

一、案例练习

二、函数

概念：可以实现某个特定功能的代码块。【可重复利用的代码块】

两个步骤：

第一、定义函数

第二、调用函数

```
1 // 定义函数的三种方式：
2 // 第一种：关键字 function 声明的函数 可以在声明之前调用 （预先解析）
3 // function 函数名称() { }
4 // 函数名称： 在符合web规范的前提下 自定义
5 // 2.0 调用函数
6 // foo1()
7 //
8 // 1.0 定义函数
9 function foo1() {
10     console.log("hello foo1 ");
11 }
12 // 2.0 调用函数
13 foo1();
14
15 // 第二种：直接量
16 // var 关键字
17 // 声明变量接收函数体
18 // foo2 undefined
19 // foo2(); // foo2 is not a function
20 // 如果是第二种方式，需要先声明后调用，否则报错
21 // 1.0 定义函数
22 var foo2 = function(){
23     console.log("hello foo2")
24 }
25 // 2.0 调用函数
26 foo2();
27
28
29 // 第三种：构造函数的方式
```

```

30 // new 操作符
31 // 通过new的方式调用的函数，叫做构造函数
32 // 创建函数的实例 foo3
33 // 1.0 定义函数
34 var foo3 = new Function("console.log('hello foo3')");
35 // 2.0 调用函数
36 foo3();
37
38
39
40 // 总结：
41     // 有三种方式定义函数，推荐第一种
42     // 其次，第二种
43     // 最后，第三种
44
45 重点：函数不调用，不执行。

```

作用域

```

1 // 挂载在window作用域的变量
2 // window 是一个全局对象
3
4 // 世界 > 国家 > 省份 > 城市 > 街道 ...
5 // 世界 > 国家
6 // window作用域 > 函数作用域
7
8
9 // 全局变量
10 var count = 6;
11 //console.log(window);
12 console.log(window.count);// 6
13
14 // 函数 demo
15 // 1.0 定义函数
16 function demo() { // 函数作用域1
17     // 局部变量(私有变量)
18     var index = 0;
19     console.log("demo index:",index)
20     //因为全局环境下，已经存在count变量，所以不能重复声明
21     //重复声明，变量冲突（变量污染）
22     // var count = 99;
23     console.log(count);// 6

```

```

24  }
25  // 2.0 调用函数
26  demo();
27
28
29  // 函数 foo
30  // 1.0 定义函数
31  function foo() { // 函数作用域2
32      // 局部变量
33      var index = 99;
34      console.log("foo index:",index)
35
36      console.log(count);// 6
37  }
38  // 2.0 调用函数
39  foo()
40
41
42
43
44  // 作用域:
45  // 两种:
46  // a. 全局作用域 window
47  // b. 函数作用域 function f(){ 代码可执行范围 }
48  // 函数作用域:代码块的范围
49
50
51  // 变量
52  // a. 在全局作用域声明的变量 叫做全局变量
53  // b. 在函数作用域声明的变量 叫做局部变量

```

函数传递参数

```

1  // 1.0 定义函数
2  function foo(a){//形参 （形式参数）
3      // 形参仅仅在作用域中使用
4      console.log(a);// undefined
5  }
6  // 2.0 调用函数
7  foo(99);//实参（值）
8  foo("hello");//实参（值）

```

```
9  foo(true);//实参（值）
10 foo();// 没有实参
11
12
13
14 // ===== 思考 =====
15 function add(a,b){
16     // a  === 100
17     // b  === 99
18     console.log(a+b);
19 }
20 add(100,99);
21 // 实参和形参的顺序要一一对应
```