

DH Einführungen

HTML, CSS, JavaScript & Co

Den Aufbau moderner Webseiten verstehen, um Informationen
automatisiert auszulesen

DH Einführungen

- **Was sind die Digital Humanities?**
 - Interdisziplinäres Feld.
 - Bearbeitung geistes- und sozialwissenschaftlicher Fragen mit digitalen Methoden
 - Digitale Editionen, digitale Sprach- und Literaturanalyse, Georeferenzierung von Daten, Netzwerkanalysen, Linked Open Data,...
- **Digital Humanities an der RUB**
 - **Netzwerk Digital Humanities** (<https://dh-netzwerk.blogs.ruhr-uni-bochum.de/>)
 - **Digital Humanities Center** der UB (<https://dh.rub.de>)
 - **Methodenzentrum** (<https://methodenzentrum.ruhr-uni-bochum.de/>)
 - **DH Ruhr** (<https://dh-ruhr.ruhr-uni-bochum.de/>)

DH Einführungen – Ziele der Veranstaltungsreihe

- **Grundverständnis von für die Digital Humanities wichtigen Formaten, Techniken und Tools**
- **Einblick in die Möglichkeiten der jeweiligen Techniken**
- **Mehr gibt es**
 - Im Lehrplan der Institute
 - Im Angebot des Methodenzentrums
 - Selbstlernmaterialien im Internet
 - Co-working & Coding im DH Space (alle 14-Tage Donnerstags)

DH Einführungen – Ziele dieser Veranstaltung

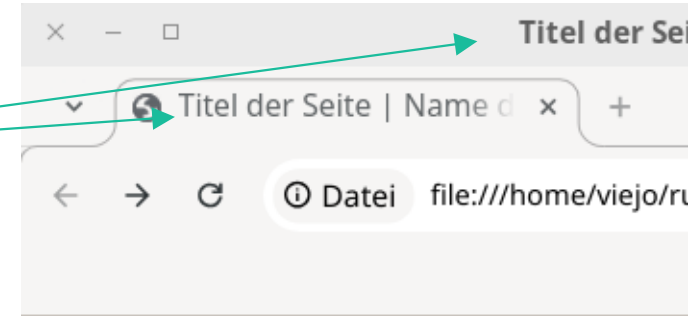
- **Verständnis des Aufbaus moderner Webseiten**
- **Kennenlernen der drei Formate HTML, CSS und JavaScript**
- **Anwendung der Entwicklertools im Browser (Firefox)**
- **Kennenlernen der Möglichkeiten des Webscraping mit Python**

Trennung von Struktur, Layout und Funktionslogik

- **HTML = HyperText Markup Language**
 - Dokument Struktur und Inhalt
 - Textauszeichnungssprache, in aktueller Version (HTML5) XML ähnlich.
- **CSS = Cascading Style Sheets**
 - Layout, positioniert Inhalt, definiert Schriftfont, -größe usw.
 - Gestaffelte Definitionen, Eigenschaften vererben & überschreiben
- **JS = JavaScript**
 - Seitendynamik: Verarbeitung von Eingaben und Umgebungsbedingungen
 - Nachladen von Inhalten

HTML

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Titel der Seite | Name der Website</title>
  </head>
  <body>
    <!-- Sichtbarer Dokumentinhalt im body -->
    <h1>Meine Überschrift</h1>
    <p id="p0815">Mein Absatz mit Text.</p>
    <p>Ein Absatz mit einem <a href="https://dh.rub.de">Link</a></p>
    <p class="infobox">Text für eine Infobox</p>
  </body>
</html>
```



Meine Überschrift

Mein Absatz mit Text.

Ein Absatz mit einem [Link](https://dh.rub.de)

Text für eine Infobox

HTML

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Titel der Seite | Name der Website</title>
  </head>
  <body>
    <!-- Sichtbarer Dokumentinhalt im body -->
    <h1>Meine Überschrift</h1>
    <p id="p0815">Mein Absatz mit Text.</p>
    <p>Ein Absatz mit einem <a href="https://dh.rub.de">Link</a></p>
    <p class="infobox">Text für eine Infobox</p>
  </body>
</html>
```

Aufbau eines HTML Elements:

Startmarke des Elements (Starttag)					Elementinhalt (optional)	Endmarke des Elements
	Elementname	Attribut (optional)		Attributwert (optional)		
<	p	id	=	"p0815"	> Mein Absatz mit Text.	</p>

Der Elementinhalt kann aus Text und / oder Elementen bestehen

Typische Elementnamen:

p = paragraph / Absatz
h1 = heading1 / Überschrift 1. Ordnung
h2 = heading2 / Überschrift 2. Ordnung
div = division / Gruppe von Elementen
span = span / Textbereich
a = anchor / Ankerelement für interne und externe Verlinkung
table = Tabellen
thead = table head / Tabellenkopf
tbody = table body / Tabellenkörper
tr = table row / Zeile einer Tabelle
td = table data / Einzelnes Feld in einer Tabellenzeile

Typische Attribute:

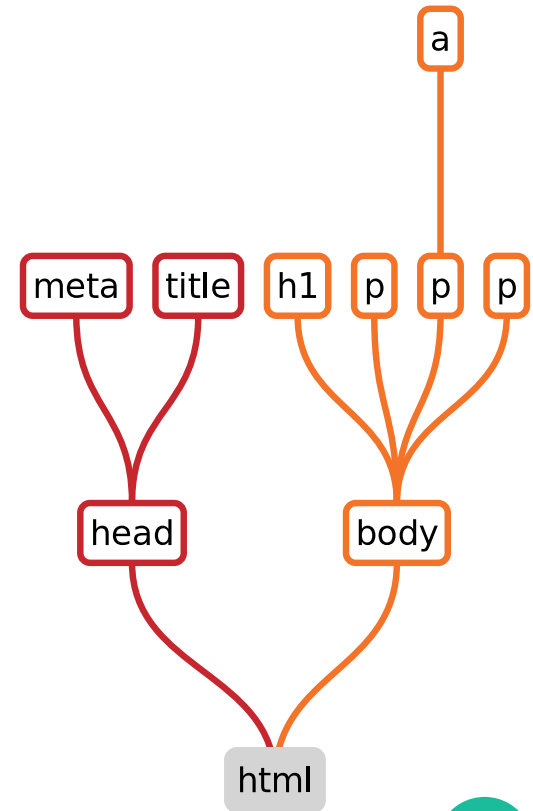
id = eindeutiger Identifikator eines Elementes
class = Bezeichnung für eine Gruppe von Elementen, auf die gleiche Layoutanweisungen (CSS) angewendet werden sollen
href = externes Linkziel (URL)

HTML

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Titel der Seite | Name der Website</title>
  </head>
  <body>
    <!-- Sichtbarer Dokumentinhalt im body -->
    <h1>Meine Überschrift</h1>
    <p id="p0815">Mein Absatz mit Text.</p>
    <p>Ein Absatz mit einem <a href="https://dh.rub.de">Link</a></p>
    <p class="infobox">Text für eine Infobox</p>
  </body>
</html>
```

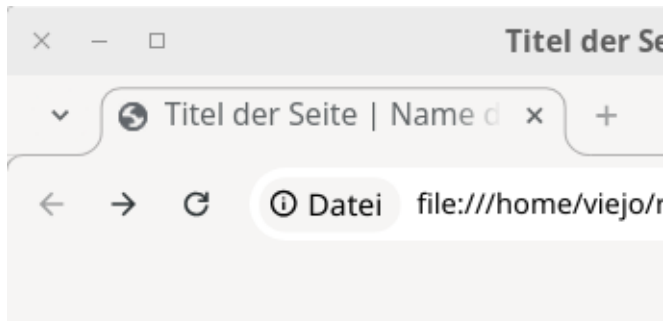
DOM (Document Object Model)

- Im Browser wird der HTML Code als Elementbaum verarbeitet
- JavaScript kann auf diesen Elementbaum gezielt zugreifen und ihn modifizieren
- HTML Elemente dürfen ineinander verschachtelt sein, sich aber nie überschneiden



CSS

```
h1 {  
  font-family: arial, helvetica, sans-serif;  
  font-size: 24pt;  
}  
  
#p0815 {  
  font-style: italic;  
  font-size: 120%;  
}  
  
a {  
  font-variant-caps: small-caps;  
  color: red;  
}  
  
.infobox {  
  line-height: 150%;  
  margin-left: 2em;  
  padding: 1em;  
  border: 3px solid blue;  
  background-color: #33ccff;  
  display: inline-block;  
}  
  
p.infobox {  
  font-weight: bold;  
  font-family: arial, helvetica, sans-serif;  
}
```

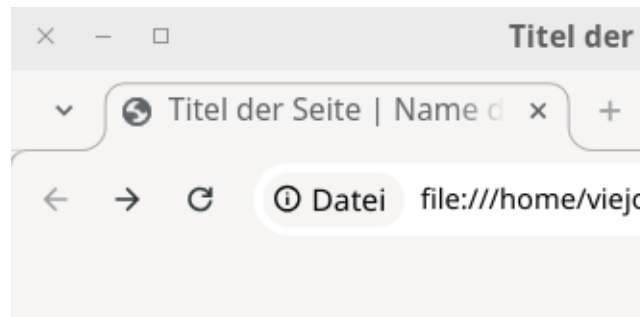


Meine Überschrift

Mein Absatz mit Text.

Ein Absatz mit einem [LINK](#)

Text für eine Infobox



Meine Überschrift

Mein Absatz mit Text.

Ein Absatz mit einem [Link](#)

Text für eine Infobox

CSS

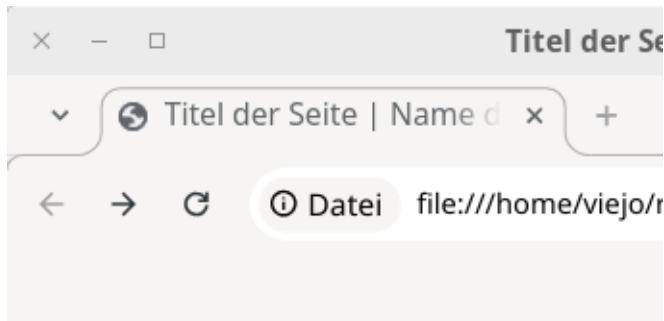
```
h1 {
  font-family: arial, helvetica, sans-serif;
  font-size: 24pt;
}

#p0815 {
  font-style: italic;
  font-size: 120%;
}

a {
  font-variant-caps: small-caps;
  color: red;
}

.infobox {
  line-height: 150%;
  margin-left: 2em;
  padding: 1em;
  border: 3px solid blue;
  background-color: #33ccff;
  display: inline-block;
}

p.infobox {
  font-weight: bold;
  font-family: arial, helvetica, sans-serif;
}
```



Meine Überschrift

Mein Absatz mit Text.

Ein Absatz mit einem [LINK](#)

Text für eine Infobox

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet"
          type="text/css"
          href="beispiel1.css"/>
    <title>
      Titel der Seite |
      Name der Website
    </title>
  </head>
  <body>
    <h1>Meine Überschrift</h1>
    <p id="p0815">
      Mein Absatz mit Text.
    </p>
    <p>Ein Absatz mit einem
      <a href="https://dh.rub.de">
        Link
      </a>
    </p>
    <p class="infobox">
      Text für eine Infobox
    </p>
  </body>
</html>
```

CSS

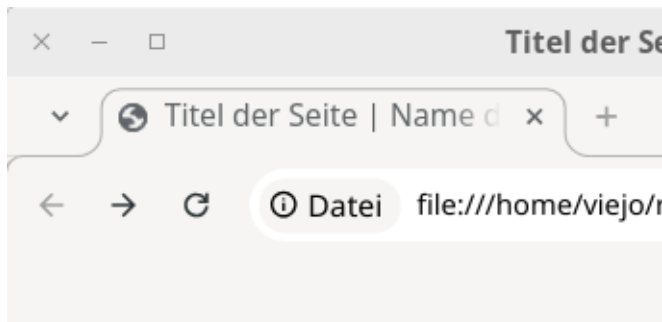
```
h1 {  
  font-family: arial, helvetica, sans-serif;  
  font-size: 24pt;  
}
```

```
#p0815 {  
  font-style: italic;  
  font-size: 120%;  
}
```

```
a {  
  font-variant-caps: small-caps;  
  color: red;  
}
```

```
.infobox {  
  line-height: 150%;  
  margin-left: 2em;  
  padding: 1em;  
  border: 3px solid blue;  
  background-color: #33ccff;  
  display: inline-block;  
}
```

```
p.infobox {  
  font-weight: bold;  
  font-family: arial, helvetica, sans-serif;  
}
```



Meine Überschrift

Mein Absatz mit Text.

Ein Absatz mit einem [LINK](#)

Text für eine Infobox

```
<!DOCTYPE html>  
<html lang="de">  
  <head>  
    <meta charset="utf-8">  
    <link rel="stylesheet"  
          type="text/css"  
          href="beispiel1.css"/>  
    <title>  
      Titel der Seite |  
      Name der Website  
    </title>  
  </head>  
  <body>  
    <h1>Meine Überschrift</h1>  
    <p id="p0815">  
      Mein Absatz mit Text.  
    </p>  
    <p>Ein Absatz mit einem  
      <a href="https://dh.rub.de">  
        Link  
      </a>  
    </p>  
    <p class="infobox">  
      Text für eine Infobox  
    </p>  
  </body>  
</html>
```

CSS und HTML

- **HTML** gibt die **Struktur** einer Seite vor
- **CSS** steuert das **Erscheinungsbild** der Seite
- Damit im CSS gezielt auf Seitenelemente verwiesen werden kann, müssen im HTML passende **Elementnamen** und **Attribute** vergeben werden
- Die Elementnamen und Attribute im **Strukturbaum** des HTML werden wir uns später für das **gezielte Auslesen von Seiteninhalten** zu Nutze machen.

JavaScript

- Ist eine **Programmiersprache**, die von jedem modernen **Webbrowser** ausgeführt werden kann
- Kann die Struktur, den Inhalt und auch die Attribute des **HTML** der Seite **verändern**
- Ermöglicht es auf **Eingaben** der Benutzer*innen oder auf die Browserumgebung (z.B. Displaygröße) zu **reagieren**
- Kann Benutzereingaben an den **Server senden** und vom Server Inhalte **empfangen** und nachladen

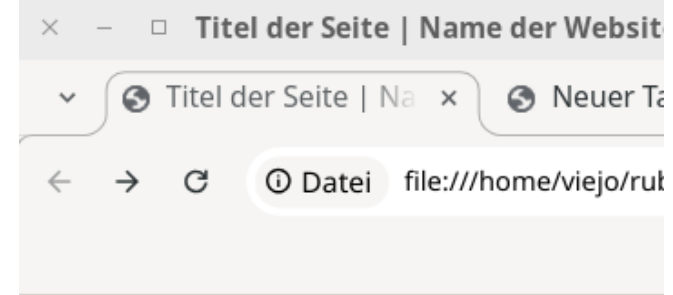
JavaScript

HTML Datei:

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css"
          href="beispiel1.css"/>
    <title>Titel der Seite | Name der Website</title>
  </head>
  <body>
    <!-- Sichtbarer Dokumentinhalt im body -->
    <h1>Meine Überschrift</h1>
    <p id="p0815">Mein Absatz mit Text.</p>
    <p>Ein Absatz mit einem <a
      href="https://dh.rub.de">Link</a></p>
    <p class="infobox">Text für eine Infobox</p>
    <script src="beispiel1.js"></script>
  </body>
</html>
```

JavaScript Datei:

```
// Ändere den Text des Elements mit der ID p0815
document.getElementById('p0815').innerHTML = "Ich bin ein neuer Text für diesen Absatz";
```



Meine Überschrift

Ich bin ein neuer Text für diesen Absatz

Ein Absatz mit einem LINK

Text für eine Infobox

JavaScript

- Wenn Webseiten mit JavaScript **Inhalte dynamisch nachladen**, wird es **schwieriger**, deren Inhalte mit einem Skript **auszulesen**.
- Manchmal ist außer der Navigation gar **kein Inhalt im Quellcode** der Website zu erkennen
- Andererseits können wir manchmal im Browser die **Serverschnittstelle** zum Nachladen der Inhalte aufspüren und so gezielt **eigene Abfragen an den Server** stellen

Beispielaufgabe:

Sammele die Abstracts aller Artikel der aktuellen Ausgabe von DH Quarterly (<https://www.digitalhumanities.org/dhq>) und Speichere sie zusammen mit dem Titel des Artikels und den Infos zur Autor*innenschaft

Seite analysieren

The screenshot shows a web browser displaying a list of articles. The left sidebar contains a table of contents with links like 2023: 17.2, 2023: 17.1, 2022: 16.4, etc. The main content area shows the text of an article titled "Introduction to the Special Issue: Using Visual AI Applied to Digital Archives" by Lise Jaillant. Below this, there are sections for "Articles" and "Augmenting Access to Embodied Knowledge Archives: A Computational Framework" by Giacomo Alliaia and Yumeng Hou. The bottom of the page shows the Chrome DevTools Inspector with the HTML structure of the article entry selected. The HTML code shows a

```
<div class="articleInfo" style="margin:0 0 1em 0;">
  <span class="monospace">[en] </span>
  <a href="/dhq/vol/18/2/000722/000722.html">...</a>
  <div style="padding-left:1em; margin:0;text-indent:-1em;">...</div>
  <span class="viewAbstract">...</span>
  <span class="Z3988"
    title="url_ver=Z39.88-2004&ctx_ver=Z39.88-2004&rft_val_fmt=info%3Ao...
    como%20Alliaia&rft.au=Yumeng%20Hou&rft.au=Sarah%20Kenderdine">...</span>
</div>
<div class="articleInfo" style="margin:0 0 1em 0;">...</div>
<div class="articleInfo" style="margin:0 0 1em 0;">...</div>
```

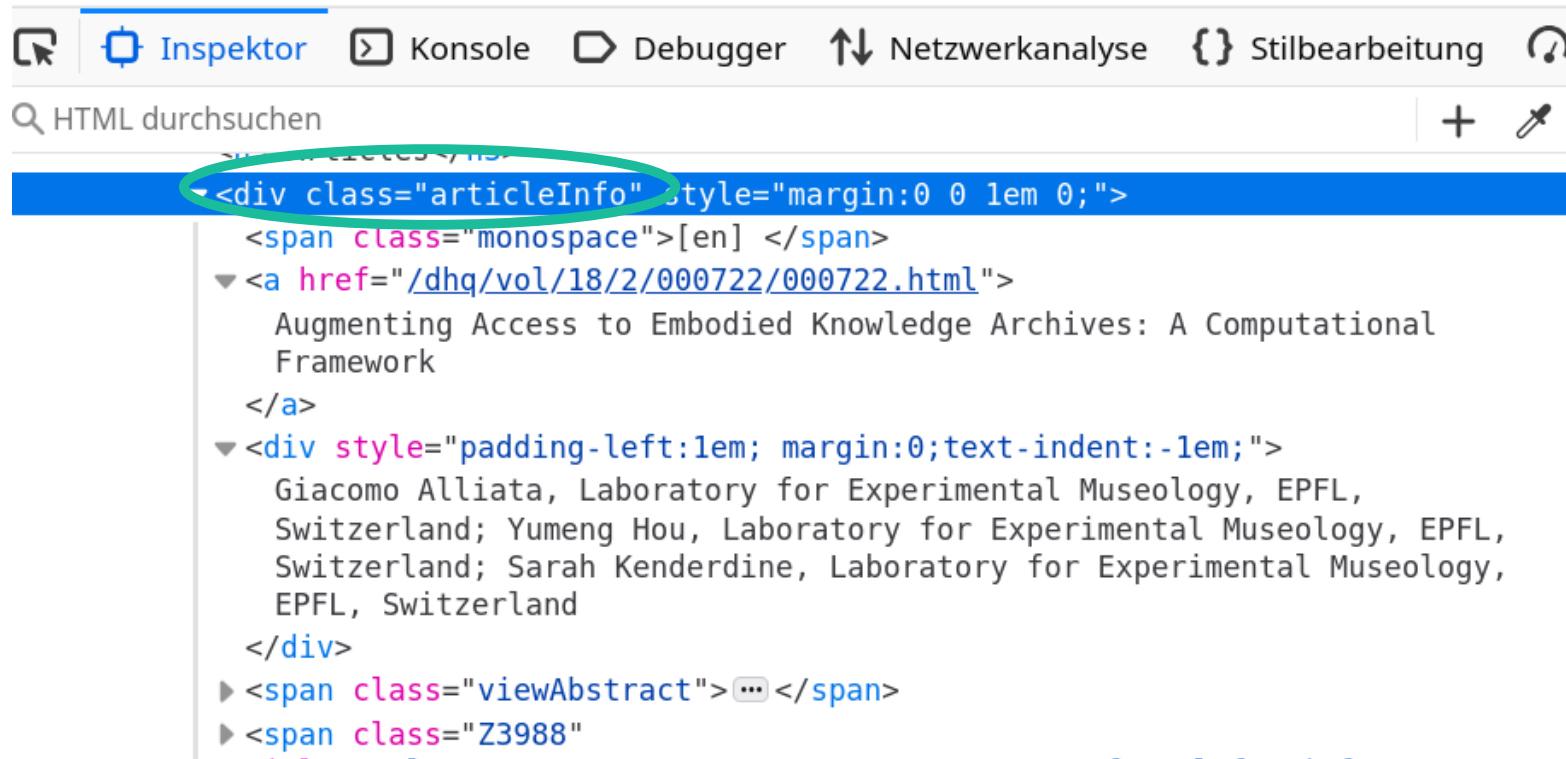
- Die Website im Browser öffnen
- Entwicklertools öffnen (Strg+Umsch+I)
- Mit dem Tool zum Auswählen eines Elements (Pfeil links) einen Artikeleintrag anklicken

11/2024

Ruhr-Universität Bochum, DHC (Olaf Berg)

17

Seite analysieren



- Mit Klick auf die kleinen Pfeile im Inspektor-Codefeld den Quellcode untersuchen
- Elementnamen und Attribute notieren, die relevante Informationen enthalten

Seite analysieren



Der Abstract Text ist auf der Webseite zunächst versteckt „display:none“, im Quellcode aber bereits enthalten

Webscraping in Python

Relevante Python-Pakete:

- Requests
Webabfragen
- BeautifulSoup
HTML Verarbeitung
- CSV
Speichern als CSV
- Selenium
Simuliert bei Bedarf Browser inkl. Interaktionen

Viel Spaß beim Ausprobieren!

```
import requests
from bs4 import BeautifulSoup
import csv

# Die URL von der wir Inhalte auslesen wollen
url = "https://www.digitalhumanities.org/dhq/vol/18/2/index.html"

# Rufe die Webseite ab
response = requests.get(url)

# lese den Website Code in BeautifulSoup ein
soup = BeautifulSoup(response.text, "html.parser")

# finde alle Artikel
artikelliste = soup.find_all("div", class_="articleInfo")

# extrahiere die relevanten Informationen aus der Artikelliste
artikelinfo = []

for artikel in artikelliste:
    info = {
        'Titel': " ".join(artikel.find_all('a')[0].text.replace("\n", "").replace('\t', ' ').split()),
        'Autor_in': " ".join(artikel.find_all('div')[0].text.replace("\n", "").replace('\t', ' ').split()),
        'Abstract': " ".join(artikel.find_all('span', class_='abstract')[0].text.replace('\n', '').replace('\t', ' ').split())
    }
    artikelinfo.append(info)

# Ausgabe der Artikelinfos auf Konsole
print(artikelinfo)

# Speichern der Liste mit Titeln, Autor*innen und Abstracts
spaltennamen = ['Titel', 'Autor_in', 'Abstract']
with open('DHQ_Artikelliste.csv', 'w', newline='') as csvfile:
    writer = csv.DictWriter(csvfile, fieldnames=spaltennamen)
    writer.writeheader()
    writer.writerows(artikelinfo)
```

Beispiel Jupyter Notebook: https://gitlab.ruhr-uni-bochum.de/dh-einfuehrungen/html_css_js

Weiterführende Aufgabe:
Versuche die Titel, Autor*innen und
Abstracts von dieser Website
auszulesen:
<https://zfdg.de/heft/9.2024>

Happy coding!

Ressourcen zum Weiterlesen

- **Einführungen und Tutorials für Python & mehr:**
 - <https://digital-history-berlin.github.io/Python-fuer-Historiker-innen/home.html>
 - <http://programminghistorian.org/>
 - <https://realpython.com/>
 - <https://www.w3schools.com/>
 - <https://www.geeksforgeeks.org/>

Unterstützungsangebote an der RUB

- Digital Humanities Center der UB
<https://dh.rub.de/>
- Methodenzentrum
<https://methodenzentrum.ruhr-uni-bochum.de/>
- DH Ruhr Projekt
<https://dh-ruhr.ruhr-uni-bochum.de/>

Vernetzungsangebote an der RUB

- **Netzwerk Digital Humanities**
<https://dh-netzwerk.blogs.ruhr-uni-bochum.de/>
- **Mailingliste DH an der RUB**
<https://lists.ruhr-uni-bochum.de/mailman/listinfo/digitalhumanities>
- **Element Chat**
<https://element.rub.de/#/room/#dh:ruhr-uni-bochum.de>