Modules JavaScript

Dernière mise à jour : 07/02/2024 12:30

Contexte

- Aux débuts de JavaScript
  - programmes assez petits réalisant des tâches isolées uniquement là où l'interactivité était nécessaire
- JavaScript 'moderne'
  - de simples programmes on est passé à des applications complètes au code complexe et volumineux
  - exécution de programmes en dehors du contexte d'un navigateur (Node.js)

→ besoins d'organiser le code

  - découpage en modules
  - chargement des modules à la demande

Contexte

- plusieurs bibliothèques on été proposées pour la mise en œuvre et l'utilisation de modules
  - AMD – initialement mis en œuvre par la bibliothèque require.js.
  - CommonJS – le système de module créé pour Node.js
  - UMD – système de module universel, compatible avec AMD et CommonJS

- 2015 apparition d'un système de modules intégré au langage → modules ES
  - pris en charge maintenant par les principaux navigateurs et Node.js

Qu'est-ce qu'un module ?

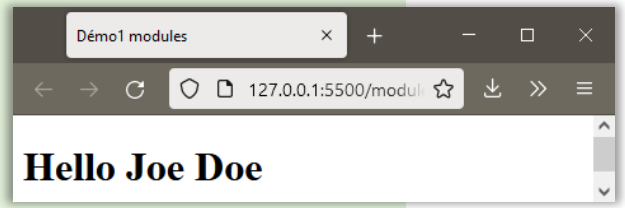
- un module ES ⇔ fichier JavaScript (un script est un module)
- chaque module a sa propre portée globale :
  - variables et fonctions globales d'un module ne sont par défaut pas visibles dans les autres scripts.
  - un module doit exporter (directive `export`) les ressources qu'il veut rendre accessibles depuis l'extérieur
  - un module doit importer (directive `import`) les ressources dont il a besoin

```
// 🗀 module1.js
export let userName = "Joe Doe";
```

```
// 🗀 module2.js
import {userName} from './module1.js';

name.innerHTML = userName;
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Démo1 modules/title>
  </head>
  <body>
    <h1>Hello <span id="user"></span></h1>
    <script src="./module1.js" type="module"></script>
    <script src="./module2.js" type="module"></script>
  </body>
</html>
```



indique que le code JavaScript constitue un module

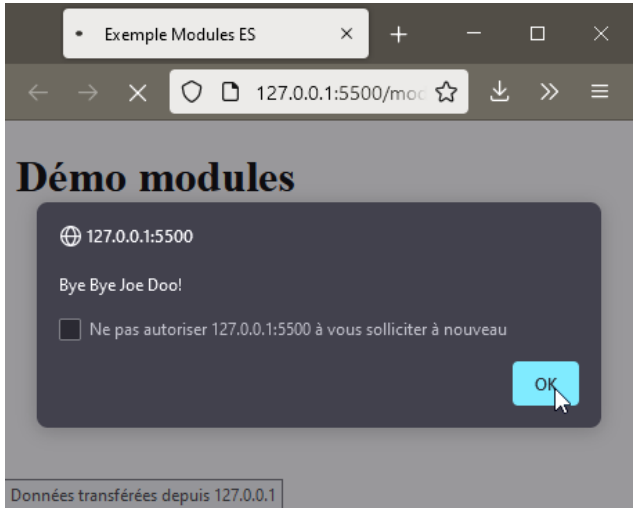
Modules dans un navigateur

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Exemple Modules ES</title>
</head>
<body>
    <h1>Démo modules</h1>
    <script src="./sayModule.js" type="module"></script>
    <script src="./mainModule.js" type="module"></script>
</body>

</html>
```

il faut indiquer que le code JavaScript constitue un module

Le navigateur extrait et évalue automatiquement le module importé (et, le cas échéant, ses importations), puis exécute le script.




```javascript
// 🗀 sayHello.js
export function sayHelloBye(user) {
  sayHello(user);
  sayByeBye(user);
}

function sayHello(user) {
  alert(`Hello, ${user}!`);
}

function sayByeBye(user) {
  alert(`Bye Bye ${user}!`);
}
```

```javascript
// 🗀 main.js
import {sayHelloBye} from './sayModule.js';

sayHelloBye('Joe Doo');
```

Les modules ne fonctionnent pas localement (via protocole file://) il faut obligatoirement passer par un serveur HTTP(s)

Modules et node.js

- ..

pour utiliser modules ES2015 avec node il faut mettre extension `.mjs`

labélise les fonctions et variables qui pourront être utilisées en dehors du module

```
// 🗀 sayHello.mjs
export function sayHelloBye(user) {
  sayHello(user);
  sayByeBye(user);
}

function sayHello(user) {
    console.log(`Hello ${user}!`);
  }

function sayByeBye(user) {
  console.log(`Bye bye ${user}!`);
}
```

permet importation de certaines fonctionnalités d'un autre module

chemin du module par rapport au fichier actuel

```
// 🗀 mainModule.mjs
import {sayHelloBye} from './sayModule.mjs';

console.log(sayHelloBye);
sayHelloBye('Joe Doo');
```

affecte la fonction exportée `sayHelloBye` à la variable correspondante.

```
λ node mainModule.mjs
[Function: sayHelloBye]
Hello Joe Doo!
Bye bye Joe Doo!
```

.mjs ou .js ?

- documentation de V8 recommande d'utiliser extension `.mjs`
  - meilleure clarté pour distinguer fichiers qui sont des modules des fichiers javascript classiques
  - permet d'analyser correctement les fichiers modules par les environnements d'exécution tels Node.js et outils de compilation tels Babel
  - par contre pour déployer des fichier `.mjs` sur le web il faut s'assurer que le serveur est configuré pour utiliser le type MIME approprié :
    - `Content-Type: text/javascript`
  - mais si vous n'avez pas accès à la configuration du serveur il se peut qu'il n'associe pas le bon type au fichier `.mjs` , dans ce cas mieux vaut garder l'extension `.js`

  - pour en savoir plus :
    - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules#aside_%E2%80%94_.mjs_versus_.js
    - https://v8.dev/features/modules#mjs