WWU
MÜNSTER

Automated Data Collection

Webscraping with Python

Dr. Jakob Jünger, Chantal Gärtner

Institut für
Kommunikationswissenschaft
Westfälische Wilhelms-Universität Münster
ifk

wissen.leben

Data collection via webscraping

User Interface
(Browser)

HTML
(Webscraping)

```
<!DOCTYPE html>
<html lang="de" class="no-js logged-in client-root">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=

        <title>
Al Jazeera English (@aljazeeraenglish) • Instagram-Fotc
</title>


        <meta name="robots" content="noimageindex, noar
        <meta name="apple-mobile-web-app-status-bar-sty
        <meta name="mobile-web-app-capable" content="ye
        <meta name="theme-color" content="#ffffff">
        <meta id="viewport" name="viewport" content="wi
        <link rel="manifest" href="/data/manifest.json"

        <link rel="preload" href="/static/bundles/es6/C
<link rel="preload" href="/static/bundles/es6/Consumer.
<link rel="preload" href="/static/bundles/es6/ProfilePa
<link rel="preload" href="/static/bundles/es6/Vendor.js
<link rel="preload" href="/static/bundles/es6/de_DE.js/
<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/aljazeer
        aenglish",
        "external_url_linkshimmed": "https://l.insta
        gram.com/?
        u=https%3A%2F%2Flinkin.bio%2Faljazeeraenglis
        h&e=ATOSdXQPc_AJxLYM-XyDHXUkd-
        6URdU6A7Tdb1_vKIKr4JFgrbVRp_pYSC1a5TJOOekDes
        4cG-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

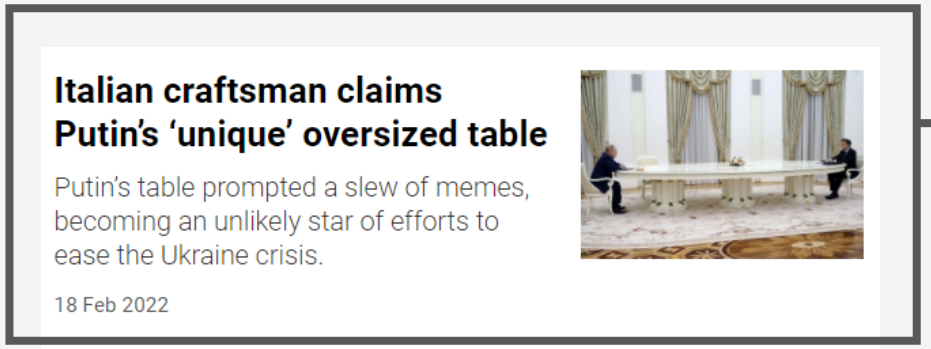
```
<!DOCTYPE html>
<html>
        <head>
                <script type="text/javascript" src="actions.js" />
                <link type="text/css" href="styles.css" rel="stylesheet">
        </head>
        <body>
                lorem ipsum dolor
        </body>
</html>
```

HTML:
Structure content

CSS:
Design content

Javascript:
Interact with content

HTML: Elements + Attributes + Text




```
<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

Grouped Contents

 container

<p> paragraph

Lists

 unordered list

 ordered list

 entry of a list

Sections:

<body> main contents of the document

<section> section of the document

<h1>, <h2>, … headlines

Links

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

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:	`article`	selects	`<article>My text</article>`
… by ID:	`#contact`	selects	`<ul id="contact"> … </ul>`
… by class:	`.teaser`	selects	`<p class="heavy teaser"> … </p>`

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`

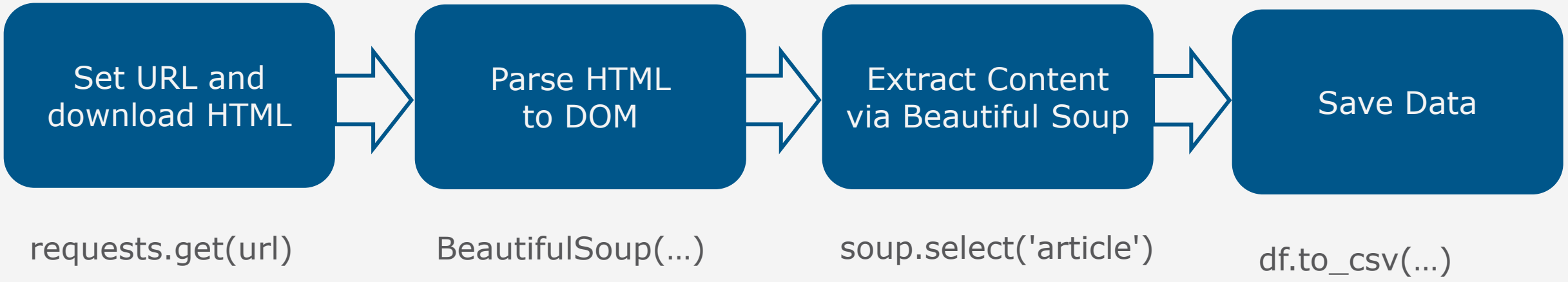
… last words: `[a-zA-Z]+$` selects `Search results: 94 pages`

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

Practical session with CSS selectors

Webscraping with Beautiful Soup

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

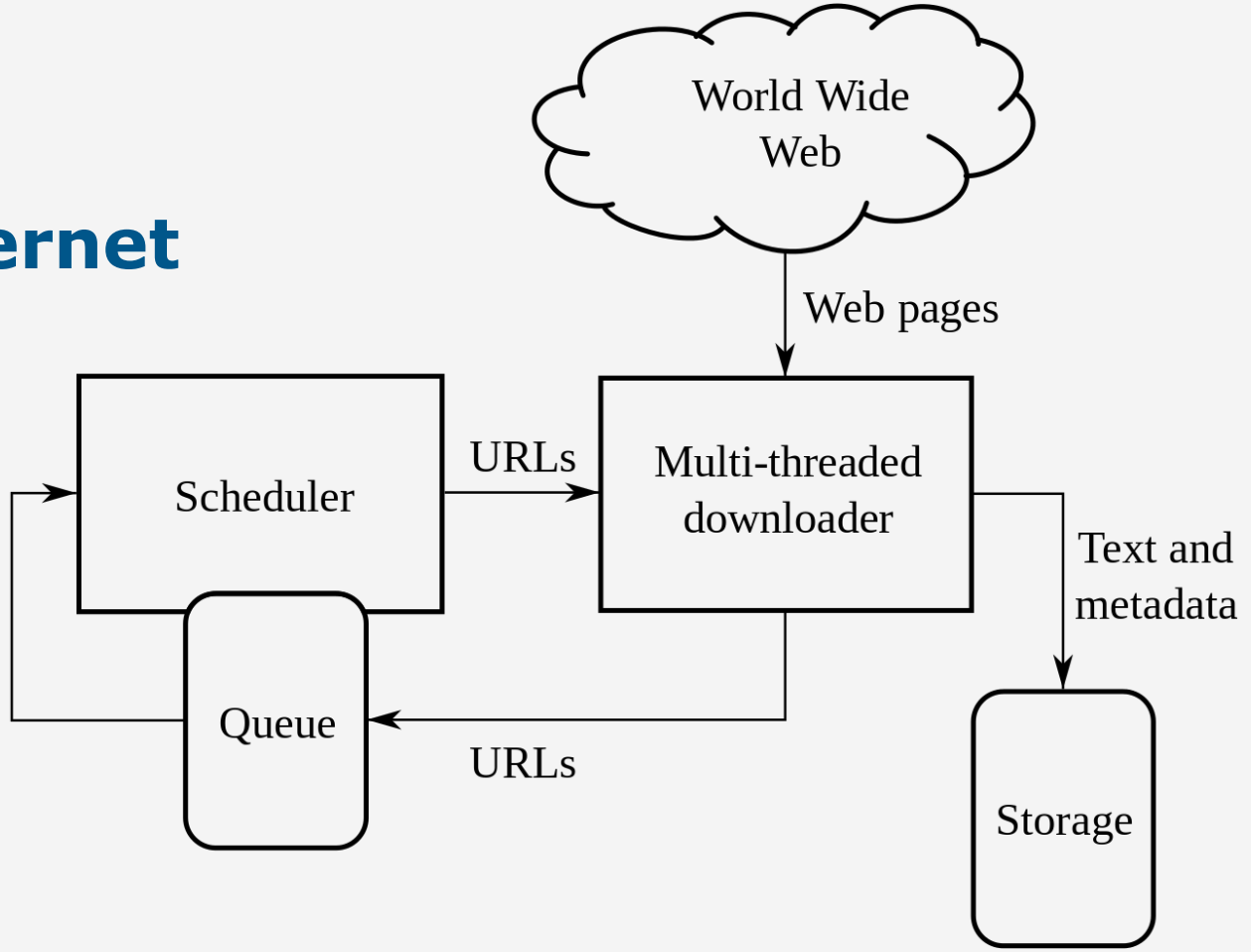


Practical sessions with Python

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

Webscraping challenges

Extraction

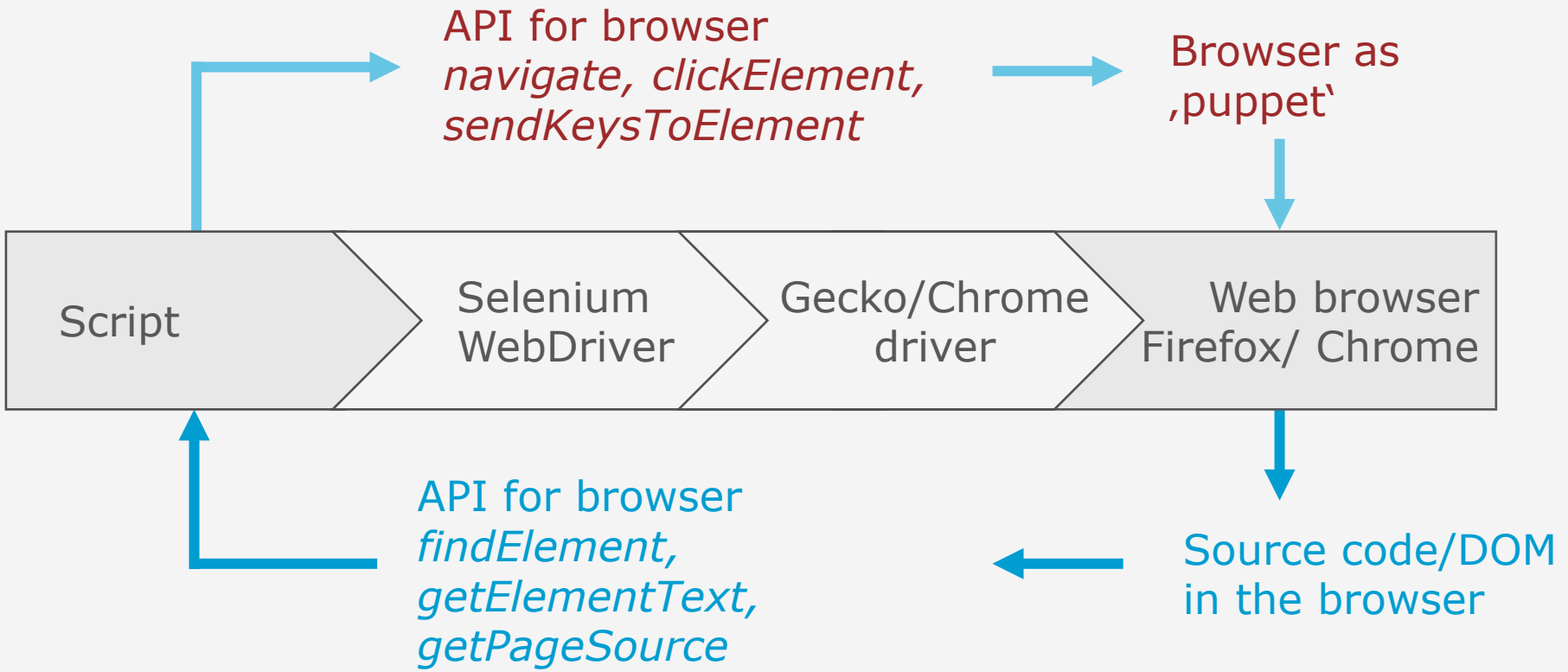
- Encoding: file types, unicode, entitites, emojis
- Pipelines: databases, infinite loops, pagination, duplicate detection
- URL parsing and normalization
- **Rendered HTML (Javascript)**
- Overloaded data vs. structured data (meta tags, microdata)
- Boilerplate removal

Resources, architecture of web & code

- Computer resources (out of memory)
- Authorization: Cookies and User-Agents, Captchas
- HTTP requests: request errors, redirects
- Unstable and dynamic websites
- **Coding skills: combination of different techniques**
- Writing replicable code
- **Policies, ethics, legal regulation**

Illustration with Facepager & Python

Webscraping with Selenium



Summary

Classical webscraping

- 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 webscraping

- 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

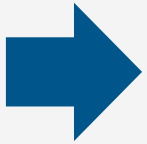
APIs

- Simple data structure
- Documented endpoints
- Different from the user interface
- Controlled by providers (authorization, rate limits)

How good or bad is webscraping?

- Ethics: AoIR Ethics Guidelines https://aoir.org/ethics/
- Copyright law
- Privacy law: General Data Protection Regulation (GDPR)

- Scientific research interest
- Freedom of research and teaching
- Social impact and societal mission



Be responsible!