02. let 和 const 声明

学习要点：

1.let 声明

2.const 声明

本节课我们来开始学习 ES6 中的 let 和 const 两种声明方式。

一. let 声明

1. ES6 新增了一个新的变量声明：let，它和 var 变量声明非常的类似；
2. 首先创建一个块级区域，分别使用 let 和 var 声明一个变量；

```
//块级区域
{
    var value   = 10;
    let count   = 20;
}

console.log(value);     //10
console.log(count);     //引用错误
```

3. 上述例子直观表现为：1.var 出了块级区域有效；2.let 出了块级区域无效；
4. var 声明具有变量提升能力，不管在哪里声明，均视为作用域顶部声明；
5. let 声明不具备变量提升能力，离开区块的作用域后，则变量立刻失效；
6. 那么，哪种更好？let 更适合局部变量，非常容易掌控且不会导致凌乱；

7. 变量提升能力还带来一个区别，就是声明之前使用时，产生的结果不同；
8. 下面例子对比中，var 在后面声明，前面输出的值从逻辑上较为怪异；
9. undefined 表示变量声明了，只是没有赋值，按理说顺序要在前面；
10. 而 let 声明方式，就算在后面声明，前面的输出依然是引用错误；

```
console.log(value);     //undefined
var value;              //变量提升导致逻辑怪异

console.log(count);     //引用错误
let count;
```

11. 在一个区块内部，只要使用 let 声明，这个区域就形成了封闭的作用域；
12. 如果在 let 声明前使用变量，这段区域被称为“临时死区(或暂时性死区)”；

```
if (true) {
    //死区开始
    value = 20;
    console.log(value);
    //死区结束
    let value = 10;
```

```
    console.log(value);
}
```

13. “临时死区”简称：TDZ，这段区域使用 typeof 也会报错；
14. 一般情况下，typeof 来判断未声明的变量，只会输出 undefined；

```
console.log(typeof value);
let value;
```

15. var 声明可以重复声明同一个变量，后面会取代前一个变量；
16. let 声明不可以重复声明一个变量，会直接报错，就算其中一个是 var；

```
let value = 20;         //两个 let 报错，let 和 var 各以一个也报错
var value = 20;         //报错，更换顺序报错
```

17. 当然，如果一个在作用域外部，一个在作用域内部，则可以并存；

```
let value = 20;
{
    let value = 10;     //不建议相同，会乱的
}
```

18. 在循环中，var 和 let 的区别尤为明显，let 只在循环内部有效；
19. var 全局有效，导致后续再使用 i 会引起干扰，而 let 则不会；

```
for (let i = 0; i < 10; i++) {
    console.log(i);
}
console.log(i);         //var 声明，则 10；let 声明，报错
```

20. 如果在循环体内设置函数方法，体外输出 var 会得到不想要的值；

```
var list = [];

for (var i = 0; i < 10; i++) {          //把这里改成 let，则会得到想要的值
    list[i] = function () {
        console.log(i);
    }
}

list[5]();              //这里不管设置多少，结果都是 10
```

二. const 声明

1. const 声明的作用是：创建一个只读的常量，一旦声明不可改变；
2. 和 let 声明一样，const 声明的常量无法提升，也存在临时死区；
3. 和 let 不同的是，const 声明后必须立刻赋值，否则会报错；

```
const PI = 3.14;
console.log(PI);    //常量约定俗成大写
```