viewed with Chrome:



Generating a local HTML file.

Note that the Chrome HTML pane will nicely format the code that you originally typed back in Notepad.

Performing Queries

A Web query allows us to pull information stored in an HTML file. Normally it is used to connect to some web site, but it is also possible to query a local file.

If you plan to extract data from the same source over and over again, save the query and just refresh it anytime you want the most recent information. Excel web queries work by identifying tables on the selected page and offering the option to import the ones you want to the workbook.

A static query

The standard Excel query is static, meaning it will not accept parameters. The next section will deal with dynamic queries.

To perform a static web query, follow these steps:

1. Go to the Data tab of the ribbon and click Get External Data
2. Choose from Web when the options are displayed
3. You will be presented with the dialog box shown below; type the page address that contains the desired table and click Go
4. Scroll down until you find the needed information and click the little arrow on the far-left side to select it.
5. Now would be a good moment to save the query; click the corresponding button which is on the top right corner of the dialog box, and you will be prompted to choose a location. A file with the IQY extension is generated.
6. To actually transfer the web table to the worksheet, press the import button. A dialog box appears asking you where to put the data.
7. Wait a few seconds and the selected data is loaded to the worksheet.

New Web Query

Address:

Click next to the tables you want to select, then click Import.

DATE	OPPONENT	RESULT	W-L	WIN	LOSS	SAVE	ATT
Thu, 7/23	vs NYY	L 4-1 F/6	0-1	Cole 1-0	Scherzer 0-1		0
Sat, 7/25	vs NYY	W 9-2	1-1	Rainey 1-0	Paxton 0-1		0
Sun, 7/26	vs NYY	L 3-2	1-2	Green 1-0	Doolittle 0-1	Britton 1	0
Mon, 7/27	vs TOR	L 4-1	1-3	Borucki 1-0	Sanchez 0-1	Bass 1	0
Tue, 7/28	vs TOR	L 5-1	1-4	Roark 1-0	Voth 0-1		0
Wed, 7/29	@ TOR	W 4-0 F/10	2-4	Hudson 1-0	Yamaguchi 0-2		0
Thu, 7/30	@ TOR	W 6-4	3-4	Harper 1-0	Ryu 0-1	Hudson 1	0
Fri, 7/31	@ MIA				POSTPONED		

The Web Query dialog box

Note: All Excel dialog boxes that have two short lines on the bottom right corner, like the one above, can be resized. Just click that corner and drag the mouse to change its height and width.

	A	B	C	D	E	F	G	H
1								
2	DATE	OPPONENT	RESULT	W-L	WIN	LOSS	SAVE	ATT
3	Thu, Jul 23	vsNew York New York	L4-1 F/6	0-1	Cole 1-0	Scherzer 0-1		0
4	Sat, Jul 25	vsNew York New York	W9-2	01/jan	Rainey 1-0	Paxton 0-1		0
5	Sun, Jul 26	vsNew York New York	L3-2	01/fev	Green 1-0	Doolittle 0-1	Britton 1	0
6	Mon, Jul 27	vsToronto Toronto	L4-1	1/3	Borucki 1-0	Sanchez 0-1	Bass 1	0
7	Tue, Jul 28	vsToronto Toronto	L5-1	1/4	Roark 1-0	Voth 0-1		0
8	Wed, Jul 29	@Toronto Toronto	W4-0 F/10	2/4	Hudson 1-0	Yamaguchi 0-2		0
9	Thu, Jul 30	@Toronto Toronto	W6-4	3/4	Harper 1-0	Ryu 0-1	Hudson 1	0
10	Fri, Jul 31	@Miami Miami	Postponed					
11	Sat, Aug 1	@Miami Miami	Postponed					
12	Sun, Aug 2	@Miami Miami	Postponed					
13	Tue, Aug 4	vsNew York New York	W5-3	4/4	Corbin 1-0	Matz 0-2	Hudson 2	0
14	Wed, Aug 5	vsNew York New York	L3-1	4/5	Porcello 1-1	Fedde 0-1	Lugo 2	0
15	Fri, Aug 7	vsBaltimore Baltimore	L11-0	4/6	Milone 1-1	Sanchez 0-2		0

The data already in Excel.

Some data at column D were mistakenly recognized as dates, but it is easy to extract the correct information using the DAY and MONTH functions.

Note: Not all tabular data in a webpage will be interpreted as a table with this feature, so you will have to either import everything or resort to other methods such as custom VBA code.

Remember the IQY file? If you right click it and choose edit with Notepad, something like the text below is displayed; we will manipulate this file in the next section.

```

WEB
1
https://www.espn.com/mlb/team/schedule/_/name/wsh
Selection=AllTables
Formatting=None
PreFormattedTextToColumns=True
ConsecutiveDelimitersAsOne=True
SingleBlockTextImport=False
DisableDateRecognition=False
DisableRedirections=False

```

A dynamic web query

Say that you need to retrieve information from a web site multiple times, and the site address follows a pattern or explicitly accepts parameters. This technique allows to pass a variable to the query. Here is the procedure:

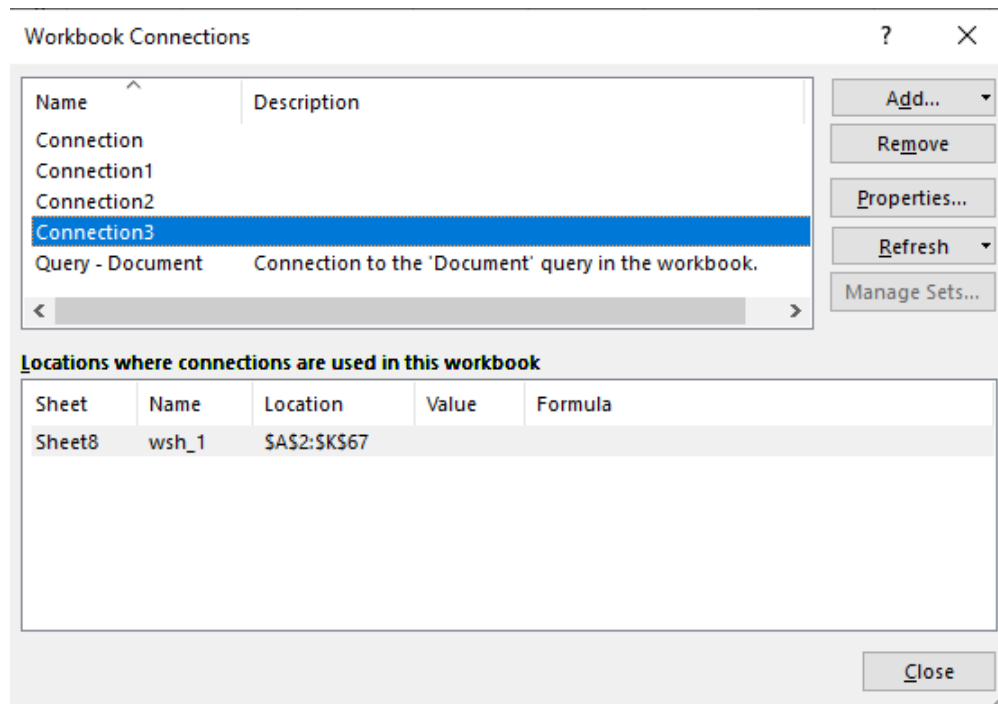
1. Modify the IQY file so it now looks like the text below.

```

WEB
1
https://www.espn.com/mlb/team/schedule/_/name/["TeamID", "Enter team ID"]
Selection=AllTables
Formatting=None
PreFormattedTextToColumns=True
ConsecutiveDelimitersAsOne=True
SingleBlockTextImport=False
DisableDateRecognition=False
DisableRedirections=False

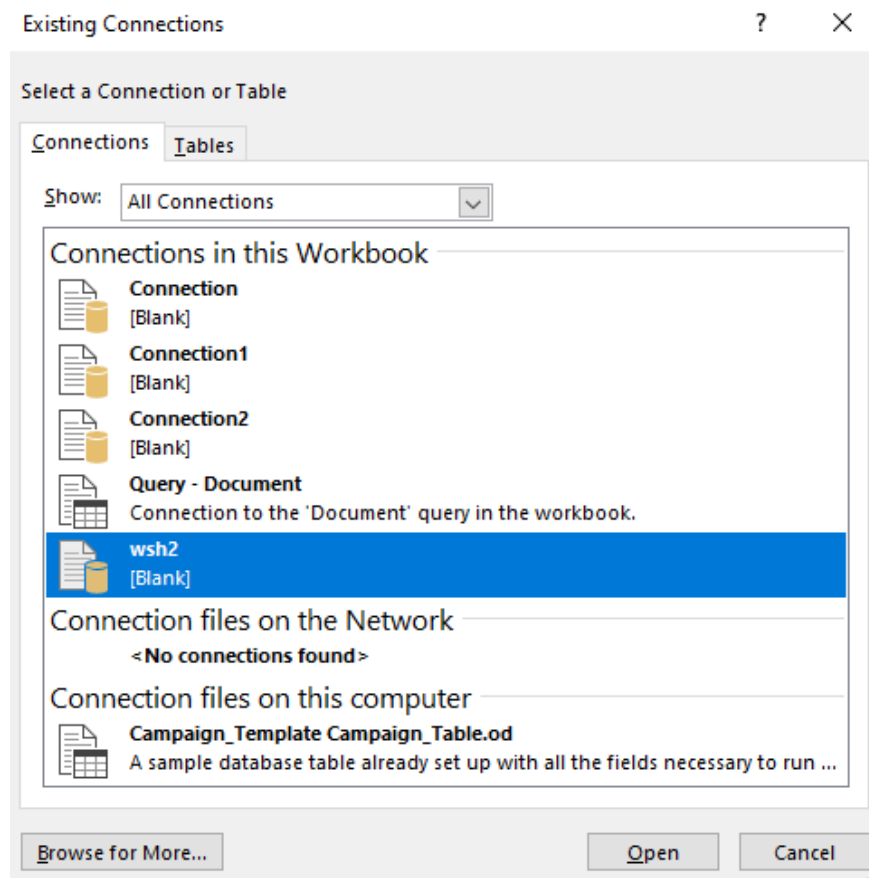
```

2. At the Data tab, click the connections option. The Workbook Connections dialog box is displayed.



The workbook connections dialog box.

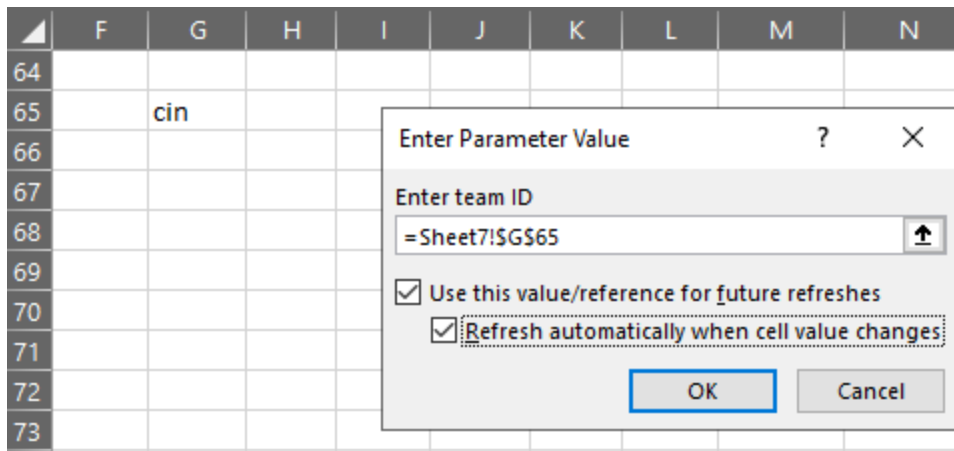
3. Select the old query that used this IQY file and click the remove button.
4. Now click the add button; the Existing Connections dialog box appears.



The existing connections dialog box.

5. Click the Browse for More button and find the modified IQY file. Press the open button and the query is added to the list. Close the dialog box.

6. At the Data tab, click Get External Data and Existing Connections.
7. Select the desired query and click Open; you will be prompted to decide where to put the data.
8. This is the cool part; a dialog box will accept a parameter of your choice. It can be typed directly, or better yet, linked to a worksheet cell. Make the selection and click OK. The tip text that is displayed inside the dialog will be the string you entered in the modified IQY file.



Linking a worksheet cell to the parameter field.

9. The data is loaded to the worksheet, as already shown before.

With this kind of setup, every time the cell value changes, the query automatically updates.

You can prepare a drop-down list on the worksheet cell with known values and this tool will be quick and simple to use.

Querying with VBA

It is also possible to write VBA code that automates this process. Below is sample code that gets data from the same web site used on the previous section. Note that the string variables can be hard coded or retrieved from the worksheet.

The resulting table is created starting at cell A4 of the active sheet.

```

Sub Query_one()
Dim part1$, part2$
part1 = "www.espn.com/mlb/team/schedule/_/name/"
part2 = "wsh"
'part1 = [b2]      ' from cell B2
'part2 = [b3]      ' from cell B3
Application.CutCopyMode = False
With ActiveSheet.QueryTables.Add(Connection:= _
    "URL;https://" & part1 & part2, Destination:=Range("$A$4"))
    .Name = "wash"
    .FieldNames = True
    .RowNumbers = False
    .FillAdjacentFormulas = False
    .PreserveFormatting = True
    .RefreshOnFileOpen = False
    .BackgroundQuery = True
    .SavePassword = False
    .SaveData = True
    .AdjustColumnWidth = True
    .RefreshPeriod = 0
    .WebSelectionType = xlAllTables
    .WebFormatting = xlWebFormattingNone
    .WebPreFormattedTextToColumns = True
    .WebSingleBlockTextImport = False
    .WebDisableDateRecognition = False
    .WebDisableRedirections = False
    .Refresh BackgroundQuery:=False
End With
End Sub

```

INTERACTING WITH SITES WITHOUT USING A BROWSER

When we talk about getting data from the Web, some people think about automating a browser such as Internet Explorer. In fact, depending of the user's specific needs, this is rather inefficient.

In this chapter other methods will be presented, for either getting or posting data over the Internet. Talking about data, let's take a step back and discuss JSON.

What is JSON?

JSON stands for JavaScript Object Notation, and is a data exchange format, just like XML.

As it is very popular these days, we will use JSON examples to demonstrate some of the techniques in this chapter.

JSON basically works with two entities, namely objects and arrays.

Here is how you would define a JSON object:

```
{
  "name": "This is my site",
  "url": "https://bananas.com"
}
```

Name and URL are called keys, with their corresponding values.

The following syntax defines an array:

```
["Nicole", "Elaine", "Samantha", "Tom"]
```

These two basic elements can be combined as shown next; the value of the Files key is an array.

```
{
  "id": 18,
  "personal": {
    "name": "Jean Ferguson",
    "Files": ["ftpA", "ftpB"]
  }
}
```

This is pretty much all you need to know to start dealing with JSON files.

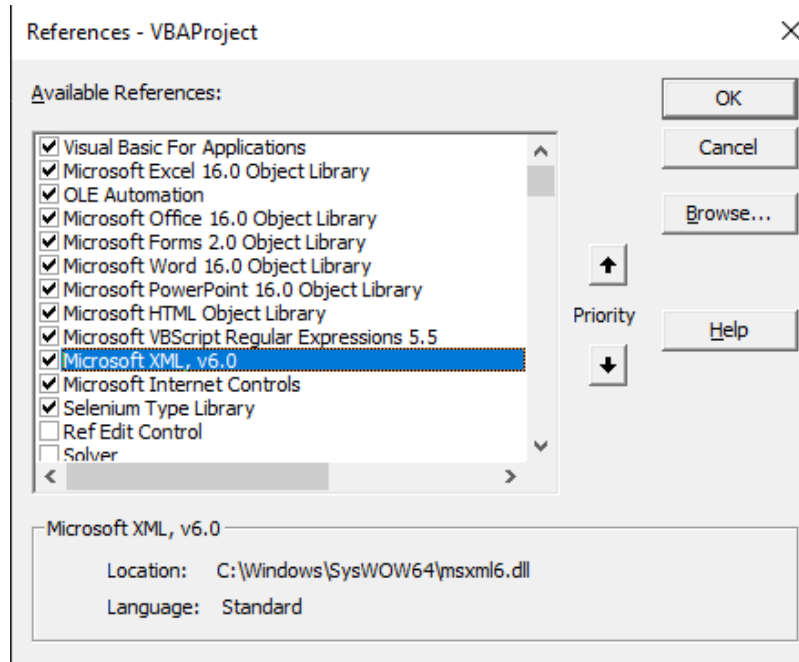
Getting data from the Web programmatically

The VBA request objects **XMLHTTP/ServerXMLHTTP** provide the necessary resources to perform communication tasks with web pages and services. To use them in your code, follow these steps:

- Go to the VBA editor
- Click Tools on the menu bar and select References
- Tick the Microsoft XML v6.0 line, as shown in the picture.

Note: When the Internet started to grow in size, the need arose to have dedicated computers to provide various services (the servers), while the others would access the services (the clients).

Common servers include Web, email and FTP services. When a browser opens a web page it is acting as a client, while the site offering information is hosted by the server. This is called the client-server architecture.



The VBA references dialog box.

In the following example we will retrieve some JSON data from the web site shown below; it provides fake data for testing purposes.



This is the Web data we want...

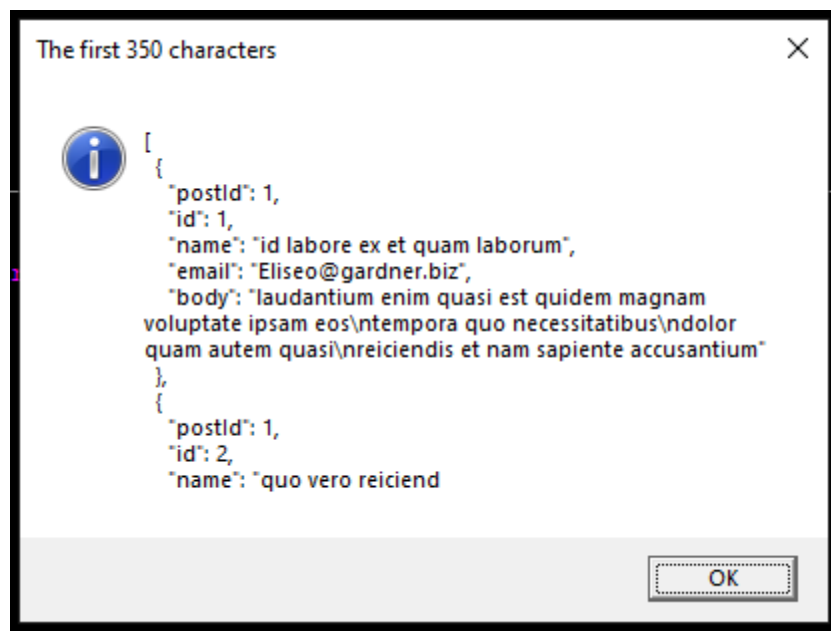
Unfortunately, there is no built-in support for JSON in Excel. This initial code version will simply dump it to a string variable:

```

Public Sub Direct()
Dim http As Object, resp$
Set http = CreateObject("MSXML2.XMLHTTP")
http.Open "GET", "http://jsonplaceholder.typicode.com/" _
& "posts/1/comments", False
http.send
' get the response
resp = http.responseText
MsgBox Left(resp, 350), 64, "The first 350 characters"
End Sub

```

As you can see below, the contents are there, but extracting the desired information from the variable will require clumsy string manipulation.

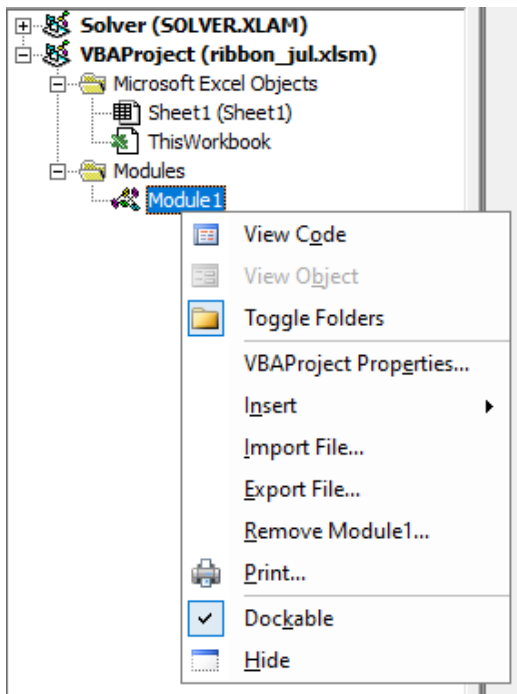


The output from the direct method

The good news is that there are several free JSON converters online, which will provide a structured way to process the data.

One of them is called VBA-JSON; to take advantage of it, follow this procedure:

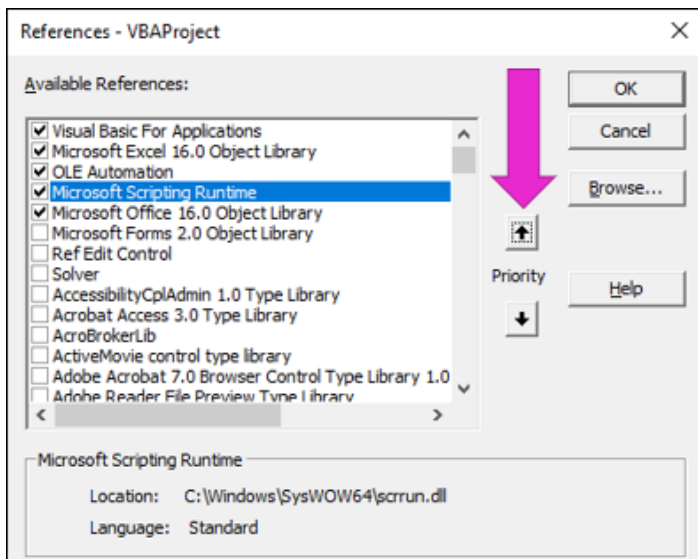
1. Go to the home page <https://github.com/VBA-tools/VBA-JSON> and click the code button to download a ZIP file.
2. Extract the contents to your computer.
3. Open the Excel workbook where you want to use it and go to the VBA editor.
4. Right click any module on the left pane and choose the import file option.



Importing a BAS file into your project.

5. In the dialog box that appears, find and select the JsonConverter BAS file from the zipped package.
6. Click open and a new module is added to your VBA project. The source code is not protected so you can study it.
7. Add a reference to the Microsoft Scripting Runtime library.

Caution: You may receive errors if this reference has a low priority in the list. To be on the safe side, click the up arrow on the references dialog box to increase its priority, as depicted on the figure.



Changing the priority of a VBA reference.

Now we can run the code below, quickly transferring the key/value pairs from the Web site to the worksheet:

```
Public Sub Use_Library()
Dim http As Object, json As Object, i%, item, _
head, dest As Range
```

```

Set http = CreateObject("MSXML2.XMLHTTP")
' create array with keys
head = Array("id", "name", "postId", "email", "body")
Set dest = [a1].Resize(1, UBound(head) + 1)
dest.Value = head      ' first row
http.Open "GET", "http://jsonplaceholder.typicode.com/" _
& "posts/1/comments", False
http.send
Set json = ParseJson(http.responseText)
i = 2
For Each item In json
    Sheets(2).Cells(i, 1) = item(head(0))
    Sheets(2).Cells(i, 2) = item(head(1))
    Sheets(2).Cells(i, 3) = item(head(2))
    Sheets(2).Cells(i, 4) = item(head(3))
    Sheets(2).Cells(i, 5) = item(head(4))
    i = i + 1      ' next row
Next
End Sub

```

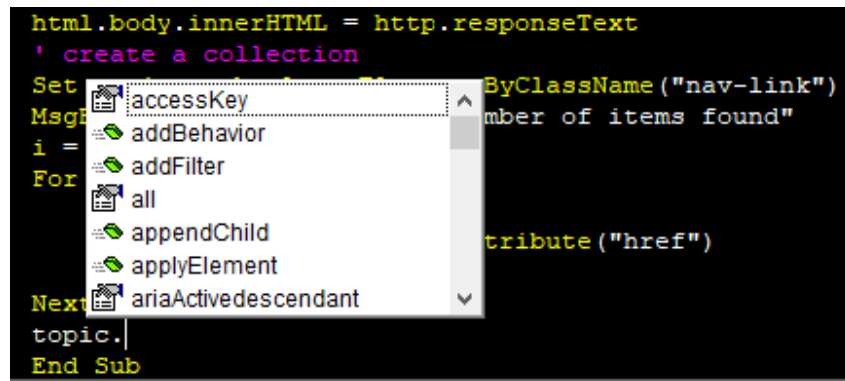
	A	B	C	D	E
1	id	name	postId	email	body
2	1	id labore ex et quam laborum	1	Eliseo@gardner.biz	laudantium enim quasi est quidem magnam voluptate ipsam eos tempora quo necessitatibus dolor quam autem quasi reiciendis et nam sapiente accusantium
3	2	quo vero reiciendis velit similique earum	1	Jayne_Kuhic@sydney.com	est natus enim nihil est dolore omnis voluptatem numquam et omnis occaecati quod ullam at voluptatem error expedita pariatur nihil sint nostrum voluptatem reiciendis et
4	3	odio adipisci rerum aut animi	1	Nikita@garfield.biz	quia molestiae reprehenderit quasi aspernatur aut expedita occaecati aliquam eveniet laudantium omnis quibusdam delectus saepe quia accusamus maiores non et atque
5	4	alias odio sit	1	Lew@alysha.tv	occaecati deserunt quas accusantium unde odit nobis qui voluptatem
6	5	vero eaque aliquid doloribus et	1	Hayden@althea.biz	harum non quasi et ratione tempore iure ex voluptates in ratione harum architecto fugit inventore cupiditate

This Excel table originated from a JSON format.

Getting data from Web page elements

On the previous chapter we saw HTML code from a web page that contained a class named “nav-link”. The following code will find all elements that belongs to this class and extract the string representing the href attribute, which specifies the destination address for that link. All addresses are then transferred to the worksheet.

Note that by declaring the topic variable as an HTML element, the VBA editor will offer the available properties and methods for that type; this list would not appear if the variable was declared as a generic object type. This feature is known as Intellisense and besides showing the applicable properties and methods, also serves to perform auto completion reducing your typing.



Intellisense in action.

```
Public Sub Get_Class()
Dim http As Object, html As New HTMLDocument, _
topics As Object, topic As HTMLHtmlElement, i%
Set http = CreateObject("MSXML2.XMLHTTP")
http.Open "GET", "https://www.mrexcel.com/", False
http.send
html.body.innerHTML = http.responseText
' create a collection
Set topics = html.getElementsByTagName("nav-link")
MsgBox topics.Length, 64, "Number of items found"
i = 2
For Each topic In topics
    ' write to worksheet
    Cells(i, 1) = topic.getAttribute("href")
    i = i + 1
Next
End Sub
```

	A
1	
2	https://www.mrexcel.com/board/
3	https://www.mrexcel.com/speaking/
4	https://www.mrexcel.com/store/
5	https://www.mrexcel.com/news/
6	https://www.mrexcel.com/excel-tips/
7	about:blank#qb-dd-mrexcel-resources
8	https://www.mrexcel.com/products/
9	https://www.mrexcel.com/excel-mall/
10	https://www.mrexcel.com/tech-tv/
11	https://www.mrexcel.com/challenges/
12	https://www.mrexcel.com/learn-excel-resources/
13	https://www.mrexcel.com/archive/
14	https://www.mrexcel.com/consulting-services/
15	https://www.mrexcel.com/search/
16	about:blank#qb-dd-about
17	https://www.mrexcel.com/about-mrexcel/

A partial view of the scraped data.

Posting data over the Internet

The majority of Web code examples you will find are for getting data from some server or site, but there are times when we need to send data, for example to tell the trading application how many shares you want to buy. In most cases these are going to be *authenticated* requests, which are covered on the next section.

The code below sends some JSON data from cell D1 to a testing endpoint; the figure displays the response text. Some relevant points are:

- The string at cell D1 is the JSON representation of range A1:C3. It begins and ends with square brackets; hence it is an array.
- The indexes "0", "1" and the ID key with the value 101 were created at the response text.
- The server XML object used in this example may request or send data.

```
Sub Post_example()
Dim HTTP As Object, json$, URL$
' get the JSON string from worksheet
json = Range("d1")
' use server object
Set HTTP = CreateObject("MSXML2.ServerXMLHTTP")
URL = "https://jsonplaceholder.typicode.com/posts"
HTTP.Open "POST", URL, False
HTTP.setRequestHeader "Content-type", "application/json"
HTTP.send (json)
MsgBox HTTP.responseText, 64, "This is the result:"
Set HTTP = Nothing
End Sub
```

	A	B	C	D
1	X	Y	Z	[{"X":"A2","Y":"B2","Z":"C2"}, {"X":"A3","Y":"B3","Z":"3 3"}]
2	A2	B2	C2	
3	A3	B3	3 3	
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				

This is the result: X



```
{
  "0": {
    "X": "A2",
    "Y": "B2",
    "Z": "C2"
  },
  "1": {
    "X": "A3",
    "Y": "B3",
    "Z": "3 3"
  },
  "id": 101
}
```

OK

The message box shows the posted data

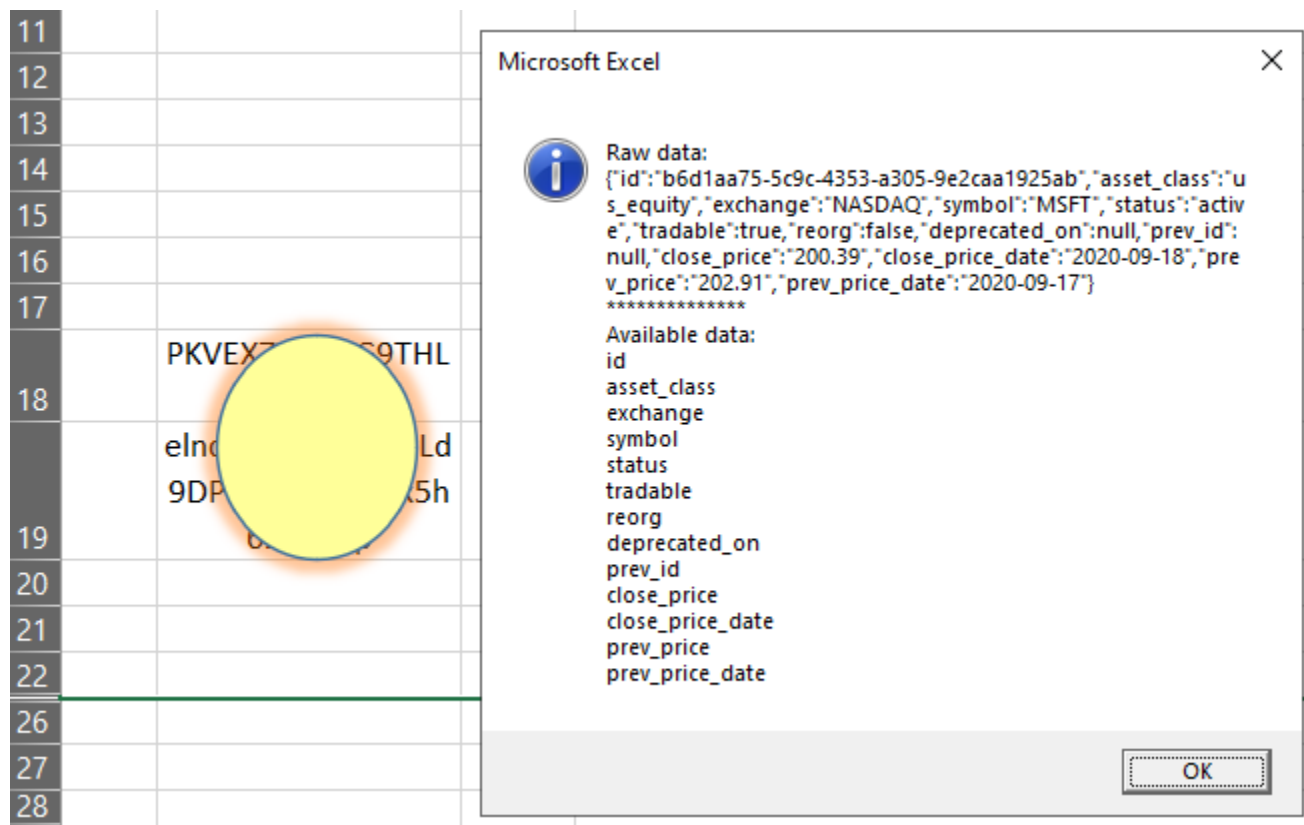
Authenticated requests

The previous sections showed that it is perfectly possible to get web data from public sites. However, many applications will require that you login to the service, a good example being financial operations. Obviously, without the user's credentials the application will not know to which account the request belongs.

The sample code that follows uses a paper account at the Alpaca website. This kind of account provides a way to perform fake operations for testing purposes, but they also have the live part where you can risk real money...

Note: A Web API (Application Programming Interface) allows the user to communicate with an Internet server, retrieving information from its database or sending specific data via POST requests. The destination addresses are often referred to as endpoints, and responses typically come in JSON or XML formats.

```
Sub authenticating()
Dim req As MSXML2.ServerXMLHTTP60, key_id, _
secretk$, secreth$, URL$, json As Object, item, s$
' create object
Set req = New MSXML2.ServerXMLHTTP60
key_id = CStr([b18])          ' id from cell B18
secretk = CStr([b19])         ' secret key from cell B19
' get Microsoft information
URL = "https://paper-api.alpaca.markets/v1/assets/MSFT"
req.Open "GET", URL, False
req.setRequestHeader "APCA-API-KEY-ID", key_id
req.setRequestHeader "APCA-API-SECRET-KEY", secretk
req.send
' start building string
s = "Raw data:" & vbCrLf & req.responseText
s = s & vbCrLf & "*****" & vbCrLf & _
"Available data:" & vbCrLf
' use the library
Set json = ParseJson(req.responseText)
For Each item In json        ' loop the collection
    s = s & item & vbCrLf
Next
MsgBox s, 64
End Sub
```



The two required keys on the worksheet and the retrieved data

To get the desired data you need to inform an ID and also a secret key, which can be stored in worksheet cells as shown in the picture above. Data is retrieved in JSON format; the message box shows it, and then uses the previously discussed library to produce a list of the available keys.

INTERNET EXPLORER AND VBA

At the present time, Edge is the current Microsoft browser of choice, and we will discuss it later.

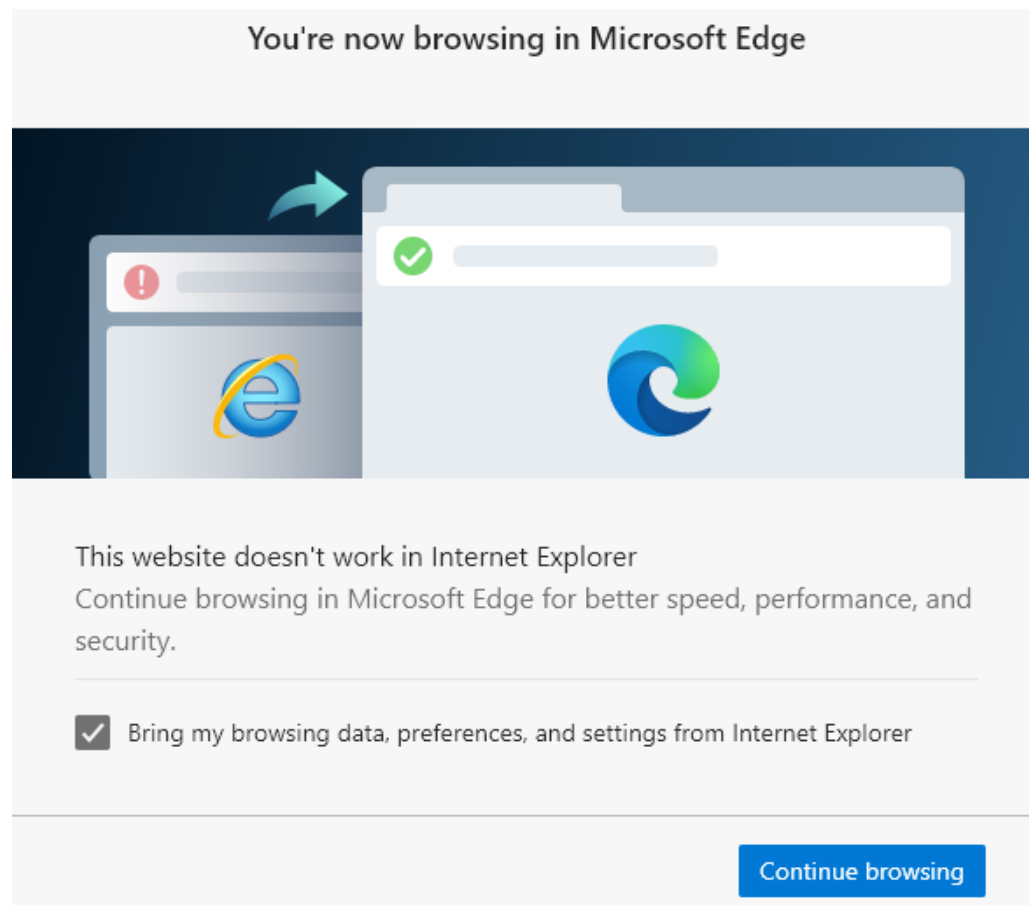
However, Internet Explorer is the only one with built in VBA support. As it is still around, it is worth to spend a few pages talking about it.

The first thing to do is to add a VBA reference to the Microsoft Internet Controls library, as already explained on the previous chapter. After doing that you are ready to try the initial example below.

Note: If you are running Windows 10, it may be tricky to find the IE icon to launch it manually. Here are some options:

- Click Start > All Apps and look under Windows Accessories
- Use the Search box on the Taskbar, just type Internet Explorer.
- Use the Run command; Press Windows key + R and type iexplore.exe

Caution: Some sites such as YouTube will not open in Internet Explorer anymore; trying to do so gives you the message below and Edge automatically takes over. Therefore, it is a good idea to manually open the site you plan to automate with IE to avoid unpleasant surprises.



Trying to open YouTube with Internet Explorer

Transferring a Web table to the worksheet

Consider the Web site shown below, along with part of its source HTML code; the task is to transfer the table to an Excel sheet, complete with the country flags.

rates.currencywholesale.com/rates/rates.html

Currency Exchange
FX CONNECTORS LTD.

Rates updated on 9/29/20 at 09:04:47 PST

	Country	Code	Currency	We Buy at	We Sell at
	United States	USD	Dollars	1.2985	1.3816
	Euro Members	EUR	Euro	1.52448	1.61878
	United	GBP	Pounds	1.67115	1.77796

This table will be copied into Excel.

Looking at the HTML code below, you will notice TR, TD and TH tags.

- The <tr> tag defines a row in an HTML table. A <tr> element contains one or more <th> or <td> elements.
- An HTML table has two kinds of cells; header cells created with the <th> element, and data cells created with the <td> element.

Also observe that there are elements defined with the IMG tag and containing the src attribute; these are used to hold the country flags.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>...</head>
  <body>
    <table width="600" align="center">
      <tbody>
        <tr>...</tr>
        <tr valign="top">
          <td height="560">
            <br>
            <table id="currency_table">
              <caption class="time_update">Rates updated on 10/10/20 at 09:04:52 PST</caption>
              <thead>
                <tr id="currency_widget_table_header">
                  <th class="flag">&nbsp;</th>
                  <th class="country">Country</th>
                  <th class="code">Code</th>
                  <th class="cur">Currency</th>
                  <th class="buy">We Buy at</th>
                  <th class="sell">We Sell at</th>
                </tr>
              </thead>
              <tbody class="main_tbody">
                <tr class="row_currencyUp row_even">...</tr>
                <tr class="row_currencyUp row_even">
                  <td class="displayColFlag">
                    
                  </td>
                  <td class="displayColCountry">Euro Members</td>
                  <td class="displayColCode">EUR</td>
                  <td class="displayColCurrency">Euro</td>
                  <td class="displayColWeBuy">1.50522</td>
                  <td class="displayColWeSell">1.59832</td>
                </tr>
                <tr class="row_currencyUp row_even">...</tr>
                <tr class="row_currencyUp row_even">...</tr>
              </tbody>
            </table>
          </td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

A partial view of the Web page's HTML code.

Here is detail of the HTML to generate a row in the table and the code to retrieve each row.

```
▼ <tbody class="main_tbody"> == $0
  ▶ <tr class="row_currencyUp row_even">...</tr>
  ▼ <tr class="row_currencyUp row_even">
    ▼ <td class="displayColFlag">
      
    </td>
    <td class="displayColCountry">Euro Members</td>
    <td class="displayColCode">EUR</td>
    <td class="displayColCurrency">Euro</td>
    <td class="displayColWeBuy">1.50522</td>
    <td class="displayColWeSell">1.59832</td>
  </tr>
  ▶ <tr class="row_currencyUp row_even">...</tr>
  ▶ <tr class="row_currencyUp row_even">...</tr>
```

A portion of the HTML to generate a row in the table.

```
Sub Web_Table()
Dim tob As Object, ws As Worksheet, im As shape, rowc%, _
colv%, objie As InternetExplorer, bod As HTMLBody, ic, s$
s = "https://rates.currencywholesale.com/rates/"
Set ws = Sheets("sheet1")
Set objie = New InternetExplorer
objie.Visible = True
objie.navigate s & "rates.html"
' wait for page loading
Application.Wait (Now + TimeValue("0:00:05"))
Set bod = objie.Document.body
Set ic = bod.getElementsByTagName("img")
' get all tables
Set tob = bod.getElementsByTagName("table")
On Error Resume Next ' if a flag is missing
For rowc = 0 To tob(1).Rows.Length - 1
  Set im = ws.Shapes.AddPicture(s & _
  ic(rowc + 1).getAttribute("src"), msoFalse, msoTrue, _
  50, 50, 50, 50)
  For colv = 0 To tob(1).Rows(rowc).Cells.Length - 1
    ws.Cells(rowc + 1, colv + 1) = _
    tob(1).Rows(rowc).Cells(colv).innerText
    With im
      .LockAspectRatio = 0
      .Top = ws.Cells(rowc + 2, 1).Top
      .Left = ws.Cells(rowc + 2, 1).Left
      .Width = ws.Cells(rowc + 2, 1).Width
      .Height = ws.Cells(rowc + 2, 1).Height
    End With
  Next
Next
On Error GoTo 0
End Sub
```

Here are the key steps of the above code:

1. Navigates to the desired page.
2. Creates a collection of all objects with the IMG tag.

3. Creates a collection of all page tables; we will be interested in the second table of this group. As the collection starts with the #0 index, we refer to the #1 index.
4. Performs a table row loop, transferring each country's information to separate worksheet rows. There is an inner loop which iterates the table columns.

Note: A common task when interacting with a web page is to locate certain elements in the HTML structure. Here are the three most popular methods:

- `GetElementbyID` – finds a single element which has a unique ID. Unfortunately, not all page elements will have such ID.
- `GetElementsbyClassName` – creates a collection of elements which belong to a particular class.
- `GetElementsbyTagName` – creates a collection of elements with a particular tag.

	A	B	C	D	E	F
1		Country	Code	Currency	We Buy at	We Sell at
2		United States	USD	Dollars	1,2985	1,3816
3		Euro Members	EUR	Euro	1,5245	1,6188
4		United Kingdom	GBP	Pounds	1,6712	1,7780
5		Mexico	MXN	Pesos	0,0552	0,0631
6		Japan	JPY	Yen	0,0116	0,0131
7		Australia	AUD	Dollars	0,9056	0,9914
8		China	CNY	Yuan Renminbi	0,1771	0,2125
9		New Zealand	NZD	Dollars	0,8295	0,9195
10		Brazil	BRL	Brazil Real	0,2130	0,2556

The resulting worksheet table.

Working with events

Some Web sites are programmed to respond to events generated by the user; this allows for a more interactive and dynamic page. The following example uses a local HTML file to show the basics of such behavior.

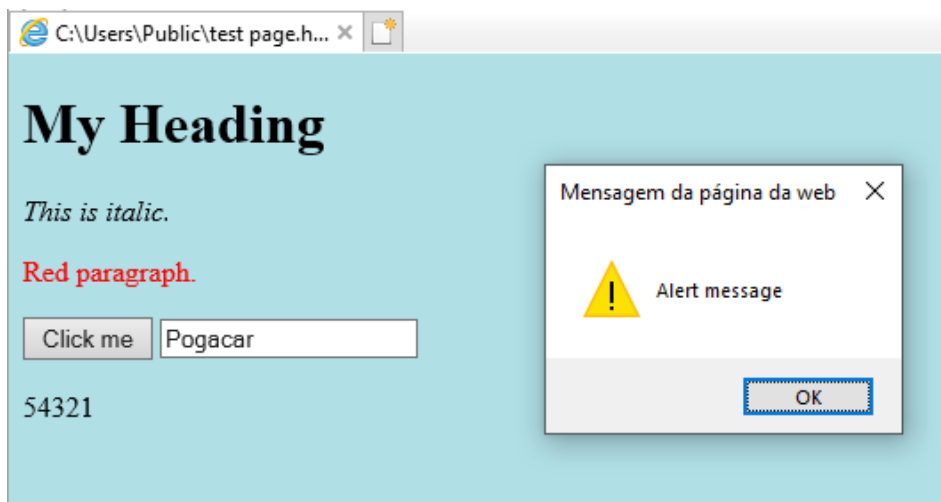
Start by typing this HTML source code, as already explained on the first chapter:

```
<!DOCTYPE html>
<html>
<body style="background-color:powderblue;">
<h1>My Heading</h1>
<p><i>This is italic.</i></p>
<p style="color:red;">Red paragraph.</p>
<button>Click me</button>
<input name="q" onchange="alert('Alert message');" /input>
<bdo dir="rtl"><br><br>12345<br><br></bdo>
</body>
</html>
```

Note that an on-change event is defined, meaning it will fire when the input box value changes. Save it to your hard drive and run the code below:

```
Sub IE_event()
Dim iea As InternetExplorer, doc, search, _
event_onChange As Object
Set iea = CreateObject("InternetExplorer.Application")
iea.Visible = True
iea.navigate "C:\Users\public\test page.htm"
Do Until Not iea.Busy
    DoEvents
Loop
Application.Wait Now + TimeValue("0:00:03")
Set doc = iea.Document
Set search = doc.getElementsByName("q")(0)
search.Value = "Pogacar"
'Exit Sub 'exiting here does not display the message
Set event_onChange = iea.Document.createEvent("HTMLEvents")
event_onChange.initEvent "change", True, False
search.dispatchEvent event_onChange
'iea.Quit 'close IE window
Set iea = Nothing
End Sub
```

The macro opens the local file using Internet Explorer, defines the search variable as the input box and writes to it. After that, the code dispatches the on-change event, making the alert message to appear, as shown below.



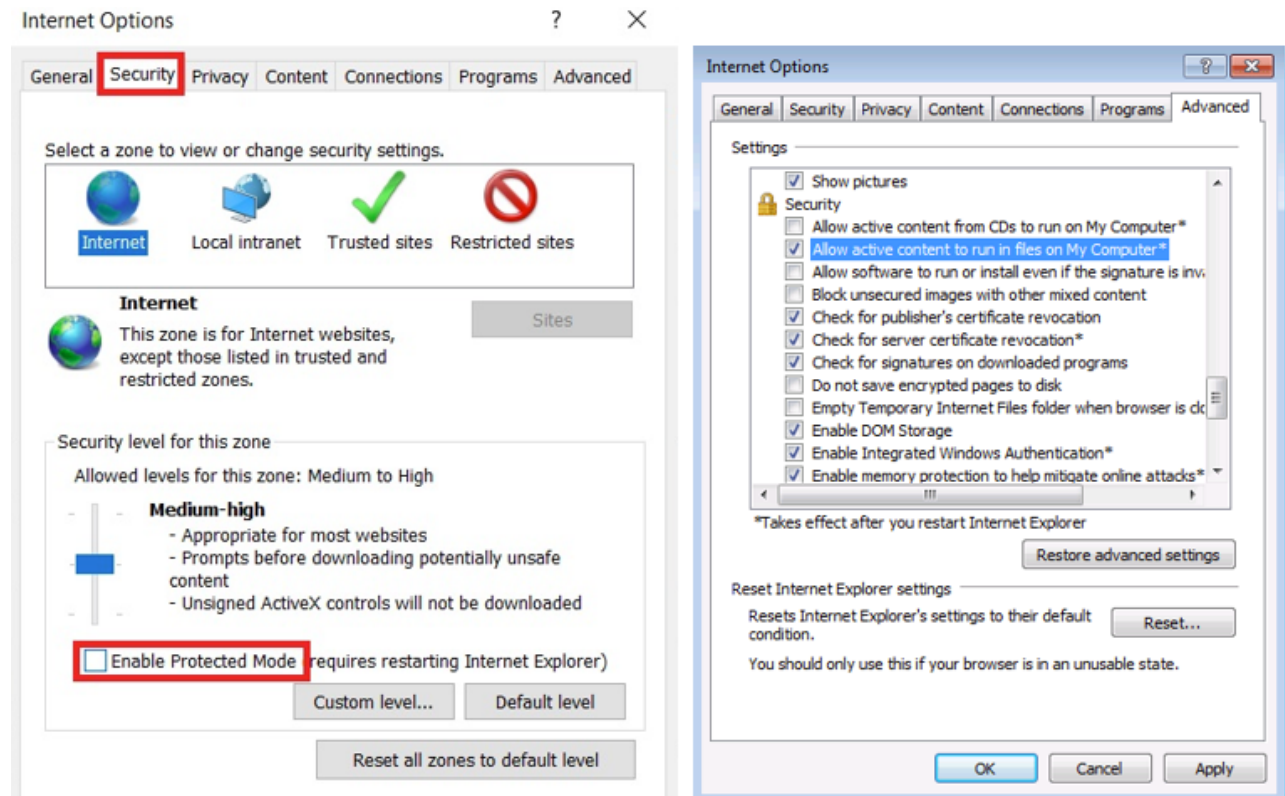
This page message was triggered by VBA code

Caution: You may get errors when automating IE with VBA, depending on the security settings that are in effect. If you trust the websites you are working with, consider changing the two options displayed on the picture below.

Go to IE tools>Internet options>Security and uncheck the Enable Protected Mode option.

Secondly, at the advanced tab, check the option to allow active content to run.

If you are still using IE to browse the Web or downloading files for local use, it is safer to restore these settings to the default values after the VBA tasks are completed.



Changing Internet Explorer security options.

INTRODUCING SELENIUM

Selenium is an open-source tool used to automate testing of Web applications. It can quickly perform dozens of functional and compatibility tests, interacting the beta scripts with browsers. When developing complex, commercial software the testing phase is essential to minimize bugs when the product is deployed for multiple platforms.

A nice guy named Florent Breheret developed Selenium Basic, which is a framework that allows direct Selenium use by Excel; so, these are two different things:

- Selenium - A browser automation framework and ecosystem originally created by Jason Huggins.
- Selenium Basic - A Selenium based browser automation framework for VB.Net, VBA and VBScript.

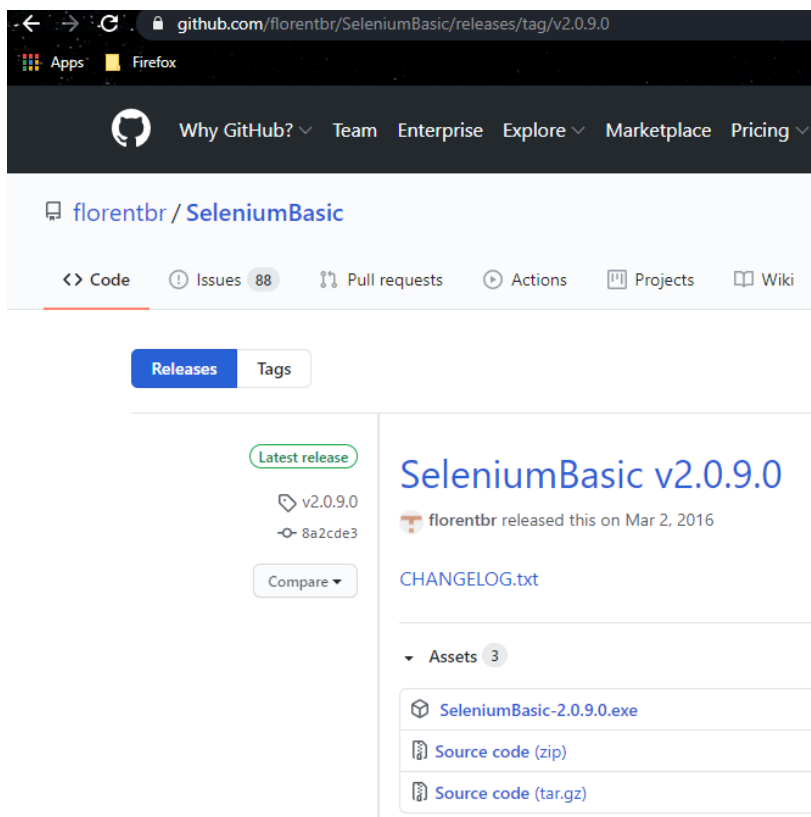
For the sake of simplicity, I will refer to SB simply as Selenium from now on.

Selenium provides a simple interface that allows your spreadsheet to automate other browsers besides Internet Explorer, such as the more stable Chrome. The next section will show how to set up Selenium on your system.

Installing Selenium

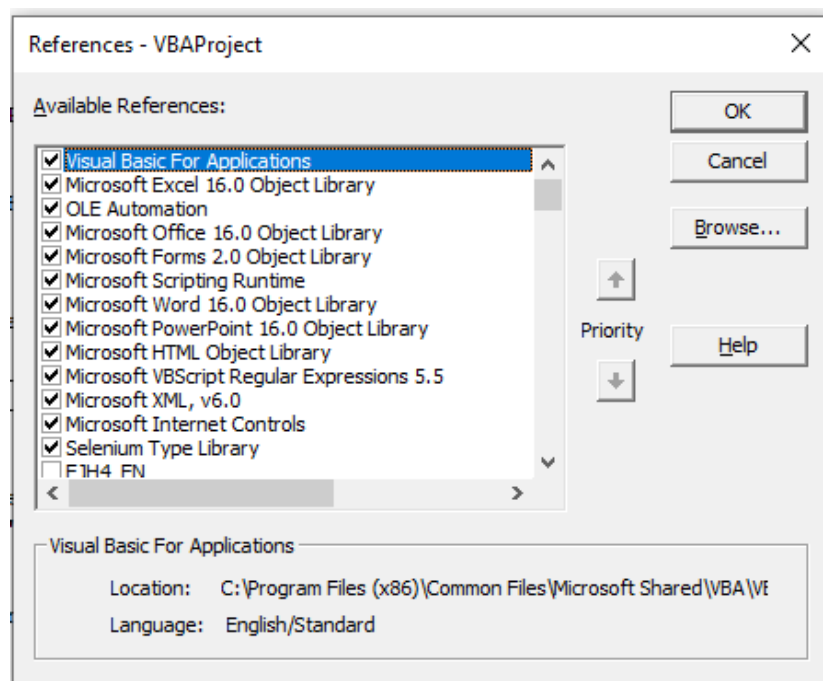
Here are the steps to install Selenium on your computer:

1. Download and install the executable file from the GitHub page below



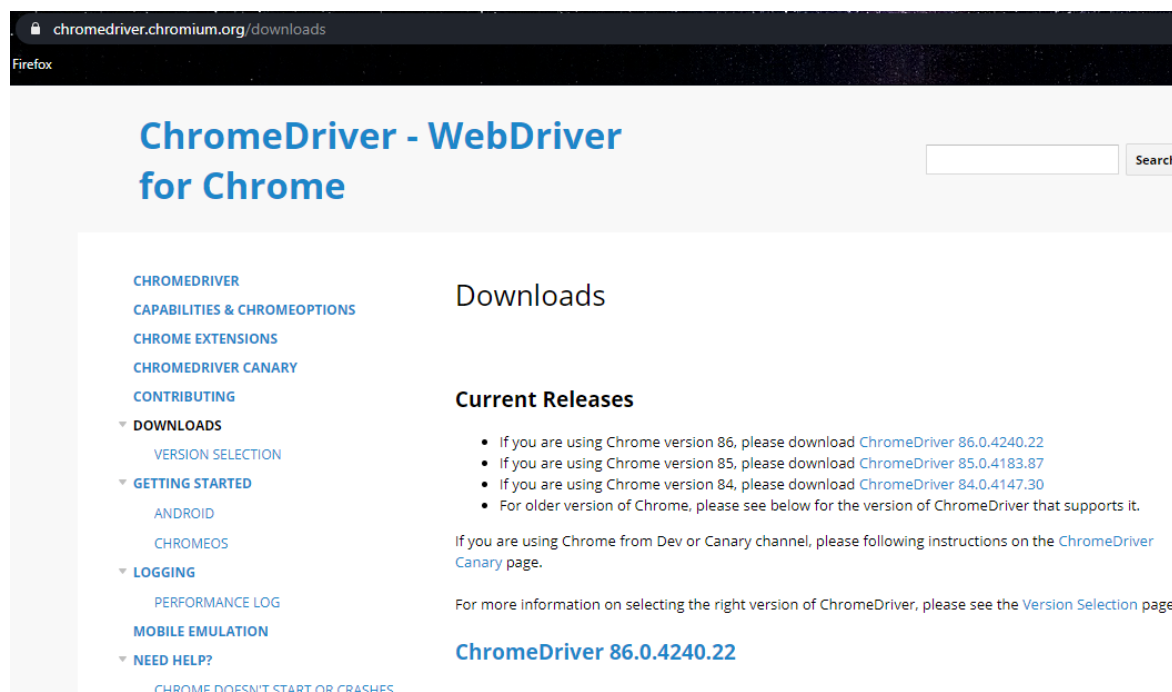
Get the free package from this page

2. At the VBE, check the reference for the Selenium type library



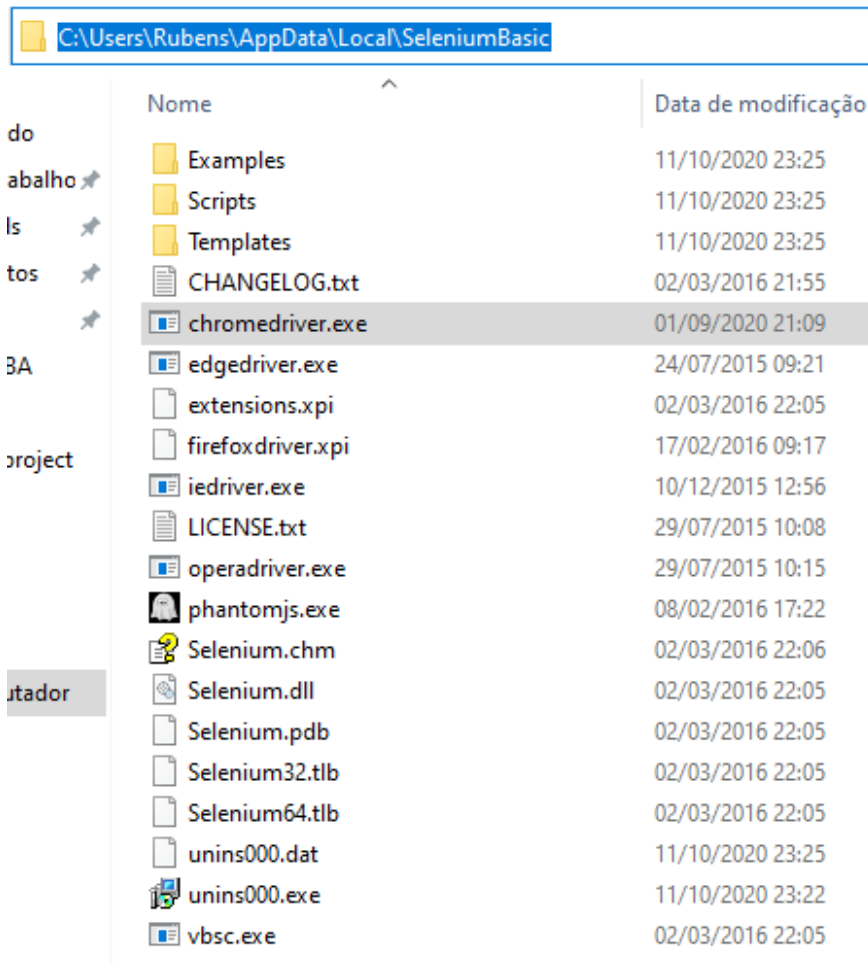
Adding the VBA Selenium reference.

3. It is likely that you will need to update the Chrome driver originally installed at the first step. Go to chromedriver.chromium.org/downloads and get the driver compatible with your current Chrome version.



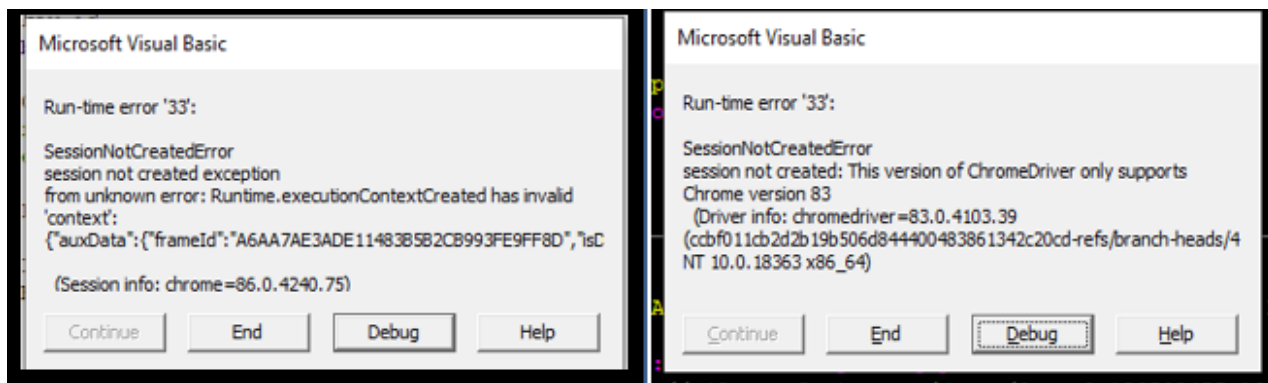
Download the latest Chrome driver.

4. Replace the installed driver at the Selenium Basic folder with the one you just downloaded.



Substitute the old driver by the correct one.

Caution: Chrome has a fast development cycle and eventually the driver you are using will become incompatible with the new browser version. When that happens, you will get errors like the ones shown below. To make things work again, repeat the manual driver update described above.



These are driver version related errors.

Internet Explorer and Selenium

In most cases, we will use the Selenium tool to automate modern browsers such as Chrome or Edge. However, you can also interface the old IE browser as well; the example below shows how.

```
Public bot As New IEDriver
Sub Test()
With bot
    .get "https://www.mrexcel.com"
    .Window.Maximize
    '.Quit
End With
End Sub
```

Just refer to the installed Internet Explorer driver and everything will work fine.

GOOGLE CHROME AND VBA

Now that Selenium is installed on your system, it is possible to automate the Chrome browser directly from VBA code. Selenium has its own tree of properties and methods, but it will work in the same fashion as Internet Explorer is automated with VBA.

Running JavaScript on a page

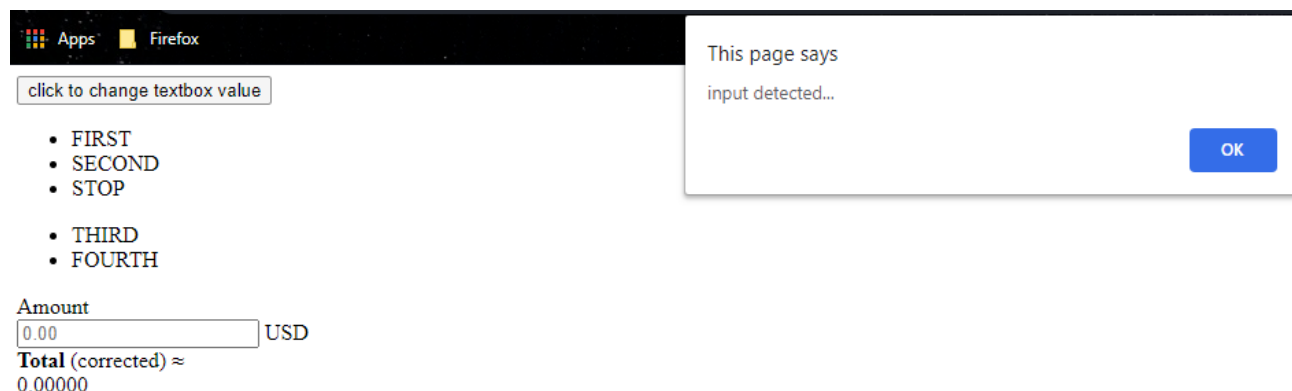
For this first example, let's create another local HTML file, with source code that looks a lot like a real Web page. It contains several elements and also embedded JavaScript code, wrapped inside `<script>` tags; type the following on Notepad, as already discussed earlier:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
</head>
<body>
  <input id="B1" type="button" value="click to change textbox value"
    onclick="TAlert()" />
  <form class="OF_form">
    <ul class="OF_t2">
      <li class="OF_3T OF_active">FIRST</li>
      <li class="OF_3T">SECOND</li>
      <li class="OF_3T">STOP</li>
    </ul>
    <ul class="OF_toggle">
      <li class="OF_ttab_ OF_buy OF_active">THIRD</li>
      <li class="OF_ttab OF_sell">FOURTH</li>
    </ul>
    <div class="MOr">
      <div class="OF_section">
        <div class="OF_section-h">Amount</div>
        <div class="OF_input-box_Xk">
          <input type="number" step="0.1" min="0" name="amount"
            placeholder="0.00" value=""
            autocomplete="off" oninput="OIHandler()">
          <span>USD</span>
        </div>
      </div>
    </div>
    <div class="OF_OT">
      <div>
        <b>Total</b>
        <span>(corrected)</span>
        <b>~</b>
      </div>
      <div class="OF_total" >0.00000</div>
    </div>
  </form>
  <script language="javascript">
    function OIHandler() {
      print_call_stack(); alert('input detected...');
    }
    function print_call_stack() { console.trace(); }
    function TAlert() { setIB(document); }
    function setIB() {
      try {
        var inputBox = document.querySelector('div.OF_input-box_Xk input');
        inputBox.value = 50;
      }
      if (document.createEvent) { var ev2 = document.createEvent("HTMLEvents");
        ev2.initEvent("input", true, true); ev2.eventName = "input";
        inputBox.dispatchEvent(ev2); }
```

```

        return ({ success: true });    }
        catch (ex) {
            return ({ exception: ex, myMsg: '#error in setIB!' });
        }
    }
}
</script>
</body>
</html>

```



A local page with a message box triggered by the input event

When saved with a HTM extension, it will produce the layout shown above. The main features it possesses are:

- Clicking the button at the top changes the contents of the input box to an arbitrary value.
- Clicking the up and down arrows that appear when you hover the mouse over the input box changes its value accordingly.
- Input actions make the message box displayed on the picture appear.

The VBA that follows executes the setIB function contained in the source HTML code; it causes the input warning to pop up. Click the OK button to see the input box value change.

```

Public d As WebDriver
Public Sub Exec()
Dim url$
    url = Environ("USERPROFILE") & _
"\Documents\ENS\Excel & VBA\Web STP\page_w_JS.htm"
Set d = New ChromeDriver
With d
    .Start "Chrome"
    .get url
    .Wait 1000
    .ExecuteScript ("setIB()")
End With
End Sub

```

Note: If you want the loaded Web page to remain open after the code finishes execution, declare the Web driver variable outside the subroutine. If declared inside it, memory is released when the code ends and the page closes automatically.

Iframes and transferring tables without a loop

Iframes are HTML inline frame elements and can be used to incorporate content from other pages into the current page.

This is great, but when dealing with such layout you must remember this fact or else your code will not work. We need to programmatically enter the frame to access its elements. Say that our goal is to transfer the Web table below to a worksheet. Observe the HTML code to conclude that the table is located inside a frame, which in turn is nested in a parent frame. Hence, the code has to enter two frames in order to fetch the desired table.

Remember the Internet Explorer chapter where we painfully looped through rows and columns to get the table? Selenium has a cool method that will copy it in one swift pass!



The latest stock merger arbitrage spreads:

Mergers 20201016.xlsx : Active								
Target	Ticker	Acquirer	Ticker	Current Price	Take Over Price	Price Diff	% Diff	Date Announced
Acacia Communications Inc Com	ACIA	Cisco Systems Inc Com	CSCO	\$67.49	\$70.00	\$2.51	3.72%	7/9/2019
Advanced Disposal Services Inc Com	ADSW	Waste Management Inc Com	WM	\$30.27	\$30.30	\$0.03	0.10%	4/15/2019
Amag Pharmaceuticals Inc Com	AMAG	Covis Group	--	\$13.63	\$13.75	\$0.12	0.88%	10/1/2020
American Renal Holdings Inc Com	ARA	Nautic Partners, LLC	--	\$11.48	\$11.50	\$0.02	0.17%	10/2/2020
Acorn International Inc Adr	ATV	Management	--	\$20.40	\$21.00	\$0.60	2.94%	10/12/2020
Avenue Therapeutics Inc Com	ATXI	Cipla Limited	--	\$3.84	\$13.92	\$10.08	262.50%	11/13/2018
Bay Banks Virginia Inc Com	BAYK	Blue Ridge Bankshares Inc (Va) Com	BRBS	\$6.75	\$6.87	\$0.12	1.78%	8/13/2020
Bitauto Hldgs Ltd Adr	BITA	Yiche Holding Limited	--	\$15.84	\$16.00	\$0.16	1.01%	6/12/2020
Bmc Stock Holdings Inc Com	BMCH	Builders Firstsource Inc Com	BLDR	\$43.60	\$43.97	\$0.37	0.85%	8/27/2020
Bankia S.A. Com	BNKXF	Caixabank Sa Adr Un-sponsored	CAIXY	\$1.27	\$1.35	\$0.08	6.40%	9/18/2020
Cincinnati Bell Inc Com	CBB	Macquarie Infrastructure Partners	--	\$15.13	\$15.50	\$0.37	2.45%	12/23/2019

Partial view of the desired table



The source HTML code

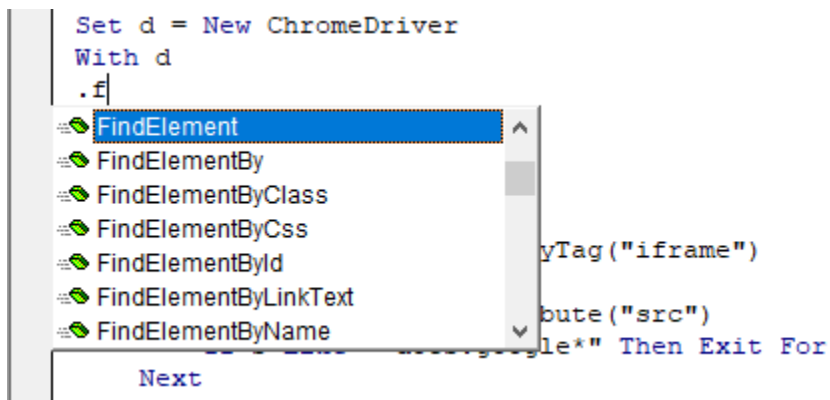
```
Public d As WebDriver
Public Sub Table_no_loop()
Dim col As Object, i%, col2 As Object, s$, col3 As Object
Const url As String = "http://intrinsicedge.blogspot.com/"
Set d = New ChromeDriver
With d
.Start "Chrome"
.get url
.Wait 800
Set col = d.FindElementsByTag("iframe")
For i = 1 To col.Count
s = col.item(i).Attribute("src")
If s Like "*docs.google*" Then Exit For
Next
.Wait 1000
.SwitchToFrame col.item(3)
Set col2 = .FindElementsByTag("iframe")
.SwitchToFrame col2.item(1)
Set col3 = .FindElementsByTag("table")
.Wait 500
End With
col3.item(1).AsTable.ToExcel _
ThisWorkbook.Worksheets("table").[a1]
End Sub
```

The code starts by loading the page and creating a collection of all Iframes present. It then finds the desired one taking advantage of the SRC attribute. After that it enters the first frame, locates the child frame and switches to it. Finally, the table is identified and copied with a single statement; the picture below shows the final result.

Similarly to Internet Explorer, Selenium offers several methods to find page objects: FindElementbyID, FindElementsbyClass and many more. Use the Intellisense feature to see the full list, as shown beneath.

	A	B	C	D	E	F	G	H	I
1									
2	1,0000	Target	Ticker	Acquirer	Ticker	Current Price	Take Over Price	Price Diff	% Diff
3	2,0000	Acacia Communications Inc Co	ACIA	Cisco Systems Inc Com	CSCO	R\$ 67,49	R\$ 70,00	R\$ 2,51	3,72%
4	3,0000	Advanced Disposal Services In	ADSW	Waste Management Inc	WM	R\$ 30,27	R\$ 30,30	R\$ 0,03	0,10%
5	4,0000	Amag Pharmaceuticals Inc Com	AMAG	Covis Group	--	R\$ 13,63	R\$ 13,75	R\$ 0,12	0,88%
6	5,0000	American Renal Holdings Inc Co	ARA	Nautic Partners, LLC	--	R\$ 11,48	R\$ 11,50	R\$ 0,02	0,17%
7	6,0000	Acorn International Inc Adr	ATV	Management	--	R\$ 20,40	R\$ 21,00	R\$ 0,60	2,94%
8	7,0000	Avenue Therapeutics Inc Com	ATXI	Cipla Limited	--	R\$ 3,84	R\$ 13,92	R\$ 10,08	262,50%
9	8,0000	Bay Banks Virginia Inc Com	BAYK	Blue Ridge Bankshares	BRBS	R\$ 6,75	R\$ 6,87	R\$ 0,12	1,78%
10	9,0000	Bitauto Hldgs Ltd Adr	BITA	Yiche Holding Limited	--	R\$ 15,84	R\$ 16,00	R\$ 0,16	1,01%
11	10,0000	Bmc Stock Holdings Inc Com	BMCH	Builders Firstsource Inc	BLDR	R\$ 43,60	R\$ 43,97	R\$ 0,37	0,85%
12	11,0000	Bankia S.A. Com	BNKXF	Caixabank Sa Adr Unspc	CAIXY	R\$ 1,27	R\$ 1,35	R\$ 0,08	6,40%
13	12,0000	Cincinnati Bell Inc Com	CBB	Macquarie Infrastructur	--	R\$ 15,13	R\$ 15,50	R\$ 0,37	2,45%
14	13,0000	Cellular Biomedicine Group In	CBMG	Management	--	R\$ 18,37	R\$ 19,75	R\$ 1,38	7,51%
15	14,0000	Canfor Corp New Com	CFPZF	Great Pacific Capital Co	--	R\$ 11,56	R\$ 12,13	R\$ 0,57	4,96%
16	15,0000	Cit Group Inc Com	CIT	First Citizens Bancshare	FCNCA	R\$ 25,00	R\$ 24,39	-R\$ 0,61	-2,43%
17	16,0000	Creso Pharma Ltd Com	COPHF	Pharmaciolo Ltd Com	PHCEF	R\$ 0,03	R\$ 0,00	-R\$ 0,03	-100,00%
18	17,0000	China Xd Plastics Com	CXDC	Management	--	R\$ 1,18	R\$ 1,20	R\$ 0,02	1,69%
19	18,0000	Dime Community Bancshares	DCOM	Bridge Bancorp Inc Com	BDGE	R\$ 12,54	R\$ 12,64	R\$ 0,10	0,82%
20	19,0000	Delmarva Bancshares Inc Com	DLMV	Bv Financial Inc Com	BVFL	R\$ 8,75	R\$ 8,90	R\$ 0,16	1,77%

Partial view of the data already in Excel



Using Intellisense to visualize available options

Using XPath to find elements

XPath stands for XML Path Language and is a query tool originally developed to select nodes in a XML document. It is implemented in Selenium and can be used to navigate through the HTML structure of the Web page, locating any element by means of a specific syntax. There are two main XPath categories:

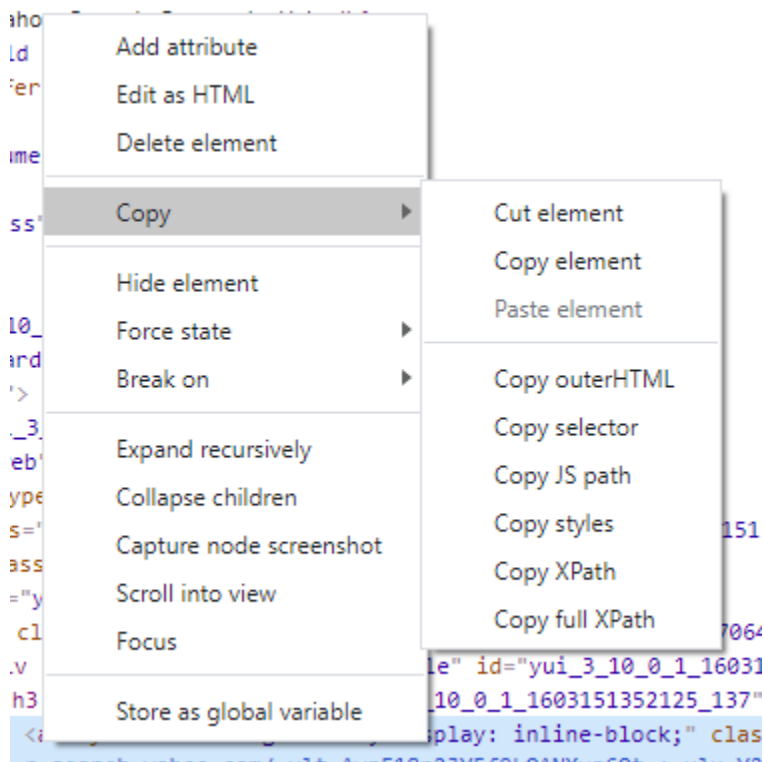
- Absolute: it is the complete path for the element, beginning from the root element. The disadvantage of this category is that it may be rather long and more prone to fail if the HTML DOM structure changes. You can recognize an absolute path by the starting single forward slash.
- Relative: this reference starts from somewhere in the middle of the DOM structure. A relative XPath starts with double forward slashes and is usually the preferred option.

Note: You will not be able to read much on Web topics before stumbling into the DOM. But what is it? According to Mozilla, “The Document Object Model (DOM) is an API for HTML and XML documents. It provides a structural representation of the document, enabling you to modify its content and visual presentation by using a scripting language such as JavaScript.”

For our purposes, think of the DOM as a virtual representation of the web page structure as a tree, complete with all its objects, styles, contents and events.

These are the steps to get the XPath for a page element:

1. Right click any page element and choose inspect
2. When the HTML source code is displayed, right click the object of interest
3. Click copy to see all the options, as depicted below
4. Click the desired option to transfer the information to the clipboard



Getting the XPath for a page element

Here is a summary for the example that follows:

- Opens a free financial site and fills in the login credentials from worksheet cells
- Gets a handle for a popup window button and clicks it to dismiss the popup
- Navigates to my personal page on that site; you can see the final result below.

```
Public driver As New ChromeDriver
Sub Filling_Fields()
Dim pt As WebElement, ac As WebElement
driver.get "https://www.investing.com/portfolio/"
Application.Wait Now + TimeValue("0:00:04")
Set pt = driver.FindElementByXPath _
("//*[@id=""loginFormUser_email""]").SendKeys _
(CStr(Sheets("test").[b85]))
Set pt = driver.FindElementByXPath _
("//*[@id=""loginForm_password""]").SendKeys _
(CStr(Sheets("test").[b86]))
```

```

Set pt = driver.FindElementByXPath("//*[@id=""signup""]/a")
DoEvents
' get popup button
Set ac = driver.FindElementById("onetrust-accept-btn-handler")
ac.Click
DoEvents
pt.Click
Application.Wait Now + TimeValue("0:00:03")
Set pt = driver.FindElementByXPath _
("//*[@id=""navMenu""]/ul/li[9]/a")
pt.Click: DoEvents
Application.Wait Now + TimeValue("0:00:02")
End Sub

```

Note that this code uses a SendKeys method for a WebElement variable; this is not the same SendKeys that VBA offers and is much frowned upon.

Investing.com Search the website... Eduardo [Block Ads]

Markets Insights Premium News Analysis Charts Technical Brokers Tools Portfolio Education Crypto More

Markets Major Indices Indices Futures All Cryptocurrencies Real Time Commodities Live Charts Technical Summary Economic Calendar

All portfolios [Import Portfolio](#)

Buy **Sell**

Portfolio +

Add symbol EUR/USD or AAPL

Summary Charts

Price		Fundamental	Technical	Performance	Candlestick Patterns	Customize				
Name	Symbol	Last	Open	High	Low	Chg.	Chg. %	Vol.	Time	
Palm Oil USD	FUPoc1	725.50	725.50	725.50	725.50	+7.25	+1.01%	-	01:00:00	
Dollar Index Futures	DX	92.727	92.720	92.773	92.700	+0.092	+0.10%	0.65K	21:10:54	
USD/CHF		0.9064	0.9053	0.9069	0.9046	+0.0011	+0.12%	9.75K	21:11:10	

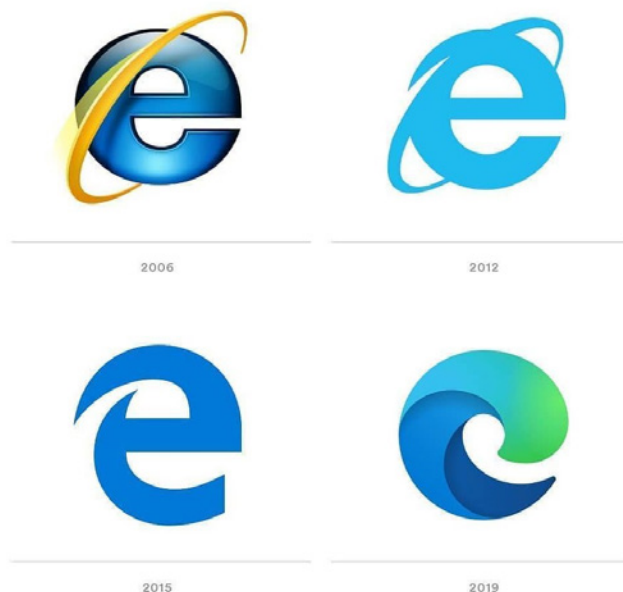
Logging in via code

MICROSOFT EDGE AND VBA

Microsoft developed eleven Internet Explorer versions from 1995 to 2013. Then, they announced that Internet Explorer 11 was going to be the last in this lineage and started focusing on Edge. The good news for IE users is that it is part of Windows 10; so, if you have some application that uses specific features of IE 11 there is no reason for immediate concern.

The next big news came in 2019 when it was revealed that this initial Edge version, often referred to as Edge HTML, was being replaced by the new Chromium based Edge.

That is right, it uses the same open source implementation that is the base for Google Chrome, Opera and other lesser known browsers. You can recognize Microsoft's family visually as shown below:



Microsoft browsers over the years

Before you can start using Edge with VBA, it is necessary to update the corresponding Selenium driver; follow the steps below to do that.

1. Download the correct driver from this page, after checking your Edge version and operating system type.

↓ Downloads

By downloading and using this software, you agree to the license terms below.

Microsoft Edge

Microsoft Edge Driver is now downloadable separately from Windows. Below you can find the latest versions of WebDriver. If your version is not listed please visit the [full version directory](#).

Release 88

Version: 88.0.685.0 | Choose your OS: [x86](#), [x64](#), [ARM64](#) | [License terms](#) | [Credits](#)

Version: 88.0.684.0 | Choose your OS: [x86](#), [x64](#), [Mac](#), [ARM64](#) | [License terms](#) | [Credits](#)

Version: 88.0.683.0 | Choose your OS: [x86](#), [x64](#), [Mac](#), [ARM64](#) | [License terms](#) | [Credits](#)

Microsoft

Microsoft We Demand.

To install run DISM.exe /Online /CapabilityN...

For builds pri...

Microsoft Ed...

[Release 17](#)

Downloading the Edge driver

2. Delete the edgedriver.exe file at the Selenium Basic folder
3. Copy the downloaded driver to that folder and rename it to edgedriver.exe

Now we are ready for the first example of this chapter.

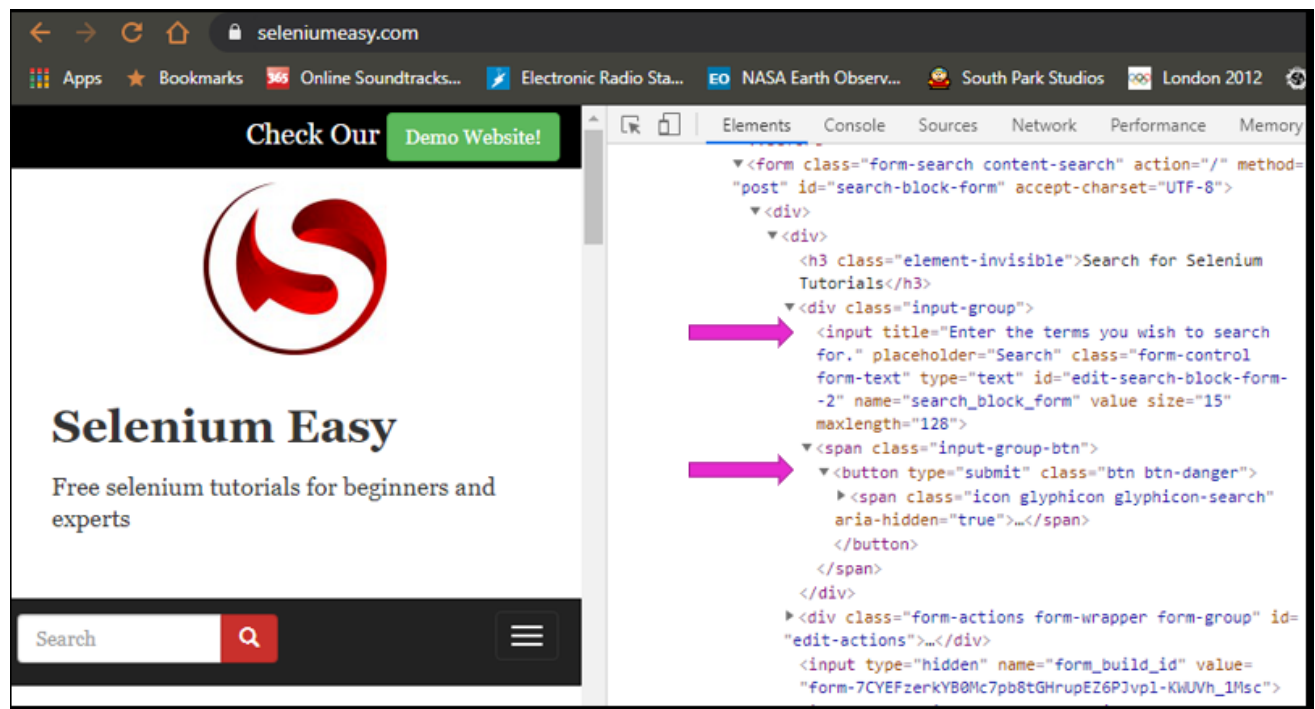
Finding elements by CSS selectors

Yet another way to locate a web page element is the CSS selector. You may remember that CSS is the language used to apply styles to the page.

Say that you want to fill in a search box on the web site shown below and click the search button to get the results. Note that the box has an input tag, while the button has... a button tag.

Since the input box has a unique ID, it can be referred directly. The button, on the other hand, has no ID, hence it is identified by its DOM path, starting at the form element. When the code concludes execution, an Edge tab displays the results of the search for the chosen subject.

To find out what is the CSS selector for a particular element, use the same steps described at the Chrome chapter to get a XPath, but instead choose the copy selector option.



Referencing two specific web page elements

```
Public driver As New EdgeDriver
Private Sub Use_Edge()
Dim e As WebElement
driver.get "https://www.seleniumeasy.com"
driver.Wait 3000
Set e = driver.FindElementByCss("#edit-search-block-form--2")
If e.IsDisplayed Then
    e.SendKeys ("css") ' write search string
    Set e = driver.FindElementByCss _
    ("#search-block-form > div > div >" & _
    " div.input-group > span > button")
    driver.Wait 1000
    e.Click
End If
'driver.Quit
End Sub
```

Downloading a file

A common task when working online is to download a file. There are efficient methods to accomplish this, such as FTP (File Transfer Protocol) clients or P2P(peer-to-peer) systems. However, you may encounter a website where the only way for downloading is by icon clicking. The example that follows shows how to automate this process.



The figure above displays part of the page's HTML source code. Each Excel file has its address contained on a href attribute, so all we need to do is to use this information to download a specific workbook.

```
Public ed As New EdgeDriver
Public Sub Dload()
Dim col, i%, s$, p$, v
ed.get "https://www.wisdomaxis.com/technology/" & _
"software/data/for-reports/"
ed.Wait 2000
Set col = ed.FindElementsByTag("a")
For i = 1 To col.Count
    s = col.item(i).Attribute("href")
    If s Like "*Refresh*" Then Exit For ' desired file
Next
ed.SetProfile "Download", persistant:=True
ed.SetPreference "browser.download.folderList", 2
ed.SetPreference "browser.helperApps.neverAsk.saveToDisk", _
"application/vnd.ms-excel, application/msword," & _
" application/msexcel," & _
" application/xls, application/csv, text/csv," & _
" application/pdf, text/html, text/plain"
ed.SetPreference "browser.download.manager.showWhenStarting", 0
ed.SetPreference "browser.download.manager.focusWhenStarting", 0
ed.SetPreference "browser.download.useDownloadDir", True
ed.SetPreference "browser.helperApps.alwaysAsk.force", False
ed.SetPreference "browser.download.manager.alertOnEXEOpen", False
ed.SetPreference "browser.download.manager.closeWhenDone", True
ed.SetPreference "browser.download.manager.showAlertOnComplete", 1
ed.SetPreference "browser.download.manager.useWindow", False
ed.SetPreference "pdfjs.disabled", True
ed.get s ' download file
p = Split(Environ$(41), "=") (1) & "\downloads\" ' build path
v = Split(s, "/")
Workbooks.Open p & Replace(v(UBound(v)), "%20", " ")
'ed.Window.Maximize
Workbooks(Workbooks.Count).Activate
End Sub
```

The main steps of this code are:

- Creates a collection of elements with the <a> tag
- Loops the collection to find the desired file
- Sets several download preferences
- Downloads and opens the workbook

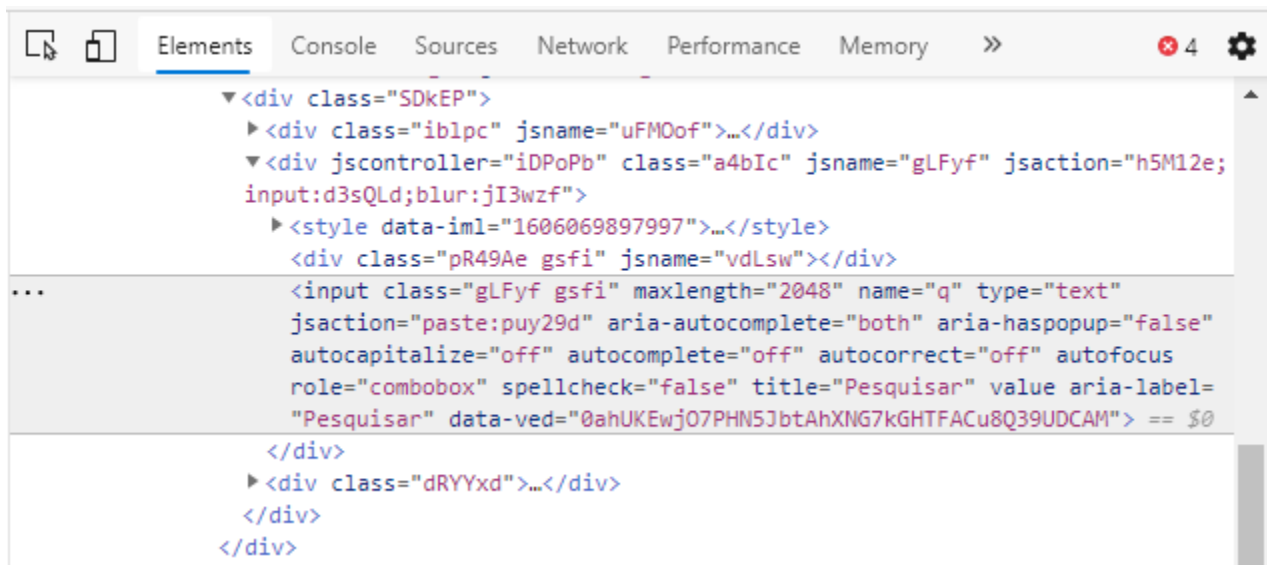
Note: Do you want to open Microsoft Edge without installing Selenium? This one liner will do it. Note that it is not necessary to know the Edge installation path.

```
Sub ShellEdge()
VBA.Shell Environ$("comspec") & " /c start microsoft-edge:" & _
"http://www.mrexcel.com"
End Sub
```

Creating a PDF file from scratch

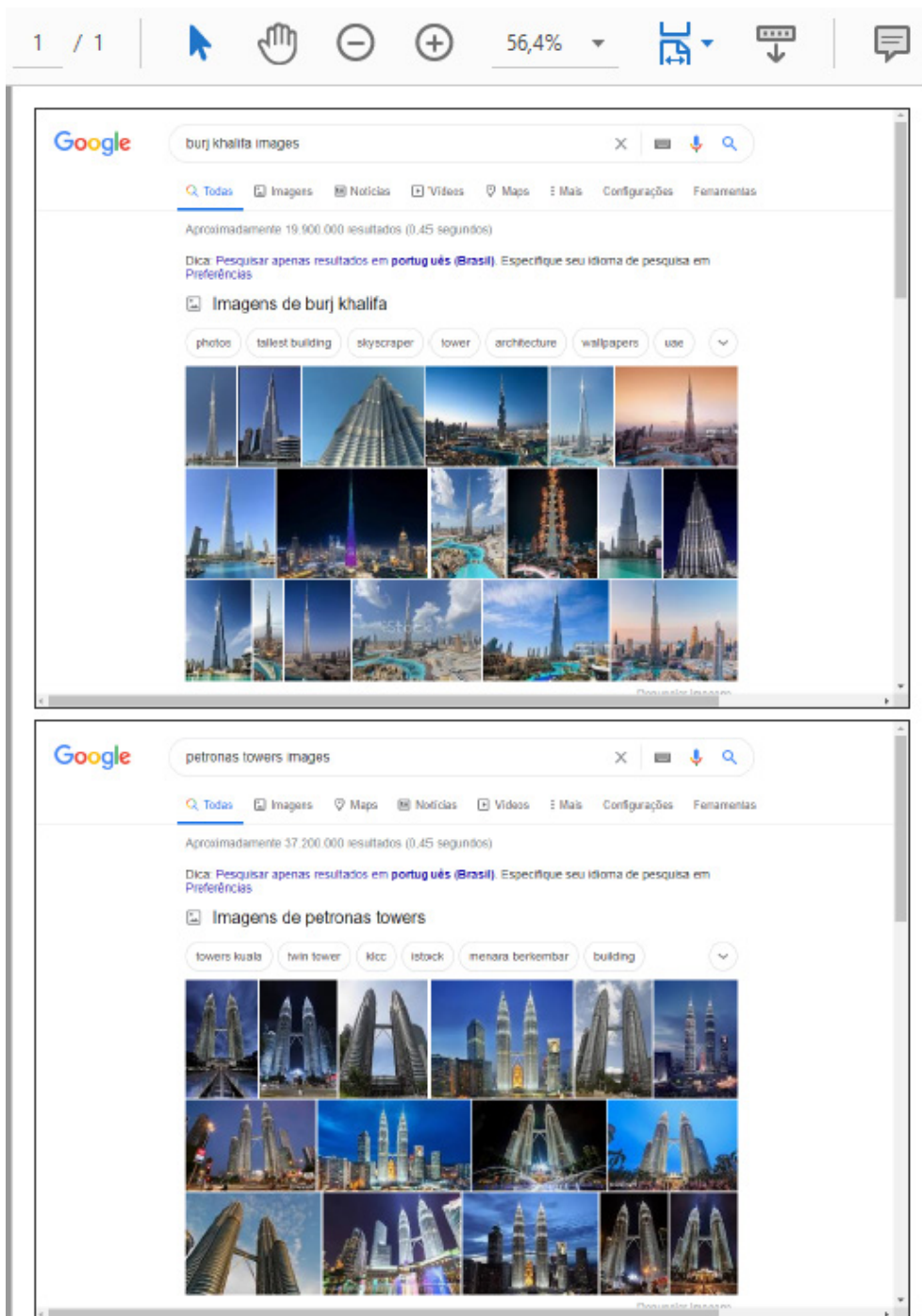
Say that you want to catalog several web pages in PDF format for later analysis. The example that follows shows how to build the file.

- It uses an array to store search terms that will be passed to Google. This can be easily modified to get the information from a worksheet range.
- The code takes advantage of the input box's title attribute to find it; this string will be different depending on your language, adjust it accordingly. The picture below shows how it looks like for my Portuguese version.
- The macro will loop the array, creating a new Google tab for each element. After performing the search, a screenshot is taken and added to the PDF file. At the end, the file is saved to disk; the final product is displayed after the code.



The HTML source code for Google's search box, Portuguese version.

```
Public Sub CreatePDF()
Dim ed As EdgeDriver, arr(), i%, pdf As Object
Set ed = New EdgeDriver
Set pdf = CreateObject("Selenium.PdfFile")
pdf.SetPageSize 210, 300, "mm"
pdf.SetMargins 5, 5, 5, 15, "mm"
arr = Array("burj khalifa", "petronas towers")
With ed
    .Start "edge"
    .get "https://google.com/"
    For i = LBound(arr) To UBound(arr)
        If Not IsEmpty(arr(i)) Then
            If i > 0 Then
                .ExecuteScript "window.open(arguments[0])", _
                    "https://google.com/"
                .SwitchToNextWindow
            End If
            .FindElementByCss("[title=Pesquisar]").SendKeys _
                arr(i) & " images"
            .SendKeys .keys.Enter
            pdf.AddImage .TakeScreenshot, True
        End If
    Next
End With
pdf.SaveAs "c:\temp\selenium\arch.pdf"
'Stop
End Sub
```



This PDF file was generated with one click.

Wrapping up Selenium

As you may have already noticed, it does not matter which browser we are using with Selenium, since its tree of properties and methods is the same. The final VBA snippet below exemplifies this; all three browsers are opened with identical statements. The chosen site is yet another one where Internet Explorer cannot perform well anymore...

```
Public ed As New EdgeDriver
Public cr As New ChromeDriver
Public ie As New IEDriver
Sub Choice()
Dim url$
url = "http://www.startrek.com"
Select Case InputBox("1 - Edge / 2 - Chrome / 3 - IE", _
"Select a browser:")
    Case 1: ed.get url
    Case 2: cr.get url
    Case 3: ie.get url ' does not load completely
End Select
End Sub
```

POWER QUERY AND THE WEB

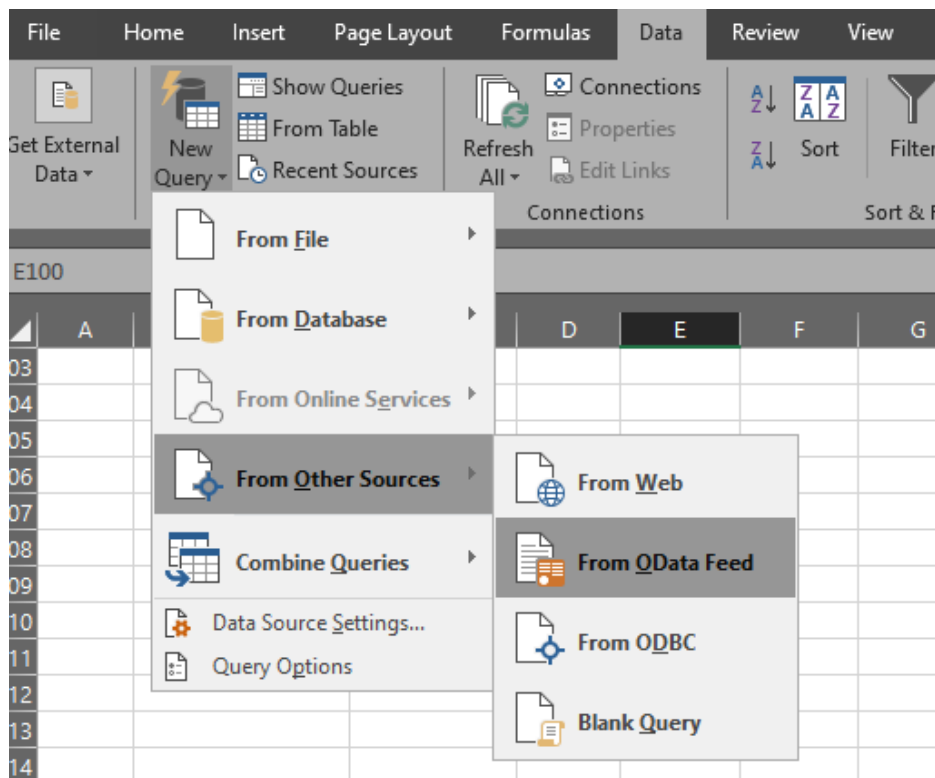
We are living in the Business Intelligence age, and Power Query is part of Microsoft's bundle of BI tools. It will allow you to retrieve data from a variety of sources, including the Internet, reshape it and transfer the finished product to an Excel worksheet, where it can be further manipulated if necessary.

For Excel 2010 and 2013, Power Query was an add-in; from Excel 2016 onwards, it is a native resource. The examples in this chapter will be based on Excel 2016.

Connecting to a Web table – the direct way

In this first example we will see a straightforward procedure to create a connection to data dwelling on a Web page.

1. At Excel's Data tab, choose New Query > From other sources > From Web



Launching Power Query from Excel

2. Paste the Web page address that contains the desired table at the dialog box and click OK. The address I used appears at this example's last screenshot.
3. Another dialog box listing all available tables will appear; click the one you want and then hit the Transform Data button.

Navigator

Table View Web View

5000 metres[edit]

Preview downloaded on segunda-feira

Rank	Name	Nationality	Time	Notes
null	Christine Bardelle	France	15:39.54	null
null	Silvia Weissteiner	Italy	15:44.53	null
null	Elena Romagnolo	Italy	16:11.17	null
4	Ourania Rebouli	Greece	16:21.45	null
null	Esma Aydemir	Turkey	DNF	null
null	Nadia Noujani	Morocco	DNF	null

Load Transform Data Cancel

Previewing the Web table

- At this point, the Power Query Editor finally enters the stage. It is important to understand that, although having a Ribbon, a formula bar, rows, columns and a programming language (called M), Power Query is not Excel! The two of them are different applications, but they do talk to each other.
- As you can see on the picture below, the data is already there, but it requires some tweaking. Let's start by deleting the Rank and Notes columns. Select the column you want to eliminate and at the Home tab, click Remove Columns. There is also an option to delete the columns that are *not* selected.

5000 metres[edit] (2) - Power Query Editor

Arquivo Home Transform Add Column View

Close & Load Close Refresh Preview Manage Query

Properties Advanced Editor

Choose Columns Remove Columns Manage Columns

Keep Rows Remove Rows Reduce Rows

Sort

Split Column Group By

Data Type: Whole Number Use First Row as Headers Replace Values

Transform

Queries

fx = Table.TransformColumnTypes(Data34,{{"Rank", Int64.Type}, {"Name", type text}, {"Nationality", type text}, {"Time", type text}, {"Notes", type text}})

	Rank	Name	Nationality	Time	Notes
1	null	Christine Bardelle	France	15:39.54	null
2	null	Silvia Weissteiner	Italy	15:44.53	null
3	null	Elena Romagnolo	Italy	16:11.17	null
4	4	Ourania Rebouli	Greece	16:21.45	null
5	null	Esma Aydemir	Turkey	DNF	null
6	null	Nadia Noujani	Morocco	DNF	null

The Power Query editor

6. Now we should add a column to hold the runners' positions. To do this, at the Add Column tab, click Index column > From 1. After the insertion, drag it by the title row all the way to the left, and the table is ready to be exported back to Excel.

= Table.ReorderColumns("#Removed Columns1",{"Index", "Name", "Nationality", "Time"})				
1 ² Index	A ^B _C Name	A ^B _C Nationality	A ^B _C Time	
1	1 Christine Bardelle	France	15:39.54	
2	2 Silvia Weissteiner	Italy	15:44.53	
3	3 Elena Romagnolo	Italy	16:11.17	
4	4 Ourania Rebouli	Greece	16:21.45	
5	5 Esma Aydemir	Turkey	DNF	
6	6 Nadia Noujani	Morocco	DNF	

The table after some Power Query editing

7. Choose File > Close and load to... determine where the data should go and click the Load button. The query is now built and can be refreshed anytime; the above steps need not to be repeated.

×

Load To

Select how you want to view this data in your workbook.

☒ Table
 ☐ Only Create Connection

Select where the data should be loaded.

☐ New worksheet
☒ Existing worksheet:

\$A\$105

☐ Add this data to the Data Model

Load

Cancel

Transferring data to the worksheet

	A	B	C	D	E	F	G	H
102								
103	https://en.wikipedia.org/wiki/Athletics_at_the_2013_Mediterranean_Games_%E2%80%93_Results							
104								
105	Index	Name	Nationality	Time				
106	1	Christine Bardelle	France	15:39.54		This is Excel		
107	2	Silvia Weissteiner	Italy	15:44.53				
108	3	Elena Romagnolo	Italy	16:11.17				
109	4	Ourania Rebouli	Greece	16:21.45				
110	5	Esma Aydemir	Turkey	DNF				
111	6	Nadia Noujani	Morocco	DNF				
112								

The final query table.

Connecting to a Web table – the long way

Power Query is an evolving product, but right now it cannot automatically detect all tables on a Web page. Sometimes we get no tables at all when opening the Power Query navigator dialog box. However, a workaround popularized by Microsoft MVP Gil Raviv allows us to manually find and import the data.

For this final example, let's walk the HTML structure of the page below until we find that table. Actually, Power Query lists this one, but we are going to pretend it is not available. This technique is generic and can be applied to a wide range of Web pages.

The screenshot shows the website **climatedata.eu** with the title "Temperature - Precipitation - Sunshine". The main content area is titled "Climate Chur - Switzerland". On the left, there is a search bar and a list of countries: Austria, France, Greece, Ireland, Italy, and Netherlands. The main table displays climate data for June, with columns for Jan, Feb, Mar, Apr, May, and June. The table includes rows for Average high in °C, Average low in °C, Av. precipitation - mm, Days with precip., and Hours of sunshine.

	Jan	Feb	Mar	Apr	May	June
Average high in °C	4	6	10	14	19	22
Average low in °C	-4	-2	0	3	7	10
Av. precipitation - mm	52	51	47	51	75	91
Days with precip.	8	7	8	9	11	11
Hours of sunshine	83	106	139	146	172	176

Looking for this table.

So, here are the necessary steps to manually import it into Excel using Power Query:

- Use the From Web option inside Excel, as in the previous example.
- After entering the page address, at the navigator dialog, choose Document. Our table is Table 0, but clicking it would be too easy...
- Click the transform button to load the Power Query editor. This puts us at the first step of the below figure, which displays how the PQE will look like as we proceed. The idea is to click the correct Table nodes at the Children column so we can navigate the HTML tree until we find the desired data.

1

f_x

= Source{3}[Data]

	A_C^B Kind	A_C^B Name	Children	A_C^B Text
1	Element	HTML	Table	null

2

f_x

= Data3{0}[Children]

	A_C^B Kind	A_C^B Name	Children	A_C^B Text
1	Element	HEAD	Table	null
2	Element	BODY	Table	null

3

f_x

= Children{1}[Children]

	A_C^B Kind	A_C^B Name	Children	A_C^B Text
1	Element	DIV	Table	null
2	Element	DIV	Table	null
3	Element	SCRIPT	null	null
4	Element	SCRIPT	null	null
5	Element	SCRIPT	null	null
6	Element	SCRIPT	null	null
7	Element	SCRIPT	null	null
8	Element	DIV	Table	null

4

f_x

= Children1{0}[Children]

	A_C^B Kind	A_C^B Name	Children	A_C^B Text
1	Element	DIV	Table	null
2	Element	DIV	Table	null
3	Element	DIV	Table	null
4	Element	DIV	Table	null

5

f_x

= Children2{1}[Children]

	A_C^B Kind	A_C^B Name	Children	A_C^B Text
1	Element	DIV	null	null
2	Element	DIV	Table	null
3	Element	DIV	Table	null

The first five steps

6

f_x

Steps six to nine.

The following picture shows the corresponding page HTML, and some of the steps mentioned above are also indicated. The target table is highlighted at the left, and also at the bottom of the HTML code.

The screenshot shows a web browser window with the URL <http://www.w3.org/TR/xhtml1/>. The page title is "Climate Chur - Switzerland". The page content includes a table with climate data for January and July, and a climate chart for Chur - Grisons. The HTML code is displayed on the right, with annotations 1 through 6 pointing to specific elements:

- 1. Points to the `<table>` element in the HTML code.
- 2. Points to the `<body>` element in the HTML code.
- 3. Points to the `<div id="wrap">` element in the HTML code.
- 4. Points to the `<div id="content-wrap">` element in the HTML code.
- 5. Points to the `<div id="main">` element in the HTML code.
- 6. Points to the `<div id="klimaattabel">` element in the HTML code.

The Web page HTML structure

Here are the instructions to perform the nine steps; compare them with the HTML above to better understand the procedure, because evidently every Web page will have a different design.

1. Click the Table link at the only row available.
2. Click at the second row (a BODY element).
3. Click at the first row (a DIV element).
4. Click at the second row (a DIV element).
5. Click at the third row (a DIV element).
6. Click at the third row (a DIV element).
7. Click at the first row, this is already our table!
8. Click at the only row available, which is a TBODY element.
9. We already have the six table rows at this point; now expand the Children column by clicking the two little arrows facing away from each other. At the dialog box, uncheck the other columns as we do not need to expand them.

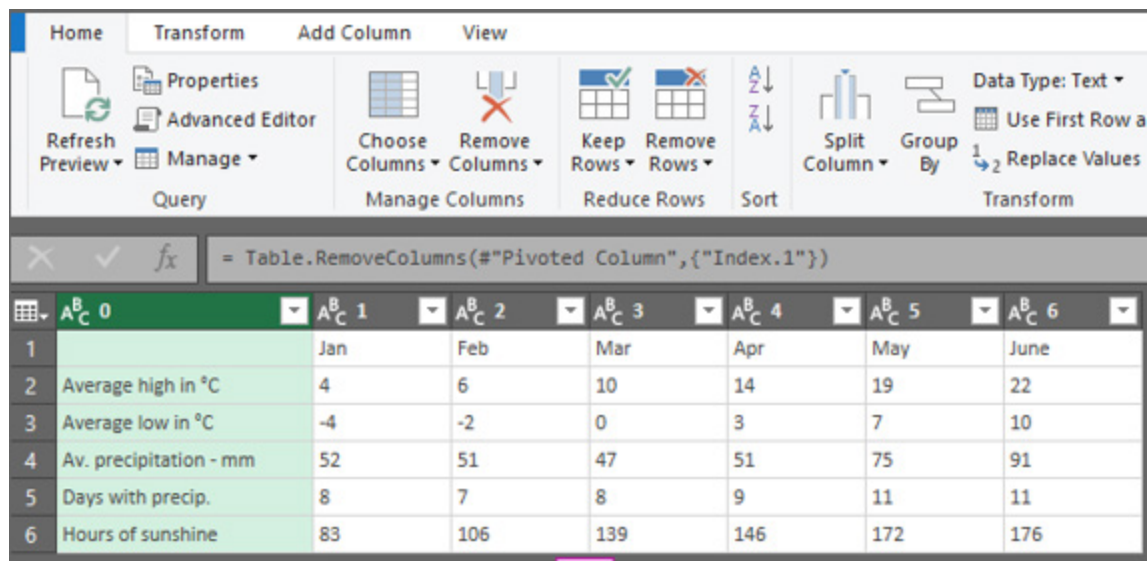
- Now Power Query is showing 42 rows; this time, expand the Children.Children column and all of our data will be displayed, although in a single column. You may now remove the other columns as already explained in the previous example.
- The last part is to rearrange the data so that it looks like the original table. At the Add Column tab, choose Index Column > from zero.
- At the Transform tab, click Standard > Modulo. When the dialog box appears, enter the number 7, which is the number of rows in the original table.
- At the Add Column tab, click Index Column > from zero.
- At the Transform tab, click Standard > integer – divide, and enter the number 7 again.
- Select the Index column, and at the Transform tab, click Pivot Column. When the dialog box pops up, choose the options shown in the image below.

The screenshot shows the Power Query Editor interface. The ribbon includes Home, Transform, Add Column, and View. The formula bar displays the query: `= Table.TransformColumns(#"Added Index1", {"Index.1", each Number.IntegerDivide(_, 7), Int64.Type})`. The data table has three columns: 'Children.Children.Text', 'Index', and 'Index.1'. The 'Index' column is highlighted in green. The 'Pivot Column' dialog box is open, showing the 'Values Column' set to 'Children.Children.Text' and the 'Aggregate Value Function' set to 'Don't Aggregate'.

	Children.Children.Text	Index	Index.1
1		0	0
2	Jan	1	0
3	Feb	2	0
4	Mar	3	0
5	Apr	4	0
6	May	5	0
7	June	6	0
8	Average high in °C	0	1
9	4	1	1
10	6	2	1
11	10	3	1
12	14	4	1
13	19	5	1
14	22	6	1
15	Average low in °C	0	2

Pivoting in Power Query

- Click the OK button and remove the column labeled Index.1
- Finally we have the data in the proper format! Now choose Close & Load to, pick your options in the dialog box and hit the Load button; the table is imported into Excel, as depicted beneath.



The Power Query ribbon shows the 'Remove Columns' step. The formula bar displays the following formula:

```
= Table.RemoveColumns(#"Pivoted Column",{"Index.1"})
```

The table below shows the data after removing the 'Index.1' column:

	A _C 0	A _C 1	A _C 2	A _C 3	A _C 4	A _C 5	A _C 6
1		Jan	Feb	Mar	Apr	May	June
2	Average high in °C	4	6	10	14	19	22
3	Average low in °C	-4	-2	0	3	7	10
4	Av. precipitation - mm	52	51	47	51	75	91
5	Days with precip.	8	7	8	9	11	11
6	Hours of sunshine	83	106	139	146	172	176

A large pink arrow points down to the resulting Excel table below:

	A	B	C	D	E	F	G	H
134								
135		0	1	2	3	4	5	6
136			Jan	Feb	Mar	Apr	May	June
137		Average high in °C	4	6	10	14	19	22
138		Average low in °C	-4	-2	0	3	7	10
139		Av. precipitation - mm	52	51	47	51	75	91
140		Days with precip.	8	7	8	9	11	11
141		Hours of sunshine	83	106	139	146	172	176

Transferring the table from Power Query to Excel

CONCLUSION

Excel and the Internet are dynamic entities, but this short book should give you some initial velocity to start dealing with this fascinating topic. One last piece of advice: it is likely that you will get stuck at some point when developing your projects. If that happens, consider asking for help on Web forums like the MrExcel Message Board or Stack Overflow. There you will find knowledgeable people who are always ready to help. Good luck!

INDEX

A

Altavista 2
API 17
Authenticated requests 16

B

Breheret, Florent 24

C

Chrome
 HTML pane 4
Chrome and VBA 28
Chrome driver 25
Client-server architecture 11
CSS 2
 Using selectors 36

D

DOM 32

E

Edge
 Opening with VBA 38
Edge browser 35
Events 21

F

FTP 37

G

GetElementby 21
Get External Data 5

H

HTML 2
Huggins, Jason 24

I

Iframes 30
Index column 44
Inspect element 2
Internet Explorer
 Security settings 22
 VBA 18
IQY
 Extension 5

J

JavaScript 2
 running with VBA 28
Jelen, Bill 1
JSON
 Defined 10

L

Looping through rows 21

M

Microsoft Scripting Runtime 13
MrExcel Message Board 51

N

Netscape 2
Notepad 3

P

P2P 37
PDF creation 39
Posting data 15
Power Query 42
 Editor 43
 From Web 42
 Index column 44
 Manual finding table 45

R

Raviv, Gil 45
Resize dialog box 5

S

Security settings in IE 22
Selenium 24
SendKeys 34
Stack Overflow 51

T

TR tags 19

V

VBA
 Internet Explorer 18
 Reading XML 10
 Web query 8
VBA-JSON converter 12

W

Web query 4
 Dynamic 6
 static 5
 with Parameter 8
 with VBA 8
Workbook Connections 6

X

XPath 32

Y

YouTube 18