



WEB SCRAPING



Contents

- Selenium
 - What it is
 - Why we need it
- Web Scraping Project Process
- Useful Notes for Implementation

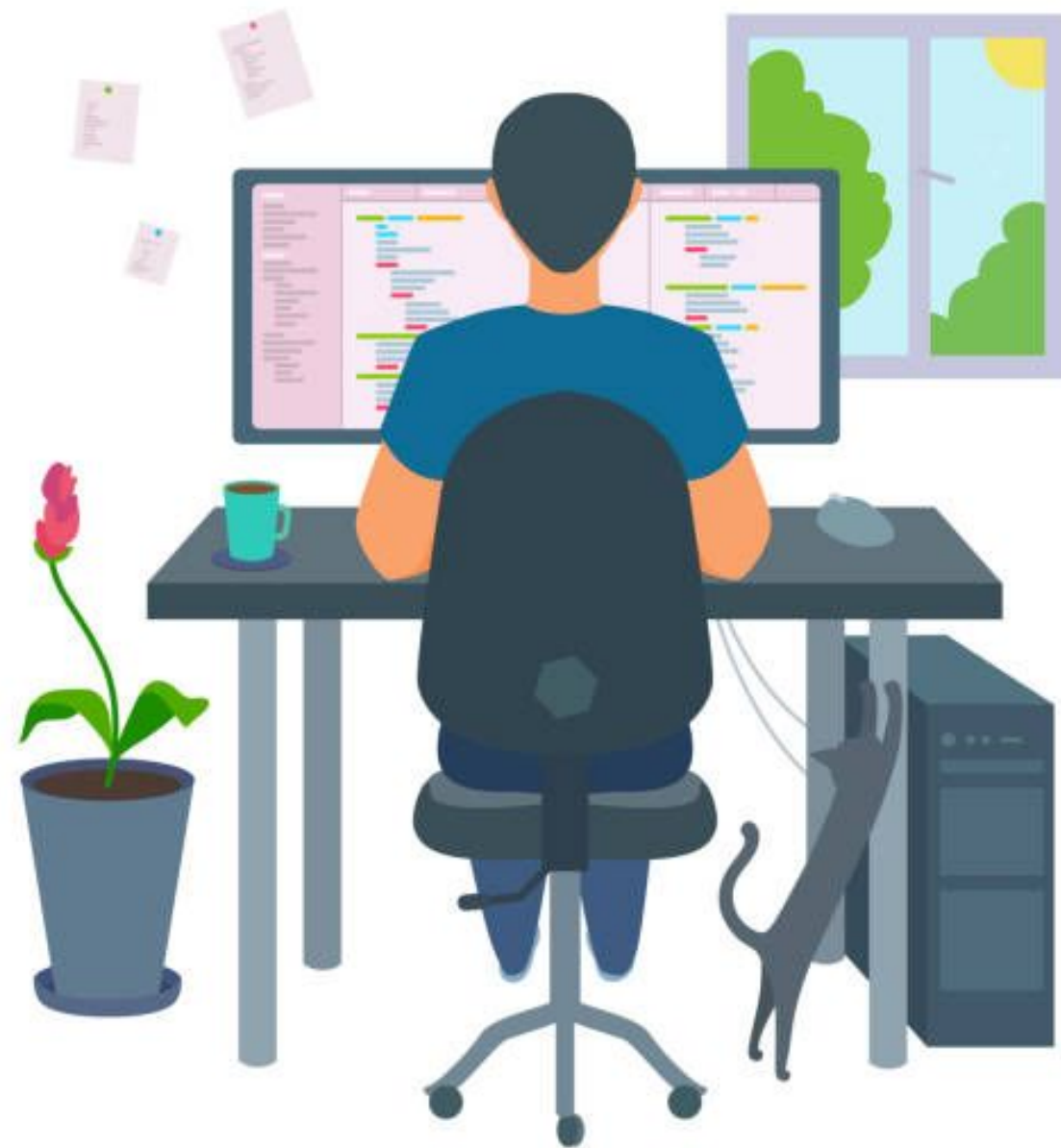


SELENIUM



What is Selenium?

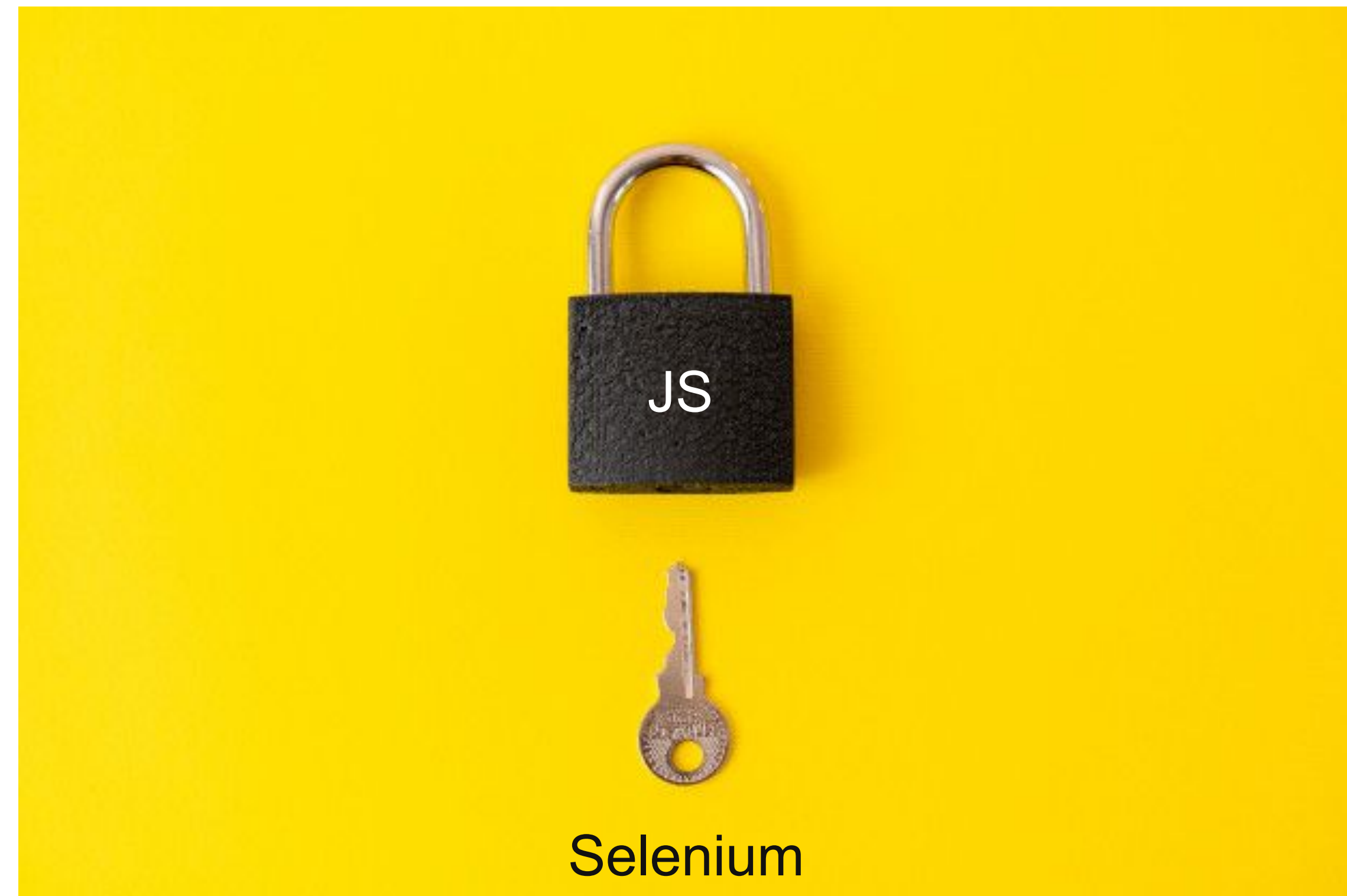
- A browser automation library
- Simulates human interaction with a browser





Why do we need Selenium?

- Websites have interactive elements nowadays
- Need Selenium to interact with the browser





WEB SCRAPING PROJECT PROCESS

Steps

1. Find website that you want to scrape
2. Inspect elements to scrape
 - a. Look at HTML code and pinpoint IDs and classes
 - b. Locate dynamic elements (buttons, text fields etc.)
3. Take note of patterns, especially URLs, IDs, classes and tags
4. Code!



Code

1. Make sure you've downloaded the right WebDriver according to your browser (if Selenium is required)
2. Install and import the required libraries (requests, BeautifulSoup etc.)
3. Get URL
4. Get URL's source code
5. Parse source code
6. Extract required data
7. Store data in required format





OUR TURN!



EXAMPLE



USEFUL NOTES FOR IMPLEMENTATION



Browser Developer Tools

- Ctrl + Shift + I and Inspect / Right-click page and Inspect

The screenshot displays the Chrome DevTools interface with three main panels visible:

- Elements Panel:** Shows the DOM tree with the `body` element selected. The HTML structure includes a `<doctype html>` declaration, a `<html>` tag with attributes `data-cast-api-enabled="true"` and `lang="en"`, and a `<head>` section. The `body` element has attributes `dir="ltr"`, `role="application"`, `class="docs-gm"`, and `itemscope itemtype="http://schema.org/CreativeWork/PresentationObject"`. It contains several `<iframe>` and `` elements, as well as a `<script>` block.
- Console Panel:** Shows a list of error messages, all of which are "Failed to load resource: net::ERR_CONTENT_LENGTH_MISMATCH".
- Styles Panel:** Shows the CSS styles for the selected `body` element. The styles include `background-color: white`, `color: black`, `font-weight: normal`, `font-size: 13px`, and `font-family: Roboto, RobotoDraft, Helvetica, Arial, sans-serif`.



Find Element/s using Selenium

- Documentation for finding elements

- `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`

To find multiple elements (these methods will return a list):

- `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`

Code

1. Make sure you've downloaded the right WebDriver according to your browser (if Selenium is required)
2. Install and import the required libraries (requests, BeautifulSoup etc.)
3. Get URL
4. Get URL's source code (if scraping directly, go to step 6 immediately)
5. Parse source code
6. Extract required data
7. Store data in required format



- **XML Path**
- Helps query markup documents like HTML
- Absolute XPath
 - `/html/body/div[1]/section/div/div[2]/div/form/div[2]/input[3]`
- Relative XPath
 - `//tagname[@attribute name= 'value']`
- **Example:**
 - `element = driver.find_element_by_xpath('//span[@class='text'])`



Common Interactions in Selenium

- **click:** clicks on an element, `click()`
- **input text:** simulates typing characters, `send_keys()`
- **submit:** submits a form, `submit()`
- **clear text:** clears all the text in a text field, `clear()`
- **wait:** simulates waiting. Used when waiting for certain elements to load, `WebDriverWait()`



WEB SCRAPING