Apprendre le JavaScript - Les modules

Les modules sont une fonctionnalité de JavaScript qui permet de séparer le code en plusieurs fichiers. Cela permet de mieux organiser le code et de le rendre plus lisible. Les modules sont très utiles pour séparer les différentes responsabilités d'un programme en plusieurs parties distinctes.

Pour utiliser les modules, il faut ajouter l'attribut `type="module"` à la balise `<script>` dans le fichier HTML. Cela indique au navigateur que le fichier JavaScript est un module.

À noter que le fait d'utiliser des modules implique que le code JavaScript est exécuté en mode strict et il n'est plus nécessaire de mettre l'attribut `defer`.

Autre point, vous devez utiliser un serveur local pour charger les modules.Ex: Live Server de Visual Studio Code. Vous ne pouvez pas charger un fichier module directement dans le navigateur.

```
<script type="module" src="script.js"></script>
```

Communiquer entre les modules

Pour communiquer entre les modules, il est possible d'exporter des fonctions, des variables ou des classes avec le mot-clé `export`. Il est ensuite possible d'importer ces éléments dans un autre module avec le mot-clé `import`. Cela est similaire aux instructions PHP `require` et `include`.

Quand utiliser les modules

Les modules sont très utiles pour organiser le code en plusieurs fichiers. Ils permettent de séparer les différentes responsabilités d'un programme en plusieurs parties distinctes.

On peut regrouper des fonctions utilitaires dans un module, des classes dans un autre module et le code principal dans un autre module. Cela permet de mieux organiser le code et de le rendre plus lisible. On peut mettre des données de test dans un module et les fonctions de manipulation de ces données dans un autre module.

Exporter des éléments d'un module

Pour exporter des éléments d'un module, il faut utiliser le mot-clé `export` suivi du nom de l'élément à exporter. Cette exportation est maintenant disponible pour être importée dans un autre module. L'avantage est que l'on peut séparer les fonctions, les variables et les classes dans des fichiers différents.Ils sont réutilisables dans différentes parties de l'application.

On peut exporter plusieurs éléments à partir du même fichier.

Ce qui n'est pas exporté, n'est pas accessible depuis l'extérieur du module. Cela rend le code plus sécurisé et plus facile à maintenir.

```
// script.js
export function direBonjour() {
    console.log("Bonjour !");
}

export const prenom = "Jean";

export class Personne {
    constructor(nom) {
        this.nom = nom;
    }
}
```

Exportation par défaut

Il est possible d'exporter un élément par défaut en utilisant le mot-clé `default`. Cela permet d'importer cet élément sans avoir à préciser son nom. Il ne peut y avoir qu'un seul export par défaut par module.

```
// script.js
export default function direBonjour() {
    console.log("Bonjour !");
}
```

Importer des éléments d'un module

Pour importer des éléments d'un module, il faut utiliser le mot-clé `import` suivi du nom de l'élément à importer entre accolades `{}`. Il est possible d'importer plusieurs éléments en les séparant par des virgules.

```
// main.js
import { direBonjour, prenom } from "./script.js";

direBonjour();
```

Importation par défaut

Pour importer un élément par défaut, il suffit de ne pas utiliser les accolades `{}`.

```
// main.js
import direBonjour from "./script.js";

direBonjour();
```

Renommmer une importation

Il est possible de renommer une importation en utilisant le mot-clé `as`. Ainsi, si vous avez plusieurs fonctions ou variables avec le même nom, vous pouvez les renommer pour éviter les conflits.

```
// main.js
import { direBonjour as hello } from "./script.js";
```