# 03-JS语法01-函数

## 1 简介

函数是由事件驱动的或者当它被调用时执行的可重复执行的代码块。

### 1.1 函数声明

函数和变量一样，需要声明：

function fun(){  
 alert("我是自定义函数")  
}  
fun(); // 函数不调用，自己不执行

声明方式一：直接量声明

var fun1 = function(){  
 alert("直接量声明")  
}  
fun1(); 也需要调用

声明方式二：Function关键字声明

var fun2 = new **Function**("var a = 10; var b = 20; alert(a+b)");  
fun2();

声明方式三：函数表达式（即自定义）

var myFun = function (a,b){

return a + b;

}

### 1.2 变量与作用域

##### 1.2.1 变量分类

全局变量：又称成员变量，在哪里都可访问到，

如进入script立即定义的变量和没有var的变量。

局部变量：函数内部的变量，只有函数内部可以访问到，

如函数内部用var定义的变量和形参。

##### 1.2.2 隐式全局变量

隐式全局变量就是隐藏的全局变量不好被发现。

function fn（）{

var a = b = c = 1; // b和c就是隐式全局变量

}

注意:

function fn（）{

var a = b = c = 1; // b和c就是隐式全局变量（等号）

var a = 1; b = 2; c = 3; // b和c就是隐式全局变量（分号）

##### 1.2.3 变量提升与预解析

在函数体内部声明变量，会把该声明提升到函数体的顶端，但是只提升声明、不赋值。

出现变量提升的原因是JS的与解析行为。

function fn(){

console.log(num);

var num = 1;

}

fn();

这时候输出 undefined，这样的写法相当于：

function fn() {

var num;

console.log(num);

num = 3;

}

变量提升：在预解析的时候，成员变量和函数被提升到最高位置，方便其他程序访问。

变量提升特点：成员变量只提升变量名，不提升变量值，但是函数会提升素有内容。

当使用的变量在定义变量之前时，很容易出现变量提升。

### 1.2 参数arguments

function fn(a,b) {

console.log(a+b)

}

fn(1,2);

形参：形式上参与运算的变量，无实际值，为实参占位置；

实参：实际参与运算的变量。形参为他占位置，真实参与运算的变量。

arguments存储的是传递过来的实参，JS在创建函数的同时，会在函数内部创建一个arguments对象实例，arguments对象只有函数开始时才可用。

arguments对象并不是一个数组，访问单个参数的方式与访问数组元素的方式相同。

arguments对象的长度由实参个数决定，而不是由形参个数绝对。

function fn(a,b){

console.log(fn.length); //得到是 函数的形参的个数

//console.log(arguments);

console.log(arguments.length); // 得到的是实参的个数

if(fn.length == arguments.length){

console.log(a+b);

}else{

console.error("对不起，您的参数不匹配，正确的参数个数为:" + fn.length);

}

//console.log(a+b);

}

fn(1,2);

fn(1,2,3);

### 1.3 定义变量写法

写法一：var a = 10, b = 20;

写法二：var a = 10;

var b = 20;

### 1.4 属性与方法

方法和属性的区别：

方法一律带有小括号 。 isNaN( ); 动词

方法给值： isNaN（“值”）;

var index = 10; //变量 谁都可以使用  
var arr = []; // 数组  
arr.**index** = 20; // 自定义属性 他只能在arr 才能使用  
alert(arr.**index**);

### 1.5 return返回值

在函数内部用return来返回计算结果，一个函数只能返回一个值，同时会终止代码的执行。

如果函数没有显式的使return语句 ，那么函数有默认的返回值：undefined；

如果函数使用return，但return后面没有任何值，函数返回值也是：undefined；

### 1.6 函数加载问题

JS加载的时候，只加载函数名，不加载函数体。所以如果想使用内部的成员变量，需要调用函数。

### 1.7 匿名函数

匿名函数就是没有名字的函数，书写相对简洁。

匿名函数的调用有三种方法：

1. 直接调用或自调用。(function(){alert(1)})()
2. 事件绑定。
3. 定时器。

### 1.8 入口函数

window.onload = function(){

内部放js

}

这个函数的意思就是说，当我们页面加载完毕之后，采取执行函数体里面的js部分。 就是说等 页面的结构 样式 节点等加载完毕。。。

所以，这句话也可以页面的顶端即可。

<script>  
 **window**.onload = function(){  
 /\*要做事，先找人\*/  
 var pic1 = **document**.getElementById("pic1");  
 var pic2 = **document**.getElementById("pic2");  
 var pic3 = **document**.getElementById("pic3");  
 pic1.onclick = function(){  
 **document**.**body**.**style**.**backgroundImage** = "url(images/1.jpg)";  
 }  
 pic2.onclick = function(){  
 **document**.**body**.**style**.**backgroundImage** = "url(images/2.jpg)";  
 }  
 pic3.onclick = function(){  
 **document**.**body**.**style**.**backgroundImage** = "url(images/3.jpg)";  
 }  
 }  
</script>

## 2 常见用法

### 2.1 事件-获取/失去焦点

常见的函数事件有onclick，onmouseover，onmouseout等，当然还有焦点的获取失去。

获得焦点: onfocus

失去焦点: onblur

var txt = document.getElementById("txt");

txt.onfocus = function(){

if(txt.value == "请输入..."){

txt.value = "";

txt.style.color = "#333";

}

}

txt.onblur = function(){ //失去焦点

if(txt.value == ""){

txt.value = "请输入...";

txt.style.color = "#ccc";

}

}

### 2.2 this innerHTML isNaN

this 主要指事件的调用者本身。

innerHtml 会替代盒子内的标签、文字

isNaN 判断是否是数字

### 2.3 表单自动获取焦点

onfocus是事件，而focus()是JS提供的方法。

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title></title>  
 <script>  
 **window**.onload = function(){  
 var txt = **document**.getElementById("txt");  
 txt.focus(); //自动获得焦点  
 }  
 </script>  
</head>  
<body>  
自动获得焦点:  
<input type="text" id="txt"/>  
</body>  
</html>

### 2.4 鼠标经过选择表单

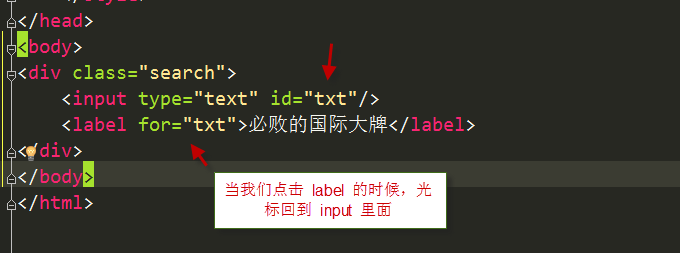
sele.onmouseover = function(){  
 this.select(); //选择  
}



### 2.5 判断用户输入事件

正常浏览器 : oninput

Ie 678 支持的 ： onpropertychange



## 3 数组

数组就是一个大变量，可以存储很多东西。

### 3.1 数组定义

声明：

//声明方式一

var arr1 = [];

//声明方式二

var arr2 = new Array();

赋值：

var arr1 = [1,3,5,7,9];

var arr2 = new Array(1,3,5);

### 3.2 数组使用

使用的方法: 数组名[索引值]; 函数名();

索引号是从0开始的。 0 1 2 3 4 5 ...

var textArr = ["刘备","诸葛亮","赵云","关羽"];  
**console**.log(textArr[3])

通过length属性可以获取数组的长度：数组名.length;

### 3.3 数组案例：隔行变色

var lis = **document**.getElementsByTagName("li"); // 得到所有的li  
 //alert(lis.length);  
 for(var i = 0; i<lis.**length**; i++){  
 if(i%2 == 0) // 只有偶数能被2整除  
 {  
 lis[i].**style**.**backgroundColor** = "#eee";  
 }  
 //鼠标经过li 的时候， 当前的底色变亮  
 lis[i].onmouseover = function(){  
 this.**className** = "current";  
 }  
 lis[i].onmouseout = function(){  
 this.**className** = "";  
 }  
 }  
 }

## 4 数组高级API

### 4.1 判断数组

Instanceof A Instanceof B

Instanceof是一个关键字，可以用来判断 A 是否属于 B，结果为布尔； Array.isArray(变量);

Html5中新增的API，判断是否是数组，结果为布尔；

### 4.2 转换数组

toString(); 字符串 = 数组.toString();

把数组转换成字符串，每一项用 , 分隔；

valueOf(); 用法同上

返回数组对象本身；

join();

变量可以有可以没有。不写默认用逗号分隔，无缝连接用空字符串。

var arr = [123,456,789];

console.log(arr.join("abc"));

输出的结果是：123abc456abc789

### 4.3 数组增删

push(); 数组1改后的长度 = 数组.push(元素1);

在数组最后面插入项，返回数组的长度;

unshift() 数组1改后的长度 = 数组.unshift(元素1);

在数组最前面插入项，返回数组的长度

pop(); 被删除的元素 = 数组.pop();

取出数组中的最后一项，返回最后一项

shift(); 被删除的元素 = 数组.shift();

取出数组中的第一个元素，返回最后一项

### 4.4 数组换位与排序

reverse(); 反转后的数组 = 数组.reverse();

翻转数组（原数组讲被反转，返回值也是被反转后的数组）

sort(); 从小到大排序后的数组 = 数组.sort(function(a,b){

return a-b;

});

给数组排序，返回排序后的数组。如何排序看参数。

无参：按照数组元素的首字符对应的Unicode编码值从小到大排列数组元素。

带参：必须为函数（回调函数--callback）。函数中带有两个参数，代表数组中的前后元素。如果计算后（a-b），返回值为负数，a排b前面。等于0不动。返回值为正数，a排b后面。

### 4.5 数组合并concat

使用concat函数合并数组后得到的是一个新数组。

var arr1 = [123,456,789];

var arr2 = ["ab","张三"];

var arr3 = arr1.concat(arr2);

### 4.6 清空数组

var array = [1,2,3,4,5,6];

array.splice(0,array.length); //删除数组中所有项目

array.length = 0; //length属性可以赋值，其它语言中length是只读

array = []; //推荐

### 4.7 其他方法

slice(); 新数组 = 数组1.slice(索引1，索引2);

从当前数组中截取一个新的数组，不影响原来的数组，参数start从0开始,end从1开始；

splice() 新数组 = 数组1.splice(起始索引，结束索引，替换内容);

删除或替换当前数组的某些项目，参数start,deleteCount,options(要替换的项目)；

indexOf()、lastIndexOf() 索引值 = 数组.indexOf/lastIndexOf(数组中的元素);

如果没找到返回-1；

迭代方法 不会修改原数组

every()、filter()、forEach()、map()、some()

数组/boolean/无 = 数组.every/filter/forEach/map/some(

function(element,index,arr){

程序和返回值；

}

);

1） 对数组中每一项运行以下函数，如果都返回true，every返回true，如果有一项返回false，则停止遍历 every返回false；不写默认返回false

array.every(function(item,index,arr) {

})

1. 对数组中每一项运行以下函数，该函数返回结果是true的项组成的新数组

var arr = array.filter(function(item,index,arr) {

});

1. 遍历数组

array.forEach(function(item,index,arr){

});

1. 对数组中每一项运行以下函数，返回该函数的结果组成的新数组

var arr = array.map(function(item,index,arr) {

return "\"" + item + "\"";

})

1. 对数组中每一项运行以下函数，如果该函数对某一项返回true，则some返回true

var b = array.some(function(item,index,arr) {

if (item == "ww") {

return true;

}

return false;

});

## 5 对象

### 5.1 空白对象

var obj = new Object();

### 5.2 构造函数

对象的实例是由构造函数来创建的。

构造函数与普通函数的区别：首字母要大写。

### 5.3 字面量

var obj = {

a:1

};

对象的字面量和json非常像，只是json的key必须加双引号

var json = {

“a”:1,

“b”:2

};

json只有一个键值对的时候可以没有逗号。

### 5.4 for in

var json = {“aaa”: 1,“bbb”: 2,“ccc”: 3,“ddd”: 4}

for(var key in json){

//key代表aaa,bbb.....等

//json[key]代表1,2,3....等

}

### 5.5 参数传值影响

1. 简单类型数据做参数，函数内部对参数的修改不应影响外部变量

简单类型传数值。

1. 复杂类型数据做参数，函数内部对参数的修改会应影响外部变量

复杂类型传地址。