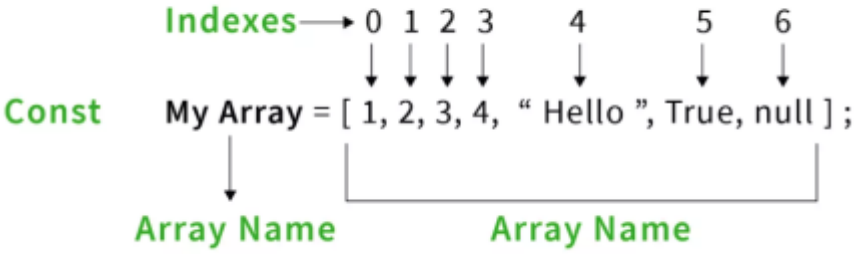
# LU03a - Arrays

## Einleitung

In der Programmierung stösst man schnell auf Situationen, in denen man nicht nur einen einzelnen Wert, sondern eine **Sammlung von Werten speichern** möchte – zum Beispiel eine Liste mit Namen, Zahlen oder Objekten.

In JavaScript gibt es dafür den Datentyp *Array*. Ein *Array* ist eine geordnete Liste, in der jedes Element über einen Index angesprochen werden kann (beginnend bei 0).




Arrays sind unglaublich vielseitig: Man kann Elemente hinzufügen, entfernen, sortieren oder mit speziellen Methoden wie *map* und *filter* weiterverarbeiten.

## Was ist ein Array?

Ein Array ist eine geordnete Sammlung von Werten. Diese Werte können Zahlen, Strings, Objekte oder sogar andere Arrays sein. Arrays sind besonders praktisch, wenn man mehrere Werte unter einem Namen speichern möchte.

**Beispiel:**

let zahlen = [1, 2, 3, 4, 5];

```
let namen = ["Anna", "Ben", "Clara"];
```

===== Zugriff auf Array-Elemente ===== + Jedes Element in einem Array hat einen Index, beginnend bei 0. +

let farben = [„rot“, „gruen“, „blau“];

```
console.log(farben[0]); // "rot"
console.log(farben[1]); // "gruen"
console.log(farben[2]); // "blau"
```

# Methoden

*Hinweis: **Wir müssen etwas vorgreifen mit dem Thema *Methoden*. In JavaScript sind Methoden nichts anderes als Funktionen, die an Objekte wie beispielsweise *ARRAYS* gebunden sind. * Eine Methode wird wie eine normale Funktion definiert, aber als Eigenschaft eines Objekts gespeichert. * D.h. man kann diese Objekte zum reagieren bringen wie beispielsweise: * Hey *myArray*, sag doch bitte wie lang Du bist –> myArray.length() Sie kann dann mit der Punkt-Notation aufgerufen werden.** Beispiel **const person = { name: „Anna“, greet: function() { return „Hallo, mein Name ist “ + this.name; } }; - - console.log(person.greet()); - + for (let i = 0; i < Employee.length; i++) { - Das Objekt mit dem Namen *Anna* hat eine Greet-Methode, kann also grüssen. die Ausgabe ist in diesem Falle „Hallo, mein Name ist Anna“. + console.log(„ID:“, Employee[i][0], „Name:“, Employee[i][1], „Stadt:“, Employee[i][2]); + } +** Ausgabe:** + ID: 100 Name: Ram Stadt: Agra + ID: 101 Name: Shyam Stadt: Aligarh + ID: 102 Name: Amit Stadt: Gwalior

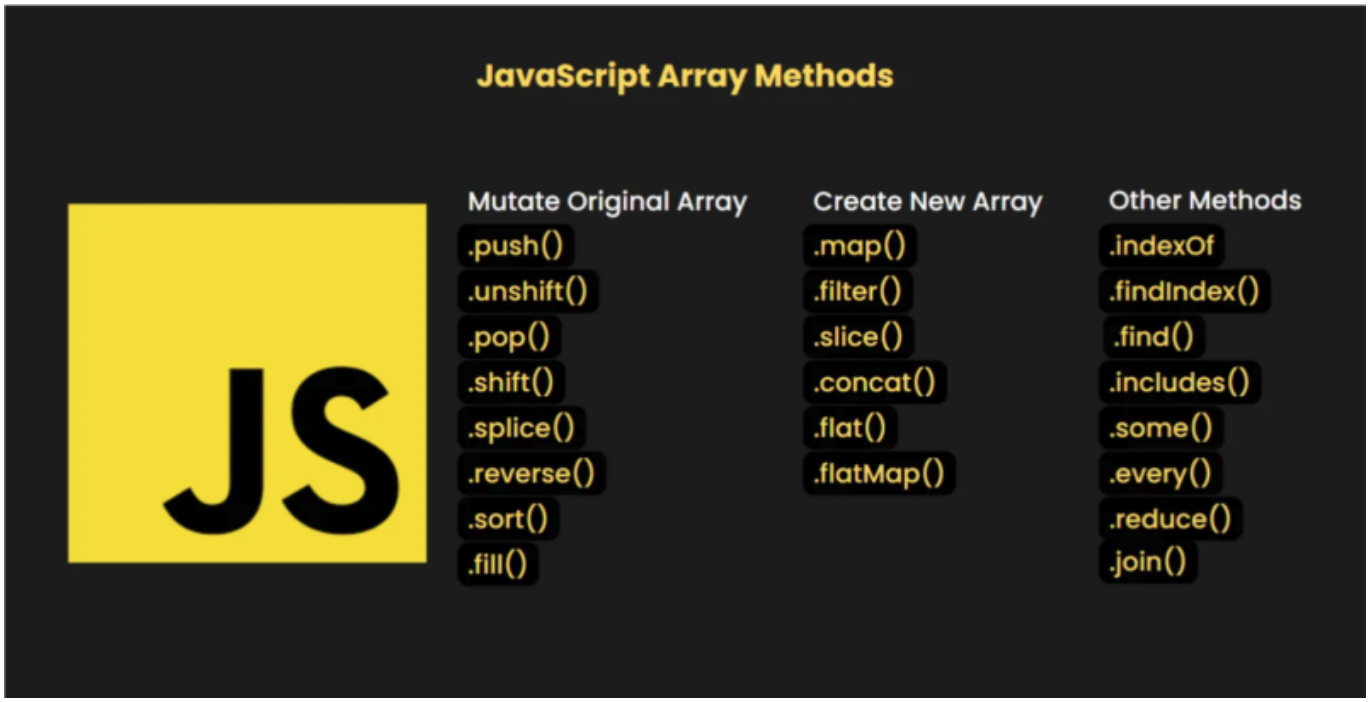
- +

# Arrays mit Attributen bauen

- ==== Array-Laenge ==== + Wenn wir, zusätzlich zu den Daten, auch Tabellenspalten speichern wollen, tun wir dies in den Array-Attributen.

- Mit der Eigenschaft (=Methode) *.length* erfährst du, wie viele Elemente im Array sind: + var Employee2 = [ + {id: 100, name:'Ram', vorname: 'Agra'}, + {id: 101, name: 'Shyam', vorname: 'Aligarh'}, + {id: 102, name: 'Amit', vorname: 'Gwalior'} + ];

- let zahlen = [10, 20, 30]; - console.log(zahlen.length); *3 - - ==== Elemente hinzufügen oder entfernen ==== - JavaScript stellt praktische Methoden bereit: - let tiere = [„Hund“, „Katze“]; + ===== Weitere Methoden ===== - + Grundsätzlich verhält sich ein mehrdimensionale Array genau so wie ein normales Array. Sprich alle Methoden, die für ein normales Array gelten, gelten auf für das mehrdimensionale Array. -* Hinzufügen + - tiere.push(„Maus“); *ans Ende + - tiere.unshift(„Vogel“);* an den Anfang + - console.log(tiere); *[„Vogel“, „Hund“, „Katze“, „Maus“]* + - + - Entfernen + - tiere.pop(); *entfernt letztes Element + - tiere.shift();* entfernt erstes Element + - console.log(tiere); *[„Hund“, „Katze“]* + - + - + - + - ==== Besondere Methoden ==== + - * map → erstellt ein neues Array basierend auf jedem Element. + - * filter → filtert Elemente nach Bedingung. + - * find → findet das erste passende Element. + - * includes → ist das Element vorhanden? + - + - ==== Übersicht über Array-Methoden ==== + -




*+ ^Methode ^Beschreibung | Anwendungsbeispiel | + ^push() | Fügt ein Element ans Ende eines Arrays hinzu. | + ^pop() | Entfernt das letzte Element.| + ^shift() | Entfernt das erste Element. | + ^unshift() | Fügt ein Element am Anfang hinzu. | + ^filter() | Gibt ein neues Array zurück mit allen Elementen, die eine Bedingung erfüllen. | + ^find() | Gibt das erste Element zurück, das eine Bedingung erfüllt. | + ^findIndex() | Gibt den Index des ersten passenden Elements zurück. | + ^splice() | Manipulation des Arrays an einer beliebigen Position. | + ^slice() | Erstellt eine Kopie eines Array-Ausschnitts, ohne das Original zu verändern.| - ===== Lernvideos ===== - | JavaScript-Tutorial - 8:31 Minuten | JavaScript-Tutorial - 20:30 Minuten | ===== Zusatzmaterial ===== ===== Zusatzmaterial ===== - * W3School- JavaScript Arrays + * W3School- JavaScript Mehrdimensionale Arrays - * SelfHTML - JS Arrays*