**JavaScript 的数据类型**

如下是js中常见的数据类型声明

// 字符串

var answer="It's alright";

var answer="He is called 'Johnny'";

var answer='He is called "Johnny"';

// 数值

var x1=34.00;      // 使用小数点来写

var x2=34;         // 不使用小数点来写

// 布尔

var x=true;

var y=false;

// undefined

var udf = undefined;

var udf2;

// null

var nul = null;

// 数组

var cars=["Saab","Volvo","BMW"];

// 对象

var person={firstname:"John", lastname:"Doe", id:5566};

// 日期

var date = new Date();

// Event 事件

var e = new Event(...);

**函数**

function myFunction(a, b) {

    return a + b;

}

**控制流程**

// if ... else

if (condition) {

    当条件为 true 时执行的代码

}

else {

    当条件不为 true 时执行的代码

}

// switch

switch (n) {

    case 1:

        执行代码块 1

        break;

    case 2:

        执行代码块 2

        break;

    default:

        n 与 case 1 和 case 2 不同时执行的代码

}

// for 循环

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

    document.write(cars[i] + "<br>");

}

var person = { fname: "John", lname: "Doe", age: 25 };

for (x in person) {

    txt = txt + person[x];

}

// while 循环

while (i < 5) {

    x = x + "The number is " + i + "<br>";

    i++;

}

// break 语句用于跳出循环。

// continue 用于跳过循环中的一个迭代。

**事件处理**

// 为元素添加事件处理函数

// 不同的元素具有不同的事件，如下是常见的事件

// change   HTML 元素改变

// click    用户点击 HTML 元素

// mouseover    用户在一个HTML元素上移动鼠标

// mouseout 用户从一个HTML元素上移开鼠标

// keydown  用户按下键盘按键

// load 浏览器已完成页面的加载

document.getElementById("id").addEventListener("click", function(e){

    // 处理元素点击事件

})

// 另一种设置元素的事件处理函数的方法

// onclick 为上面的 click + on，同理其他事件处理

document.getElementById("id").onclick = function(e){

    // 处理元素点击事件

}

**typeof 操作符**

你可以使用 typeof 操作符来检测变量的数据类型。

typeof "John"                       // 返回 string

typeof 3.14                         // 返回 number

typeof false                        // 返回 boolean

typeof [1, 2, 3, 4]                 // 返回 object

typeof { name: 'John', age: 34 }    // 返回 object

typeof undefined                    // undefined

typeof null                         // object

**类型转换**

// 将数字转换为字符串

var x = 5;

x.toString()

// 将布尔类型转字符串

false.toString()     // 返回 "false"

// 将日期类型转字符串

Date().toString()   // 返回 Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe Daylight Time)

// 将字符串转数字

Number("3.14")    // 返回 3.14

Number(" ")       // 返回 0

Number("99 88")   // 返回 NaN

// 布尔转数字

Number(false)     // 返回 0

Number(true)      // 返回 1

// 日期转数字，返回时间戳

Number(new Date())          // 返回 1404568027739

new Date().getTime()        // 返回 1404568027739

**错误处理**

function myFunction() {

    try {

        var x = document.getElementById("demo").value;

        if (x == "") throw "empty";

        if (x < 5) throw "too low";

    }

    catch (err) {

        var y = document.getElementById("mess");

        y.innerHTML = "Error: " + err + ".";

    }

}

**正则**

// 正则声明

// 修饰符  描述

// i    执行对大小写不敏感的匹配。

// g    执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。

// m    执行多行匹配。

var r = /.\*?abc/i

// 正则的方法

r.test("abcdekkkk");    // 是否匹配，返回bool

r.exec("abcdekkkk");    // 查找匹配的字符串，返回找到的字符串数组

// 字符串里应用正则

var str = "Visit w3cschool";

var n = str.search(/w3cschool/i);   // 查找匹配到的字符串的位置，返回索引

var res = str.replace(/w3cschool/i, "microsoft");   // 替换匹配到的字符串

**Json**

var jsonstr = `

{

    "employees":[

        {"firstName":"John", "lastName":"Doe"},

        {"firstName":"Anna", "lastName":"Smith"},

        {"firstName":"Peter", "lastName":"Jones"}

    ]

}

`;

// json 字符串转 js 对象

var jsonObj = JSON.parse(jsonstr);

// js 对象转 json 字符串

var str = JSON.stringify(jsonObj);

## **href="#"与href="javascript:void(0)"的区别**

****#**** 包含了一个位置信息，默认的锚是****#top**** 也就是网页的上端。

而javascript:void(0), 仅仅表示一个死链接。

在页面很长的时候会使用 ****#**** 来定位页面的具体位置，格式为：****# + id****。