# 函数

概述：在JavaScript脚本语言中也有函数概念，函数其实就是将某一个功能进行封装，当你事项用这个功能的时候，需要调用、执行这个函数

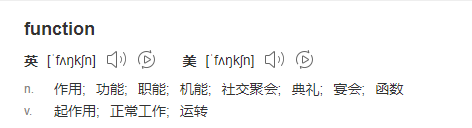
比如：我们在前面课程当中学习过一些内置函数

* 内置函数alert：这个功能是在浏览器正上方弹出一个警告框
* 内置函数prompt:这个功能是在浏览器正上方弹出一个提示框

比如他们使用的时候：函数名字+小括号

在javaScript脚本语言当中：函数使用也是分为两部分：声明+调用

注意：函数在JS当中声明方式有两种：关键字function声明函数、表达式声明函数



## 1.1关键字函数基本使用

概述：JS当中如果想声明关键字形式函数，需要用到关键字function进行声明。

基本语法：

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //关键字形式函数分为两部分声明+调用

    //声明部分(函数名字务必符合命名标识符规范)

    function fun() {

        //这里面是书写代码地方(函数体)

        console.log("鹅鹅鹅,去向向天歌");

        console.log("白毛浮绿水,红掌拨清波");

    }

    //函数调用部分

    fun();

    fun();

    fun();

</script>

* 函数声明一次可以多次使用。

## 1.2形参和实参

<script>

    //形参【形式参数】和实参【实际参数】

    //计算两个数字之和

    //a、b所谓形参

    function sum(a, b) {

        console.log(a + b);

    }

    //100、200实参

    sum(100, 200);

    sum("张三", "李四");

</script>

红色部分称之为实参：实际参数

橙色部分称之为形参：形式参数

当调用函数的时候：实参会一一对应进行赋值；

注意：在JS当中数学运算符中：加号比较特殊，如果加号遇见字符串【称之为连字符】不进行书写加法，

将数据从左到右进行字符串拼接；

## 1.3全局变量和局部变量\*\*\*\*\*

局部变量：

作为函数形参【只能在函数体中使用】、在函数体中声明变量【局部变量：只能在函数体中使用】

全局变量：

只要不是局部变量即为全局变量【全局变量可以在JS行为层中任意地方使用】

//下面代码为局部变量：只能在函数体中使用

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //局部变量：函数形参、在函数体中声明变量（只能在函数体中使用）

    //声明部分

    function fun(a, b) {

        //形参a、形参b称之为局部变量

        console.log(a);

        console.log(b);

        //在函数体中声明变量（局部变量）

        var c = "么么哒";

    }

    //调用

    fun(true, "我是祖国的老花骨朵");

    console.log(c);

</script>

### 1.3.1全局变量和局部变量区分

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //当前变量i全局变量（可以在JS行为层中任意地方使用）

    for (var i = 0; i < 10; i++) {

        console.log(i);

    }

    console.log("循环语句结束" + i);

    //当前变量str：既不是形参、也不是函数体中声明变量[全局变量]

    if (true) {

        var str = "我是全局还是局部呀";

    }

    console.log(str);

    //判断下面变量是全局还是局部

    var a = 100;

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    function fun(haha, hehe) {

        console.log(haha, hehe);

        //判断这个变量是全局还是局部

        for (var num = 1; num < 5; num++) {

            console.log(num);

        }

    }

    //调用函数

    fun(1, 2);

    // console.log(haha);

</script>

总结：

局部变量：函数形参、函数体中声明变量（只能在函数体中使用）

全局变量：全局变量可以在JS行为层中任意地方使用

## 1.4作用域

概述：作用域【scope】代码书写范围;

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //作用域书写代码范围

    //称之为全局作用域

    console.log(123);

    //函数也有作用域概念

    function fun(a, b) {

        console.log(a);

        console.log(b);

    }

    fun(5, 6);

</script>

问题：作用域是什么呀？

书写代码范围：

全局作用域：你可以理解为两个双闭合标签script之间

函数作用域：函数体

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //全局变量

    var b = 100;

    //声明一个函数

    function fun() {

        //局部变量

        var b = 200;

        console.log(b);

    }

    fun();

</script>

## 1.5JS当中没有函数重载概念\*\*\*\*\*\*

概述：在JS当中如果有多个重名的函数，永远是后者覆盖前者，

没有函数重载问题；

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    function fun(a) {

        console.log("我是第一个");

    }

    function fun(a, b) {

        console.log("我是第二个");

    }

    function fun(a, b, c) {

        console.log("我是第三个");

    }

    fun("我爱你祖国");

</script>

## 1.6关键字return

概述：return是JS当中关键字，经常结合函数一起使用，可以将函数计算完记过返回；

注意：return关键字只能在函数体中使用

1：在函数体中return关键字后面语句不在执行

2：可以将函数体中计算完结果返回，可以在函数体外面使用

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    function mul() {

        console.log(123);

        console.log(456);

        return;

        console.log(789);

    }

    mul();

</script>

* Return后面语句不在执行

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    function mul(a, b) {

        return a \* b;

    }

    var result = mul(6, 7);

    console.log(result);

</script>

* Return可以将函数体中计算完结果返回，在函数体中使用；

# 拓展canvas



## 2.1canvas简介

<https://developer.mozilla.org/zh-CN/docs/Web/API/Canvas_API> 学习手册

概述：canvas是2014年（超文本标记语言第五次重大修改中新增的）结构层中一个双闭合标签；

我们可以通过JS在行为层中操作画布，可以在canvas中绘制一些简单图形、对于图片进行裁切、

可以实现一些简单2D动画、显示视频；

### 2.1.1canvas基本使用

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

        }

        canvas {

            border: 1px solid black;

        }

    </style>

</head>

<body>

    <canvas width="600" height="400"></canvas>

</body>

</html>

<script>

    //获取canvas标签

    var canvas = document.querySelector("canvas");

    //画布当中任何操作必须通过2D上下文进行操作

    var ctx = canvas.getContext("2d");

    //一定要注意：画布当中任何操作都是通过ctx属性、方法进行操作

    //设置矩形填充颜色

    ctx.fillStyle = "cyan";

    //绘制矩形

    ctx.fillRect(100, 100, 100, 100);

    //绘制圆

    //通过绘制路径形式绘制圆

    ctx.beginPath();

    //用户在描述绘制图形

    ctx.arc(200, 200, 50, 0, Math.PI \* 2, false);

    //设置填充颜色

    ctx.fillStyle = "red";

    //开始填充

    ctx.fill();

    //边框设置

    ctx.lineWidth = 20;

    ctx.strokeStyle = "pink";

    //开始绘制

    ctx.stroke();

</script>

* canvas是HTML5中新增的一个双闭合标签【浏览器认为他是一张图片】
* Canvas标签是由默认w（300）、h（150）
* canvas标签w、h务必通过属性进行设置（别写样式设置）否则变形
* canvas标签文本、儿子标签都没有任何意义

Canvas最基本功能是绘制图形，需要注意画布任何操作都是通过2D上下文进行渲染；

[https://threejs.org/examples/#webgl\_camera](https://threejs.org/examples/" \l "webgl_camera) webGL学习手册

# 数组

array:数组

概述：在JS当中也有数组存在，数组你可以理解为是一个‘容器’，可以存储很多有序数据。

数组在JS当中是引用类型数据，在JS当中数组是用中括号表示

## 3.1数组基本使用

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //在JS当中数组使用[]进行表示

    //JS当中数组是引用类型数据

    console.log([]);

    console.log(typeof []);

    //数组目的：可以一次性存储很多有序数据

    console.log([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]);

</script>

* JS当中[]代表是数组，数组引用类型数据
* JS当中数组可以存储很多元素（数据：可以是任意类型属）

## 3.2数组经常存储、读取、修改、新增数据

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //在JS当中数组使用[]进行表示

    //JS当中数组是引用类型数据

    console.log([]);

    console.log(typeof []);

    //数组目的：可以一次性存储很多有序数据

    console.log([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]);

</script>

<script>

    //为了在JS当中使用数组方便

    //经常将右侧数组赋值给左侧变量（通过变量名字可以访问存储数组）

    //存储数据

    var arr = ["我爱你祖国", 12306, true, NaN, [2, 3, 4]];

    //为了读取数据:通过枚举法 + 下角标（索引值）获取数组里面存储数据

    console.log(arr[0]);

    console.log(arr[3]);

    console.log(arr[4]);

    //修改数组里面数据

    arr[0] = "我爱你母亲";

    arr[1] = 110;

    arr[4][0] = 66666;

    console.log(arr);

    //新增数据

    arr[5] = "我是后来的";

    arr[6] = "稍等我也来了";

    arr[9999] = "我是老嘎达";

    console.log(arr);

    console.log(arr[100]);

</script>

* JS当中数组：可以存储、可以读取、可以修改、可以新增数据

## 3.3length属性

概述：length是数组一个属性，主要的作用是可以获取数组元素总个数；

因此它返回的是一个数字；

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //length属性：可以获取数组里面元素总个数

    var arr = ["吃饭", "睡觉", "打豆豆", "喝酒", "烫头"];

    //数组里面一共是五个元素

    console.log(arr.length);

    //数组里面元素是从零开始的

    //数组的length属性常用语遍历数组

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

        //获取数组里面元素

        console.log(arr[i]);

    }

</script>

<script>

    //数组常见算法题：

    //比如：计算数组里面元素累加和问题

    var arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    var sum = 0;

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

        sum += arr[i];

    }

    //计算完毕获取结果

    console.log(sum);

</script>

<script>

    //获取数组里面最大元素

    var arr = [66, 3, 2, 99, 26, 21, 19, 88];

    //获取最大数字

    var max = arr[0];

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

        //后者元素大于前者进行重新赋值

        if (arr[i] > max) {

            max = arr[i];

        }

    }

    console.log(max);

</script>

## 3.4数组方法

<https://www.w3school.com.cn/jsref/jsref_obj_array.asp> 数组方法学习手册

### 3.4.1pop||push\*\*\*\*\*

概述：他们两者是数组方法，主要的作用是可以在数组尾处移除、添加元素。

单词少的：移除元素 单词多的：新增元素

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //声明数组

    var arr = ["幺鸡", "五万", "三饼", "二条", "二筒", "发财"];

    //pop:是数组的一个方法，主要的作用是可以在数组尾巴出移除 一项元素

    var result = arr.pop();

    var result1 = arr.pop();

    console.log(arr);

    console.log(result);

    console.log(result1);

</script>

* pop是数组方法，可以在数组尾出移除掉一项元素
* Pop方法有返回数值，返回的是移除掉的那项元素

    var result = arr.push("曹操");

    var result1 = arr.push("刘备", "关二爷", "张飞");

    console.log(arr);

    console.log(result);

    console.log(result1)

* push也是数组方法，主要的作用是可以向数组尾出添加一项、多项元素
* push方法执行完毕有返回数值，返回的数值是数组新增完元素总个数；

总结：

pop是数组方法，可以在数组尾巴出移除一项元素，返回的是删除调的那个元素

push是数组方法， 可以在数组尾巴出添加一个、多个元素，返回的是添加完元素总个数

### 3.4.2shift||unshift\*\*\*\*\*

概述：shift||unshift他们两者是数组方法，可以在数组头部添加、移除元素。

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //shift:在数组头部删除一项元素

    var arr = ["北京", "上海", "广州", "深圳"];

    //头部删除一项元素

    // var result = arr.shift();

    // console.log(arr);

    // console.log(result);

    //头部添加一个、多个元素

    var result = arr.unshift("杭州", "苏州", "小米粥");

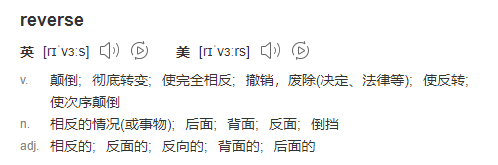
    console.log(arr);

    console.log(result);

</script>

* Shift在数组头部移除一项元素、返回删除掉的哪像元素
* unshift在数组头部添加一个、多个元素，返回的是数组元素总个数（新的总个数）

### 3.4.3reverse



概述：它也是数组方法，主要的作用是可以让数组里面元素进行倒置。

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //reverse:数组方法让数组元素进行倒置

    var arr = ["香港港独", "最近最好别吃香蕉", "最近广厦事件"];

     arr.reverse();

    console.log(arr);

</script>

### 3.4.4indexOf

概述：它也是数组方法，主要的作用是可以获取数组当中某一个元素索引值。

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //reverse:数组方法让数组元素进行倒置

    var arr = ["香港港独", 1, 2, 3, 4, 5, "香蕉", "香蕉", "香蕉", "最近广厦事件"];

    // arr.reverse();

    // console.log(arr);

    //indexOf：可以检测某一个元素索引值

    console.log(arr.indexOf("香蕉"));

    console.log(arr.indexOf("贾成豪"));

</script>

* indexOf是数组方法，可以获取数组当中某一个元素索引值
* 注意1：获取到的是从左到右第一个符合条件索引值
* 注意2：如果获取的是数组里面没有元素，返回索引值是-1

### 3.4.5join\*\*\*\*\*

概述：它也是数组方法，主要的作用是可以通过某一个字符将数组转换为字符串。

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //join:将数组转换为字符串

    var arr = ["北京", "南京", "燕京", "东京", "吴京", "亮晶晶"];

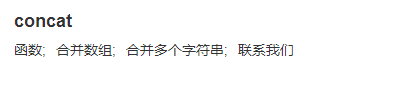
    console.log(arr.join());

    console.log(arr.join("\*"));

    console.log(arr.join(""));

</script>

### 3.4.6concat



概述：它也是数组方法，主要的做用时将多个数组合并为一个数组；

    //concat:将多个数组合并为一个数组

    var a = [1, 2, 3]；

    var b = [4, 5, 6];

    var c = [7, 8, 9];

    console.log(a.concat(b, c));

    console.log(a, b, c);

### 3.4.7slice

概述:它也是数组方法，主要的作用是可以切割数组；

语法格式：

注意：切割的时候，包含起始位置，但是不包含结束位置,

slice方法对于起始数组没有任何影响

arr.slice(起始索引值，结束索引值);

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //slice:数组方法用于切割数组

    //代表起始位置索引值，结束索引值 （包含起始位置，但是不包含结束为）

    var arr = ["小浣熊", "麻花", "包子", "烤鸭", "棒棒糖"];

    //一个参数：代表起始位置

    var newArr1 = arr.slice(1);

    var newArr2 = arr.slice(1, 3);

    var newArr3 = arr.slice(-2);

    console.log(newArr3);

</script>

### 3.4.8splice

概述：它也是数组方法，主要的作用是切割数组。

语法格式：

Arr.splice(起始位置，长度);

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    var arr = ["王者荣耀", "吃鸡", "lol", '魔兽', "传奇", "劲舞团", "CF", "DNF", "逆战"];

    //切割

    // var newArr1 = arr.splice(2, 3);

    // console.log(newArr1);

    // console.log(arr);

    //插入

    // arr.splice(2, 0, "哈哈哈插入数据", "么么哒");

    // console.log(arr);

    //替换

    arr.splice(2, 2, "哈哈哈插入数据", "么么哒");

    console.log(arr);

</script>