## 匿名函数

var f = function(){

console.log('这是一个匿名函数')

}

var f1 = function(book){

console.log("参数bool："+book)

}

## 回掉函数

function add(a,b){

console.log(a()+b());

}

function a(){

return 5;

}

function b(){

return 15;

}

var a1 = function(){

return 55;

}

add(a,b);//20

add(a1,b);//70

function totalPrice(price1,price2,price3,callback){

arr = [];

for(var i = 0; i < 3; i++){

arr[i] = callback(arguments[i]);

}

return arr

}

function add1(num){

return num\*2;

}

console.log(totalPrice(100,200,300,add1));

## 自执行函数

(function(){console.log(1)}())

(function(){console.log(1)})()

## 内部函数

function outer(param){

function inner(num){

return num \* 2;

}

return '结果:'+inner(param);

}

console.log(outer(10))

//inner将只能在outer函数内部被访问

## 返回函数的函数

function a(){

console.log('a');

return function(){

console.log('b')

}

}

b = a();

b()

## 重写自己的函数

function a(){

console.log('a1');

a = function(){

console.log('b1')

}

}

a();

a()

a()

var a = (function(){

var setp = 0;

function aa(){

var setop = 'done';

}

function bb(){

console.log('word')

console.log('bb:'+(setp++))

}

aa();

return bb

}())

console.log(a)

a();

a();

a()

## 闭包

## getter与setter

var getValue,setValue;

(function(){

var secret = 0;

getValue = function(){return secret}

setValue = function(v){if(typeof v === 'number'){secret = v}}

}())

console.log(getValue());

setValue(10)

console.log(getValue());

setValue('20')

console.log(getValue());

## 迭代器

function setup(x){

var i = 0;

var len =x.length

return function(){

// console.log('i:'+i)

// console.log('len:'+len)

if(i < len){

return x[i++];

}else{

return 'eof';

}

}

}

var next = setup(['a','b','c','d']);

console.log(next())

console.log(next())

console.log(next())

console.log(next())

console.log(next())

## 高阶函数

## 函数表达式