# 03. 块级作用域

学习要点：

1.全局作用域

2.块级作用域

本节课我们来开始学习全局作用域和块级作用域的问题。

## 一. 全局作用域

1. 浏览器环境有一个顶层对象 window，其属性和 var 的全局变量等价；
2. 如果是 window 对象的内置属性，则输出默认值，而非内置则 undefined；
3. var 如果设置了 window 对象的内置属性作为变量，则会直接覆盖；

```
console.log(window.name);       //空，内置属性
console.log(window.abcd);       //undefined，非内置

console.log(name);              //空，内置全局
console.log(abcd);              //报错

console.log(window.name === name);          //true，等价

var value = 10;
console.log(window.value);      //var 的全局变量即 window 的属性

var name = 'Mr.Lee';
console.log(window.name);       //内置属性值被覆盖
```

4. 从上面的例子可以看出，对比现在模块化编程的理念，显得格格不入；
5. 从 WebStorm 环境中根本无法识别 window，因为它是 Node 环境，不支持对象；
6. 而 Node 环境的顶层对象 global，只不过其它环境均不支持，不深入展开；

## 二. 块级作用域

1. ES6 之前只有全局作用域和函数作用域，并没有所谓的块级作用域；
2. 上一节课的循环体和条件体就是块级作用域，也就是两个花括号区域：{}；
3. 如果在块级区域不使用 let，就会造成全局变量污染的问题；
4. {{{...}}}块级作用域支持多层嵌套，每一层均为封闭的，只作用于此；

```
{{{{
    {let value = 10;}
    console.log(value);         //报错
}}}}
```

5. 在 ES6 之前，采用自我立即执行匿名函数的方式来防止变量污染；

```
(function () {
    var value = 10;
```

```
}());

console.log(value);        //报错
```

6. ES6 之前函数必须在顶层声明，但违反并不报错，而 ES6 则开始支持；
7. 只不过块级作用域内的函数声明，还是可以在全局可访问，并没有封闭；

```
{
    function fn() {
        console.log('块级函数');
    }
}
fn();          //正常访问
```

8. 那么此时，推荐使用函数表达式的方式去构建函数即可；

```
{
    let fn = function() {
        console.log('块级函数');
    };
    fn();
}
```