## 函数的概念

### 1.1函数基本使用

什么是函数：具有某种特定功能的代码块

为什么要有函数

1、函数可以解决代码复用问题

2、函数可以把整个项目模块化

前面的图形案例引入(四边形)

### 1.2函数声明

function functionName(parameters) {

要执行的代码

}

函数表达式

var x = function (a, b) {return a \* b};

上面的函数实际上是一个匿名函数

**匿名函数：**就是没有名字的函数

匿名函数通常与自执行函数结合使用，因为匿名函数没有函数名，没办法调用，通过自执行调用

匿名函数的基本形式为

(function(){...})();

(function(){...}());

函数定义（字面量定义和表达式定义 先定义后使用 三要素\*\*\*）

//函数定义或者调用都要考虑函数的三要素：功能 参数 返回值；

//功能：一般说的就是函数能做什么 通常情况下函数名就代表着函数的功能

// 侧面告诉我们函数的名字要见名思意；

//参数：函数的小括号当中，写的是参数，函数定义的参数，我们被称作形参（形式参数）;

// 函数调用的时候,小括号里面也写的是参数,这些参数我们称作实参(实际参数);

// 函数调用的实参本质上是在给函数定义的形参 进行赋值(把实参的值赋值给形参，通常叫传参)；

注意：//形参写出来就相当于在函数内部 var了一个变量

//返回值： 每一个函数都会有返回值，有的函数返回值比较明确，写了return,但是有的函数

//没有写return，并不是代表这个函数没有返回值，而是省略了return，实际上返回值是undefined

### 1.3函数参数

参数默认

如果调用参数时省略了参数（少于被声明的数量），则丢失的值被设置为：undefined。

arguments 对象

JavaScript 函数有一个名为 arguments 对象的内置对象。

arguments 对象包含函数调用时使用的参数数组。

当不清楚要有多少个参数时才使用！

这样，就可以简单地使用函数来查找（例如）数字列表中的最高值：

x = findMax(1, 123, 500, 115, 44, 88);

function findMax() {

var i;

var max = -Infinity;

for (i = 0; i < arguments.length; i++) {

if (arguments[i] > max) {

max = arguments[i];

}

}

return max;

}

### 1.3函数调用

在函数被定义时，函数内部的代码不会执行。

在函数被调用时，函数内部的代码会被执行。

//函数调用也被函数调用表达式；既然函数调用是一个表达式，那么函数调用就是有值的

//函数返回值就是函数调用表达式的值；

## 函数的分类

1) 无参无返回值

2) 有参无返回值

3) 无参有返回值

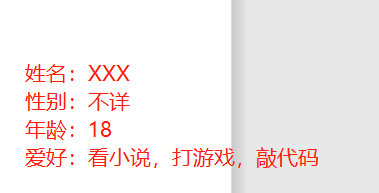
4) 有参有返回值

## 函数的封装

1.编写求和函数

2.编写函数求数组的最大值，最小值

3.编写函数打印个人简介 (姓名、性别、年龄、爱好)



4.封装函数加工数组，每一项加10输出

5.根据用户输入的数值求1到该数值的累加和

6.封装函数实现排序

## 4.作用域

### 4.1.作用域概念

作用域指的是您有权访问的变量集合。

### 4.2.在 JavaScript 中有两种作用域类型：

局部作用域

全局作用域

JavaScript 拥有函数作用域：每个函数创建一个新的作用域。

作用域决定了这些变量的可访问性（可见性）。

函数内部定义的变量从函数外部是不可访问的（不可见的）。

**作用域链**： 一般情况下，变量取值到 创建 这个变量 的函数的作用域中取值。

但是如果在当前作用域中没有查到值，就会向上级作用域去查，直到查到全局作用域，这么一个查找过程形成的链条就叫做作用域链。

### 4.3.局部 JavaScript 变量

在 JavaScript 函数中声明的变量，会成为函数的局部变量。

局部变量的作用域是局部的：只能在函数内部访问它们。

实例

// 此处的代码不能使用 carName 变量

function myFunction() {

var carName = "porsche";

// 此处的代码能使用 carName 变量

}

由于只能在函数内部识别局部变量，因此能够在不同函数中使用同名变量。

在函数开始时会创建局部变量，在函数完成时会删除它们。

### 4.4.全局 JavaScript 变量

函数之外声明的变量，会成为全局变量。

全局变量的作用域是全局的：网页的所有脚本和函数都能够访问它。

实例

var carName = " porsche";

// 此处的代码能够使用 carName 变量

function myFunction() {

// 此处的代码也能够使用 carName 变量

}

### 4.5自动全局

实例

myFunction();

// 此处的代码能够使用 carName 变量

function myFunction() {

carName = "porsche";

}

### 4.6面试题

var num = 10;

function fun() {

var num = 20;

fun2();

}

function fun2() {

console.log(num);

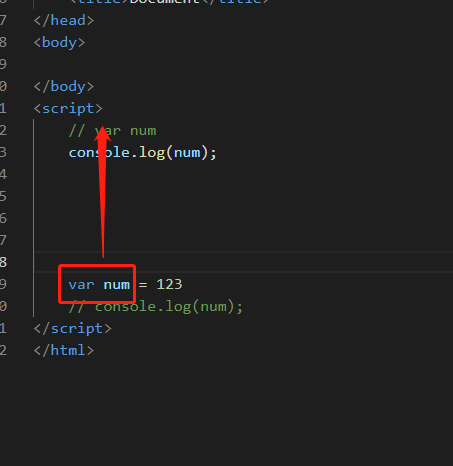
}

fun();

## 5.预解析

### 5.1概念

a) 预解析演示

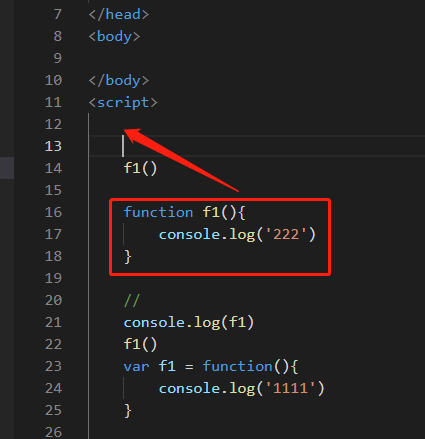
 变量 带var和不带var: 预解析只会解析带var的变量，如果不带var，不进行预解析；



函数

function f1(){} 如果是这种写法；函数整体会提升

var f1 = function(){} 如果是这种写法；只会提升var f1; 不会提升函数的表达式；



b) 预解析效果

 全局 （函数）

带var 的都会被提升到最上面执行，函数内部遵循同样的道理，提升到函数的最上方执行

**字面量作为一种通用的，跨平台的数据交换格式。表达式的右边就是字面量。**

**字面量(string literal )包括4种：字符串字面量(string literal)、数组字面量(array literal)和对象字面量(object literal)，函数字面量(function literal)**

**函数字面量和函数关键字声明的函数的区别**

**函数字面量由4部分组成。**

**第一部分，关键词 function**

**第二部分，函数名，但是可有可无。**

**第三部分，包含在括号内的参数，当然参数也是可有可无的，括号不能少。**

**第四部分，是一组包裹在大括号的语句块，也就是函数要执行的具体代码，当然不写代码也没问题，{}是必须要的。**

### 5.2面试题:

--------------------------------------------------------

alert(a);

a = 0;

报 错 ：没有var 定义 所以没有被预解析

--------------------------------------------------------

alert(a);

var a = 0;

alert(a);

Undefined

0

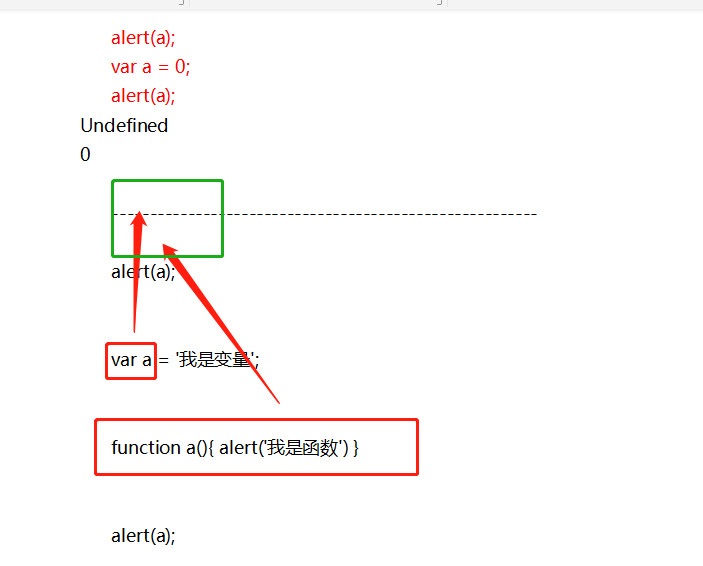
--------------------------------------------------------

alert(a);

var a = '我是变量';

function a(){ alert('我是函数') }

alert(a);



function a(){ alert('我是函数') }

我是变量

--------------------------------------------------------

alert(a);

a++;

alert(a);

var a = '我是变量';

function a(){ alert('我是函数') }

alert(a)

function a(){ alert('我是函数') }

NaN

我是变量

--------------------------------------------------------

alert(a);

var a = 0;

alert(a);

function fn(){

alert(a);

var a = 1;

alert(a);

}

fn()

alert(a);

--------------------------------------------------------

alert(a);

var a = 0;

alert(a);

function fn(){

alert(a);

a = 1;

alert(a);

}

fn()

alert(a);

--------------------------------------------------------

## 回调函数

一个函数被作为参数传递给另一个函数（在这里我们把另一个函数叫做“otherFunction”），回调函数在otherFunction中被调用。

回调函数定义：如果一个函数被当作参数传给另外一个函数，那么这个函数就被称作回调函数，虽然概念是这么定义的，但是真正的回调函数还要符合下面

几个特点；如果不符合，我们不能称作是真正回调；

回调函数：函数是我定义的 我没有调用 最终执行了~

function fn1(a,fn){

        console.log(a)

        var num = 1234

        // fn2()

        fn(num)

    }

    function fn2(num) {

        console.log(num)

    }

    fn1(1,fn2)

晚自习强化练习

作业:

面试题: 函数，作用域，预解析（必须理解）

封装函数：

数组去重

合并数组，原生实现

