

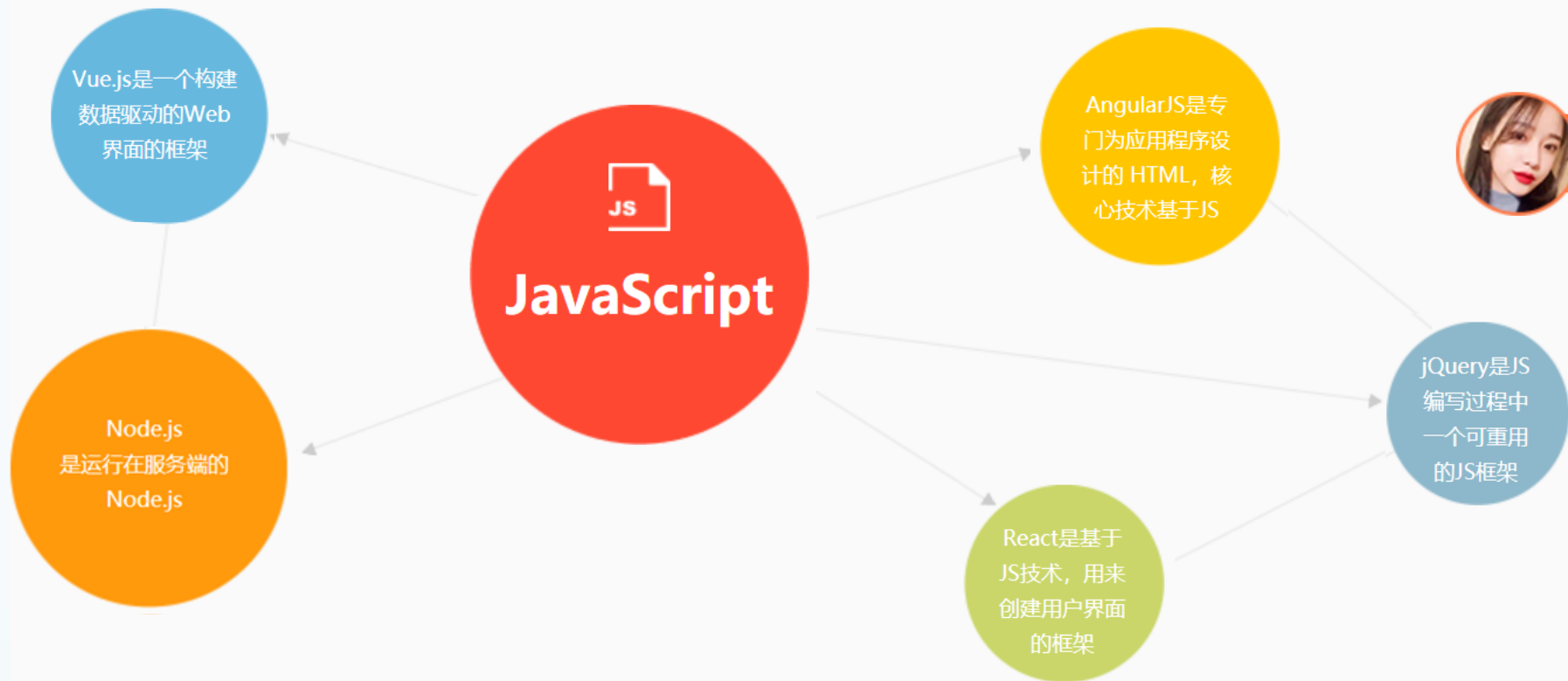


HTML5

CSS3

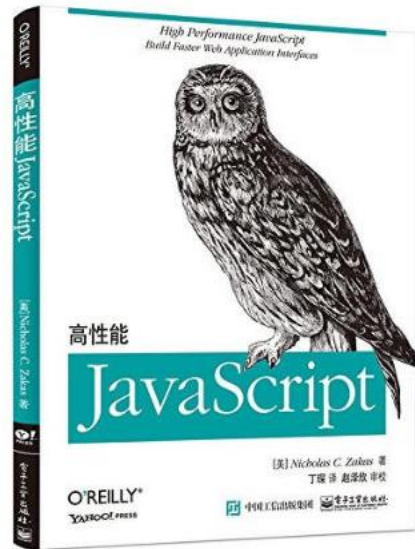
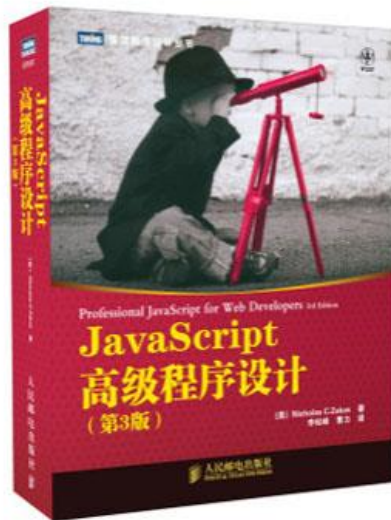
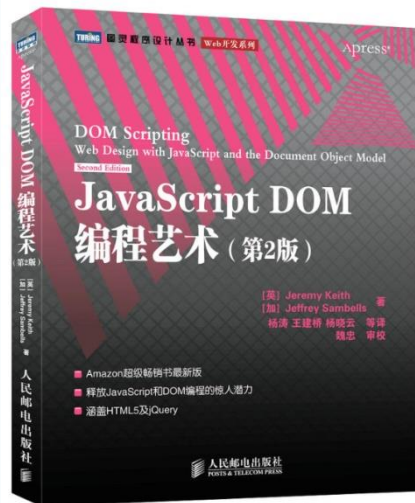
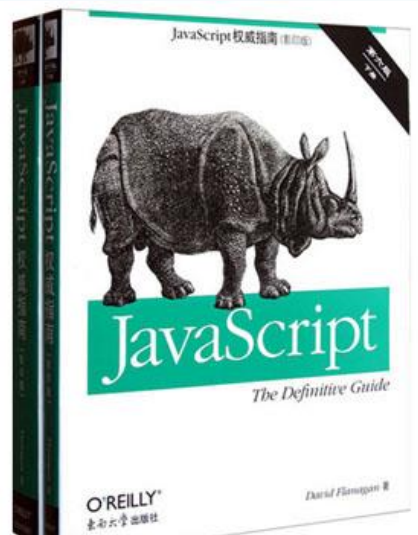
JS







入门与初级书籍





智能社-JavaScript视频【石川blue讲师】

<https://www.bilibili.com/video/BV1Ys411F7Zc?from=search&seid=943601700549317114>

HTML5

CSS3

JS





1

JavaScript概述

2

初识JavaScript

3

JavaScript语言基础

4

JavaScript对象编程

5

JavaScript事件处理





JavaScript 概述

HTML5

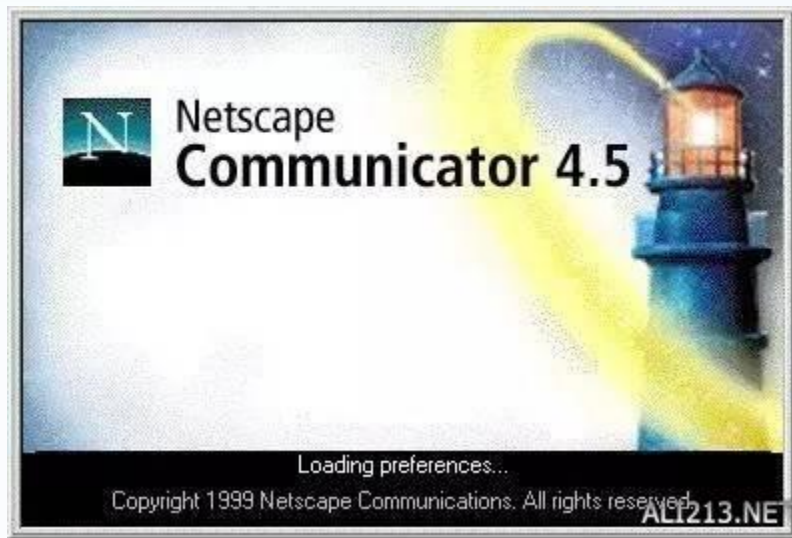
CSS3

JS





布兰登·艾奇（Brendan Eich，1961年～）
JavaScript之父



Netscape（网景）浏览器





ECMAScript是一种规范

JavaScript一种通用目的的脚本语言，遵循 ECMAScript 规范





版本号	发布时间	备注
ECMAScript 1.0（正式）	1997年7月	
ECMAScript 2.0（正式）	1998年6月	
ECMAScript 3.0（正式）	1999年12月	成为JavaScript的通行标准，得到了广泛支持。 增加正则表达式，try / catch异常处理
ECMAScript 4.0（草案）	2007年10月	目标过于激进，各浏览器厂方发生严重分歧。 未全部发布。
ECMAScript 5.0（正式）	2009年12月	
ECMAScript 2015（正式）	2015年6月	每年发布一个ECMAScript版本
ECMAScript 2016（正式）	2016年6月	与前一年相比，增加了两个较小的特性
ECMAScript 2017	2017年6月	
ECMAScript 2018 ES9	2018年6月底	
ECMAScript 2019 ES10	2019/02/21	
ECMAScript 2020		
ECMAScript 2021		





JavaScript概念:

JavaScript是一种**嵌入到HTML文件中的基于对象(Object)和事件驱动(Event Driven)**并具有**安全性**的脚本语言。





- HTML5 CSS3 JS



- **ECMAScript:** 定义了基本的语法和基本对象。现在每种浏览器都有对**ECMAScript**标准的实现。
- **DOM(Document Object Model):** 文档对象模型，它是**HTML**和**XML**文档的应用程序编程接口。浏览器中的**DOM**把整个网页规划成由节点层级构成的文档。用**DOM API**可以轻松地删除、添加和替换节点。
- **BOM(Browser Object Model):** 浏览器对象模型，描述了与浏览器窗口进行访问和操作的方法和接口。





		2	2
8	8	4	
8	16	32	64
1024	512	256	128



用户注册

用户名: ✓

密码: ✗ 以字母开头，长度在6-16之间，必须包含数字和字母。

确认密码: ✗ 请再次输入密码。

手机号码: ✗ 请输入手机号码。

电子邮箱: ✗ 请输入电子邮箱。

验证码: ✗ 请输入验证码。

注册按钮:





控制多平台的能力



适配兼容





- ① 在html元素中嵌入JavaScript
- ② 在页面中直接嵌入JavaScript
- ③ 链接外部JavaScript





1、将JS代码嵌入在元素"事件"中

onclick : 当单击元素时所做的操作

```
<div id="" onclick="JS代码">xxx</div>
```

例:

```
<html>
```

```
<body>
```

```
<button onclick="console.log( 'Hello World' );"> 打印消息 </button>
```

```
</body>
```

```
</html>
```





1、

<script></script>

允许出现网页的任意位置处

例：

<html>

<body>

页头 <hr/>

<script>

```
document.write( '<b>欢迎</b>');
```

```
console.log( '脚本执行结束了...' );
```

</script>

 页尾

</body>





3、将JS代码写在外部脚本文件中 (*.js)

- 1、创建js 文件，并编写JS代码 `***.js`
- 2、在页面中引入 js文件 `<script src="js文件路径"></script>`

```
<html>
  <head>
    <script src="myscript.js"></script>
  </head>
  <body>
  </body>
</html>
```



属性	说明
src	设置一个外部脚本文件的路径位置
type	设置所使用的脚本语言，此属性已代替language属性
defer	此属性表示当HTML文档加载完毕后再执行脚本语言
language	设置所使用的脚本语言及版本。（不推荐使用，用type替代）





语法格式:

```
<script type="text/javascript" src="your-Javascript.js"></script>
```

注意:

```
<script src=""></script>
```

该对标记中, 是不允许出现任何内容的。

如:

```
<script src="a.js"> console.log(); </script>
```

以上代码是错误的。





第一个 JavaScript 小程序

这里显示日期

显示当前日期





创建一个JavaScript文件，并实现弹出一个“欢迎访问网站”的对话框。





初识JavaScript

HTML5

CSS3

JS





福州: ☁ 18℃ 优 28 | 换肤 消息

新闻

hao123

地图

视频

贴吧

学术

irene12345056

设置

更多产品

个人中心

帐号设置

退出

Bai 百度



百度一下

淘宝网
Taobao.com

宝贝

天猫

店铺

Q 菲丽菲拉唇釉



搜索

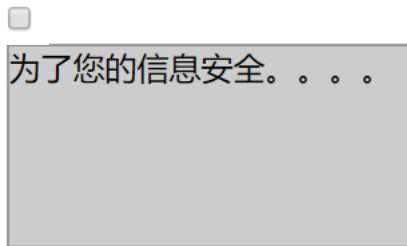
积木 现代装饰画 无人机 新款男鞋 沙发 夹克 蓝牙耳机 男士内裤 定制窗帘 男皮鞋 电脑椅 餐桌 电视柜

HTML5

CSS3

JS





如何实现？

1、分析效果实现原理

- 效果：鼠标悬停在按钮出现div，移开则隐藏。
- 样式：div的display
- 事件：onmouseover、onmouseout
- 动手编写此效果

2、特效基础

- 事件驱动：onmouseover
- 元素属性操作：obj.style.[.....]
- 特效实现原理概括：响应用户操作，对页面元素（标签）进行某种修改

知识点：

- 1、getElementById获取对象
- 2、事件

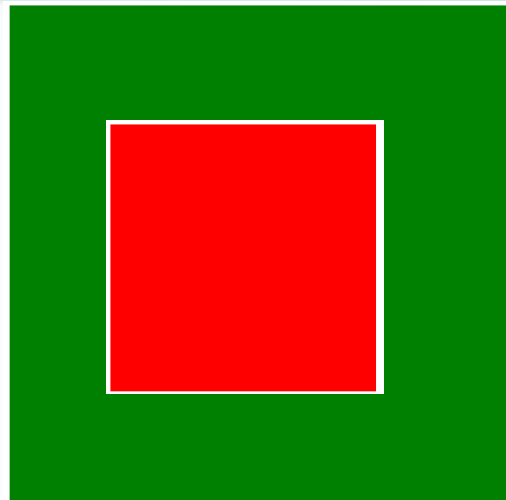
源代码





• 鼠标移入移出效果

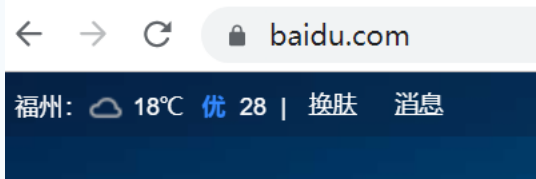
- 当鼠标移入时div宽度、高度和背景变化
- 当鼠标移出时恢复（div宽度、高度和背景变化）



源代码

知识点：函数

• 换肤（类似百度）



源代码

HTML5

CSS3

JS





直接在事件内写代码会很乱

- ## ● 定义和调用

- 1、函数定义：只是告诉系统有这个函数，不会实际执行
- 2、函数调用：真正执行函数里的代码
- 3、关系和区别



- 第一个JS兼容性问题

在Firefox下直接使用ID存在问题

- 引入document.getElementById()
- document.getElementById在任何浏览器下均可使用

- 网页换肤

- 任何标签都可以加ID，包括link
- 任何标签的任何属性，也都可以修改
- HTML里怎么写，JS里就怎么写





显示隐藏

如何实现？

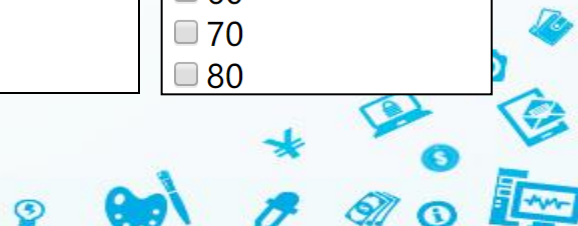
点击按钮显示/隐藏Div (弹出菜单)

- 逻辑判断：if的基本形式
- Div的隐藏和出现
- 按钮上文字变化

知识点扩展：

- 为a链接添加JS
``
- className的使用





- ## 源代码

HTML5



- 按钮的实现

- ✓ 添加事件

- this的使用

- ✓ 先清空所有按钮，再选中当前按钮

- 内容的实现(div)

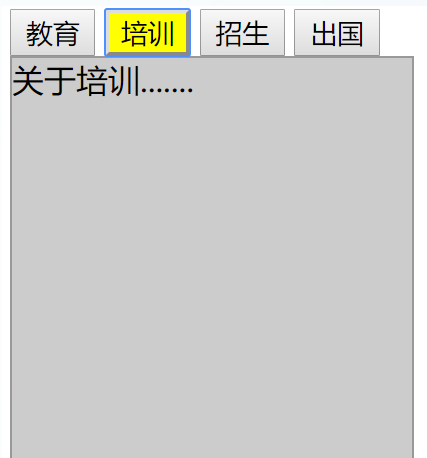
- ✓ 先隐藏所有Div，再显示”当前” Div

- 索引值的使用

- 什么时候用索引值——需要知道“第几个”的时候

- html添加index——会被FF过滤

- js添加index



源代码





- 程序实现思路
 - 类似选项卡，只是下面只有一个div
 - innerHTML的使用
- 数组的使用
 - 定义：arr=[1,2,3]
 - 使用：arr[0]
- 字符串连接
 - 作用：连接两个字符串
 - 问题：连接中的优先级





JavaScript 语言基础

HTML5

CSS3

JS





05 函数

JS



JavaScript 数据结构

HTML5

CSS3

JS





在JavaScript中，标识符用来命名变量和函数。

命名规则：

- 1.第一个字符，可以是任意Unicode字母（包括英文字母和其他语言的字母），以及美元符号（\$）和下划线（_）。
- 2.第二个字符及后面的字符，除了Unicode字母、美元符号和下划线，还可以用数字0-9。
- 3.不能使用JavaScript中的保留字或关键字，如var、int等。
- 4.应使用有意义的变量名，可以使用“驼峰式”或“下划线式”的变量名，如userName，user_name。

※注意：JavaScript对大小字母是“敏感”的，即区分大小写字母。





例：合法的标识符

```
i  
my_name  
_name  
$str  
n1
```

例：不合法的标识符

```
1a // 第一个字符不能是数字  
23 // 同上  
*** // 标识符不能包含星号  
a+b // 标识符不能包含加号  
-d // 标识符不能包含减号  
或连词线
```





<script>

var 1x=1; (X)

var while="please click the button"; (X)

var _while="Next page";

var y2=12.5;

var y3=Y2;

var the sum=100; (X)

var errorMessage=""; (X)

</script>





JavaScript关键字是指在JavaScript语言中有特定含义，成为JavaScript语法中的那些关键词。

JavaScript关键字是不能作为变量名和函数名使用的。使用JavaScript关键字作为变量名或函数名，会使JavaScript在载入过程中出现编译错误。



JavaScript的關鍵字

前端开发技术



abstract	continue	finally	instanceof	private	this
boolean	default	float	int	public	throw
break	do	for	interface	return	typeof
byte	double	function	long	short	true
case	else	goto	native	static	var
catch	extends	implements	new	super	void
char	false	import	null	switch	while
class	final	synchronized	package	in	with

HTML5

CSS3

JS





当程序运行时，值不能改变的量为**常量**。

常量主要用于为程序提供固定的和精确的值（包括数值和字符串），比如数字、逻辑值真（**true**）、逻辑值假（**false**）等都是常量。

声明常量使用**const**关键字来进行声明。

语法：

`const 常量名=值；` 如：`const a=true;`





计算机中的程序需要对“值”进行操作。如（数字、字符串）等。当程序需要将这些值保存起来时，就需要将这些“值”保存到变量中，为后续使用。

所以，变量是对“值”的再次使用。也称为“引用”。

变量的工作机制是编程语言的基本特性。





例：

```
var a=1;
```

- 1.首先通过**var**关键字先声明变量**a**，然后在变量**a**与数值**1**之间建立引用(使用)关系，也称为将数值**1**“赋值”给变量**a**。
- 2.以后，引用（使用）变量**a**就会得到数值**1**。





- 命名规范及必要性
 - 可读性——能看懂
 - 规范性——符合规则
- 匈牙利命名法
 - 类型前缀
 - 首字母大写
- 骆驼命名法





类型	前缀	类型	实例
数组	a	Array	altems
布尔值	b	Boolean	blsComplete
浮点数	f	Float	fPrice
函数	fn	Function	fnHandler
整数	i	Integer	iltemCount
对象	o	Object	oDiv1
正则表达式	re	RegExp	reEmailCheck
字符串	s	String	sUserName
变体变量	v	Variant	vAnything



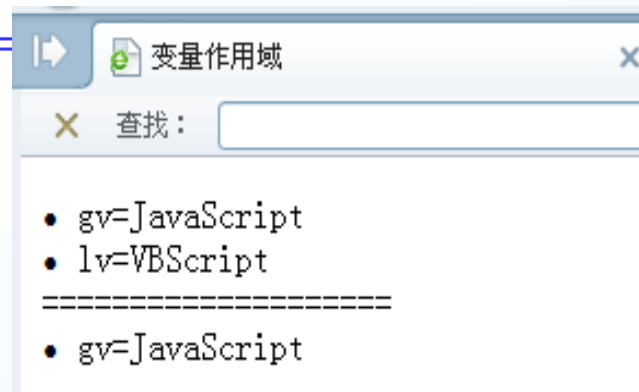


- 变量的作用域是指变量在程序中的有效范围，也就是程序中使用这个变量的区域。
- 根据作用域，变量可分为以下三种类型：
 - **全局变量**：全局变量声明在**所有函数之外**，作用于页面的整个脚本代码(**var**声明的变量)或从声明语句开始的整个页面脚本代码(**let**声明的变量)。
 - **局部变量**：指在**函数体内**声明的变量或函数中的**形参**。
 - **块级变量**：块级变量是**在块中**使用**let**声明的变量，只在块中有效。





```
<script>
  document.title = "变量作用域";
  var gv = "JavaScript";      //gv是全局变量
  test();
  function test() {
    var lv = "VBScript";      //lv是局部变量
    document.write("<LI>gv=" + gv);
    document.write("<LI>lv=" + lv);
  }
  document.write("<br/>=====");
  document.write("<LI>gv=" + gv);
  document.write("<LI>lv=" + lv);
</script>
```





JavaScript 数据类型

HTML5

CSS3

JS





- ① 数字型数据
- ② 字符串型数据
- ③ 布尔型数据
- ④ 特殊数据类型
- ⑤ 数据类型的转换规则





- 类型： **typeof**运算符
 - 用法、返回值
 - 常见类型：
 - number、string、boolean、undefined、object、function
- 建议： 一个变量应该只存放一种类型的数据





JavaScript内部，所有数字都是以64位浮点数形式储存，即使整数也是如此。所以，1与1.0是相同的，是同一个数。

例：1===1.0

常用方法：

`parseInt()`

用于将字符串转为整数。

`parseFloat()`

用于将字符串转为浮点数。





字符串就是零个或多个排在一起的字符，放在单引号或双引号中。

例： 'abc' "abc"

length属性返回字符串的长度，该属性也是无法改变的。

例：

```
var s = 'hello';  
s.length;                      // 5  
s.length = 3;  
s.length;                      // 5
```





布尔型数据通常在JavaScript程序中用来比较所得的结果。

布尔型数据类型只有两个值，分别是“true”和“false”，它用来表示某种情况是真还是假。

例：

```
var n=2;  
n==1
```





undefined与**null**都可以表示“没有”,含义非常相似。

1)**undefined**声明一个变量但未初始化,这个变量的值就自动被赋予**undefined**值。 如:

```
var a;  
console.log(a); //undefined
```

null表示一个空指针对象,通常用来将对象变量(如数组、对象、函数等)初始化。

```
var oNull=document.getElementById("div1"); //div1是一个不存在的网页元素。  
alert(oNull); //null
```

2)将一个变量赋值为**undefined**或**null**,实际讲,几乎没区别。

```
var a=null;  
var b=undefined;  
console.log(a==b); //true  
console.log(a===b); //false
```





- 数据类型转换
 - 例子：计算两个文本框的和
 - 显式类型转换(强制类型转换)
 - parseInt()、parseFloat()
 - NaN的意义和检测
 - 隐式类型转换
 - ==、===
 - 减法





常见数据类型

Number 类型

```
var a = 10;           // 十进制(整数)
var b = 1.1;          // 浮点数
var c = 0x12ac;        // 十六进制
var d = 5.0;           // 解析成整数5
console.log(c);        // 打印值: 4780
console.log(d);        // 打印值: 5
```

String 类型

```
var string1 = "51购商城"; // 双引号
var string2 = '51购商城'; // 单引号
var string3 = "51\购\";   // 转义字符
console.log(string3);     // 51'购'
```

Boolean 类型

```
var a = "Hello world!";
if (a){
  console.log("该语句已执行打印!");
}
```

特殊 数据类型

```
var a;           // 声明后未初始化
console.log(a);   // undefined

var b=null;       //null 数据类型
console.log(b);   // null
```

dataType.js





① 算术运算符

③ 赋值运算符

⑤ 布尔运算符

⑦ 运算符优先级

② 比较运算符

④ 字符串运算符

⑥ 条件运算符

⑧ 表达式





- 算术：+ 加、- 减、* 乘、/ 除、% 取模
 - 实例：隔行变色、秒转时间
- 赋值：=、+=、-=、*=、/=、%=
- 关系：<、>、<=、>=、==、===、!=、!==
- 逻辑：&& 与、|| 或、! 否
 - 实例：全选与反选
- 运算符优先级：括号





例12-2





- 判断：if、switch、?:
- 循环：while、do-while、for/for-in
- 跳出：break、continue
- 什么是真、什么是假：
 - 真：true、非零数字、非空字符串、非空对象
 - 假：false、数字零、空字符串、空对象、undefined





登录

 邮箱/手机/用户名

 请输入密码

☐ 记住密码 [注册](#)

登 录

合作账号

 QQ登录  微博登录  微信登录

源码





```
<script>
function login(){
    var user=document.getElementById("user");//获取账户信息
    var password=document.getElementById("password");//获取密码信息
    if(user.value!=='mr' && password.value!=='mrsoft' ){
        alert('您输入的账户或密码错误！ ');
    }else{
        alert('登录成功！ ');
    }
}
</script>
```





- **for...in**语句主要用于遍历数组元素或者对象的属性。

- 基本语法：

```
for(变量 in 对象) {  
    循环体;  
}
```

- 说明：

变量：用于指定数组元素或对象的属性。对于数组，变量的值等于所遍历到的元素所对应的索引；对于对象，变量的值等于所遍历到的属性名。

对象：为数组名或对象名。

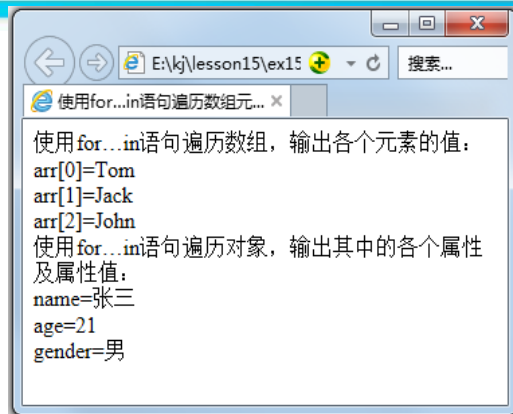
- 功能：当条件表达式为**true**时，执行循环体语句，否则跳出循环体。



for in 循环示例



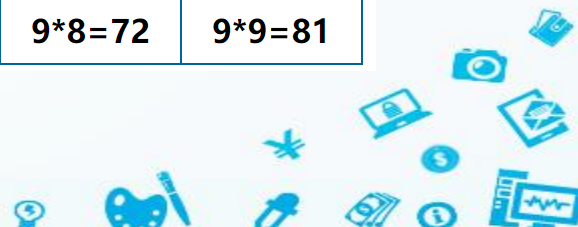
```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>使用for...in语句遍历数组元素和对象属性</title>
<script>
var arr=new Array("Tom", "Jack", "John");
var obj=new Object();
obj.name="张三";
obj.age=21;
obj.gender='男';
document.write("使用for...in语句遍历数组，输出各个元素的值：<br/>")
for(var index in arr){/*index是数组arr的元素下标*/
    document.write("arr["+index+"]="+arr[index]+"<br/>");
}
document.write("使用for...in语句遍历对象，输出其中的各个属性及属性值：<br/>")
for(var attr in obj){/*attr是对象obj的属性*/
    document.write(attr+"="+obj[attr]+"<br/>");
}
</script>
</head>
<body>
</body>
</html>
</html>
```





JavaScript实现的九九乘法表

1*1=1								
2*1=2	2*2=4							
3*1=3	3*2=6	3*3=9						
4*1=4	4*2=8	4*3=12	4*4=16					
5*1=5	5*2=10	5*3=15	5*4=20	5*5=25				
6*1=6	6*2=12	6*3=18	6*4=24	6*5=30	6*6=36			
7*1=7	7*2=14	7*3=21	7*4=28	7*5=35	7*6=42	7*7=49		
8*1=8	8*2=16	8*3=24	8*4=32	8*5=40	8*6=48	8*7=56	8*8=64	
9*1=9	9*2=18	9*3=27	9*4=36	9*5=45	9*6=54	9*7=63	9*8=72	9*9=81





定义

1 function命令

```
function print(s) {  
    console.log(s);  
}
```

2 函数表达式

```
var print = function(s) {  
    console.log(s);  
};
```





带参函数

```
function f(x, y) {  
    return x+y;  
}
```

函数参数不是必需的，Javascript允许省略参数。

函数调用





例12-5



关键代码

```
<div class="pay">
  <div class="pay-opt">
    <a href="index.html"><span class="mr-icon-home mr-icon-fw">
      首页</span></a>
    <a><span class="mr-icon-heart mr-icon-fw">收藏</span></a>
  </div>
  <li>
    <div class="clearfix tb-btn tb-btn-buy theme-login">
      <a id="LikBuy" title="点此按钮到下一步确认购买信息"
        href="javascript:void(0)" onclick="mr_function();">立即购买</a>
    </div>
  </li>
  <li>
    <div class="clearfix tb-btn tb-btn-basket theme-login">
      <a id="LikBasket" title="加入购物车"
        href="javascript:void(0)" onclick="mr_function();"><i></i>加入购物车</a>
    </div>
  </li>
</div>
```

```
<script>
  function mr_function(){
    alert('触发了一个函数！');
  }
</script>
```






- 可变参（不定参）：arguments
 - ✓ 参数的个数可变，参数数组
- 例子1：求和（求所有参数的和）
- 例子2：CSS函数
 - ✓ 判断arguments.length
 - ✓ 给参数取名，增强可读性
 - ✓ 取非行间样式(不能用来设置):
 - obj.currentStyle[attr]
 - getComputedStyle(obj, false)[attr]





函数	说明
eval()	求字符串中表达式的值
isFinite()	判断一个数值是否为无穷大
isNaN()	判断一个数值是否为NaN
parseInt()	将字符型转化为整型
parseFloat()	将字符型转化为浮点型
encodeURIComponent()	将字符串转化为有效的URL
decodeURI()	对encodeURIComponent()编码的文本进行解码
DecodeURIComponent()	对encodeURIComponent()编码的文本进行解码





请输入要格式化的数字:

请输入格式化后数字的长度:

格式化后的数字:





关键代码

```
<script language="javascript">
function deal(){
    if(form1.str.value==""){
        alert("请输入要格式化的数字！");
        form1.str.focus();
        return false;
    }
    if(isNaN(form1.str.value)){
        alert("您输入的数字不正确!");
        form1.str.focus();
        return false;
    }
    if(form1.le.value==""){
        alert("请输入格式化后的长度！");form1.le.focus();return false;}
    if(isNaN(form1.le.value)){
        alert("您输入的格式化的长度不正确!");form1.le.focus();return false;
    }
    form1.lastStr.value=formatNO(form1.str.value,form1.le.value);
}
</script>
```





用XML表示中国部分省市数据如下:

```
<?xml version="1.0" encoding="utf-8"?>
<country>
  <name>中国</name>
  <province>
    <name>黑龙江</name>
    <cities>
      <city>哈尔滨</city>
      <city>大庆</city>
    </cities>
  </province>
  <province>
    <name>广东</name>
    <cities>
      <city>广州</city>
      <city>深圳</city>
      <city>珠海</city>
    </cities>
  </province>
  <province>
    <name>台湾</name>
    <cities>
      <city>台北</city>
      <city>高雄</city>
    </cities>
  </province>
  <province>
    <name>新疆</name>
    <cities>
      <city>乌鲁木齐</city>
    </cities>
  </province>
</country>
```

用JSON表示如下

```
{
  "name": "中国",
  "province": [{
    "name": "黑龙江",
    "cities": {
      "city": ["哈尔滨", "大庆"]
    }
  }], {
    "name": "广东",
    "cities": {
      "city": ["广州", "深圳", "珠海"]
    }
  }], {
    "name": "台湾",
    "cities": {
      "city": ["台北", "高雄"]
    }
  }], {
    "name": "新疆",
    "cities": {
      "city": ["乌鲁木齐"]
    }
  }
}]
```

比较可以看出:

- 1) JSON 简单的语法格式和清晰的层次结构明显要比 XML 容易阅读,
- 2) 在数据交换方面, 由于 JSON 所使用的字符要比 XML 少得多, 可以大大得节约传输数据所占用的带宽。





● 数组的使用

✓ 定义

- `var arr=[12, 5, 8, 9];`
- `var arr=new Array(12, 5, 8, 9);`

几乎没有差别，`[]`的性能略高，因为代码短。

● 数组的属性

✓ length

- 既可以获取，又可以设置
- 例子：快速清空数组





- 数组的方法

- ✓ 添加

- push(元素), 从尾部添加
- unshift(元素), 从头部添加

- ✓ 删除

- pop(), 从尾部弹出
- shift(), 从头部弹出





● 数组的方法

✓ 排序

- `sort([比较函数])`，排序一个数组
 - ✓ 排序一个字符串数组
 - ✓ 排序一个数字数组

✓ 转换类

- `concat(数组2)`
 - ✓ 连接两个数组
- `join(分隔符)`
 - ✓ 用分隔符，组合数组元素，生成字符串
 - ✓ 字符串`split`





- splice

- ✓ splice(开始, 长度, 元素...)
- ✓ 先删除, 后插入

- 删除

- ✓ splice(开始, 长度)

- 插入

- ✓ splice(开始, 0, 元素...)

- 替换





- JavaScript语法规则
- 变量及常量
- 数据类型及转换
- 运算符
- 控制结构
- 函数
- 数组





JavaScript 对象编程

HTML5

CSS3

JS





01 JavaScript内置对象

02 BOM (window对象)

03 DOM

04 JavaScript与表单操作





- **JavaScript对象**指的是这样一类特殊的数据类型，它不仅
可以保存一组不同类型的数据（属性），而且还可以包含
有关处理这些数据的函数（方法）
- **JavaScript对象类型**:
 - JavaScript内置对象
 - 浏览器模型(BOM)的对象
 - 文档模型(DOM)的对象
 - 自定义对象



- Array对象
- String对象
- Math对象
- Date对象
- RegExp对象

注：请查阅相关资料学习

-



Window 窗口对象





顶层Window对象是所有其他子对象的父对象，它出现在每一个页面上，并且可以在单个JavaScript应用程序中被多次使用。

属性	描述
document	对话框中显示的当前文档
frames	表示当前对话框中所有frame对象的集合
location	指定当前文档的URL
name	对话框的名字
status	状态栏中的当前信息
defaultstatus	状态栏中的当前信息
top	表示最顶层的浏览器对话框
parent	表示包含当前对话框的父对话框
opener	表示打开当前对话框的父对话框
closed	表示当前对话框是否关闭的逻辑值
self	表示当前对话框
screen	表示用户屏幕，提供屏幕尺寸、颜色深度等信息
navigator	表示浏览器对象，用于获得与浏览器相关的信息





方法	描述
alert()	弹出一个警告对话框
confirm()	在确认对话框中显示指定的字符串
prompt()	弹出一个提示对话框
open()	打开新浏览器对话框并且显示由 URL 或名字引用的文档，并设置创建对话框的属性
close()	关闭被引用的对话框
focus()	将被引用的对话框放在所有打开对话框的前面
blur()	将被引用的对话框放在所有打开对话框的后面
scrollTo(x,y)	把对话框滚动到指定的坐标
scrollBy(offsetx,offsety)	按照指定的位移量滚动对话框
setTimeout(timer)	在指定的毫秒数过后，对传递的表达式求值
setInterval(interval)	指定周期性执行代码





01:26:16

如何实现？

思路点拨：

1. 获取系统时间

① Date对象

② getHours、getMinutes、getSeconds

2. 显示系统时间

① 字符串连接

② 空位补零

3. 设置图片路径

charAt方法

源码





- HTML5 CSS3 JS



- 开启定时器
 - `setInterval` 间隔型
 - `setTimeout` 延时型
 - 两种定时器的区别
- 停止定时器
 - `clearInterval`
 - `clearTimeout`





- **DOM**(Document Object Model): 即文档对象模型。
- 引入案例
 - 子节点





- DOM基础
 - 什么是DOM
 - 浏览器支持情况
- DOM节点
 - childNodes nodeType （文本节点3和元素节点1）
 - 获取子节点
 - children
 - parentNode
 - 例子：点击链接，隐藏整个li
 - offsetParent
 - 例子：获取元素在页面上的实际位置





● DOM节点(2)

● 首尾子节点

- 有兼容性问题
- firstChild、firstElementChild
- lastChild、lastElementChild

● 兄弟节点

- 有兼容性问题
- nextSibling、nextElementSibling
- previousSibling、previousElementSibling





效果演示

实现方法：

1.原来的方法

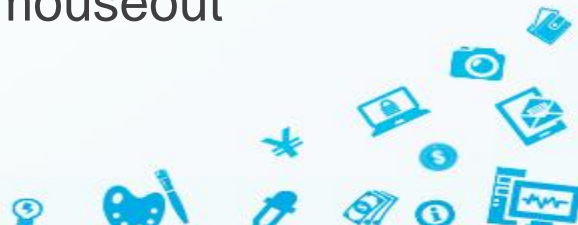
移入显示，移出隐藏

2.移出延时隐藏

移入下面的Div后，还是隐藏了

3.简化代码

合并两个相同的mouseover和mouseout





JavaScript 事件处理





01

事件与事件处理概述

02

DOM事件模型

03

鼠标键盘事件

04

页面事件

05

表单事件

[更多内容见JS事件](#)

