

# Développement front avancé

MMI 3 – TP#3 S6

Danielo **JEAN-LOUIS**

# Import de scripts

- Utilisation de la balise `<script>`
- Utilisation de modules
  - Via npm et un bundler (S5)
  - L'attribut `type="module"`
- Et...

# importmap

- Nouveauté gérée par tous les navigateurs depuis 2023
- Valeur de l'attribut "type" de la balise `<script>`
- Permet de lister les modules front à charger et où les télécharger sous forme de JSON

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/script/type/importmap>

# importmap



```
<script type="importmap">
{
  "imports": {
    "lodash": "https://cdn.skypack.dev/lodash",
    "luxon": "/node_modules/luxon/src/luxon.js"
  }
}
</script>
```

Ici, nous définissons deux modules : lodash (distant) et luxon (local)

**Source(s) :**

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/script/type/importmap>

# Importmap – Règles à respecter

- Respecter le format JSON
- Les chemins doivent être une URL valide ou commencer par /, ./ ou ../
  - Le chemin se fait entre l'importmap et le module
- Si une clé termine par un /, la valeur doit faire de même

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/script/type/importmap>

# importmap



JS

```
import lodash from "lodash";
```

On importe notre module comme l'accoutumée  
(On n'oublie pas de rajouter "type=module" pour la balise <script>)

## importmap - Avantages

- Simplification de la gestion des modules
  - Raccourcissement des noms
  - Possibilité de gérer facilement plusieurs versions du même module
- Centralisation des versions de modules

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/script/type/importmap>



# importmap - Avantages

- Pas besoin de bundlers ou de npm/yarn
- Possibilité de renommer les modules lors de l'import
  - Impossible avec un bundler

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/script/type/importmap>

# Pratiquons ! - importmaps (Partie 1/2)



Pré-requis :

- Avoir la ressource `ressources/importmap`

A télécharger ici :

[https://github.com/DanYellow/cours/raw/refs/heads/main/developement-web-et-dispositif-interactif-s6/travaux-pratiques/numero-3/developpement-web-et-dispositif-interactif-s6\\_travaux-pratiques\\_numero-3.ressources.zip](https://github.com/DanYellow/cours/raw/refs/heads/main/developement-web-et-dispositif-interactif-s6/travaux-pratiques/numero-3/developpement-web-et-dispositif-interactif-s6_travaux-pratiques_numero-3.ressources.zip)

# importmap



```
<script type="importmap">
{
  "imports": {
    "@modules/": "/modules/"
  }
}
</script>
```



```
import myFunction from "@modules/utils.js";
```

Possibilité de lister un dossier

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/script/type/importmap>

# Importmap – Fichier externe

- Fichier au format json
- Doit avoir le MIME type  
“application/importmap+json”
  - Pas géré par les navigateurs (01/2025)

## Source(s) :

- <https://github.com/WICG/import-maps?tab=readme-ov-file#installation>

# importmap – Fichier externe



```
<script type="importmap" src="importmap.json"></script>
```

On charge un importmap externe si son MIME type est “application/importmap+json”.

Il est possible, à l’avenir, que l’extension “.importmap” arrive.

## Source(s) :

- <https://github.com/WICG/import-maps?tab=readme-ov-file#installation> - anglais
- <https://www.honeybadger.io/blog/import-maps/> - anglais

# importmap - Scopes

- Possibilité de charger les modules en fonction de l'emplacement du module qui les charge
- Définition d'une clé "scopes" dans l'importmap

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/script/type/importmap>

# Importmap - Scopes



```
<script type="importmap">
  {
    "imports": {
      "lodash": "https://cdn.jsdelivr.net/npm/lodash@4.17.21/+esm"
    },
    "scopes": {
      "/test/": {
        "utils": "../test/utils.js"
      }
    }
  }
</script>
```

Le module “utils” ne sera disponible que si l’URL contient “test”

# Importmap - Scopes

- Si une url correspond à plusieurs modules identiques, le navigateur utilisera celui avec le chemin le plus précis

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/script/type/importmap>



# Importmap - Scopes



```
<script type="importmap">
  {
    "imports": {
      "lodash": "https://cdn.jsdelivr.net/npm/lodash@4.17.21/+esm"
    },
    "scopes": {
      "/test/": {
        "lodash": "https://cdn.jsdelivr.net/npm/lodash@4.6.0/+esm"
      }
    }
  }
</script>
```

Si le lien contient “test”, il chargera la version 4.6.0 de lodash,  
les autres la version 4.17.21

# importmap - Scopes

- Si le module est chargé dans une page où il ne peut pas l'être, le code sera interrompu
  - Charger le script de façon asynchrone si vous n'êtes pas sûr(e)

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/script/type/importmap>

# Pratiquons ! - importmaps (Partie 3)

Pré-requis :

- Avoir la ressource `ressources/importmap`

A télécharger ici :

[https://github.com/DanYellow/cours/raw/refs/heads/main/developement-web-et-dispositif-interactif-s6/travaux-pratiques/numero-3/developpement-web-et-dispositif-interactif-s6\\_travaux-pratiques\\_numero-3.ressources.zip](https://github.com/DanYellow/cours/raw/refs/heads/main/developement-web-et-dispositif-interactif-s6/travaux-pratiques/numero-3/developpement-web-et-dispositif-interactif-s6_travaux-pratiques_numero-3.ressources.zip)

# Importmap - Limites

- Pas adapté pour les gros projets
  - A préférer pour le prototypage
- Impossibilité de :
  - Faire du tree shaking ou du hot reloading
  - Charger plusieurs importmap par page

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/script/type/importmap>
- [https://developer.mozilla.org/fr/docs/Glossary/Tree\\_shaking](https://developer.mozilla.org/fr/docs/Glossary/Tree_shaking)

**Questions ?**

