Wechselkurs - Devisenkurs

Diese Webseite, bestehend aus HTML, CSS und JavaScript, visualisiert den Wechselkurs für 31 Währungen. Weiterhin kannst du einen Betrag für die Umtauschwährung eingeben und die Zielwährung wählen. Du erhältst den Preis auf zwei Nachkommastellen gerundet.

Für mich ist es zwingend notwendig darauf hinzuweisen, dass ich für die Nutzung dieser Webseite keinerlei Haftung übernehme. Ohne Gewähr! Natürlich habe ich das Programm getestet, jedoch bei 31 Währungen kann ich nicht alle Kombinationen testen. Ich bitte um Verständnis.

Diese Applikation nutzt eine Exchange-Rate-API. Mit dem Hochfahren dieser Software kontaktiert das JavaScript eine API mit dem Namen „frankfurter“. Die "Frankfurter API" ist eine kostenlose, quelloffene API für Währungsdaten, die auf Referenzwechselkursen der Europäischen Zentralbank basiert. Sie ermöglicht es Entwicklern, Wechselkurse automatisch abzurufen, die täglich aktualisiert werden. Anfragen können direkt an https://api.frankfurter.dev gerichtet werden, was die Integration in Anwendungen zur Währungsumrechnung vereinfacht.

- ***Kostenlos und quelloffen:*** **Die API steht unter einer Open-Source-Lizenz zur Verfügung und erfordert keine Bezahlung.**
- ***Datenquelle:*** **Sie nutzt Daten der Europäischen Zentralbank, um verlässliche Wechselkursinformationen zu liefern.**
- ***Aktualisierung:*** **Die Daten werden an jedem Werktag gegen 16:00 Uhr MEZ aktualisiert.**
- ***URL:*** **Die API ist unter der Adresse https://api.frankfurter.dev erreichbar.**
- ***Hinweis:*** **Es ist wichtig, zwischen dieser API und der Frankfurter Wertpapierbörse zu unterscheiden, die eine eigene Organisation ist.**

Wer hat die API entwickelt? Hakan Ensari, https://github.com/hakanensari https://github.com/lineofflight.

Die Bedienung der Webseite ist so intuitiv, dass sie keiner weiteren Erklärung bedarf. Trotzdem möchte ich auf den nachfolgenden Seiten dieser Beschreibung einige Funktionen erläutern.

Die „frankfurter API“ wird mit dem Hochlauf der Software dreimal aufgerufen. Zuerst ein „json-Format“ mit den Namen und Kurzbezeichnungen (Schlüssel) der 31 Währungen. Danach erneut ein „json-Format“ mit dem Preis der Währungen in Bezug auf einem Euro. Zum Schluss eine Devisen-Anforderung für eine bestimmte Währung, die als Default-Wert im Eingabeeditor steht. Später kannst du mit dem Editor deine eigene Devisenberechnung vornehmen. Wenn man so eine Software baut, dann ist es vom Vorteil alle drei „json-Formate“ in einem String zu speichern und damit entwickeln. Ansonsten musst du zigmal den Server kontaktieren, weil irgendetwas nicht passt. Das führt zu unnötigen Belastungen für den Server, das wollen wir nicht. Das nachfolgende Bild zeigt diesen Vorgang im Detail.

```
45  const currenciesJsonString = `
46  {
47    "AUD": "Australian Dollar",
48    "BGN": "Bulgarian Lev",
49    "BRL": "Brazilian Real",
50    "CAD": "Canadian Dollar",
51    "CHF": "Swiss Franc",
52    "CNY": "Chinese Renminbi Yuan",
53    "CZK": "Czech Koruna",
54    "DKK": "Danish Krone",
55    "EUR": "Euro",
56    "GBP": "British Pound",
57    "HKD": "Hong Kong Dollar",
58    "HUF": "Hungarian Forint",
59    "IDR": "Indonesian Rupiah",
60    "ILS": "Israeli New Sheqel",
61    "INR": "Indian Rupee",
62    "ISK": "Icelandic Króna",
63    "JPY": "Japanese Yen",
64    "KRW": "South Korean Won",
65    "MXN": "Mexican Peso",
66    "MYR": "Malaysian Ringgit",
67    "NOK": "Norwegian Krone",
68    "NZD": "New Zealand Dollar",
69    "PHP": "Philippine Peso",
70    "PLN": "Polish Złoty",
71    "RON": "Romanian Leu",
72    "SEK": "Swedish Krona",
73    "SGD": "Singapore Dollar",
74    "THB": "Thai Baht",
75    "TRY": "Turkish Lira",
76    "USD": "United States Dollar",
77    "ZAR": "South African Rand"
78  }
79  `;
```

Das rechte Bild zeigt ein json-Format mit der Kurzbezeichnung (Schlüssel) und den Namen der Währung. Beispiel: Zeile 51, „CHF“ (Schlüssen [Key]) und als Wert den Namen: „Swiss Franc“. Beim json-Format ist der Schlüssel eine tragende Säule.

Wir wollen ja mit den Daten eine Tabelle füllen! Also weisen wir JavaScript an, die Daten in einen Array zu extrahieren. Beispiel: name[Australian Dollar, Bulgarian Lev, Brazilian Real,…..]. Das gleiche mit der Kurzbezeichnung: key[AUD, BGN,…..]. Jetzt haben wir die Möglichkeit für einen einfach Zugriff geschaffen. Beispiel: Get_Name = name[i] Wenn Index i = 8, dann steht in der Variable Get_Name -> Euro.

Wir verwenden JavaScript-Syntax und müssen dann folgendes basteln: 1) Array für den Schlüssel (Key): const currency_key = Object.keys(currenciesJsonString.json()); 2) Array für den Wert (hier Name Währung): const currency_name = Object. values(currenciesJsonString.json()); Wir haben jetzt zwei Array Variablen: const currency_key[0…30] mit dem Inhalt [AUD….ZAR] & const currency[0…30] mit dem Inhalt [Australian Dollar….South African Rands]. Gleichzeitig haben wir dafür gesorgt, dass diese Variablen in der Länge identisch sind mit den Spalten der Tabelle, die auf der HTML-Seite angezeigt wird.

Funktioniert alles, dann wird der String „currenciesJsonString“ gelöscht und durch die Daten ersetzt, die die fetch()-Funktion zurückliefert!

Der nächste Schritt zur Vervollständigung der Tabelle ist die Einbindung der Preise für die unterschiedlichen Devisen. In Abhängigkeit einer Basis-Währung, hier „base – EUR“, stellt die „frankfurter API“ 30 Preise zur Verfügung. Die Basis-Währung (EUR) ist in der Liste nicht enthalten! Folglich gibt es zum Namen-Array und Devisen-Array eine Verschiebung (31 zu 30)! Das müssen wir korrigieren.

```
const exangeJsonString =
{
  "amount": 1,
  "base": "EUR",
  "date": "2025-11-27",
  "rates": {
    "AUD": 1.7753,
    "BGN": 1.9558,
    "BRL": 6.1958,
    "CAD": 1.6273,
    "CHF": 0.9337,
    "CNY": 8.2068,
    "CZK": 24.174,
    "DKK": 7.4688,
    "GBP": 0.8754,
    "HKD": 9.0118,
    "HUF": 381.68,
    "IDR": 19279,
    "ILS": 3.7942,
    "INR": 103.58,
    "ISK": 147.4,
    "JPY": 181.11,
    "KRW": 1694.15,
    "MXN": 21.256,
    "MYR": 4.7873,
    "NOK": 11.803,
    "NZD": 2.0277,
    "PHP": 68.089,
    "PLN": 4.2345,
    "RON": 5.0911,
    "SEK": 10.9945,
    "SGD": 1.5039,
    "THB": 37.336,
    "TRY": 49.171,
    "USD": 1.1586,
    "ZAR": 19.8968
  }
}
`;
```

Erneut weisen wir JavaScript an, aus dem String „exangeJsonString“, bestehend aus Key und Value, ein tabellenkonformes Array zu bauen: `const rate = Object.values(exangeJsonString.json());`

Die zwei Arrays mit Währungsname und Währungs-schlüssen aus der vorherigen Seite sind mit dem neu geschaffenen Array „rate“ in der Länge nicht synchron (31 zu 30). Es fehlt die Basiswährung Euro. Array „rate“ müssen wir erweitern. Genau nach Index 7, wir starten mit 0, muss der Euro eingefügt werden! Also weisen wir JavaScript an: `rate.splice(8, 0, 1.0);` Das Array rate ist nun mit allen anderen Arrays synchron und sieht folgendermaßen aus: rate[1.7753, ….7.4688, 1.0,0.8754,……]. An Indexposition 8 steht nun der Euro mit 1.0 als Basis!

Die Tabelle auf der HTML-Seite sieht dann so aus:

| Australian Dollar | AUD| 1.777753 |
| British Pound | GBP | 0.8754 |

Die zugehörige Datenstruktur in JavaScript sieht dann so aus:

Tab_Spalte_1.1 = const currency_name[0]
Tab_Spalte_1.2 = const currency_key[0]
Tab_Spalte_1.3 = const rate[0]
.....
Tab_Spalte_10.1 = const currency_name[9]
Tab_Spalte_10.2 = const currency_key[9]
Tab_Spalte_20.3 = const rate[9]

Zum Schluss, wenn das Zusammenspiel JavaScript und HTML funktioniert, wird auch dieser String „exangeJsonString“ gelöscht und durch die Daten ersetzt, die die fetch()-Funktion zurückliefert!

Du willst dir das gesamte Projekt ansehen (HTML, CSS & JavaScript)! Kein Problem. Visualisiert Google Chrome die Wechselkurs Applikation, dann betätige die Taste F12. Parallel zur Webseite wird das Werkzeug „Google DevTools“ geöffnet. Die Ähnlichkeit zu einem Datei Explorer ist gegeben. Du kannst dir alle Skripte anschauen, markieren und kopieren.

Liebe Leserinnen und Leser! Grundschüler, Realschüler, Gesellen, Handwerker und Facharbeiter. Hallo du! Ja, genau >>du<<. Wir haben was Gemeinsames! Du sitzt wahrscheinlich gerade vor deinem Bildschirm, starrst auf eine Fehlermeldung, die keinen Sinn ergibt, oder versuchst, dir zum zehnten Mal zu erklären, was dein „Eingeklammertes“ eigentlich macht. Dein Kopf raucht, die Motivation sinkt, und vielleicht denkst du gerade: „Warum tue ich mir das an? Ist Programmieren einfach nichts für mich?“

Stopp! Atme einmal tief durch. Ich möchte dir etwas ganz Wichtiges sagen: Es ist OK, dass es sich schwer anfühlt! JavaScript (oder jede andere Programmiersprache) ist nicht einfach. Es ist eine neue Sprache, ein neues Denksystem. Es ist, als würdest du nicht nur Vokabeln lernen, sondern gleichzeitig versuchen, ein Gedicht in dieser neuen Sprache zu schreiben, während du dir die Grammatik selbst beibringst.

Jeder Code-Krieger, den du bewunderst – der brillante Frontend-Entwickler, der coole Game-Programmierer – jeder von ihnen ist durch genau dieses Tal gegangen. Dieses Gefühl der Frustration, wenn der Code nicht läuft, obwohl er es „sollte“? Das ist nicht das Zeichen, dass du scheiterst. Das ist das Zeichen, dass du lernst. Es ist der Schmerz des Wachstums.

Klein anfangen und feiern! Hat dein `if/else`-Statement funktioniert? Fantastisch! Hat deine Variable den richtigen Wert gespeichert? Mega! Jeder noch so kleine Sieg ist ein Beweis dafür, dass du es verstehst. Feiere diese kleinen Momente. Der Debugger ist dein Freund. Eine Fehlermeldung ist nicht das Ende der Welt. Sie ist eine Nachricht deines Computers. Sie sagt dir: "Hey, ich verstehe das nicht ganz. Kannst du das bitte hier korrigieren?" Lerne, diese Nachrichten zu entschlüsseln. Nutze `console.log()` wie ein Detektiv seine Lupe. Höre auf, dich zu vergleichen. Deine Lernreise ist deine Reise. Es ist egal, wie schnell dein Freund das Framework verstanden hat oder wie viele Projekte jemand auf GitHub hat. Du bist da, wo du sein musst, und du machst Fortschritte. Die Pause ist Produktivität. Wenn dein Gehirn blockiert, steh auf! Mach einen Spaziergang, trink etwas, schau ein YouTube-Video (das nichts mit Code zu tun hat). Die Lösung für das knifflige Problem kommt oft, wenn du nicht direkt darauf starrst. Dein Unterbewusstsein arbeitet weiter.

November 2025, Hans Busche