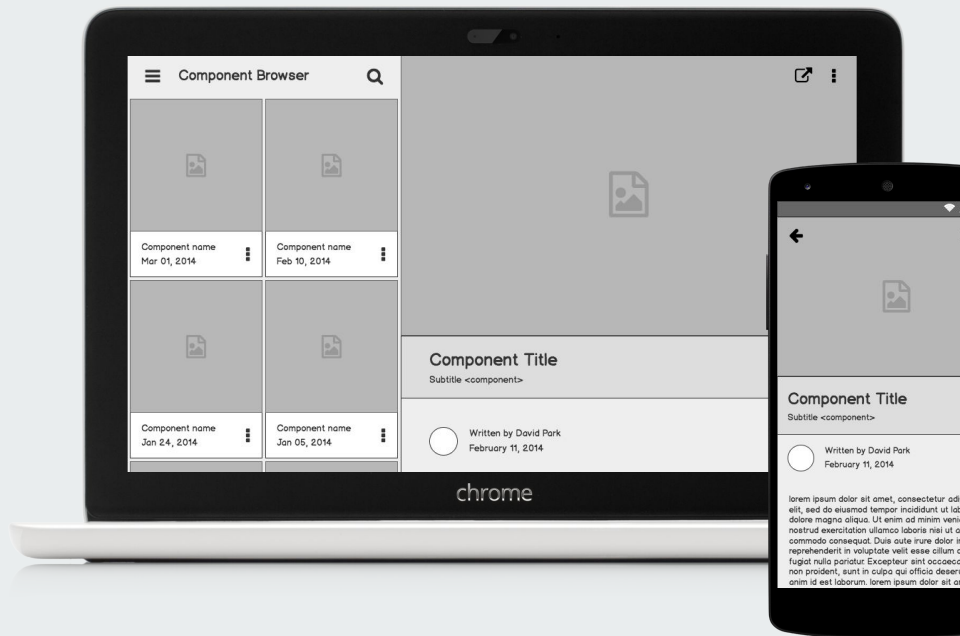




Grundlagen Webprogramm ierung

Tobias Joschko, MongoDB
Daniel Münch, SAP



Inhalt

Intro + Allgemeines

HTML + CSS

JavaScript

Vue + Quasar

Backend + Apis

Intro + Allgemeine Themen



Organisatorisches

- 2 Dozenten
 - Daniel Münch, SAP
 - Tobias Joschko, MongoDB
- 11 Termine
- Bewertung:
 - Klausur muss mindesten 50% der Note ausmachen
 - Die anderen 20-50% können von Hausaufgaben bzw. Labor kommen
- GitHub repository für Code Beispiele und Übungen



Inhalt

- Grundlagen zu HTML + CSS
- JavaScript Einführung
- Moderne Web App Entwicklung mit Frontend Frameworks (Vue/Quasar)
- Backend/Webserver (Express) + API Grundlagen
- Datenbanken (NoSQL vs SQL)
- Admin Panel mit Authentifizierung
- Web Deployment + GitHub
- Grundlagen zu Tests



Webentwicklung

1

Webdesign: Das visuelle Design einer Website, einschließlich Layout, Farbschema und Benutzererfahrung.

2

Web-Publishing: Das Veröffentlichen von Inhalten auf einer Website, oft durch Content-Management-Systeme.

3

Web-Programmierung: Die Entwicklung von Webanwendungen und Funktionalitäten, die auf Server- und Clientseite laufen.

4

Datenbankverwaltung: Das Organisieren und Verwalten von Daten, die für das Funktionieren einer Website benötigt werden, oft unter Verwendung von Datenbanksystemen wie MySQL oder MongoDB.



Beispiel

E-Commerce-Website

Eine E-Commerce-Website beinhaltet
Webdesign für die Benutzeroberfläche,
Web-Publishing für
Produktbeschreibungen,
Web-Programmierung für
Warenkorb-Funktionalitäten und
Datenbankverwaltung für die Speicherung
von Kundendaten.



Grundlagen

- Allgemeine Entwicklungskonzepte
- Tech stack (LAMP vs. MEVN)
- APIs
- Server vs. Client side (backend - frontend) Entwicklung
- Deployment
- Setup



Grundlegende Entwicklungsprinzipien

- **Agile Entwicklung:** Iterative Entwicklung, die Flexibilität und Kundeneinbindung betont.
- **Responsive Design:** Ansatz zur Webentwicklung, der sich an verschiedene Bildschirmgrößen anpasst.
- **Code-Wiederverwendung und -Modularität:** Nutzung von Modulen oder Komponenten, um Code effizienter zu gestalten.
- **Versionierung:** Einsatz von Tools wie Git zur Verwaltung verschiedener Versionen des Codes.



Best Practices in der Entwicklung

- **Clean Code:** Klare und verständliche Code-Schreibung.
- **Testgetriebene Entwicklung (TDD):** Erst Tests schreiben, dann Code.
- **Sicherheit:** Implementierung von Sicherheits Praktiken, um Angriffe und Datenlecks zu verhindern.
- **Performance-Optimierung:** Techniken zur Beschleunigung der Ladegeschwindigkeit und Effizienz von Webseiten.

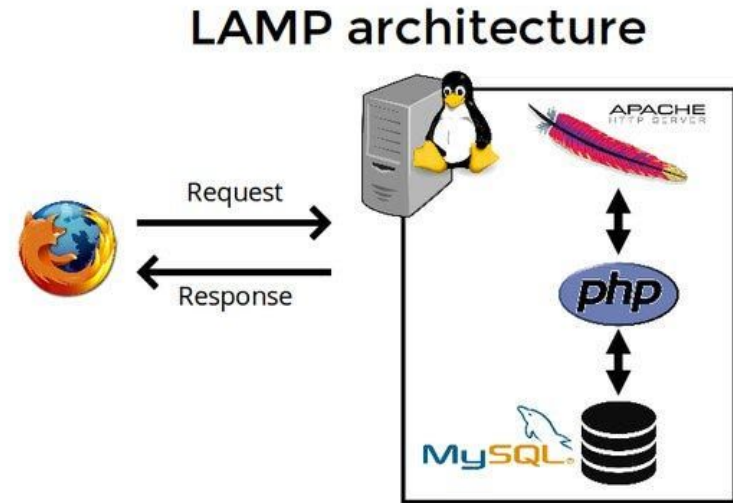
LAMP Stack

Linux: Das Betriebssystem.

Apache: Der Webserver.

MySQL: Das Datenbanksystem.

PHP/Perl/Python: Programmiersprachen



MEVN Stack



MongoDB: Die Datenbank.

Express.js: Das Backend-Framework.

Vue.js: Das Frontend-Framework.

Node.js: Die Laufzeitumgebung.





Grundlagen von APIs

- Definition: APIs (Application Programming Interfaces - Programmierschnittstellen) als Mittler zwischen verschiedenen Softwareanwendungen.
- REST (API): Ein populärer API-Stil, der HTTP-Anfragen nutzt (z.B. GET, POST, DELETE).

```
{  
  "original_title": "Interstellar",  
  "budget": 165000000,  
  "genres": [  
    { "id": 12, "name": "Adventure" },  
    { "id": 18, "name": "Drama" },  
    { "id": 878, "name": "Science Fiction" }  
  ],  
  "homepage": "http://www.interstellarmovie.net/",  
  "id": 157336,  
  "overview": "The adventures of a group of  
explorers who ...",  
  "poster_path": "/gEU2QniE6E77NI6lCU6MxlNBvIx.jpg"  
}
```

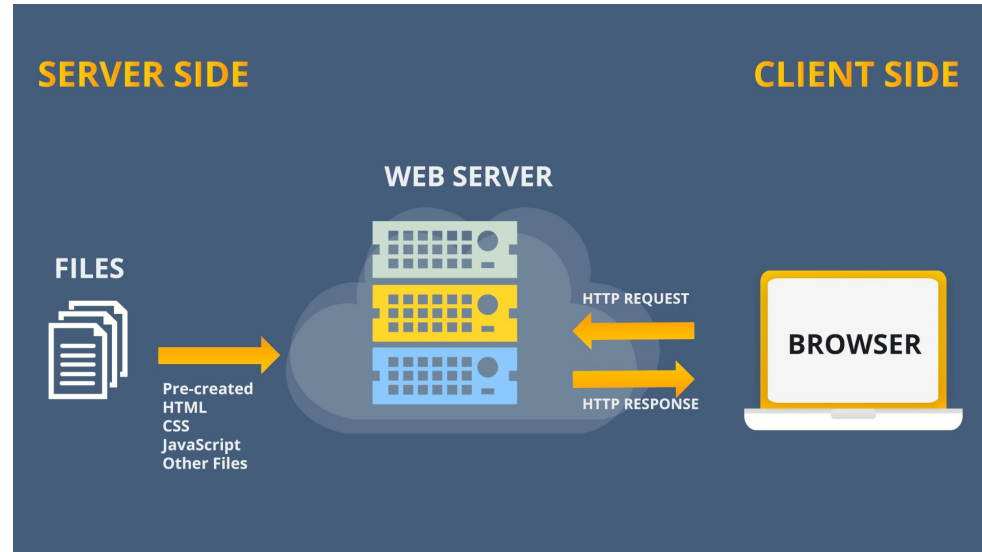


Nutzung von APIs

- Web Services: APIs werden zur Erstellung von Webseiten verwendet.
- Integration Dritter: Einbeziehung von Diensten wie Google Maps ,Twitter oder Instagram über APIs.

Server vs Client side

- Serverseitige Programmierung:
 - Erstellung von Logik, die auf dem Server läuft.
 - Benutzung unterschiedlicher Web Frameworks (z.B. Django, Express) und Verwendung verschiedener Programmiersprachen (z.B. Python, Javascript, PHP, Java, etc).
 - Datenbank-Interaktion des Servers.
- Frontend-Entwicklung:
 - Entwicklung von Benutzeroberflächen.
 - HTML/CSS/Javascript
 - Frameworks wie Vue, React oder Angular.





Deployment

- Deployment Prozess:
 - Staging vs. Production: Unterschiede zwischen Test- und Produktionsumgebungen.
 - Continuous Integration/Continuous Deployment (CI/CD): Automatisierung des Deployments.
- Hosting:
 - Hosting-Optionen: Unterschiede zwischen Shared Hosting, VPS, Cloud Hosting.
 - Wartung: Wichtigkeit der regelmäßigen Aktualisierung und Wartung.



Setup

- [IDE](#): VsCode
- [Node](#): Laufzeitumgebung
- Npm: Paketverwaltung
- [Pexels](#): Webseite mit Lizenfreien Bildern

HTML + CSS



Was ist HTML

Hyper Text Markup Language

- Grundlegende Sprache um Web um Webseiten und Anwendungen zu strukturieren
- HTML-Seiten sind **statisch** (sie liegen vorgefertigt und abrufbereit auf dem (Server))
- HTML ist das grundlegende Gerüst aller Webseiten und bildet die Basis für weitere Technologien wie CSS und JavaScript



Was ist HTML

- HTML ist keine Programmiersprache
 - Es gibt keinerlei Schleifen und bedingte Verzweigungen zur Steuerung des Codes.
 - If / while / for... kommt hier nicht vor!
 - Die Verwendung von Variablen ist nicht möglich
- HTML Code allein ist niemals dynamisch
 - Der Browser fordert die Webseite vom Server an und stellt sie lediglich dar
 - Die Benutzer Interaktion beschränkt sich im wesentlichen auf Links und Formulare, die ausgefüllt werden können



Struktur einer HTML-Seite

HTML verwendet sogenannte 'Tags' oder 'Elemente' um Inhalte wie Text, Bilder, Videos und Hyperlinks zu strukturieren.

- `<!DOCTYPE html>`: Deklariert den Dokumenttyp und die HTML-Version.
- `<html>`: Das Wurzelement, das den gesamten HTML-Inhalt umschließt.
- `<head>`: Enthält Metadaten, Titel, Skripte und andere Ressourcen.
 - `<title>` Fenstertitel im Browser
- `<body>`: Der Hauptteil der Webseite, der sichtbare Inhalt für den Benutzer.
- Texte und Bilder sollten idealerweise innerhalb eines Block Elements stehen (z.B. `<p>` oder `<div>`)

```
<!DOCTYPE html>
<html>
<head>
  <title>Seitentitel</title>
</head>
<body>
  <h1>Willkommen auf meiner Webseite</h1>
  <p>Dies ist ein Paragraph.</p>
</body>
</html>
```





Wichtige HTML-Elemente

- Überschriften (<h1> bis <h6>): Definieren Überschriften von verschiedenen Ebenen.
- Paragraph (<p>): Für Textabsätze.
- Anker (<a>): Zum Erstellen von Hyperlinks.
- Bild (): Zum Einbinden von Bildern.
- Listen (,): Ungeordnete (bullets) oder geordnete (nummern) Listen.
- Container (<div>): Zum Gruppieren von Inhalten und Layout-Steuerung.



HTML Versionen

HTML 1.0	1991	Sehr grundlegend und eingeschränkt in den Möglichkeiten.
HTML 2.0	1995	Wird standard im Webdesign
HTML 3.2	1997	Unterstützung neuer Tags
HTML 4.01	1999	Führte Stylesheets (CSS) und verbesserte Unterstützung für Skriptsprachen (wie JavaScript) ein
HTML 5	2014	Aktuell die neueste Version. HTML5 brachte viele bedeutende Verbesserungen, darunter neue APIs, die Integration von Multimedia-Elementen (wie Audio und Video) ohne zusätzliche Plugins, und bessere Unterstützung für moderne Webanwendungen



Inline, Block & Empty Elemente

- `<tags>` innerhalb des `<body>` werden in unterschiedliche Arten von Elementen unterteilt
 - **Block-Elemente:** Automatischer Zeilenumbruch vor und nach dem Element
 - `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`
 - `<p>`, `<blockquote>`, ``, ``, ``, `<table>`
 - `<div>`, `<form>`
 - **Inline-Elemente:** Innerhalb des Textflusses von Block-Elementen.
 - `<a>`, ``, ``, ``, `<i>`, `<u>`, `<q>`, `<input>`
 - **Empty (Selbstschließende) Elemente:** Enthalten keine Inhalte, können aber Attribute haben, die Inhalte anzeigen.
 - ``
 - Horizontale Linie: `<hr/>`, Zeilenumbruch: `
` (block)



Weiter Element-Typen

- Sections:
 - `<header>`: für Kopfzeile, ersetzt `<div id="header">`
 - `<nav>`: für Navigation, ersetzt `<div id="menu">`.
 - `<article>`: enthält den eigentlichen Inhalt.
 - `<footer>`: für Fußzeile, ersetzt `<div id="footer">`.
 - `<aside>`: für Inhalte wie Werbung oder eine Sidebar.
 - `<section>`: Platzhalter für thematische Inhaltsabschnitte.



Auswahl der Schrift/Font

- Neue Schrift/Font auswählen über [Google Fonts](#)

```
<head>
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Source+Code+Pro:ital,wght@0,400;1,200&display=swap" rel="stylesheet">
</head>
```



Übung 1: Grundaufbau

- Erstellen Sie eine Webseite mit den Grundelementen (html, head, title, body)
- Fügen Sie folgende Elemente ein:
 - Überschrift
 - Paragraph
 - Ein Image
 - Anderes Font auswählen und einfügen

Seitenlayout mit Tabellen

```
<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>Alter</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Max Mustermann</td><td>30</td>
    </tr>
    <tr><td>Erika Musterfrau</td><td>25</td></tr>
  </tbody>
  <tfoot>
    <tr>
      <td>Gesamt</td>
      <td>55 Jahre</td>
    </tr>
  </tfoot>
</table>
```

Tabelle <table>

Zeile <tr>

Zelle <th>

Zelle <th>

Zelle <th>

Zeile <tr>



Zelle <td>

Zelle <td>

Zelle <td>

Übung 2

- Erstellen einer Top 3 Filmeliste
 - Position, Titel, Bild und Streamingplattform
 - Übersichtlich gestalten → Tabelle verwenden
 - Überschriften sollen sich vom Rest abheben → <th>

Position	Bild	Filmtitel	Streaming-Plattform
1		The Matrix	Netflix
2		Dune	Amazon Prime



Listen Element

```
<h2>Geordnete Listen (mit Nummerierung)</h2>
  <ol type="A" start="3">
    <li>Erstes Element</li>
    <li>Zweites Element</li>
    <li>Drittes Element</li>
  </ol>

<h2>Ungeordnete Liste (ohne Nummerierung)</h2>
<ul>
  <li>Erstes Element</li>
  <li>Zweites Element</li>
  <li>Drittes Element</li>
</ul>
```

Geordnete Listen

- C. Erstes Element
- D. Zweites Element
- E. Drittes Element

Ungeordnete Liste

- Erstes Element
- Zweites Element
- Drittes Element



Hyperlinks

```
<a href="https://www.example.com" target="_blank">Öffne Beispiel.com in einem neuen Tab</a>
```

```
a href="mailto:someone@example.com">Sende eine E-Mail</a>
```

```
<!-- Link, der zu einem Abschnitt der Seite führt -->
```

```
<a href="#section1">Gehe zu Abschnitt 1</a>
```

```
<!-- Später auf der Seite: Der Zielabschnitt -->
```

```
<div id="section1">
```

```
  <h2>Abschnitt 1</h2>
```

```
</div>
```


Übung 3

- Verlinken der Filmtitel mit der Filmwebsite
- Streaming Plattform in Liste umbauen damit man die unterschiedlichen Plattformen anzeigen kann

Position	Bild	Filmtitel	Streaming-Plattform
1	 The poster for the movie 'The Matrix' features the main characters: Morpheus, Neo, Trinity, and the Architect, standing in a dark, rainy environment. The title 'THE MATRIX' is prominently displayed at the top.	The Matrix	<ul style="list-style-type: none">• Amazon Prime• Netflix• Disney+



Formulare

Input-Felder (<input>)

Textarea (<textarea>)

Select-Box (<select> mit <option>-Elementen)

Label (<label>)

Formulare

Name:

E-Mail:

Geschlecht:

☐ Männlich

☐ Weiblich

Interessen:

☐ Programmieren

☐ Musik

Land:

Nachricht:

Absenden



Input-Felder (<input>):

- Verschiedene Typen wie **text**, **email**, **password**, **radio**, **checkbox**, etc.
- Wichtige Eigenschaften:
 - **name**: Der Name des Eingabefeldes, wichtig für die Verarbeitung auf der Serverseite.
 - **id**: Eindeutige Kennung, wird oft für CSS-Styling und Verbindung mit <label> verwendet.
 - **value**: Der Wert des Eingabefeldes, der an den Server gesendet wird.



Weitere Formularfelder

- **Textarea** (<textarea>):
 - Für mehrzeilige Texteingabe.
 - Ähnliche Eigenschaften wie Input-Felder (name, id).
- **Select-Box** (<select> mit <option>-Elementen):
 - Für Dropdown-Menüs.
 - Jede <option> kann einen value haben, der gesendet wird, wenn sie ausgewählt ist.
- **Label** (<label>):
 - Verbessert die Zugänglichkeit und Benutzerfreundlichkeit.
 - Durch Verknüpfung mit einem Formularelement (mittels for-Attribut, das den id-Wert des Elements enthält) wird das Klicken auf das Label gleichgesetzt mit dem Klicken auf das Formularelement.



Übung 3

- Kontaktformular erstellen
 - Namensfeld → Required
 - Email Adressfeld → Required
 - Textarea für eine Nachricht



CSS

Cascading Style Sheets



Einführung in CSS

- Definition: CSS (Cascading Style Sheets) ist eine Stylesheet-Sprache, die für die Beschreibung des Aussehens und der Formatierung von Dokumenten verwendet wird, die in einer Markup-Sprache wie HTML geschrieben sind.
- Zweck: CSS ermöglicht es Entwicklern und Designern, das Design und Layout von Webseiten zu kontrollieren.
- Trennung von Inhalt und Design: CSS trennt den Inhalt (HTML) vom Design, was die Wartung und das Webdesign vereinfacht.



Grundlagen von CSS

- Selektoren: Der Teil eines CSS-Rulesets, der bestimmt, auf welche Elemente der Stil angewendet wird. Z.B. `p`, `#id`, `.class`.
- Deklarationen: Ein Style-Regel, bestehend aus einem Eigenschaftsnamen und einem Wert. Z.B. `color: red;`.
- Regelset: Eine Kombination aus Selektor(en) und einer Deklaration. Z.B. `p { color: red; }`.



CSS in HTML einbinden

Externe Stylesheets: Mittels `<link rel="stylesheet" href="styles.css">` im `<head>`-Bereich.

Interne Stylesheets: Mit `<style>`-Tags im `<head>`-Bereich.

Inline-Stile: Direkt im `style`-Attribut von HTML-Elementen.



CSS-Beispiel

HTML-Code:

```
<p class="text-red">Dieser Text ist rot.</p>
```

CSS-Code:

```
.text-red {  
    color: red;  
}
```



Fortgeschrittene CSS-Konzepte

Box-Modell: Beschreibt das Layout von Blockelementen und beinhaltet Margin, Border, Padding und Content.

Flexbox: Ein Layout Modell, das es einfacher macht, komplexe Layouts zu gestalten.

CSS Grid: Ein leistungstarkes Layoutsystem für zweidimensionale Layouts.

Layout mit Flexboxen

```
display: flex;  
justify-content: space-between;
```



justify-content

flex-start



flex-end



center



space-between

