

Internet, WWW und HTML

Wie eine Seite in den Browser kommt, und woraus sie besteht

Informatik – 3. Semester

Sie tippen `www.th-luebeck.de` in die Adresszeile und drücken Enter.
Was passiert alles, bis die fertige Seite zu sehen ist?

3 Min

Zurufe

1 Von der URL zur Seite

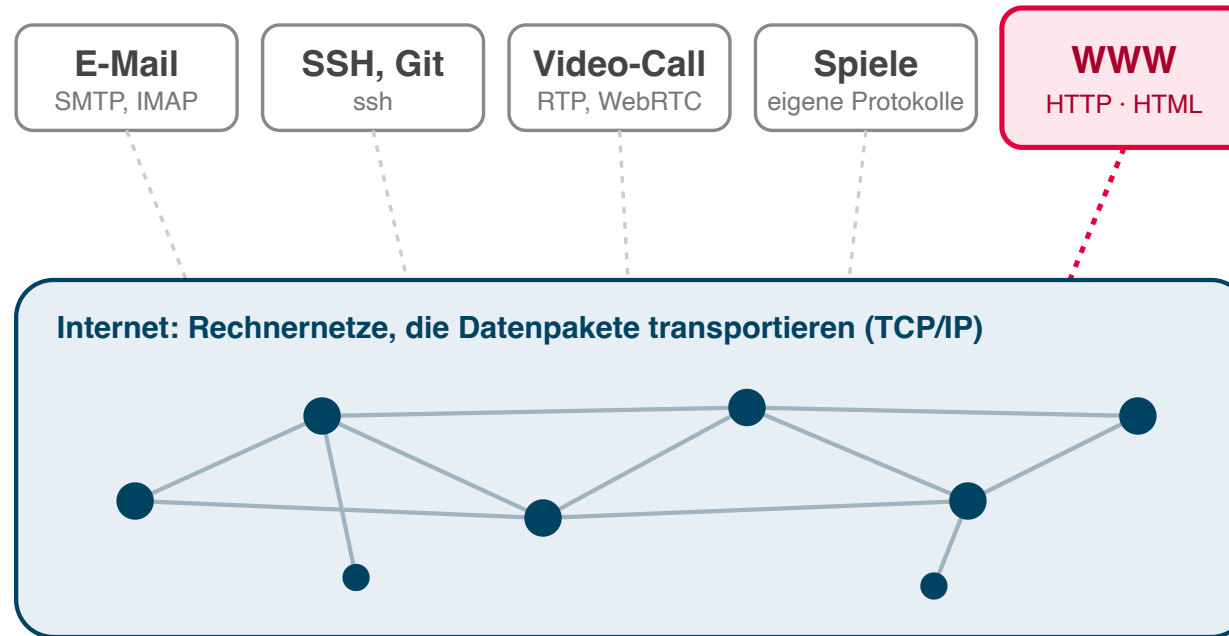
2 HTTP

3 Wo läuft was im Projekt?

4 HTML

5 Selbst bauen & Ausblick

Wie kommt eine Seite vom Server in den Browser?



- **Das Internet** transportiert Datenpakete zwischen Rechnern, egal wofür
- **Das WWW** ist einer von vielen Diensten darauf: Dokumente mit Links, übertragen per HTTP

NUTZT DAS NUR DAS INTERNET, ODER AUCH DAS WORLD WIDE WEB?

Sie lesen einen Wikipedia-Artikel im Browser.

nur Internet

WWW

Richtig. Eine Seite per HTTP holen und Links folgen: das ist das WWW.

1 / 7

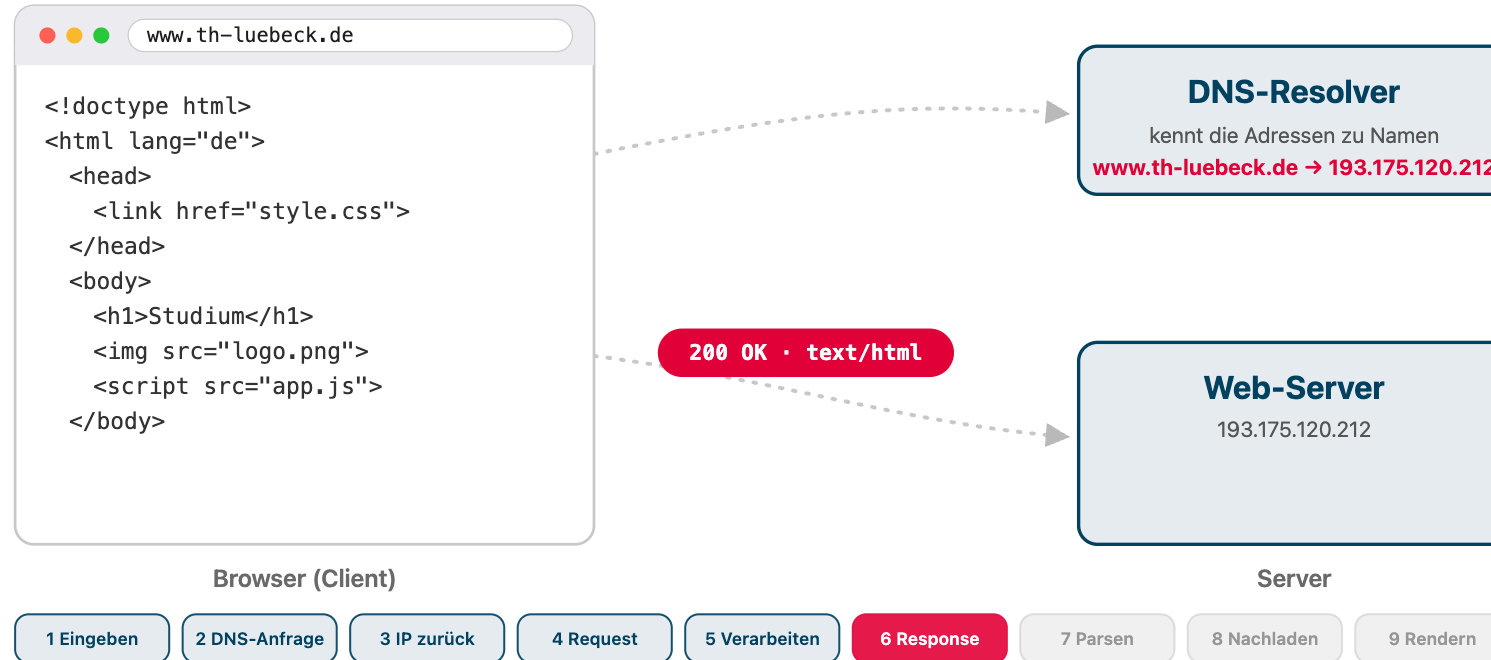
Weiter

2 Min



swe1.mylab.th-luebeck.de/web-02

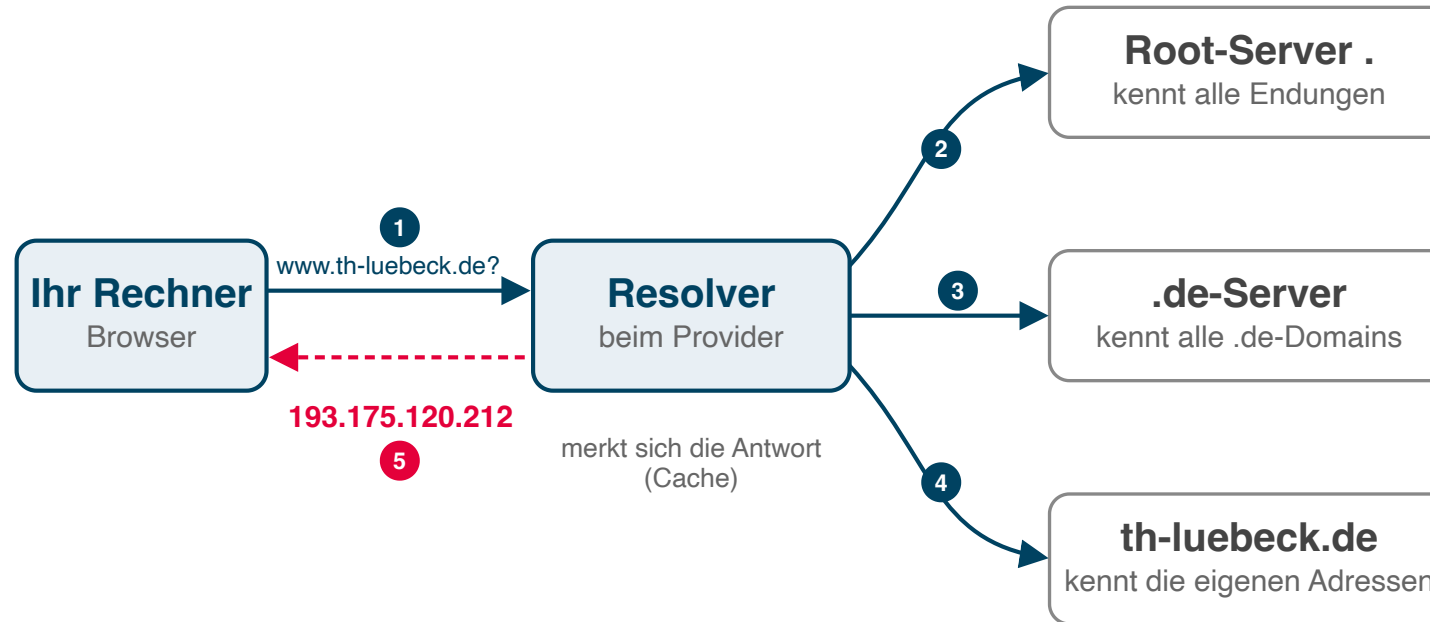
Auf dem Handy mitmachen



Die **Response** kommt zurück: Statuscode 200 und im Body ein HTML-Dokument, also Text.

Desoxyribonukleinsäure?





- Name rein, IP-Adresse raus. Kein zentrales Buch, sondern eine Kette von Zuständigen
- Die Antwort wird zwischengespeichert, der zweite Aufruf geht schneller

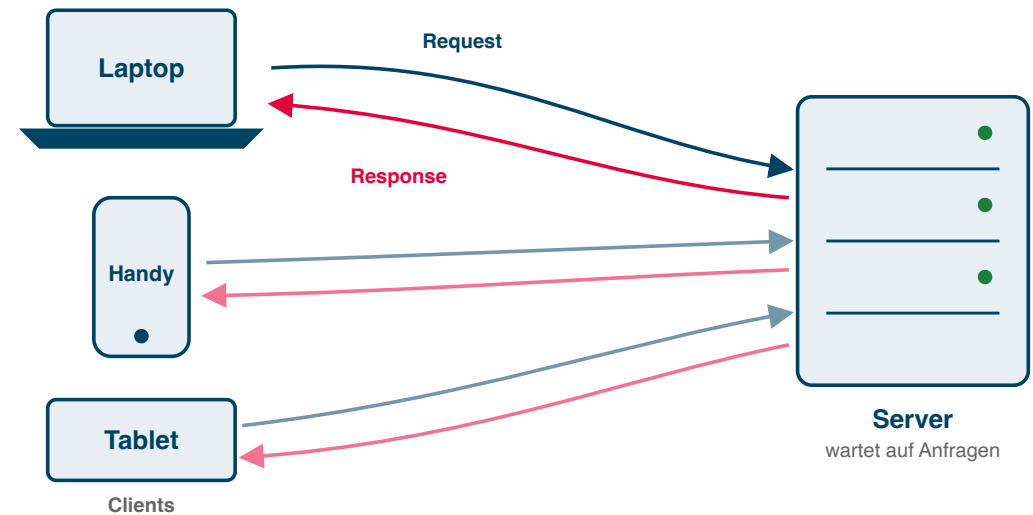
```
$ nslookup www.th-luebeck.de
Server:      192.168.0.1
Address:     192.168.0.1#53
Non-authoritative answer:

Name:        www.th-luebeck.de

Address:     193.175.120.212
```

- Geht auf Windows, macOS und Linux
- „Non-authoritative“: Die Antwort kommt aus dem Zwischenspeicher des Resolvers

- Der **Server** läuft dauernd und wartet passiv auf Anfragen
- Der **Client** fragt an: Request
- Der Server antwortet: Response
- Ohne Request keine Response. Der Server meldet sich nie von selbst



- 1 Von der URL zur Seite
- 3 Wo läuft was im Projekt?
- 5 Selbst bauen & Ausblick

2 HTTP

4 HTML

Was über die Leitung geht, ist lesbarer Text

Request

```
GET /studium HTTP/1.1
Host: www.th-luebeck.de
Accept: text/html
Accept-Language: de
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 48213

<!doctype html>
<html lang="de">
...
```

Erste Zeile: Methode und Pfad, bzw. Statuscode

Header: Metadaten, eine Zeile pro Angabe

Leerzeile, dann Body: der eigentliche Inhalt, hier das HTML

Statuscodes: Die erste Ziffer sagt, wer schuld ist

Klasse	Bedeutung	Beispiele
2xx	Hat geklappt	200 OK , 201 Created
3xx	Woanders nachsehen	301 Moved Permanently , 304 Not Modified
4xx	Der Client hat etwas falsch gemacht	400 Bad Request , 404 Not Found
5xx	Der Server hat ein Problem	500 Internal Server Error

Beim Debuggen zuerst auf den Statuscode schauen: 4xx heißt meist Fehler im Front-End, 5xx im Back-End.

WELCHE KLASSE VON STATUSCODE KOMMT ZURÜCK?

Sie vertippen sich in der URL.

2xx

3xx

4xx

5xx

Richtig. 404 Not Found. Der Client hat etwas angefragt, das es nicht gibt.

1 / 8

Weiter

2 Min



swe1.mylab.th-luebeck.de/web-02

Auf dem Handy mitmachen

GET: etwas holen

- Seiten, Bilder, CSS, Daten
- Parameter stehen in der URL: `/suche?q=html`
- Darf beliebig oft wiederholt werden
- Lesezeichen und Neuladen sind harmlos

POST: etwas schicken

- Neuer Eintrag, Anmeldung, Bestellung
- Daten stehen im Body
- Zweimal senden heißt zweimal anlegen
- Darum fragt der Browser beim Neuladen nach

WELCHE METHODE PASST?

Mitfahrten nach Datum suchen.

GET

POST

Richtig. Nur lesen. Die Suche darf in der URL stehen und als Lesezeichen funktionieren.

1 / 6

Weiter



swe1.mylab.th-luebeck.de/web-02

Auf dem Handy mitmachen

1 Min

REQUEST SCHICKEN

GET labor/hallo Daten holen

POST labor/eintraege Etwas anlegen

GET labor/umgezogen Umgezogen

GET labor/gibt-es-nicht Vertippt

GET labor/kaputt Server kaputt

201 Created

```
HTTP/1.1 201 Created
content-type: application/json
content-length: 33
date: Mon, 05 Oct 2026 14:36:05 GMT
```

```
{"id": 42, "status": "angelegt"}
```

201 Created: angelegt. Schauen Sie im Network-Tab unter Payload nach, was mitging. (2 ms)



[.../labor.html](#)

Mit DevTools öffnen

- Öffnen: `F12` bzw. `Cmd + Opt + I` → Tab **Network**
- **Gästebuch** aus Ihrer Projektvorlage: `swe1.mylab.th-luebeck.de/web-02/gaestebuch`
 - erst `GET /`, dann `style.css` und `app.js`, dann `api/entries` als JSON
 - neuer Eintrag: ein `POST api/entries`
- **th-luebeck.de** im Vergleich: Wie viele Requests sind es dort?

Die DevTools brauchen Sie ab jetzt jede Woche: für CSS, für JavaScript und spätestens bei REST.

8 Min

live

1 Von der URL zur Seite

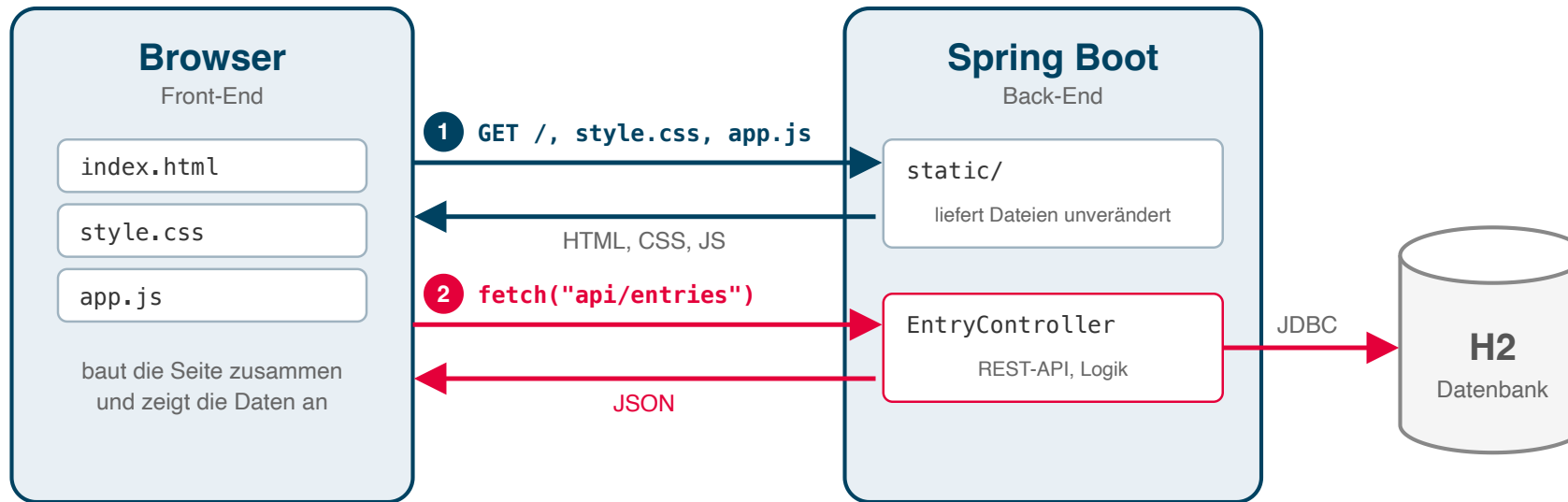
2 HTTP

3 Wo läuft was im Projekt?

4 HTML

5 Selbst bauen & Ausblick

Das Gästebuch im Kleinen ist Ihr Projekt im Großen



- 1 Spring Boot liefert HTML, CSS und JS einfach aus, unverändert
- 2 Danach holt sich der Browser die Daten per JavaScript, als JSON

Wie man den Server in Schichten aufteilt und was MVC ist: bei Leif in KW 48 bis 50.

- 1 Von der URL zur Seite
- 3 Wo läuft was im Projekt?
- 5 Selbst bauen & Ausblick

2 HTTP

4 **HTML**

Hypertext Markup Language

- Der Server könnte ja auch einfach ein Bild der fertigen Seite schicken
- Text ist aber
 - **kleiner:** ein paar Kilobyte statt Megabyte
 - **geräteunabhängig:** Handy, Laptop und Screenreader machen jeweils das Passende daraus
 - **maschinenlesbar:** Suchmaschinen, Übersetzer, Kopieren und Einfügen
 - **veränderbar:** JavaScript kann die Seite im Browser umbauen

- HTML sagt, *was* etwas ist: Überschrift, Absatz, Liste, Formular
- Wie es aussieht, bestimmt CSS, nächste Woche
- Der Browser liest den Text (parsen) und zeichnet daraus die Seite (rendern)
- HTML ist ein **Living Standard** der WHATWG, es gibt keine Versionsnummern mehr

Nachschlagen: developer.mozilla.org (MDN). Gut erklärt, aktuell, auch auf Deutsch.

```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Gästebuch</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1>Gästebuch</h1>
    <!-- Der sichtbare Inhalt -->
    <script src="app.js"></script>
  </body>
</html>
```

Modernes HTML, kein Kompatibilitätsmodus

Wurzel. lang: Vorlesen, Übersetzen

Infos über die Seite, unsichtbar

Zeichensatz, sonst Umlaut-Salat

Handy: sonst winzig

Tab, Lesezeichen, Suchtreffer

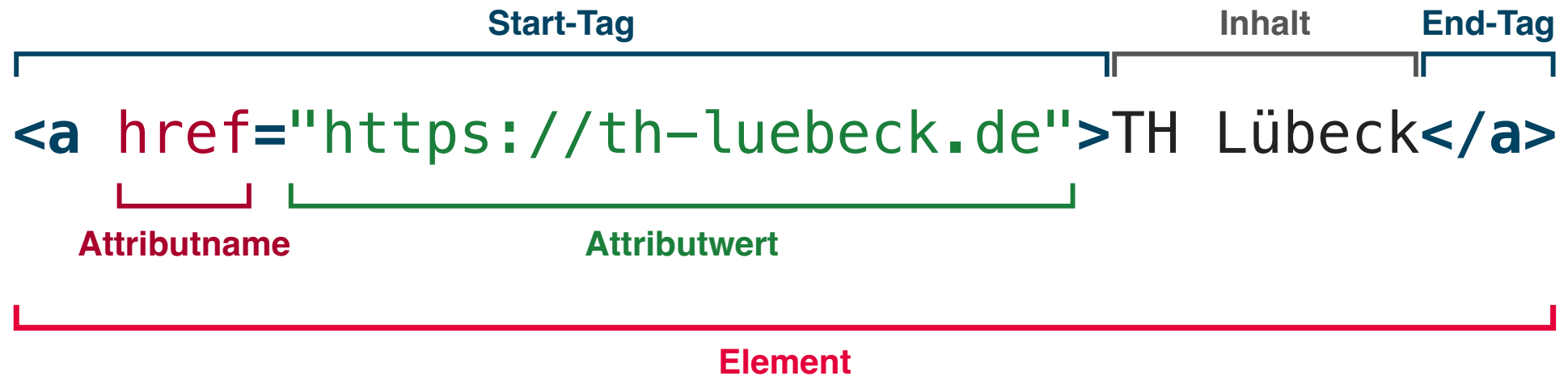
CSS laden: eigener Request

Alles, was man sieht

Hauptüberschrift, genau eine

Kommentar, steht nur im Quelltext

JavaScript laden, als Letztes



- Attribute stehen immer im Start-Tag: Name, Gleichheitszeichen, Wert in Anführungszeichen
- Elemente liegen ineinander, nie über Kreuz

Manche Elemente haben keinen Inhalt und darum auch kein End-Tag:

```
  
<input type="email" name="mail">  
<meta charset="utf-8">  
<br>
```

- Das sind genau die, die Sie im Formular und im `head` brauchen
- `<br />` mit Schrägstrich ist erlaubt, aber unnötig

Text, Links und Bilder ▾

Prüfen

Zurücksetzen

11 Elemente im Baum

HTML10 ZEILEN

```
<h1>Mitfahrbörse</h1>
<h2>Heute zum Campus</h2>
<p>
  Abfahrt <strong>7:45 Uhr</strong>
  am Bahnhof.
</p>
<p>
  <a href="https://www.th-luebeck.de">Zur
</p>

  <label for="ziel">Ziel</label>
  <input id="ziel" name="ziel" required>

  <label for="tag">Datum</label>
  <input id="tag" name="tag" type="date">

  <label for="plaetze">Freie Plätze</label>
  <input id="plaetze" name="plaetze"
    type="number" min="1" max="8">

  <button type="submit">Anbieten</button>
</form>
```

- `label for` gehört zur `id` des Felds
- `name` ist der Schlüssel, unter dem der Wert verschickt wird
- `type` liefert passende Eingabehilfen: Kalender, Zahlentastatur
- `required`, `min`, `max`: Prüfung im Browser, gratis

Was beim Absenden über die Leitung geht

method="get"

method="post"

JavaScript: fetch + JSON

☐ Datum ohne name

IM BROWSER

```
<form action="/fahrten" method="post">
```

Ziel

Campus

Datum

07.10.2026

Freie Plätze

3

Anbieten

ÜBER DIE LEITUNG

DATEN IM BODY

```
POST /fahrten HTTP/1.1
Host: mitfahren.example
Content-Type: application/x-www-form-urlencoded

ziel=Campus&tag=2026-10-07&plaetze=3
```

POST: Die Daten stehen im Body. Für alles, was etwas anlegt oder ändert.

Finde die Fehler ▾

Prüfen

Zurücksetzen

14 Elemente im Baum

HTML15 ZEILEN

```
<html>
  <head>
    <title>Kicker-Turnier</title>
  </head>
  <body>
    <h1>Kicker-Turnier</h1>
    <h3>Anmeldung</h3>
    <p>Jetzt <b><i>schnell</i></b></i> anmeld
    
```

1. **Z. 1** Es fehlt <!doctype html> ganz am Anfang.
2. **Z. 1** <html> ohne lang-Attribut, z. B. lang="de".
3. **Z. 1** Es fehlt <meta charset="utf-8"> im head.
4. **Z. 8** Überkreuzt: schließt, aber zuerst müsste </i> kommen (geöffnet in Zeile 8).
5. hat kein alt-Attribut.
6. Das Feld #team hat kein <label>.
7. Das Feld #team hat kein name-Attribut.

BROWSER

Kicker-Turnier

Anmeldung

Jetzt *schnell* anmelden!

 Teamname:

DOM-BAUMGRAU = ERGÄNZT

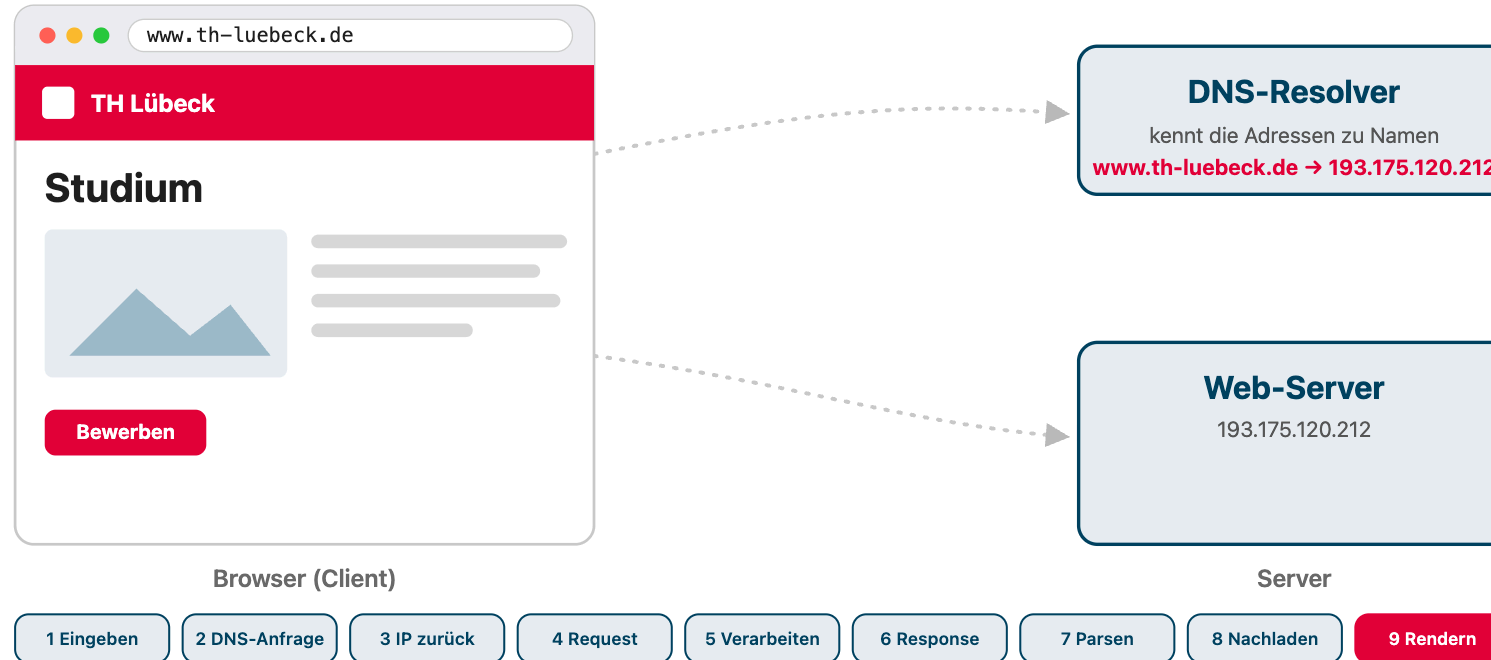
```
document
  html
    head
      title
        "Kicker-Turnier"
    body
      h1
        "Kicker-Turnier"
      h3
        "Anmeldung"
      p
        "Jetzt"
        b
          i
            "schnell"
        "anmelden!"
      img src="pokal.png"
      form
```

3 Min

zu zweit

```
document
├── html
│   ├── head
│   │   └── title
│   └── body
│       ├── h1
│       └── ul
│           ├── li
│           └── li
```

- Beim Parsen baut der Browser aus dem HTML einen Baum aus Objekten, das **Document Object Model**
- Die Wurzel ist `document`, jedes Element ist ein Knoten
- Der Elements-Tab der DevTools zeigt diesen Baum, nicht die Datei
- Mit JavaScript ändern Sie später genau diesen Baum



Erst jetzt **rendert** der Browser die Seite. Das alles dauert meist weniger als eine Sekunde.

1 Von der URL zur Seite

2 HTTP

3 Wo läuft was im Projekt?

4 HTML

5 Selbst bauen & Ausblick

Jetzt Sie

Nur HTML, kein CSS:

1. `header` mit Überschrift und `nav`
2. In `main` eine Liste, z. B. Ihre wichtigsten Funktionen
3. Ein Formular mit mindestens zwei Feldern samt `label`
4. „Prüfen“ klicken und den DOM-Baum ansehen

12 Min

allein oder zu zweit



swe1.mylab.th-luebeck.de/web-02

dort „HTML-Werkstatt“. Grundgerüst ist schon da, „Als Datei speichern“ nicht vergessen

Fingerübung, kein Prototyp. Die Wireframes in P3 sind bewusst **kein** HTML.

- Das WWW ist ein Dienst des Internets: Browser fragen, Server antworten, per HTTP
- Vor jedem Request steht DNS, nach jeder Response steht ein Statuscode
- GET holt, POST schickt. Formulare können beides, Ihr Projekt nutzt `fetch` und JSON
- HTML beschreibt Struktur: semantische Elemente, `label` und `name` im Formular
- Der Browser macht daraus das DOM, und die DevTools zeigen Ihnen alles davon

1. P1 abgeben: `docs/IDEA.md` , Tag `p1-abgabe` , bis **So 11.10., 23:59 Uhr**
2. DevTools auf einer Seite Ihrer Wahl öffnen: Wie viele Requests sind es, welcher ist der größte?
3. Ihre HTML-Seite von heute aufheben, wir bauen darauf auf

Di 13.10. bei Leif

- Versionskontrolle

Mi 14.10.

- Nächste Web-Vorlesung