Tag 5 HTML: Beyond the Basics

Was ist ein Favicon?

Ein Favicon (Kurzform für "favorite icon") ist ein kleines Symbol, das mit einer Website verknüpft ist. Es erscheint im Browser-Tab, in der Lesezeichenleiste und an anderen Stellen im Browser, wo deine Seite visuell dargestellt wird.

Obwohl es sich nur um ein kleines Icon handelt (oft nur 16x16 Pixel), kann es einen großen Einfluss auf das Branding, die Wiedererkennung und die Benutzerfreundlichkeit einer Website haben.

- In der Registerkarte (Tab) des Browsers
- In der Lesezeichen-/Favoritenleiste
- In der Chronik (Verlauf) des Browsers
- Beim Speichern der Website auf dem Startbildschirm mobiler Geräte

Größe und Auflösung

Ein Favicon ist in der Regel ein quadratisches Bild, typischerweise in einer dieser Größen:

- 16×16 Pixel – Standard für Browser-Tabs
- 32×32 Pixel – Für höher auflösende Anzeigen
- 48×48 Pixel oder größer – Für mobile Geräte und Retina-Displays

Tipp: Um auf verschiedenen Geräten und in verschiedenen Auflösungen gut auszusehen, sollten mehrere Versionen in verschiedenen Größen bereitgestellt werden.

Dateiformate

Favicons können in verschiedenen Formaten vorliegen:

Format	Beschreibung
.ico	Traditionelles Icon-Format für Favicons (sehr kompatibel)
.png	Hohe Qualität und Transparenz möglich
.gif	Wird selten verwendet, erlaubt einfache Animationen

Favicon in HTML einbinden

Um ein Favicon auf deiner Website zu verwenden, musst du im `<head>`-Bereich deiner HTML-Datei einen entsprechenden Link einfügen.

```
<link rel="icon" href="favicon.ico" type="image/x-icon">
```

Erklärung:

- `rel="icon"` beschreibt die Beziehung zwischen der Datei und dem Dokument
- `href="favicon.ico"` gibt den Pfad zur Icon-Datei an
- `type="image/x-icon"` bestimmt den MIME-Typ

```
<link rel="icon" href="favicon.png" type="image/png">
```

Mehrere Größen für bessere Kompatibilität

Du kannst mehrere Versionen deines Favicons bereitstellen, indem du das Attribut `sizes` nutzt:

```
<link rel="icon" href="favicon.png" type="image/png" sizes="16x16 32x32">
```

Das bedeutet, dass der Browser die passendste Version für das jeweilige Gerät auswählt.

Speicherort des Favicons

Platziere die Favicon-Datei idealerweise im Root-Verzeichnis deiner Website:

```
/
├── index.html
├── favicon.ico
```

So wird sie vom Browser leicht gefunden – einige Browser suchen sogar automatisch im Root-Verzeichnis nach `favicon.ico`, auch ohne HTML-Link.

---

iFrame?

Ein iFrame (kurz für inline frame) ist ein HTML-Element, das es ermöglicht, Inhalte von einer externen Quelle in eine eigene Webseite einzubetten. Man kann sich iFrames wie Fenster vorstellen, die einen Blick auf eine andere Webseite oder ein anderes Dokument erlauben – ohne die eigene Seite zu verlassen.

Die grundlegende Syntax eines iFrames sieht so aus:

```
<iframe src="URL" width="Breite" height="Höhe" frameborder="0" allowfullscreen></iframe>
```

YouTube-Video einbinden

```
<iframe
  src="https://www.youtube.com/embed/VIDEO_ID"
  width="560"
  height="315"
  frameborder="0"
  allowfullscreen>
</iframe>
```

Erklärung:

- src – Die Quelle (z. B. YouTube-Video oder andere Webseite)
- width / height – Festlegung der Größe
- frameborder="0" – Kein sichtbarer Rahmen
- allowfullscreen – Erlaubt das Vollbild-Ansehen des Videos

Google Maps einbinden

```
<iframe
  src="https://www.google.com/maps/embed?pb=..."
  width="600"
  height="450"
  style="border:0;"
  allowfullscreen
  loading="lazy">
</iframe>
```

Eine andere Webseite einbinden

```
<iframe
  src="https://www.example.com"
  width="800"
  height="600">
</iframe>
```

Wichtige Attribute von iFrames

Attribut	Beschreibung	Werte / Beispiele
src	URL der eingebetteten Seite	src="https://example.com"
width / height	Größe des iFrames	width="600" height="400"
frameborder	Rahmen um das iFrame (veraltet, besser style)	frameborder="0"
allowfullscreen	Erlaubt Vollbildmodus bei Videos	allowfullscreen
scrolling	Steuert das Scrollverhalten im iFrame	yes, no, auto
sandbox	Aktiviert Sicherheitsbeschränkungen für eingebettete Inhalte	sandbox="allow-scripts"
loading	Steuert das Laden des iFrames (Performance-Optimierung)	lazy, eager
title	Barrierefreiheits-Attribut, beschreibt den eingebetteten Inhalt für Screenreader	title="YouTube Video"

Scrolling deaktivieren

```
<iframe src="https://example.com" width="600" height="400" scrolling="no"></iframe>
```

Sandbox nutzen

```
<iframe src="https://example.com" sandbox="allow-scripts allow-popups"></iframe>
```

Responsive iFrames

```
<iframe
  src="https://www.example.com"
  width="100%"
  height="315">
</iframe>
```

Die Breite passt sich damit automatisch dem umgebenden Container an.

Fortgeschrittener Tipp mit CSS:

```
<div style="position: relative; padding-bottom: 56.25%; height: 0; overflow: hidden;">
  <iframe
    src="https://www.youtube.com/embed/VIDEO_ID"
    style="position: absolute; top: 0; left: 0; width: 100%; height: 100%;"
    frameborder="0"
    allowfullscreen>
  </iframe>
</div>
```

Dieses Vorgehen wird oft für responsive Videos verwendet (z. B. YouTube).

Sicherheit und Einschränkungen

Ein iFrame lädt Inhalte von anderen Servern – das kann Sicherheitsrisiken bergen:

- Cross-Origin Policy: Einige Websites erlauben keine Einbettung in iFrames.
- sandbox-Attribut: Hilft, das Ausführen von Skripten oder Formularen einzuschränken.
- Datenschutz: Vorsicht bei eingebetteten Inhalten, die Nutzerverhalten tracken (z. B. durch Cookies).

HTML Entities und Emojis in HTML

1. HTML Entities

(auch: Character Entity References) sind spezielle Codes, die verwendet werden, um Zeichen darzustellen, die entweder in HTML eine besondere Bedeutung haben oder auf der Tastatur schwer einzugeben sind.

Aufbau einer HTML Entity

Jede Entity besteht aus drei Teilen:

- Einem kaufmännischen Und-Zeichen (`&`)
- Einem Code oder Namen
- Einem abschließenden Semikolon (`;`)

Beispiel: `<` wird im Browser als < (Kleiner-als-Zeichen) dargestellt.

Warum brauchen wir HTML Entities?

Einige Zeichen wie < oder & haben in HTML eine spezielle Bedeutung:

- < leitet ein HTML-Tag ein
- & beginnt eine Entity

Wenn du diese Zeichen einfach so in deinem HTML-Code einfügst, interpretiert der Browser sie als Steuerzeichen und nicht als Text. Das kann zu fehlerhafter Anzeige oder Funktionsproblemen führen.

Häufig verwendete HTML Entities

Entity	Symbol	Beschreibung
`<`	<	Kleiner-als-Zeichen
`>`	>	Größer-als-Zeichen
`&`	&	Kaufmännisches Und-Zeichen
`&quot;`	"	Anführungszeichen (gerade)
`&apos;`	'	Apostroph
`&copy;`	©	Copyright-Zeichen
`&reg;`	®	Eingetragenes Markenzeichen
`&trade;`	™	Warenzeichen (Trademark)
`&hearts;`	♥	Herzsymbol
`&starf;`	★	Gefüllter Stern
`&check;`	✓	Haken (Checkmark)
`&larr;`	←	Pfeil nach links
`&rarr;`	→	Pfeil nach rechts
`&hellip;`	…	Auslassungspunkte (Ellipsis)

---

2. Emojis in HTML verwenden

sind kleine, bunte Symbole, die Emotionen, Objekte oder Ideen darstellen. Sie stammen ursprünglich aus Japan und erfreuen sich weltweit großer Beliebtheit in Chats, sozialen Netzwerken und Webseiten.

Wie kann man Emojis einfügen?

Es gibt zwei Hauptwege:

- **HTML Entity (dezimal oder hexadezimal):** z. B. `&#128516;` oder `&#x1F604;`
- **Direktes Einfügen des Emojis:** Viele Editoren und Browser unterstützen direkt eingefügte Emojis 😊

Wo werden Emojis verwendet?

- Innerhalb von HTML-Tags wie `<p>`, `<span>`, `<h1>` usw.
- In Buttons, Überschriften, Kommentaren oder dekorativen Texten

Beliebte Emojis und ihre Codes

Beschreibung	HTML Entity	Unicode
😂 Tränenlachendes Gesicht	`&#128514;`	`\uD83D\uDE02`
❤️ Rotes Herz	`&#10084;&#65039;`	`\u2764`
😍 Herzaugen-Gesicht	`&#128525;`	`\uD83D\uDE0D`
👏 Klatschende Hände	`&#128079;`	`\uD83D\uDE4C`
🤔 Denkendes Gesicht	`&#129300;`	`\uD83E\uDD14`

Nützliche Quellen

- **Unicode Emoji Charts:** https://unicode.org/emoji/charts/full-emoji-list.html
- **Emojipedia:** https://emojipedia.org/

HTML APIs

Ein HTML API (Application Programming Interface) ist eine Schnittstelle, über die verschiedene Softwarekomponenten, Anwendungen oder Systeme miteinander kommunizieren können. In der Webentwicklung sind HTML APIs Werkzeuge, die es uns ermöglichen, erweiterte Funktionen in unsere Webseiten zu integrieren, ohne alles selbst von Grund auf programmieren zu müssen.

Beispiele für API-Einsatz auf Webseiten:

- Einbindung eines individuellen Videoplayers
- Anzeige interaktiver Karten
- Speicherung von Benutzereinstellungen
- Anzeige von Benachrichtigungen

Geolocation API

Mit der Geolocation API kannst du die geografische Position eines Geräts ermitteln. Das ist besonders praktisch, wenn du standortbasierte Dienste anbieten willst.

Beispielszenarien:

- Wettervorhersage für den aktuellen Standort
- Anzeige von Geschäften oder Restaurants in der Nähe
- Navigation oder Kartendienste

Beispielcode

```
<!DOCTYPE html>
<html>
```

```
<head>
  <title>Geolocation Example</title>
</head>
<body>
  <h1>Your Location</h1>
  <p>Click the button to get your current location.</p>

  <button id="getLocation">Get Location</button>

  <p id="result"></p>

  <script>
    const getLocationButton = document.getElementById('getLocation');
    const resultElement = document.getElementById('result');

    getLocationButton.addEventListener('click', () => {
      if ("geolocation" in navigator) {
        navigator.geolocation.getCurrentPosition((position) => {
          const latitude = position.coords.latitude;
          const longitude = position.coords.longitude;
          resultElement.textContent = `Latitude: ${latitude}, Longitude: ${longitude}`;
        });
      } else {
        resultElement.textContent = "Geolocation is not supported by your browser.";
      }
    });
  </script>
</body>
</html>
```

Erklärung:

- **HTML:** Struktur mit Überschrift, Button und Ergebnisanzeige.
- **JavaScript:**
  - Prüft, ob der Browser Geolocation unterstützt.
  - Holt die aktuelle Position (Breiten- und Längengrad).
  - Zeigt die Koordinaten im Browser an.

Zusätzliche Beispiele:

1. **Standortbasierte Begrüßung:** Zeige eine Begrüßung wie "Willkommen in Berlin!" anhand des Standorts.
2. **Automatisches Setzen der Sprache:** Verwende den Standort, um die Spracheinstellung automatisch festzulegen (z. B. Deutsch für Deutschland).

Audio- und Video-APIs

Grundlagen

HTML bietet die Elemente `<audio>` und `<video>`, mit denen du Mediendateien einfach einbinden kannst. Diese Elemente besitzen eingebaute APIs, mit denen du Wiedergabe, Lautstärke oder Pausen programmatisch steuern kannst.

Einfaches Beispiel:

```
<audio controls>
  <source src="your-audio-file.mp3" type="audio/mp3">
  Your browser does not support the audio element.
</audio>
```

Hinweis: Das Attribut `controls` zeigt automatisch Play-, Pause- und Lautstärkeregler an.

Zusätzliche Beispiele:

1. **Eigenes Musik-Interface:** Mit JavaScript kannst du Buttons für "Play", "Pause" oder "Lauter" selbst erstellen.
2. **Podcast Player:** Lade eine Liste von Episoden dynamisch und spiele sie ab.

Web Storage API

Die Web Storage API erlaubt es Webseiten, Daten lokal im Browser des Benutzers zu speichern. Es gibt zwei Speicherarten:

- **Session Storage:** Speichert Daten bis zum Schließen des Browser-Tabs.
- **Local Storage:** Speichert Daten dauerhaft im Browser, bis sie explizit gelöscht werden.

Beispielcode:

```
// Speichert das Theme in localStorage
localStorage.setItem('theme', 'dark');

// Holt das Theme bei erneutem Laden
const userTheme = localStorage.getItem('theme');
document.body.classList.add(userTheme);
```

Erläuterung:

- Speichert den Wert "dark" unter dem Schlüssel "theme".
- Beim erneuten Laden liest die Seite den Wert aus und aktiviert das dunkle Design.

Zusätzliche Beispiele:

1. **Benutzernamen speichern:** Begrüße den Nutzer mit seinem gespeicherten Namen beim nächsten Besuch.
2. **Sprachwahl merken:** Die ausgewählte Sprache bleibt auch nach Schließen des Tabs erhalten.

Notifications API

Mit der Notifications API kann deine Webseite Benachrichtigungen anzeigen, auch wenn der Nutzer nicht aktiv auf der Seite ist. Dies eignet sich hervorragend für Erinnerungen, Hinweise oder Updates.

Beispielanwendungen:

- Eine To-Do-App erinnert an fällige Aufgaben
- Nachrichtenportale schicken Eilmeldungen

Achtung: Der Nutzer muss der Anzeige von Benachrichtigungen explizit zustimmen.

Zusätzliche Beispiele:

1. **Einkaufsplattform:** Erinnerung an einen vergessenen Warenkorb.
2. **Kalender-App:** Hinweis auf ein bald beginnendes Meeting.

Canvas API

Diese API erlaubt es dir, Grafiken mit JavaScript direkt im Browser zu zeichnen. Damit kannst du Animationen, Diagramme oder sogar Spiele erstellen.

Beispielhafte Anwendungen:

- Interaktive Diagramme (z. B. für Statistiken)
- Visuelle Effekte für Buttons
- Mini-Spiele auf Webseiten

Zusätzliche Beispiele:

1. **Digitale Unterschrift:** Nutzer unterschreiben ein Formular mit der Maus.

2. **Zeichenbrett:** Besucher können online malen oder skizzieren.

HTML-Grafiken

Grafiken spielen eine zentrale Rolle bei der Gestaltung moderner Webseiten. Sie sorgen nicht nur für ein attraktives Erscheinungsbild, sondern erhöhen auch die Benutzerfreundlichkeit und die Interaktivität. In HTML stehen dir zwei zentrale Technologien zur Verfügung, um Grafiken zu erzeugen und zu integrieren:

Zwei Ansätze in HTML:

Technologie	Beschreibung	Verwendung
`<canvas>`	Eine Zeichenfläche, auf der mit JavaScript pixelbasiert gezeichnet wird	Spiele, Animationen, Visualisierungen
`<svg>`	Eine auf Vektoren basierende Grafikbeschreibung in XML-Form	Logos, Icons, skalierbare Illustrationen

---

1. Die Canvas

Das `<canvas>`-Element erzeugt eine leere Zeichenfläche in deiner HTML-Seite. Die eigentliche Grafik erstellst du mit JavaScript – über die sogenannte **Canvas API**.

Eigenschaften von Canvas:

- Pixelbasiertes Zeichnen (Rastergrafiken)
- Sehr präzise Kontrolle über einzelne Bildpunkte
- Ideal für dynamische Inhalte
- Erfordert JavaScript für jegliche Funktionalität

Beispiel: Ein einfaches Rechteck zeichnen

```
<canvas id="myCanvas" width="300" height="200"></canvas>

<script>
  const canvas = document.getElementById("myCanvas");
  const ctx = canvas.getContext("2d");

  ctx.fillStyle = "black";
  ctx.fillRect(10, 10, 200, 100);
</script>
```

1. `<canvas>` erzeugt ein 300x200 Pixel großes Zeichenfeld.
2. Mit JavaScript wird über `getContext("2d")` ein 2D-Zeichenkontext erstellt.
3. Die Funktion `fillRect()` zeichnet ein schwarzes Rechteck auf das Canvas.

---

2. SVG – Scalable Vector Graphics

SVG steht für **Scalable Vector Graphics** und beschreibt Grafiken mit Hilfe von **XML**-Elementen. Diese Grafiken sind **auflösungsunabhängig**, was bedeutet, dass sie auf jedem Bildschirm scharf dargestellt werden – ideal für Icons, Logos und Illustrationen.

Eigenschaften von SVG:

- Vektorbasierte Grafiken (kein Qualitätsverlust beim Skalieren)
- Kann direkt in HTML geschrieben oder extern eingebunden werden
- Unterstützt CSS-Styling und Animationen
- Barrierefreier und durchsuchbarer Code

Beispiel: Ein schwarzer Kreis

```
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" fill="black" />
</svg>
```

Erklärung:

- `<svg>` erzeugt eine Grafikfläche von 100x100 Einheiten.
- `<circle>` zeichnet einen Kreis:
  - Mittelpunkt bei (50,50)
  - Radius: 40
  - Füllfarbe: schwarz

Ein rotes Rechteck

```
<svg width="200" height="100">
  <rect width="150" height="80" x="25" y="10" fill="red" />
</svg>
```

> Zeichnet ein rotes Rechteck mit 150x80 Pixeln, das bei (25,10) beginnt.

Text mit SVG

```
<svg width="300" height="100">
  <text x="10" y="50" font-size="30" fill="blue">Hallo SVG</text>
</svg>
```

> Zeigt den Text „Hallo SVG“ in blauer Farbe und 30px Schriftgröße an.

- **FontAwesome** – große Sammlung an Web-Icons
- **SVGRepo** – Tausende kostenlose SVG-Grafiken für private und kommerzielle Nutzung

---

3. Vergleich: Canvas vs. SVG

Merkmal	Canvas	SVG
Grafiktyp	Raster (Pixel)	Vektor
Skalierbarkeit	Nicht verlustfrei	Verluste frei skalierbar
Interaktivität	Hoch (mit JavaScript)	Möglich (mit CSS und JS)
Einsatzgebiet	Spiele, Animationen, Datenvisualisierung	Icons, Logos, Illustrationen
Performance bei vielen Objekten	Besser geeignet	Kann langsam werden
Zugriff	Nur über JavaScript	Direkter HTML/XML-Code zugänglich

---