Das ULTIMATIVE JavaScript-Grundlagen-Studienhandbuch

Kurslektionen

- Lektion 1
- Lektion 2
- Lektion 3
- Lektion 4
- Lektion 5
- Lektion 6
- Lektion 7
- Lektion 8
- Lektion 9
- Lektion 10
- Lektion 11
- Lektion 12
- Lektion 13
- Lektion 14

Begriffe & Definitionen

- \n
- `+` Operator
- Additionsoperator
- `alert()`
- API
- API-Parameter
- API-URL
- `append()`
- Argumente

- Arithmetische Operatoren
- Arrays (Felder)
- Zuweisungsoperator
- `async` Schlüsselwort
- `await` Schlüsselwort
- Block Scope (Blockgültigkeit)
- Boolean (Boolescher Wert)
- Boolescher Datentyp
- Bracket Notation (Klammernotation)
- Callback-Funktion
- Aufrufen (Calling)
- CamelCase (Kamelschreibweise)
- Change-Event (Änderungsereignis)
- `classList`-Eigenschaft
- `classList.add()`
- `classList.remove()`
- Vergleichsoperatoren
- Zusammengesetzte Zuweisungsoperatoren
- Verketten (Concatenating)
- Bedingte Anweisung (Conditional Statement)
- Konsole
- `console.log()`
- `const` Schlüsselwort
- `const` vs. `let` vs. `var`
- Kontext
- `createElement()`
- Datentyp

- `Date()`
- Debugging (Fehlerbehebung)
- Zerlegung (Decomposition)
- `defer`-Attribut
- Delimiter (Trennzeichen)
- `disabled`-Eigenschaft
- DOM (Document Object Model)
- DOM-Events
- DOM-Baum
- Dot Notation (Punktnotation)
- Elemente
- `else if` Schlüsselwort
- `else` Schlüsselwort
- Endpunkte
- Event Handler (Ereignisbehandler)
- Event Listener (Ereignisüberwacher)
- Ausdrücke (Expressions)
- Factory Functions (Fabrikfunktionen)
- `fetch()`
- Fließkommazahl (Floating Point Number)
- `for`-Schleife
- `for...of`-Schleife
- `for…in`-Schleife
- `forEach()`
- Funktion
- Funktionskörper
- Funktionsausdruck

- Function Scope (Funktionsgültigkeit)
- `getHours()`
- Global Scope (Globale Gültigkeit)
- `if` Schlüsselwort
- Index
- `innerHTML`-Eigenschaft
- `innerText`-Eigenschaft
- Input-Event (Eingabeereignis)
- Integer (Ganzzahl)
- Iterieren (Iterating)
- JSON-Dateien
- `json()`
- Keydown-Event (Tastendruckereignis)
- Keys (Schlüssel)
- Key-Value Pair (Schlüssel-Wert-Paar)
- `let` Schlüsselwort
- Loop (Schleife)
- Loop Body (Schleifenkörper)
- `match()`
- `matches()`
- `Math.floor()`
- `Math.max()`
- `Math.min()`
- `Math.random()`
- Methode (Lektion 4)
- Methode (Lektion 10)
- Modal Window (Modales Fenster)

- Modulus (`%`) Operator
- Mouse Events (Mausereignisse)
- Nested `if`-Anweisung (Verschachtelte `if`-Anweisung)
- Null
- `Number()`
- Object Literal (Objektliteral)
- Objekte
- Parameter
- Primitive Datentypen
- `prompt()`
- Eigenschaften (Properties)
- `querySelector()`
- `querySelectorAll()`
- Refactored Code (Refaktorierter Code)
- Regular Expression (Regulärer Ausdruck)
- Reserved Keywords (Reservierte Schlüsselwörter)
- REST APIs
- `return` Schlüsselwort
- Scope (Gültigkeit)
- Statement (Anweisung)
- String (Zeichenkette)
- `style`-Eigenschaft
- Template Literals (Template-Literale)
- `this` Schlüsselwort
- `toFixed()`
- `toLowerCase()`
- `toUpperCase()`

- `trim()`
- Type Conversion (Typumwandlung)
- Undefined
- `value`-Eigenschaft
- Values (Werte)
- Variable

Lektion 1 - Erste Schritte mit JavaScript

Variable - Ein Werkzeug, um auf Informationen zu verweisen. Der zugeordnete Wert kann variieren oder sich ändern. Variablen können mit dem Schlüsselwort `var` (Lektion 1) und den Schlüsselwörtern `let` und `const` (Lektion 9) deklariert werden.

Values - Informationen, die in einer Variablen gespeichert sind. Genauer gesagt ist ein Wert eine Folge von Bits, die gemäß einem Datentyp interpretiert wird (z. B. Zahl, Zeichenkette, Boolean).

Zuweisungsoperator - Ein Operator, der einer Variablen einen Wert zuweist. Der Zuweisungsoperator verwendet das Gleichheitszeichen (=). Siehe das JavaScript-Operatoren-Spickzettel (Lektion 3) für eine Liste von Zuweisungsoperatoren.

Statement - Eine einzelne Anweisung an das Programm. Oft erscheinen Semikolons am Ende einer Anweisung, um anzuzeigen, dass sie vollständig ist.

```
var anzahlCerealien = 16;
console.log("Wir lieben JS!");
```

String - Eine Reihe von Zeichen, wie Zahlen, Buchstaben und Symbole. Zeichenketten werden in Anführungszeichen gesetzt, um die Zeichen zu gruppieren und in einer Sequenz zu halten.

```
var urlaubsort = "Strand";
var telefonnummer = "555-555-1234";
```

CamelCase - Die Standardnamenskonvention für Variablen in JavaScript. Die ersten Wörter sind alle Kleinbuchstaben, während jedes folgende Wort mit einem Großbuchstaben beginnt. Beispiele: `bankEinzahlung`, `benutzerEingabeDatum` und `altersLimit18`.

Lektion 2 - Datentypen & Arithmetische Operatoren

Template Literals - Ausgabe von Zeichenketten mit Platzhaltern und Backticks (`). Im Vergleich zur Ausgabe von Zeichenketten mit einfachen oder doppelten Anführungszeichen und dem Plus-Operator erleichtern Template Literals die Ausgabe von mehrzeiligen Zeichenketten und das Kombinieren von Variablen mit Zeichenketten. Außerdem können Sie Ausdrücke innerhalb der Zeichenkette berechnen.

```
var schmuck = "Uhr";
var ereignis = "Abendessen";
console.log(`Sie trug eine ${schmuck} zum ${ereignis}.`);
// Sie trug eine Uhr zum Abendessen.

var pizzaSorte = "vegetarisch";
var stueckeGegessen = 4;
console.log(`Die ${pizzaSorte} Pizza hat ${8 - stueckeGegessen} Stücke übrig.`);
// Die vegetarische Pizza hat 4 Stücke übrig.
```

Expressions - Code, der zu einem Wert führt. Zum Beispiel können Ausdrücke zu numerischen, Zeichenketten- und logischen Werten führen (Lektion 3).

```
console.log(8 - 5);  // 3
console.log("Ich liebe" + " Programmieren.");  // Ich liebe Programmieren.
console.log(5 < 8);  // true
```

Integer - Eine ganze Zahl, wie 5100 oder -258. Ganzzahlen können positiv oder negativ sein.

Floating Point Number - Eine Zahl mit einem Dezimalpunkt, wie 2134.3625 oder -562.12. Fließkommazahlen können positiv oder negativ sein.

Addition Operator - Ein Operator zum Addieren zweier Zahlen. Der Additionsoperator verwendet das Pluszeichen (+).

```
var aepfelBananen = 5 + 8;
console.log(aepfelBananen);  // 13

var etage1 = 10;
var etage2 = 15;
console.log(`Es gibt ${etage1 + etage2} Tische im Restaurant.`);  // Es gibt 25 Tische
```

```
im Restaurant.
```

Arithmetic Operators - Symbole für mathematische Operationen, wie die Additions- (+), Subtraktions- (-), Multiplikations- (*) und Divisionsoperatoren (/). Siehe das JavaScript-Operatoren-Spickzettel (Lektion 3) für eine vollständige Liste arithmetischer Operatoren.

Data Type - Der Werttyp, auf den eine Variable verweist. Beispiele sind Zahlen, Zeichenketten, Boolesche Werte (Lektion 3), Undefined, Null, Arrays (Lektion 8) und Objekte (Lektion 10).

Primitive Data Types - Werte mit nur einem einzelnen Wert, wie Zahlen, Zeichenketten, Boolesche Werte (Lektion 3), Undefined und Null.

Undefined - Eine Variable, der kein Wert zugewiesen wurde.

```
var glueck;
console.log(glueck);  // undefined
```

Null - Ein Datentyp, der einen absichtlich leeren oder nicht vorhandenen Wert darstellt.

```
var ideen = null;
console.log(ideen);  // null
```

Type Conversion - Ändern eines Werts in einen anderen Wert, um einen Operator abzuschließen. Typumwandlung ist nützlich, um Zeichenketten in Zahlen umzuwandeln, damit Sie sie berechnen können.

Number() - Konvertiert eine Zeichenkette in eine Zahl. `Number()` ist nützlich, wenn Sie Eingaben von einem Benutzer erfassen und diese dann in eine Zahl ändern, damit Sie einen Wert berechnen können.

```
var fernsehSendungen = Number("23");
var filme = 12;
console.log(fernsehSendungen + filme);  // 35
```

prompt() - Zeigt ein Feld an, um Informationen vom Benutzer zu sammeln. Benutzer sehen ein Popup-Dialogfeld auf ihrem Bildschirm, das zur Eingabe auffordert.

```
var lieblingsMusikrichtung = prompt("Was ist deine Lieblingsmusikrichtung?");
console.log(lieblingsMusikrichtung);

var altesGuthaben = 1500;
```

```
var neuesGuthaben = Number(prompt("Wie viel Geld wurde gesammelt?"));
console.log(
`Das Spendenaktion-Gesamt beträgt jetzt $${altesGuthaben + neuesGuthaben}!.`
);
```

toFixed() - Konvertiert einen Zahlendatentyp in eine Zeichenkette und rundet dann auf eine angegebene Anzahl von Dezimalstellen. Fügen Sie eine Zahl in `toFixed()` ein, um die Anzahl der Dezimalstellen anzugeben, auf die gerundet werden soll.

```
var steuerBetrag = 7.23335651;
console.log(steuerBetrag.toFixed(2));  // 7.23

var tempFahrenheit = 98.6785;
console.log(`Ihre Temperatur beträgt ${tempFahrenheit.toFixed(1)}.`);  // Ihre
Temperatur beträgt 98.7.

var personen = 27;
var auszahlung = 800.29;
console.log(`Sie haben $${(auszahlung / personen).toFixed(2)} gewonnen.`);  // Sie haben
$29.64 gewonnen.
```

Lektion 3 - Vergleiche & Bedingungen

Conditional Statement - Code, der nur ausgeführt wird, wenn eine Bedingung wahr ist.

Boolean - Repräsentiert nur zwei Werte: wahr oder falsch.

Boolean Data Type - Ein primitiver Datentyp mit True- oder False-Werten.

```
var lichterAn = true;
var ventilatorAn = false;
console.log(lichterAn);  // true
```

Comparison Operators - Operatoren, die Symbole verwenden, um zwei oder mehr Werte zu vergleichen, wie >, < und ===. Siehe das JavaScript-Operatoren-Spickzettel (Lektion 3) für eine vollständige Liste von Vergleichsoperatoren.

if Keyword - Schlüsselwort zur Verwendung in einer Anweisung, um eine Bedingung zu testen. Wenn die Bedingung zu "true" auswertet, führt das Programm den Code innerhalb des If-Blocks aus. Sie fügen nach der Bedingung kein Semikolon ein.

```
var heissesWetter = true;
if (heissesWetter === true) {
console.log("Tragen Sie heute Shorts und ein Tanktop!");
}
// Tragen Sie heute Shorts und ein Tanktop!
```

else Keyword - Schlüsselwort zur Verwendung in einer Anweisung, um eine andere Aktion auszuführen, wenn die vorherige Bedingung zu "false" auswertet.

```
var heissesWetter = false;
if (heissesWetter === true) {
console.log("Tragen Sie heute Shorts und ein Tanktop!");
} else {
console.log("Schnappen Sie sich einen Pullover, es könnte kühl sein.");
}
// Schnappen Sie sich einen Pullover, es könnte kühl sein.
```

else if Keyword - Schlüsselwort zur Verwendung in einer Anweisung, um eine neue Bedingung zu testen und dann eine Aktion auszuführen, wenn die vorherige Bedingung zu "false" auswertet. Sobald eine Bedingung zu "true" auswertet, wird der Codeblock, dem die Bedingung zugeordnet ist, ausgeführt und der bedingte Block wird verlassen, unabhängig davon, ob es nachfolgende Bedingungen gibt, die ebenfalls zu "true" auswerten würden.

```
var heissesWetter = false;
var verschneitesWetter = true;
var windigesWetter = true;
if (heissesWetter === true) {
console.log("Tragen Sie heute Shorts und ein Tanktop!");
} else if (verschneitesWetter === true) {
console.log("Ziehen Sie eine schwere Jacke und Stiefel an!");
} else if (windigesWetter === true) {
console.log("Zeit, den Windbreaker anzuziehen.");
} else {
console.log("Schnappen Sie sich einen Pullover, es könnte kühl sein.");
}
```

```
// Ziehen Sie eine schwere Jacke und Stiefel an!
```

alert() - Zeigt eine Popup-Nachricht an, die Benutzer sehen können. Die Eingabeaufforderung enthält eine OK-Schaltfläche, auf die Benutzer klicken können, um das Popup zu schließen.

```
alert("Hallo, willkommen auf meiner Seite!");
```

Date() - Eine Methode, um das aktuelle Datum abzurufen.

```
var wochentag = new Date().toLocaleString("de-DE", { weekday: "long" });
```

getHours() - Eine Methode, um die aktuelle Uhrzeit abzurufen. Die Uhrzeit spiegelt die 24-Stunden-Uhr wider, AKA militärische Uhrzeit.

```
var zeit = new Date().getHours();
```

Lektion 4 - JS, HTML & CSS

defer Attribute - Weist den Browser an, das Skript nach dem Laden der Seite zu laden. Das Attribut erzeugt eine schnellere Ladeerfahrung für den Benutzer, da zuerst das gesamte HTML gerendert wird, auch wenn das JavaScript noch nicht ausgeführt wurde. Es stellt auch sicher, dass die HTML-Elemente geladen werden, damit das JavaScript sie ändern kann. Sie fügen das `<script>`-Tag und die Defer-Attribute im Head-Bereich der HTML-Seite hinzu.

```
<!DOCTYPE html>
<html>
<head>
  <script src="js/script.js" defer></script>
</head>
```

DOM - Kurz für Document Object Model, das DOM repräsentiert die Struktur und den Inhalt einer Webseite. Das Dokument ist die Webseite. Die Objekte umfassen HTML-Elemente, Text und Attribute.

DOM Tree - Eine grafische Darstellung des DOM, die Beziehungen zwischen Objekten zeigt. Der DOM-Baum ist nützlich, um zu bestimmen, wie auf verschiedene Objekte im Dokument zugegriffen werden kann.

Methods - JavaScript-Aktionen, die an Objekten ausgeführt werden. Beispiele für Methoden sind `console.log()`, `prompt()`, `alert()`, und `querySelector()`. Methoden sind auch eine Art von Objekteigenschaft (Lektion 10).

querySelector() - Eine Methode, um auf das erste Element zuzugreifen, das einem angegebenen Selektor entspricht. Um mehrere Elemente auszuwählen, benötigen Sie ein Array (Lektion 8) mit `querySelectorAll()` (Lektion 9).

```
var verfuegbar = document.querySelector("h3");
var hauptTitel = document.querySelector(".main-title");
var erstesElement = document.querySelector("ul li");
var einleitung = document.querySelector(".intro p");
console.log(verfuegbar, hauptTitel, erstesElement, einleitung);

var erstesBild = document.querySelector("img");
erstesBild.src = "img/lulu.jpeg";
erstesBild.alt = "Lulu Strauß";
console.log(erstesBild);
```

style Property - Eine Eigenschaft, mit der Sie den Stil eines Elements ändern können. Wenn der Eigenschaftsname aus zwei Wörtern besteht, wie z. B. `background-color`, ändern Sie ihn mit CamelCase (`backgroundColor`) in ein Wort.

```
var einleitung = document.querySelector(".intro p");
einleitung.style.color = "purple";
einleitung.style.fontSize = "3em";
einleitung.style.fontStyle = "italic";
console.log(einleitung);
```

innerText Property - Eine Eigenschaft, die auf den Text innerhalb eines Elements zugreift. Diese Eigenschaft ist nützlich, wenn Sie den Text innerhalb eines Elements ändern oder abrufen möchten.

```
var ersteBeschriftung = document.querySelector("figcaption");
ersteBeschriftung.innerText = "Die Lulu.";
console.log(ersteBeschriftung);
```

innerHTML Property - Eine Eigenschaft, die das HTML eines Elements auf der Seite ändert. Diese Eigenschaft ist nützlich, um Elemente auf einer Seite zu aktualisieren oder hinzuzufügen.

```
ersteBeschriftung.innerHTML = "Die <strong>Lulu</strong>";
console.log(ersteBeschriftung);

var einleitung = document.querySelector(".intro p");
einleitung.innerHTML = 'Verfügbar <span class="increase__size">heute</span>';
console.log(einleitung);
```

Debugging - Identifizieren und Entfernen von Fehlern in Ihrem Code.

Lektion 5 - Events & Event Listeners

DOM Events - Aktionen, die im Dokument (Webseite) stattfinden. Ereignisse können vom Browser oder vom Benutzer ausgelöst werden. In dieser Klasse verwenden Sie Maus-, Änderungs-, Keydown- und Eingabeereignisse. Siehe Mozillas Ereignisreferenzseite für eine vollständige Liste von Ereignissen.

Mouse Events - Ein Ereignis, das stattfindet, wenn ein Zeigegerät, wie eine Maus, Joysticks, Tastatur oder ein adaptiver Schalter, mit dem Dokument interagiert. Häufige Mausereignisse sind "click", "mouseover" und "select".

Event Listener - Eine Methode, die auf Ereignisse "hört", die stattfinden, und dann Maßnahmen ergreift. Verwenden Sie die Methode `addEventListener()`, um auf Ereignisse im DOM zu hören.

```
var titel = document.querySelector("h1");
titel.addEventListener("mouseover");
```

Event Handler - Eine Funktion, die Code ausführt, wenn ein Ereignis auftritt.

```
var titel = document.querySelector("h1");
titel.addEventListener("mouseover", function () {
  titel.innerText = "Lasst uns feiern!";
  titel.style.color = "maroon";
});
```

Function - Ein Codeblock, der aufgerufen oder aufgerufen werden kann, um so oft wie nötig ausgeführt zu werden, ohne Code zu wiederholen. Funktionen sind wichtig, um optimiertes JavaScript zu schreiben. Lektion 6 enthält einen vollständigen Einblick in Funktionen.

Function Body - Der Teil der Funktion, der die Anweisungen enthält, die angeben, was die Funktion tut. Geschweifte Klammern umschließen den Funktionskörper.

classList Property - Eine Eigenschaft zum Hinzufügen, Entfernen oder Umschalten von CSS-Klassen auf einem Element. Mit dieser Eigenschaft können Sie mehrere Stile gleichzeitig anwenden (oder entfernen). Sie können die `classList`-Eigenschaft mit den Methoden `add()` und `remove()` verwenden: `classList.add()` und `classList.remove()`.

classlist.add() - Eine Methode zum Hinzufügen einer neuen Klasse.

```
var darkModeButton = document.querySelector(".dark-mode");
var body = document.querySelector("body");
darkModeButton.addEventListener("click", function () {
  body.classList.add("dark-palette");
});
```

classlist.remove() - Eine Methode zum Entfernen einer neuen Klasse.

```
var lightModeButton = document.querySelector(".light-mode");
lightModeButton.addEventListener("click", function () {
  body.classList.remove("dark-palette");
});
```