

# Automated Data Collection

Webscraping with Python

Dr. Jakob Jünger, Chantal Gärtner



# Data collection via webscraping

## User Interface (Browser)



## HTML (Webscraping)

```

1 <!DOCTYPE html>
2 <html lang="de" class="no-js logged-in client-root">
3   <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=
6
7     <title>
8 Al Jazeera English (@aljazeeraenglish) • Instagram-Foto
9 </title>
10
11
12     <meta name="robots" content="noimageindex, noar
13     <meta name="apple-mobile-web-app-status-bar-sty
14     <meta name="mobile-web-app-capable" content="ye
15     <meta name="theme-color" content="#ffffff">
16     <meta id="viewport" name="viewport" content="wi
17     <link rel="manifest" href="/data/manifest.json"
18
19     <link rel="preload" href="/static/bundles/es6/C
20     <link rel="preload" href="/static/bundles/es6/Consumer.
21     <link rel="preload" href="/static/bundles/es6/ProfilePa
22     <link rel="preload" href="/static/bundles/es6/Vendor.js
23     <link rel="preload" href="/static/bundles/es6/de_DE.js/
24     <link rel="preload" href="/static/bundles/es6/ConsumerL
25     <link rel="preload" href="/static/bundles/es6/ConsumerL

```

## JSON (API)

```

{
  "graphql": {
    "user": {
      "biography": "Your Voice. Your Story. Your Platform.",
      "blocked_by_viewer": false,
      "restricted_by_viewer": false,
      "country_block": false,
      "external_url": "https://linkin.bio/aljazeeraenglish",
      "external_url_linkshimmed": "https://l.instagram.com/?u=https%3A%2F%2Flinkin.bio%2Faljazeeraenglish%3Fh&e=ATOSdXQPc_AJxLYM-XyDhXUkd-6URdU6A7Tdb1_vKIKr4JFgrbVRp_pYSC1a5TJ00ekDes4cG-xdKCApES5gquNECq-nuOUSVKGEgA&s=1",
      "edge_followed_by": {
        "count": 2409554
      },
      "fbid": "17841400896010580",
      "followed_by_viewer": false,
    }
  }
}

```

# Webscraping

## Classical approach:

Cut out HTML



## Interactive approach:

Remote browser control



## Commercial approach:

Buy data



# Structure of web pages

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript" src="actions.js" />
    <link type="text/css" href="styles.css" rel="stylesheet">
  </head>
  <body>
    lorem ipsum dolor
  </body>
</html>
```

HTML:  
Structure content

CSS:  
Design content

Javascript:  
Interact with content

# HTML: Elements + Attributes + Text

## Italian craftsman claims Putin's 'unique' oversized table

Putin's table prompted a slew of memes, becoming an unlikely star of efforts to ease the Ukraine crisis.

18 Feb 2022



## Can Mexico turbocharge pan- demic nearshoring by US firms?

The case is building for US companies to move operations they offshored to Asia closer to home.

18 Feb 2022



```
<div class="gc__content">
  <h3 class="gc__title">
    <a class="u-clickable-card__link"
      href="/news/2022/2/18/italian-craftsman-claims-putins
        unique-oversized-table">
    <span>
      Italian craftsman claims Putin's 'unique' oversized table
    </span>
  </a>
</h3>
<p>
  Putin's table prompted a slew of memes, becoming an
  unlikely star of efforts to ease the Ukraine crisis.
</p>
</div>
```

(HTML-Code shortened for illustration purposes)

# Some HTML-Elements

## Metadata:

`<head>` collection of metadata

`<title>` title of the document

## Sections:

`<body>` main contents of the document

`<section>` section of the document

`<h1>`, `<h2>`, ... headlines

## Grouped Contents

`<div>` container

`<p>` paragraph

## Links

`<a>` Hyperlink, refers via href-attribute to resource

## Lists

`<ul>` unordered list

`<ol>` ordered list

`<li>` entry of a list

## Tables

`<table>` table

`<tr>` row of a table

`<td>` cell of a table

# Practical session with the developer console

# Data extraction

## CSS selectors

- Used in web design
- Select HTML elements by name and attributes

Example:

`ul#contact li`

## XPath

- Used for XML handling
- Select HTML elements, attributes and text

Example:

`//ul/li/text()`

## Regular expressions

- Used for text extraction
- Select text snippets by search patterns

Example:

`[0-9]+ results`



# CSS selectors: best for HTML elements

Selection of HTML Elements ...

... by name:	<code>article</code>	selects <code>&lt;article&gt;My text&lt;/article&gt;</code>
... by ID:	<code>#contact</code>	selects <code>&lt;ul id="contact"&gt;</code>
... by class:	<code>.teaser</code>	selects <code>&lt;p class="heavy teaser"&gt;</code>

Selectors can be concatenated to select nested elements: `ul#contact li.ceo.vip a`

Recommendation: Use element name and class: `h2.search-counter_hits`

## XPath: best for text and attributes

Selection of HTML Elements ...

... by name: `//article` selects `<article>My text</article>`

... by attribute: `//.[@id== 'contact']` selects `<ul id="contact">`

Selection of ...

... attributes: `//a/@href` selects `<a href="https://www.example.org">`

... text: `//li/text()` selects `<li>Hello world</li>`

You can use functions to get plain text: `string(//tr/td)`

# Regex: best for extracting text snippets

Selection of ...

... numbers: `[0-9]+` selects Search results: 94 pages

... first words: `^[a-zA-Z]+` selects Search results: 94 pages

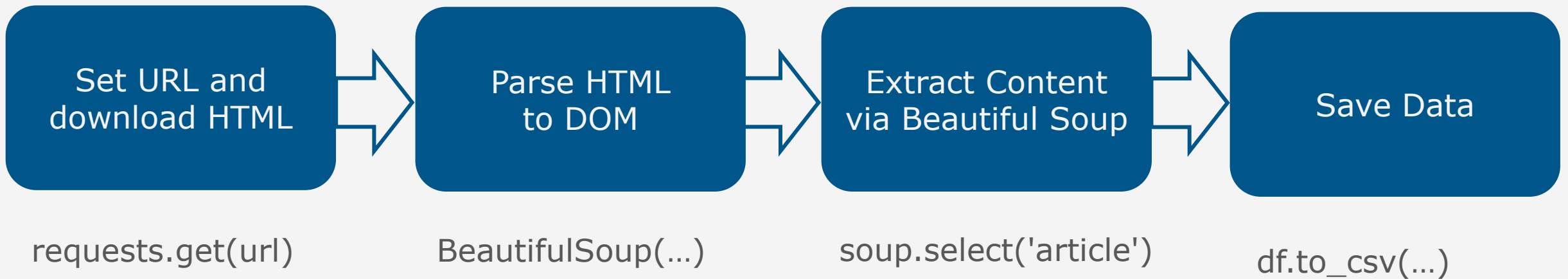
... last words: `[a-zA-Z]+$` selects Search results: 94 pages

Recommendation: use regexes in the last step,  
avoid using them for extracting HTML elements

# Practical session with CSS selectors

# Webscraping with BeautifulSoup

- Python library for extracting data from HTML and XML files
- Official documentation: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/#>

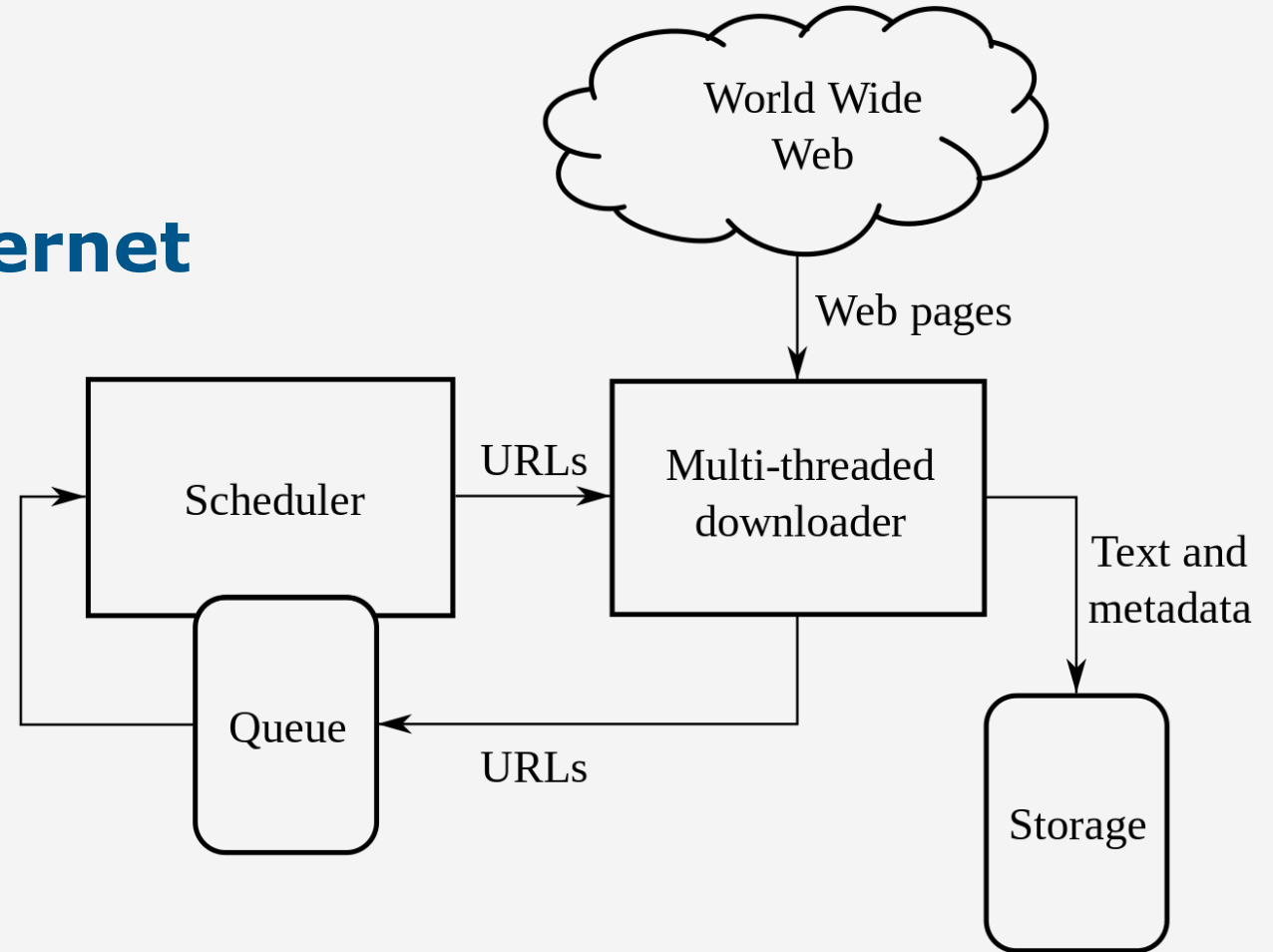


# Practical sessions with Python

# Webcrawling - how to download the Internet

## Repeat...

- Download page
- Save data / metadata
- Extract links
- Add links to queue



Source: [https://en.wikipedia.org/wiki/Web\\_crawler](https://en.wikipedia.org/wiki/Web_crawler)

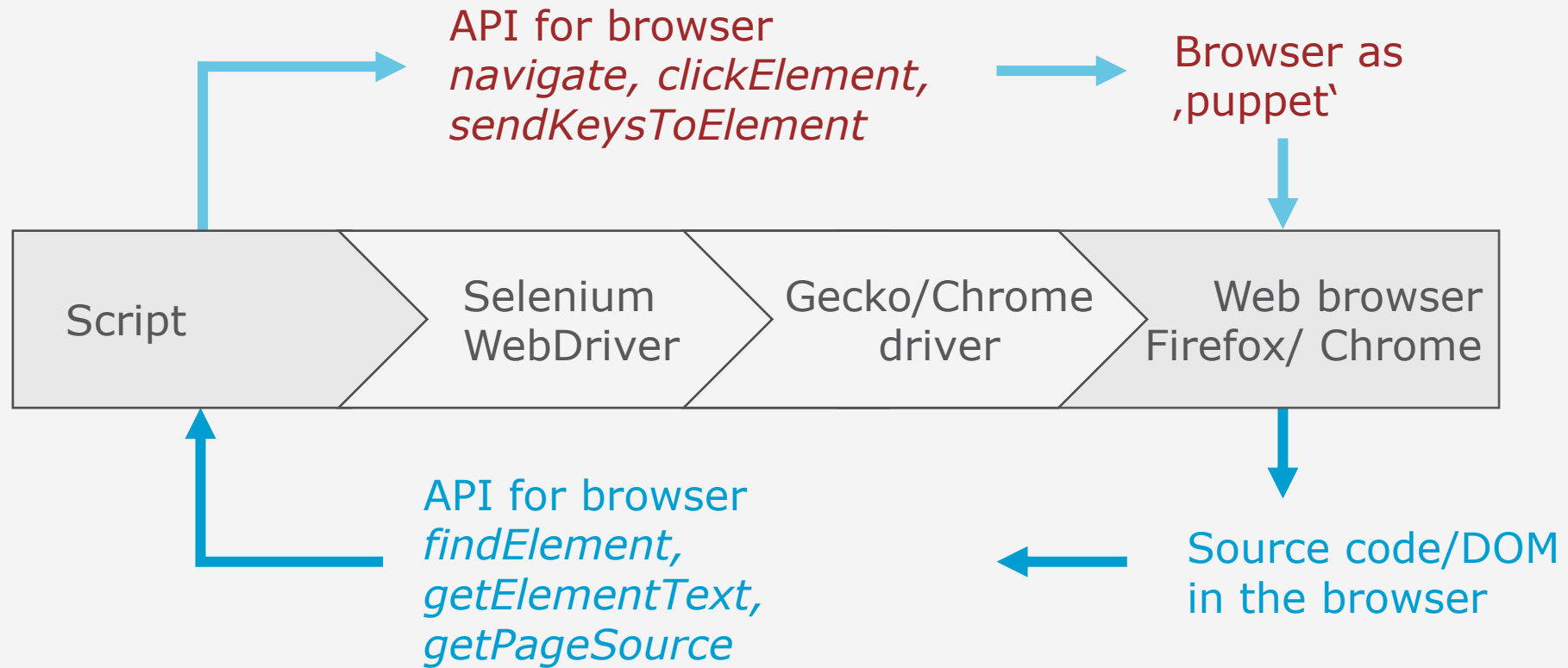
# Webscraping challenges

1. Authorization: Cookies and User-Agents
2. HTTP requests: request errors, redirects
3. Encoding: file types, unicode, entitites, emojis
4. Pipelines: databases, infinite loops, pagination
5. URL parsing and normalization
6. Rendered HTML (Javascript)
7. Boilerplate removal
8. Structured data (meta tags, microdata)
9. Policies, ethics, legal regulation



# Illustration with Facepager & Python

# Webscraping with Selenium



# Summary

## Classical web scraping

- More simple: no JavaScript, no interactivity, just download the source code
- Faster: Only one request necessary
- More complex: authentication by cookies and user agent headers

## Interactive web scraping

- More complex: Installation of a webdriver, manage interactivity
- Slower: The complete page including JavaScript, CSS-Styling and media is loaded
- Hybrid: manual and automated interaction
- More authentic: use the browser rendering