SCRAPING FOR JOURNALISTS

How to grab data from hundreds of sources, put it in a form you can interrogate – and still hit deadlines

PAUL BRADSHAW

ONLINE JOURNALISM BLOG
A conversation.

Scraping for Journalists (2nd edition)

How to grab information from hundreds of sources, put it in data you can interrogate - and still hit deadlines

Paul Bradshaw

This book is for sale at http://leanpub.com/scrapingforjournalists

This version was published on 2017-06-19

Tweet This Book!

Please help Paul Bradshaw by spreading the word about this book on Twitter!

The suggested hashtag for this book is #scrapingforjournos.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#scrapingforjournos

Also By Paul Bradshaw

8000 Holes: How the 2012 Olympic Torch Relay Lost its Way

Model for the 21st Century Newsroom - Redux

Stories and Streams

Organising an Online Investigation Team

Data Journalism Heist

Finding Stories in Spreadsheets

Excel para periodistas

Periodismo de datos: Un golpe rápido

Learning HTML and CSS by making tweetable quotes

Snapchat for Journalists

Snapchat para periodistas

For Joseph, who loves robots, Max, who likes asking questions, and Claire, who has all the answers.

Contents

1. Scraper #1: Start scraping in 5 minutes

David Bowie discography

From Wikipedia, the free encyclopedia

English singer David Bowie (born David Jones) released 27 studio albums, 9 live albums, 49 compilation albums, 8 extended plays (EPs), 121 singles, including 5 UK number one singles, and 3 soundtracks. Bowie also released 14 video albums and 59 music videos.

Bowie's debut release was the 1964 single "Liza Jane" by Davie Jones & the King Bees. He released two more singles in 1965 under the names of The Manish Boys and Davy Jones & the Lower Third. His first release using the name David Bowie was the 1966 single "Can't Help Thinking About Me", which was released with The Lower Third. Bowie's next single, "Do Anything You Say", also released in 1966, was the first release by simply David Bowie. Bowie released four more singles and his debut album, *David Bowie*, but the first success in the United Kingdom was with the 1969 single "Space Oddity". The single reached number five on the UK Singles Chart after it was released five days before the Apollo 11 moon mission.[1]

Bowie released three more albums – *David Bowie* (1969), *The Man Who Sold the World* (1970) and *Hunky Dory* (1971) – before he eventually made it on to the UK Albums Chart with *The Rise and Fall of Ziggy Stardust and the Spiders from Mars* (1972), which peaked at number five. Following the success of *Ziggy Stardust*, sales of *Hunky Dory* improved and it eventually peaked at number three in the UK. RCA re-released the 1969 *David Bowie* under the title Space Oddity (with an updated Ziggy Stardust-era photo) and *The Man Who Sold the World*, which reached numbers 17 and 26 in the UK respectively. Bowie released nine more studio albums with RCA, all

David Bowie discography

Bowie during the Heathen Tour in 2002

Studio albums	27
Live albums	9
Compilation albums	49

In this chapter I want to show you just how quickly you can get a simple scraper running - but along the way start exploring some key concepts which you'll need as you start to work on different challenges and more advanced scrapers.

You can write a very basic scraper by using Google Drive, selecting **Create>Spreadsheet**, and adapting this formula - it doesn't matter where you type it:

```
=ImportHTML("ENTER THE URL HERE", "table", 1)
```

This formula will go to the **URL** you specify, look for a **table**, and pull the **first one** into your spreadsheet.

If you're using a Portuguese, Spanish or German version of Google Drive - or have any problems with the formula - use semi colons instead of commas. We're using commas here because this convention will continue when we get into programming in later chapters.

Let's imagine you want to get an overview of the discography of a particular artist: you could use this formula by typing it into the first cell of an empty Google Drive spreadsheet and replacing `"ENTER THE URL HERE"` with `"https://en.wikipedia.org/wiki/David_Bowie_discography"`.

Try it and see what happens. It should look like this:

```
=ImportHTML("https://en.wikipedia.org/wiki/David_Bowie_discography", "table", 1)
```

Don't copy and paste this - it's always better to type directly to avoid problems with hyphenation and curly quotation marks, etc.

After a moment, the spreadsheet should start to pull in data from the first table on that webpage.

So, you've written your first scraper! That didn't take long at all. Now, it *is* a very basic one, and we might want to change some things about it, but by understanding how it works and building on it you can start to make more and more ambitious scrapers with different languages and tools.

How it works: functions and parameters

Let's break this formula down:

```
=ImportHTML("https://en.wikipedia.org/wiki/David_Bowie_discography", "table", 1)
```

The scraping formula above has two core ingredients: a **function**, and **parameters**:

- `importHTML` is the **function**. Functions (as you might expect) *do* things. According to Google Drive's Help pages[1] this one "Imports data from a table or list within an HTML page"
- All the items within the parentheses (brackets) are the **parameters**. Parameters are the *ingredients* that the function needs in order to work. In this case, there are three: a URL, the word `"table"`, and a number `1`.

You can use different functions in scraping to tackle different problems, or achieve different results. Google Drive, for example, also has functions called `importXML`, `importFeed` and `importData` - some of which we'll cover later. And if you're writing scrapers with languages like Python, Ruby, R or PHP you can create your own functions that extract particular pieces of data from a page or PDF (which we'll also cover in later chapters).

What are the parameters? Strings and indexes

Back to the formula:

```
=ImportHTML("https://en.wikipedia.org/wiki/David_Bowie_discography", "table", 1)
```

In addition to the function and parameters, it's important to explain some other things you should notice:

- Firstly, the = sign at the start. This tells Google Drive that this is a **formula**, rather than a simple number or text entry
- Secondly, notice that two of the three parameters use straight quotation marks: the URL, and `"table"`. This is because they are **strings**: strings are basically words, phrases or any other collection (i.e. *string*) of characters. The computer treats these differently to other types of information, such as numbers, dates, or cell references - we'll come across these again later
- The third parameter does not use quotation marks, because it is a number. In fact, in this case it's a number with a particular meaning: an **index** - the position of the table we're looking for (first, second, third, etc)

Knowing these things helps both in avoiding mistakes (for example, if you omit a quotation mark or use curly quotation marks it won't work) and in adapting a scraper...

For example, perhaps the table you got wasn't the one you wanted. Indeed, the first table on that webpage is a general list of the number of studio albums, live albums, and so on, that he recorded. But perhaps we want the table which goes into more detail on each album and how they charted.

Try replacing the number `1` in your formula with a number `2`. This should now scrape the second table (in Google Drive an index starts from `1`):

[1] https://support.google.com/docs/answer/3093339

```
=ImportHTML("https://en.wikipedia.org/wiki/David_Bowie_discography", "table", 2)
```

You should now see the second table from that URL appear (if there is no second table, you will get an `#N/A` error).

Knowing to search for information (often called '**documentation**') on a function is important too. The page on Google Drive Help[2], for example, explains that we can use `"list"` instead of `"table"` if you wanted to grab a list from the webpage.

So try that, and see what happens (make sure the webpage has a list).

```
=ImportHTML("https://en.wikipedia.org/wiki/David_Bowie_discography", "list", 1)
```

You can also try replacing either string with a cell reference. For example:

```
=ImportHTML(A2, "list", 1)
```

In this case it will look in cell A2 for the URL instead. So in cell A2 type or paste:

```
https://en.wikipedia.org/wiki/David_Bowie_discography
```

Notice that *you don't need quotation marks around the URL if it's in another cell.*

Using cell references like this makes it easier to change your formula: instead of having to edit the whole formula you only have to change the value of the cell that it's drawing from.

For examples of scrapers that do all of the above, see this example[3].

Although this is described as a 'scraper' the results only exist as long as the page does. The advantage of this is that your spreadsheet will update every time the page does (you can set the spreadsheet to notify you by email whenever it updates by going to **Tools>Notification rules** in the Google spreadsheet and selecting how often you want to be updated of changes).

The disadvantage is that if the webpage disappears, so will your data. So it's a good idea to keep a static copy of that data in case the webpage is taken down or changed. You can do this by selecting all the cells and clicking on **Edit>Copy** then going to a new spreadsheet and clicking on **Edit>Paste values only**.

Tables and lists?

There's one final element in this scraper that deserves some further exploration: what it means by "table" or "list".

When we say "table" or "list" we are specifically asking it to look for a **HTML tag** in the code of the webpage. You can - and should - do this yourself…

Look at the raw HTML of your webpage by right-clicking on the webpage and selecting **View Page Source**, or using the shortcuts CTRL+U (Windows) and CMD+U (Mac) in Firefox. You can also view it by selecting **Tools > Web Developer > Page Source** in Firefox or **View > Developer > View Source** in Chrome. *Note: for viewing source HTML, Firefox and Chrome are generally better set up than other browsers.*

You'll now see the HTML. Use **Edit>Find** on your browser (or CTRL+F) to search for `<table` (don't search for `<table>` with the `>` character because sometimes table tags have extra information before that, like `<table width="100">`).

When `=importHTML` looks for a table, this tag is what it looks for - and it will grab everything between `<table ...>` and `</table>` (which marks the end of the table)

[2] https://support.google.com/docs/answer/3093339

[3] https://docs.google.com/spreadsheet/ccc?key=0ApTo6f5Yj1iJdDBSb0FPQm9jUjYzdjcyNWlUTjVYMFE

When `"list"` is specified, `=importHTML` will look for the tags `<ul>` (unordered list - normally displayed as bullet lists) or `<ol>` (ordered list - normally displayed as numbered lists). The end of each list is indicated by either `</ul>` or `</ol>`.

Both tables and lists will include other tags, such as `<li>` (list item), `<tr>` (table row) and `<td>` (table data, i.e. a table cell) which add further structure - and that's what Google Drive uses to decide how to organise that data across rows and columns - but you don't need to worry about them.

Choosing the right index number: the role of trial and error

How do you know what index number to use? Well, there are two ways: you can look at the raw HTML and count how many tables there are - and which one you need. Or you can just use trial and error, beginning with 1, and going up until it grabs the table you want. That's normally quicker.

Trial and error, by the way, is a common way of learning in scraping - it's quite typical not to get things right first time, and you shouldn't be disheartened if things go wrong at first.

Don't expect yourself to know everything there is to know about programming: half the fun is solving the inevitable problems that arise, and half the skill is in the techniques that you use to solve them (some of which I'll cover here). Along the way you'll learn through experience and solve things more quickly in future.

Scraping tip #1: Finding out about functions

We've already mentioned one of those problem-solving techniques, which is to look for the Help pages relating to the function you're using - what's often called the '**documentation**'.

When you come across a function (pretty much any word that comes after the = sign) it's always a good idea to Google it. Google Drive has extensive help pages - documentation - that explain what the function does, as well as discussion around particular questions.

Likewise, as we explore more powerful scrapers such as those hosted on Morph.io, Scraperwiki Classic, or GitHub, we'll talk about searching for 'documentation' and the name of the function to find out more about how it works.

Recap

Before we move on, here's a summary of what we've covered:

- **Functions** *do things*. The are like one-word references to recipes that someone has already written for you
- Functions need ingredients to do this, supplied in **parameters**
- There are different kinds of parameters: **strings**, for example, are collections of characters, indicated by quotation marks…
- …and an **index** is a position indicated by a number, such as first (1), second (2) and so on.
- The strings `"table"` and `"list"` in this formula refer to particular **HTML tags** in the code underlying a page

- You can store parameters elsewhere - for example in another cell - and use a cell reference to bring them in to your formula. This makes it easier to tweak your scraper without having to go into the formula every time

We'll come back to these concepts again and again, beginning with HTML. But before you do that - try this...

Tests

To reinforce what you've just learned - or to test you've learned it at all - here are some tasks to get you solving problems creatively:

- Let's say you need a list of towns in Hungary (this was an actual task I needed to undertake for a story, believe it or not). What **formula** would you write to scrape the first table on this page: `http://en.wikipedia.org/wiki/List_of_cities_and_towns_in_Hungary`
- To make things easier for yourself, how can you change the formula so it uses **cell references** for each of the three parameters? (Make sure each cell has the relevant parameter in it)
- How can you change one of those cells so that the formula scrapes the *second* table?
- How can you change it so it scrapes a *list* instead?
- Look at the **source code** for the page you're scraping - try using the Find command (CTRL+F) to count the tables and work out which one you need to scrape the table of smaller cities - adapt your formula so it scrapes that
- Try to explain what a **parameter** is (tip: choose someone who isn't going to run away screaming)
- Try to explain what an **index** is
- Try to explain what a **string** is
- Look for the **documentation** on related functions like `importData` and `importFeed` - can you get those working?
- Find another musician or band and see if you can scrape their discography. Note: some HTML tables are generated by the website and don't exist in the HTML itself, so this Google Drive scraper won't work. That's something we'll have to deal with elsewhere.

Once you're happy that you've nailed these core concepts, it's time to move on to Scraper #2...