

Javascript Avancé partie 3

Module

Les modules permettent de diviser les programmes JavaScript en plusieurs parties qu'on pourrait importer les uns dans les autres.

Cette fonctionnalité était présente dans Node.js et à été intégré dans les navigateurs.

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Modules>

Module compatibilités

														
	 Chrome	 Edge	 Firefox	 Internet Explorer	 Opera	 Safari	 WebView Android	 Chrome Android	 Firefox for Android	 Opera Android	 Safari on iOS	 Samsung Internet	 Deno	 Node.js
<code>import</code>	✓ 61	✓ 16	✓ 60	✗ No	✓ 48	✓ 10.1	✓ 61	✓ 61	✓ 60	✓ 45	✓ 10.3	✓ 8.0	✓ 1.0	✓ 13.2.0 ✱
Dynamic import	✓ 63	✓ 79	✓ 67	✗ No	✓ 50	✓ 11.1	✓ 63	✓ 63	✓ 67	✓ 46	✓ 11.3	✓ 8.0	✓ 1.0	✓ 13.2.0 ✱
Available in workers	✓ 80	✓ 80	✗ No ✱	✗ No	✗ No	✓ 15	✓ 80	✓ 80	✗ No ✱	✗ No	✓ 15	✓ 13.0	✓ 1.0	✗ No

✓ Full support ✗ No support ✱ See implementation notes.

Charger le module via le document HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Module JS</title>
  </head>
  <body>
    <script type="text/javascript" src="index.js"></script>
  </body>
</html>
```



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Module JS</title>
    <script type="module" src="index.js"></script>
  </head>
  <body>

  </body>
</html>
```

Exporter des fonctionnalités

La méthode la plus simple consiste à placer l'instruction export devant chaque valeur qu'on souhaite exporter, par exemple :

```
export function ucFirst(texte) {  
    if(typeof texte === "string" && texte !== "") {  
        return texte.replace(texte[0], texte[0].toUpperCase());  
    }  
  
    return "Merci de mettre une string valide";  
}
```

Importer des fonctionnalités

Lorsque des fonctionnalités sont exportées par un premier module, on peut les importer dans un script afin de les utiliser. Voici la méthode la plus simple pour ce faire :

```
import { ucFirst } from "../modules/script.js"
```

On utilise ici l'instruction `import`, suivie d'une liste d'identifiants séparés par des virgules et délimitée par des accolades, suivie du mot-clé `from` puis du chemin vers le fichier du module. Le chemin est relatif à la racine du site.

Exercie pratique

Modifier l'organisation de votre dossier avec les exercices de la partie 1 pour exporter les fonctions et les utiliser dans le document html.

Attention il faut que vos fichiers soit sur un serveur pour que cela puisse fonctionner.

Promise

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Promise

Promise est un objet qui permet de réaliser des traitements de façon asynchrone.

- Une promesse représente une valeur qui peut être disponible maintenant, dans le futur voire jamais.
- L'objet Promise prend en paramètre une callback avec deux paramètres, par convention on les nomme resolve et reject

```
const maPromesse = new Promise((resolve, reject) => {  
  setTimeout(() => {  
    resolve('toto');  
  }, 300);  
});
```


Promise

- Pour pouvoir utiliser la valeur qui a été resolve par notre promesse les Promise ont la méthode [.then](#)
- Dans certain cas on va vouloir reject une certaine valeur, dans ce cas les Promise ont la méthode [.catch](#)

```
const maPromesse = new Promise((resolve, reject) => {
  setTimeout(() => {
    if (Math.floor(Math.random() * 2) === 1) {
      resolve('foo');
    }
    reject("this aint it");
  }, 300);
});

maPromesse
  .then(value => { return value + " and bar"; })
  .then(value => { return value + " and bar again"; })
  .then(value => { console.log(value); })
  .catch(value => { console.log(value); })
```

Exemple

- Créez un bouton qui va quand il est cliqué exécuter une fonction testPromise
- Créez une fonction testPromise qui va :
 - Quand elle commence afficher le text "1"
 - Créez une Promise qui va dans sa callback afficher le text "2" puis va dans un setTimeout avec un temps aléatoire ($\text{Math.random()} * 2000 + 1000$) resolve "3"
 - Utilisez le then sur la Promise créez précédemment pour récupérer la valeur que vous avez résolue et afficher la
 - Avant de finir la fonction afficher le text "4"
- Avant d'exécuter le code, essayer de deviner l'ordre de sortie

Exercice

- 1) Créez une fonction getToken
- 2) Cette fonction retourne une promesse
- 3) Utiliser un setTimeout qui durera 2000 millisecondes
- 4) Tester dans la callback du setTimeout :
 if (Math.random() > 0.5)
- 5) resolve avec la valeur d'un token = "qsdfEDLSoie5d8899;dEDd"
- 6) Sinon reject avec l'erreur "Vous n'avez pas pu obtenir de token"
- 7) Utiliser le then sur la promise pour récupérer la valeur et l'afficher
- 8) Utiliser le catch sur la promise pour récupérer le message d'erreur et l'afficher

Exercice chainage de promesses

- 1) Créez une fonction `getUser` qui prend un token en paramètre
- 2) Cette fonction retourne une promesse
- 3) Utiliser un `setTimeout` qui durera 2000 millisecondes
- 4) Tester dans la callback du `setTimeout` :
 `if (Math.random() > 0.5)`
- 5) `resolve` avec la valeur correspondant à un objet user : `{ id: 1, token: token }`
- 6) Sinon `reject` avec l'erreur « Pas d'utilisateur »
- 7) Le premier `then` appelle la promesse `getToken` pour récupérer le token qui va retourner l'appel à la promesse `getUser` en lui passant le token récupéré.
- 8) Le deuxième `then` affiche la valeur renvoyée par la promesse `getUser`
- 9) Un `catch` affiche le message d'erreur

Async / Await

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Statements/async_function

Le but des fonctions async / await est de simplifier l'utilisation synchrone des promesses et d'opérer sur des groupes de promesses

```
function waitFor(time) {  
  return new Promise((resolve, reject) => {  
    setTimeout(function() {  
      resolve();  
    }, time)  
  });  
}  
  
function test() {  
  waitFor(1000).then(() => {  
    console.log("Helloworld");  
  });  
}  
  
test(); // Helloworld (after 1 sec)
```

```
function waitFor(time) {  
  return new Promise((resolve, reject) => {  
    setTimeout(function() {  
      resolve();  
    }, time)  
  });  
}  
  
async function test() {  
  await waitFor(1000).then(() => {  
    console.log("Helloworld");  
  });  
}  
  
test(); // Helloworld (after 1 sec)
```

Exercice

- 1) En utilisation les deux promise précédente : getToken et getUser
- 2) Créez une fonction getTokenUser qui sera async
- 3) Dans un try ... catch, appeler dans cette fonction la promise getToken pour récupérer le token avec await
- 4) Appeler la promise getUser pour récupérer le user avec await
- 5) Afficher le token et le user
- 6) Afficher dans le catch le message d'erreur

Ajax / Fetch

- Comment faire des requêtes en JS ?
- On peut utiliser [XMLHttpRequest](#) qui a été historiquement utilisé pour faire du [AJAX](#)
- Mais on va préférer utiliser la [Fetch API](#) qui est plus souples et plus puissantes.

```
let ajax = new XMLHttpRequest();  
// ouvre le url  
ajax.open("GET", "https://jsonplaceholder.typicode.com/posts/1");  
// on écoute l'évènement load de la requête  
ajax.addEventListener("load", () => {  
    // quand c'est terminé, on affiche le résultat  
    console.log(JSON.parse(ajax.response));  
});  
console.log("test");  
// envoi la requête  
ajax.send();
```

```
fetch("https://jsonplaceholder.typicode.com/posts/1")  
    .then(result => {  
        return result.json()  
    }).then(data => {  
        console.log(data);  
    });
```

Exercice – Récupérer des GIF

- 1) Créez un compte : <https://tenor.com/developer/keyregistration>
- 2) Validez votre compte et récupérez votre API KEY
- 3) Créer un fichier gifs.html qui contient un formulaire avec 2 champs: un pour du texte qui servira au terme recherché, et l'autre pour le nombre max de gif à récupérer
- 4) Depuis un fichier JS, écrire le code pour faire un appel à l'API (fetch) sur <https://g.tenor.com/v1/search>
 - a) Vous devez y renseigner plusieurs QueryStrings :
 - q={le terme utilisé pour la recherche}
 - key={Votre API KEY}
 - limit={le nombre de résultat}

Exemple = <https://g.tenor.com/v1/search?q=pikachu&key=87FVZSODBI&limit=10>

- 5) Récupérer les urls des gifs (.gif) et les injecter dans des balises dans votre page HTML