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

```
// 定义函数的三种方式：
// 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）
// function 函数名称() { }
// 函数名称 ：  在符合web规范的前提下 自定义
// 2.0 调用函数
// foo1()
//
// 1.0 定义函数
function foo1() {
    console.log("hello foo1 ");
}
// 2.0 调用函数
foo1();

// 第二种：直接量
// var 关键字
// 声明变量接收函数体
// foo2  undefined
// foo2(); // foo2 is not a function
// 如果是第二种方式，需要先声明后调用，否则报错
// 1.0 定义函数
var foo2 = function(){
    console.log("hello foo2")
}
// 2.0 调用函数
foo2();


// 第三种：构造函数的方式
```

```
// new  操作符
// 通过new的方式调用的函数，叫做构造函数
// 创建函数的实例 foo3
// 1.0 定义函数
var foo3 = new Function("console.log('hello foo3')");
// 2.0 调用函数
foo3();



// 总结：
    // 有三种方式定义函数，推荐第一种
    // 其次，第二种
    // 最后，第三种

 重点：函数不调用，不执行。
```

作用域

```
// 挂载在window作用域的变量
// window 是一个全局对象

// 世界   >  国家       >       省份 > 城市 > 街道 ...
// 世界   >  国家
// window作用域 >  函数作用域


// 全局变量
var count = 6;
//console.log(window);
console.log(window.count);// 6

// 函数 demo
// 1.0 定义函数
function demo() { // 函数作用域1
    // 局部变量(私有变量)
    var index = 0;
    console.log("demo  index:",index)
    //因为全局环境下，已经存在count变量，所以不能重复声明
    //重复声明，变量冲突（变量污染）
    // var count = 99;
    console.log(count);// 6
```

```
}
// 2.0 调用函数
demo();


// 函数 foo
// 1.0 定义函数
function foo() { // 函数作用域2
    // 局部变量
    var index = 99;
    console.log("foo  index:",index)

    console.log(count);// 6
}
// 2.0 调用函数
foo()




// 作用域:
// 两种:
// a. 全局作用域  window
// b. 函数作用域  function f(){ 代码可执行范围 }
// 函数作用域:代码块的范围


// 变量
// a. 在全局作用域声明的变量 叫做全局变量
// b. 在函数作用域声明的变量 叫做局部变量
```

函数传递参数

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

```
foo(true);//实参（值）
 foo();// 没有实参



// ========= 思考 =========
function add(a,b){
    // a  === 100
    // b  === 99
    console.log(a+b);
}
add(100,99);
// 实参和形参的顺序要一一对应
```