

ANGULAR  
**ARCHITECTS**

# Einführung in HTML, CSS & JavaScript

AngularArchitects.io | @ManfredSteyer

# Agenda

- #1 Überblick Webentwicklung
- #2 Einführung HTML
- #3 Einführung CSS
- #4 Einführung JavaScript
- #5 Einführung TypeScript

# Einführung in die Webentwicklung

# Webentwicklung ist Komplex

- Verlagerung klassischer Desktop-Applikationen in den Browser
- Unterschiedliche Plattformen und Endgeräte müssen unterstützt werden
- Unterschiedliche Webstandards, abhängig von Browsern
- Abbildung komplexer Businessprozesse & Corporate Designs

# Webentwicklung ist einfach

- Single Page Application (SPA) Frameworks wie Angular, Vue oder React
- CSS-Frameworks wie Bootstrap oder Tailwind CSS
- TypeScript als moderne Programmiersprache
- Einheitliche Browser APIs unabhängig von Betriebssystemen
- Open Source Bibliotheken für komplexe Anforderungen wie Charts und Formulare, unabhängig von Frameworks

# Einführung HTML

# Was ist HTML?

- HTML steht für **HyperText Markup Language**
- Grundgerüst jeder Webseite
- Bestimmt die Struktur und den Inhalt einer Seite
- HTML-Dokumente bestehen aus Elementen (Tags)

# Grundstruktur eines HTML-Dokuments



index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meine erste HTML-Seite</title>
  </head>
  <body>
    <h1>Willkommen zu meinem Workshop</h1>
    <p>Dies ist eine einfache HTML-Seite.</p>
  </body>
</html>
```

# Grundstruktur eines HTML-Dokuments

- <!DOCTYPE html>: Dokumenttyp-Deklaration
- <html>: Wurzelement des HTML-Dokuments
- <head>: Metadaten des Dokuments
- <title>: Titel der Seite (sichtbar im Browser-Tab)
- <body>: Inhalt der Seite



# Demo

## Hello World! – mit HTML

# HTML Elemente & Attribute



index.html

```
<!-- Ein HTML-Element einen Start- und End-Tag -->
```

```
<placeholder>Ich bin ein Element</placeholder>
```

```
<!-- Ich habe keinen Inhalt und daher keinen End-Tag -->
```

```
<br>
```

```
<!-- HTML-Elemente lassen sich über Attribute erweitern -->
```

```
<placeholder foo="foo" bar="bar">Ich habe Attribute</placeholder>
```

# Überschriften



index.html

```
<h1>Lorem ipsum</h1>
<h2>Lorem ipsum</h2>
<h3>Lorem ipsum</h3>
<h4>Lorem ipsum</h4>
<h5>Lorem ipsum</h5>
<h6>Lorem ipsum</h6>
```

# Absätze



index.html

```
<p>  
  Lorem, ipsum dolor sit amet consectetur adipisicing elit. Ipsum esse totam  
  minima, possimus tempore cumque maxime nihil velit facilis quo doloribus, rem  
  maiores ad quos reiciendis laudantium quae enim similiique.  
</p>
```

# Listen



index.html

```
<li>
  <ul>
    Lorem ipsum
  </ul>
  <ul>
    Lorem ipsum
  </ul>
</li>

<li>
  <ol>
    Lorem ipsum
  </ol>
  <ol>
    Lorem ipsum
  </ol>
</li>
```

# Links & Atribute



index.html

```
<a href="https://angulararchitects.io">Klick mich!</a>
```

# Bilder



# Tabellen



The screenshot shows a dark-themed browser window with three circular tabs at the top left. The main content area displays an HTML file named "index.html". The code is as follows:

```
<table>
  <thead>
    <tr>
      <th>ID</th>
      <th>From</th>
      <th>To</th>
      <th>Date</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>#1</td>
      <td>Berlin</td>
      <td>Salzburg</td>
      <td>2019-01-01</td>
    </tr>
  </tbody>
</table>
```

# Formulare & Buttons



index.html

```
<form>
  <label for="id">ID:</label>
  <input type="text" id="id" name="id" />

  <label for="from">From:</label>
  <input type="text" id="from" name="from" />

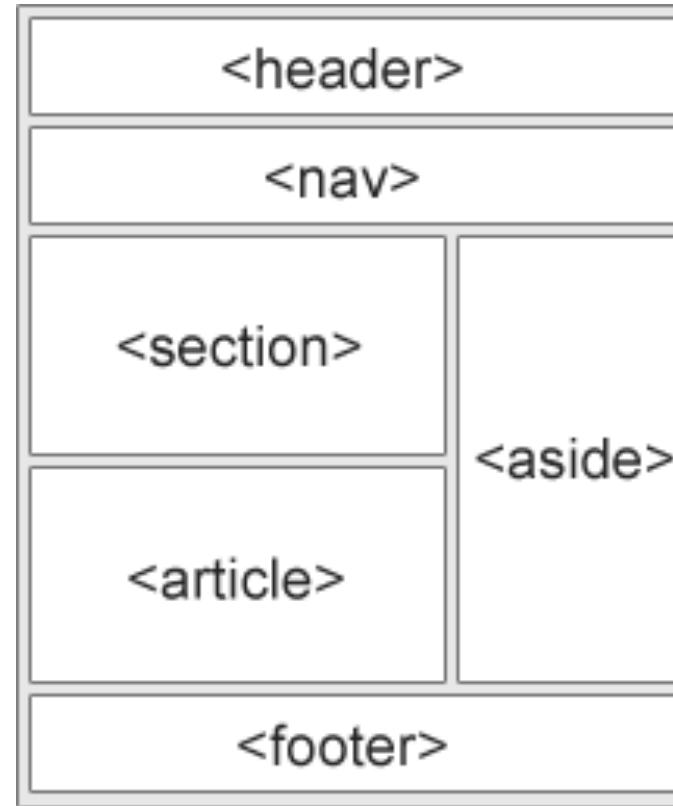
  <label for="to">To:</label>
  <input type="text" id="to" name="to" />

  <label for="date">Date:</label>
  <input type="date" id="date" name="date" />

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

# HTML Layout Elemente

- Header
- Nav
- Section
- Article
- Aside
- Footer
- Details
- Summary



# **Exercise #1**

## **Erstelle deine erste HTML-Seite**

# Exercise #1: Aufgabenstellung

- Erstelle deine erste index.html Datei
- Die Seite soll einen Header und einen Footer darstellen, sowie ein Menü auf der linken Seite
- Der Content soll eine Suchmaske mit zwei Input-Felder (From & To) sowie wie einen Button (Suche) anzeigen
- Unterhalb der Suche soll eine Liste/Tabelle von Flügen dargestellt werden
- Verwende die gezeigten HTML-Elemente oder experimentiere selber herum ☺

# Einführung CSS + Bootstrap

# Einführung CSS

- CSS steht für **Cascading Style Sheets**
- Trennung von Struktur (HTML) und Layout (CSS)
- Bestimmt das Aussehen und die Gestaltung einer Webseite

# Einbindung von CSS: Inline

- **Inline-Styles:** Direkt im HTML-Tag



Index.html

```
<h1 style="color: blue;">Willkommen</h1>
```

# Einbindung von CSS: Interne

- **Internes CSS:** Innerhalb des `<style>`-Tags im `<head>`-Bereich



Index.html

```
<style>
body {
    background-color: #f0f0f0;
}
</style>
```

# Einbindung von CSS: Extern

- **Externes CSS:** Über eine separate CSS-Datei



# CSS-Eigenschaften: Farben & Hintergründe



index.css

```
body {  
    background-color: #f0f0f0;  
    color: #333;  
}
```

# CSS-Eigenschaften: Schriften



index.css

```
h1 {  
    font-family: Arial, sans-serif;  
    font-size: 2em;  
}
```

# CSS-Eigenschaften: Abstände (Margin & Padding)



index.css

```
p {  
  margin-left: 20px;  
  padding: 10px 10px 10px 10px;  
}
```

# CSS-Eigenschaften: Rahmen (Border)



index.css

```
div {  
    border: 1px solid #000;  
}
```

# Exercise #2: Anwenden von CSS

# **Exercise #2: Anwenden von CSS**

- Ziel: Verwende CSS, um die HTML-Seite zu stylen
- Erstelle eine neue CSS-Datei `styles.css` und binde diese in der `index.html` ein
- Verwende eine andere Schriftart
- Gestalte die Überschriften, die Input-Felder und Butten sowie die Tabelle

# Responsive Design

# Einführung Responsive Design

- Ziel: Optimale Darstellung auf allen Geräten (Desktop, Tablet, Smartphone)



```
index.css

@media (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

# Exercise #3: Responsive Design anwenden

# **Exercise #3: Responsive Design anwenden**

- Ziel: Passe die Webseite an unterschiedliche Bildschirmgrößen an
- Skaliere die Schriftgrößen auf Tablets und Smartphones herunter
- Lass den Inhalt der Tabelle horizontal scrollen

# CSS-Frameworks am Beispiel Bootstrap

# Bootstrap

- CSS-Framework für schnelleres und einfacheres Webdesign
- Enthält vorgefertigte Komponenten und Utility-Klassen
- Anpassbar an eigene Style Vorgaben
- JavaScript Funktionalitäten wie Dropdowns
- Einbindung lokal oder per CDN



# Demo

# Einbindung von Bootstrap

# Zusammenfassung

- CSS ermöglicht das Design und Layout von Webseiten
- Responsive Design sorgt für eine optimale Darstellung auf allen Endgeräten
- CSS-Frameworks wie Bootstrap bieten vorgefertigte Komponenten für die schnelle Entwicklung von Web-Apps

# Einführung JavaScript

# Einführung JavaScript

- JavaScript ist eine Skriptsprache für dynamische Webinhalte
- Macht Webseiten interaktiv
- Läuft im Browser, aber auch serverseitig (z.B. mit Node.js)
- Wird unter dem ECMA-Script Standard stetig weiterentwickelt

# Einbindung von JavaScript in HTML

- Inline JavaScript: Direkt im HTML-Tag

```
<button onclick="alert('Hallo!')">Klick mich</button>
```

# Einbindung von JavaScript

- Internes JavaScript: Innerhalb des <script>-Tags

```
...
<script>
function zeigeNachricht() {
    alert('Hallo!');
}
</script>
<button onclick="zeigeNachricht()">Klick mich</button>
```

# Einbindung von JavaScript

- Externes JavaScript: Über eine separate JavaScript-Datei

```
<script src="script.js"></script>
```

# Grundlegende JavaScript-Syntax

## – Variablen

The image shows a dark-themed code editor window. In the top right corner, the file name "index.js" is visible. To its left are three small circular icons. The main area of the editor contains the following three lines of JavaScript code:

```
var name = 'Max';
let alter = 25;
const geburtsjahr = 1996;
```

- **Datentypen:** String, Number, Boolean, Object, Array, Function
- **Operatoren:** Arithmetische, Zuweisungs-, Vergleichs- und Logische Operatoren

# Funktionen in JavaScript

```
index.js

● ● ●

// Declaration
function begruessung() {
    console.log('Hallo Welt!');
}
begruessung();

// Funktionsausdruck
const begruessung = function() {
    console.log('Hallo Welt!');
}
begruessung();

// Lambda-Funktion
const begruessung = () => {
    console.log('Hallo Welt!');
}
begruessung();
```

# Ereignisse und Event-Listener

```
...  
index.js  
  
// HTML-Ereignis  
<button onclick="zeigeNachricht()">Klick mich</button>  
  
// Event-Listener  
document.querySelector('button').addEventListener('click', function() {  
    alert('Button geklickt!');  
});
```

# Exercise #4: Einbindung von JavaScript

# **Exercise #4: Einbindung von JavaScript**

- Ziel: Binde JavaScript in die bestehende Anwendung ein
- Verwende einen Click-Listener um den Klick auf den Suche-Button zu registrieren und gebe eine Alert-Meldung aus

# Einführung DOM-Manipulation mit JavaScript

# Einführung DOM-Manipulation

- DOM steht für Document Object Model: Baumstruktur der HTML-Dokumente

```
index.js

● ● ●

// Selektierung von HTML-Elementen
const element = document.getElementById('meinElement');
const elements = document.getElementsByClassName('meineKlasse');
const query = document.querySelector('.meineKlasse');

// Hinzufügen von HTML-Elementen
const neuesElement = document.createElement('div');
neuesElement.textContent = 'Hallo!';
document.body.appendChild(neuesElement);
```

# Exercise #5: DOM-Manipulierung mit JavaScript

# Exercise #5: DOM-Manipulation mit JavaScript

- Ziel: Verwende die DOM-API von JavaScript um die HTML-Seite interaktiv zu gestalten
- Füge der Liste von Flügen weitere Zeilen durch einen Klick auf den Search-Button hinzu
- Optional: Füge einen Clean-Button hinzu um die Liste der Flüge zu leeren

# Asynchrone Entwicklung mit JavaScript

# Asynchrone Entwicklung mit JavaScript

- Asynchron: Ausführung von Code ohne Blockieren des Main-Threads

```
index.js

function ladeDaten(callback) {
    setTimeout(function() {
        callback('Daten geladen');
    }, 1000);
}

ladeDaten(function(nachricht) {
    console.log(nachricht);
});
```

# Asynchrone Entwicklung: Promise

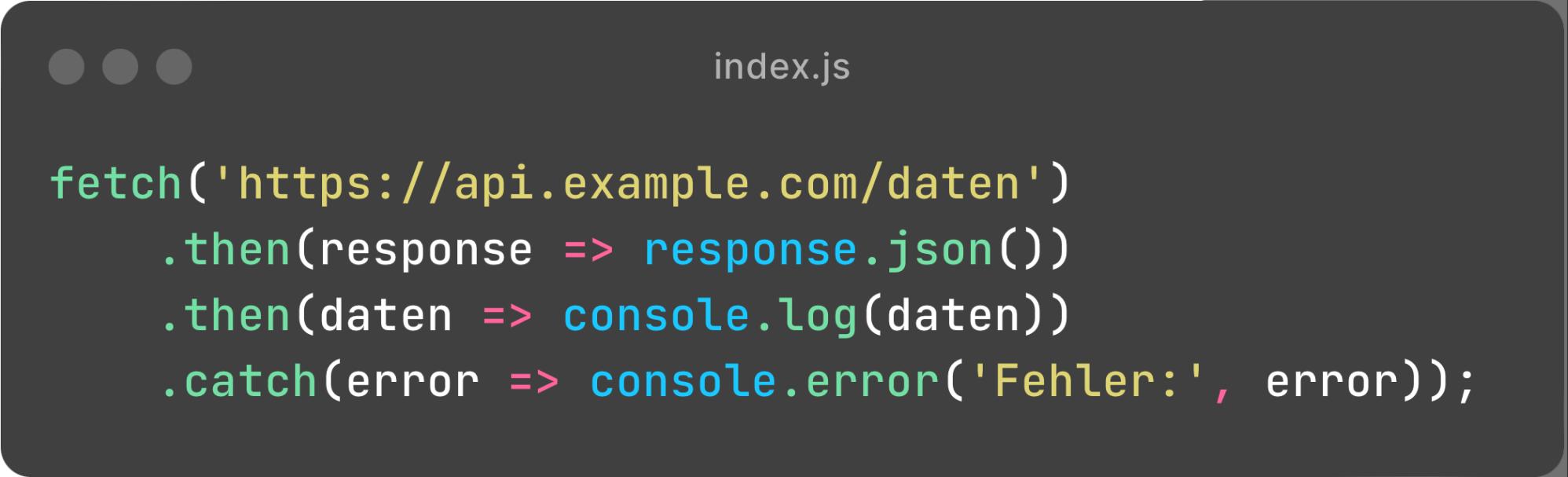


index.js

```
const promise = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve('Daten geladen');
  }, 1000);
});
promise.then((nachricht) => {
  console.log(nachricht);
});
```

# API-Aufrufe mit der Fetch-API

# Fetch-API



index.js

```
fetch('https://api.example.com/daten')
  .then(response => response.json())
  .then(daten => console.log(daten))
  .catch(error => console.error('Fehler:', error));
```

# Exercise #6: Interaktive Flugsuche

# Exercise #6: Interaktive Flugsuche

- Ziel: Verwendung der DOM-, Event- und Fetch-API
- Verwende die Flights-API um eine Liste von Flügen zu laden
- Übergebe der API die Werte der Input-Felder
- Lade die Flüge nach einem Klick auf den Search-Button
- Rendere die Liste der Ergebnisse mit Hilfe der DOM-API
- Optional: Behandle Fehler oder ein leeres Ergebnis mit einer passenden Meldung