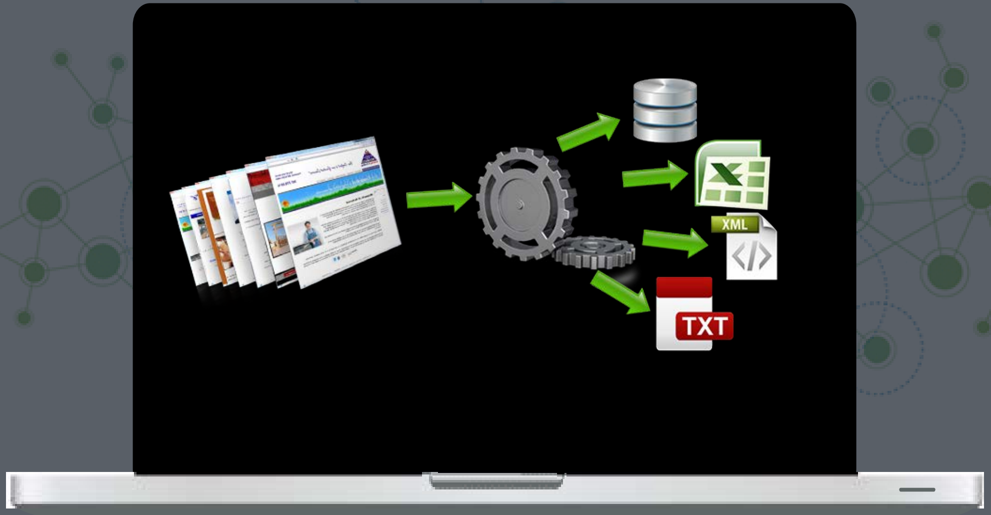


Gathering Data from the web

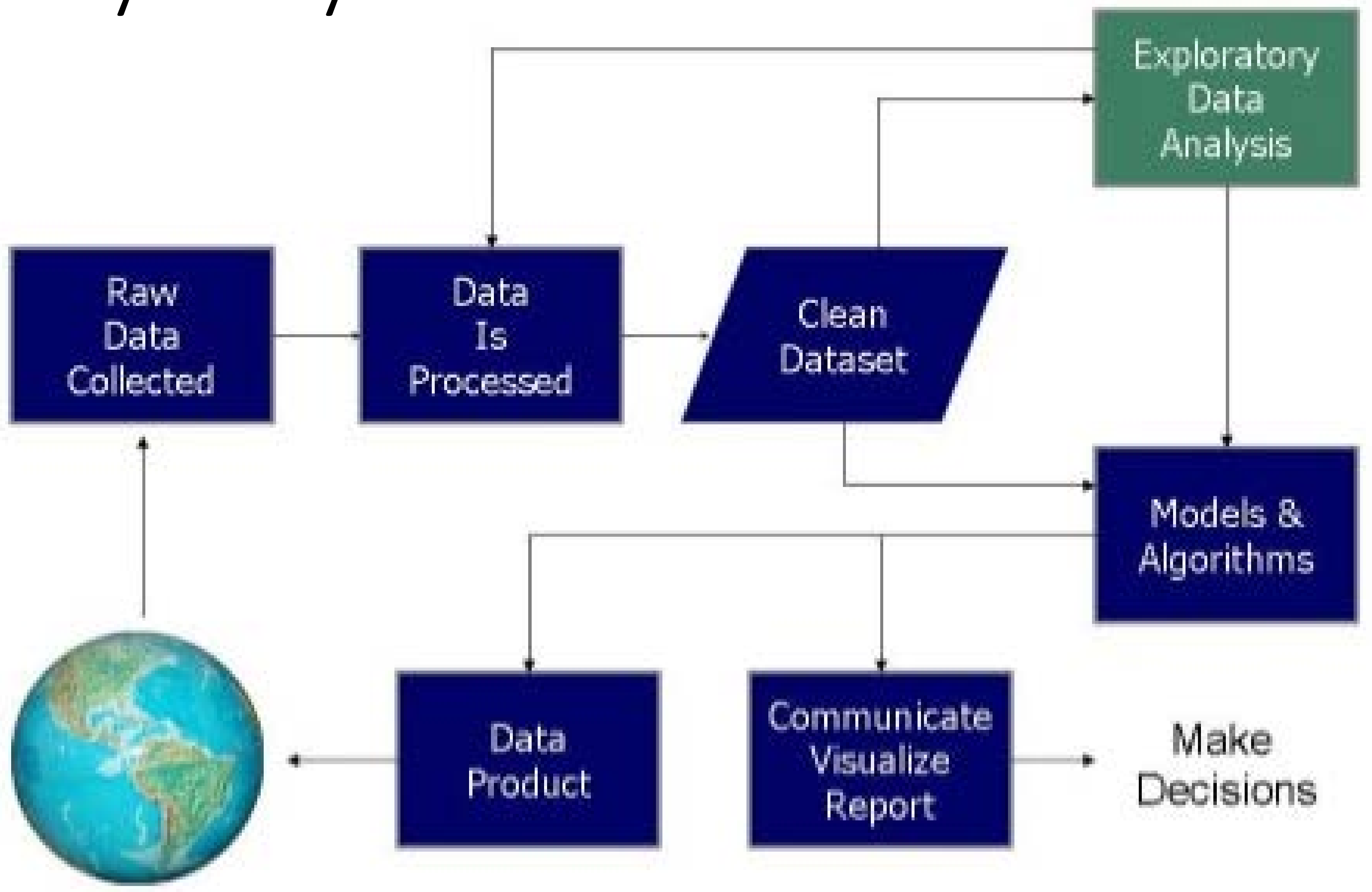
Rafael Caballero Roldán

Departamento de Sistemas Informáticos y Computación

UCM

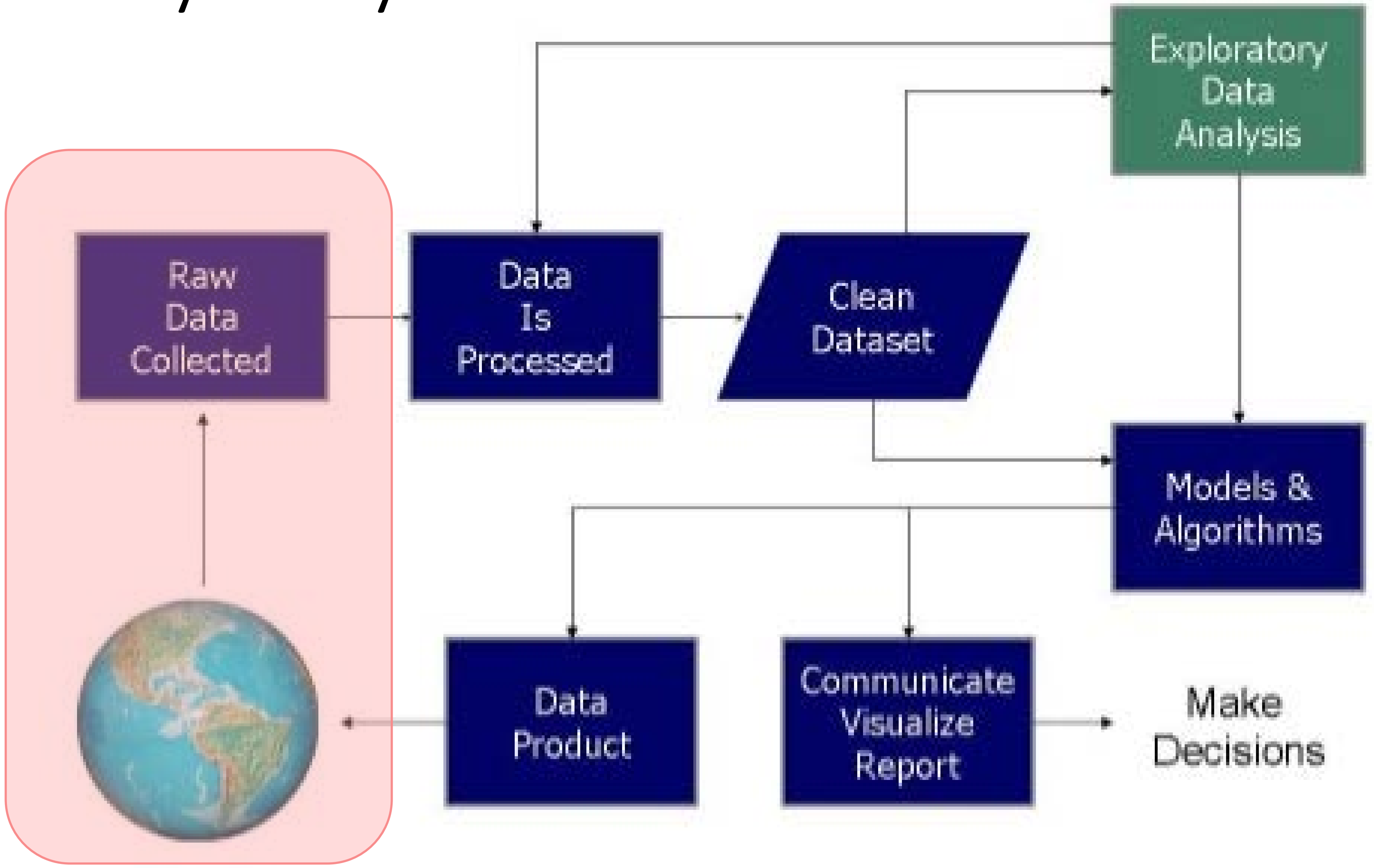
Web Scraping

Data Analysis cycle



https://www.slideshare.net/SparkSummit/data-science-lifecycle-with-apache-zeppelin-and-spark-by-moonsoo-lee

Data Analysis cycle



https://www.slideshare.net/SparkSummit/data-science-lifecycle-with-apache-zeppelin-and-spark-by-moonsoo-lee

Access categorization

- Files directly accesible kwnow its URI → direct download and local treatment (*requests* library in Python)
- APIs: usually they require autentification and know the details of the API
- Web Scraping: the data are embebed in the page and we need to extract them accessing to the underlying HTML code

Note: Web scraping vs Web crawling: not the same!

- Web scraping consists in collecting data from a given web page knowing its structure and where to locate the elements

- Web crawling: a “robot” traverses a lot of web pages, often following their links looking for particular information

Web scraping

- A powerful (but complex) way of obtaining data
- Very common in IoT: web pages that show the data yielded by sensors (medioambiental, seccurity, weather, etc.).
- Often is not easy to “catch” the data automatically
- A first step is to examine the code in which the web page is written: HTML

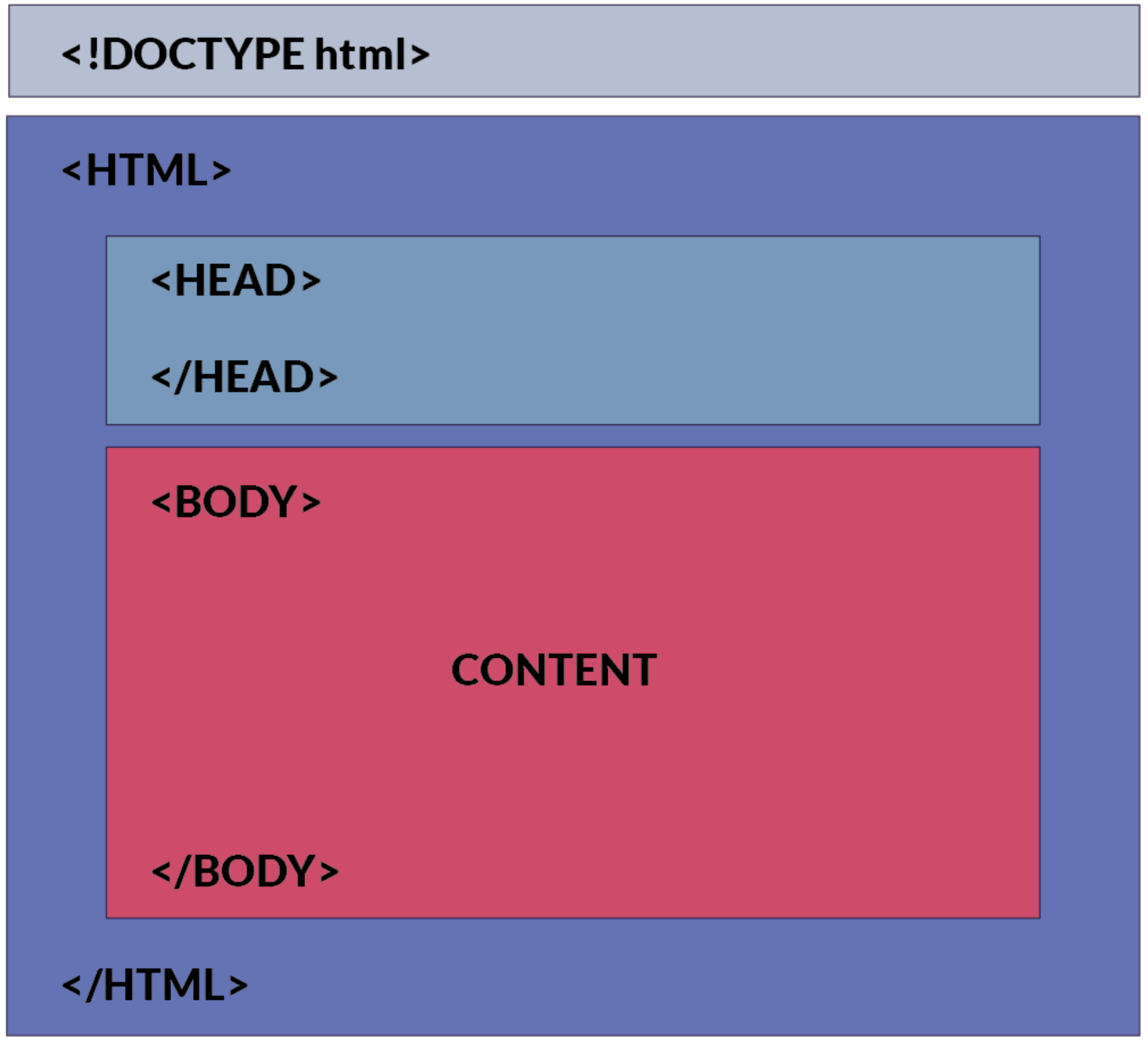
Scraping: types of pages

1. Static: when the URL is typed in the browser the displayed page already contains the information we are looking for. In this case we can employ the Python library BeautifulSoup in order to analyze the page and obtain the information

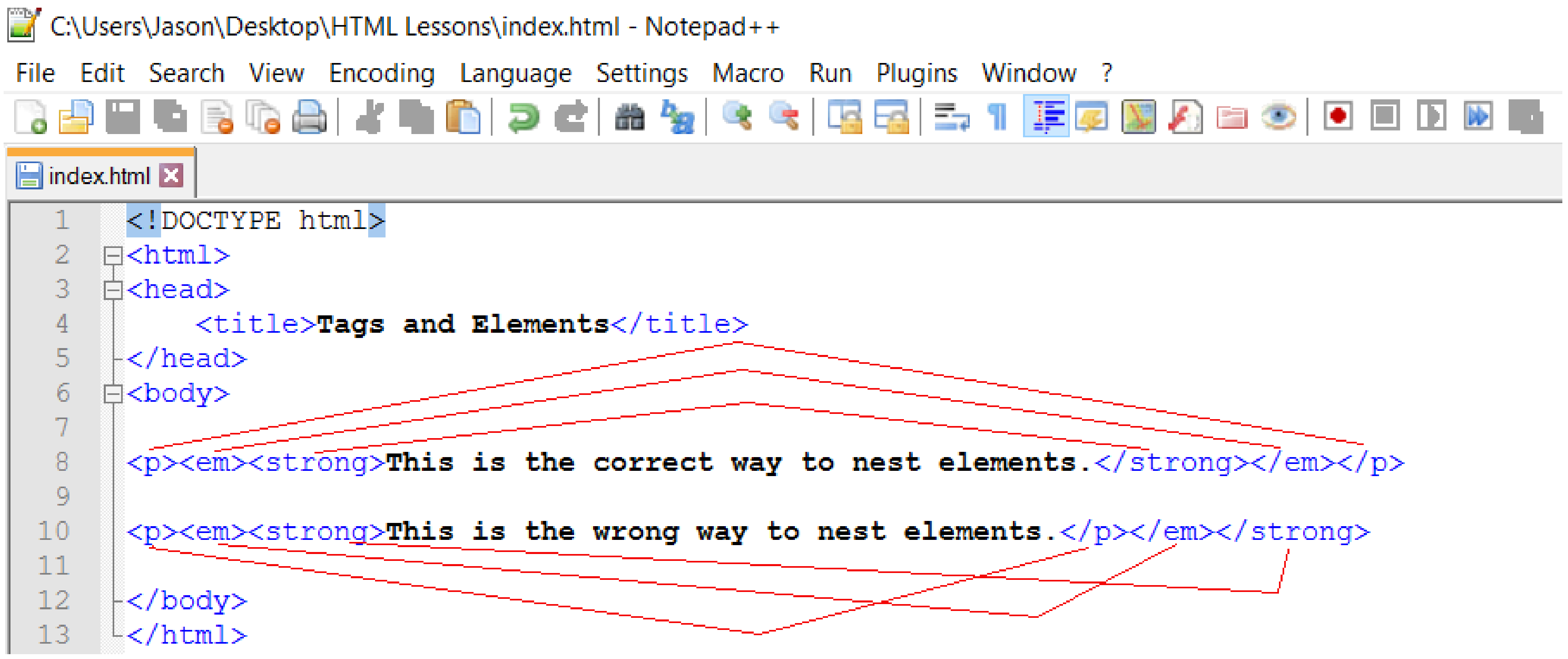
1. Dynamic: the web page requires some kind of interaction (a login form, for instance) , which then generates the page with the data. In this case we use the library Selenium

Sometimes we need to take into account particular characteristics of the page (frames, code, requests, etc.)

HTML: basic structure



HTML: nested labels


C:\Users\Jason\Desktop\HTML Lessons\index.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
index.html

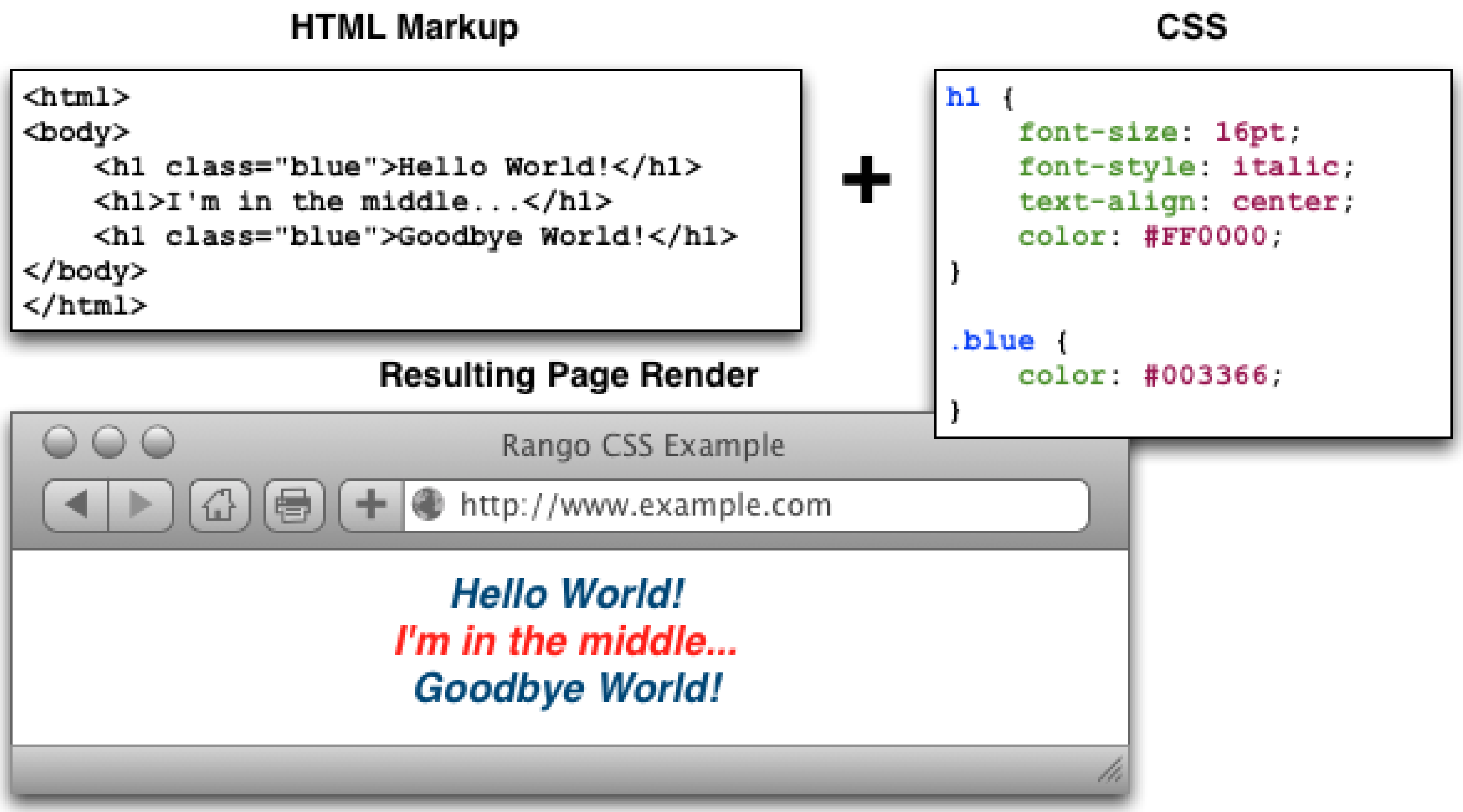
```
<!DOCTYPE html>
<html>
<head>
    <title>Tags and Elements</title>
</head>
<body>

<p><em><strong>This is the correct way to nest elements.</strong></em></p>

<p><em><strong>This is the wrong way to nest elements.</p></em></strong>

</body>
</html>
```

Web pages: HTML + CSS


HTML Markup

```
<html>
<body>
    <h1 class="blue">Hello World!</h1>
    <h1>I'm in the middle...</h1>
    <h1 class="blue">Goodbye World!</h1>
</body>
</html>
```

+

CSS

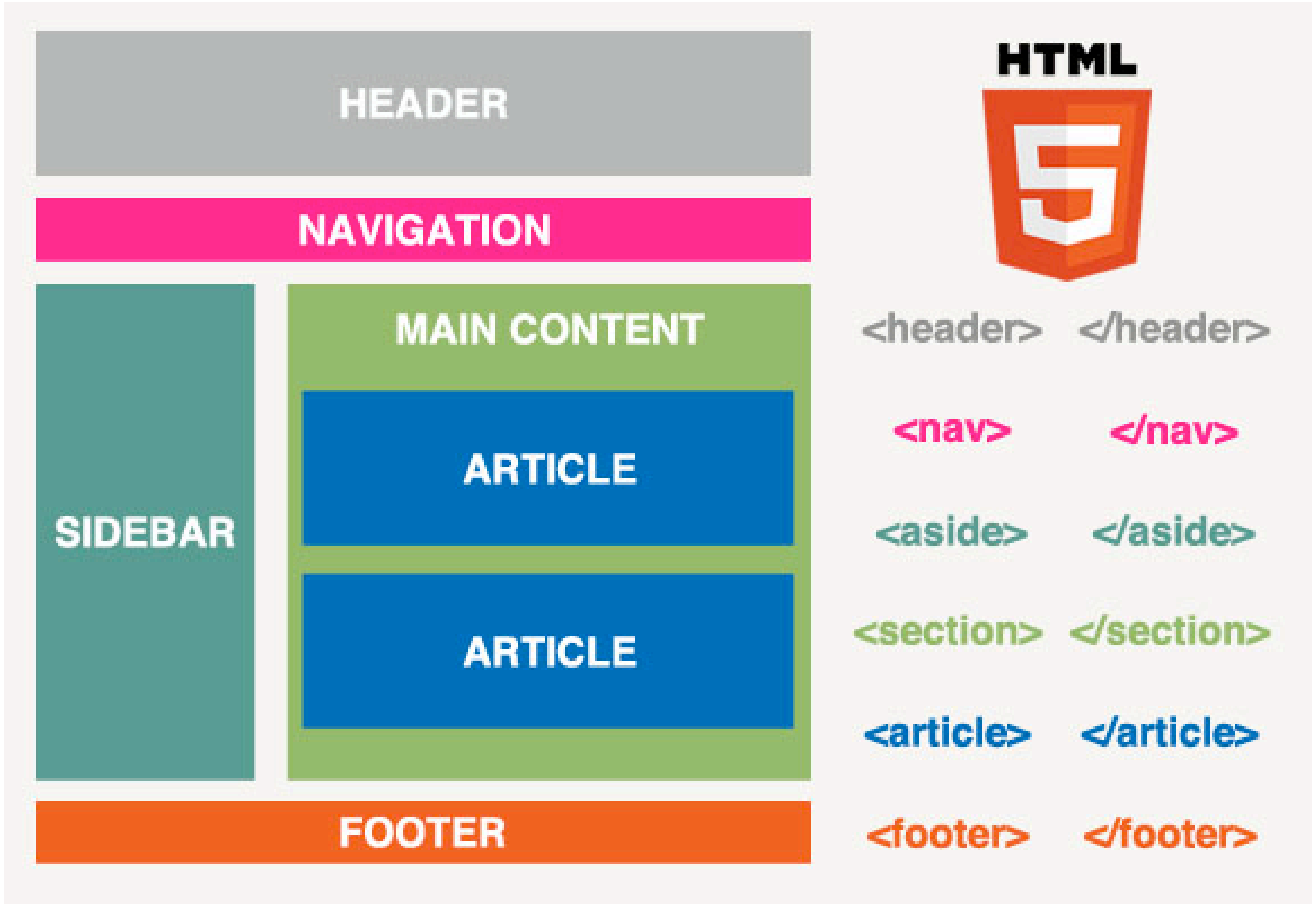
```
h1 {
    font-size: 16pt;
    font-style: italic;
    text-align: center;
    color: #FF0000;
}

.blue {
    color: #003366;
}
```

Resulting Page Render

Hello World!
I'm in the middle...
Goodbye World!

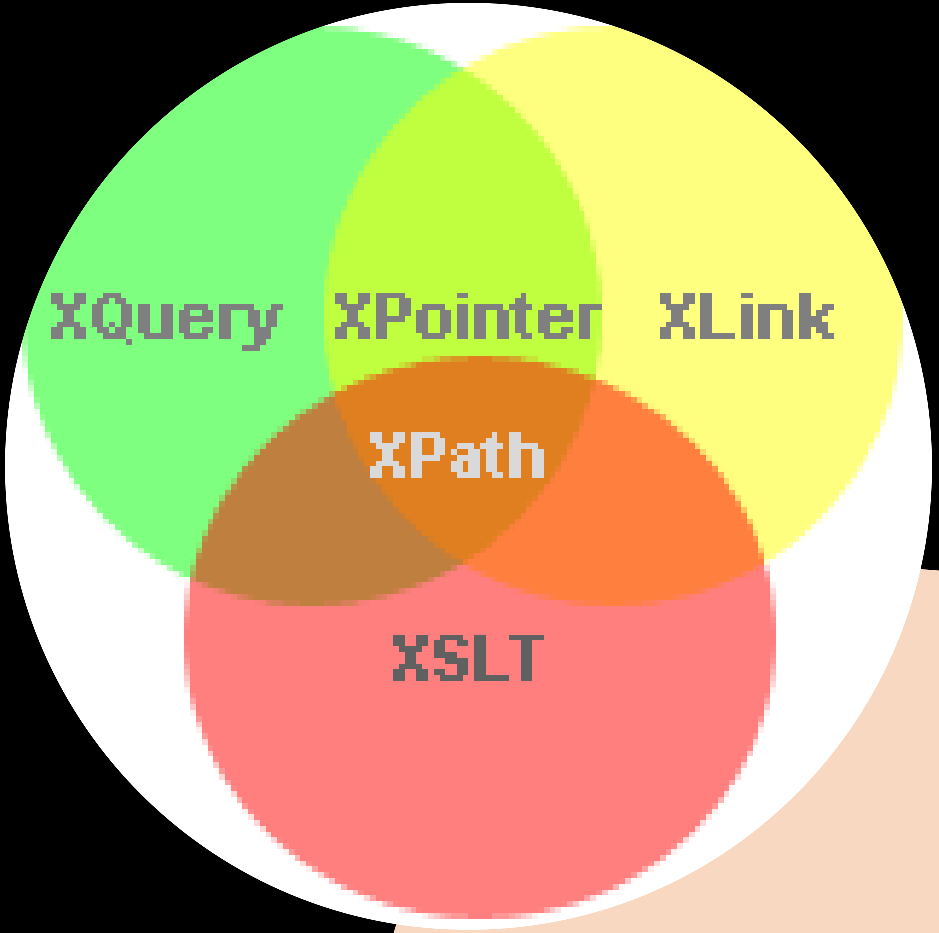
Web page: design point of view



Web Scraping

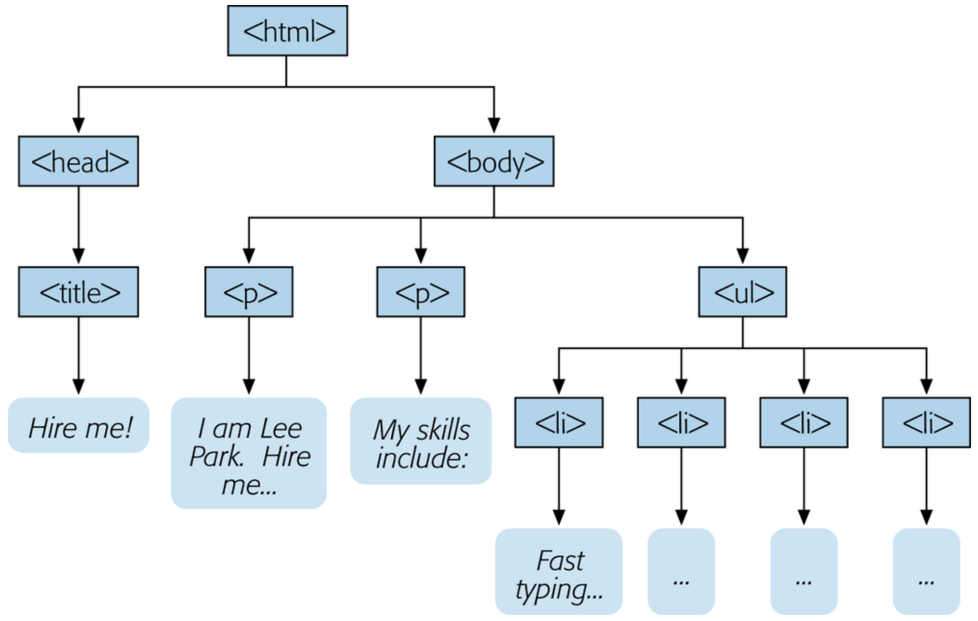
What is XPATH

- A simple language to traverse XML/HTML documents and to locate elements
- Part of selenium; however we should use it only if there are no more simple choice available
- Similar to the command line “cd” PATH to traverse folders in LINUX/Windows
- Defined in the W3C recommendations
- XPATH expressions are employed in other more complex languages



XPath: motivation

The idea is to consider the web page like a tree structure



The XPath expressions indicate how to traverse the tree top-down

Xpath: XML example

```xml
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>

<book>
  <title lang="en">Learning XML</title>
  <price>39.95</price>
</book>

</bookstore>
```

Source: https://www.w3schools.com/xml/xpath_syntax.asp

Xpath expressions

Expression	Description
nodename	Selects all nodes with the name "*nodename*"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Xpath expressions: Examples

bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore**Note:** If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//*[contains(@lang, ‘eng‘)]	Selects nodes with attribute lang and value ‘eng’

```xml
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>

<book>
  <title lang="en">Learning XML</title>
  <price>39.95</price>
</book>

</bookstore>
```

Web Scraping

Interacting with web pages: selenium

- Library for interacting with web pages. Available in Python, Java, etc
- Usually employed to perform regression tests of web sites
- It allows to open automatically a browser and:
  - Load URLs
  - Type automatically test, press buttons, etc

Selenium: open the browser

- We need an external file that acts like an intermediator between Python and the browser
- In Chrome is “chromedriver.exe” (see example)
- Before starting is usual to prepare some options and environment variables

```
driver = webdriver.Chrome(executable_path=chromedriver,
                          options=chrome_options)
```

The variable ‘driver’ is the Access to the browser

Selenium: loading web pages

driver.get(url) # url is the web page

It is usual to wait until some element has been loaded:

```
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
…
timeout = 5
try:
    element_present = EC.presence_of_element_located((By.ID,'ele'))
    WebDriverWait(driver, timeout).until(element_present)
except TimeoutException:
    print "Timed out waiting for page to load"
```

Selenium: locating elements

find_element_by_XX

Returns the first matching web element

Throws **NoSuchElementException** if the element is not found

find_element_by_id
find_element_by_name
find_element_by_xpath
find_element_by_link_text
find_element_by_partial_link_text
find_element_by_tag_name
find_element_by_class_name
find_element_by_css_selector

find_elements_by

Returns a list of multiple matching web elements

Returns an empty list if no matching element is found

find_elements_by_name
find_elements_by_xpath
find_elements_by_link_text
find_elements_by_partial_link_text
find_elements_by_tag_name
find_elements_by_class_name
find_elements_by_css_selector

Selenium: interaction

Once the element has been located we can…

- If it is a text field we can clear ir, enter text o press keys

```
element.clear()
element.send_keys("Bertoldo")
element.send_keys(" and some", Keys.ARROW_DOWN)
```

- If it is a button, we can press it

```
coord.click()
coord.send_keys(Keys.SPACE) # alternative
```

- More possibilities: list selection, swapping windows, cookies, etc.

Thanks!