# JavaScript im Browser einbinden

## JavaScript mit einer HTML-Seite verbinden

Damit JavaScript im Browser ausgeführt wird, muss dein HTML-Dokument wissen, *wo* sich der JavaScript-Code befindet. Dafür verwendest du das HTML-Element `<script>`.

Es gibt zwei gebräuchliche Möglichkeiten:

1. JavaScript direkt in die HTML-Datei schreiben
2. JavaScript in einer eigenen Datei speichern und diese einbinden

Für kleine Tests ist die erste Variante praktisch. In echten Projekten solltest du JavaScript jedoch fast immer in einer **separaten Datei** schreiben. Das hält HTML, CSS und JavaScript übersichtlich getrennt.

---

## JavaScript direkt in HTML schreiben

Mit einem `<script>`-Element kannst du JavaScript unmittelbar in eine HTML-Datei einfügen:

```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Mein erstes JavaScript</title>
  </head>
```

```
  <body>
    <h1>Willkommen!</h1>

    <script>
      console.log("JavaScript funktioniert!");
    </script>
  </body>
</html>
```

Der Bereich zwischen `<script>` und `</script>` enthält JavaScript-Code.

Wenn du diese Datei im Browser öffnest, siehst du die Ausgabe nicht direkt auf der Webseite. Öffne stattdessen die **Entwicklerwerkzeuge** und wechsle zur Konsole. Dort erscheint:

```
JavaScript funktioniert!
```

> **Merke:** `console.log()` gibt Informationen in der Browser-Konsole aus. Es ist besonders hilfreich, um Code zu testen und Fehler zu suchen.

---

# JavaScript in einer eigenen Datei speichern

Bei mehr als wenigen Zeilen Code wird eine separate Datei deutlich übersichtlicher.

Lege zunächst einen Ordner für dein Projekt an. Darin speicherst du zum Beispiel diese beiden Dateien:

```
mein-projekt/
├── index.html
└── script.js
```

Die Datei `index.html` enthält die Struktur der Webseite. In `script.js` steht ausschließlich dein JavaScript-Code.

## Die HTML-Datei

```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>JavaScript einbinden</title>
  </head>
  <body>
    <h1>Meine erste Webseite mit JavaScript</h1>

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

Das Attribut `src` steht für *source*, also Quelle. Es teilt dem Browser mit, welche Datei geladen werden soll:

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

# Die JavaScript-Datei

Schreibe in `script.js`:

```
console.log("Die Datei script.js wurde geladen.");
```

Öffne anschließend `index.html` im Browser und kontrolliere die Konsole. Wenn alles stimmt, erscheint dort:

```
Die Datei script.js wurde geladen.
```

---

# Warum separate Dateien sinnvoll sind

Die Aufteilung in HTML- und JavaScript-Dateien hat mehrere Vorteile:

- **Bessere Übersicht:** HTML beschreibt den Inhalt der Seite, JavaScript ihr Verhalten.
- **Leichtere Pflege:** Änderungen am JavaScript betreffen nicht direkt die HTML-Struktur.

- **Wiederverwendung:** Dieselbe JavaScript-Datei kann auf mehreren HTML-Seiten eingebunden werden.
- **Sauberer Code:** Größere Projekte bleiben einfacher verständlich.

Eine übliche Grundstruktur sieht so aus:

```
mein-projekt/
├── index.html
├── css/
│   └── style.css
└── js/
    └── script.js
```

Dann muss der Pfad in der HTML-Datei angepasst werden:

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

Der Schrägstrich bedeutet: Die Datei `script.js` liegt im Unterordner `js`.

---

# Wo gehört das `<script>`-Element hin?

Ein `<script>`-Element kann grundsätzlich im `<head>` oder im `<body>` stehen. Die Position ist wichtig, weil der Browser HTML und JavaScript nicht unabhängig voneinander verarbeitet.

# Möglichkeit 1: Am Ende des `<body>`

Für den Einstieg ist dies eine sehr einfache und sichere Lösung:

```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Beispiel</title>
  </head>
  <body>
    <h1 id="ueberschrift">Hallo!</h1>
```

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

Das Skript steht **nach** dem sichtbaren HTML-Inhalt. Dadurch wurden Überschrift, Absätze und Buttons bereits geladen, bevor JavaScript darauf zugreift.

Das ist besonders wichtig, wenn du später Elemente der Seite verändern möchtest.

---

# Möglichkeit 2: Im `<head>` mit `defer`

Eine moderne und sehr verbreitete Variante ist das Attribut `defer`:

```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Beispiel</title>
    <script src="script.js" defer></script>
  </head>
  <body>
    <h1 id="ueberschrift">Hallo!</h1>
  </body>
</html>
```

`defer` bedeutet sinngemäß: „Lade die Datei schon jetzt, führe sie aber erst aus, wenn das HTML vollständig verarbeitet wurde.“

Das hat zwei Vorteile:

- Das JavaScript kann frühzeitig geladen werden.
- Der Code wird erst ausgeführt, wenn die HTML-Elemente verfügbar sind.

> **Empfehlung:** Verwende für externe JavaScript-Dateien häufig `<script src="script.js" defer></script>` im `<head>`. Alternativ kannst du das Skript ohne `defer` direkt vor `</body>` einfügen.

---

# Ein erstes sichtbares Ergebnis

Bisher hast du JavaScript nur über die Konsole überprüft. JavaScript kann aber auch den Inhalt einer Webseite verändern.

Erstelle diese `index.html`:

```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Meine Begrüßung</title>
    <script src="script.js" defer></script>
  </head>
  <body>
    <h1 id="begrüßung">Willkommen!</h1>
  </body>
</html>
```

Schreibe dann in `script.js`:

```
const überschrift = document.querySelector("#begrüßung");


überschrift.textContent = "Hallo, JavaScript!";
```

Nach dem Neuladen der Webseite verändert sich der Text der Überschrift zu:

```
Hallo, JavaScript!
```

Noch musst du nicht jedes Detail dieses Codes verstehen. Wichtig ist zunächst:

- `document` steht für die aktuelle HTML-Seite.
- `querySelector()` sucht ein HTML-Element.
- `#begrüßung` meint das Element mit `id="begrüßung"`.
- `textContent` verändert den Text eines Elements.

Mit dem DOM und `querySelector()` beschäftigst du dich später ausführlich.

---

# Häufige Fehler beim Einbinden

## Der Dateiname stimmt nicht

Wenn deine Datei `script.js` heißt, muss der Eintrag exakt so aussehen:

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

Diese Varianten funktionieren nicht, wenn es keine passende Datei gibt:

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

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

Achte besonders darauf, dass dein Editor die Datei wirklich als `.js` speichert und nicht unbemerkt `.txt` anhängt.

---

## Der Pfad zur Datei ist falsch

Liegt die JavaScript-Datei in einem Unterordner, muss dieser Teil des Pfads angegeben werden:

```
mein-projekt/
├── index.html
└── js/
    └── script.js
```

Richtig wäre dann:

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

Ein falscher Pfad führt dazu, dass der Browser die JavaScript-Datei nicht findet.

---

## Die Datei wurde nicht gespeichert

Wenn Änderungen keine Wirkung zeigen:

1. Speichere `index.html`.

2. Speichere `script.js`.
3. Lade die Webseite im Browser neu.
4. Prüfe die Browser-Konsole auf Fehlermeldungen.

Eine typische Fehlermeldung könnte so aussehen:

```
Uncaught SyntaxError: Unexpected token
```

Sie weist darauf hin, dass JavaScript an einer Stelle nicht korrekt geschrieben wurde. In den nächsten Lektionen lernst du, solche Meldungen besser zu lesen.

---

# Das Skript wird zu früh ausgeführt

Dieses Beispiel kann Probleme verursachen:

```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Zu früh</title>
    <script src="script.js"></script>
  </head>
  <body>
    <h1 id="begrüßung">Willkommen!</h1>
  </body>
</html>
```

Ohne `defer` versucht der Browser, JavaScript bereits im `<head>` auszuführen. Das Element mit der ID `begrüßung` wurde zu diesem Zeitpunkt möglicherweise noch nicht geladen.

Besser:

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

Oder du setzt das Skript ans Ende des `<body>`.

---

# Inline-Skript und externe Datei im Vergleich

| Variante | Beispiel | Geeignet für |
| --- | --- | --- |
| JavaScript direkt im HTML | `<script>console.log("Test");</script>` | Sehr kurze Experimente |
| Externe JavaScript-Datei | `<script src="script.js" defer></script>` | Praktisch alle eigenen Projekte |

Für dein weiteres Lernen empfiehlt sich die zweite Variante: Lege eine Datei namens `script.js` an und binde sie mit `defer` im `<head>` ein.

---

# Mini-Übung

Erstelle einen Projektordner mit diesen Dateien:

```
übung-javascript/
├── index.html
└── script.js
```

# Aufgabe

1. Erstelle eine HTML-Seite mit einer Überschrift.
2. Binde `script.js` mit `defer` ein.
3. Gib in `script.js` deinen Namen mit `console.log()` aus.
4. Öffne die HTML-Datei im Browser und überprüfe die Ausgabe in der Konsole.

# Mögliche Lösung

**index.html**

```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="utf-8">
```

```
    <title>Meine JavaScript-Übung</title>
    <script src="script.js" defer></script>
  </head>
  <body>
    <h1>Ich lerne JavaScript</h1>
  </body>
</html>
```

**script.js**

```
console.log("Mein Name ist Alex.");
```

Ersetze „Alex“ durch deinen eigenen Namen.

---

# Das Wichtigste auf einen Blick

- Mit `<script>` bindest du JavaScript in eine HTML-Seite ein.
- JavaScript kann direkt im HTML oder in einer eigenen `.js`-Datei stehen.
- Eine externe Datei bindest du mit `src` ein.
- Mit `defer` wird das Skript erst ausgeführt, nachdem das HTML geladen wurde.
- `console.log()` hilft dir, schnell zu prüfen, ob dein JavaScript funktioniert.
- Bei Problemen kontrollierst du Dateinamen, Pfade, gespeicherte Änderungen und die Browser-Konsole.

---

Revision #1
Created 2026-07-24 13:44:14 UTC by art10m
Updated 2026-07-24 13:44:14 UTC by art10m