

Netzwerke und Internet II - B

Dr. Philipp Hurni, Kantonsschule Sursee

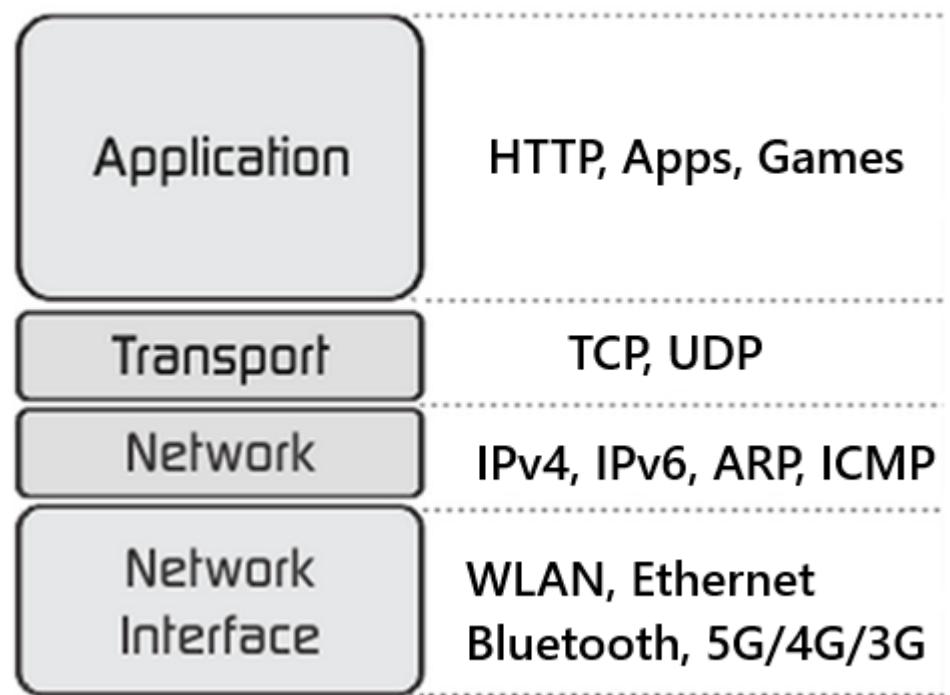
Netzwerke und Internet II

- Repetition Webseiten
 - Schichtenmodell
 - HTML Seiten erstellen
 - Hochladen auf einen Webserver
- Erweiterung
 - Cascading Style Sheets
 - JavaScript
- Web-Applikations-Frameworks
 - Übersicht
 - Einführung in Flask
 - Übungen I und II



Das Netzwerk-Schichtenmodell nach TCP/IP

TCP/IP Schichtenmodell



Anwendung des Users (z.B. Web-Browser, Spotify-App, Game, etc.)

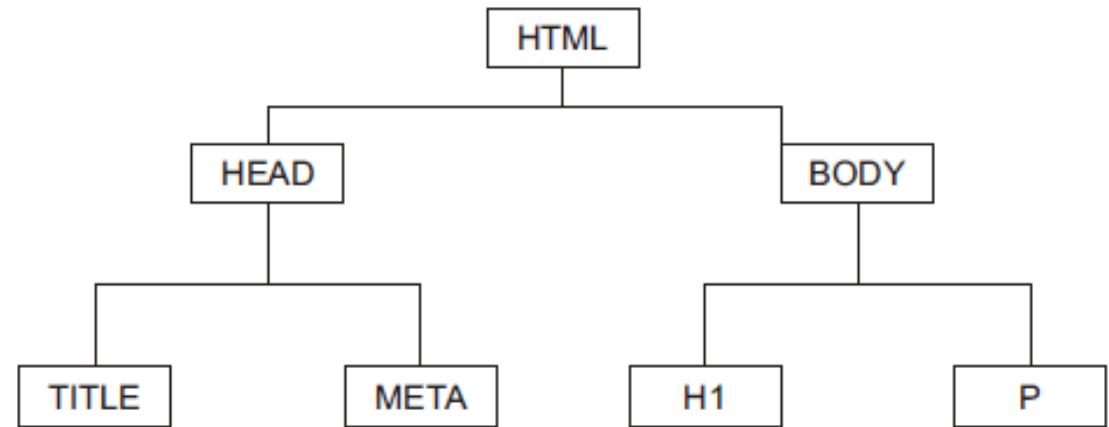
Ermöglicht (zuverlässige) **Punkt-zu-Punkt Kommunikation** zwischen zwei Computern, gegeben dass der Pfad gefunden ist.

Findet den **Pfad zum Ziel** im Netz

Steuert das **Sende- und Empfangs-Gerät**
Sorgt für die Kommunikation über das Medium. Konvertiert 1en und 0en zu Signalen.

HTML Head und Body

```
<html>
  <head>
    <title>Titel der Seite</title>
  </head>
  <body>
    <h1>Dies ist eine Webseite</h1>
    <p>Wirklich! Es ist eine!</p>
  </body>
</html>
```



Unsere erste Webseite

- Wir benutzen den Online-Webseiten Editor von OneCompiler.com

Link: <https://onecompiler.com/html>

- Wir speichern diese Struktur unter dem index.html tab

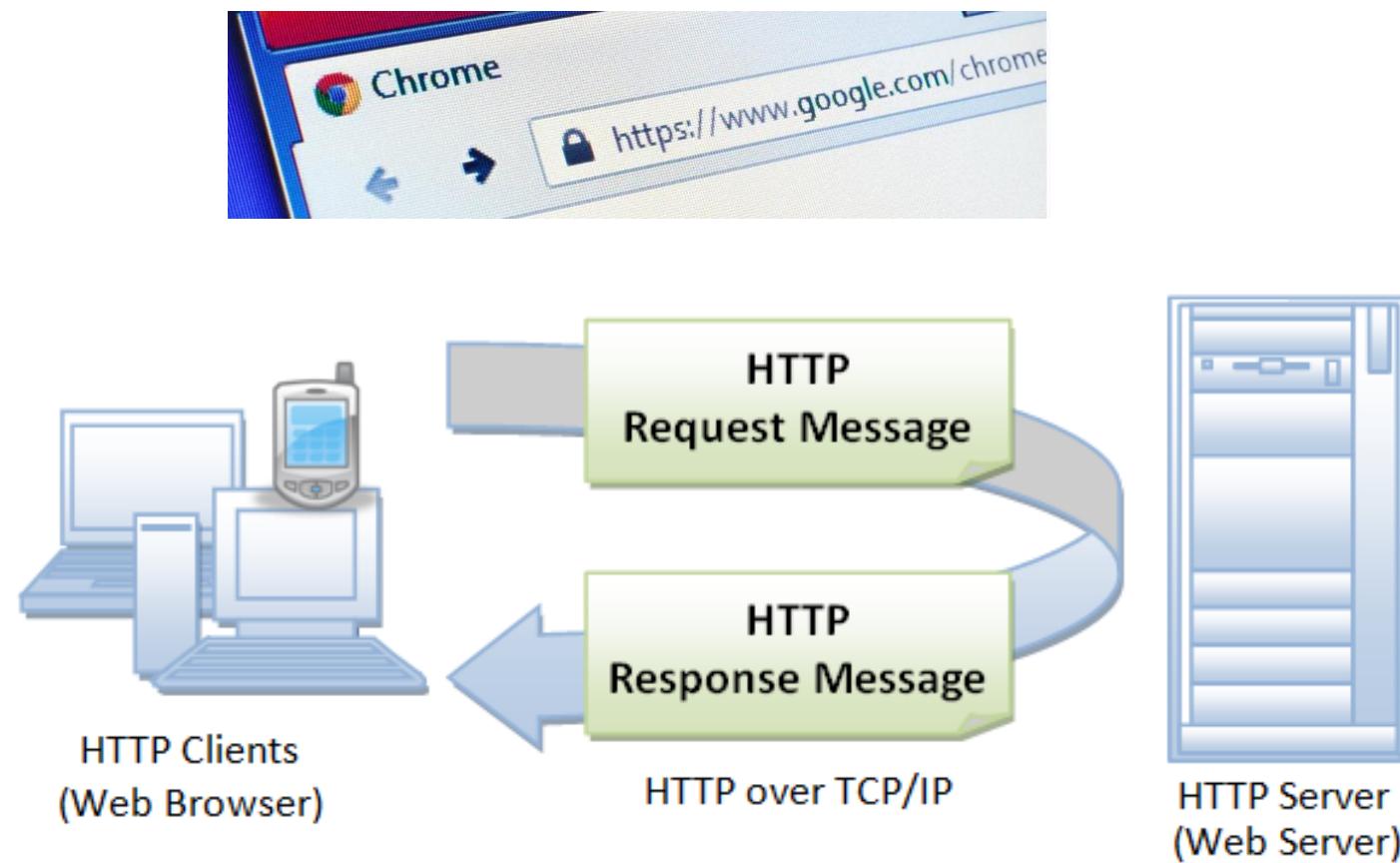


The screenshot shows the OneCompiler web-based IDE interface. At the top, there's a navigation bar with tabs for 'index.html', 'styles.css', and 'script.js'. Below the tabs are buttons for 'NEW', 'HTML', and 'RUN'. The main left panel displays the HTML code:

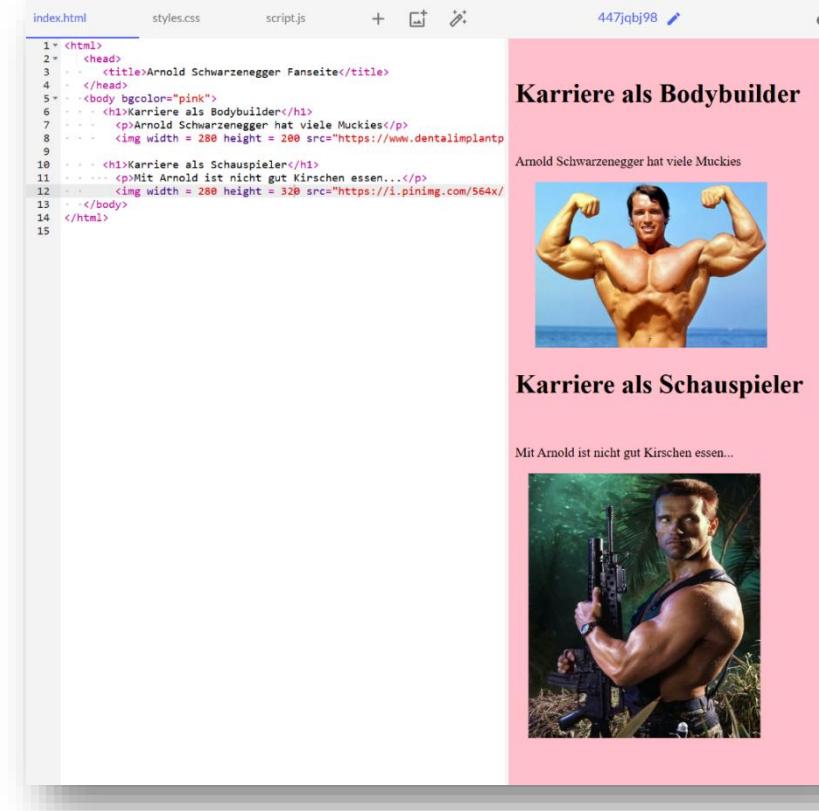
```
1 <html>
2   <head>
3     <title>Titel der Seite</title>
4   </head>
5   <body>
6     <h1>Dies ist eine Webseite!</h1>
7     <p>Wirklich! Es ist eine!</p>
8   </body>
9 </html>
```

The right panel shows the rendered output of the code: "Dies ist eine Webseite!" above the text "Wirklich! Es ist eine!".

HTTP Request / Response Verfahren



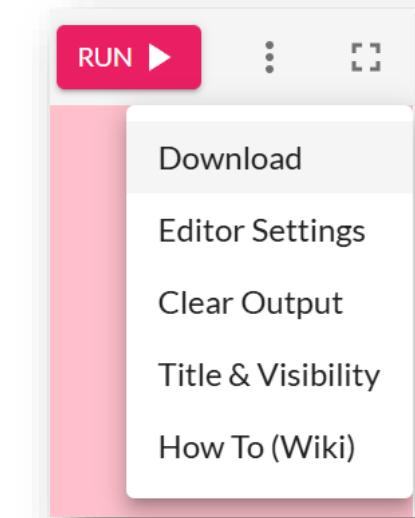
Aufgabe: gestaltet eure erste Webseite



The screenshot shows a web development interface with three tabs: index.html, styles.css, and script.js. The index.html tab contains the following code:

```
1<html>
2  <head>
3    <title>Arnold Schwarzenegger Fanseite</title>
4  </head>
5  <body bgcolor="pink">
6    <h1>Karriere als Bodybuilder</h1>
7    <p>Arnold Schwarzenegger hat viele Muckies</p>
8    
14 </html>
15
```

The rendered output on the right side of the interface shows two sections: "Karriere als Bodybuilder" with a pink background and "Karriere als Schauspieler" with a white background. Each section contains a heading, a paragraph, and an image of Arnold Schwarzenegger.



A modal menu is open, showing the following options:

- RUN ►
- Download
- Editor Settings
- Clear Output
- Title & Visibility
- How To (Wiki)

Speichert sie unter eurem Namen,
ohne Leerzeichen

Beispiel: die Schwarzenegger-Seite

```
<html>
  <head>
    <title>Arnold Schwarzenegger Fanseite</title>
  </head>
  <body bgcolor="pink">
    <h1>Karriere als Bodybuilder</h1>
    <p>Arnold Schwarzenegger hat viele Muckies</p>
    
    <h1>Karriere als Schauspieler</h1>
    <p>Mit Arnold ist nicht gut Kirschen essen...</p>
    
  </body>
</html>
```

Hoch laden ins Netz zu host: hurni.4lima.de

- Einfache Webseite zum Hochladen und betrachten im Web
- <https://hurni.4lima.de/uploader/>

Upload Your HTML File

Select HTML file to upload: hurni.html



Cascading Style Sheet (CSS)



Cascading Style Sheets (CSS)

- ist eine Sprache für elektronische Dokumente. Zusammen mit HTML und JavaScript ist es eine der Kernsprachen des World Wide Webs
- Sie ist wird vom World Wide Web Consortium (W3C) laufend weiterentwickelt.
- Mit dieser Sprache kann man das optische Erscheinungsbild einer Webseite «zentral» steuern, ohne sämtliche Seiten einzeln anpassen zu müssen

CSS



Unser erstes Cascading Style Sheet

- Wir benutzen wieder den Online-Webseiten Editor von OneCompiler.com

Link: <https://onecompiler.com/html>

The screenshot shows the OneCompiler interface with three tabs: index.html, styles.css, and script.js. The index.html tab contains the following code:

```
1 <html>
2   <head>
3     <title>Arnold Schwarzenegger Fanseite</title>
4     <link rel="stylesheet" href="styles.css" />
5   </head>
6   <body bgcolor="pink">
7     <h1>Karriere als Bodybuilder</h1>
8     <p>Arnold Schwarzenegger hat viele Muckies</p>
9     
14 </html>
```

The styles.css tab contains the following CSS rule:

```
1 body { background-color: pink; }
```

The right panel displays two sections of text:

Karriere als Bodybuilder
Arnold Schwarzenegger hat viele Muckies


Karriere als Schauspieler
Mit Arnold ist nicht gut Kirschen essen...


Unser erstes Cascading Style Sheet

- Der Link verknüpft nun das Layout der Seite mit dem styles.css

```
<link rel="stylesheet" href="styles.css"/>
```

- Dieses ist auch im OneCompiler sichtbar (siehe rotes Rechteck)

- Wir wollen nun den Style der Seite anpassen

The screenshot shows the OneCompiler interface with two tabs: 'index.html' and 'styles.css'. The 'index.html' tab contains the following code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Arnold Schwarzenegger Fanseite</title>
5     <link rel="stylesheet" href="styles.css" />
6   </head>
7   <body bgcolor="pink">
8     <h1>Karriere als Bodybuilder</h1>
9     <p>Arnold Schwarzenegger hat viele Muckies</p>
10    
15 </html>
```

The 'styles.css' tab is highlighted with a red box. To the right, the generated HTML is displayed with a pink background. It features two sections: 'Karriere als Bodybuilder' with a photo of Arnold Schwarzenegger flexing his muscles, and 'Karriere als Schauspieler' with a photo of him holding a machine gun in a scene from a movie.

Definitionen auf Basis der Tags

```
body {  
    background-color: #1a1a1a; /* Dunkelgrauer Hintergrund */  
    color: #eeeeee;           /* Nahe an Weiss Text */  
    font-family: 'Arial Black', sans-serif;  
    line-height: 1.6;  
    padding: 40px;  
    max-width: 800px;  
    margin: 0 auto;  
}  
  
/* Neuer Stil für die h1 Headings */  
h1 {  
    color: #ff4500;          /* Überschriften */  
    text-transform: uppercase;  
    border-bottom: 3px solid #ff4500;  
    padding-bottom: 10px;  
    margin-top: 40px;  
}  
  
/* Bilder */  
img {  
    border: 4px solid #555;  
    border-radius: 8px;        /* Abgerundete Ecken */  
    box-shadow: 0 10px 20px rgba(0,0,0,0.5);  
    display: block;  
    margin: 20px 0;  
    max-width: 100%;          /* Responsive Bilder */  
    height: auto;             /* Seitenverhältnis automatisch */  
}  
  
/* Hover-Effekt über dem Bild: Zoom von 20% */  
img:hover {  
    transform: scale(1.2);  
    transition: transform 0.3s ease;  
    border-color: #ff4500;  
}
```

Neues Erscheinungsbild

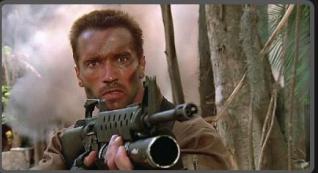
KARRIERE ALS BODYBUILDER

Arnold Schwarzenegger hat viele Muckies



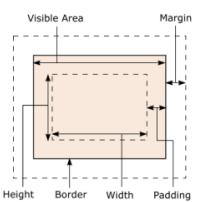
KARRIERE ALS SCHAUSPIELER

Mit Arnold ist nicht gut Kirschen essen...



Cheatography

CSS2 Cheat Sheet
by Dave Child (DaveChild) via cheatography.com/1/cs/14/

CSS2 Selectors		CSS2 Sizes		CSS2 Positioning	
*	All elements	0	0 requires no unit	display	clear
div	<div>	em	1em equal to font size of parent (same as 100%)	position	z-index
div *	All elements within <div>	ex	Height of lower case "x"	top	direction
div span	 within <div>	%	Percentage	right	unicode-bidi
div, span	<div> and 	px	Pixels	bottom	overflow
div >	 with parent <div>	cm	Centimeters	left	clip
span		mm	Millimeters	float	visibility
div +	 preceded by <div>	in	Inches		
span		pt	1pt = 1/72in		
.class	Elements of class "class"	pc	1pc = 12pt		
div.class	<div> of class "class"				
#itemid	Element with id "itemid"				
div#itemid	<div> with id "itemid"				
a[attr]	<a> with attribute "attr"				
a[attr=x]	<a> when "attr" is "x"				
a[clas-s=x]	<a> when class is a list containing 'x'				
a[lang]=e	<a> when lang begins "en" n'				
CSS2 Pseudo Selectors and Pseudo Classes		CSS2 Dimensions		CSS2 Colour and Background	
:first-child	First child element	width	min-height	color	background-repeat
:first-line	First line of element	min-width	max-height	background	background-image
:first-letter	First letter of element	max-width	vertical-align	background-color	background-position
:hover	Element with mouse over	height		background-attachment	
:active	Active element	CSS2 Text		CSS2 Fonts	
:focus	Element with focus			text-indent	font-weight
:link	Unvisited links			text-align	font-family
:visited	Visited links			text-decoration	font-style
:lang(var)	Element with language "var"			text-shadow	font-size
:before	Before element			letter-spacing	font-variant
:after	After element	CSS2 Box Model		Visible Area	
					

Siehe auch "Cheat Sheet"
Google... ChatGPT!

Vorteil von externen .css Dateien

- **Zentrale Verwaltung:** Änderungen an einer Datei aktualisieren das Design der gesamten Website (viele Unterseiten gleichzeitig).
- **Schnellere Ladezeiten:** Die Datei wird einmalig vom Browser geladen und für alle weiteren Seiten im Cache gespeichert.
- **Sauberer Code:** Trennung von Inhalt (HTML) und Design (CSS) sorgt für bessere Übersichtlichkeit.
- **Teamarbeit:** Entwickler können gleichzeitig am HTML und am CSS arbeiten, ohne sich gegenseitig zu überschreiben.

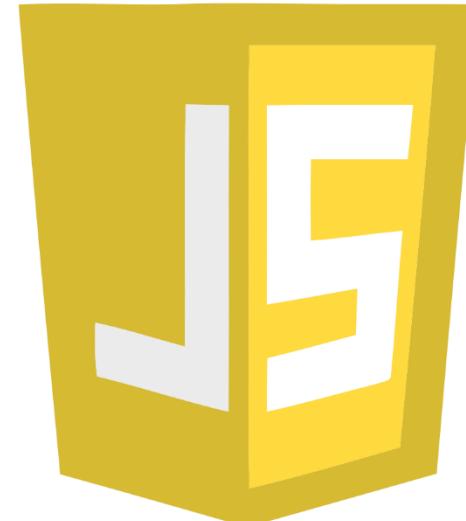
Javascript

```
25      }).then(response => {
26        setProgress('finished');
27      );
28    };
29
30    return (
31      <div className={ `progress-button ${progress}` }>
32        <span className="loading-text" loading={loading}>
33          <button className="downLoadButton" onClick={onClick}>
34            <span className="buttonText" (progress={progress})>
35              </button>
36            <span className="percentage">(progress){progress}</span>
37          </span>
38        </div>
39      );
40    );
41  }
42
43  export default App;
```

JavaScript

- JavaScript (kurz JS) ist eine Skriptsprache, die dafür entwickelt wurde, Benutzerinteraktionen auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und so die Möglichkeiten von HTML zu erweitern.
- JavaScript wird hauptsächlich für die Entwicklung interaktiver Webseiten eingesetzt. Seit der Geburt des Web sind Webseiten mit JavaScript verknüpft. Praktisch jede Webseite, die ihr lädt, hat JavaScript Elemente eingebettet
- Mittlerweile ist JS zu einer Art "universalen Programmiersprache" geworden – es werden sogar Microcontroller damit programmiert.

JavaScript



JavaScript – script.js

"Objekte"

```
const canvas = document.getElementById('snowCanvas');
const ctx = canvas.getContext('2d');

canvas.width = window.innerWidth;      "Variablen"
canvas.height = window.innerHeight;

const flockenAnzahl = 100;
const groesse = 3; // Alle Flocken sind 3 Pixel gross
const flocken = []; // Array

// Flocken erstellen
for (let i = 0; i < flockenAnzahl; i++) {
  flocken.push({
    x: Math.random() * canvas.width,
    y: Math.random() * canvas.height,
    speed: Math.random() * 2 + 1 // Unterschiedliche Geschwindigkeit
  });
}
```

Funktion



```
function zeichnen() {
  ctx.clearRect(0, 0, canvas.width, canvas.height);
  ctx.fillStyle = "white";

  for (let i = 0; i < flockenAnzahl; i++) {
    let f = flocken[i];
    // Kreis zeichnen
    ctx.beginPath();
    ctx.arc(f.x, f.y, groesse, 0, Math.PI * 2);
    ctx.fill();
    // Bewegung
    f.y += f.speed;
    // Reset am oberen Rand
    if (f.y > canvas.height) {
      f.y = -groesse;
      f.x = Math.random() * canvas.width;
    }
  }
  requestAnimationFrame(zeichnen);
}
zeichnen();
```

JavaScript

- JavaScript ist eine clientseitige Skriptsprache. Der Code wird vom Webbrowser des Clients und nicht auf dem Webserver verarbeitet. Dies bedeutet, dass JavaScript-Funktionen ausgeführt werden können, nachdem eine Webseite vom Browser geladen wurde.
- In HTML wird JavaScript-Code zwischen den Tags `<script>` und `</script>` eingefügt, oder Link auf ein externes Skript. Externe Skripte sind praktisch, wenn derselbe Code auf vielen verschiedenen Webseiten verwendet wird.
- JavaScript-Dateien haben die Dateiendung .js. Um ein externes Skript zu verwenden, geben Sie den Namen der Skriptdatei im src-Attribut (Source) eines `<script>`-Tags an – wie bei uns mit `<script src="script.js"></script>`

Nun schneit es bei Arnold aber noch nicht...

The screenshot shows a development environment with three tabs: index.html, styles.css, and script.js. The index.html tab contains the following code:

```
1<html>
2  <head>
3    <title>Arnold Schwarzenegger Fansseite</title>
4    <link rel="stylesheet" href="styles.css" />
5  </head>
6<body>
7  <h2>KARRIERE ALS BODYBUILDER</h2>
8  <p>Arnold Schwarzenegger hat viele Muckies</p>
9  
10 <h2>KARRIERE ALS SCHAUSPIELER</h2>
11 <p>Mit Arnold ist nicht gut Kirschen essen...</p>
12 
13 </body>
14 </html>
```

The preview window on the right shows two sections: "KARRIERE ALS BODYBUILDER" featuring a photo of Arnold Schwarzenegger flexing his muscles, and "KARRIERE ALS SCHAUSPIELER" featuring a photo of Arnold Schwarzenegger in a scene from the movie Predator.

A) baue das Skript ein, mit `<script src="script.js"></script>` im body Tag!
Zusätzlich noch einen "canvas" (Zeichenfläche)
`<canvas id="snowCanvas"></canvas>`

B) Baue das Canvas-design ein in styles.css:
`#snowCanvas {
 position: fixed;
 top: 0;
 left: 0;
}`

Frohes Neues Jahr Arnold!

KARRIERE ALS BODYBUILDER

Arnold Schwarzenegger hat viele Mückies



KARRIERE ALS SCHÄUSPIELER

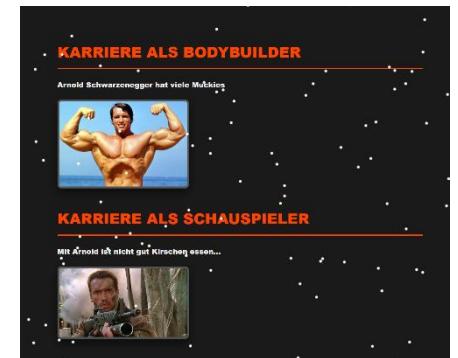
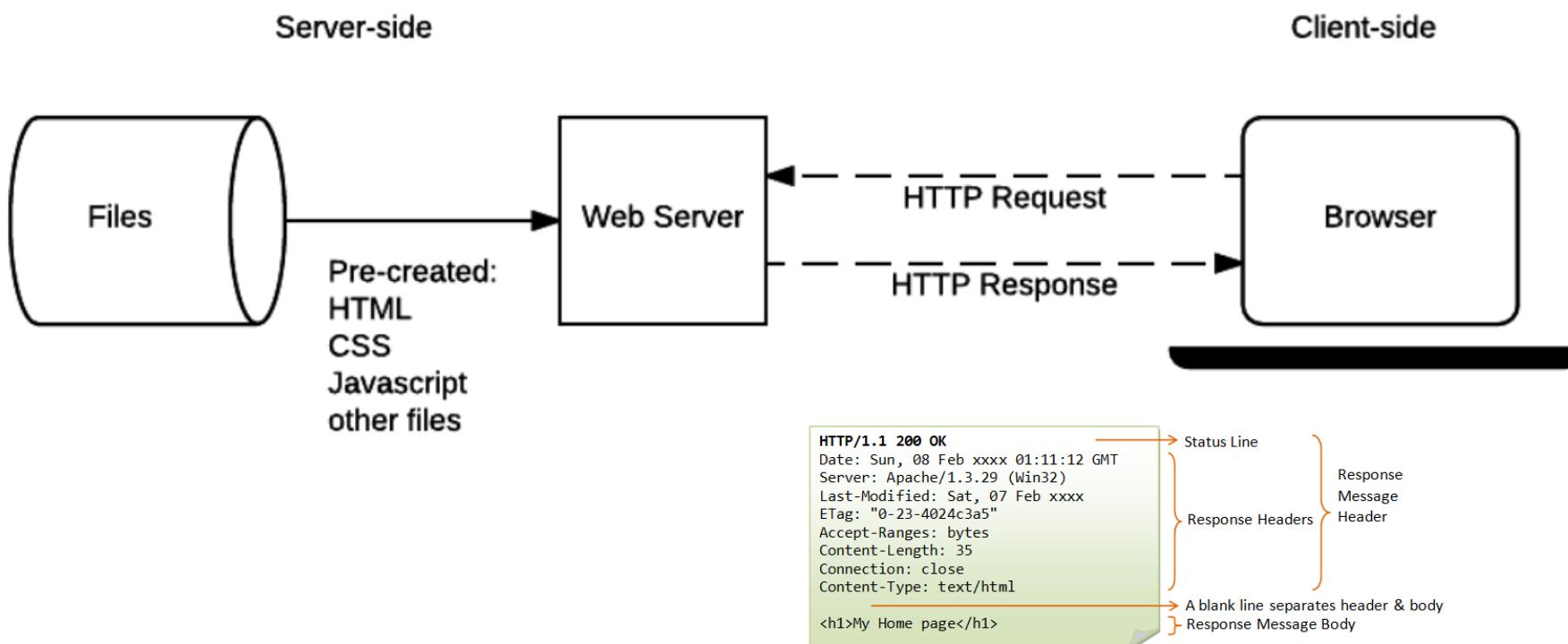
Mit Arnold ist nicht gut Kirschen essen...



Übersicht: Was läuft wo?

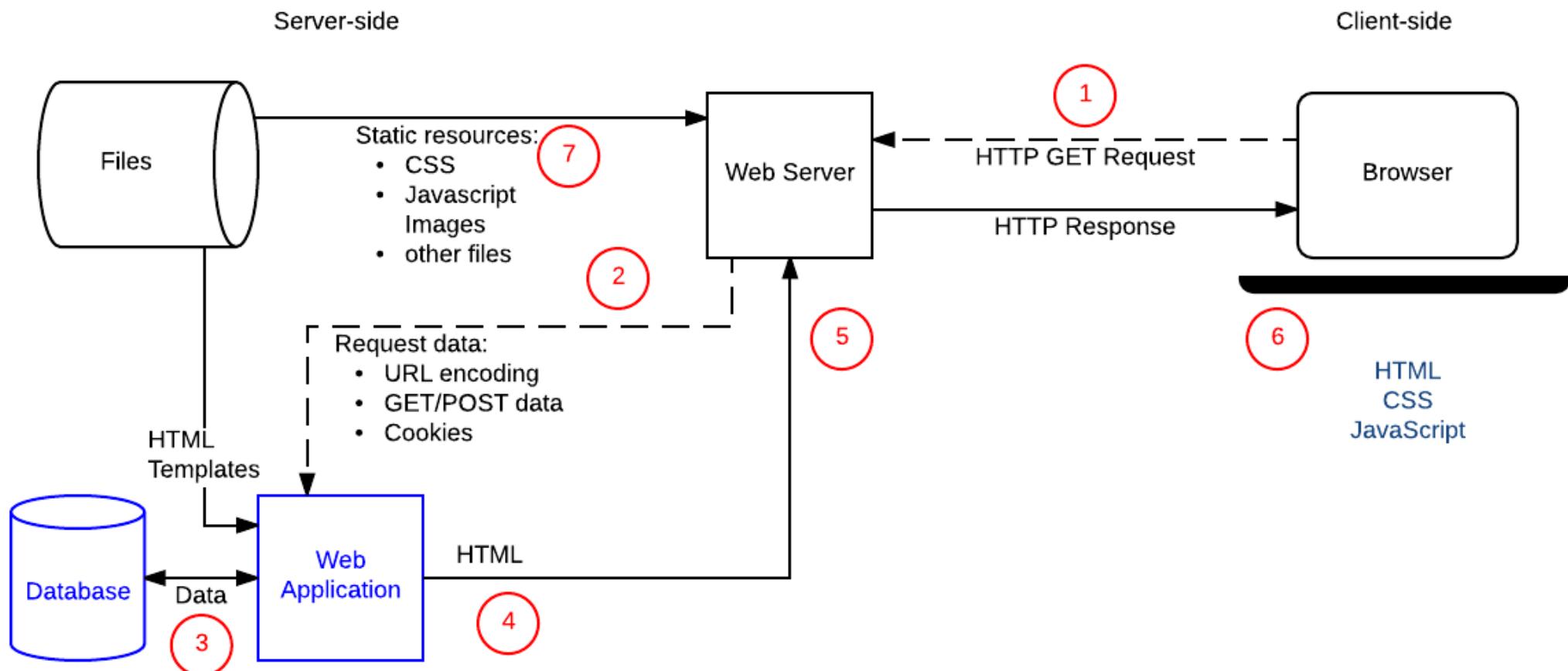


Bisher: rein statisch – Seiten waren einfach Dateien



Ausführen von HTML,
Javascript, CSS: Browser

Die meisten Seiten: dynamisch – durch Webapplikationen

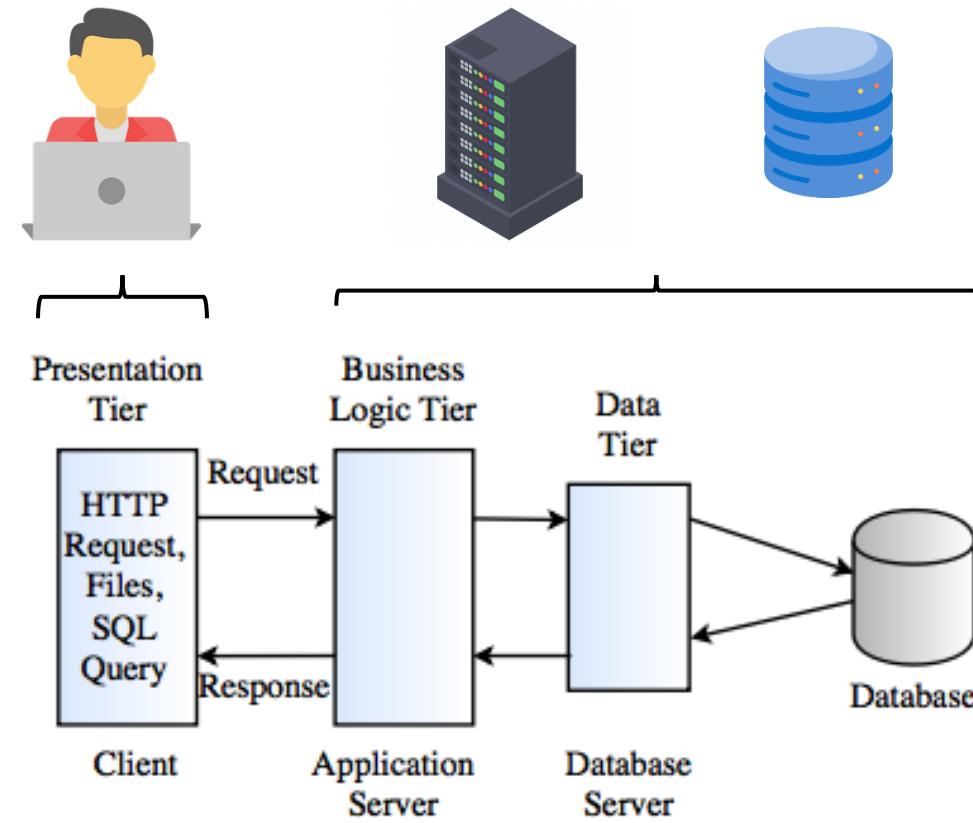


Die 3 Tier Architektur



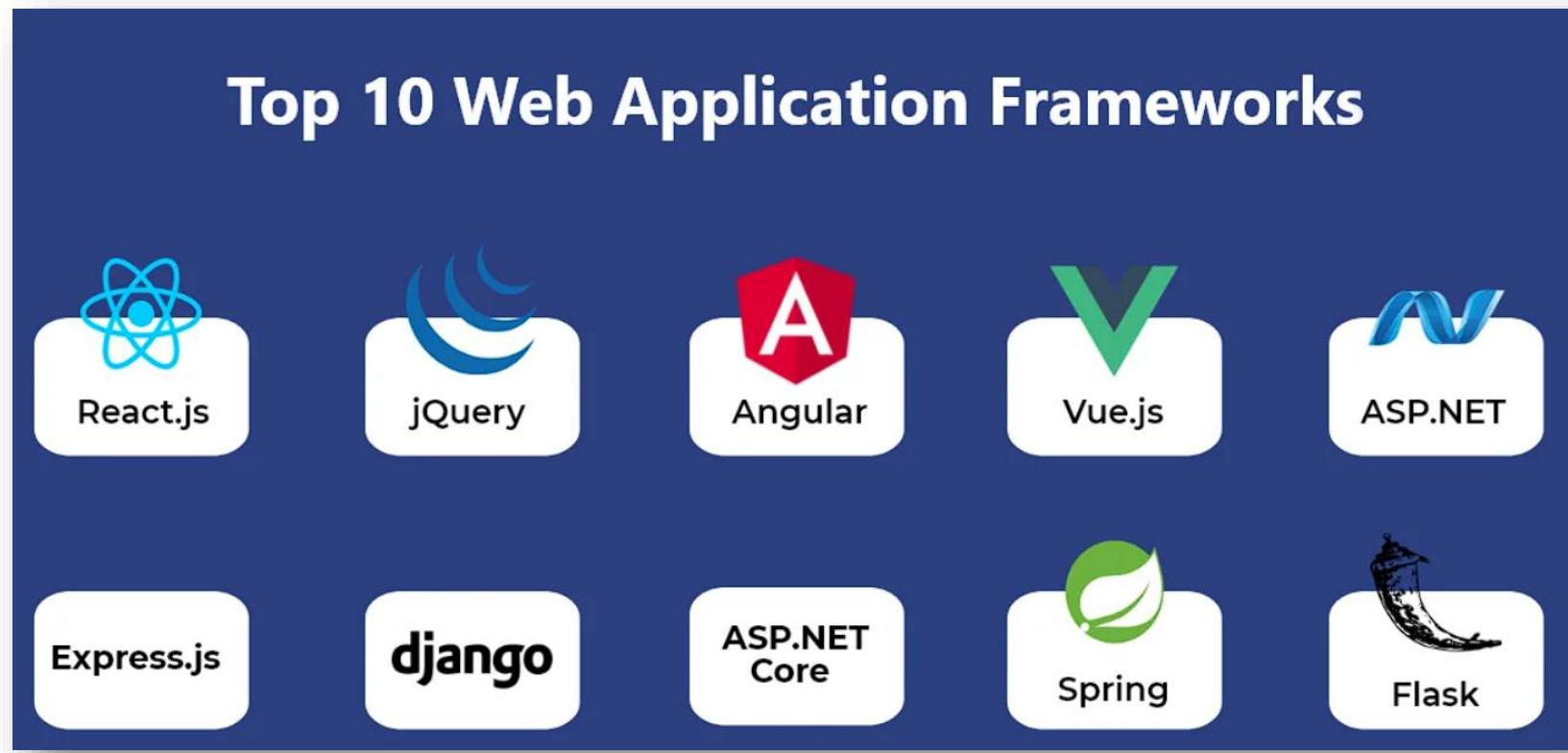
Die klassische 3-Tier-Architektur

- **Tier** steht für «Schicht»
- **Presentation Tier:** Browser des Clients (auch Client Tier genannt)
- **Business Logic Tier/Application Tier:** Webserver-Applikation auf Webserver-Computer. Bei uns: HTML-Files auf Apache Webserver
- **Data Tier:** Datenbankserver mit Zugriff auf Datenbank. Das haben wir bisher noch nicht gemacht



3 - Tier Architecture

Heutige Web Applikations Frameworks



Beispiele für Django (python)

Instagram



Beispiele für Flask (python)



Flask



- **Micro-Framework-Ansatz:** minimalistisch, verzichtet bewusst auf integrierte Datenbanken oder Validierungstools
- **Schnell & Flexibel:** Prototypen und Web-Apps lassen sich mit sehr wenig Code-Aufwand erstellen.
- **Hohe Erweiterbarkeit:** reichhaltiges Ökosystem an Extensions (z.B. für Authentifizierung, Datenbank-Anbindung (SQLAlchemy) oder REST-APIs)
- **Entwicklerfreundlich:** Eingebauter Server zum Testen, sowie einen Debugger für die Fehlersuche

Beispiel Witze-Generator: app.py

```
from flask import Flask, render_template, request
from random import *
from datetime import datetime
app = Flask(__name__)

WITZ_DATEN = [
    "Was sagt ein Genervter zu einem Taschenmesser? 'Klappe zu!'",
    "Wie nennt man ein Kaninchen im Fitnessstudio? Pumpernickel.",
    "Warum können Skelette so schlecht lügen? Weil sie so leicht zu durchschauen sind.",
    "Was ist die Lieblingsspeise von Autos? Parkplätze."
]

@app.route('/', methods=['GET', 'POST'])
def index_seite():
    ausgabe = None
    if request.method == 'POST':
        # Wir schauen, welcher Button-Wert (action) gesendet wurde
        aktion = request.form.get('action')

        if aktion == 'witz':
            zufallszahl = randint(0, len(WITZ_DATEN)-1)
            ausgabe = WITZ_DATEN[zufallszahl]

        elif aktion == 'zeit':
            # Datum und Zeit ermitteln
            jetzt = datetime.now()
            ausgabe = jetzt.strftime('Heute ist der %d.%m.%Y. Es ist %H:%M Uhr.')

    return render_template('index.html', witz=ausgabe)

if __name__ == '__main__':
    app.run(debug=True, port=8087, use_reloader=False)
```

Witze und Zeit Service

Wie nennt man ein Kaninchen im Fitnessstudio?
Pumpernickel.

Gib mir einen Witz!

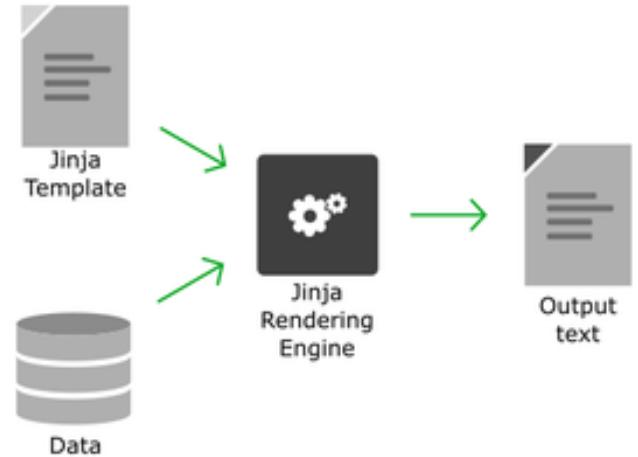
Wie spät ist es?

Beispiel Witze-Generator: index.html

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Witze und Zeit Service</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="card">
        <h1>Witze und Zeit Service</h1>
        <div class="witz-display">
            {% if witz %}
                {{ witz }}
            {% else %}
                Bereit für einen Lacher oder die Uhrzeit?
            {% endif %}
        </div>
    </div>

    <form method="POST">
        <button type="submit" name="action" value="witz">Gib mir einen Witz!</button>
        <br/>
        <button type="submit" name="action" value="zeit" style="background-color: #4CAF50;">Wie spät ist es?</button>
    </form>
</div>
</body>
</html>
```

} **Jinja ([link](#))**



Wie nennt man ein Kaninchen im Fitnessstudio?
Pumpernickel.

Gib mir einen Witz!

Wie spät ist es?

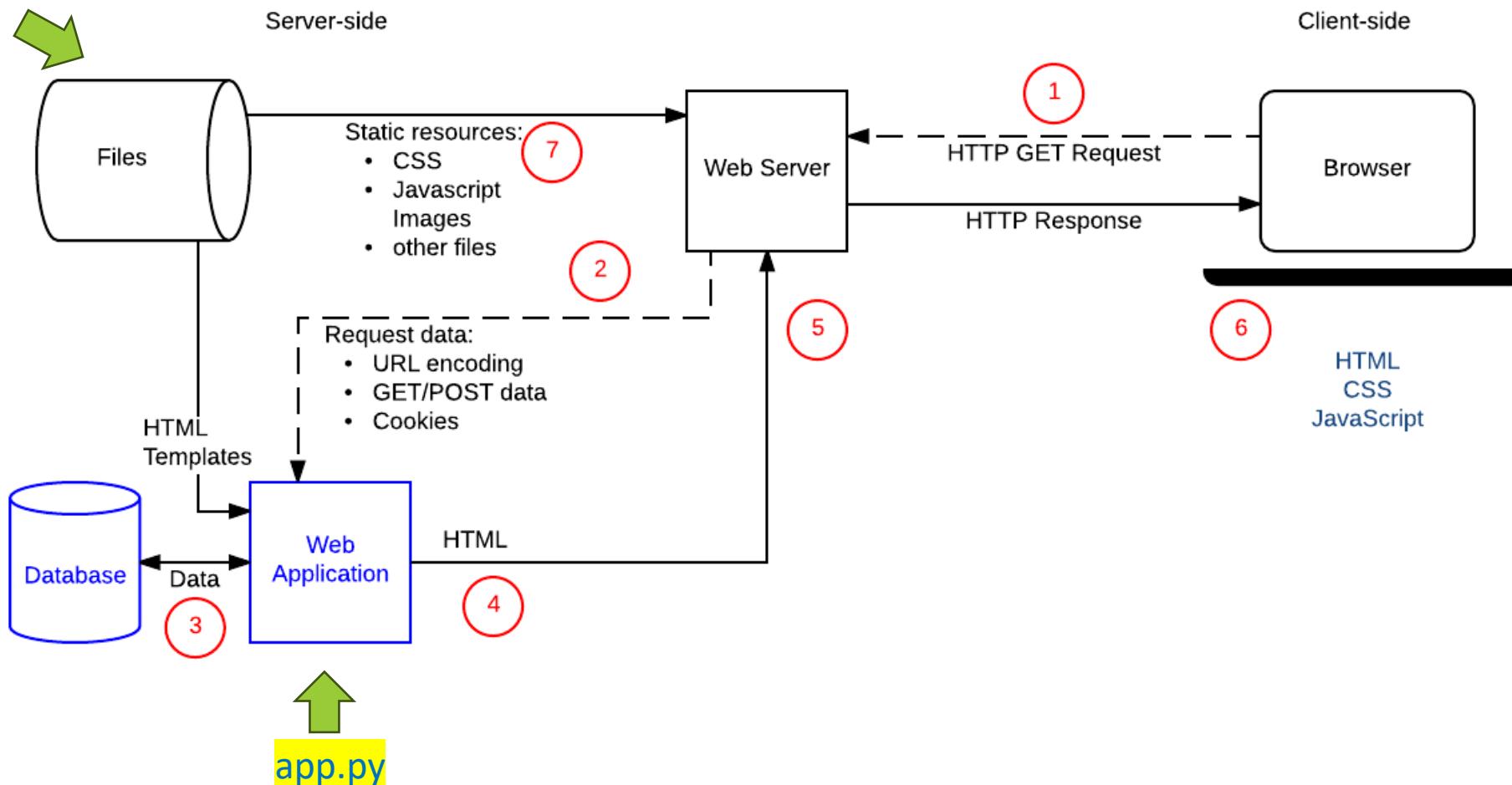
Ablauf – Schritt für Schritt

- **Der Klick (HTML):** Der Prozess beginnt in der index.html, wenn der Benutzer in seinem Browser den `<button type="submit">` innerhalb des `<form method="POST">` drückt. Und zwar denjenigen mit `value="witz"` das heisst der Blaue. Der Browser sendet dann eine HTTP-POST-Anfrage an den Web-Server.
- **Der Web-Server (app.py) führt index_seite aus** (denn das ist die Funktion die zu index.html gehört).
- **Die Funktion index_seite()** stellt fest, dass der Request vom Typ POST ist (`if request.method == 'POST'`). Dann prüft die funktion den Wert des action Attributs. Hier ist es gesetzt auf "witz". Die Funktion definiert dann `zufallszahl = randint(0, len(WITZ_DATEN)-1)` und holt mit der definierten Zufallszahl einen Witz aus der Liste aus und speichert ihn in der Variable auswahl.
- **Die Übergabe:** Der Befehl `render_template('index.html', witz=auswahl)` schickt die HTML-Seite zurück an den Benutzer-Browser und übergibt den gewählten Witz an den definierten Platzhalter `witz`. Dies ist dann auch der Name der Variable in der HTML-Datei index.html mit dem Jinja Skript.
- **Die Anzeige (HTML):** In der index.html prüft Jinja, ob der Wert `witz` definiert ist `{% if witz %}`. Dies ist der Fall, woraufhin der Inhalt der Variable durch `{{ witz }}` innerhalb des witz-display-Bereichs eingebettet wird und dann beim Benutzer auf dem Bildschirm erscheint.

Beispiel Witze-Generator

static
templates
app.py

templates/index.html
static/styles.css



Aufwärm-Übung I: Lehrerwitze

- Füge einen weiteren Knopf ein "Lehrerwitze"
- Falls der gedrückt wird, soll ein Lehrerwitz angezeigt werden
- Hier ist eine Auswahl
 - **Lehrer** : "Nenne mir die drei Steigerungsstufen von *leer*." **Max**: "Leer, leerer, Lehrzimmer."
 - **Schüler**: "Warum ist meine Informatik-Note so schlecht?" – **Lehrer**: "Du hast im Binärtest eine 2 geschrieben."
 - **Lehrer**: "Wieso kommst du 20 Minuten zu spät?" Schüler: "Sie haben doch gesagt, es ist nie zu spät zum Lernen!"
 - **Lehrer**: "Wo wurden die Friedensverträge des Dreissigjährigen Krieges unterschrieben?" Schüler: "Ganz unten rechts auf der letzten Seite!"

Lösung Aufwärm-Übung I: Lehrerwitze

```
@app.route('/', methods=['GET', 'POST'])
def index_seite():
    ausgabe = None

    if request.method == 'POST':
        # Wir schauen, welcher Button-Wert (action) gesendet wurde
        aktion = request.form.get('action')

        if aktion == 'witz':
            zufallszahl = randint(0, len(WITZ_DATEN)-1)
            ausgabe = WITZ_DATEN[zufallszahl]

        if aktion == 'lehrer':
            zufallszahl = randint(0, len(LEHRERWITZE)-1)
            ausgabe = LEHRERWITZE[zufallszahl]

        elif aktion == 'zeit':
            # Datum und Zeit ermitteln
            jetzt = datetime.now()
            ausgabe = jetzt.strftime('Heute ist der %d.%m.%Y. Es ist %H:%M Uhr.')

    return render_template('index.html', witz=ausgabe)

if __name__ == '__main__':
    app.run(debug=True, port=8087, use_reloader=False)
```

Beispiel II: der Robo-Therapeut

```
app.py > ...
1  from flask import Flask, render_template, request
2  import random
3
4  app = Flask(__name__)
5
6  # Eine einfache Liste für alles
7  CHAT_VERLAUF = ["Hallo! Wie kann ich helfen?"]
8
9  # die möglichen Antworten des Bots
10 BOT_STATEMENTS = [
11     "Interessant. Erzählen Sie mir mehr darüber.",
12     "Du meine Güte... Wie fühlen Sie sich dabei?",
13     "Interessant. Was löst das in Ihnen aus?",
14     "Das ist bestimmt nicht einfach... Und weiter?",
15     "Interessant. Haben Sie das schon öfter erlebt?",
16     "Verstehe... Was bedeutet das für Sie?",
17     "Interessant. Wie gehen Sie damit um?",
18     "Verstehe... führen Sie das bitte weiter aus."
19 ]
20
21 @app.route('/', methods=['GET', 'POST'])
22 def chat_seite():
23     if request.method == 'POST':
24         # die Nachricht aus dem Form holen
25         user_text = request.form.get('nachricht')
26         if user_text.strip() != "":
27             # die Nachricht geht hier rein
28             CHAT_VERLAUF.append(user_text)
29
30             # der Bot antwortet mit zufälliger Nachricht
31             zufall_index = random.randint(0, len(BOT_STATEMENTS) - 1)
32             antwort = BOT_STATEMENTS[zufall_index]
33             CHAT_VERLAUF.append(antwort)
34
35     return render_template('index.html', verlauf=CHAT_VERLAUF)
36
37 if __name__ == '__main__':
38     # port 8081 wie bei dir, debug=True hilft Fehler zu finden
39     app.run(debug=True, port=8086, use_reloader=False)
40
```



Ergaenzungsfach-Therapeut 2026

Hallo! Wie kann ich helfen?

Aufgabe II: Ergänze den Therapeuten

- 1) Wenn der Benutzer das Wort "Freude" verwendet, soll der Bot ein Smiley zeichnen 😊
- 2) Füge einen Button hinzu, der die Liste CHAT_VERLAUF komplett leert.

Kopiere im index.html das Formular, wo der Text gesendet wird, und mach ein neues für den Knopf "Löschen". Gib dem Button im HTML den value="loeschen". In app.py prüfst du dann:

```
action = request.form.get('action')
if action == 'senden':
    ...
elif action == 'loeschen':
    ...
```

Aufgabe II: Robo Therapeut

```
21 @app.route('/', methods=['GET', 'POST'])
22 def chat_seite():
23     if request.method == 'POST':
24
25         # Aktion aus dem Formular holen
26         action = request.form.get('action')
27         if action == 'senden':
28             # die Nachricht aus dem form holen
29             user_text = request.form.get('nachricht')
30             if user_text.strip() != "":
31                 # die Nachricht geht hier rein
32                 CHAT_VERLAUF.append(user_text)
33                 if "Freude" in user_text:
34                     antwort = "Das klingt wunderbar! 😊";
35                     CHAT_VERLAUF.append(antwort)
36                 else:
37                     # der Bot antwortet mit zufälliger Nachricht
38                     zufall_index = random.randint(0, len(BOT_STATEMENTS) - 1)
39                     antwort = BOT_STATEMENTS[zufall_index]
40                     CHAT_VERLAUF.append(antwort)
41
42             elif action == 'loeschen':
43                 # Hier lösche ich den Verlauf
44                 CHAT_VERLAUF.clear()
45
46
47
48     return render_template('index.html', verlauf=CHAT_VERLAUF)
49
50 if __name__ == '__main__':
51     # port 8081 wie bei dir, debug=True hilft Fehler zu finden
52     app.run(debug=True, port=8086,use_reloader=False)
```

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Mein Chat</title>
5      <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
6  </head>
7  <body>
8
9      <div class="chat-box">
10         <h2>Ergaenzungsfach-Therapeut 2026</h2>
11         <ul>
12             {% for zeile in verlauf %}
13                 <li>{{ zeile }}</li>
14             {% endfor %}
15         </ul>
16
17         <form method="POST">
18             <input type="text" name="nachricht" placeholder="Schreibe etwas..." autofocus>
19             <button type="submit" name="action" value="senden">Senden</button>
20         </form>
21
22         <br>
23
24         <form method="POST">
25             <button type="submit" name="action" value="loeschen">Loeschen</button>
26         </form>
27     </div>
28
29
30
31 </body>
32 </html>
```