ES6 Javascript module

Objectif : faciliter la structuration et éviter les conflits de noms !

Cours

Module ES6

Objectif

😈Comprendre la problématique des conflits de noms.

Mise en place

Dans VScode créez un fichier index.html

index.html

```
1. <!DOCTYPE html>
2. <html lang="en">
3.
4. <head>
5.   <meta charset="UTF-8">
6.   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7.   <title>Document</title>
8.   <script type="module" src="script.js"></script>
9. </head>
```

```
10.
11. <body>
12.
13. </body>
14.
15. </html>
```

🪛Créez un fichier Square.js

Square.js

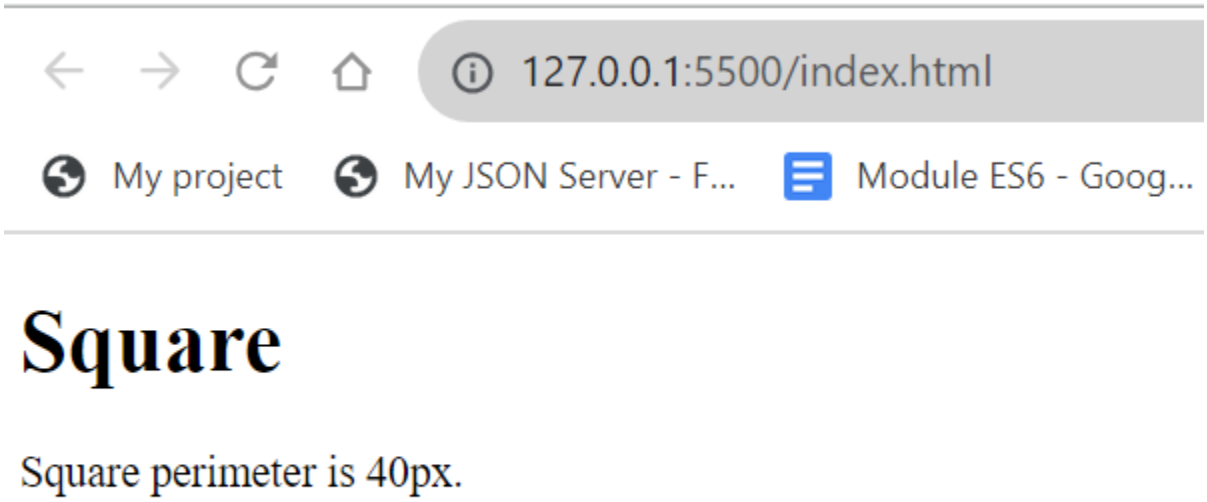
```
1.  export const name = `Square`;
2.  export default class Square {
3.    constructor(x) {
4.      this.x = x;
5.    }
6.    perimeter() {
7.      return this.x * 4;
8.    }
9.  };
10.
11. export function printPerimeter(square) {
12.  let para = document.createElement('p');
13.  para.textContent = `${name} perimeter is ${square.perimeter()}px.`
14.  document.body.appendChild(para);
15. }
```

🪛Créez un fichier script.js

script.js

```
1. import Square, { name, printPerimeter } from "./Square.js";
2. 
3. document.body.insertAdjacentHTML("beforeend", `<h1> ${name} `)
4. const square = new Square(10);
5. printPerimeter(square);
```

🪛Testez le code en lançant un serveur (extension : live-serveur).



🪛En s'inspirant du code de l'exemple précédent, créez un fichier Circle.js.

Voici le code associé.

Circle.js

```
1.  export const name = `Circle`;
2.  export default class Circle {
3.    constructor(radius) {
4.      this.radius = radius;
5.    }
6.    circumference() {
7.      return Math.round(2 * Math.PI * this.radius)
8.    }
9.  };
10.
11. export function printPerimeter(circle) {
12.   let para = document.createElement('p');
13.   para.textContent = `${name} perimeter is ${circle.circumference()}px.`
14.   document.body.appendChild(para);
15. }
```

🪛Maintenant, modifiez le fichier  script.js pour obtenir le résultat suivant :

Dans ce fichier, vous importez les deux classes et fonctions pour afficher le périmètre d'un carré et d'un cercle.

💡Il nous faut comprendre le problème de l'importation de fonctions (variable, class, …) du même nom. Il est clair que nous ne pouvons pas changer le code des modules importés (une des raisons serait que le code de ces modules est géré par une autre équipe).

🥷La seule solution est de régler le problème de renommage lors de l'importation.

Solution

🪛Modifiez le fichier script.js

script.js

```
1. import Square, { name as squareName, printPerimeter as
   printSquarePerimeter } from "./code.js";
2. import Circle, { name as circleName, printPerimeter as
   printCirclePerimeter } from "./circle.js";
3.
4. document.body.insertAdjacentHTML("beforeend", `<h1> ${squareName}
   </h1>`);
5. const square = new Square(10);
6. printSquarePerimeter(square);
7.
```

8. document.body.insertAdjacentHTML("beforeend", `<h1> ${circleName} </h1>`);
9. const circle = new Circle(10);
10. printCirclePerimeter(circle);

Explications

Alias des importations :

Lig : 1 import { name as squareName, printPerimeter as printSquarePerimeter } from "./code.js"; : Ceci importe name et printPerimeter de code.js et les renomme respectivement en squareName et printSquarePerimeter.

Lig : 2 import { name as circleName, printPerimeter as printCirclePerimeter } from "./circle.js"; : Ceci import name et printPerimeter de circle.js et les renomme respectivement circleName et printCirclePerimeter.

Utilisation d'importations aliasées :

Lig : 4 : utilise l'alias squareName pour insérer le nom du carré.

Lig. 6 : printSquarePerimeter(square); : Cette commande utilise l'alias

🥷L'utilisation de l'alias permet d'éviter les conflits de noms et de distinguer clairement les importations provenant de différents modules.