DIPARTIMENTO DI MATEMATICA E INFORMATICA

UNIVERSITÀ degli STUDI di CATANIA

Social Media Data Analysis 2023/2024

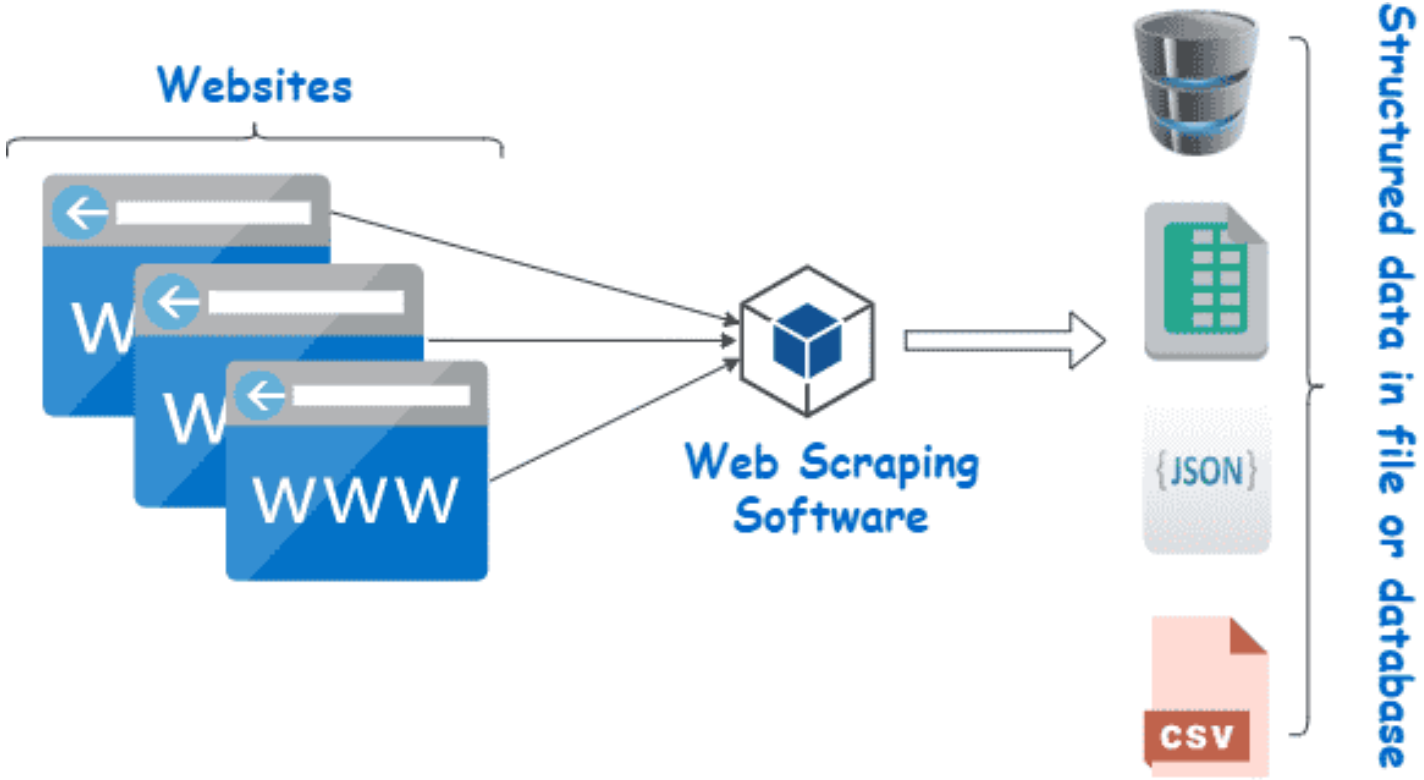
Web Scraping

Francesco Ragusa
francesco.ragusa@unict.it
https://iplab.dmi.unict.it/ragusa/

IMAGE PROCESSING LABORATORY

https://iplab.dmi.unict.it/fpv/

Definition



Web scraping is the process to extract data from a website and export it into a structured format

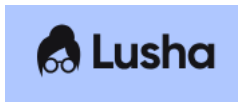
Applications

- Price Comparison

Applications

- Price Comparison

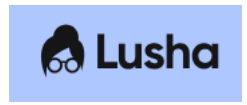
- Email address gathering



Applications

- Price Comparison

- Email address gathering



- Social Media Scraping

Applications

- Price Comparison

- Email address gathering

- Social Media Scraping

- Research and Development

Applications

- Price Comparison

- Email address gathering

- Social Media Scraping

- Research and Development

- Job listings

Is it legal?

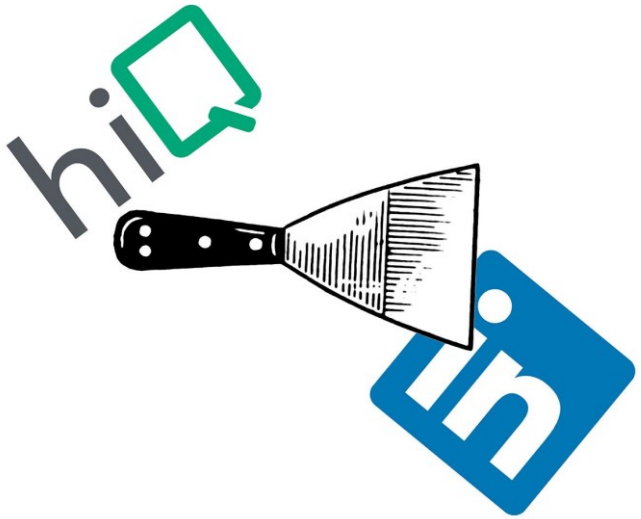
- How do you use the extracted data?
- Are you violating the ‘Terms & Conditions’ statements?
- Are you violating copyright of the extracted data?

Is it legal?

The data displayed by most of the websites are generally accessible to the public

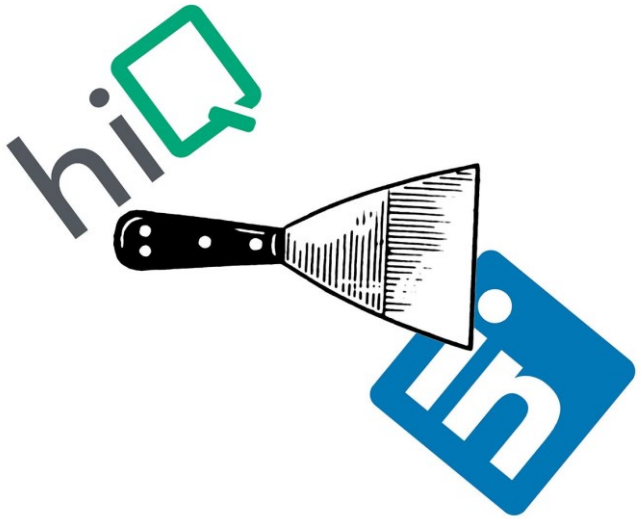
Is it legal?

The data displayed by most of the websites are generally accessible to the public



Is it legal?

The data displayed by most of the websites are generally accessible to the public



POWER VENTURES, INC.

Is it legal?

Is it legal?

Robots.txt

A text file create to instruct web robots (typically search engine robots) how to crawl pages on their website

These crawl instructions are specified by “disallowing” or “allowing” the behavior of certain (or all) user agents

```
User-agent: [user-agent name]Disallow: [URL string not to be crawled]
```

Robots.txt Syntax

• **User-agent:** The specific web crawler to which you're giving crawl instructions (usually a search engine). A list of most user agents can be found: http://www.robotstxt.org/db.html

• **Disallow:** The command used to tell a user-agent not to crawl particular URL. Only one "Disallow:" line is allowed for each URL.

• **Allow** (Only applicable for Googlebot): The command to tell Googlebot it can access a page or subfolder even though its parent page or subfolder may be disallowed.

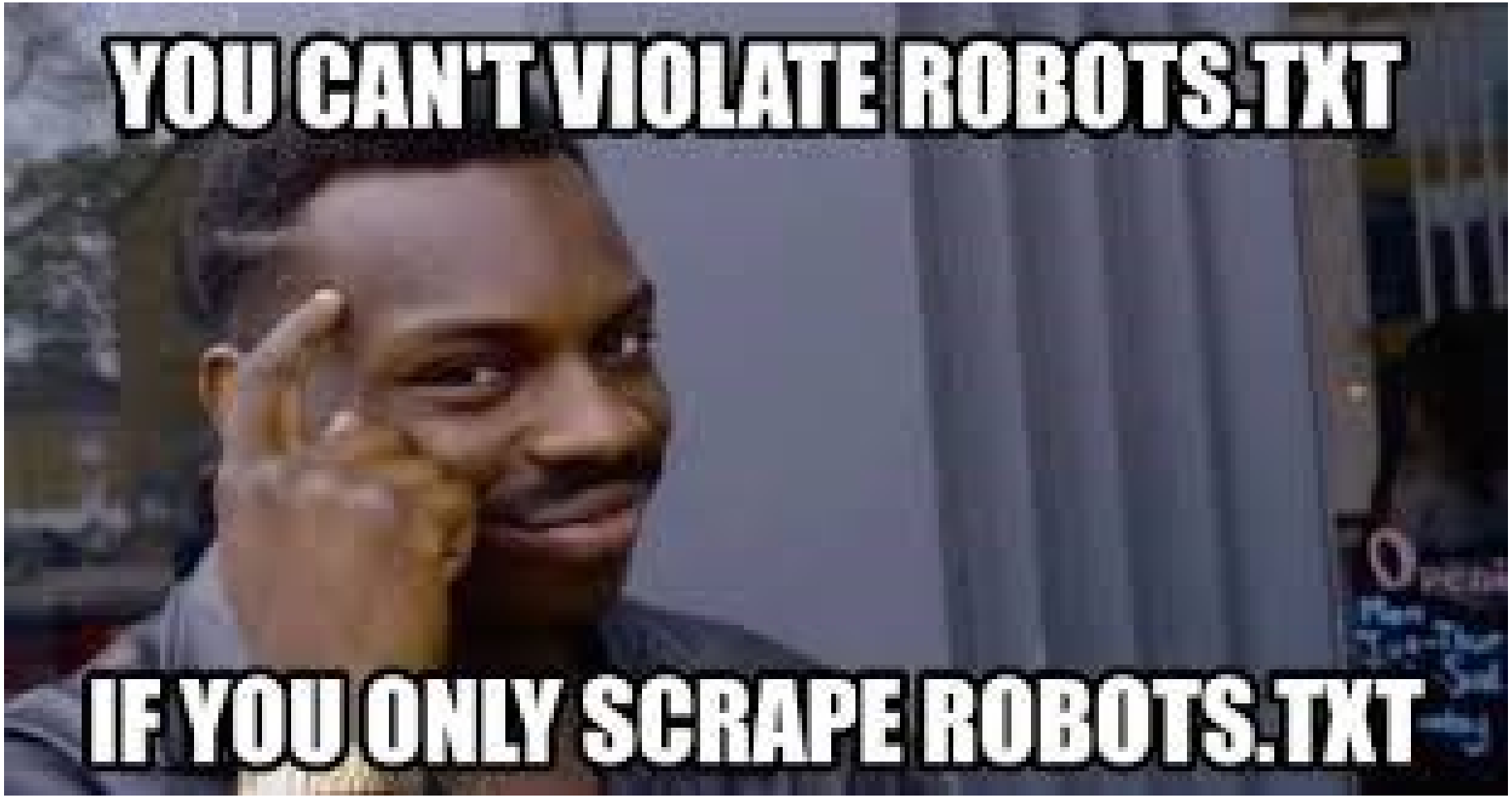
• **Crawl-delay:** How many seconds a crawler should wait before loading and crawling page content.

• **Sitemap:** Used to call out the location of any XML sitemap(s) associated with this URL.

Robots.txt Syntax

```
# Example 1: Block only Googlebot
User-agent: Googlebot
Disallow: /

# Example 2: Block Googlebot and Adsbot
User-agent: Googlebot
User-agent: AdsBot-Google
Disallow: /

# Example 3: Block all crawlers except AdsBot (AdsBot crawlers must be named explicitly)
User-agent: *
Disallow: /
```

Robots.txt Syntax



DOM

The Document Object Model (DOM) is a ***programming interface*** for **HTML(HyperText Markup Language)** and **XML**(Extensible markup language) documents.

It defines the **logical structure** of documents and the way a document is accessed and manipulated.

This **model** allows JavaScript to access the text content and elements of the website **document** as **objects**.

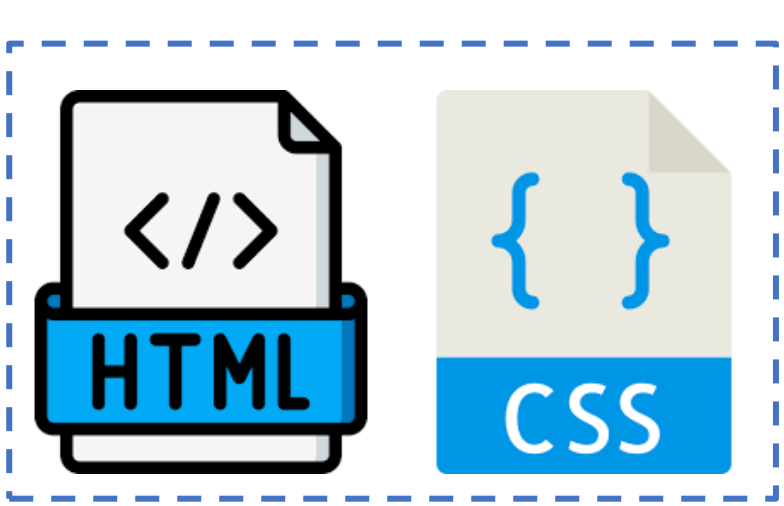
Document Object Model is an API that represents and interacts with HTML or XML documents.

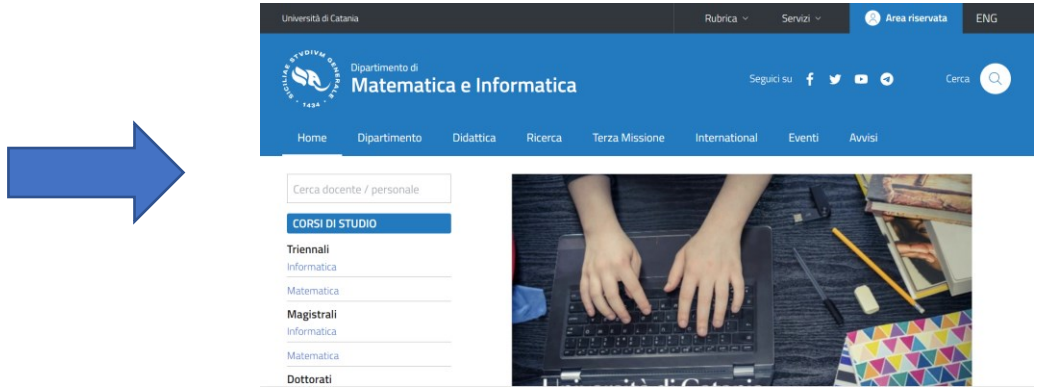
DOM

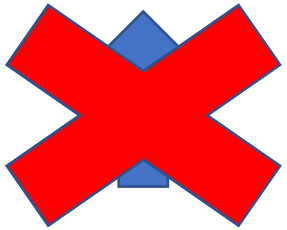
Amazon.it: elettronica, libri, musica, fashion, videogiochi, DVD e tanto altro

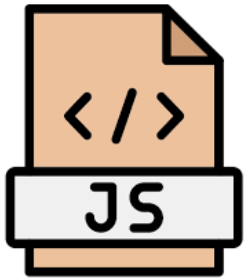
DOM

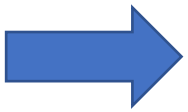
Web Site

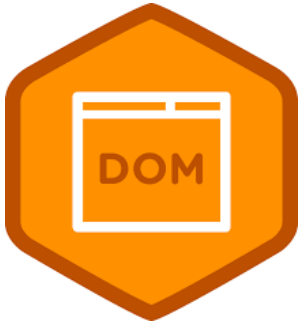

HTML
CSS


Università di Catania
Rubrica
Servizi
Area riservata
ENG
Dipartimento di
Matematica e Informatica
Seguici su
Cerca
Home
Dipartimento
Didattica
Ricerca
Terza Missione
International
Eventi
Avvisi
Cerca docente / personale
CORSI DI STUDIO
Triennali
Informatica
Matematica
Magistrali
Informatica
Matematica
Dottorati

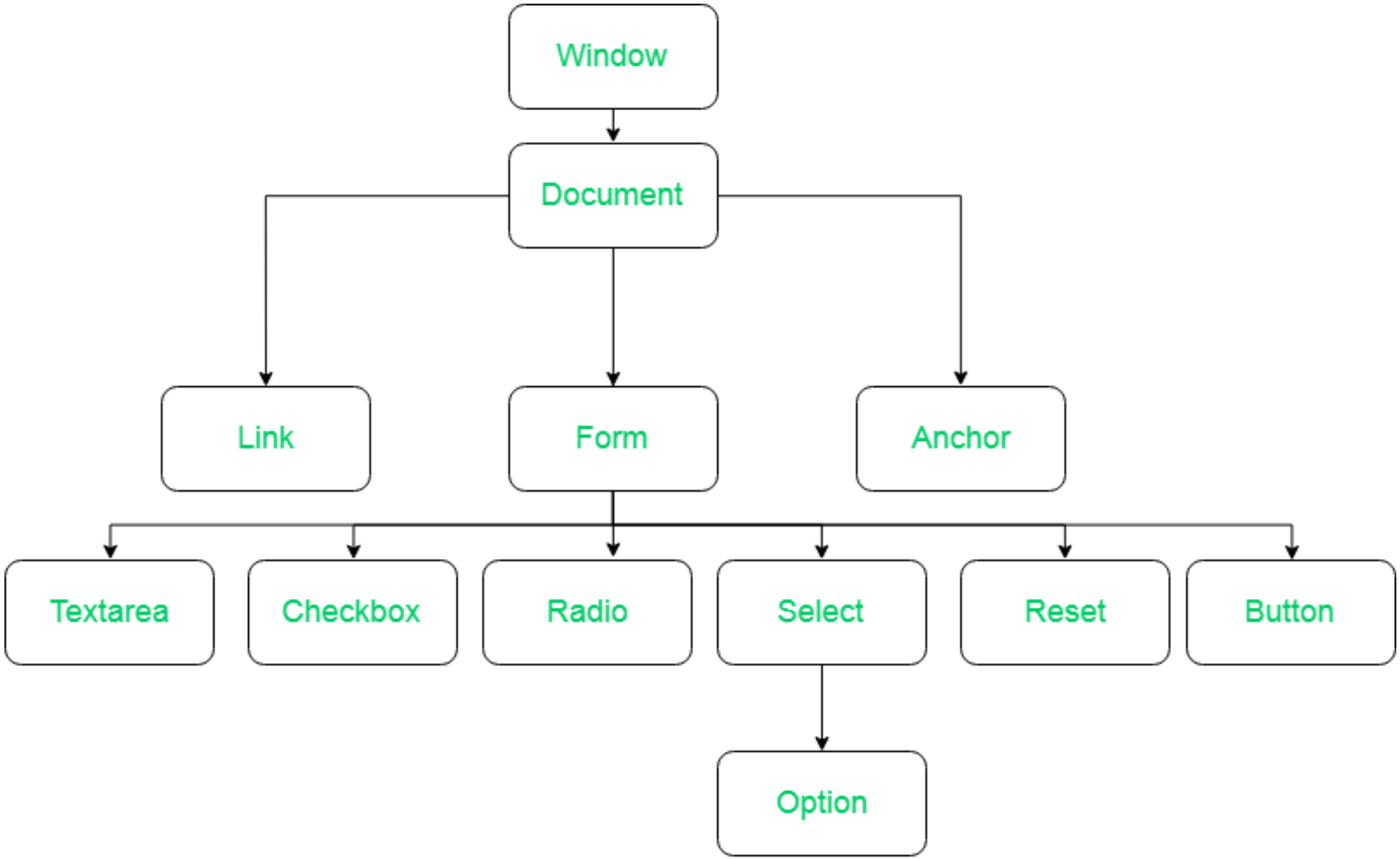



JS

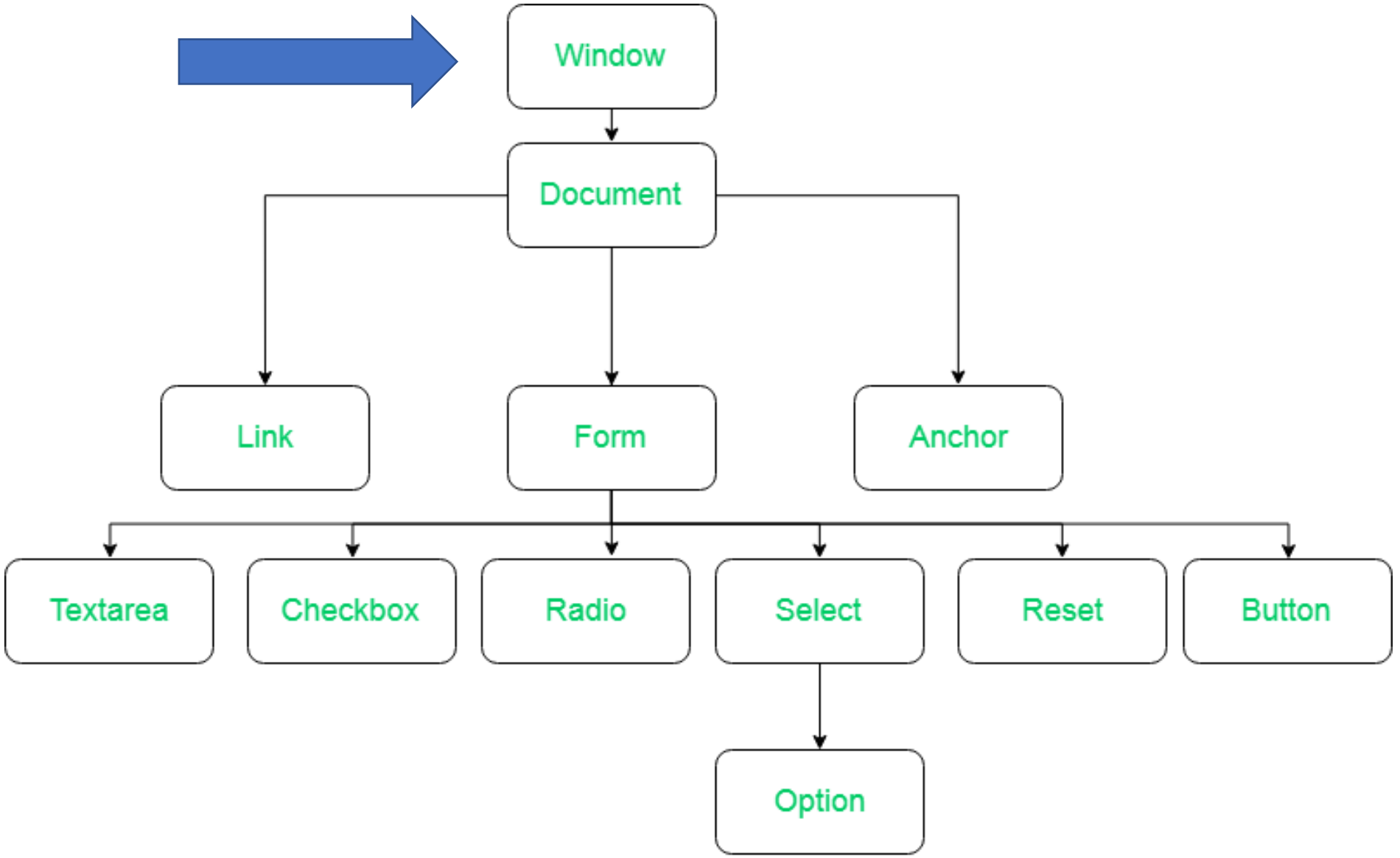



DOM

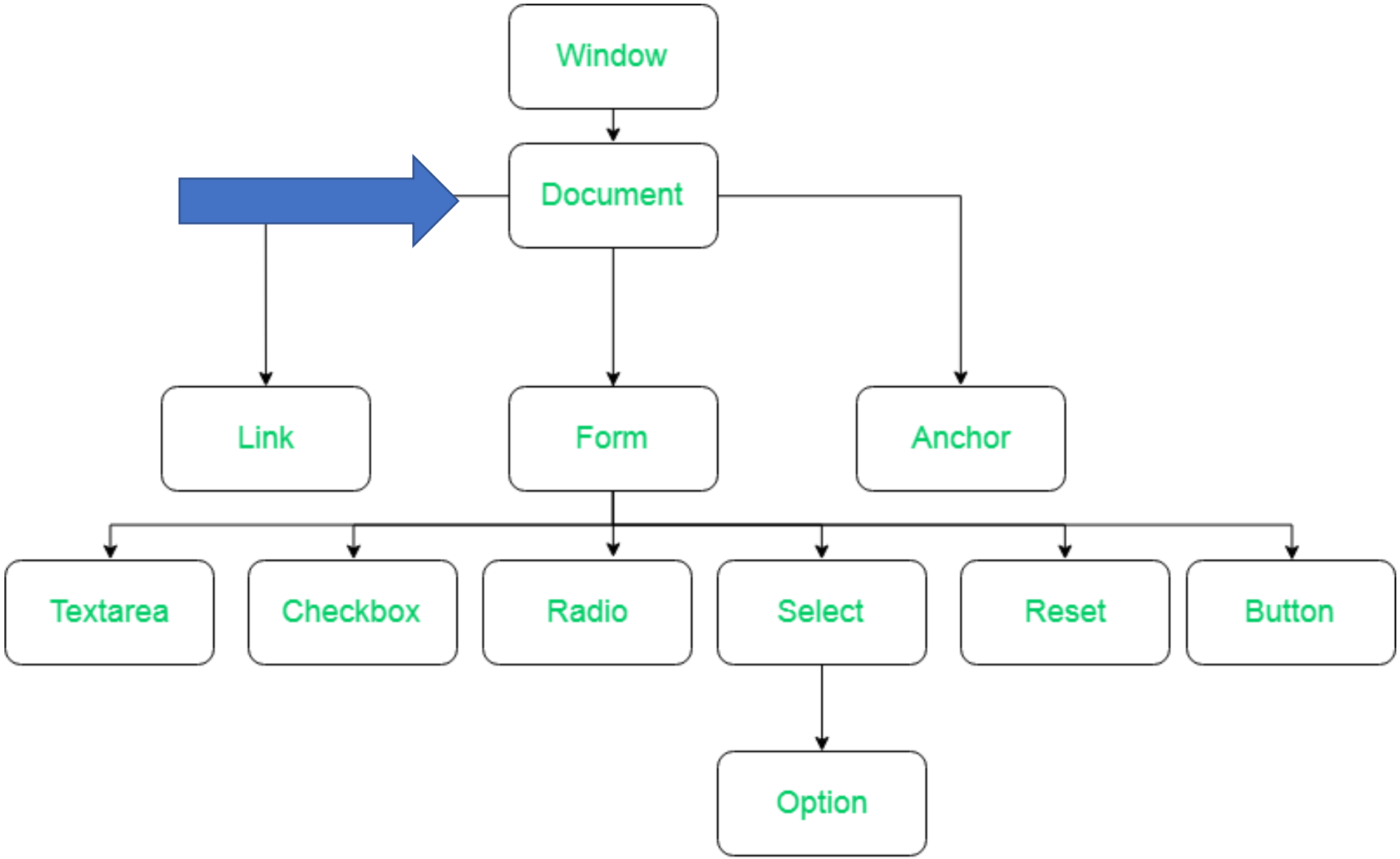
Properties


Window
Document
Link
Form
Anchor
Textarea
Checkbox
Radio
Select
Reset
Button
Option

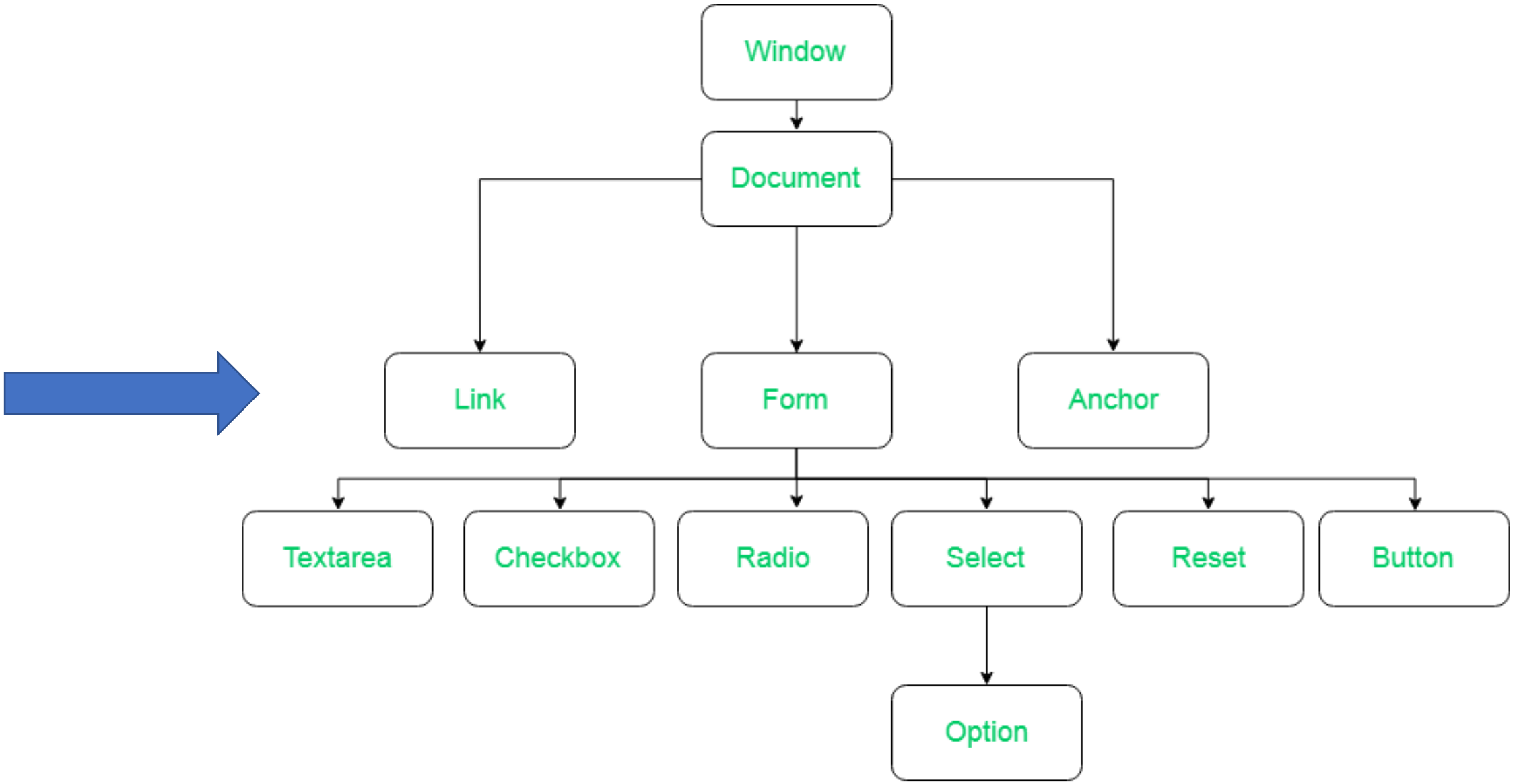
Properties


Window
Document
Link
Form
Anchor
Textarea
Checkbox
Radio
Select
Reset
Button
Option

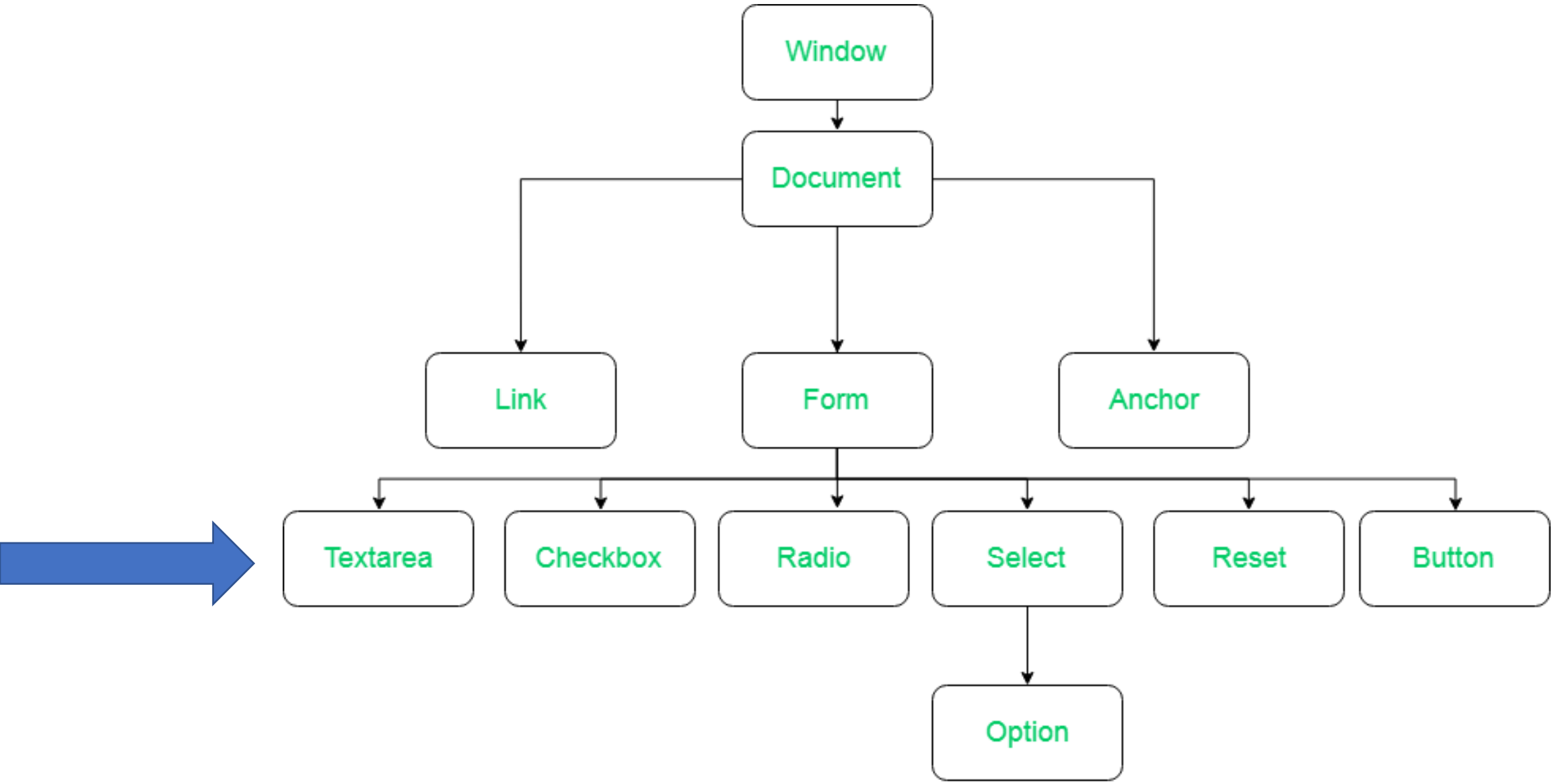
Properties


Window
Document
Link
Form
Anchor
Textarea
Checkbox
Radio
Select
Reset
Button
Option

Properties


Window
Document
Link
Form
Anchor
Textarea
Checkbox
Radio
Select
Reset
Button
Option

Properties


Window
Document
Link
Form
Anchor
Textarea
Checkbox
Radio
Select
Reset
Button
Option

Methods

- **write("string"):** Writes the given string on the document
- **getElementById():** returns the element having the given id value
- **getElementsByName():** returns all the elements having the given name value
- **getElementsByTagName():** returns all the elements having the given tag name
- **getElementsByClassName():** returns all the elements having the given class name

Methods

SMDA Course

This is a lesson on Web Scraping.

This example illustrates the **getElementById** method.

The introduction is: This is a lesson on Web Scraping.

Methods

getElementById()

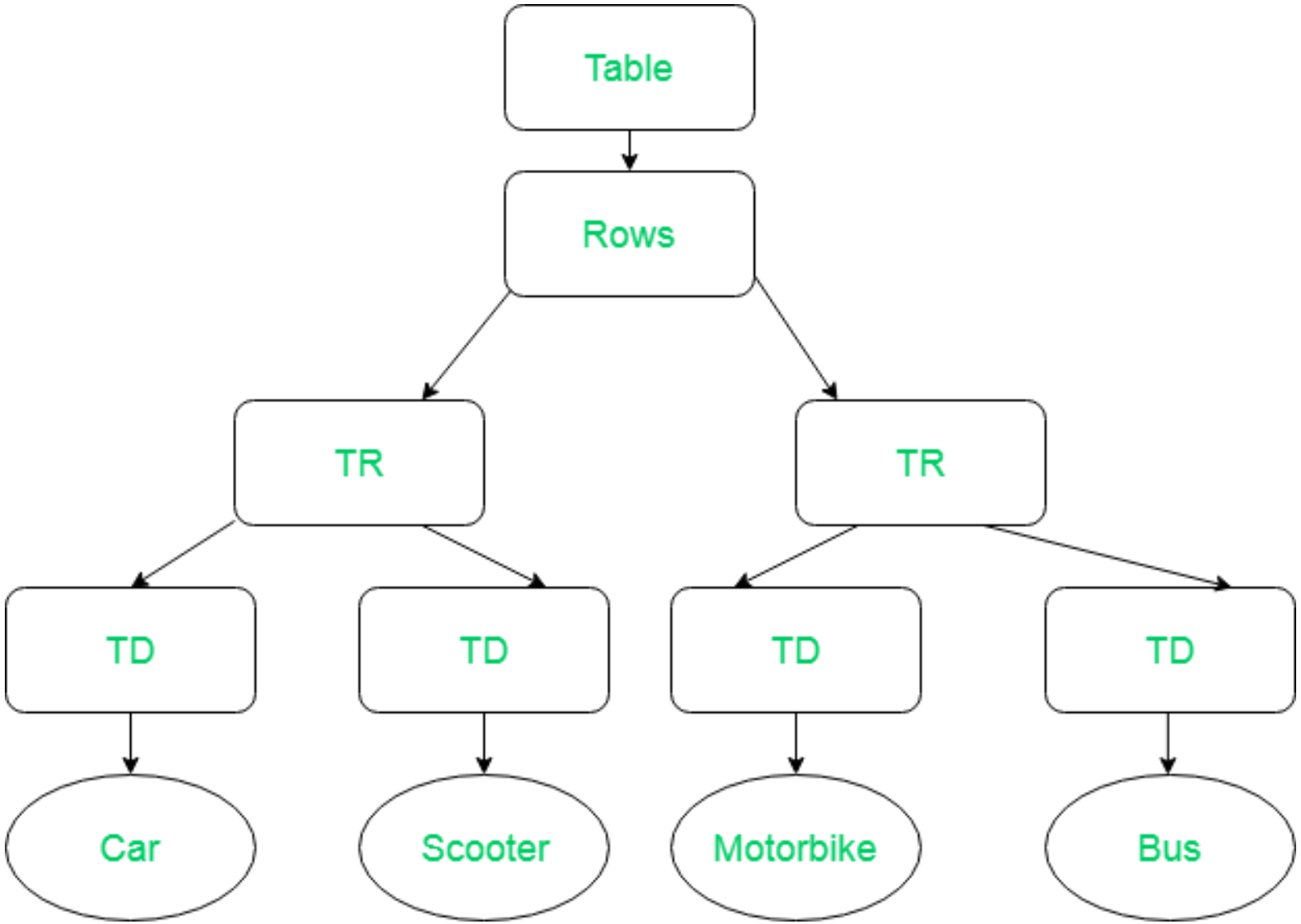
```
<!DOCTYPE html>
<html>

<body>
    <h2>SMDA Course</h2>

    <!-- Finding the HTML Elements by their Id in DOM -->
    <p id="intro">This is a lesson on Web Scraping.</p>
    <p>This example illustrates the <b>getElementById</b> method.</p>
    <p id="demo"></p>
    <script>
        const element = document.getElementById("intro");
        document.getElementById("demo").innerHTML =
        "The introduction is: " + element.innerHTML;
    </script>
</body>

</html>
```

Tree Structure

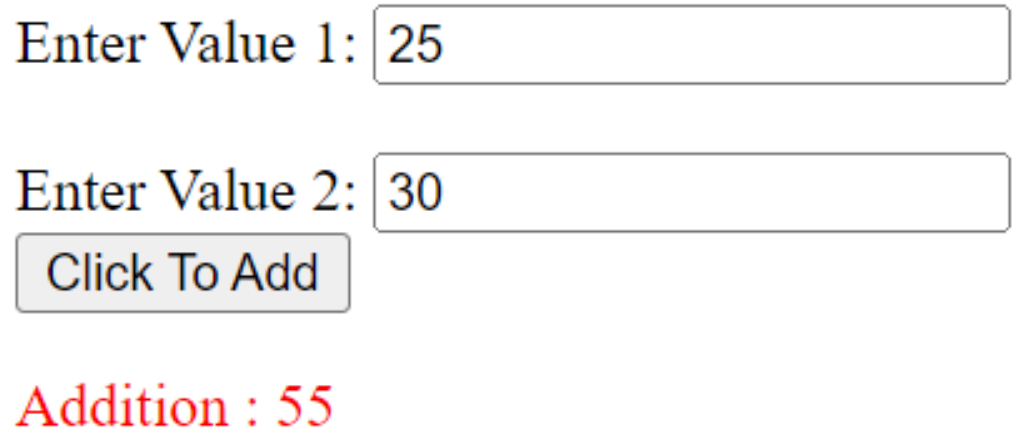
Car	Scooter
MotorBike	Bus

```
<table>
    <ROWS>
        <tr>
            <td>Car</td>
            <td>Scooter</td>
        </tr>
        <tr>
            <td>MotorBike</td>
            <td>Bus</td>
        </tr>
    </ROWS>
</table>
```

Tree Structure



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

<head>
    <title>DOM manipulation</title>
</head>

<body>
    <label>Enter Value 1: </label>
    <input type="text" id="val1" />
    <br />
    <br />
    <label>Enter Value 2: </label>
    <input type=".text" id="val2" />
    <br />
    <button onclick="getAdd()">Click To Add</button>
    <p id="result"></p>
    <script type="text/javascript">
        function getAdd() {

            // Fetch the value of input with id val1
            const num1 = Number(document.getElementById("val1").value);

            // Fetch the value of input with id val2
            const num2 = Number(document.getElementById("val2").value);
            const add = num1 + num2;
            console.log(add);

            // Displays the result in paragraph using dom
            document.getElementById("result").innerHTML = "Addition : " + add;

            // Changes the color of paragraph tag with red
            document.getElementById("result").style.color = "red";
        }
    </script>
</body>

</html>
```



What DOM is not?

The Document Object Model is not used to describe **objects in XML or HTML** whereas the DOM describes XML and HTML documents as objects.

The Document Object Model is not represented by a **set of data structures**; it is an interface that specifies object representation.

The Document Object Model does not show the **criticality of objects** in documents i.e it doesn’t have information about which object in the document is appropriate to the context and which is not.

Web Scraping with Python

Why Python?

Ease of Use: Python programming is simple to code. You do not have to add semi-colons ";" or curly-braces "{}" anywhere.

Large Collection of Libraries: Python has a huge collection of libraries such as Numpy, Matplotlib, Pandas etc.

Duck typing: In Python, you don't have to define datatypes for variables, you can directly use the variables wherever required.

Easily Understandable Syntax: Python syntax is easily understandable mainly because reading a Python code is very similar to reading a statement in English.

Small code, large task: Web scraping is used to save time.

Selenium

Selenium is an open-source automated testing framework used to validate web applications across different browsers and platforms.

It is an executable module that runs a script on a browser instance and hence is also called **headless browser scraping**.

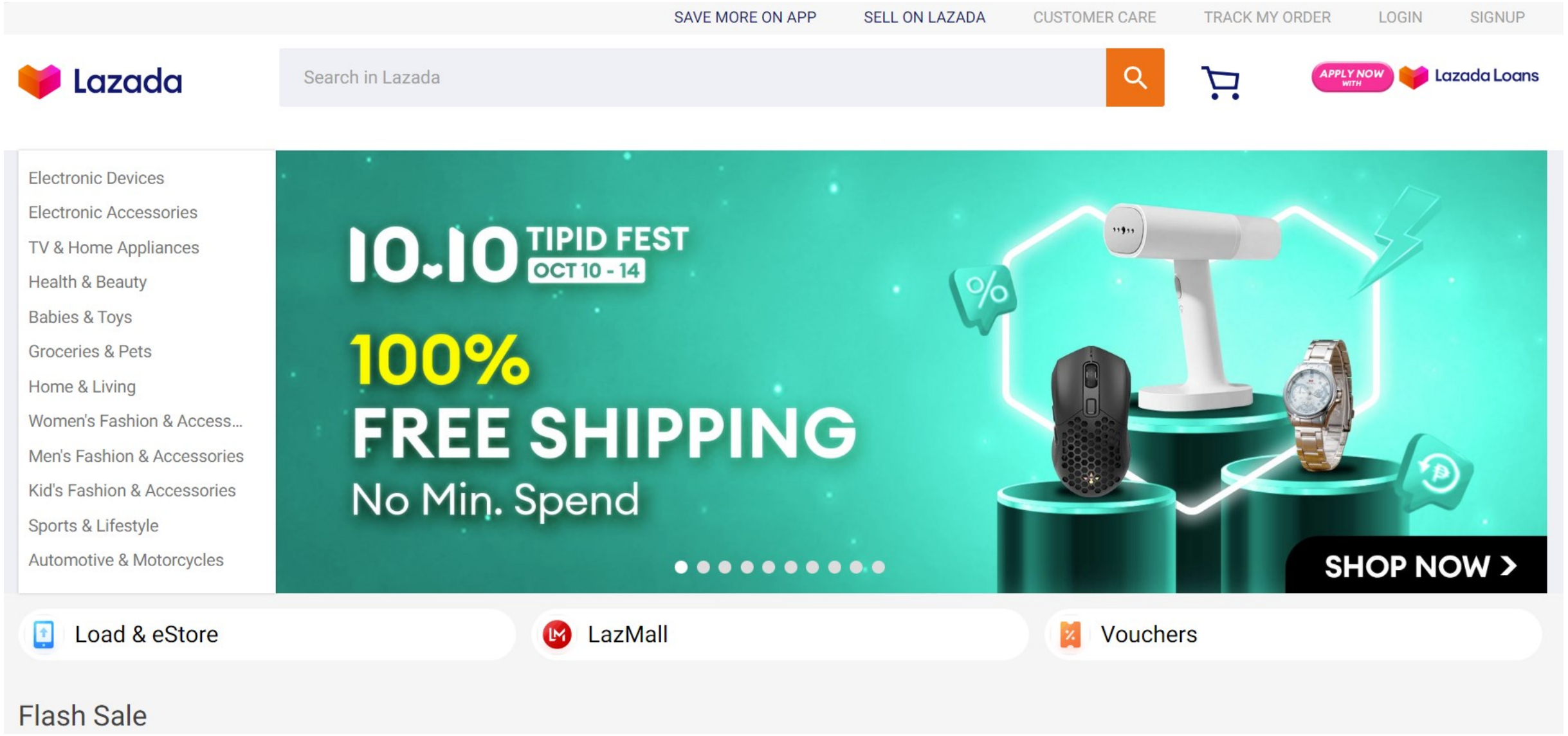
Selenium with Python

Why Selenium?

Today selenium is mainly used for web scraping and automation purposes.

- ***clicking on buttons***
- ***filling forms***
- ***scrolling***
- ***taking a screenshot***

E-commerce Web Scraping



https://www.lazada.com.ph/

E-commerce Web Scraping

```
# Web Scraping
from selenium import webdriver
from selenium.common.exceptions import *
# Data manipulation
import pandas as pd
# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
```

E-commerce Web Scraping

```
webdriver_path = 'C://Users//me//chromedriver.exe' # Enter the file directory of the Chromedriver
Lazada_url = 'https://www.lazada.com.ph'
search_item = 'Nescafe Gold refill 170g'
```

E-commerce Web Scraping

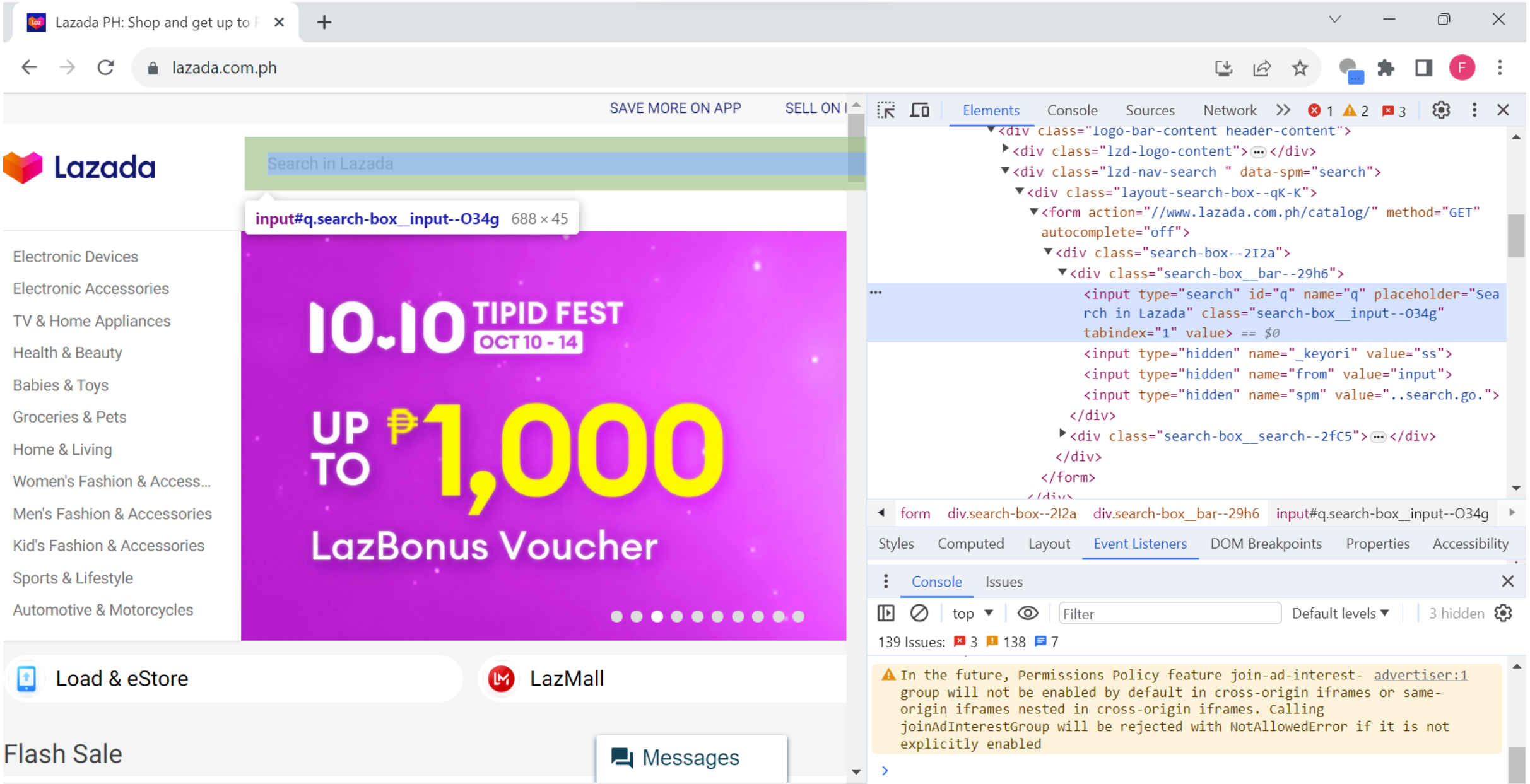
```
# Select custom Chrome options
options = webdriver.ChromeOptions()
options.add_argument('--headless')
options.add_argument('start-maximized')
options.add_argument('disable-infobars')
options.add_argument('--disable-extensions')
# Open the Chrome browser
browser = webdriver.Chrome(webdriver_path, options=options)
browser.get(Lazada_url)
```

E-commerce Web Scraping

But how do you identify which elements to find?

An easy way to do this is to use Chrome’s very own inspect tool

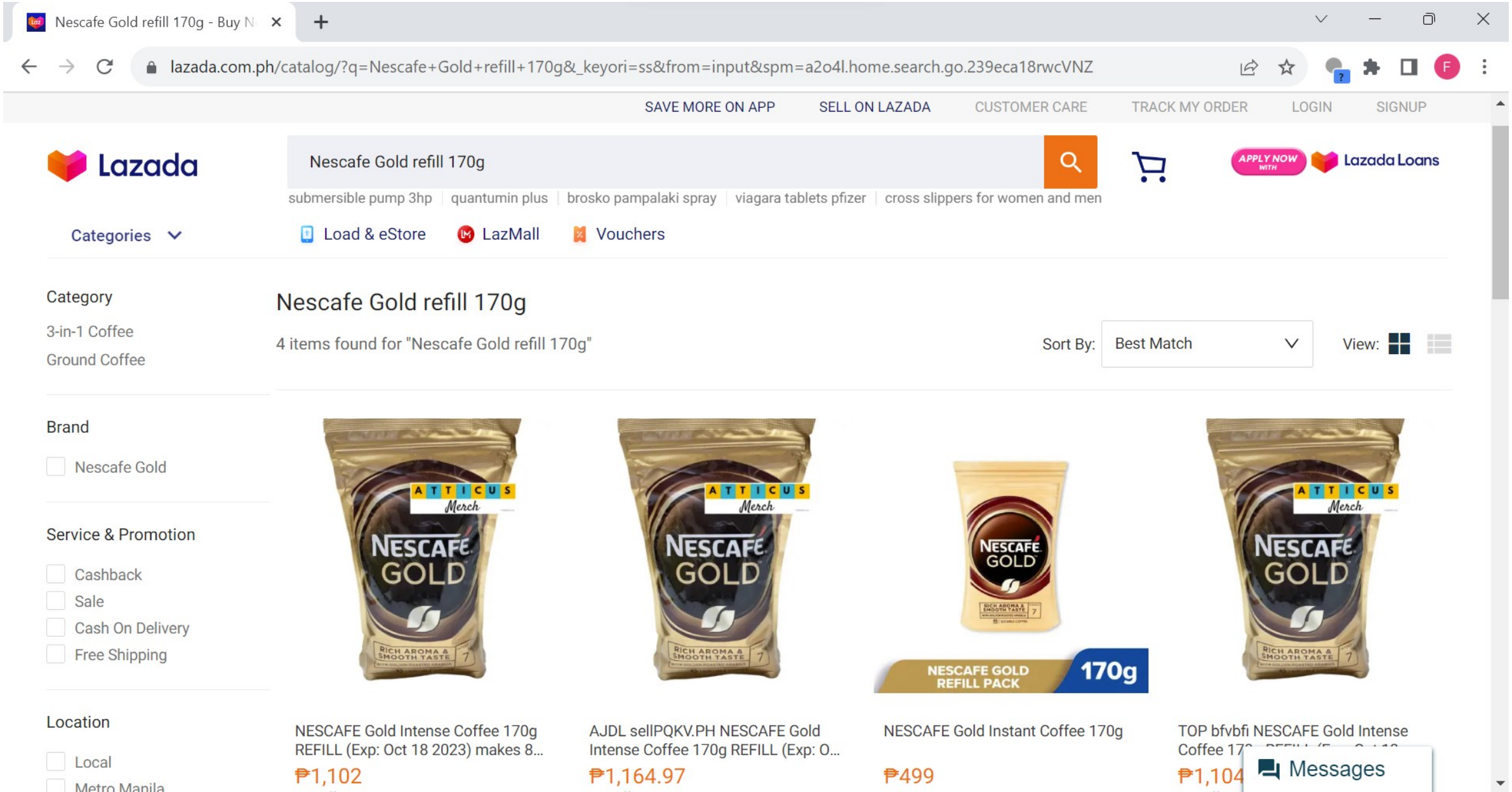
E-commerce Web Scraping



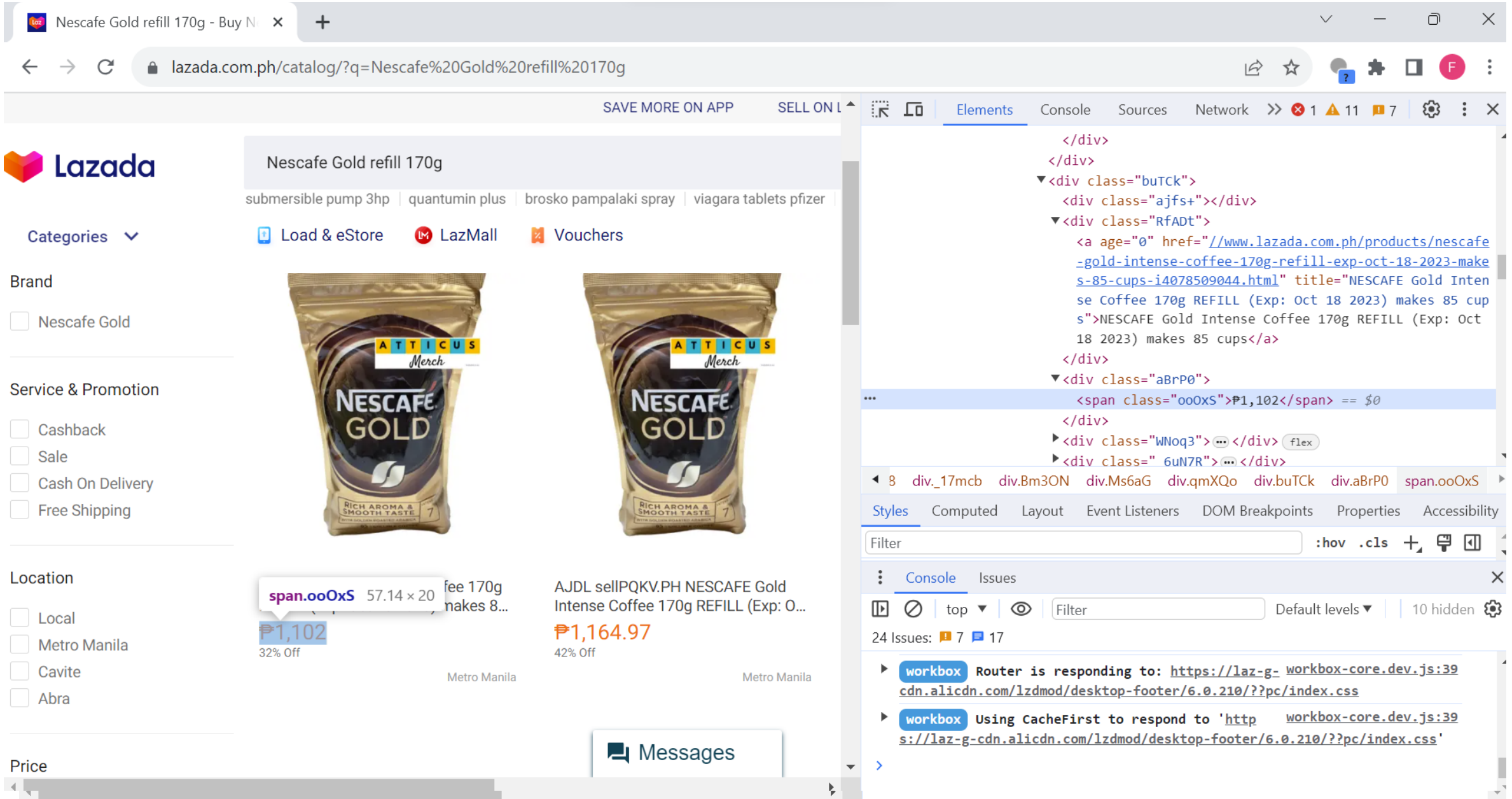
E-commerce Web Scraping

```
search_bar = browser.find_element_by_id('q')
search_bar.send_keys(search_item).submit()
```

E-commerce Web Scraping



E-commerce Web Scraping



E-commerce Web Scraping

```
item_titles = browser.find_elements_by_class_name('RfADt')
item_prices = browser.find_elements_by_class_name('ooOxS')



# Initialize empty lists
titles_list = []
prices_list = []
# Loop over the item_titles and item_prices
for title in item_titles:
    titles_list.append(title.text)
for price in item_prices:
    prices_list.append(prices.text)
```

E-commerce Web Scraping

```
['NESCAFE Gold Intense Coffee 170g REFILL (Exp: Oct 18 2023) makes 85 cups',
 'AJDL sellPQKV.PH NESCAFE Gold Intense Coffee 170g REFILL (Exp: Oct 18 2023) makes 85 cups',
 'TOP bfvbfi NESCAFE Gold Intense Coffee 170g REFILL (Exp: Oct 18 2023) makes 85 cups',
 'NESCAFE Gold Instant Coffee 170g']

['₱1,102', '₱1,164.97', '₱1,104', '₱499']
```

```
dfL = pd.DataFrame(zip(titles_list, prices_list),
 columns=['ItemName', 'Price'])
```

	ItemName	Price
0	NESCAFE Gold Intense Coffee 170g REFILL (Exp: ...	₱1,102
1	AJDL sellPQKV.PH NESCAFE Gold Intense Coffee 1...	₱1,164.97
2	TOP bfvbfi NESCAFE Gold Intense Coffee 170g RE...	₱1,104
3	NESCAFE Gold Instant Coffee 170g	₱499

E-commerce Web Scraping

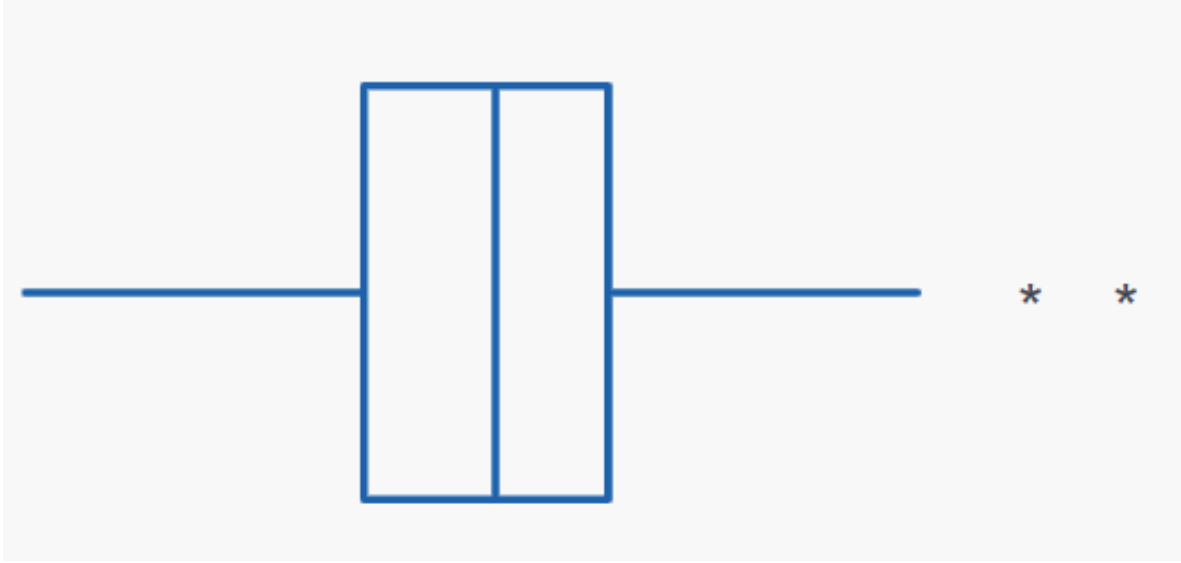
```
dfl['Price'] = dfl['Price'].str.replace('₱', '').astype(float)
```

```
dfl['Platform'] = 'Lazada'
```

	ItemName	Price	Platform
0	NESCAFE Gold Intense Coffee 170g REFILL (Exp: ...	1.102	Lazada
1	AJDL sellPQKV.PH NESCAFE Gold Intense Coffee 1...	1.164	Lazada
2	TOP bfvbfi NESCAFE Gold Intense Coffee 170g RE...	1.104	Lazada
3	NESCAFE Gold Instant Coffee 170g	0.499	Lazada

E-commerce Web Scraping

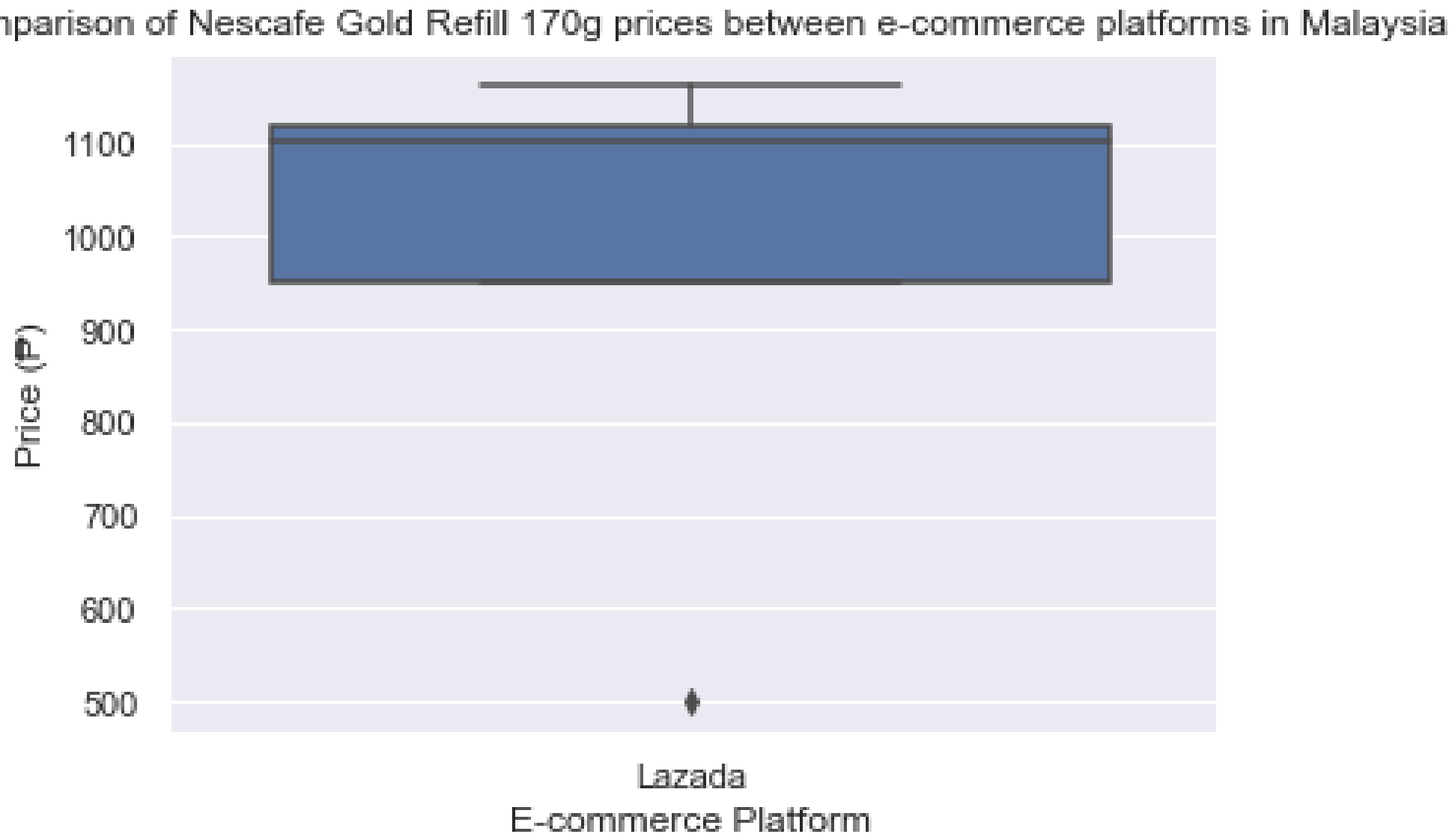
- Lowest price
- Highest price
- Median price
- 25th and 75th percentile price



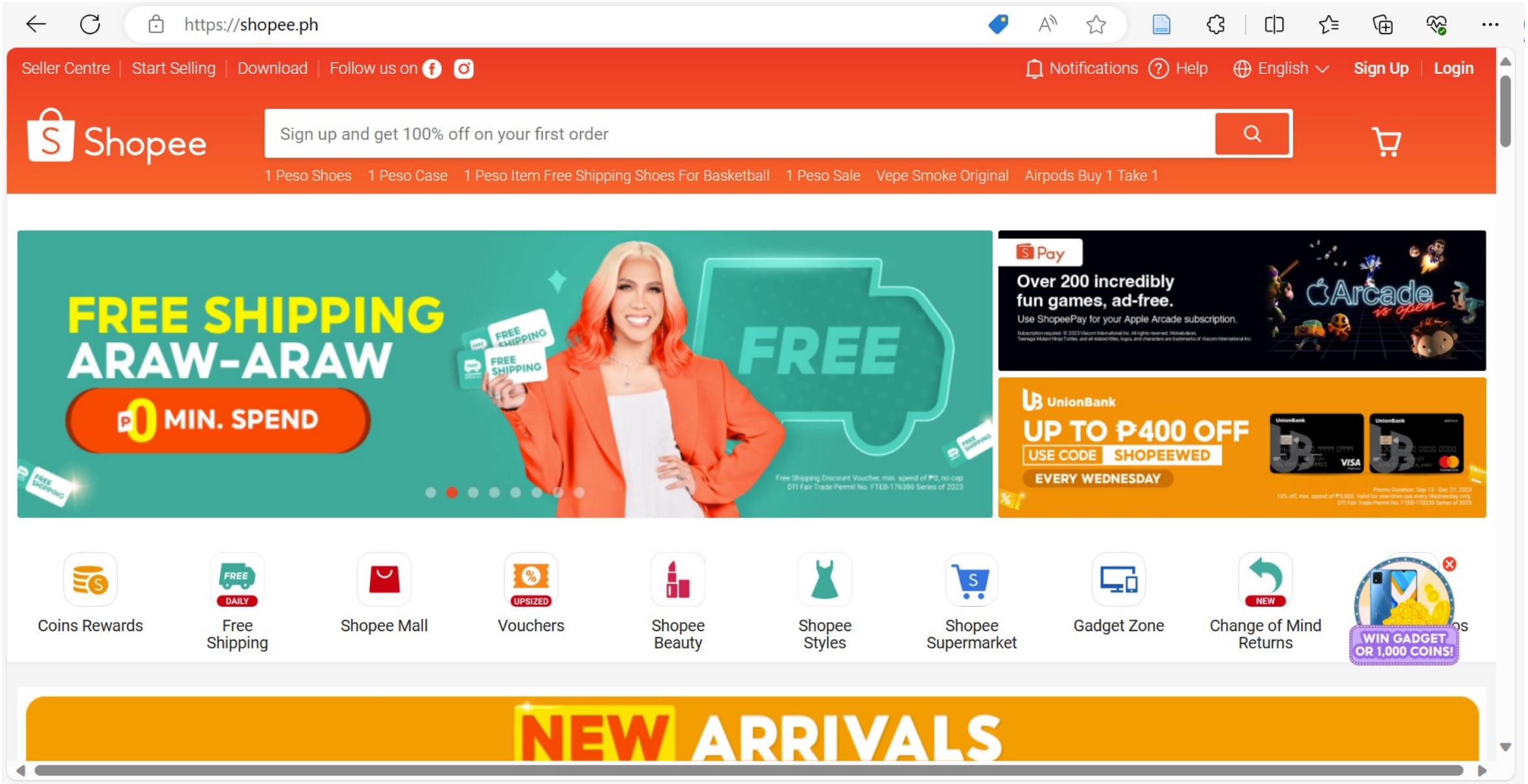
E-commerce Web Scraping

```
# Plot the chart
sns.set()
_ = sns.boxplot(x='Platform', y='Price', data=df1)
_ = plt.title('Comparison of Nescafe Gold Refill 170g prices between e-commerce platforms in Malaysia')
_ = plt.ylabel('Price (₱)')
_ = plt.xlabel('E-commerce Platform')
# Show the plot
plt.show()
```

E-commerce Web Scraping



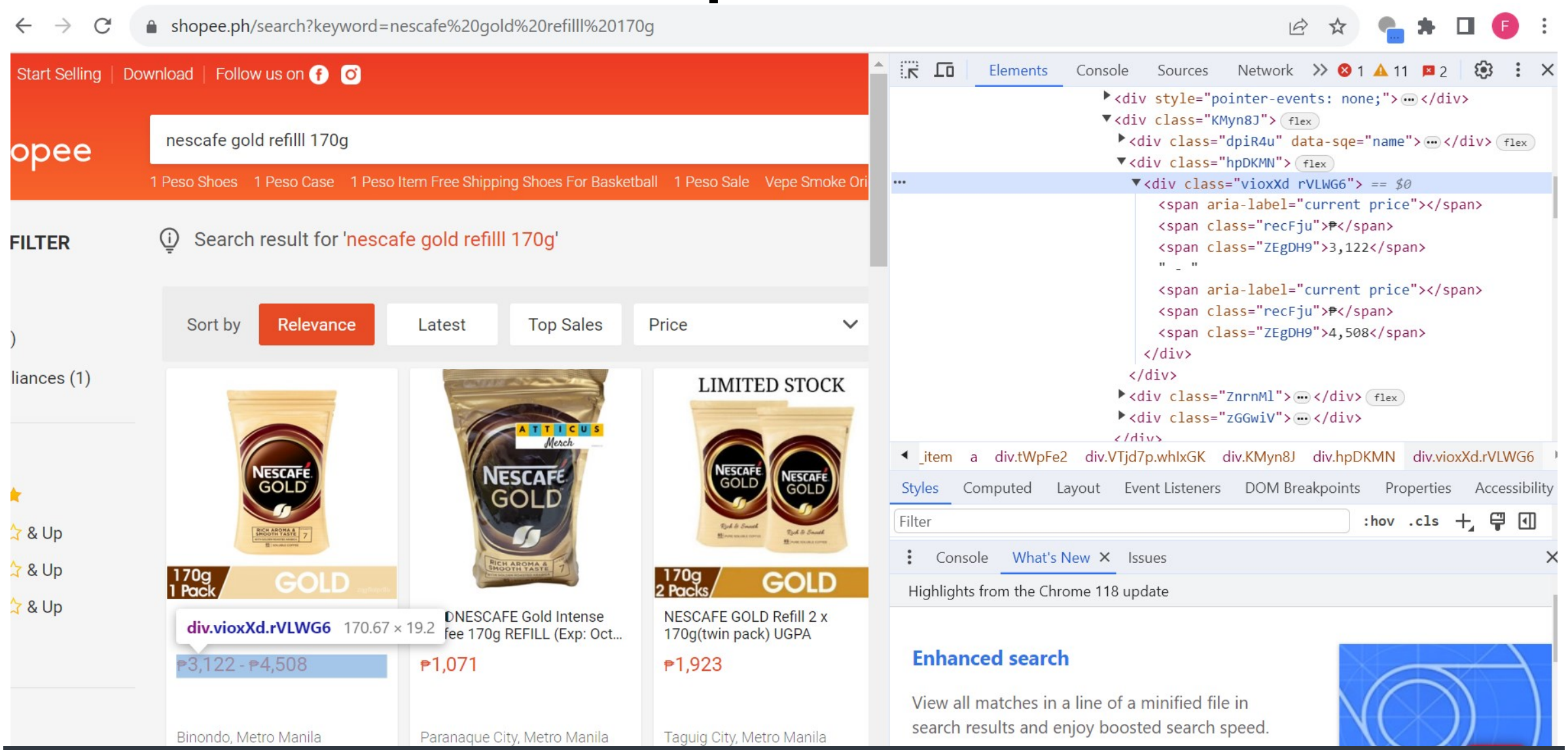
E-commerce Web Scraping



Popup Alerts


```
WebDriverWait(browser, 20).until(EC.element_to_be_clickable(
    (By.XPATH, "//div[@class='shopee-modal__container']//button[text()='English']"))).click()
```

Multiple Prices




```
product_prices = browser.find_elements_by_xpath('//a/div/div[2]/div[2]/div[@*]/span[2]')
```

Item Name are not selectable

```
import requests
Shopee_url = 'https://shopee.com.my'
keyword_search = 'Nescafe Gold refill 170g'
headers = {
 'User-Agent': 'Chrome',
 'Referer': '{}search?keyword={}'.format(Shopee_url, keyword_search)
}
url = 'https://shopee.com.my/api/v2/search_items/?by=relevancy&keyword={}&limit=100&newest=0&order=desc&page_type=search'.
format(keyword_search)
# Shopee API request
r = requests.get(url, headers = headers).json()
# Shopee scraping script
titles_list = []
prices_list = []
for item in r['items']:
    titles_list.append(item['name'])
    prices_list.append(item['price_min'])
```

E-commerce Web Scraping

```
dfS = pd.DataFrame(zip(titles_list, prices_list), columns=['ItemName', 'Price'])
```

	ItemName	Price
0	NESCAFE Gold REFILL (170g)	₱832
1	Nescafe Gold Refill 170g	₱732
2	Nescafe gold refill 170g Original Import	₱1191
3	NESCAFE Gold Intense Coffee 170g REFILL	₱1071
4	NESCAFE Gold Intense Coffee 170g REFILL	₱1082
5	NESCAFE Gold Intense Coffee 170g REFILL	₱1082
6	[Hot Sale] NESCAFE Gold Intense Coffee 170g RE...	₱1081

E-commerce Web Scraping

```
dfS['Platform'] = 'Shopee'
# Concatenate the Dataframes
df = pd.concat([dfL,dfS])
```

	ItemName	Price	Platform
0	NESCAFE Gold Intense Coffee 170g REFILL (Exp: ...	1102.0	Lazada
1	AJDL sellPQKV.PH NESCAFE Gold Intense Coffee 1...	1164.0	Lazada
2	TOP bfvbfi NESCAFE Gold Intense Coffee 170g RE...	1104.0	Lazada
3	NESCAFE Gold Instant Coffee 170g	499.0	Lazada
0	NESCAFE Gold REFILL (170g)	832.0	Shopee
1	Nescafe Gold Refill 170g	732.0	Shopee
2	Nescafe gold refill 170g Original Import	1191.0	Shopee
3	NESCAFE Gold Intense Coffee 170g REFILL	1071.0	Shopee
4	NESCAFE Gold Intense Coffee 170g REFILL	1082.0	Shopee
5	NESCAFE Gold Intense Coffee 170g REFILL	1082.0	Shopee
6	[Hot Sale] NESCAFE Gold Intense Coffee 170g RE...	1081.0	Shopee

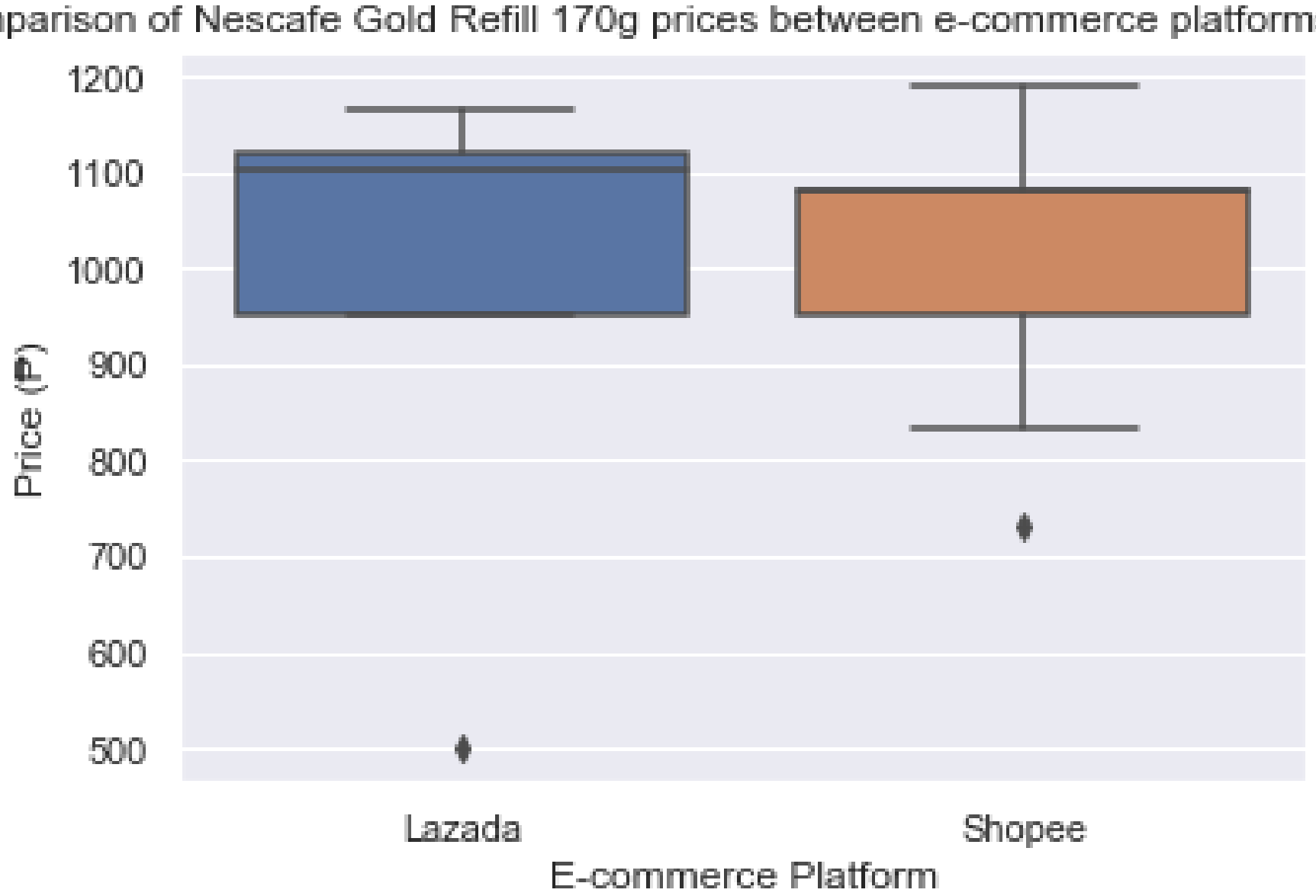
E-commerce Web Scraping

```
print(df.groupby(['Platform']).describe())
```

	Price							
	count	mean	std	min	25%	50%	75%	max
Platform								
Lazada	4.0	967.250000	313.489367	499.0	951.25	1103.0	1119.0	1164.0
Shopee	7.0	1010.142857	163.737013	732.0	951.50	1081.0	1082.0	1191.0

E-commerce Web Scraping

```
sn
_
_
_
_
#
pl
```




```
e')
```

Libraries

It allows us to make an http request to different websites

It opens a socket to the target website and asks them for their permission to connect

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Beautiful Soup

Table Of Contents

Beautiful Soup Documentation

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

These instructions illustrate all major features of Beautiful Soup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

The examples in this documentation should work the same way in Python 2.7 and Python 3.2.

You might be looking for the documentation for Beautiful Soup 3. If so, you should know that Beautiful Soup 3 is no longer being developed, and that Beautiful Soup 4 is recommended for all new projects. If you want to learn about the differences between Beautiful Soup 3 and Beautiful Soup 4, see Porting code to BS4.

This documentation has been translated into other languages by Beautiful Soup users:

- 这篇文档当然还有中文版.
- このページは日本語で利用できます(外部リンク)
- 이 문서는 한국어 번역도 가능합니다. (외부 링크)

Getting help

Beautiful Soup Example

```
uClient=uRequest("https://www.thomann.de/it/ukulele_soprani.html")
page_html = uClient.read()
page_soup = soup(page_html)
containers=page_soup.findAll('div',{'class':'extensible-article'})
print(len(containers))
print(containers[0])
```

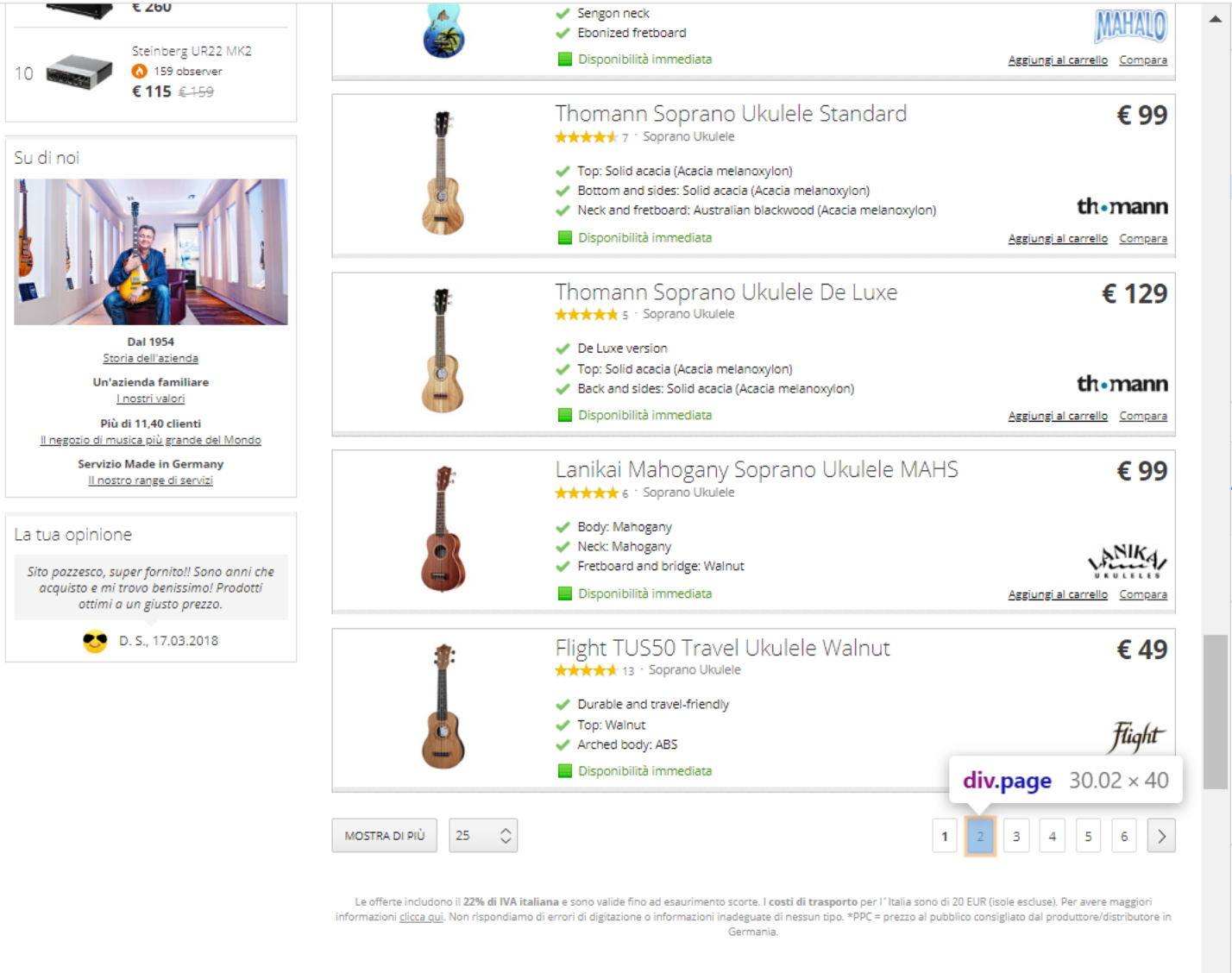
```
25
<div class="extensible-article list-view compare parent" rel="320760"> <div clas
s="left"> <div class="product-image image-block"> <a class="article-link link" h
ref="https://www.thomann.de/it/harley_benton_ukulele_uk_11dw_brown.htm"> <img al
t="Harley Benton Ukulele UK-11DW Brown" class="image" height="150" original="htt
ps://cdt.static-thomann.de/thumb/thumb150x150/pics/prod/320760.jpg" src="http
s://cdt.static-thomann.de/thumb/thumb150x150/pics/prod/320760.jpg" width="150"/>
```

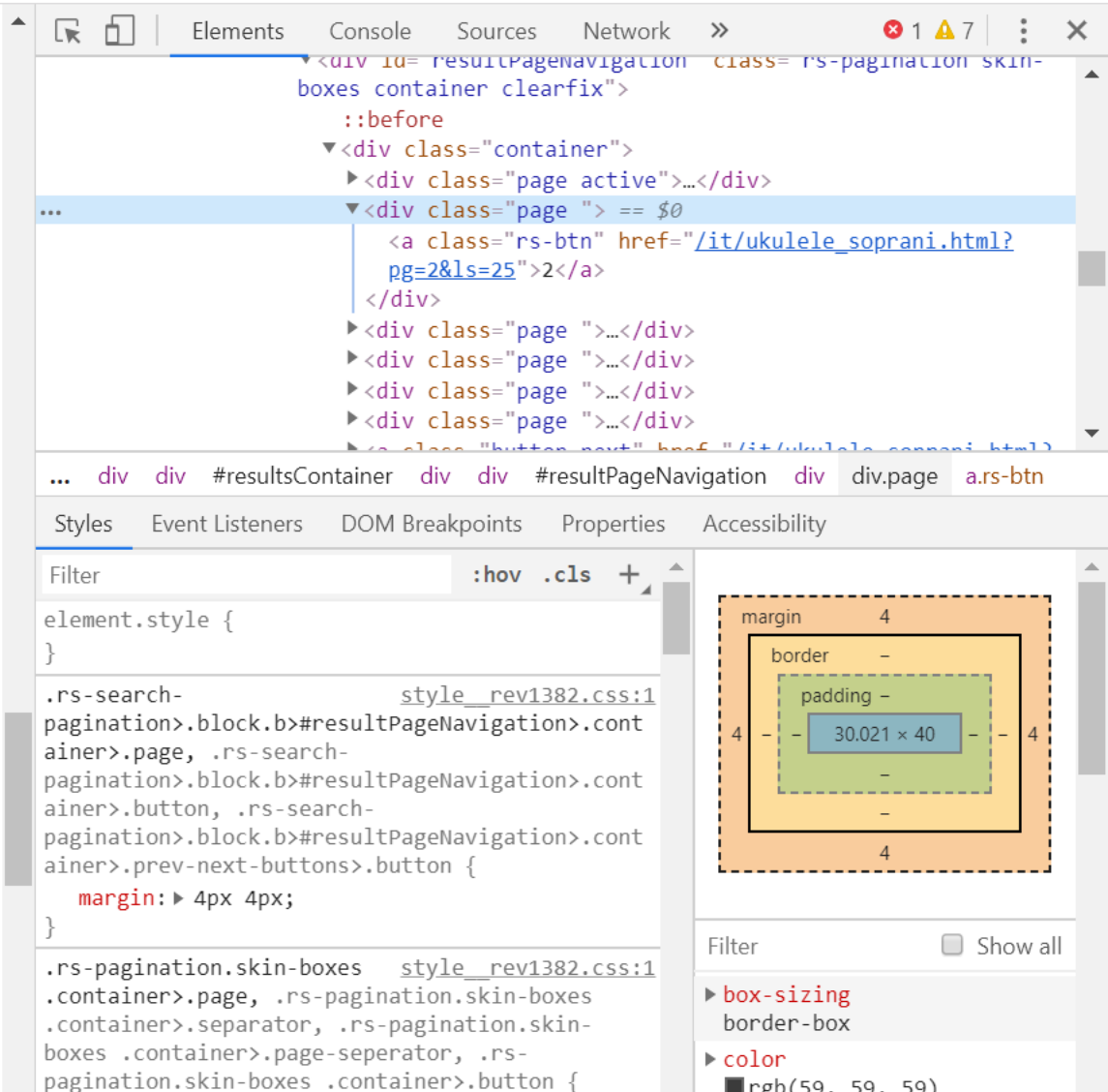
Beautiful Soup Example

```
print(containers[0].findAll('span',{'class':'manufacturer'})[0].text)
print(containers[0].findAll('span',{'class':'model'})[0].text)

print()

print(containers[1].findAll('span',{'class':'manufacturer'})[0].text)
print(containers[1].findAll('span',{'class':'model'})[0].text)
```

```
Harley Benton
Ukulele UK-11DW Brown

Baton Rouge
V2-S sun
```

Beautiful Soup Example





Addiotional References

- Web Scraping With Python - Full Guide to Python Web Scraping (edureka.co)
- What Is A Robots.txt File? Best Practices For Robot.txt Syntax – Moz
- DOM (Document Object Model) - GeeksforGeeks