如何理解 ES6 模块化

ES6 模块化是指将一个大的程序文件拆分成许多小的文件，然后将小文件组合起来的过程。在 ES6 模块化中，每个 js 文件都是一个独立的模块，导入模块用import 关键字，导出用 export 关键字。

其设计思想是尽量的静态化，使得编译时就能确定模块的依赖关系及输入和输出的变量。CommonJS 和 AMD 模块，都只能在运行时确定这些东西。比如，CommonJS 模块就是对象，输入时必须查找对象属性。

```
// CommonJS模块
let { stat, exists, readfile } = require('fs');

// 等同于
let _fs = require('fs');
let stat = _fs.stat;
let exists = _fs.exists;
let readfile = _fs.readfile;
```

上面代码的实质是整体加载 `fs` 模块（即加载 `fs` 的所有方法），生成一个对象（`_fs`），然后再从这个对象上面读取 3 个方法。这种加载称为“运行时加载”，因为只有运行时才能得到这个对象，导致完全没办法在编译时做“静态优化”。

ES6 模块不是对象，而是通过 `export` 命令显式指定输出的代码，再通过import命令输入。

```
// ES6模块
import { stat, exists, readFile } from 'fs';
```

上面代码的实质是从 `fs` 模块加载 3 个方法，其他方法不加载。这种加载称为“编译时加载”或者静态加载，即 ES6 可以在编译时就完成模块加载，效率要比 CommonJS 模块的加载方式高。

1. export

export 命令能够对外输出的就是三种接口：函数（Functions），类（Classes），var、let、const 声明的变量（Variables）。

2. import

`import` 命令接受一对大括号，里面指定要从其他模块导入的变量名。大括号里面的变量名，必须与被导入模块对外接口的名称相同。

`import` 命令输入的变量都是只读的，因为它的本质是输入接口。也就是说，不允许在加载模块的脚本里面，改写接口。

3. export default

`export default` 为模块指定默认输出。

```
// export-default.js
export default function () {
  console.log('foo');
}
```

其他模块加载该模块时，import 命令可以为该匿名函数指定任意名字。这时import 命令后面，不使用大括号。

```
// import-default.js
import customName from './export-default';
customName(); // 'foo'
```

4. import()

import() 函数，支持动态加载模块。

- （1）按需加载。

import()可以在需要的时候，再加载某个模块。

```
button.addEventListener('click', event => {
  import('./dialogBox.js')
  .then(dialogBox => {
    dialogBox.open();
  })
  .catch(error => {
    /* Error handling */
  })
});
```

上面代码中，import() 方法放在 click 事件的监听函数之中，只有用户点击了按钮，才会加载这个模块。

- （2）条件加载

import() 可以放在 if 代码块，根据不同的情况，加载不同的模块。

```
if (condition) {
  import('moduleA').then(...);
} else {
  import('moduleB').then(...);
}
```

上面代码中，如果满足条件，就加载模块 A，否则加载模块 B。

- （3）动态的模块路径

`import()` 允许模块路径动态生成。

```
import(f())
.then(...);
```

上面代码中，根据函数 f 的返回结果，加载不同的模块。