Come2Data

# TRAINING: Python für die Datenanalyse in den Sozialwissenschaften

## Teil 2: Analyse verschiedener Datenarten und Web-Ressourcen

## *Einführung zu Web Scraping – Begriffe und Konzepte*

**SPEAKER:** Matthias Täschner

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

SACHSEN
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

# INHALT UND ZIELE

- **Crashkurs** Web Scraping - Begriffe und Kern-Konzepte für Einsteiger
- Web Scraping
  - Automatisierte Extraktion von Daten / Informationen aus Websites
  - Häufig mithilfe von Programmiersprachen und entsprechenden Bibliotheken
  - Unterschied zur Nutzung einer API:
    - Web Scraping extrahiert Daten aus gerenderter HTML
    - API stellt programmatischen Zugriff auf strukturierte Daten bereit
- Grundlagen, Konzepte, Methoden
  - HTTP Grundlagen
  - HTML und das Document Object Model (DOM)
  - Statische und Dynamische Websiten
  - Ethische und Rechtliche Aspekte
  - High-Level Workflow und Werkzeuge für Python

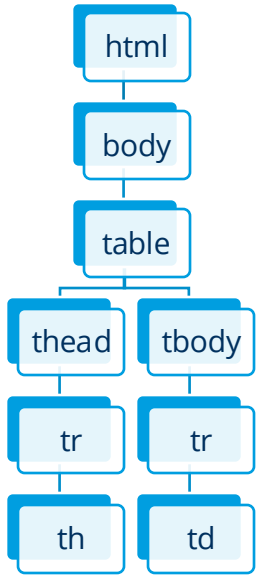
# HTTP Grundlagen

- „Hypertext Transfer Protocol“ für Informationsaustausch in Netzwerken
- Informationsaustausch über HTTP Requests (Anfrage) und Responses (Antwort)
  - URL mit Parametern
  - HTTP Methode
  - HTTP Header (z.B. Info zu UserAgent, Sessions, …)
  - HTTP Body (z.B. Payload)
  - HTTP Status-Code (z.b. 200 für „OK“, 404 für „Not Found“, …)
- HTTP Methoden (GET, PUT, POST, DELETE, …)
  - GET – Anforderung von Daten von einer bestimmten Ressource (z.b. Datei)

# HTML und das Document Object Model (DOM)

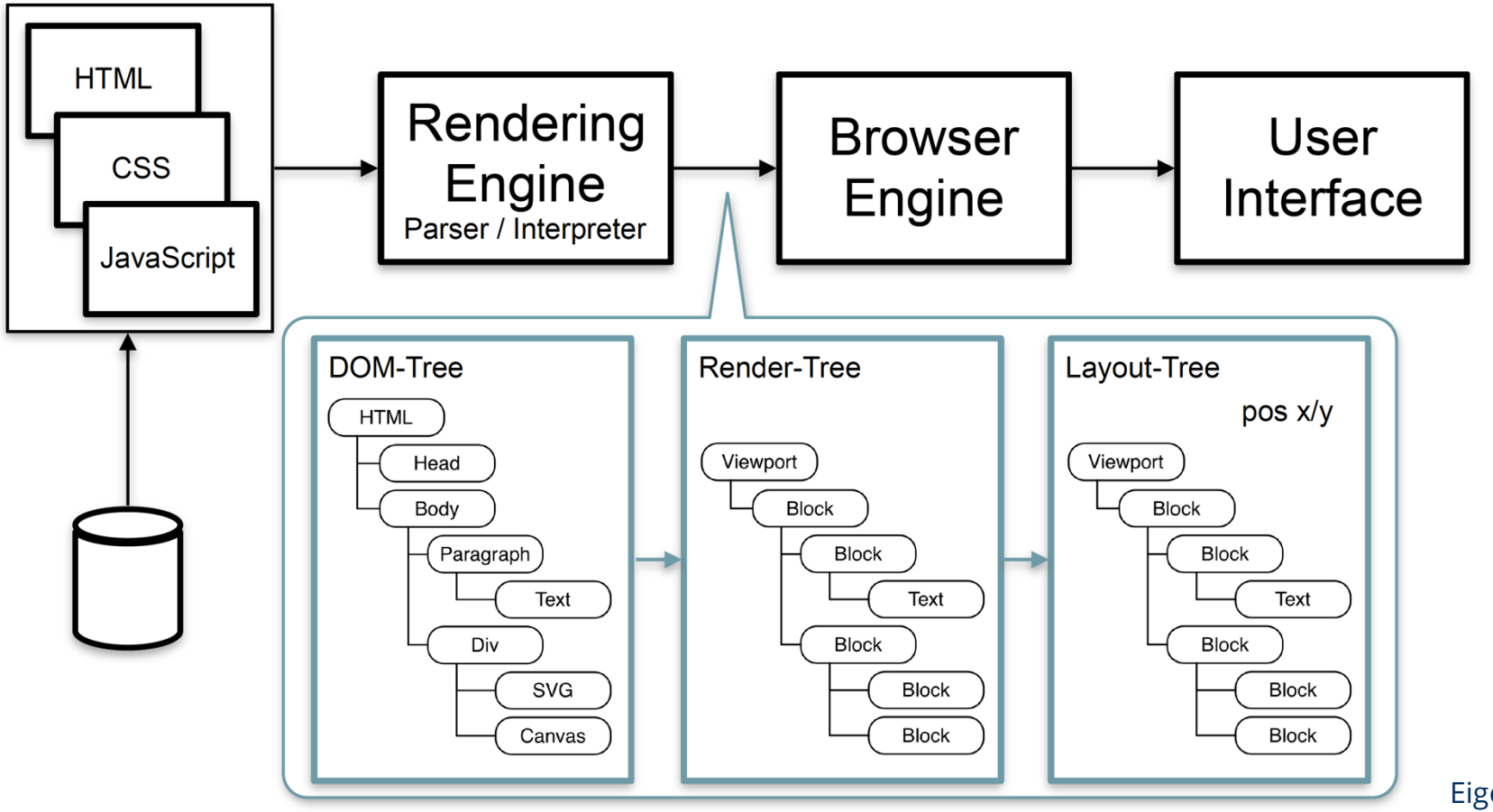
- Hypertext Markup Language, Auszeichnungssprache für elektronische Dokumente
  - Strukturiert Inhalte semantisch, formatiert diese aber nicht für Darstellung
  - Eigene Syntax, Definiert verschiedene HTML Elemente, Attribute, Events, …
- Document Object Model ist Spezifikation zur Darstellung von HTML als Baumstruktur
  - Jeder Knoten im Baum ist ein Objekt, das Teil des Dokuments representiert

```
<html>
 <table>
  <thead>
    <tr>
      <th>Vorname</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Matthias</td>
    </tr>
  </tbody>
 </table>
</html>
```



# HTML und das Document Object Model (DOM)

- Darstellung von HTML im Browser




Eigene Darstellung

# Statische und Dynamische Websites

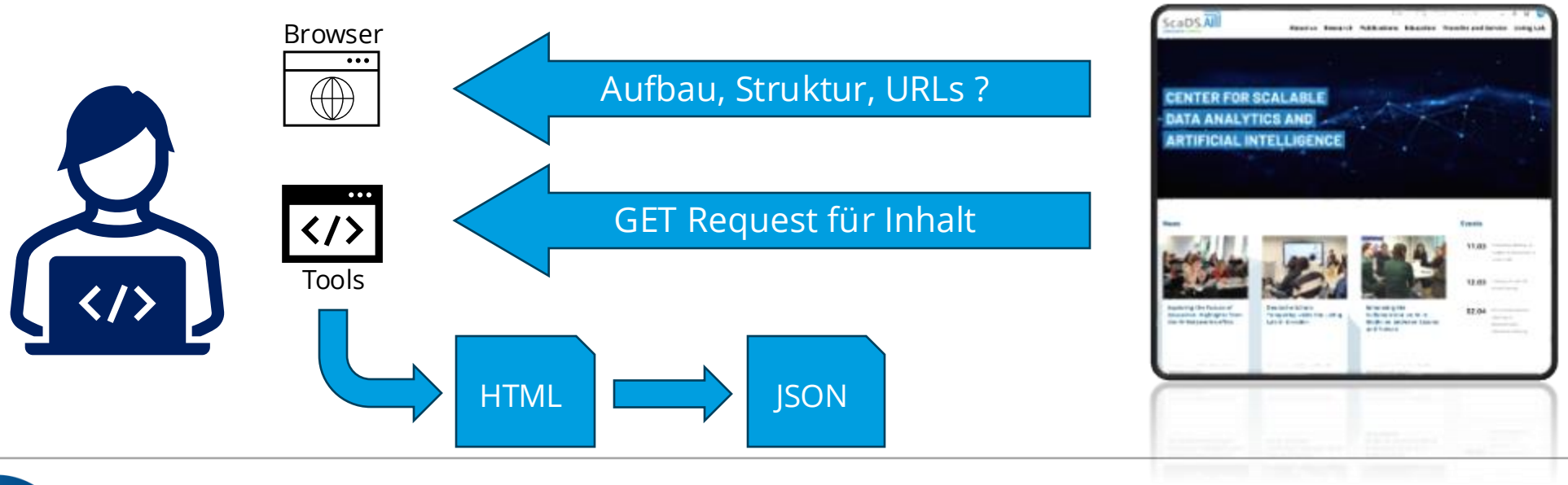
- Statische Webistes
  - Feste Inhalte und Strukturen
  - Unveränderlich auch bei Nutzerinteraktionen
  - Unabhängig von genutztem Browser, etc
- Dynamische Websites
  - Interaktive, veränderliche Inhalte
  - Werden erst bei Aufruf der Website im Browser verarbeitet (z.b. mit JavaScript))
  - Z.b. Füllen der Timeline bei Social Media, Laden von tagesaktuellen Inhalten aus Datenbank

# Rechtliche Aspekte und Etiquette

- Nutzungsbedingungen beachten (können z.b. Scraping verbieten)
- Copyright von Inhalten beachten
- Datenschutz bei Inhalten mit Personenbezug u.ä. beachten
- robots.txt regelt Bedingungen bei automatisiertem Zugriff, z.b. für Crawler
  - Z.b. https://www.zeit.de/robots.txt
- Evtl. Ausnahmen für Forschung und Lehre möglich, aber vorab zu prüfen
- Automatisierte regelmäßige Zugriffe erzeugen Traffic / Last auf den Servern

# High-Level Workflow

- Aufbau, Struktur und Mechanismen der Zielseite identifizieren
- Benötigte URLs zu den gewünschten Informationen zusammentragen
- Mit HTTP GET Request den gewünschten Inhalt von der Website beziehen
- Den HTML Code der Website entlang der DOM-Baumstruktur parsen
- Gewünschte Daten extrahieren und lokal in strukturiertem Format speichern (JSON, CSV, ...)



# Werkzeuge für Web Scraping

- Für allgemeine Struktur einer Website: Browser-Tools
- Für Automatisierung mit Python
  - requests: HTTP Requests
  - beautifulsoup: Parsen und Info-Extraktion von HTML und XML
  - parsel: Parsen und Info-Extraktion von HTML, XML, JSON
  - scrapy: High-level Web Scraping inkl. HTTP Requests, Parsen, Info-Extraktion und –Verarbeitung
  - selenium: Automatisierung von Browser-Interaktionen, insb. für dynamische Websites