# Application of Big Data in Social Science

# week 2

Heidi Hyeseung Choi
Fall 2022
HYSIS
heidichoi@hanyang.ac.kr

# Changed schedule ☺

## Original Schedule

| | | |
|---|---|---|
| 9.1 | 1 | Introduction |
| 9.8 | 2 | Web Scraping |
| 9.15 | 3 | |
| 9.22 | 4 | Natural Language Processing |
| 9.29 | 5 | |
| 10.6 | 6 | Text Analysis |
| 10.13 | 7 | |
| 10.20 | 8 | Revision? |
| 10.27 | 9 | Mid term exam |
| 11.3 | 10 | Social Network Analysis |
| 11.10 | 11 | Machine Learning: Supervised Learning |
| 11.17 | 12 | |
| 11.24 | 13 | Machine Learning: Unsupervised Learning |
| 12.1 | 14 | |
| 12.8 | 15 | Data Visualization |
| 12.15 | 16 | Final Exam |

## Updated version

| | | |
|---|---|---|
| 9.1 | 1 | Introduction |
| 9.8 | 2 | Web Scraping |
| 9.15 | 3 | |
| 9.22 | 4 | Natural Language Processing |
| 9.29 | 5 | |
| 10.6 | 6 | Text Analysis (recorded lecture on week 7) |
| 10.13 | 7 | |
| 10.20 | 8 | Mid term exam (as school schedule) |
| 10.27 | 9 | Social Network Analysis |
| 11.3 | 10 | |
| 11.10 | 11 | Machine Learning: Supervised Learning |
| 11.17 | 12 | |
| 11.24 | 13 | Machine Learning: Unsupervised Learning |
| 12.1 | 14 | |
| 12.8 | 15 | Data Visualization |
| 12.15 | 16 | Final Exam |

# By today's class

- Would have installed Anaconda
- The classes would be based on windows not mac.
- Highly encourage you to google your way out first.

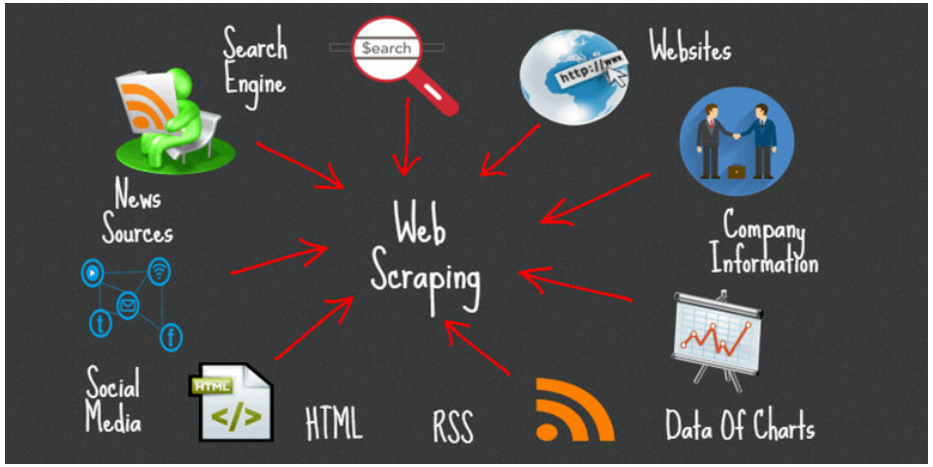
1. What is web scraping?
2. Components of a web page.
   - HTML
   - HTTP
3. HTML basics
4. What we will go through today

# What is Web Scraping?

- A LOT of data is stored on websites.
- Sometimes a need to gather information of specific data on website arises.
- “How we do acquire this data?” → Answer: Web scraping!
- Definition of web scraping :

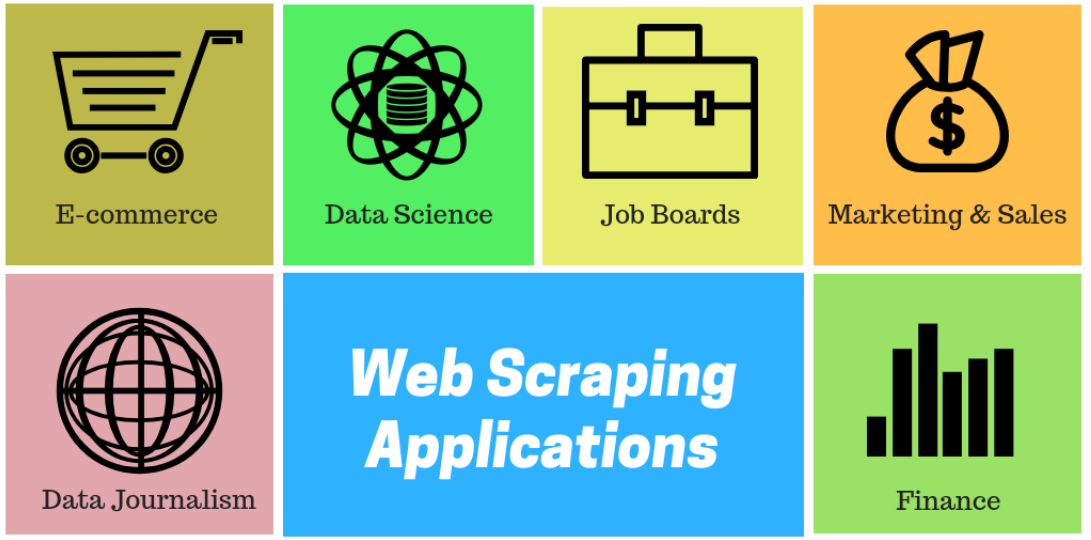
  “the construction of an agent to download, parse and organize data from the web in an automated manner”. (Brouckes and Baesens, 2018)

- Instead of you manually copy and pasting off websites, let’s let computer programs do that for you, much faster and more accurately.



# Why do we need to do this?

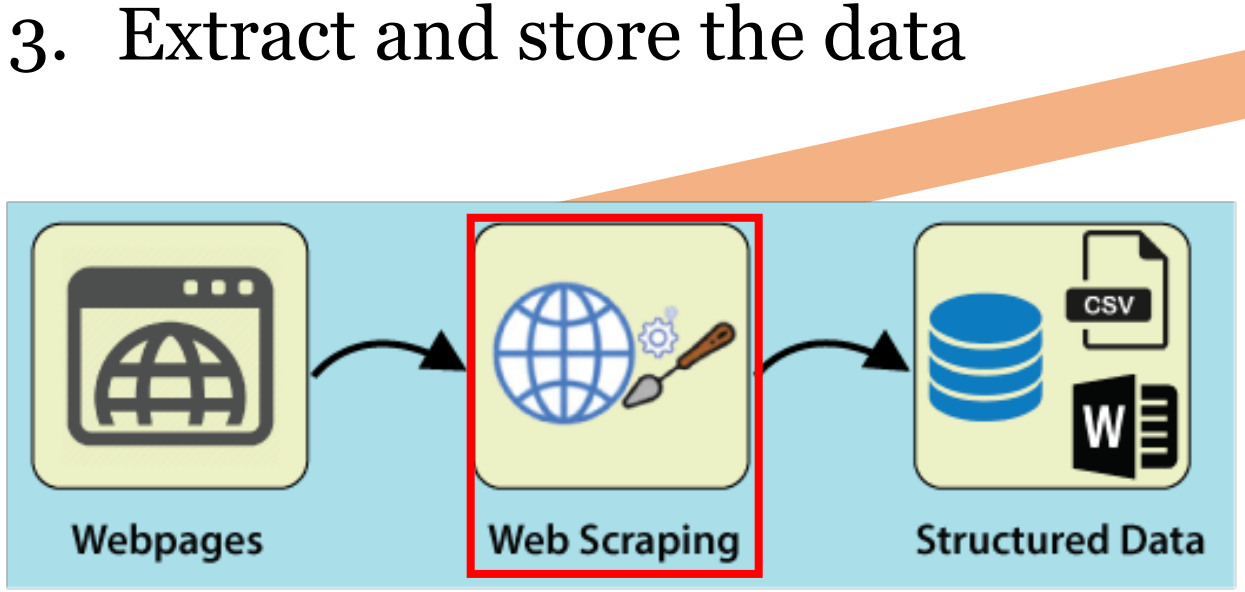
- Websites have tons of data that we may find useful to analyze.
  - list of properties on a real-estate site.
  - list of reviews from a movie site for text analysis etc.
  - to monitor a news site for trending news story of your interest.
  - to keep track of stock prices
- If data files are readily available in files, we are lucky.
- Reality: most of the times data files are not readily available in neat format for you.

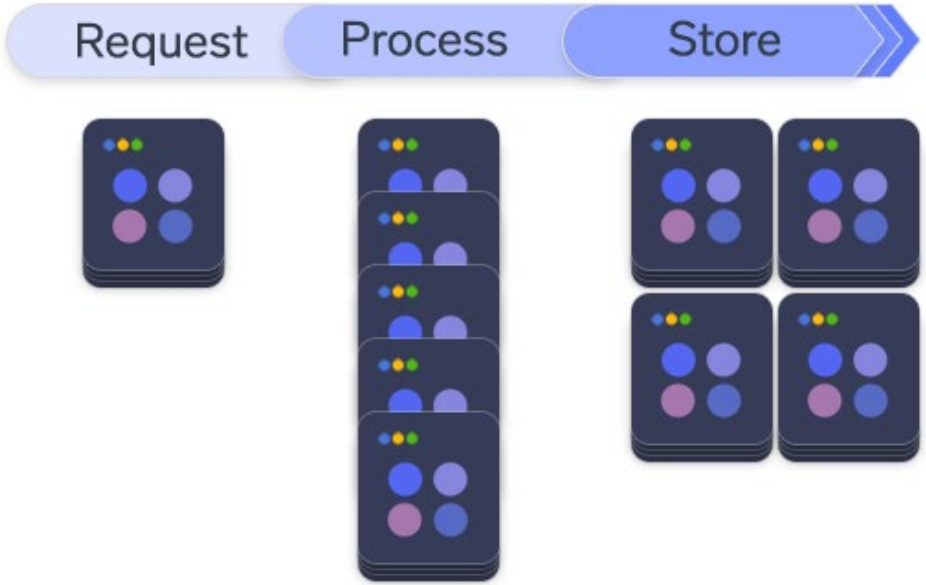


# Steps for web scraping

- Web scraping is organized in the following steps:

1. Make a ‘request’ to the URL which contains the necessary HTML data.
2. ‘Parse’ the HTML and find the element with particular data you are looking for
3. Extract and store the data



# Rules and ethics of web scraping

- Website owners have their rights to protect themselves (their data or website) from scraping because they are under these risks:
  - Contents can be unintendedly stolen
  - Websites can crash (when requests are sent in bulks and accidentally create attacks on the web server)
  - Website's algorithms and inner workings may be exploited and manipulated. Etc

- Therefore, there are guidelines for scraping websites due to legal issues.
- Always remember to check the "Terms and Conditions" for legal use of data.
- Unless you use it for commercial purposes, scraping data off websites are normally OK.

- Don't be a burden to websites!
  - Do not spamming websites with tons of requests.
  - It may bring server breakdown due to large traffic.

- Don't violate copyrights
  - on articles, videos, pictures, stories, music, data etc.
  - Ask for permissions

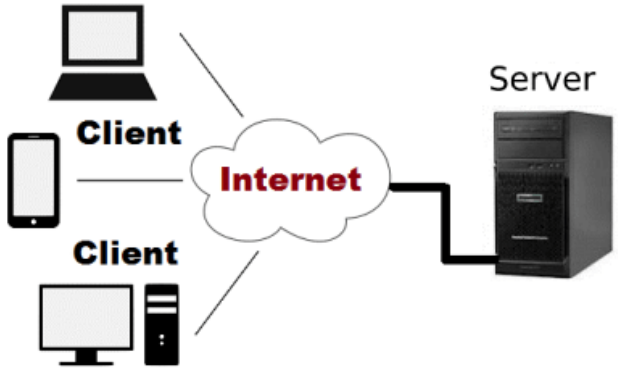
# Components of a webpage

- HTML
- HTTP

# The Components of a Web Page: HTML

- When we access a website, our web browser (**client**) makes a 'request' to the web **server**.
- Then, the server sends back 'HTML documents' that tells our browser how to render the page.
- The documents that the server sent fall into a few main types:
  - Hyper Text Markup Language (HTML): the standard markup language designed to be displayed in a web browser.
    - contains the main content of the page.
    - it can be assisted by CSS and JS
  - Cascading Style Sheets (CSS): a style sheet language used for the layout of a document written in HTML
    - decorates the page by adding styles (eg. fonts, layout, colors etc.)
  - JavaScript (JS) : programming language on the behaviors of web pages
    - scripting language for doing dynamic things on the web.
    - eg) extra content added to a page, dynamically changing element colors, etc


- Once our browser receives all the necessary files on the website, it displays it to us (rendering).
- When web scraping we are primarily interested in HTML, which has the actual contents.

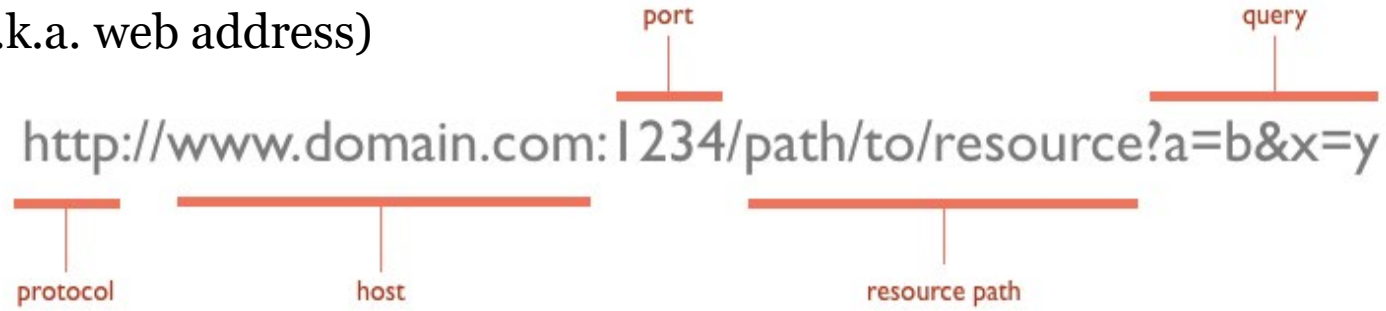


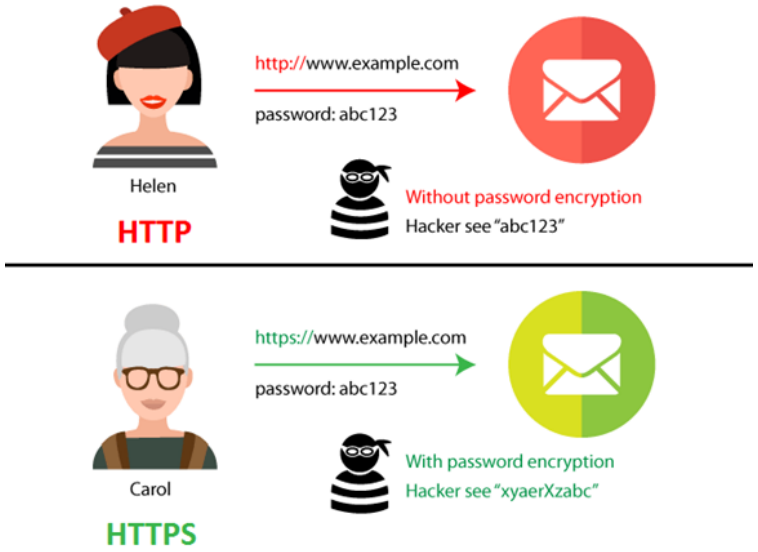
# The Components of a Web Page: HTTP

- Hypertext Transaction Protocol (HTTP) : a **protocol** which allows the fetching of resources such as HTML documents
  - A **protocol** is a system of rules that define how data is exchanged within or between computers.
  - Communications between devices require that the devices agree on the format of the data that is being exchanged.
- It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser.
- In this request-response process, HTTP uses pre-defined standards and rules for the exchange of information. In general, HTTP is the protocol that clients and servers use to communicate.
- HTTPS is the encrypted version of HTTP (HTTP Secure)
- **Uniform Resource Locator (URL):**

  - reference to a web resource that specifies its location on a computer network.

  (a.k.a. web address)



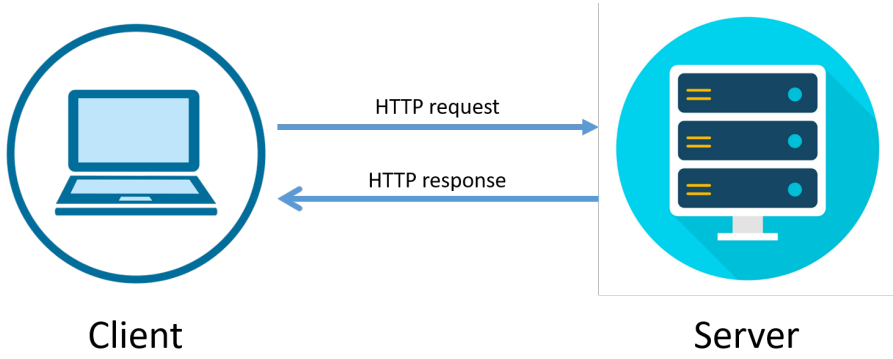
# HTTP request

**HTTP request**

- To access internet, my browser needs to send a request to the servers and display the resources for me.
- HTTP is the underlying format that is used to structure requests and responses for communication between a client and a server.
- When we type in the URL, the message is sent to the server which is known as an HTTP request.

**When we scrap web pages**

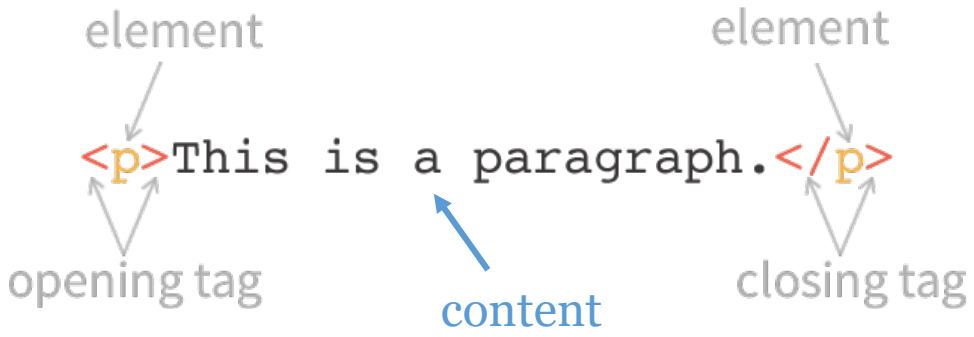
- we write code that sends a request to the server to download that page's source code.
- we filter through the page for HTML elements,
- and extract information that we need.



# HTML basics

# Hyper Text Markup Language (HTML)

- HTML is the standard markup language* for making web pages.
  - Markup language is a computer language that uses tags to define elements.
  - It is human-readable, meaning it contain standard words, rather than typical programming syntax.
- HTML describes the structure of a web page.
- HTML consists of a series of elements.
- Elements are like toolbox used to shape the websites.
  - eg) elements that insert paragraphs, headings, sections etc.
- Tags are used to mark up the start of an HTML element.
- An HTML element is normally composed by two tags: an opening tag `< >` and a closing tag `</ >`
- Most tags must be opened `<h1>` and closed `</h1>` in order to function.
- Whatever falls in between its opening and closing tags is the content

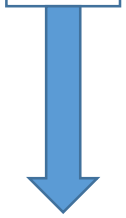


# Hyper Text Markup Language (HTML)

- Attributes are the properties of an element.
- All HTML elements can have **attributes**
- Attributes provide **additional information** about elements
- Attributes are always specified in **the start tag.**
- Attributes usually come in key/value pairs like: **name="value"**

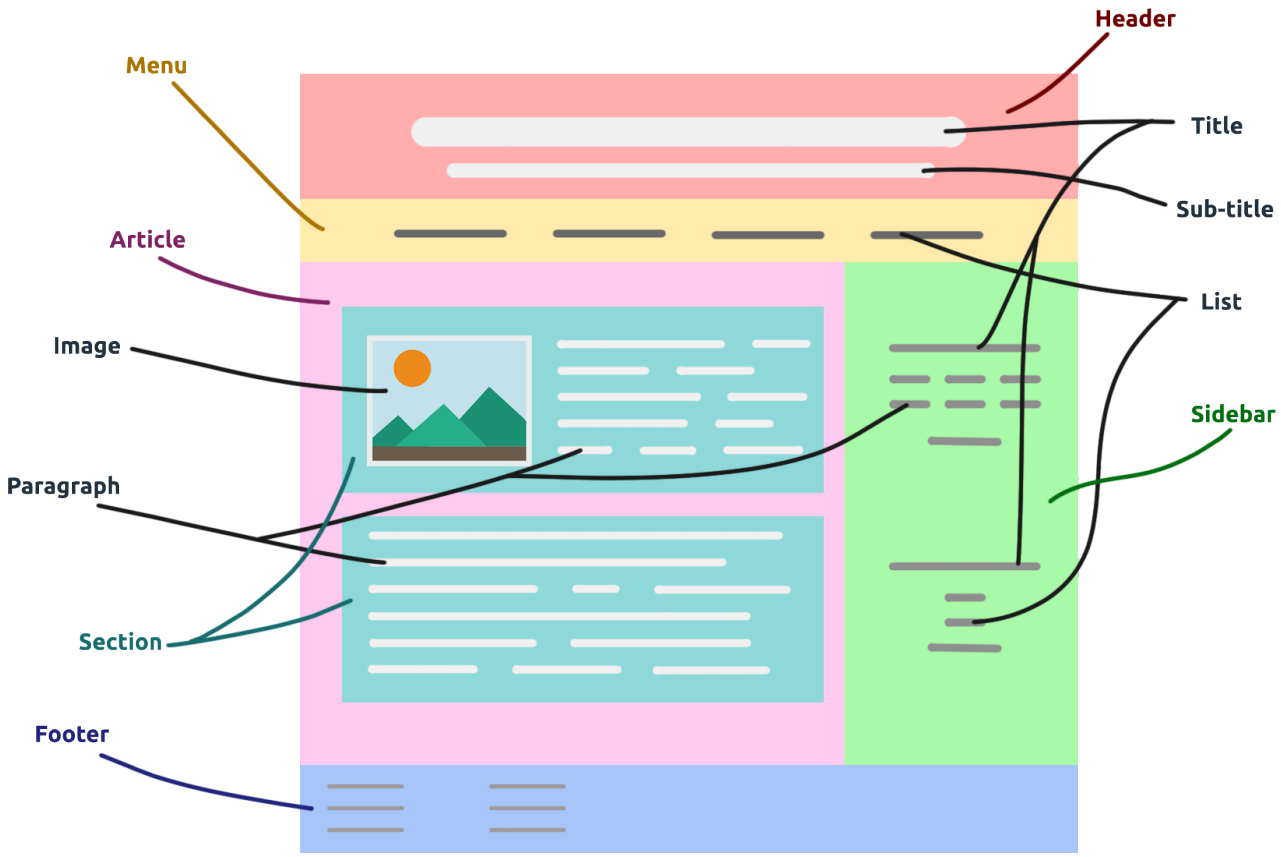
examples..

```
<b style = "color:red"> Red Bold text </b>
<p> Let's access <a href = https://www.google.com> google</a>! </p>
```



href = hypertext reference,
specifies the URL of the page the link goes to.

# Basic structure of a HTML page




```
<html>
  <head>
    <title>This Is Your Title </title>
  </head>
  <body>
    <h1> This Is Your Header </h1>
    <p> This is your paragraph. </p>
  </body>
</html>
```

https://www.development-tutorial.com/basic-structure-html-page/

https://html.com/

# HTML

## Document Summary

**<html> … </html>**
The tag shows up at the beginning and end of an HTML document (known as the root element). It indicates that the webpage is written in HTML5, and all other page markup comes in between these beginning and ending tags.

**<title> … </title>**
The title tag is the title for that page, useful for both search engines (when they scan and index pages) and users (showing up up in a browser's title bar) by explicitly stating the primary topic of each page.

**<head> … </head>**
The contains information that specific page, including the title tags, meta data, and links to scripts and style sheets.

**<body> … </body>**
Body tags include all content that will be shown to users, including everything they'll see & read.

### Example

```
<html>
  <head>
    <title>My Beautiful Website</title>
  </head>
  <body>

  </body>
</html>
```

# HTML

## Document Structure

**<h1..h6> ... </h1..h6>**
All six levels of Headings, with 1 being the most important on a page and 6 being the least. These elements are used to describe content sections on a page.

** ... **
A generic container used to denote a page section or

** ... **
An inline section or block container, typically used for grouping styling elements.

** ... **
This foundational tag is used to organize paragraphs of text.

**<br/>**
Creates a line break (or old-school carriage-return), useful for writing blocks of text that need to be on different lines (think addresses, etc.)

**<hr/>**
Creates a horizontal rule, a sectional break in an HTML page. Typically used to denote a change in topic or section of a page.

### Example

```
<div>
  <h1>Ways to make your cat happy</h1>
  <p>You have a <span>mini-lion</span> at
home and you want to make it as happy as pos-
sible.</p>

  <hr/>

  <h2>Feed your cat well.</h2>
  <p>The right diet is <span>extremely</span>
important for the wellbeing of your cat.<br/>
Obesity is a common source of problems among
domesticated animals.</p>
</div>
```

**Ways to make your cat happy**

You have a mini-lion at home and you want to make it as happy as possible.

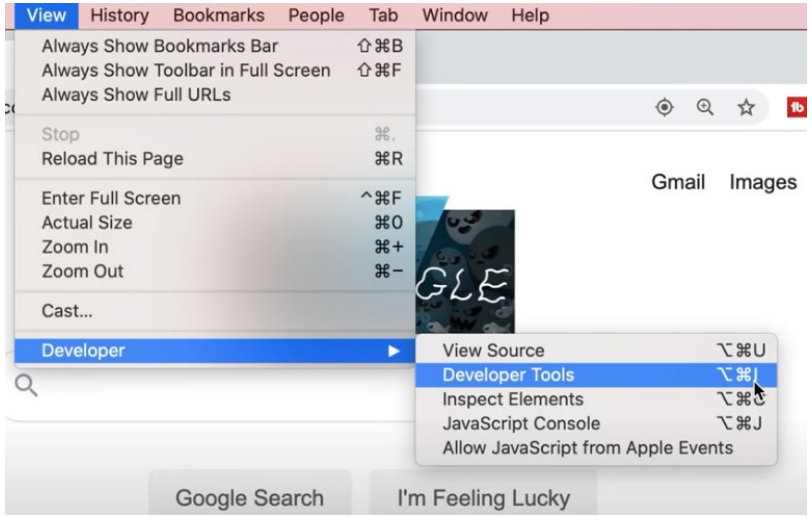
---

**Feed your cat well.**

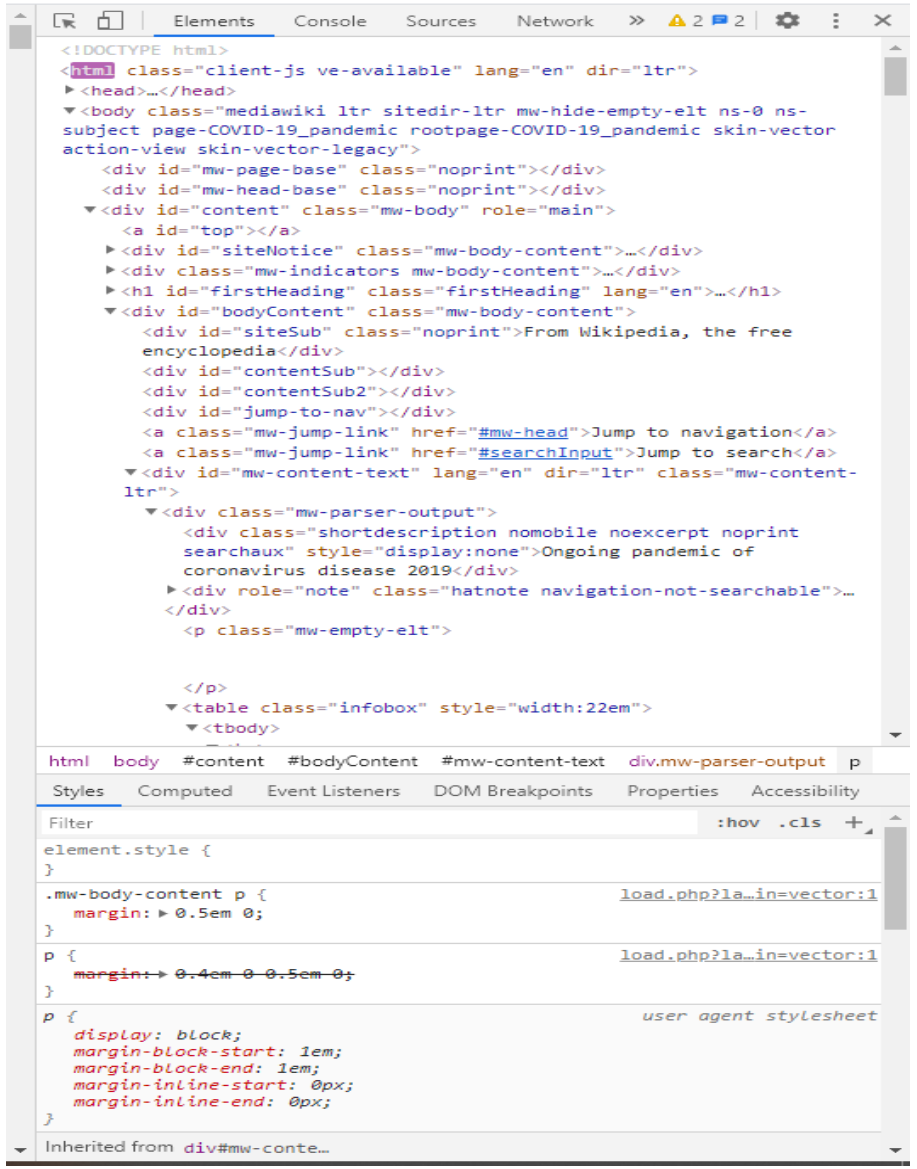
The right diet is extremely important for the wellbeing of your cat.
Obesity is a common source of problems among domesticated animals.

# Where to find it: DevTools

- Developer tools (DevTools)
- Chrome DevTools is a set of web developer tools built directly into the Google Chrome browser.
- On chrome:
  - ‘F12’ for windows, ‘Cmd +opt + i’ on mac




Developer Tools on mac



# Steps in web scraping that we will go through.

1. Retrieve HTML data from a domain name
2. Parse that data for target information
3. Store the target information

# Steps in web scraping that we will go through.

1. Retrieve HTML data from a domain name
2. Parse that data for target information
3. Store the target information

# HTTP request

Let’s try making a request.

We will be scraping quotes from this site (quotes : quotes.toscrape.com)

Let’s surf through that site first.

- import requests library

```
import requests
page = requests.get("http://quotes.toscrape.com/")
page
```

```
<Response [200]>
```

- and send an HTTP request.
- If we get Response [200] it means we downloaded html successfully.

# Steps in web scraping that we will go through.

1. Retrieve HTML data from a domain name
2. Parse that data for target information
3. Store the target information

# HTML parsing

- What is HTML parsing?
  - reads through downloaded HTML and recognizes and classifies its contents into its components.
  - recognizes tags such as <head>, the content of its elements.
  - we will be using BeautifulSoup library of Python

```
>>> from bs4 import BeautifulSoup as bs
>>> soup = bs(page.content, 'html.parser')
>>> print(soup)
<!DOCTYPE html>

<html lang="en">
<head>
<meta charset="utf-8"/>
<title>Quotes to Scrape</title>
<link href="/static/bootstrap.min.css" rel="stylesheet"/>
<link href="/static/main.css" rel="stylesheet"/>
</head>
<body>
<div class="container">
```

# HTML parsing

- It is hard to visually see how it is structured.
- I want to make it pretty. Thus, use the code `.prettify`

```
>>> print(soup.prettify())
```

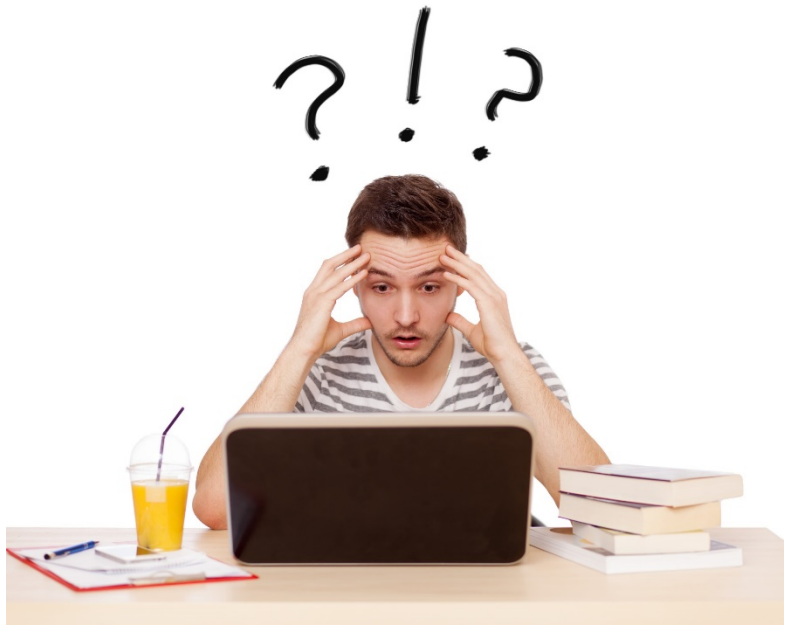
```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   Quotes to Scrape
  </title>
  <link href="/static/bootstrap.min.css" rel="stylesheet"/>
  <link href="/static/main.css" rel="stylesheet"/>
 </head>
 <body>
  <div class="container">
   <div class="row header-box">
    <div class="col-md-8">
     <h1>
      <a href="/" style="text-decoration: none">
       Quotes to Scrape
      </a>
```

# HTML parsing

- It is hard to visually see how it is structured.
- I want to make it pretty. Thus, use the code `.prettify`

```
>>> print(soup.prettify())
```

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   Quotes to Scrape
  </title>
  <link href="/static/bootstrap.min.css" rel="stylesheet"/>
  <link href="/static/main.css" rel="stylesheet"/>
 </head>
 <body>
  <div class="container">
   <div class="row header-box">
    <div class="col-md-8">
     <h1>
      <a href="/" style="text-decoration: none">
       Quotes to Scrape
      </a>
```




**Let's go back to our website and use DevTool! and find where our quotes are!**

# Steps in web scraping that we will go through.

1. Retrieve HTML data from a domain name
2. Parse that data for target information
3. Store the target information

# store the target information

- Now that we located where information that we are looking for(quotes) is..

```
::before
<div class="col-md-8">
  <div class="quote" itemscope itemtype="http://schema.org/CreativeWork">
    <span class="text" itemprop="text">
      ""The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."" == $0
    </span>
    <span>…</span>
    <div class="tags">…</div>
  </div>
  <div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
  <div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
  <div class="quote" itemscope itemtype="http://schema.org/CreativeWork">…</div>
```

# store the target information

- We will try to find our quotes.

```
>>> soup.find('span', class='text')
>>> # this would give out errors, because the word class is a reserved word in Python
```

- So, to avoid the conflict, we will add _ at the end of class_ to tell Python that it should not understand it as a reserved word.

```
>>> soup.find('span', class_='text')
```

- To find all of them, we can use `find_all()`

```
>>> soup.find_all('span', class_='text')
```

```
find(name, attributes)
```

# store the target information

- We will save this data in a variable called txt.
- Then it will give out..

```
>>> txt= soup.find_all('span', class_='text')
```

```
[<span class="text" itemprop="text">“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.” </span>, <span class="text" itemprop="text">“It is our choices, Harry, that show what we truly are, far more than our abilities.” </span>, <span class="text" itemprop="text">“There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle.” </span>, <span class="text" itemprop="text">“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.” </span>, <span class="text" itemprop="text">“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.” </span>, <span class="text" itemprop="text">“Try not to become a man of success. Rather become a man of value.” </span>, <span class="text" itemprop="text">“It is better to be hated for what you are than to be loved for what you are not.” </span>, <span class="text" itemprop="text">“I have not failed. I've just found 10,000 ways that won't work.” </span>, <span class="text" itemprop="text">“A woman is like a tea bag; you never know how strong it is until it's in hot water.” </span>, <span class="text" itemprop="text">“A day without sunshine is like, you know, night.” </span>]
```

# store the target information

- Then, we will get the text out by using .get_text()
- and we will also loop through the entire txt list to get quotes out.

```
>>> for i in txt:
>>>     print(i.get_text())
```

```
 “The world as we have created it is a process of our thinking. It cannot be changed without changing our thin
king.”
 “It is our choices, Harry, that show what we truly are, far more than our abilities.”
 “There are only two ways to live your life. One is as though nothing is a miracle. The other is as though eve
rything is a miracle.”
 “The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”
 “Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely borin
g.”
 “Try not to become a man of success. Rather become a man of value.”
 “It is better to be hated for what you are than to be loved for what you are not.”
 “I have not failed. I've just found 10,000 ways that won't work.”
 “A woman is like a tea bag; you never know how strong it is until it's in hot water.”
 “A day without sunshine is like, you know, night.”
```

# store the target information

- I want to save quotes in a list called quotes.
- To do so, we will first create an empty list called quotes.
- Then append what we saw in the previous slide into empty quotes list.

```
>>> quotes = []
>>> for i in txt:
>>>     quotes.append(i.get_text())
```

**Let’s launch our jupyter notebook**

**and try it for ourselves**

>>>pip install BeautifulSoup4

# Done for today!

Next week, we will learn a little more advanced techniques.