

Les Modules

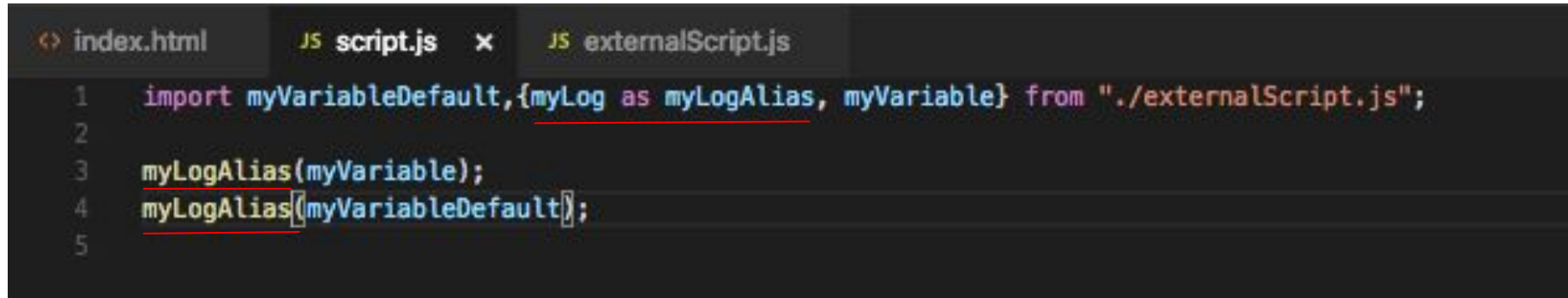
Note : Il faut bien prendre en considération que une variable, un éléments...etc, importer dans un autre fichier puis est modifié le seras dans l'ensemble du programme car on importe bien la variable ou la référence si il s'agit d'un objet, on ne créer pas de copie.

```
<> index.html x JS script.js JS externalScript.js
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Javascript</title>
5   <meta charset="utf-8" />
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <link rel="stylesheet" href="style.css">
8 </head>
9 <body>
10   <script type="module" src="script.js"></script>
11
12 </body>
13
14 </html>
```

```
<> index.html JS script.js x JS externalScript.js
1 import {myLog, myVariable} from "../externalScript.js";
2
3 myLog(myVariable);
4
```

```
<> index.html x JS script.js JS externalScript.js x
1 const myLog = message => console.log(`** My Message is: ${message}`);
2
3 let myVariable = "Coucou !";
4
5 export {myLog, myVariable};
```

On peut donner un alias à nos import et les utiliser comme tel.

A screenshot of a code editor with three tabs: 'index.html', 'JS script.js', and 'JS externalScript.js'. The 'JS script.js' tab is active. The code in the editor is as follows:

```
1 import myVariableDefault,{myLog as myLogAlias, myVariable} from "../externalScript.js";
2
3 myLogAlias(myVariable);
4 myLogAlias(myVariableDefault);
5
```

The code uses ES6 import syntax to import 'myLog' from 'externalScript.js' and assigns it the alias 'myLogAlias'. It then uses 'myLogAlias' to call functions from the same module. Red underlines are present under 'myLogAlias' in both the import statement and the function calls.

Importer la totalité d'un module avec *

En important la totalité d'un module avec *, on doit lui donner un Alias, à partir de là le module que l'on importe est sous la forme d'un objet, et comme tout objet pour accéder à ces propriétés il faut d'abord précéder le nom de la propriété de l'Alias que on lui aura donné.

Alias

```
index.html  JS script.js  JS externalScript.js
1  import * as external from "./externalScript.js";
2
3  console.log(external);
4
5  console.log(external.myVariable);
6
```

```
script.js:3
Module {myVariable: "Coucou !", myVariable2: "Important", Symbol(Symbol.toStringTag): "Module"}
  ▶ myLog: message => console.log("*** MyLog ***")
    myVariable: "Coucou !"
    myVariable2: "Important"
```

```
*** MyLog ***: Coucou !
>
```

Les exports par DEFAULT

On peut avoir qu'un seul export par défaut par fichier, on a la possibilité dans le fichier ou on import l'élément par défaut de le nommer comme on le veut cela n'impactera pas la valeur de l'élément qu'on aura préalablement définis dans le fichier la ou on l'aura déclarer.

```
<> index.html JS script.js JS externalScript.js x
1  const myLog = message => console.log(`** My Message is: ${message}`);
2
3  let myVariable = "Coucou !";
4
5  let myVariable2 = "Important";
6
7  export{myLog, myVariable};
8  export default myVariable2;
```

La valeur restera la String "Important" car il a été déclaré ainsi.

Le nom donné dans notre fichier script doit nous servir de repère utile et adapté au contexte du fichier.

```
<> index.html JS script.js x JS externalScript.js
1  import myVariableDefault,{myLog, myVariable} from "./externalScript.js";
2
3  myLog(myVariable);
4  myLog(myVariableDefault);
5
```