单个js文件中使用ES6模块化

在实际的开发项目中，存在着多种角色。前端就是其中一种，在大型项目中，前端也不可能只有一个，在这样的项目中，怎样协同工具，怎么解决js代码之间的依赖关系，代码怎样维护是个棘手的问题。 ES6终于有了module来解决这个问题，那些实现S模块化的框架终将成为前尘往事，AMD、CMD、CommonJs、ES6的对比。

ES6模块化写法

- 一个js文件就可以看成一个模块，如我们现在有两个模块moduleA.js和moduleB.js
- 模块中（.js文件中）的变量是不会被外界所访问到，除非使用export命令对外暴露接口
- 用import命令可以导入其他模块中的数据（属性或方法）供自己使用。
- 引入有模块化语法的js文件的时候，需要在script标签中加上type='module'属性

导出语法

```
//config.js
const num=1;
let max=50000;
const banks=['中国银行','建设银行'];
export {
  min,
  max as maxMoney,  //可以用as关键字起别名
  banks
}
//money.js
let moneyUtils = {
  formatMoney: function (money) { //...
  },
  undoMoney: function (money) { //...
  }
};
export default moneyUtils;
```

别名

在import和export命令中，可以用as关键字给数据设置别名，设置了别名后，原名就不起作用了。 如：

```
import{xxx as xxx} from 'xxx';
export {xxx as xxx} from 'xxx';
```

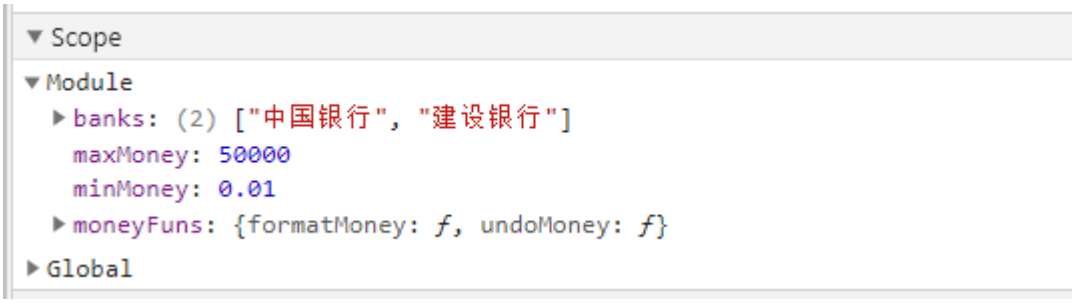
export default

- export default 向外暴露的成员，可以使用任意的变量来接收
- 在一个模块中，export default 只允许向外暴露1次
- 在一个模块中，可以同时使用 export default 和 export 向外暴露成员

导入语法

html文件

```
<script type="module">
import {min as minMoney,maxMoney,banks} from './config.js';
import moneyFuns from './money.js';  //使用moneyFuns接收money.js中export default的数据
console.log(minMoney,maxMoney,banks);
console.log(moneyFuns);
script>
```



- 每一个模块只加载一次， 每一个JS只执行一次， 如果下次再去加载同目录下同文件，直接从内存中读取。
- 每一个模块内声明的变量都是局部变量， 不会污染全局作用域；
- 使用*可以实现整体导入

```
import * as config from "./config.js";
```