



Web Scrapping

Gathering Data from the Web

Web Scraping!

You may want to...

- download all videos from a website;
- download all news articles from a media platform;
- download all academic papers from a journal;
- download all tweets/weibo of a specific person.

You may need to spend days and nights downloading these data manually, and you can easily make a lot of mistakes.

Web Scraping!

In today's class, we are going to learn about webscraping.
In Chinese, it is called 网络爬虫.

What is webscraping?

Using tools to gather data you can see on a webpage.
Almost anything you see on a website can be scraped.

It can be done with python, R,... We are doing it on R.



Learning about HTML

HTML: HyperText Markup Language.

Websites are written on the HTML language.

Webscraping is based on reading and interpreting the HTML of a webpage.

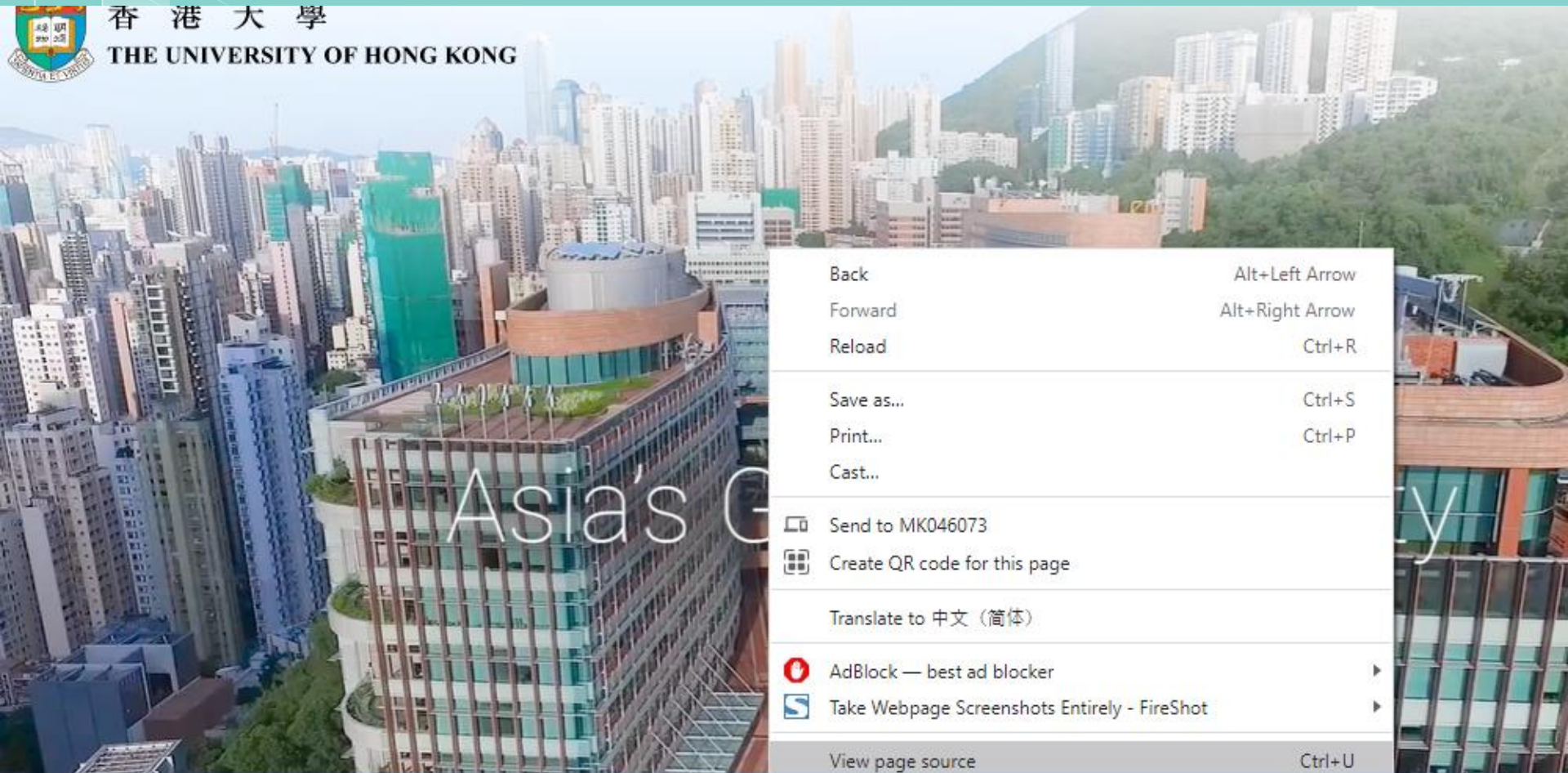
But how to find the HTML of a webpage?





香港大學

THE UNIVERSITY OF HONG KONG



Asia's Gateway

Back	Alt+Left Arrow
Forward	Alt+Right Arrow
Reload	Ctrl+R
Save as...	Ctrl+S
Print...	Ctrl+P
Cast...	
Send to MK046073	
Create QR code for this page	
Translate to 中文 (简体)	
AdBlock — best ad blocker	▶
Take Webpage Screenshots Entirely - FireShot	▶
View page source	Ctrl+U

```
<!DOCTYPE html>
<!--[if lt IE 9]><html class="no-js lte-ie9 lt-ie9lang-en" lang="en"><![endif]-->
<!--[if IE 9]><html class="no-js lte-ie9 ie9lang-en" lang="en"><![endif]-->
<!--[if gt IE 9]><!-->
<html class="no-js" xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta http-equiv="Content-Style-Type" content="text/css" />
  <meta http-equiv="Content-Script-Type" content="text/javascript" />
  <link rel="apple-touch-icon" sizes="180x180" href="/assets/img/apple-touch-icon.png">
  <link rel="icon" type="image/png" href="/assets/img/favicon-32x32.png" sizes="32x32">
  <link rel="icon" type="image/png" href="/assets/img/favicon-16x16.png" sizes="16x16">
  <link rel="manifest" href="/assets/img/manifest.json">
  <link rel="mask-icon" href="/assets/img/safari-pinned-tab.svg" color="#5bbad5">
  <link rel="shortcut icon" href="/assets/img/favicon.ico">
  <meta name="msapplication-config" content="/assets/img/browserconfig.xml">
  <meta name="theme-color" content="#ffffff">
  <noscript><style>
    [data-aos] {
      visibility: visible !important;
      opacity: 1 !important;
      transform: none !important;
    }
  </style></noscript>
```

Learning about HTML

The data you want to scrape appears in certain place of the HTML. For example, suppose that you want to scrape data from the HKU marketing faculty webpage:



Learning about HTML

You can find the name and images of the professors from the HTML file:

```
▶ <noscript>...</noscript>
  
  ▼ <div class="people-info">
    <h5>Dr. Chu (Ivy) DANG</h5> == $0
  </div>
```


Learning about HTML

For example, it provides you with the link to their profile photos:

```

```

Webscraping

Suppose that you want to download the names of each individual marketing faculty, what should you do?

First, you need to get the HTML for the webpage.

Second, you need to analyze the HTML to get the desired information --- **this is much more difficult.**

Webscraping

```
install.packages("rvest")  
library(rvest)
```

```
url =
```

```
"https://www.fbe.hku.hk/people/faculty?pg=1&s  
taff_type=faculty&subject_area=marketing&trac  
k=all"
```

```
webpage = read_html(url, encoding = "UTF-8")  
print(webpage)
```

Webscraping

Now, you get the HTML source file here. The next thing you need to do it to understand the HTML file, which is very challenging.

```
> print(webpage)
{html_document}
<html lang="en-US" prefix="og: https://ogp.me/ns#">
[1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=U ...
[2] <body class="page-template page-template-people-listing page-template ...
```

Webscraping

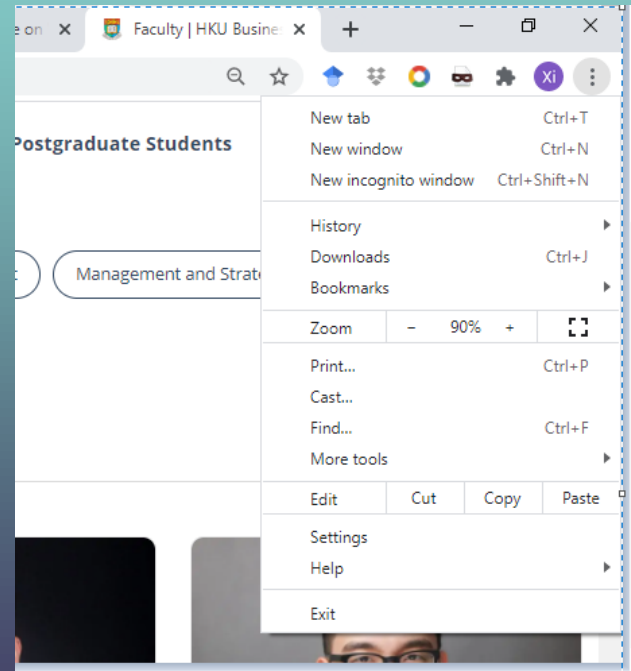
To better understand the HTML code, you are strongly recommended to use **Chrome** as your browser.

Chrome allows you to check the HTML code in a convenient matter.

Check HTML with Chrome

Open the webpage in your Chrome browser.

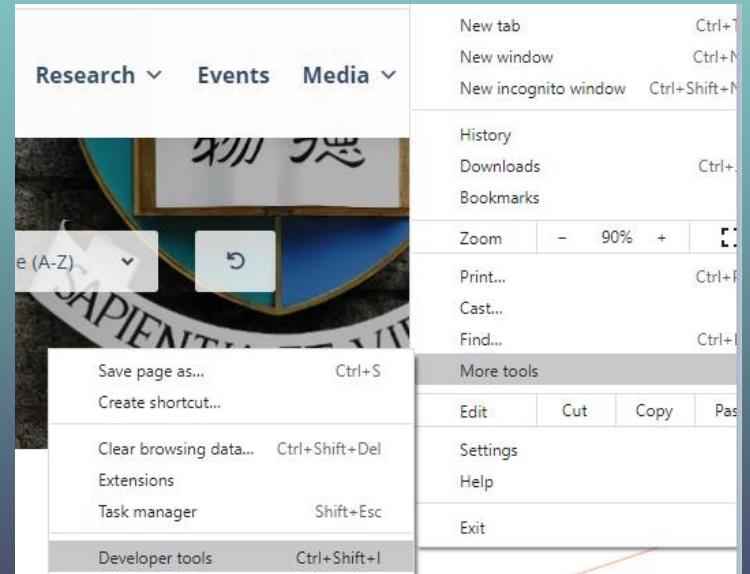
Click the upper right Chrome setting button of your browser and you will be directed here.



Check HTML with Chrome

Choose “More tools” ...

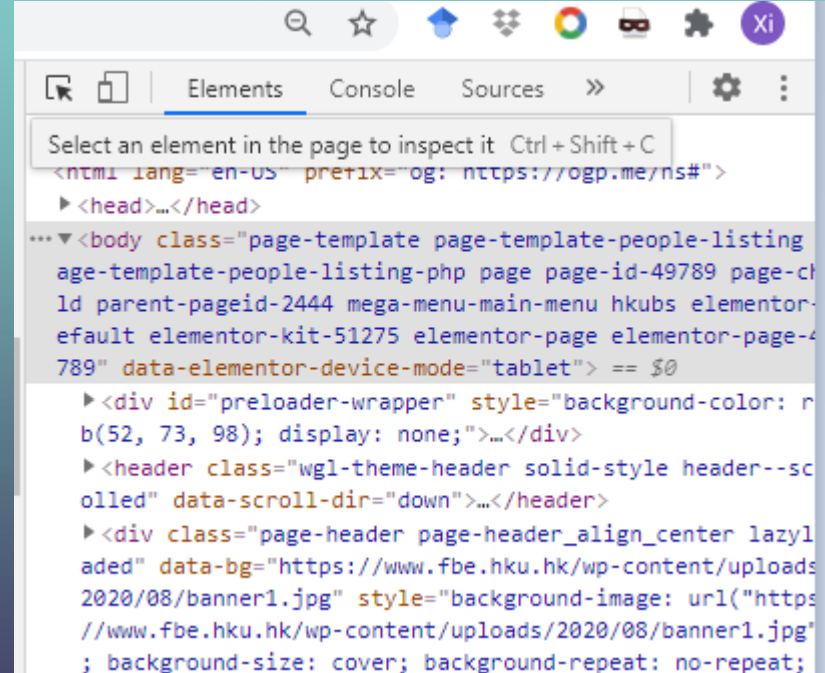
Choose “Developer tools” ...



Check HTML with Chrome

Click the  button and you will get to “select an element in the page to inspect it”.

Alternatively, use “Ctrl + Shift + C”



Check HTML with Chrome

Take Prof. Dang's information as an example.

You can see her name appears here in the HTML code.

But what does this mean?

```
00-ct.jpg" data-src="https://w
ww.fbe.hku.hk/wp-content/uploa
ds/fly-images/11554/FBE_0712_w
eb--scaled-800x800-ct.jpg"
class="attachment-people-thumb
nail lazyloaded" alt="FBE_0712
_web">
  <div class="people-info">
    <h5>Dr. Chu (Ivy) DANG</h5> ==
  </div>
</a>
</div>
</div>
▶<div class="wgl_col-3 people-item">
...</div>
▶<div class="wgl_col-3 people-item">
...</div>
```

Check HTML with Chrome

Take Prof. Dang's information as an example.

You can see her name appears here in the HTML code.

But what does this mean?

```
00-ct.jpg" data-src="https://w
ww.fbe.hku.hk/wp-content/uploa
ds/fly-images/11554/FBE_0712_w
eb--scaled-800x800-ct.jpg"
class="attachment-people-thumb
nail lazyloaded" alt="FBE_0712
_web">
  <div class="people-info">
    <h5>Dr. Chu (Ivy) DANG</h5> ==
  </div>
  </a>
</div>
</div>
▶<div class="wgl_col-3 people-item">
...</div>
▶<div class="wgl_col-3 people-item">
...</div>
```

```
▶ <div class="people-top-tab" id="staff_type">...</div>
▼ <div class="people-main" id="people-listing-container">
  ▶ <div class="filter">...</div>
  ▼ <div class="listing">
    ▼ <div class="row">
      ::before
      ▶ <div class="wgl_col-3 people-item">...</div>
      ▶ <div class="wgl_col-3 people-item">...</div>
      ▶ <div class="wgl_col-3 people-item">...</div>
      ▼ <div class="wgl_col-3 people-item">
        ▼ <div class="people-card fadeInUp animated" data-animate="fadeInUp">
          ▼ <a href="https://www.fbe.hku.hk/people/chu-ivy-dang/" class="el-proce
            ed">
            ▶ <noscript>...</noscript>
              
              ▼ <div class="people-info">
                <h5>Dr. Chu (Ivy) DANG</h5> == $0
              </div>
            </a>
          </div>
        </div>
      </div>
      ▶ <div class="wgl_col-3 people-item">...</div>
      ▶ <div class="wgl_col-3 people-item">...</div>
      ▶ <div class="wgl_col-3 people-item">...</div>
      ▶ <div class="wgl_col-3 people-item">...</div>
```

WHAT IS

HTML

WHAT DOES IT DO &
WHAT IS IT USED FOR?

'Working IT?'

UNDERSTANDING HTML

Here, the name information is within an “h5” node.

And this node belongs to a “div” node.

This “div” node further belongs to another “a” node.

And so on....

We call this is “path”: ...div/div/div/a/div/h5



UNDERSTANDING HTML

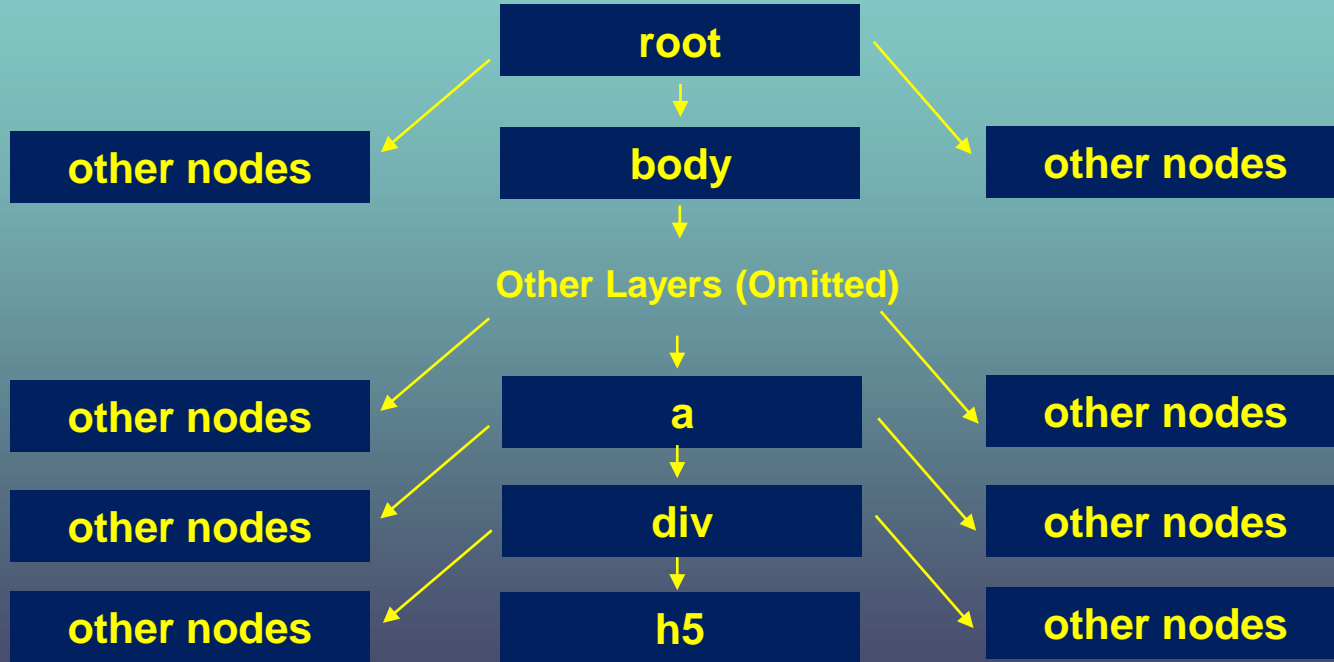
You can see that we have various types of nodes, including “div”, “a”, and “h5”. You may wonder, “what do these types mean?”

Here, these types are called “tag”. For example, a “figure” tag is used to mark up a figure in the HTML language.

For detailed information, check [here](#).



UNDERSTANDING HTML





UNDERSTANDING HTML

This is something like your home address:

We have something like...

Country/Province/City/District/Street/Building/Floor/Room

The path helps us locate nodes and find the content of the nodes.



UNDERSTANDING HTML

However, unlike your home address, here each node does not have its name.

For example, we know it is an “**h5**” node (not a “**div**” node) but there may be multiple “**h5**” nodes.

My building is in a **street** (not an **avenue** or **road**) but there may be multiple **streets** here.

UNDERSTANDING HTML

Let's get all "h5" nodes. This can be done by running this:

```
nodes <- html_nodes (webpage, xpath = '//h5')
```

You can see that in total we have 28 "h5" nodes.

```
print (length (nodes))
```



UNDERSTANDING HTML

We want to make the path more accurate to pin down to the “h5” nodes that we are interested in. That is, we want to remove other unrelated “h5” nodes.

We can do this by putting more restrictions on the path.



UNDERSTANDING HTML

```
page_text <- html_nodes (webpage, xpath =  
' //div[@class="people-info"]/h5 ')
```

Here we restrict the parent of the “h5” node must be a “div” node, and moreover, the parent div node must have a class which is equal to “people-info”.

```
▼<div class="people-info">  
  <h5>Dr. Chu (Ivy) DANG</h5> == $0  
</div>
```

UNDERSTANDING HTML


Now, we only have 16 h5 nodes selected. These are actually all HKU marketing faculties. Let us print their names:

```
nodes <- html_nodes(webpage, xpath =  
'//div[@class="people-info"]/h5')  
for (node in nodes)  
  print(html_text(node))
```



Exercise

Great! You now have a sense of how to scrape data from the web. It is very preliminary, and you will need a lot more exercises. Let us try the following exercise.



Scraping Exercise

In marketing, the most premier academic journal is *Marketing Science*. It covers the latest, most important progress in the marketing community. Let us try to scrape data from it.



Exercise

Each year, Marketing Science publishes 6 issues. We take its first issue in 2021 as an example. The URL for the issue is here:

<https://pubsonline.informs.org/toc/mksc/40/1>

You can see 10 articles in this issue. We want to download the information about these 10 articles.

Exercise

Let's try to scrape the title and author information:

Frontiers: Algorithmic Collusion: Supra-competitive Prices via Independent Algorithms

Karsten T. Hansen, Kanishka Misra , Malleesh M. Pai

Pages: 1–12

Published Online: January 8, 2021

<https://doi.org/10.1287/mksc.2020.1276>

[Preview Abstract](#) ▾

[First Page](#) | [PDF \(1353 KB\)](#) | [References](#) | [Permissions](#)

Exercise

First, let us scrape the titles. We must understand the corresponding HTML code to scrape the data.

```
▼ <h5 class="issue-item_title">  
  ▼ <a href="/doi/abs/10.1287/mks  
    c.2020.1276" title="Frontiers:  
    Algorithmic Collusion: Supra-co  
    mpetitive Prices via Independen  
    t Algorithms"> == $0  
    "Frontiers: Algorithmic  
    Collusion: Supra-competitive  
    Prices via Independent  
    Algorithms"  
  </a>  
</h5>
```

Exercise

```
library(rvest)
url =
  "https://pubsonline.informs.org/toc/mksc/40/1"
webpage = read_html(url, encoding = "UTF-8")
nodes <- html_nodes(webpage, xpath =
  '//h5[@class="issue-item__title"]/a')
for (node in nodes)
  print(html_text(node))
```

Exercise



Now, we are done!

```
[1] "Frontiers: Algorithmic Collusion: Supra-competitive Prices via Independent Algorithms"  
[1] "Frontiers: Moment Marketing: Measuring Dynamics in Cross-Channel Ad Effectiveness"  
[1] "The Effect of Home-Sharing on House Prices and Rents: Evidence from Airbnb"  
[1] "The Impact of Coupons on the Visit-to-Purchase Funnel"
```

Exercise Continued...

Now, let us try to scrape the author names from the *Marketing Science* website. For example, this article is written by Jia Liu and Shawndra Hill.

Frontiers: Moment Marketing: Measuring Dynamics in Cross-Channel Ad Effectiveness

Jia Liu , Shawndra Hill 

Pages: 13–22

Published Online: January 12, 2021

Exercise Continued...

Again, we read the HTML code to understand the path and then decide how to reach the nodes and scrape data.

```
▼<div class="issue-item">
  <div class="badges"></div>
  ▶<h5 class="issue-item__title">
  ...</h5>
  ▼<ul class="rlist--inline loa"
  aria-label="author">
    <a class="entryAuthor linkable h1Fld-ContribAuthor"
    href="/author/Liu%2C+Jia">
    Jia Liu </a> == $0
    ▶<a href="https://orcid.org/000-0002-0279-724X">...</a>
    ", "
    <a class="entryAuthor linkable h1Fld-ContribAuthor"
    href="/author/Hill%2C+Shawndra">Shawndra Hill </a>
    ▶<a href="https://orcid.org/000-0003-1980-727X">...</a>
  </ul>
```

Exercise Continued...



```
nodes <- html_nodes(webpage, xpath =  
'//a[@class="entryAuthor linkable h1Fld-  
ContribAuthor"]')  
print(length(nodes))  
for (node in nodes)  
  print(html_text(node))
```

Exercise Continued...

Yes, we can now print the authors for all articles.

However, this is another issue: We may want to the authors for each article, not the authors for all articles. That is, we want to know that the following paper is written by these specific two authors, not anyone else.

Frontiers: Moment Marketing: Measuring Dynamics in Cross-Channel Ad Effectiveness

Jia Liu , Shawndra Hill 

Exercise Continued...

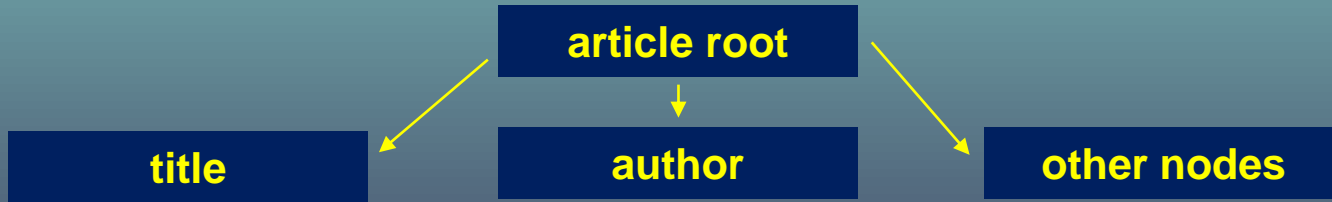
So, let us try to do the following:

For each article, we first print its title, and then we print its authors. We want to get the following output:

(Title) **Frontiers: Moment Marketing: Measuring Dynamics in Cross-Channel Ad Effectiveness**
(Authors) **Jia Liu, Shawndra Hill**

Exercise Continued...

The idea is as follows. Each article starts from the same node, with title and author names being its descendants.





We can first locate the root for the article and then only print its title and author (not the authors of other articles).

div.issue-item 490.66 × 263

Go to Section

Frontiers: Moment Marketing: Measuring Dynamics in Cross-Channel Ad Effectiveness

Jia Liu  Shawndra Hill 

Pages: 13–22

Published Online: January 12, 2021

<https://doi.org/10.1287/mksc.2020.1277>

[First Page](#) | [PDF \(1194 KB\)](#) | [References](#) | [Permissions](#)

Check@
CityULib

[Preview Abstract](#) ▾

Pages 1-191, iii

EDITOR-IN-CHIEF
K. Sudhir

Quicklinks & Resources

- [Special Issues](#)
- [Online Databases](#)
- [Press](#)

```
</div>
<div class="issue-item"> ==
  <div class="badges"></div>
  <h5 class="issue-item_tit
e">...</h5>
  <ul class="rlist--inline l
a" aria-label="author">...</ul>
  <div class="rlist--inline :
parator toc-item_detail">...
</div>
  <p>...</p>
  <div class="toc-item_foot
r">...</div>
</div>
<div class="issue-item">...
</div>
<div class="issue-item">...
```

... icko_side-content div.table-of-content div.issue-item

Styles Computed Layout Event Listeners >>

Exercise Continued...

Here, each article starts from a same div node with class “issue-item”. In this sense, we can start from these nodes for our articles. Then, we search within each of these nodes for title and authors.

```
article_nodes <- html_nodes (webpage, xpath =  
'//div[@class="issue-item"]')  
print (length (article_nodes))
```

Exercise Continued...

```
for (article in article_nodes)
{
  titles <- html_nodes(article, xpath =
'./h5[@class="issue-item__title"]/a')
  print(html_text(titles[1]))
}
```

Exercise Continued...

Here, there are two major differences.

First, we start the path from the root of each article (i.e., we start from each **article** instead of the **webpage**).

Second, we use `“./”` instead of `“/”` in the path. Here, `“./”` means a path starting from the local root instead of the whole webpage.

Exercise Continued...

```
for (article in article_nodes)
{
  titles <- html_nodes(article, xpath =
  './h5[@class="issue-item__title"]')
  print(html_text(titles[1]))
  authors <- html_nodes(article, xpath =
  './a[@class="entryAuthor linkable h1Fld-
  ContribAuthor"]')
  for (author in authors)
    print(html_text(author))
}
```

Exercise #2

Now, let us visit the MIT Open course website here:
<https://ocw.mit.edu/courses/most-visited-courses/>

MIT COURSE#	COURSE TITLE	LEVEL
6.0001	Introduction to Computer Science and Programming in Python	Undergraduate
18.01SC	Single Variable Calculus	Undergraduate
18.06	Linear Algebra	Undergraduate
6.006	Introduction to Algorithms	Undergraduate
15.S12	Blockchain and Money	Graduate
18.02SC	Multivariable Calculus	Undergraduate



Exercise #2

In this exercise, we attempt to scrape the course code, course name, and course level for each course listed on the website. For example, the information we are scraping for the first course is 6.0001, “Introduction to Computer Science and Programming in Python”, “Undergraduate”.

Try this exercise yourself!



Exercise #2

You can see that each class is represented by a “tr” node.

Interestingly, the 1st, 3rd, 5th ... classes have a class attribute “**odd**”.

For the 2nd, 4th, 6th ... classes, they have a class attribute “**even**”.

Exercise #2

You can use the following code to select the “tr” nodes whose class attribute is either “odd” or “even”:

```
url = "https://ocw.mit.edu/courses/most-visited-courses/"
webpage = read_html(url, encoding = "UTF-8")
course_nodes <- html_nodes(webpage, xpath =
  '//tr[@class="odd" or @class="even"]')
print(length(course_nodes))
```

Exercise #2

Then, we need to analyze the course HTML code to understand how we could extract the course information (e.g., course code and course title).

```
▼<tr class="odd"> == $0
  ▼<td>
    <a rel="coursePreview" class="preview" href="/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016">6.0001</a>
  </td>
  ▼<td>
    <a rel="coursePreview" class="preview" href="/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016">Introduction to Computer Science and Programming in Python</a>
  </td>
  ▼<td>
    <a rel="coursePreview" class="preview" href="/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016">Undergraduate</a>
  </td>
</tr>
```

Exercise #2

You can see that, interestingly, the course code, course title, and course level are all represented by an “a” node.

Moreover, they share the same “rel” attribute (`coursePreview`) and the same “class” attribute (`preview`).

This means we can extract all information with the same code (good news)!


Exercise #2

```
for (course in course_nodes){  
  info <- html_nodes(course, xpath =  
  './a[@rel="coursePreview"]')  
  print(html_text(info[1]))  
  print(html_text(info[2]))  
  print(html_text(info[3]))  
  writeLines("")  
}
```



Exercise #2

Note that each course has a link. That is, you can click the course information and you will be directed to the specific course's website. Now, we want to get these links. What should we do?



Exercise #2

The link information is embedded here. This is, it is in the “href” attribute of the “a” node, and we need to read the attribute value.

```
▼<tr class="odd">
***
  ▼<td == $0
    <a rel="coursePreview" class="preview" href="/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016">6.0001</a>
  
```

<https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-programming-in-python-fall-2016>

Exercise #2

```
for (course in course_nodes){  
  info <- html_nodes(course, xpath =  
'./a[@rel="coursePreview"]')  
  courselink <- html_attr(info[1], "href")  
  print(html_text(info[1]))  
  print(html_text(info[2]))  
  print(html_text(info[3]))  
  print(courselink)  
  writeLines("")}
```

Exercise #2

You got something like this:

```
[1] "18.06"  
[1] "Linear Algebra"  
[1] "Undergraduate"  
[1] "/courses/mathematics/18-06-linear-algebra-spring-2010"
```

But the real URL is “<https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/>”



Exercise #2

This is because the webpage has its base URL “<https://ocw.mit.edu>”

You need to add this base to the scraped URL to be able to get the complete URL of each course website.





Exercise #2

This is because the webpage has its base URL “<https://ocw.mit.edu>”

You need to add this base to the scraped URL to be able to get the complete URL of each course website.



Exercise #2

```
for (course in course_nodes){  
  info <- html_nodes(course, xpath =  
'./a[@rel="coursePreview"]')  
  courselink <- html_attr(info[1], "href")  
  print(html_text(info[1]))  
  print(html_text(info[2]))  
  print(html_text(info[3]))  
  print(paste0("https://ocw.mit.edu", courselink))  
  writeLines("") }  
1.
```



Exercise #2

But why should we care about the link?

The link is super important!

For example, you can get the 20 links on the first page.

Then, using the links, you can next crawl the content of these 20 courses, and so on.

