


import

[Sprachen](#)

Das **import Statement** wird verwendet um Funktionen, Objekte und Primitives zu importieren die von einem externen Modul, einem anderen Script, etc. exportiert wurden.

 **Hinweis:** Zur Zeit wird dieses Feature nicht von jedem Browser nativ unterstützt. Viele Transpiler implementieren es, wie beispielsweise der [Traceur Compiler](#), [Babel](#), [Rollup](#) oder [Webpack](#).

Syntax

```
import name from "module-name";
import * as name from "module-name";
import { member } from "module-name";
import { member as alias } from "module-name";
import { member1 , member2 } from "module-name";
import { member1 , member2 as alias2 , [...] } from "module-name";
import defaultMember, { member [ , [...] ] } from "module-name";
import defaultMember, * as alias from "module-name";
import defaultMember from "module-name";
import "module-name";
```

name

Name des Objekts, das die importierten Daten empfängt

member, memberN

Namen der exportierten Member, die importiert werden

defaultMember

Name des exportierten Defaults, das importiert wird

alias, aliasN

Name des Objekts, das die importierte Property empfängt

module-name

Der Name des Moduls, das importiert wird. Also der Dateiname.

Beschreibung

Der Parameter `name` ist der Name des Objekts, das die exportierten Member empfängt. Die `member`-Parameter legen einzelne Einheiten fest, während der `name` Parameter alles importiert. `name` kann auch eine Funktion sein, wenn das Modul nur eine Einheit hat. Es folgen ein paar Beispiele für die Syntax:

Importieren der gesamten Inhalte des Moduls. Folgendes fügt `myModule` in den aktuellen Namensraum ein, mit allen exportierten Verbindungen von "my-module" bzw. "my-module.js".

```
1 | import * as myModule from "my-module";
```

Einfügen einer einzelnen Einheit eines Moduls. Folgendes fügt `myMember` in den aktuellen Namensraum ein.

```
1 | import {myMember} from "my-module";
```

Einfügen von mehreren Einheiten eines Moduls. Folgendes fügt `foo` und `bar` in den aktuellen Namensraum ein.

```
1 | import {foo, bar} from "my-module";
```

Einfügen und Vergeben eines Alias. Folgendes fügt `shortName` in den aktuellen Namensraum ein.

```
1 | import {reallyReallyLongModuleMemberName as shortName} from "my-module";
```

Einfügen und Vergeben von mehreren Aliasen

```
1 | import {reallyReallyLongModuleMemberName as shortName, anotherLongModuleName as short} f
```

Einfügen eines ganzen Moduls, ohne dessen Namensbindungen zu importieren.

```
1 | import 'my-module';
```

Defaults importieren

Ein Standardexport ist möglich (egal, ob es sich um ein Objekt, eine Funktion, eine Klasse oder anderes handelt). Dementsprechend ist es auch möglich einen Standard-`import` zu benutzen, um diese Standards zu importieren.

Die einfachste Version importiert die Standards direkt:

```
1 | import myModule from "my-module";
```

Man kann diese Syntax auch benutzen, um die oben genannten imports durchzuführen. In diesem Fall müssen die Standards aber wie folgt definiert werden:

```
1 | import myDefault, * as myModule from "my-module";  
2 | // myModule wird als namespace benutzt
```

oder

```
1 | import myDefault, {foo, bar} from "my-module";  
2 | // spezifische Imports nach Namen
```

🔗 Beispiele

Importieren einer weiteren Datei um AJAX JSON-Anfragen zu bearbeiten:





























```
1 // --file.js--
2 function getJSON(url, callback) {
3   let xhr = new XMLHttpRequest();
4   xhr.onload = function () {
5     callback(this.responseText)
6   };
7   xhr.open("GET", url, true);
8   xhr.send();
9 }
10
11 export function getUsefulContents(url, callback) {
12   getJSON(url, data => callback(JSON.parse(data)));
13 }
14
15 // --main.js--
16 import { getUsefulContents } from "file";
17 getUsefulContents("http://www.example.com", data => {
18   doSomethingUseful(data);
19 });
```

🔗 Spezifikationen

| Spezifikation | Status | Kommentar |
|--|--------------------|---------------------|
| 🔗 ECMAScript 2015 (6th Edition, ECMA-262) Die Definition von 'Imports' in dieser Spezifikation. | ST Standard | Initiale Definition |
| 🔗 ECMAScript Latest Draft (ECMA-262) Die Definition von 'Imports' in dieser Spezifikation. | D Entwurf | |

🔗 Browserkompatibilität

Neue Kompatibilitätstabellen sind in der Beta ▼

| Grundlegende Unterstützung | |
|---|---|
|   | 61 |
|   | 16 |
| ▼ | |
|   | 60 |
| ▼ | |
|   | Nein |
|   | 47 |
|   | 10.1 |
|   | Nein |
|   | 61 |
|   | Ja |
|   | 60 |
| ▼ | |
|   | 47 |
|   | 10.1 |
|   | Nein |
|  | 8  |
| ▼ | |



Vollständige Unterstützung



Keine Unterstützung



Benutzer muss dieses Feature explizit aktivieren.

🔗 Siehe auch

- [export](#)
- [Vorschau von Modulen und mehr von ES2015, ES2016 und darüber](#)
- [ES6 in Depth: Modules](#), Hacks Blog Post von Jason Orendorff

- [Axel Rauschmayer's Buch: "Exploring JS: Modules"](#)

🔖 Schlagwörter: [ECMAScript 2015](#) [JavaScript](#) [Module](#) [Statement](#)

👤 Mitwirkende an dieser Seite: [kdex](#), [Snapstromegon](#), [Kani1013](#), [michaelze](#), [yampus](#), [yannick_versley](#), [BennyAlex](#), [Marzelpa](#), [schlagi123](#), [Breaker222](#), [Simmarith](#), [matbad](#)

🕒 Zuletzt aktualisiert von: [kdex](#), 25.04.2018, 00:43:15

[Webtechnologien für Entwickler](#) > [JavaScript](#) > [JavaScript-Referenz](#) > [Anweisungen und Deklarationen](#) > [import](#)

Verwandte Themen

JavaScript

Tutorials:

- ▶ [Einleitend](#)
- ▶ [JavaScript Guide](#)
- ▶ [Fortgeschritten](#)
- ▶ [Erweitert](#)

Referenzen:

- ▶ [Standardobjekte](#)
- ▶ [Ausdrücke & Operatoren](#)
- ▼ [Anweisungen & Deklarationen](#)

[Legacy generator function](#) [\[Übersetzen\]](#)

[async function](#) [\[Übersetzen\]](#)

[block](#)

[break](#)

[Klasse](#)

[const](#)

[continue](#)

[debugger](#) [\[Übersetzen\]](#)

[default](#)

[do...while](#)

[empty](#)

[export](#)

[for](#)

🗨️ 🗑️ for each...in

for...in

for...of

Funktion

function*

if...else

import

label

let

return

switch

throw

try...catch

var

while

🗨️ with [Übersetzen]

- ▶ Funktionen
- ▶ Klassen
- ▶ Fehler
- ▶ Weiteres
- ▶ Neu in JavaScript

Dokumentation:

- ▶ Nützliche Listen
- ▶ Mitmachen

Lernen Sie das Beste aus dem Bereich Web-Entwicklung

×

Erhalten Sie das Neueste und Wichtigste von MDN direkt in Ihren Posteingang.

Der Newsletter wird derzeit nur auf Englisch angeboten.

Melden Sie sich jetzt an





MDN web docs

moz://a

[Web-Technologien](#)

[Lernen Sie Web-Entwicklung](#)

[Über MDN](#)

[Rückmeldung](#)



moz://a

[Über](#)

[Kontakt](#)

[Firefox](#)



Weitere Sprachen: Deutsch (de) ▼

[Nutzungsbedingungen](#) [Datenschutz](#) [Cookies](#)

© 2005–2018 Mozilla und einzelne Mitwirkende. Inhalt steht unter [diesen Lizenzen](#).