WEEK 2: WEB DATA COLLECTION 2

SECU0050

BENNETT KLEINBERG

23 JAN 2020

Data Science for Crime Scientists

WEEK 2: WEB DATA COLLECTION 2

TODAY

- more HTML
- basic browser simulation

THREE ELEMENTS OF A WEBPAGE

1. Structure
2. Behaviour
3. Style

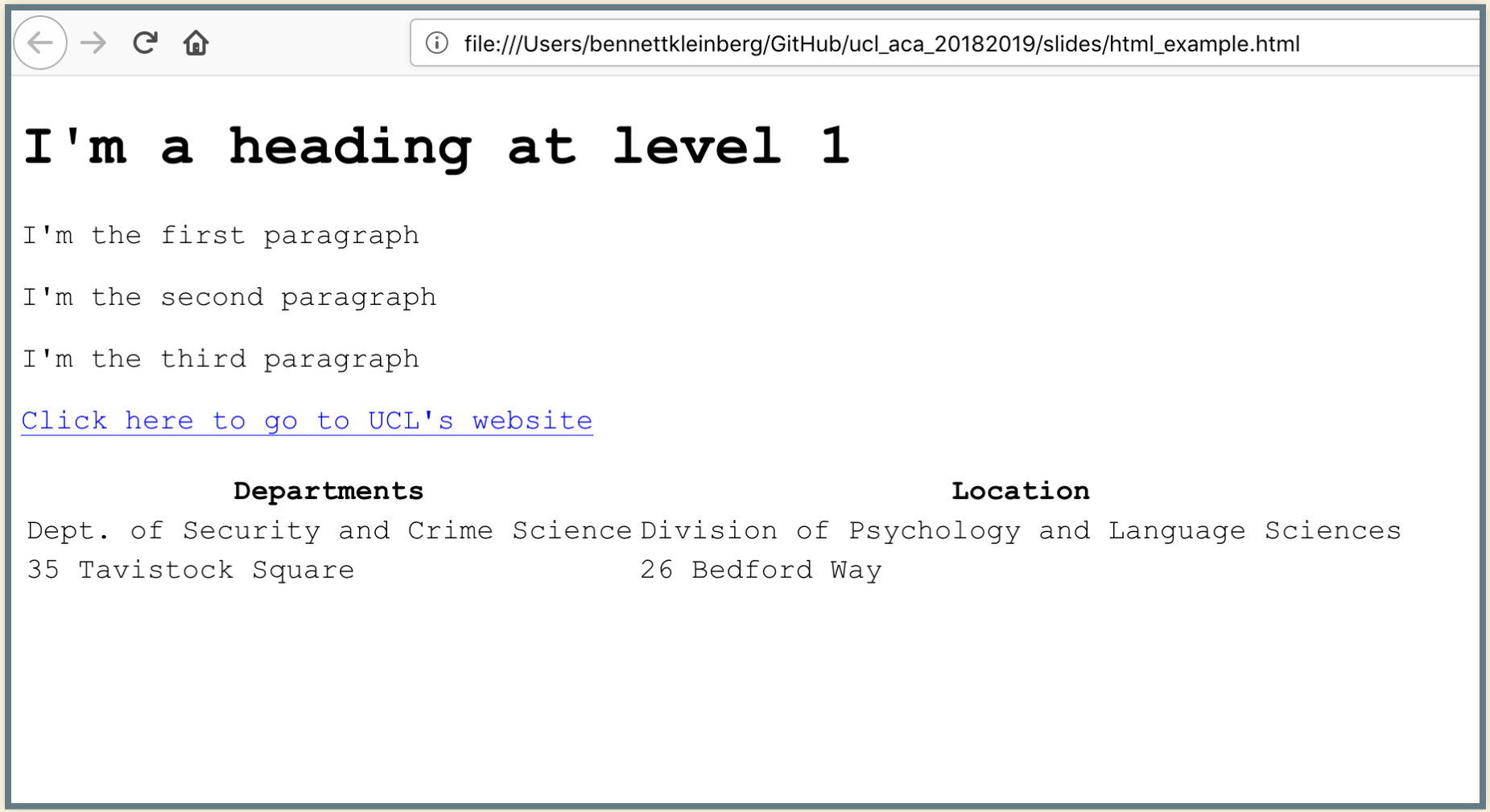
RAW HTML ARCHITECTURE

```
<!DOCTYPE html>
<html>
<body>

HERE COMES THE VISIBLE PART!!

</body>
</html>
```

TABLES

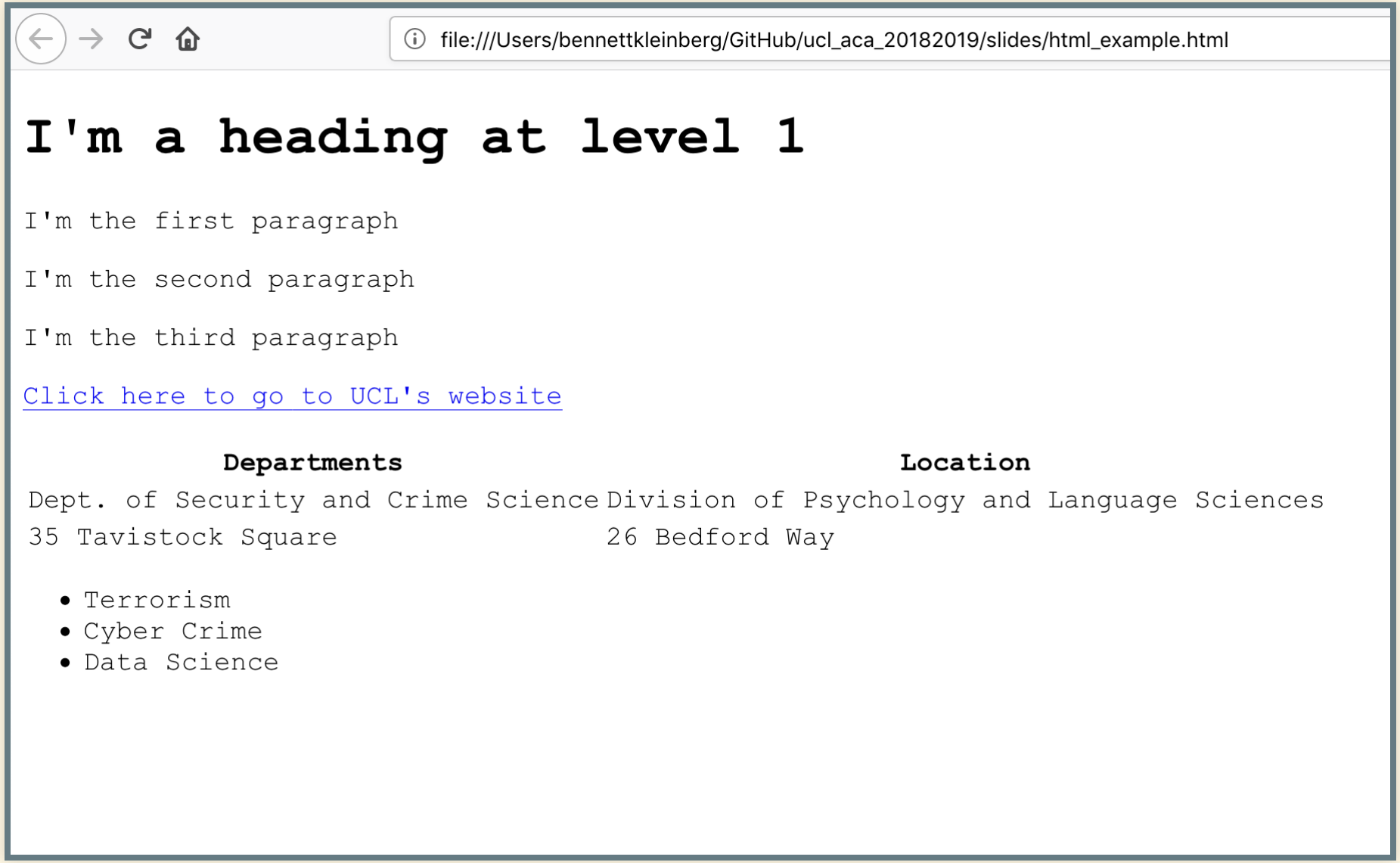
```
<table>
  <tr>
    <th>Departments</th>
    <th>Location</th>
  </tr>
  <tr>
    <td>Dept. of Security and Crime Science</td>
    <td>Division of Psychology and Language Sciences</td>
  </tr>
  <tr>
    <td>35 Tavistock Square</td>
    <td>26 Bedford Way</td>
  </tr>
</table>
```

HTML <TABLE>...</TABLE>



LISTS

```
<ul>
  <li>Terrorism</li>
  <li>Cyber Crime</li>
  <li>Data Science</li>
</ul>
```


file:///Users/bennettkleinberg/GitHub/ucl_aca_20182019/slides/html_example.html
I'm a heading at level 1
I'm the first paragraph
I'm the second paragraph
I'm the third paragraph
Click here to go to UCL's website
Departments
Location
Dept. of Security and Crime Science
35 Tavistock Square
Division of Psychology and Language Sciences
26 Bedford Way
Terrorism
Cyber Crime
Data Science

IDENTIFYING ELEMENTS: IDS

Elements (can) have IDs:

```
<p id='paragraph1'>This is a paragraph</p>
<img id='ucl_image' src="../img/ucl.jpg">
```

Same for tables, links, etc.

Every element can have an ID.

You need unique IDs! Two elements cannot have the same ID.

IDENTIFYING ELEMENTS: CLASSES

Common elements (can) have CLASSES:

```
<p id="paragraph1" class="paragraph_class">I am the first paragraph</p>
<p class="paragraph_class">I am the second paragraph</p>
<p class="paragraph_class">I am the third paragraph</p>
```

Multiple elements can have the same class.

WEBSCRAPING IN PRACTICE

- FBI’s missing persons
- Dodgy exotic animals trading page (tutorial)

FBI'S MISSING PERSONS

https://www.fbi.gov/wanted/kidnap

How could we explore the target page?

AIMS

1. Getting a list of all names
2. Storing the bio information
3. Downloading the poster

GETTING STARTED

Set up your workspace first:

```
library(rvest)
```

```
## Loading required package: xml2
```

```
target_url = 'https://www.fbi.gov/wanted/kidnap'
```

1. GETTING A LIST OF ALL NAMES

Access the full html page (snapshot-mode):

```
target_page = read_html(target_url)
target_page
```

```
## {xml_document}
## <html lang="en" data-gridsystem="bs3">
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charse
## [2] <body id="visual-portal-wrapper" class="  portaltype-folder site-l
```

1. GET A LIST OF ALL NAMES

Key here: look for the `<h3>` heading with class `title`:

```
all_titles = target_page %>%
  html_nodes('h3.title')

#note: equivalent to "html_nodes(target_page, 'h3.title')"
```

```
head(summary(all_titles))
```

```
##      Length Class    Mode
## [1,] 2      xml_node list
## [2,] 2      xml_node list
## [3,] 2      xml_node list
## [4,] 2      xml_node list
## [5,] 2      xml_node list
## [6,] 2      xml_node list
```

```
length(all_titles)
```

```
## [1] 40
```

What do you notice?

TAKING A CLOSER LOOK

```
all_titles[1]
```

```
## {xml_nodeset (1)}
## [1] <h3 class="title">\n<a href="https://www.fbi.gov/wanted/kidnap/lis
```

It's the text of the `<a href=...>` tag.

1. GETTING A LIST OF ALL NAMES

1. Access the full html page `read_html(target_url)`
2. Search all h3 headings with class “title”
   `html_nodes('h3.title')`
3. Find all `<a >` tags (= links) `html_nodes('a')`
4. Extract the text `html_text()`

COMBINED

```
all_names = target_page %>%
  html_nodes('h3.title') %>%
  html_nodes('a') %>%
  html_text()
```

```
all_names
```

```
##  [1] "LISA MARIA SZASZ"
##  [2] "WILLIAM EBENEEZER JONES, JR."
##  [3] "VANESSA MORALES"
##  [4] "FELIX BATISTA"
##  [5] "MARK HIMEBAUGH"
##  [6] "MICHAELA JOY GARECHT"
##  [7] "JANE MCDONALD-CRONE"
##  [8] "JALIEK L. RAINWALKER"
##  [9] "ARANZA MARIA OCHOA LOPEZ"
## [10] "KARLIE LAIN GUSÉ"
## [11] "KARLA RODRIGUEZ"
## [12] "ENRIQUE RIOS"
## [13] "ELIJAH MOORE"
## [14] "LISA IRWIN"
## [15] "DULCE MARIA ALAVEZ"
## [16] "TARA LEIGH CALICO"
## [17] "SHANNA GENELLE PEOPLES"
## [18] "ABBY LYNN PATTERSON"
```

Getting all names: done!

2. STORING THE BIO INFORMATION

We know: there's a table with class `wanted-person-description` that contains the data we want.

But: we need to access each missing person!

For-loops to the rescue…

2. STORING THE BIO INFORMATION

1. *Access the full html page*
2. *Search all h3 headings with class “title”*
3. *Find all* `<a >` *tags (= links)*
4. Extract the ~~text~~ actual link
5. Access that page
6. Extract the table with class `wanted-person-description`

GETTING THE LINK TO EACH PERSON

```
all_persons_links = target_page %>%
  html_nodes('h3.title') %>%
  html_nodes('a') %>%
  html_attr('href')
```

```
head(all_persons_links)
```

```
## [1] "https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz"
## [2] "https://www.fbi.gov/wanted/kidnap/william-ebeneezer-jones-jr"
## [3] "https://www.fbi.gov/wanted/kidnap/vanessa-morales"
## [4] "https://www.fbi.gov/wanted/kidnap/felix-batista"
## [5] "https://www.fbi.gov/wanted/kidnap/mark-himebaugh"
## [6] "https://www.fbi.gov/wanted/kidnap/michaela-joy-garecht"
```

```
length(all_persons_links)
```

```
## [1] 40
```

2. STORING THE BIO INFORMATION

Before you write a loop...

```
lisa_maria = all_persons_links[1]
temp_target_url = lisa_maria
temp_target_page = read_html(temp_target_url)
```

SINGLE-CASE PROOF

```
description = temp_target_page %>%
  html_nodes('table.wanted-person-description') %>%
  html_table()

description
```

```
## [[1]]
##                      X1             X2
## 1 Date(s) of Birth Used July 16, 1962
## 2        Place of Birth           Ohio
## 3                  Hair          Black
## 4                  Eyes          Hazel
## 5                Height           5'7"
## 6                Weight    135 pounds
## 7                   Sex         Female
## 8                  Race          White
```

THE FOR-LOOP

1. do this for each link
2. store it somewhere (easiest: in a list)
3. log progress

```
list_for_data = list()
for(i in all_persons_links){
  print(paste('Accessing:', i))
  temp_target_url = i
  temp_target_page = read_html(temp_target_url)
  description = temp_target_page %>%
    html_nodes('table.wanted-person-description') %>%
    html_table()
  index_of_i = which(i == all_persons_links)
  list_for_data[[index_of_i]] = description
  print('--- NEXT ---')
}
```

```
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/william-ebeneezer-jo
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/vanessa-morales"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/felix-batista"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/mark-himebaugh"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/michaela-joy-garecht
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/jane-mcdonald-crone"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/jaliek-l.-rainwalker
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/aranza-maria-ochoa-l
```

Now we have a list of tables.

Each table contains the details of one missing person:

```
# thirteenth element in the list
list_for_data[[13]]
```

```
## [[1]]
##                        X1                                                  X2
## 1 Date(s) of Birth Used                                   November 3, 1999
## 2                    Hair                                              Black
## 3                    Eyes                                              Brown
## 4                  Height                                             5'11"
## 5                  Weight                                         200 pounds
## 6                     Sex                                               Male
## 7                    Race                                              Black
## 8         Scars and Marks Moore has a burn scar on his left hand.
```

3. DOWNLOADING THE POSTER

Each kidnapped person has a ‘download link’…

```
https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz/@@download.pdf
```

COMPARE THESE TWO

```
https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz/@@download.pdf
```

```
lisa_maria
```

```
## [1] "https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz"
```

Notice something?

3. DOWNLOADING THE POSTER

We can just 'work around' this:

```
download_url_lisa_maria = paste(tolower(lisa_maria), '/@@download.pdf', s
download_url_lisa_maria
```

```
## [1] "https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz/@@download.pdf
```

https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz/@@download.pdf

3. DOWNLOADING THE POSTER

Make use of R's vectorised structure:

```
all_download_links = paste(tolower(all_persons_links), '/@@download.pdf'
head(all_download_links)
```

```
## [1] "https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz/@@download.pdf
## [2] "https://www.fbi.gov/wanted/kidnap/william-ebeneezer-jones-jr/@@do
## [3] "https://www.fbi.gov/wanted/kidnap/vanessa-morales/@@download.pdf"
## [4] "https://www.fbi.gov/wanted/kidnap/felix-batista/@@download.pdf"
## [5] "https://www.fbi.gov/wanted/kidnap/mark-himebaugh/@@download.pdf"
## [6] "https://www.fbi.gov/wanted/kidnap/michaela-joy-garecht/@@downloa
```

Now:

1. access each
2. “download” (= write) the file
    - needs a filename on your computer

CREATE FILENAMES

```
library(stringr)
file_names = paste(all_names, '.pdf', sep="")
head(file_names)
```

```
## [1] "LISA MARIA SZASZ.pdf"                  "WILLIAM EBENEEZER JONES, JR..p
## [3] "VANESSA MORALES.pdf"                   "FELIX BATISTA.pdf"
## [5] "MARK HIMEBAUGH.pdf"                    "MICHAELA JOY GARECHT.pdf"
```

REFINE FILENAMES

```
refined_file_names = tolower(str_replace_all(string = file_names, patter
head(refined_file_names)
```

```
## [1] "lisa_maria_szasz.pdf"                "william_ebeneezer_jones,_jr..
## [3] "vanessa_morales.pdf"                 "felix_batista.pdf"
## [5] "mark_himebaugh.pdf"                  "michaela_joy_garecht.pdf"
```

DOWNLOAD

Download each pdf from the url and use the refined filenames

```
for(i in 1:length(all_download_links)){
  print(paste('Accessing URL: ', all_download_links[i], sep=""))
  download.file(url = all_download_links[i]
                , destfile = refined_file_names[i]
                , mode = "wb")
}
```

```
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/lisa-maria-szas
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/william-ebeneeze
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/vanessa-morales/
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/felix-batista/@@
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/mark-himebaugh/@
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/michaela-joy-gar
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/jane-mcdonald-cr
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/jaliek-l.-rainwa
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/aranza-maria-och
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/karlie-lain-guse
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/karla-rodriguez/
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/enrique-rios/@@d
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/elijah-moore/@@d
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/lisa-irwin/@@dow
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/dulce-maria-alav
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/tara-leigh-calic
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/shanna-genelle-p
```

STATIC VS DYNAMIC WEB-SCRAPING

What is dynamic content?

```
setTimeout(function(){
  alert("This is a delayed alert");
  }, 4000);
```

SCRAPING DYNAMIC WEBPAGES

- problem for the snapshot method
- what if content loads after, say, 5 seconds?
- or if you can only send a request every 5 seconds?

SIMULATING TIMEOUTS

1. we need a way to simulate a browser
2. we need to simulate human user interaction

Enter: **RSelenium**

SETUP

```
library(RSelenium)

#make a connection
selenium_firefox = rsDriver(browser=c("firefox"))

#start a driver
driver = selenium_firefox$client
```

Live demo

BACKUP

```
#set target url
target_url = 'https://www.fbi.gov/wanted/kidnap'

#navigate the driver (= simulated browser) to the target url
driver$navigate(target_url)
```

BACKUP (1)

```
# 1. set wait intervals
list_for_requests = list()

for(i in 1:5){
  parsed_pagesource <- driver$getPageSource()[[1]]
  result <- read_html(parsed_pagesource) %>%
    html_nodes('h3.title') %>%
    html_nodes('a')

  list_for_requests[[i]] = result
  print(paste('Sent request at:', Sys.time(), sep=" "))

  Sys.sleep(5)
}
```

BACKUP (2)

```
# 2. simulate scroll
#navigate the driver (= simulated browser) to the target url
driver$navigate(target_url)

#find the html body
page_body = driver$findElement("css", "body")

#send a scroll command (note that this is a page_down request in Javascri
page_body$sendKeysToElement(list(key = "page_down"))
```

BACKUP (3)

```
# 3. simulate multiple scrolls
#navigate the driver (= simulated browser) to the target url
driver$navigate(target_url)

#find the html body
page_body = driver$findElement("css", "body")

#send multiple scroll commands in a loop
for(i in 1:10){
  page_body$sendKeysToElement(list("key"="page_down"))

  # allow some time for this to happen (here: 3 seconds)
  Sys.sleep(3)
}
```

BACKUP (3 CONT'D)

```
#now access the page source (important: you need to do this through the d
parsed_pagesource <- driver$getPageSource()[[1]]

#now we can scrape from the page after the simulation
full_results <- read_html(parsed_pagesource) %>%
  html_nodes('h3.title') %>%
  html_nodes('a') %>%
  html_attr('href')

length(full_results)
```

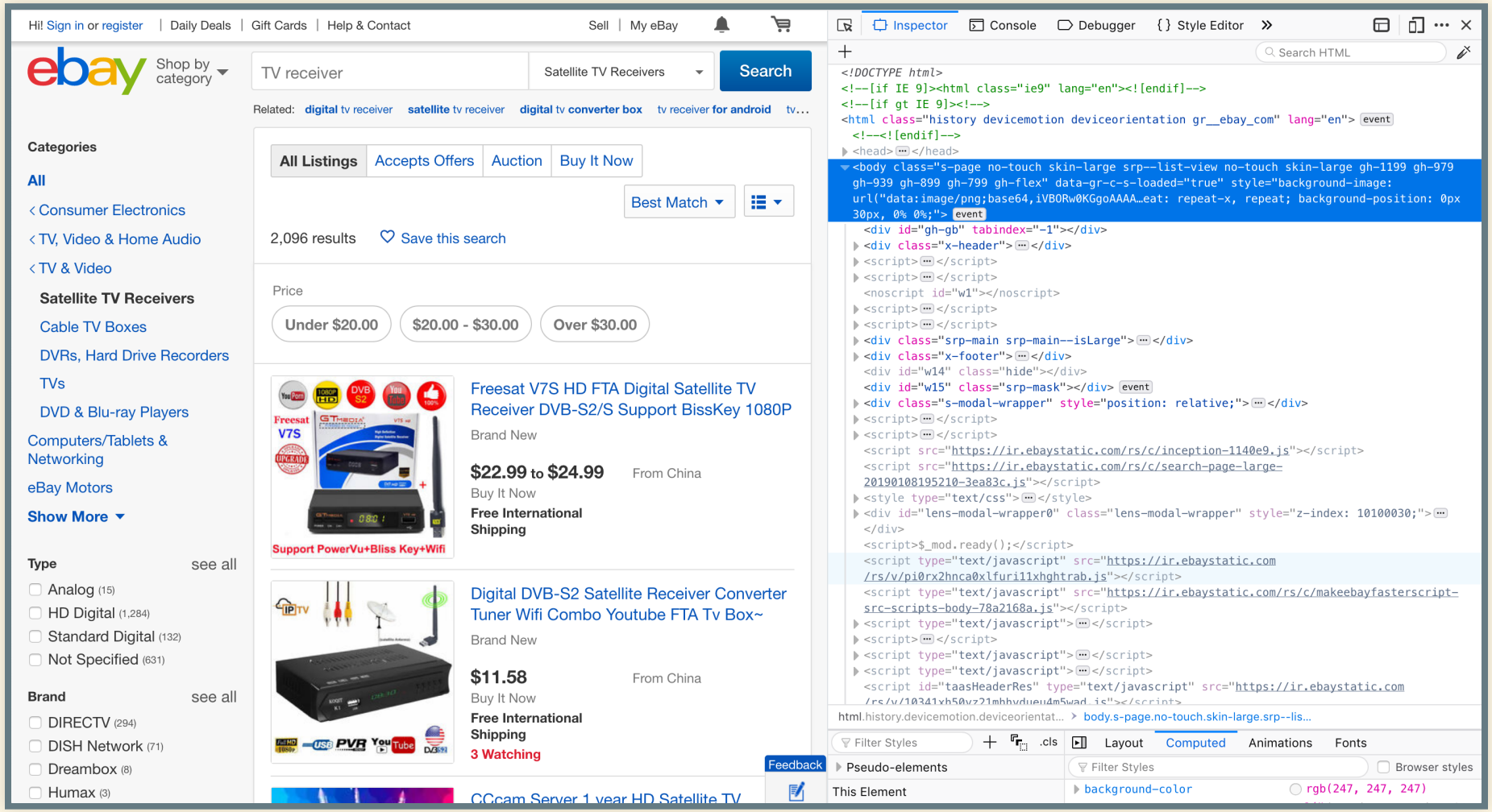
BACKUP (CLOSE)

```
# close the driver and the server
driver$close()

selenium_firefox$server$stop()
```

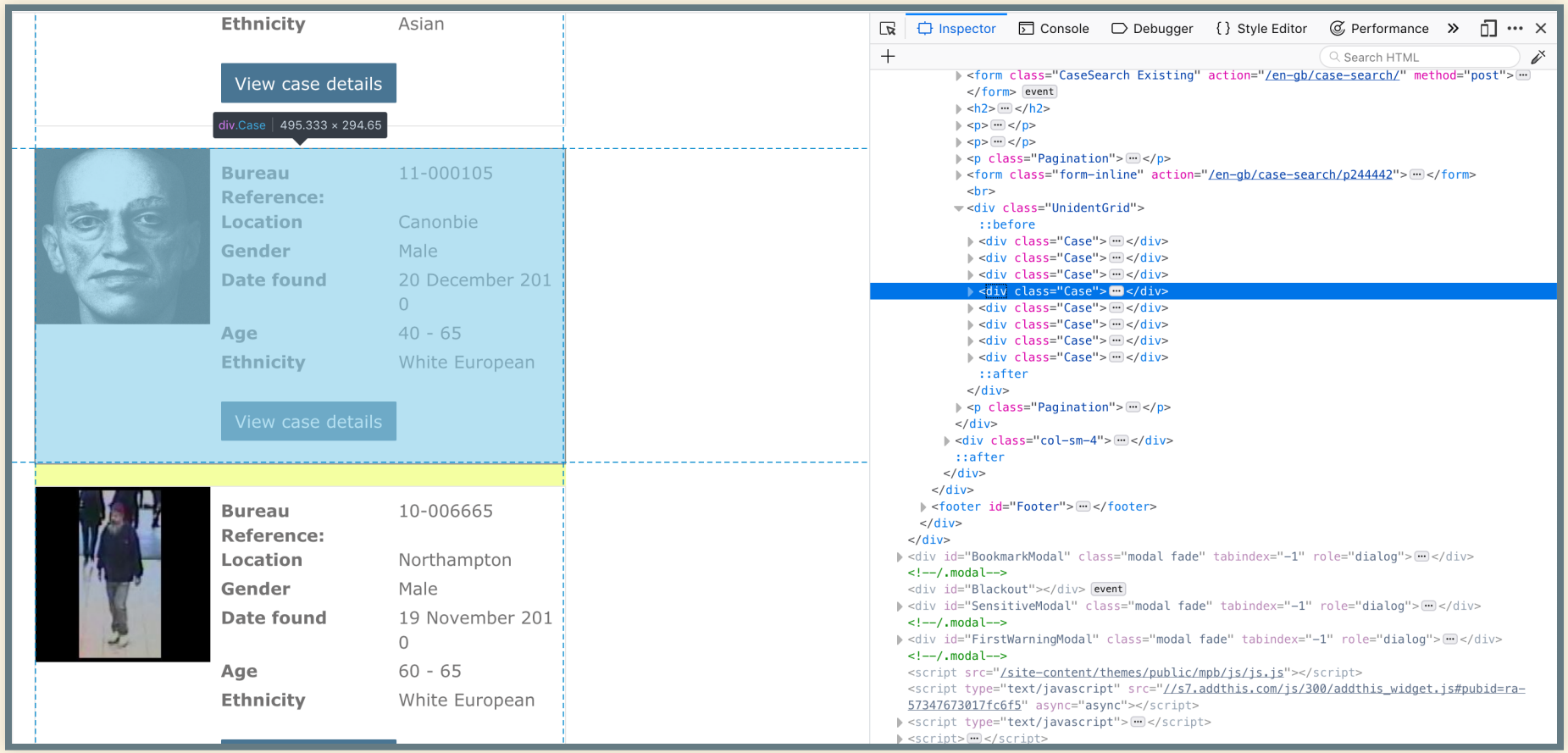
NOTES ON WEBSCRAPING

- highly customisable (= juicy data)
- basically: “anything goes”
- can be unstable/sensitive to html changes

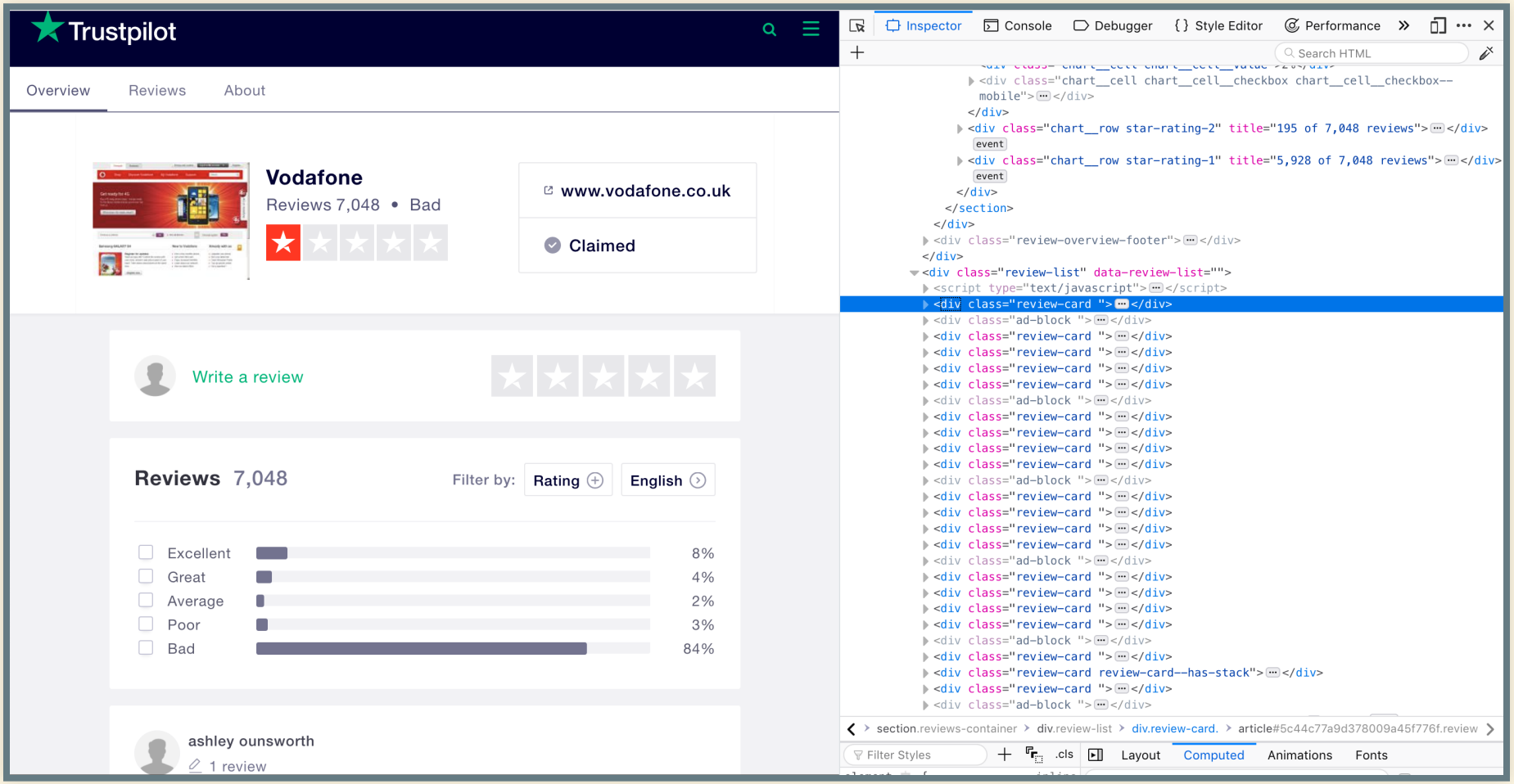
SAME IDEA, DIFFERENT HTML



SAME IDEA, DIFFERENT HTML



SAME IDEA, DIFFERENT HTML



SAME IDEA, DIFFERENT HTML

Phone: (NA) -NA
Email: Email Seller
Location: Michigan
Website: NA

We have available one tame hand raised baby female Kinkajou, perfect as an educational ambassador or pet. We are USDA licensed, serious and educated inquiries only please.

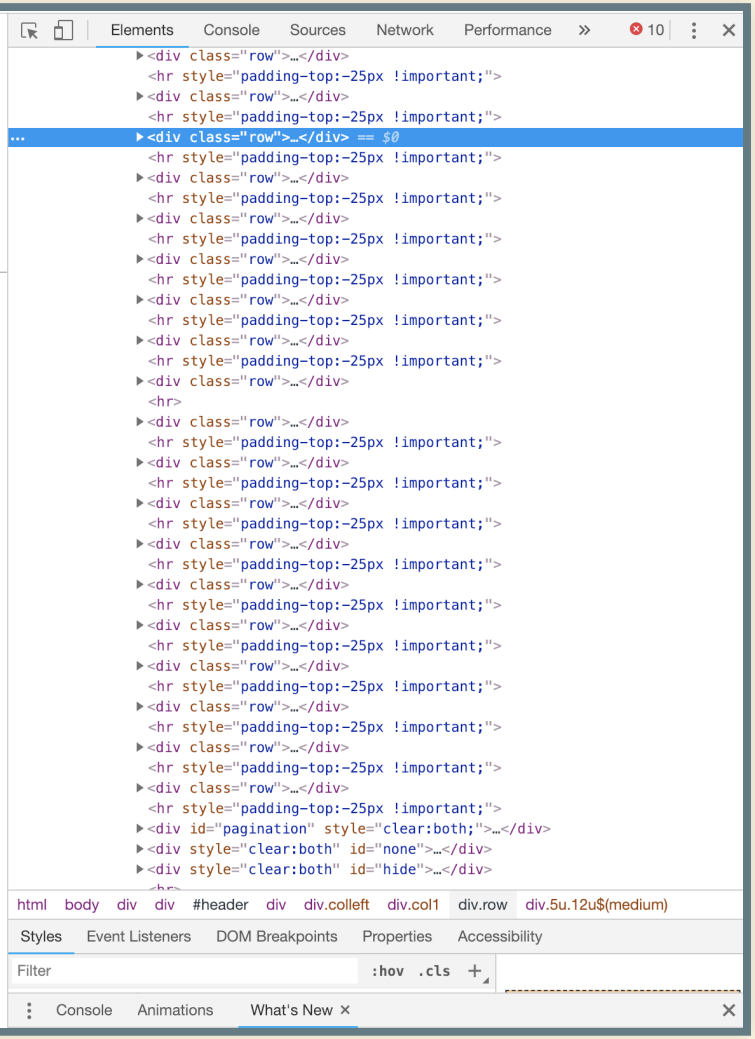
View Details

Adult Kinkajous

Name: CJG EXOTICS View Profile
Posted: 1/20/2019
Phone: (NA) -NA
Email: Email Seller
Location: Michigan
Website: NA

We have available several unrelated adult Kinkajou pairs and trios. These would make an excellent breeding project or zoo exhibit. These are not tame and not pets, we are USDA licensed.

View Details



RECAP

- **Always: problem first, never the method first!**
- **Method follows problem!**
- HTML structure key to webscraping
- Webscraping:
  - understanding the structure of a webpage
  - exploiting that structure for web-scraping
- principle is always the same: understand + exploit the html structure

WHAT'S NEXT?

- Today's tutorial: dynamic scraping of the FBI's website, full pipeline for exotic animal trading forum
- Homework: Rvest tutorials, webscraping practice

Next week: Text Mining 1