Module in TypeScript

# ECMAScript

* Lange wurden Module nur über Bibliotheken implementiert und nicht in die Sprache eingebaut
* Seit ES6 gibt es built-in modules
* Module werden in ihrem eigenen Bereich ausgeführt, nicht global (heißt sie sind außerhalb des Moduls nicht sichtbar)
* AUßER sie werden explizit exportiert
* Umgekehrt muss eine Variable, Funktion, Klasse, Schnittstelle etc., die aus einem anderen Modul exportiert wurde, importiert werden, um darauf zugreifen zu können
* Jede Deklaration (Variable, Funktion, Klasse, Schnittstelle, Typ-Alias) kann exportiert bzw. importiert werden
* In TypeScript wird also jede Datei, die einen Import oder Export der obersten Ebene enthält, als Modul betrachtet

### Export and import

Export Anweisungen

* Benannte Exports sind nützlich, um mehrere Werte zu exportieren
* Beim Import kann man denselben Namen verwenden, um auf den entsprechenden Wert zu verweisen

export {*name1, name2, …, nameN*};

//exportiert z.B. mehrere Funktionen, die zuvor deklariert wurden

export { *variable1* as *name1,* *variable2* as *name2, …, nameN*};

//exportiert mehrere Variablen, die direkt in Exportanweisung umbenannt werden

export let *name1, name2, …, nameN;* // oder: var

//exportiert eine zuvor deklarierte Konstante

export let *name1*= …,*name2*= …, …,*nameN;* // oder: var, const

//exportiert eine Konstante

* Pro Modul kann es nur einen einzigen Default-Export geben
* Ein Default-Export kann eine Funktion, Klasse, Objekt, … sein

export default *expression;*

export default function (…) { … } // oder: class, function\*

export default function *name1*(…) { … } // oder: class, function\*

export { name1as default, … };

export \* from *…;*

export { *name1, name2, …, nameN*}from …;

export {*import1* as *name1, import2* as *name2, …, nameN*} from …;

* Module werden importiert durch absolute oder relative Verweise und müssen daher mit „/“, „./“ oder „ ../ “ beginnen
* Parameter name 🡪 Name des Objekts, das die exportierten Member empfängt (kann auch eine Funktion sein)
* Parameter member legt einzelne Einheiten fest

Import Anweisungen

import *name* from "module-name";

//gesamter Inhalt des Moduls wird importiert

import \* as *name* from "module-name";

// ein Namespace-Objekt wird importiert, mit den Eigenschaften des Exports des Moduls

import {*member*} from "module-name";

//einzelne Einheit (member) wird importiert

import {*member* as *alias* } from "module-name";

//einzelne Einheit wird importiert als … (Umbenennung)

import { *member1* *, member2* } from "module-name";

//mehrere Einheiten eines Moduls werden importiert

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

//Default Imports

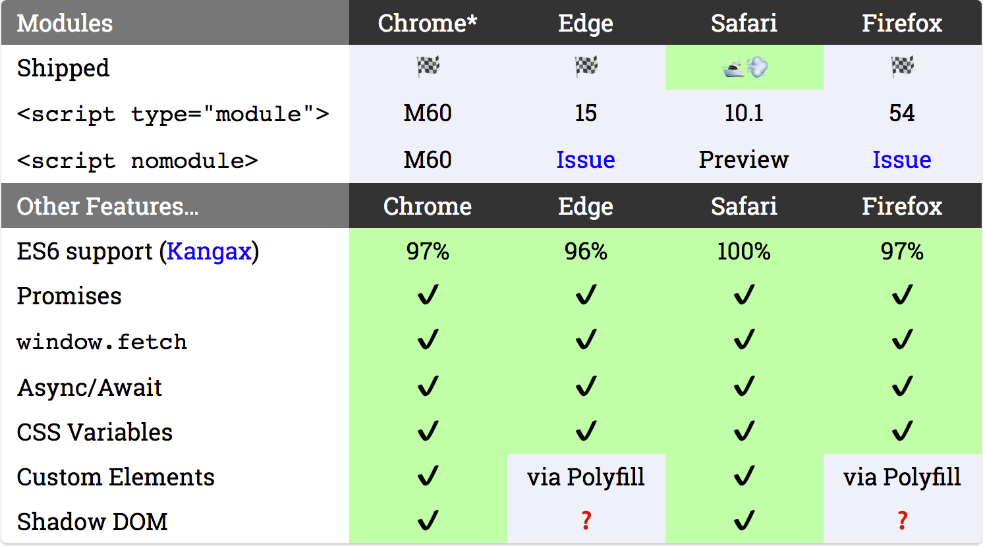
import *defaultMember,*{*member*[ , [...] ] } from "module-name";

import *defaultMember*, \* as *alias* from "module-name";

import *defaultMember* from "module-name";

import "module-name";

//ganzes Modul wird importiert, ohne dass die Namensbindung mit importiert wird



### Support

Browsers with ES6 modules support

### einbinden in html-Datei

* Module müssen in die HTML-Datei eingebunden werden durch type =“module“
* Dies kann durch einen inline- oder externen Script-Tag geschehen
* Das Dokument muss erst geladen sein, um die Module zu verwenden

<script **type="module"** src="main.js"></script>  
<script **type="module"**>  
 // or an inline script  
</script>

### Quellen und weiterführende Infos:

<https://medium.com/backticks-tildes/introduction-to-es6-modules-49956f580da>

<https://www.typescriptlang.org/docs/handbook/declaration-files/templates/module-d-ts.html>

<https://www.typescriptlang.org/docs/handbook/modules.html>

<https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Statements/import>