DAY04 函数

2023年4月21日 16:29

函数(函数对象)

1.函数的意义 - 非常重要

1.1.函数是ES里面的"一等公民",函数的地位非常高。

1.2.函数是ES的底层。

1.3.函数式编程是当前开发的主流(区分面向对象)。

1.4.类也是函数，ES里面类也叫做构造函数

1.5.方法也是函数实现的

2.函数的概念

对于ES来说，函数就是把任意一段代码放在一个 "盒子" 里面

给这个盒子取一个名字

盒子里面的代码具有一定的功能

通过名字使用盒子里面的代码

优势：封装一起，重复使用，非常灵活，方便扩展.

函数的声明和调用

1.函数的声明(定义函数)

1.1.声明式函数的定义：有名称的函数

```
// function fnName() {
//    函数体
// }
```

function：声明函数的关键字，表示接下来是一个函数了

fnName：函数的名字，我们自己定义的（遵循变量名的命名规则和命名规范）

()：必须写，是用来放参数的位置（一会我们再聊）

{}：就是我们用来放一段代码的位置（也就是我们刚才说的 "盒子"）

```
// fn1();//可以使用
// function fn1() {
//    console.log('你好，我是声明式函数里面的代码');
// }
// fn1();//可以使用
```

1.2.赋值式函数的定义：没有函数名称(匿名函数)

```
// fn2();//变量提升有关，变量提升值为undefined，fn2() <===> undefined() 报错
// var fn2 = function fn3() {
```

```
//    console.log('你好，我是赋值式函数里面的代码');
// }
// fn2();// 可以使用
// fn3();// 报错，赋值式函数不能再添加函数名称
```

2.函数的调用(使用函数)

函数调用就是直接写 "函数名()" 就可以了

注意：函数不调用，可以当作函数根本不存在。

```
// fn1();
// fn2();
```

3.对比声明式函数和赋值式函数写法的区别

3.1.声明式函数可以任意位置(前面和后面)调用，赋值式函数只能下面调用。

3.2.赋值式函数不能再添加函数名称

3.3.两种方式当前都可以随意使用

参数的声明

一.参数的声明

我们在定义函数和调用函数的时候都出现过 ()，括号就是用来放参数的

参数分为形参和实参

1.形参：出现在函数声明中，在整个函数体内都可以使用

2.实参：出现在函数调用中，实参将值传递给形参，实参是具体的值。

二. 重点关注：形参类似于定义在函数内部的变量，只是形式上存在的，即声明了没赋值

形参的值必须来自实参，否则就类似于声明了变量没有赋值，就是undefined.

```
// function sum(a, b) {//类似于var a , b;
//    console.log(a + b);//undefined + undefined  = NaN
// }
// sum();
```

三.函数不介意传递进来多少参数，也不会因为参数不统一而错误。

```
// function sum(a, b, c, d) {
//    console.log(a + b + c + d);
// }
// sum(1, 2, 3, 4, 5);//1 + 2 + 3 + 4 = 10
// sum(1, 2, 3);//1 + 2 + 3 + undefined = NaN
```

```
sum(13, 14);//27

// 声明函数   a,b是形参
function sum(a, b) {// a,b相当于函数内部声明的两个变量var a,b
  console.log(a + b);
}
                实参传递给形参的顺序
// 调用函数
sum(3, 4);//7  3,4是实参,实参传递给形参
sum(30, 40);//70
```

四.arguments:函数内部天生具有特殊对象。

arguments对象来接收传递进来的参数，

并且arguments对象具有length属性和非负整数的下标。

1.length:获取传入的参数长度

2.非负整数的下标

给每个传入的参数进行自动编号，编号是非负整数，从0开始。

```
function sum() {
  console.log(arguments.length);
  // console.log(arguments[0]);
  for(var i =0;i<arguments.length;i++){
    console.log(arguments[i]);
  }
}
sum(1,2,3,4,4,5);
```

函数的默认值

函数允许添加默认的参数值

如果你没有传递实参，使用默认值，如果传递了实参继续使用你的传递的值

1.利用逻辑运算符解决默认值 - 早期的方式

```
// function sum(a, b) {//参数类似于var a,b;
//   a = a || 5; //如果右边a没有值，默认就是undefined,undefined || 5 => 5   如果右边的a有值为20   20 || 5 => 20
//   b = b || 5;
//   console.log(a + b);
// }
// sum();//5+5
// sum(20, 20);//20+20
```

2.直接在形参后面赋值 - 新的方式

```
// function sum(a = 5, b = 5) {
//   console.log(a + b);
// }
// sum();//5+5
// sum(20, 20);//20+20
```

函数的返回值以及和对象的关系

一.return关键字

return 返回的意思，其实就是给函数一个 返回值 和 终断函数

return 语句导致函数停止执行，并返回return后面表达式的值给调用者。

1.返回值

2.打断函数

函数里面的return之后的代码就不再执行，函数碰到return就结束。

```
// function fn() {
//    console.log(1);
//    return
//    console.log(2);//不会执行
//    console.log(3);//不会执行
// }
// fn()
```

3.函数默认都有一条return语句，如果没有显式声明，则整个函数默认返回undefined。

```
// function fn() {
//    console.log(1);
// }
// console.log(fn());//undefined
```

二.完整的函数组成

function

函数名称

参数

函数体(return返回值)

注意：上面的组成部分不是每一项都是必须的.

函数属于对象，同时也是对象的构造器。

函数的好处及注意事项

1.好处：函数是用来帮助我们封装、重用、扩展及调用代码的最方便的工具。

封装：独立的功能拿出来，打包成一个函数

重用：反复使用，多次调用

扩展：继续扩展功能，灵活的参数配置

2.注意实现

2.1.函数名等于函数体

```
// function fn() {
//    return '我是普通函数的返回值'
// }
// console.log(fn); //没有调用，仅仅输出函数名    function fn() {return '我是普通函数的返回值'}
```

2.2.typeof 函数，返回function

```
// function fn() {
//    return '我是普通函数的返回值'
// }
// console.log(typeof fn); // function
```

2.3.声明的函数如果不调用，函数相当于不存在，通过调用找函数。

了解预解析

一.为什么了解预解析

解答两个疑问

1.为什么变量会提升，变量在声明之前就可以使用，但是值是undefined

2.为什么声明式函数可以任何地方调用。

二.JS的预解析过程分为两个阶段：预编译期与执行期

1.第一阶段(预编译期)：代码进入浏览器逐行执行之前干的事情(不可见)

关注两个问题：变量和声明式函数

预编译是ES天生具有的特点

1.1.在程序里面找到var关键字，找到提前赋值undefined给变量名. 找到有名称function，提前将整个函数赋值给函数名称

1.2.如果预编译时函数和变量出现重名，函数优先

2.第二阶段(执行期)：在编译后的基础上开始通过浏览器从上到下执行代码

遇到错误时中断代码执行

同时遇到函数声明直接跳过，函数必须调用才有意义。