

# 1. JavaScript 简介

## 1.1. JavaScript 由来

Netscape 发明了 JavaScript

JavaScript 由 Netscape 在 1995 年发明。早期的主要目的是处理一些用户的输入验证操作。而在 JavaScript 语言出现之前客户端的页面时需要提交到服务器端，由服务器去检测的。在刚刚普及的电话线调制解调器时代，对用户是一种考验，著名的 Netscape Navigator（早期浏览器）通过引入 JavaScript 来解决该问题

随着互联网的流行，网页已经不断变得更大和复杂，如果用户想要注册表单，需要直接将表单提交到服务器进行验证，需要和服务器进行多次的往返交互，例如，用户注册一个表单，点击提交按钮，等待 30 秒服务器返回处理后，返回的是用户名不符合规则。这种用户体验是很不友好的。此时的 Netscape（网景）开始着手解决这个问题。

Netscape 在 1995 年发行的 Netscape Navigator 2.0 开发一个称之为 LiveScript 的脚本语言，当时的目的是在浏览器和服务器（本来要叫它 LiveWire）端使用它。后来 Netscape 和 Sun 公司合作，所以最后 Netscape 与 Sun 及时完成 LiveScript 实现。就在 Netscape Navigator 2.0 即将正式发布前，Netscape 将其更名为 JavaScript，目的是为了利用 Java 这个因特网时髦词汇，此后 JavaScript 从此变成了因特网的必备组件。

三足鼎立

微软进军微软决定进军浏览器，发布了 IE 3.0 并搭载了一个 JavaScript 的克隆版，叫做 JScript（这样命名是为了避免与 Netscape 潜在的许可纠纷）。

在微软进入后，有 3 种不同的 JavaScript 版本同时存在：Netscape Navigator 3.0 中的 JavaScript、IE 中的 JScript 以及 CEnv 中的 ScriptEase。JavaScript 并没有一个标准来统一其语法或特性，而这 3 种不同的版本恰恰突出了这个问题。随着业界担心的增加，这个语言的标准化显然已经势在必行。

标准化

1997 年，JavaScript 1.1 作为一个草案提交给欧洲计算机制造商协会（ECMA）。第 39 技术委员会（TC39）被委派来“标准化一个通用、跨平台、中立于厂商的脚本语言的语法和语义”。由来自 Netscape、Sun、微软、Borland 和其他一些对脚本编程感兴趣的公司的程序员组成的 TC39 锤炼出了 ECMA-262，该标准定义了名为 ECMAScript 的全新脚本语言。随后，国际标准化组织及国际电工委员会（ISO/IEC）也采纳 ECMAScript 作为标准。

从此，Web 浏览器就开始努力将 ECMAScript 作为 JavaScript 实现的基础。

JavaScript 是属于网络的脚本语言！JavaScript 被数百万计的网页用来改进设计、验证表单、检测浏览器、创建 cookies，以及更多的应用。JavaScript 是因特网上最流行的脚本语言。

注：

javascript 运行必须依赖于宿主环境语言，即页面语言 HTML。

是解释型的语言，解释型：不需要编译，解释器程序会每读取一条语句就执行。运行速度慢，浏览器中默认内置了 javascript 的解释器程序。

浏览器中默认内置了 javascript 的解释器程序。

常见的脚本语言：

ECMAScript 主要进行所有脚本语言的标准制定。

JavaScript

JScript

VBScript

ActionScript

JavaScript 是基于对象和事件的脚本语言。

## 1.2. JavaScript 特点

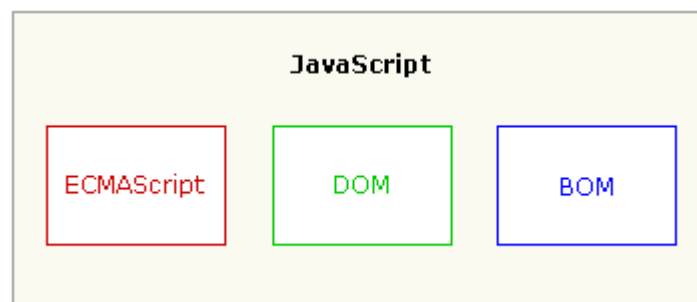
1. 安全性（不允许直接访问本地硬盘），它可以做的就是信息的动态交互。
2. 跨平台性。（只要是可解释 Js 的浏览器都可以执行，和平台无关。）

## 1.3. JavaScript 与 Java 不同

1. JS 是 Netscape 公司的产品，Java 是 Sun 公司的产品
2. JS 是基于对象，Java 是面向对象。
3. JS 只需解释就可以执行，Java 需要先编译成字节码文件，再执行。
4. JS 是弱类型，Java 是强类型。

## 1.4. JavaScript 内容

尽管 ECMAScript 是一个重要的标准，但它并不是 JavaScript 唯一的部分，一个完整的 JavaScript 实现是由以下 3 个不同部分组成的：



目前我们学习 JavaScript 也就是需要学习：

- ✓ JavaScript 语法基础
- ✓ 使用 JS 操作网页（DOM）
- ✓ 使用 JS 操作浏览器（BOM）

## 2. JavaScript 基础

### 2.1. 语法

#### 2.1.1. 如何编写 js 代码

向 HTML 页面插入 JavaScript 的主要方法，就是使用<script 元素>。使用<script>元素的方式有两种：直接在HTML页面中嵌入JavaScript代码和包含外部的JavaScript文件。

1. JS 代码存放在标签对<script>... </script>中。
2. 使用 script 标签的 src 属性引入一个 js 文件。（方便后期维护，扩展）  
例：<script src="test.js" type="text/javascript"></script>

注：规范中 script 标签中必须加入 type 属性。

例如：在 html 页面的<script>标签中编写 js 代码。

```
<html >
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />

    <title>在HTML中使用JavaScript</title>

  </head>
  <body>

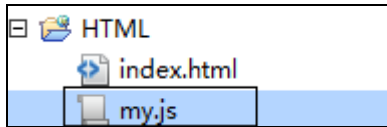
    <h1>在页面中嵌入JavaScript</h1>

    <script type="text/javascript">
      window.document.write("hello,world");
    </script>
  </body>
</html>
```

例如：在 html 文件中引入外部的 js 文件。

```
<html >
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>在HTML中使用JavaScript</title>
  </head>
  <body>
    <h1>在页面中引入外部JavaScript文件</h1>
    <script type="text/javascript" src="my.js"></script>
  </body>
</html>
```

JavaScript 文件



✓ <script>标签的位置:

<script>应该放在页面的<head>元素中。

```
<html >
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
    <title>在HTML中使用JavaScript</title>
    <script type="text/javascript" >
      window.document.write("世界你好!!! ");
    </script>
  </head>
  <body>
    <h1>将JavaScript标签放在head上</h1>
  </body>
</html>
```

注意:

- 1) 页面上可以有多个<script>标签;
- 2) <script>标签按顺序执行;
- 3) <script>标签可以出现在任意的页面位置;
- 4) <script>标签一定要写</script>关闭, 而不能<script/>这样关闭。否则没有任何错误信息, 但是没有运行结果;

## 2.1.2. 注释

**JavaScript 注释分为两种:**

第一种: 单行注释以双斜杠开头 (//)

第二种: 多行注释以单斜杠和星号开头 (/\*), 以星号和单斜杠结尾 (\*/)

## 2.2. 变量声明

Javascript 是弱类型的语言。与 Java 不同, ECMAScript 中的变量无特定的类型, 定义变量时只用 var 运算符, 可以将它初始化为任意值。因此, 可以随时改变变量所存数据的类型。

```
<script type="text/javascript" >
  //定义变量
  var color = "red";
```

```
var num = 25;
var visible = true;
</script>
```

#### 使用细节:

- 1、var 关键字在定义变量的时候可以省略不写;
- 2、变量名可以重复，后面的将覆盖前面的变量;
- 3、变量的类型取决于值的类型;

## 2.3. 基本数据类型

### 2.3.1. typeof 操作符

typeof 操作符可以获取一个变量的类型。

```
<script type="text/javascript" >
  var a = 100;
  var b = 3.14;
  var c = true;
  var d = 'a';
  var e = "hello";
  var f;
  document.write( typeof a + "<br/>" );
  document.write( typeof b + "<br/>" );
  document.write( typeof c + "<br/>" );
  document.write( typeof d + "<br/>" );
  document.write( typeof e + "<br/>" );
  document.write( typeof f + "<br/>" );
</script>
```

### 2.3.2. 基本数据类型

1. JavaScript 的基本数据类型: Number、String、Boolean、Date、undefined。

2. 基本数据类型的总结:

- ✓ 所有的数值都是 number 类型;
- ✓ 字符和字符串都是 string 类型;
- ✓ 布尔是 boolean 类型;
- ✓ 如果一个变量没有初始化值的时候，其类型为 undefined 类型。表示没有定义;

### 2.3.3. 转换成数字

ECMAScript 提供了两种把非数字的原始值转换成数字的方法，即 `parseInt()` 和 `parseFloat()`。只有对 `String` 类型调用这些方法，它们才能正确运行；对其他类型返回的都是 `NaN`。

#### 1. 转换成整数：`parseInt()`

- 1) 如果 `parseInt` 方法接收的字符串含有非数字的字符，那么 `parseInt` 方法会从字符串的首个字符开始寻找，直到第一个非数字字符为止，然后就使用前面的数字字符转换成数字；
- 2) 如果第一个字符是非数字字符，那么就返回 `NaN`（非数字）；
- 3) 如果字符串的首位是 `0`，那么转换时候会自动把 `0` 去掉；
- 4) 如果字符串是以 `0x` 开头，那么转换时候就会把字符串中的内容当作十六进制的数据进行处理；

#### 2. 转换成小数：`parseFloat()`

- 1) 如果字符串是整数，那么使用 `parseFloat` 转换后的数据也是整数；
- 2) 如果 `parseFloat` 方法接收的字符串含有非数字的字符，那么 `parseFloat` 方法会从字符串的首个字符开始寻找，直到第一个非数字字符为止，然后就使用前面的数字字符转换成数字；

## 2.4. 运算符

### 2.4.1. 算术运算符

#### 1. 加号运算符

加号运算符使用 `(+)` 表示。加号也可以作为正数、字符串连接符使用。

注意： `true` 也可以表示 `1`，`false` 也可以表示 `0`。

#### 2. 乘法运算符

乘法运算符由星号 `(*)` 表示，用于两数相乘。

#### 3. 除法运算符

除法运算符由斜杠 `(/)` 表示。

注意：如果两个整数相除不能被整除，那么结果还是小数。

### 2.4.2. 比较运算符

#### 1. 数字与数字的比较

```
var bResult1 = 2 > 1    //true
var bResult2 = 2 < 1    //false
```

## 2. 字符串与字符串的比较

```
var bResult = "25" < "3";  
alert(bResult); //输出 "true"
```

上面这段代码比较的是字符串 "25" 和 "3"。两个字符串在比较的时候 比较的是两个字符串对应的字符顺序。

## 3. 字符串与数字的比较

如果把某个运算数改为数字，那么结果就不一样了：

```
var result = "25" < 3;  
alert(result); //输出 "false"
```

注意：无论何时比较一个数字和一个字符串，ECMAScript 都会把字符串转换成数字，然后按照数字顺序比较它们。

## 2.4.3. 逻辑运算符

&& 逻辑与

|| 逻辑或

! 逻辑非

## 2.4.4. 赋值运算符

### 1. 简单的赋值运算符由等号(=)实现，只是把等号右边的值赋予等号左边的变量。

例如：  

```
var iNum = 10;
```

### 2. 复合赋值运算是由乘法运算符、加法运算符或位移运算符加等号(=)实现的。这些赋值运算符是下列这些常见情况的缩写形式：

```
var iNum = 10;  
iNum = iNum + 10;
```

可以用一个复合赋值运算符改写第二行代码：

```
var iNum = 10;  
iNum += 10;
```

每种主要的算术运算以及其他几个运算都有复合赋值运算符：

```
乘法/赋值 (*=  
除法/赋值 (/=  
取模/赋值 (%=  
加法/赋值 (+=  
减法/赋值 (-=
```

## 2.4.5. 三目运算符

运算符是 ECMAScript 中功能最多的运算符，它的形式与 Java 中的相同。

语法格式：

条件表达式 ? 值 1 : 值 2;

如果“条件表达式”为 true，就返回值 1，如果条件表达式为 false，就返回值 2。

## 2.5. 流程控制

### 2.5.1. 判断

✚ if 语句的语法：

```
if (条件表达式) statement1 else statement2
```

如果条件表达式为 true，则执行 statement1；如果条件表达式为 false，则执行 statement2。

✚ if 语句的条件表达式除了可以是布尔表达式，还可以是任何类型的数据。

number: 如果非 0 为 true, 0 为 false;  
string: 如果非 null 或非空为 true, 否则为 false;  
undefined: false;  
NaN: false;  
对象类型: 非 null 为 true, 否则为 false;

### 2.5.2. 选择

✚ switch 语句的语法：

```
switch (expression){  
    case value1:  
        statement1;  
        break;  
    case value2:  
        statement2;  
        break;  
    ...  
    default:  
        statementN;  
}
```

✚ ECMAScript 和 Java 中的 switch 语句的区别：



在 ECMAScript 中, switch 语句中 case 后面可以是字符串, 也可以是变量或表达式。

```
function test(){
    var color = "red";
    var value1 = "red", value2 = "green";
    switch(color) {
        case value1:
            alert("红色");
            break;
        case value2:
            alert("绿色");
            break;
        default:
            alert("执行默认default");
    }
}

function test2(){
    var num = 20;
    switch(true){
        case num >= 0 && num <= 10:
            alert("大于 0 小于等于 10");
            break;
        case num>10&&num<=20:
            alert("大于 10 小于等于 20");
            break;
    }
}
```

### 2.5.3. 循环

#### while 语句

while 语句是先执行条件判断, 再执行循环体内容, 因此, 如果一开始时候就不满足条件, 那么循环体可能根本不会被执行。它的语法如下:

```
while (条件表达式){
    循环体
}
```

#### do-while 语句

do-while 语句是先执行循环体, 然后再进行条件判断。也就是说, 循环体至少会被执行一次。它的语法如下:

```
do{
    循环体
}while(条件表达式);
```

#### for 语句

for 语句是先判断是否满足循环条件，然后再执行循环体的内容。而且在进入循环体之前，还可以初始化变量，并定义循环后要执行的代码。它的语法如下：

```
for (初始化语句; 条件判断语句; 循环后执行的语句) {  
    循环体  
}
```

#### for-in 语句

for 语句是严格的迭代语句，用于数据元素或对象的属性。它的语法如下：

```
for (property in 数组或对象) {  
    循环体  
}
```

**注意：**

- 1) 使用 for-in 语句遍历数组元素的时候，遍历出来的是该数组的下标；
- 2) 使用 for-in 语句遍历对象属性的时候，遍历出来的是该对象的属性名；

## 2.5.4. With 语句

1. with 语句的作用：有了 With 语句，在存取对象属性和方法时就不用重复指定参考对象。
2. 语法格式：

```
with (obj) {  
    操作 obj 的属性语句;  
}
```

例如：

```
<script type="text/javascript">  
    with(document){  
        write("您好 !");  
        write("<br>这个文档的标题是 :\"" + title + "\".");  
        write("<br>这个文档的 URL 是: " + URL);  
        write("<br>现在您不用每次都写出 document 对象的前缀了 !");  
    }  
</script>
```

## 2.6. 函数

### 2.6.1. 函数定义

✚ 方式一：定义函数。

```
function 函数名(形参列表){  
    函数体...  
}
```

✚ 方式二：定义函数变量。

```
var 函数名 = function(形参列表){  
    函数体  
}
```

✚ 方式三：定义匿名函数。

```
(function(形参列表){  
    函数体...  
})(实参列表);
```

✚ 方式四：创建 Function 对象。

```
var 变量名 = new Function(形参列表, 函数体);  
注意：形参、函数体都需要使用双引号引起来。
```

✚ 定义函数需要注意的地方：

- 1) js 函数中，定义形参时候是不能够使用 **var** 关键字声明变量的；
- 2) js 函数是没有返回值类型的，如果函数需要返回数据给调用者，那么使用 **return** 返回数据即可；
- 3) js 中是没有函数重载的概念，后定义的同名函数会直接覆盖前面定义的同名函数；
- 4) js 函数的内部都隐式地维护了一个 **arguments**（数据）对象，给函数传递数据的时候，首先会传递到 **arguments** 对象中，然后再由 **arguments** 对象分配数据给形参；

### 2.6.2. 函数调用

语法格式：函数名(实参列表);

### 2.6.3. 变量的搜索顺序

当在函数中获取变量的时候，浏览器首先会在函数体中查找该变量的定义，如果没有找到就从 **window** 对象中查找。

## 2.6.4. 变量的范围

✚ 全局变量：在函数外面定义的变量；

- 1) 全局变量都是属于 **window** 这个内置对象的.要么是它的属性要么是它的方法；
- 2) 全局变量所在页面的所有的函数都可以访问；
- 3) 函数外定义的变量，从变量的声明开始，到页面关闭时候才结束；

✚ 局部变量：在函数体中定义的变量。

- 1) 局部变量只能够在函数体中使用，函数结束后该变量会自动被撤销；
- 2) 不同函数中定义的变量可以出现重名；
- 3) 如果声明局部变量时候不加上 **var**，那么浏览器首先会在自己函数体中查找该变量，如果找到就赋值，如果没有找到就到 **window** 中查找，如果找到就赋值，如果还是没有找到，就直接定义一个 **window** 级别全局变量；

## 3. 面向对象

JavaScript 面向对象的脚本语言，此时开发者在开发的时候需要找对象，默认提供了内置的对象。也可以根据开发者的需求自己定义对象。

### 3.1. 基本数据类型包装类

为了便于操作基本类型值，ECMAScript 提供了 3 个特殊的引用类型：**Boolean**, **Number**, **String**。它们是引用类型。当读取基本数据类型时，后台就会创建一个对应的基本类型的包装类对象，所以我们在操作基本数据类型时，可以直接调用一些方法。

#### 3.1.1. String

1. 创建字符串的方式：

✚ 方式一：使用双引号；

```
var str = "hello";
```

✚ 方式二：使用 **new** 关键字来创建字符串对象；

```
//通过构造函数创建String 对象
var b = new String("java");
var c = new String("java");
document.write(b + "," + c);
document.write(b == c); //false
//比较字符串的值是否相等
document.write(b.valueOf() == c.valueOf());
```

2. 字符串的常用方法:

- 1) `charAt()`: 返回指定索引值的字符串;
- 2) `charCodeAt()`: 返回指定索引值的字符串的码值;
- 3) `concat()`: 连接多个字符串;
- 4) `fontcolor()`: 把带有 `COLOR` 属性的一个 HTML `<FONT>` 标记放置在 `String` 对象中的文本两端;
- 5) `indexOf()`: 返回指定字符串第一次出现的索引值;
- 6) `lastIndexOf()`: 返回指定字符串最后一次出现的索引值;
- 7) `replace()`: 替换匹配正则表达式的字符串;
- 8) `split()`: 按照字符串或正则表达式切割字符串;
- 9) `substr()`: 返回指定位置开始的指定长度的子字符串;
- 10) `toUpperCase()`: 把字符串中的所有字母转换成大写字母;
- 11) `toLowerCase()`: 把字符串中的所有小写字母转换成小写字母;

例如:

```
//返回在指定位置的字符。
document.write("第五章".charAt(0));
document.write("a".charCodeAt(0));
document.write("第五章".concat("abc"));
document.write("第五章".fontcolor("#ff0000"));
document.write("第五章".indexOf("五"));
document.write("helloworld".replace(/l/g, "L"));
document.write("<br/>");
var names = "jack-lili-lucy".split("-");
for (var temp in names) {
    document.write(names[temp] + "<br/>");
}
document.write("<br/>");
document.write("helloworld".substr(1, 2)); //el
document.write("helloworld".substring(1, 2)); //e
document.write("helloworld".toUpperCase());
document.write(new String("java").toString() == new String("java").toString());
```

### 3.1.2. Number

1. 创建 `Number` 对象的方式

✚ 方式一: `var 变量名 = 数字;`

```
var a = 10;
var b = 3.14;
```

✚ 方式二: `var 变量名 = new Number(数字);`

```
var a = new Number(10);
```

```
var b = new Number(3.14);
```

## 2. Number 对象的方法:

- ✓ `toString()`: 把数字转换成指定进制的字符串;
- ✓ `toFixed()`: 指定保留小数位, 而且还带四舍五入功能;

```
document.write(new Number(12).toString(10) + "<br/>");  
document.write(new Number(12).toString(2) + "<br/>");  
document.write(new Number(12).toString(8) + "<br/>");  
document.write(new Number(12).toString(16) + "<br/>");  
document.write(new Number(12.12345) + "<br/>");  
document.write(new Number(12.12345).toFixed() + "<br/>");  
document.write(new Number(12.12345).toFixed(2) + "<br/>");  
document.write(new Number(12.12345).toFixed(3) + "<br/>");
```

## 3.2. Date 对象

### 1. 创建 Date 对象的方式:

- ✚ 方式一: `new Date()`;
- ✚ 方式二: `new Date(dateVal)`;
  - 1) 如果 `dateVal` 是数字, 则代表指定日期到 1970 年 1 月 1 日之间的毫秒数;
  - 2) 如果 `dateVal` 是字符串, 则代表一个日期格式的字符串;
- ✚ 方式三: `new Date(year, month, date, hours, minutes, seconds)`;

### 2. Date 对象的方法:

- ✓ `getFullYear()`: 获取当前时间到 1900 年之间相隔多少年;
- ✓ `getFullYear()`: 获取本地时间的年份;
- ✓ `getMonth()`: 获取本地时间的月份;
- ✓ `getDate()`: 获取本地时间的月份的日期;
- ✓ `getHours()`: 获取本地时间的小时;
- ✓ `getMinutes()`: 获取本地时间的分钟;
- ✓ `getSeconds()`: 获取本地时间的秒数;
- ✓ `toLocaleString()`: 获取系统时间;

## 3.3. Math 对象

**Math** 对象提供基本数学函数和常数。它的常用方法有:

- ✓ `ceil()`: 向上取整;
- ✓ `floor()`: 向下取整;
- ✓ `random()`: 生成 0-1 之间的随机数, 包括 0 但不包括 1;
- ✓ `round()`: 四舍五入;

```
document.write(Math.ceil(12.34) + "<br/>"); //输出13  
document.write(Math.floor(12.34) + "<br/>"); //输出12  
document.write(Math.round(12.54) + "<br/>"); //输出13
```

```
document.write(Math.round(12.35) + "<br/>"); //输出12
document.write(Math.random() + "<br/>");
```

### 3.4. Array 对象

#### 1. 创建数组的方式

✚ 方式一: `var 数组名 = new Array();`

✚ 方式二: `var 数组名 = new Array(size);`

注意: js 数组的长度是可变的, 它的长度取决于数组中元素的个数。

✚ 方式三: `var 数组名 = new Array(元素 1, 元素 2, ...);`

✚ 方式四: `var 数组名 = [元素 1, 元素 2, ...];`

#### 2. Array 对象的方法:

- ✓ `concat()`: 把多个元素添加到该数组的末尾并返回一个新的数组;
- ✓ `join()`: 指定分隔符把数组中的所有元素拼接成一个字符串并返回;
- ✓ `pop()`: 删除并返回数组中的最后一个元素;
- ✓ `push()`: 往数组添加新的元素并返回数组的长度;
- ✓ `reverse()`: 反转数组的元素;
- ✓ `shift()`: 移除并返回数组中的第一个元素;
- ✓ `slice()`: 返回数组中指定开始和结束位置之间元素所组成的数组;
- ✓ `sort()`: 按照指定的规则对数组进行排序;
- ✓ `splice()`: 移除指定位置的一个或多个元素, 然后在移除位置插入新的元素, 并返回由移除元素所组成的数组;

```
var arr = [1, 2, 3];
//连接两个或更多的数组, 并返回结果。
document.write(arr.concat([100, 30]) + "<br/>");
document.write(arr.concat([100, 30], [60, 90]) + "<br/>");
// 把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。
var b = arr.join("$");
document.write(b + "<br/>");
//删除并返回数组的最后一个元素
document.write(arr.pop() + "<br/>");
//向数组的末尾添加一个或更多元素, 并返回新的长度。
document.write(arr.push(99, 78) + "<br/>");
//把数组转换为字符串, 并返回结果。
document.write(arr.toString() + "<br/>");
```

排序:

```
var arr = [100, 9, 20, 3, 7, 96];
document.write(arr + "<br/>");
//如果调用该方法时没有使用参数, 将按字母顺序对数组中的元素进行排序
arr.sort();
document.write(arr + "<br/>");
```

```
//指定比较方式
a.sort(compareTo);
document.write(a + "<br/>");
function compareTo(a, b) {
    return a - b;
}
```

### 3. JavaScript 数组与 Java 数组的区别:

- 1) Java 数组的长度是固定的，而 JavaScript 数组的长度是可变的;
- 2) Java 数组元素的数据类型是相同的，而 JavaScript 数组中的元素可以是任意类型;

## 3.5. 自定义对象

1. js 中没有类的概念，只要有函数就可以创建对象。

2. 创建对象的方式:

✚ 方式一: 使用无参的构造函数创建对象;

```
<script type="text/javascript">
//自定义对象
function Person() {};
//创建对象p
var p = new Person();
//设置对象的属性
p.id = 110;
p.name = "张三";
//设置对象的动作
p.say = function() {
    alert("我的名字是: " + this.name);
}

document.write("编号: " + p.id + ", 姓名: " + p.name);
p.say();
</script>
```

✚ 方式二: 使用带参的构造函数创建对象 (推荐);

```
<script type="text/javascript">
//自定义对象
function Person(id, name) {
    this.id = id;
    this.name = name;
    this.say = function() {
        alert("我的名字是: " + this.name);
    };
};
//创建对象p
var p = new Person(110, "张三");

document.write("编号: " + p.id + ", 姓名: " + p.name);
p.say();
</script>
```

✚ 方式三: 使用 Object 函数创建对象;



```

<script type="text/javascript">
    //创建对象p
    var p = new Object();
    //设置对象的属性
    p.id = 110;
    p.name = "张三";
    //设置对象的动作
    p.say = function() {
        alert("我的名字是：" + this.name);
    }

    document.write("编号：" + p.id + "，姓名：" + p.name);
    p.say();
</script>

```

✚ 方式四：使用字面量的方式创建对象；

```

<script type="text/javascript">
    //创建对象p
    var p = {
        //设置对象的属性
        id:110,
        name:"张三",
        //设置对象的动作
        say:function() {
            alert("我的名字是：" + this.name);
        }
    };

    document.write("编号：" + p.id + "，姓名：" + p.name);
    p.say();
</script>

```

**需求：自定义一个数组的工具类。**

```

function ArrayTool(){}
var tool = new ArrayTool();

//求最大值
tool.getMax = function(arr)
{
    var max = 0;
    for(var x=1; x<arr.length; x++)
    {
        if(arr[x]>arr[max])
            max = x;
    }
    return arr[max];
};

//求最小值
tool.getMin = function(arr)
{
    var min = arr[0];

```

```
for(var x=1; x<arr.length; x++)
{
    if(arr[x]<min)
        min = arr[x];
}
return min;
}

//查找数组元素
tool.binarySearch = function(arr, key)
{
    var min,max,mid;

    min = 0;
    max = arr.length-1;

    while(min<=max)
    {
        mid = (min+max)>>1;

        if(key>arr[mid])
            min = mid + 1;
        else if(key<arr[mid])
            max = mid - 1;
        else
            return mid;
    }
    return -1;
}
```

### 3.6. Prototype 属性

“prototype”字面翻译是“原型”，是 javascript 实现继承的主要手段。粗略来说就是：prototype 是 javascript 中的函数(function)的一个保留属性，并且它的值是一个对象（我们可以称这个对象为“prototype 对象”）。

#### 注意事项：

- 1) prototype 是函数的一个必备属性；
- 2) prototype 的值是一个对象；
- 3) 可以任意修改函数的 prototype 属性的值；
- 4) 一个对象会自动拥有 prototype 的所有成员属性和方法；

例如：求数组的最大值。

```
function array_max(){
    var i, max = this[0];
    for (i = 1; i < this.length; i++) {
        if (max < this[i]){
            max = this[i];
        }
    }
    return max;
}
Array.prototype.max = array_max;
var x = new Array(1, 2, 3, 4, 5, 6);
var y = x.max();
document.write("数组的最大值: " + y);
```

**练习 1：把数组的工具方法添加到 `Array` 对象中。**

```
//获取元素在数组中的下标位置
Array.prototype.getIndex = function(element)
{
    for(var x=0; x<this.length; x++)
    {
        if(this[x]==element)
            return x;
    }
    return -1;
}

//求最小值
Array.prototype.getMin = function()
{
    var min = this[0];
    for(var x=1; x<this.length;x++)
    {
        if(this[x]<min)
            min = this[x];
    }
    return min;
}
```

**练习 2：扩展 `String` 对象的功能，添加以下方法：**

**1) 去除字符串前后空格；**

2) 把字符串转换成数组;

3) 将字符串进行反转;

```
//去除字符串两端的空格。
String.prototype.trim = function(){
    var start,end;
    start = 0;
    end = this.length-1;

    while(start<=end && this.charAt(start)==' '){
        start++;
    }
    while(start<=end && this.charAt(end)==' '){
        end--;
    }
    return this.substring(start,end+1);
}

//将字符串转换成数组。
String.prototype.toCharArray = function()
{
    var arr = new Array(this.length);
    for(var x=0; x<this.length; x++){
        arr[x] = this.charAt(x);
    }
    return arr;
}

//将字符串进行反转。
String.prototype.reverseString = function()
{
    var arr = this.toCharArray();
    return arr.reverse().join("");
}
```

## 4. BOM 编程

### 4.1. BOM 编程基础

BOM(Browser Object Model)的全称是浏览器对象模型。浏览器对象模型把浏览器的各个部分都使用了一个对象进行描述。如果我们需要操作浏览器的一些属性,那么就需要通过浏览器对象的对象进行操作。

常见的 BOM 对象：window、location、screen。

### 4.1.1. window 对象

1. window 对象代表一个新打开的窗口。
2. window 对象的方法：
  - ✓ open(): 打开一个新的窗口；
  - ✓ resizeTo(): 将窗口的大小更改为指定的高度和宽度（只支持 IE 浏览器）；
  - ✓ moveBy(): 相对于原来窗口移动 x 和 y 个像素值（只支持 IE 浏览器）；
  - ✓ moveTo(): 将窗口的左上角移动到屏幕的 x 和 y 的位置（只支持 IE 浏览器）；
  - ✓ setInterval(): 每隔指定的毫秒值后执行指定的代码；
  - ✓ clearInterval(): 取消指定的定时任务；
  - ✓ setTimeout(): 经过指定毫秒值后执行指定的代码的一次；

注意：使用 window 对象的任何属性和方法都可以省略不写 window 关键字。

### 4.1.2. location 对象

1. location 对象代表地址栏对象。
2. location 对象的属性和方法：
  - ✓ href: 设置或获取地址栏的内容；
  - ✓ reload(): 刷新当前页面；

### 4.1.3. screen 对象

1. screen 对象代表了整个屏幕对象。
2. screen 对象的属性：
  - ✓ availHeight: 获取屏幕的工作区域的高度（不包含任务栏区域）；
  - ✓ availWidth: 获取屏幕的工作区域的宽度（不包含任务栏区域）；
  - ✓ height: 获取屏幕的垂直分辨率；
  - ✓ width: 获取屏幕的水平分辨率；

### 4.1.4. document 对象

1. document 代表整个文档页面。
2. document 对象的集合：
  - ✓ all: 获取页面所有元素对象；
  - ✓ forms: 获取页面所有表单对象；
  - ✓ images: 获取页面所有图片对象；
  - ✓ link: 获取所有超链接或 area 对象；

## 4.1.5 history 对象

1. history 代表浏览器的历史列表。
2. history 对象的方法：
  - ✓ back: 返回历史列表中的前一个 URL（与在浏览器点击后退按钮相同）；
  - ✓ forward: 访问历史列表中下一个 URL（与在浏览器中点击按钮向前相同）；
  - ✓ go: 从历史列表中加载 URL；

## 4.2. 事件

基本上所有的 HTML 元素中都可以指定事件属性。所有的事件属性都是以 on 开头，后面的是事件的触发方式，如：onclick、onkeydown 等等。

### 4.2.1. 注册事件的方式

1. 方式一：直接在 html 元素上进行注册；

```
<input type="button" onclick="事件处理函数"/>
```

2. 方式二：可以在 js 代码找到对应的对象再注册；

```
<script>
  html 元素对象.onclick = function() {
    事件处理代码...
  }
</script>
```

方式一与方式二的区别？

方式一可以给事件处理函数传入参数，但是方式二就不能够传入参数。

### 4.2.2. 常见的事件类型

- ✓ 鼠标事件：

onclick 在用户用鼠标左键单击对象时触发。  
ondblclick 当用户双击对象时触发。  
onmousedown 当用户用任何鼠标按钮单击对象时触发。  
onmouseup 当用户在鼠标位于对象之上时释放鼠标按钮时触发。  
onmouseout 当用户将鼠标指针移出对象边界时触发。  
onmousemove 当用户将鼠标划过对象时触发。

- ✓ 焦点相关的：

onblur 在对象失去输入焦点时触发。  
onfocus 当对象获得焦点时触发。

- ✓ 其他：

<p><b>onchange</b> 当对象或选中区的内容改变时触发。</p> <p><b>onload</b> 在浏览器完成对象的装载后立即触发。</p> <p><b>onsubmit</b> 当表单将要被提交时触发。</p>
--------------------------------------------------------------------------------------------------------------------

## 5. DOM

### 5.1. DOM 简介

DOM (Document Object Model) 文档对象模型，定义了访问和处理 HTML 文档的标准方法。浏览器在解析HTML页面标记的时候，其实不是按照一行一行读取并解析的，而是将HTML页面中的每一个标记按照顺序在内存中组建一颗DOM树，组建好之后，按照树的结构将页面显示在浏览器的窗口中。

### 5.2. 节点层次

#### 5.2.1. 什么是节点

根据 DOM，HTML文档中的每个成分都是一个节点。

- ✓ 整个文档是一个文档节点；
- ✓ 每个 HTML 标签是一个元素节点；
- ✓ 包含在 HTML 元素中的文本是文本节点；
- ✓ 每一个 HTML 属性是一个属性节点；
- ✓ 注释属于注释节点；

#### 5.2.2. 节点的层级关系

HTML 文档中的所有节点组成了一个文档树。HTML 文档中的每个元素、属性、文本等都代表着树中的一个节点。树起始于文档节点，并由此继续伸出枝条，直到处于这棵树最低级别的所有文本节点为止。

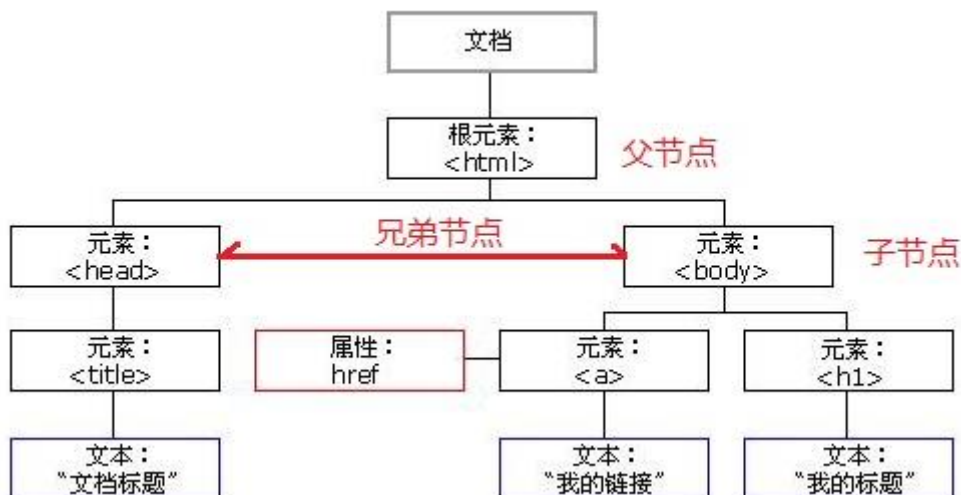


图 5-1 节点的层级关系

## 5.3. 操作节点

### 5.3.1. 查找节点

#### 1. 获取节点集合

- ✓ `document.all`: 获取所有标签的集合;
- ✓ `document.links`: 获取所有 `a` 标签的集合;
- ✓ `document.images`: 获取页面所有图片对象;
- ✓ `document.forms`: 获取所有的表单对象;

#### 2. 根据标签名查找节点

```
document.getElementsByTagName("标签名"); //返回指定标签名的 html 元素的集合
```

#### 3. 根据标签属性查找节点

```
document.getElementById("id 属性"); //返回指定 id 属性的 html 元素对象  
document.getElementsByName("name 属性"); //返回指定 name 属性的 html 元素的集合
```

#### 4. 根据关系查找节点

```
parentNode: 获取当前节点的父节点对象;  
childNodes: 获取当前节点的所有下一级子节点对象;  
firstChild: 获取当前节点的第一个子节点;  
lastChild: 获取当前节点的最后一个子节点;  
nextSibling: 获取当前节点的下一个兄弟节点;  
previousSibling: 获取当前节点的上一个兄弟节点;
```



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script type="text/javascript">
//节点和节点之间的关系.
//获取dom树
var dom = window.document;
//获取指定id 的标签节点.
function test() {
    var form = dom.getElementById("form1");
    //获取父节点.
    //alert(form.parentNode.nodeName);
    // 获取子节点(Node 包含 文本,注释,标签)
    var childArr = form.childNodes;
    //alert(childArr.length);
    /*
    for (var i = 0; i < childArr.length; i++) {
        alert(childArr[i]);
    }
    */
    // 获取第一个孩子.
    var first = form.firstChild;
    //alert(first);
    //最后一个孩子.
    var last = form.lastChild;
    //alert(last);
    // 获取下兄弟(获取弟弟)
    var sibling = form.nextSibling;
    //alert(sibling.nodeName);
    // 获取大哥
    var previous = form.previousSibling;
    alert(previous.nodeName);
}
test();
</script>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />

<title>javascript</title>
</head>
<body onmousemove="test(this)">
    <a>哈哈</a>
    <form id="form1">
```

```
<label>姓名:</label>
<input type="text" />
</form>
</body>
</html>
```

### 5.3.2. 创建新节点

1. 创建新节点:

```
document.createElement("标签名");
```

2. 设置属性:

```
element.setAttribute("属性名", "属性值");
```

3. 添加元素到 element 的后面:


```
element.appendChild(e);
```

4. 添加元素到 element 中, 在 child 之前:

```
element.insertBefore(new, child);
```

5. 删除 element 中的 child 节点:

```
element.removeChild(child);
```

 **案例 1:** 生成二级城市联动菜单。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style type="text/css">
select{
    width:100px;
}
</style>
<script type="text/javascript">
    function selcity()
    {
        //定义数据对应关系
        //城市有很多，所以通过数组存储。每一个省对应一个城市数组，怎么建立对应关系
        呢？
        //每一个省都有自己的角标。通过角标和数组建立对应关系，这就是二维数组。

        var arr = [['--选择城市--'], ['海淀区', '朝阳区', '东城区', '西城区']
            , ['沈阳', '大连', '鞍山', '抚顺']
            , ['济南', '青岛', '烟台', '威海']
            , ['石家庄', '廊坊', '唐山', '秦皇岛']];

        //获取选择的省的角标。
        var selNode = document.getElementById("selid");
        var index = selNode.selectedIndex;

        var cities = arr[index];

        var subSelNode = document.getElementById("subselid");

        //有更简单清除方式，只要改变下拉菜单的长度即可。
        subSelNode.options.length = 0;
        /*
        //清除上一次选择的子菜单内容。
        for(var x=1; x<subSelNode.options.length;)
        {

            alert(subSelNode.options.length+"..." +subSelNode.options[x].innerHTML+"..." +x);
            subSelNode.removeChild(subSelNode.options[x]);
        }
    }

```

```

        */

        for(var x=0; x<cities.length; x++)
        {
            var optNode = document.createElement("option");

            optNode.innerHTML = cities[x];

            subselid.appendChild(optNode);
        }
    }
</script>
</head>
<body>
<select id="selid" onchange="selcity()">
    <option>--选择省市--</option>
    <option>北京市</option>
    <option>辽宁省</option>
    <option>山东省</option>
    <option>河北省</option>
</select>
<select id="subselid">
    <option>--选择城市--</option>
</select>
</body>
</html>

```

## 案例 2：动态生成年、月、日字段

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script type="text/javascript">
/**
 * @author Administrator
 */
//通过js创建年,月,日
//获取Dom
var dom = window.document;

```

```

function myYear() {
    //获取年的select
    var year = dom.getElementById("year");
    //创建年
    var minYear = 1900;
    var maxYear = new Date().getFullYear();
    for (var i = minYear; i <= maxYear; i++) {
        //创建Option
        var opt = dom.createElement("option");
        //设置Option, 标签体.
        opt.innerHTML = i;
        opt.value = i;
        //挂载节点
        year.appendChild(opt);
    }
}

function myMonth() {
    var month = dom.getElementById("month");
    //创建月
    for (var i = 1; i <= 12; i++) {
        //创建Option
        var opt = dom.createElement("option");
        //设置Option, 标签体.
        if (i < 10) {
            opt.innerHTML = "0" + i;
            opt.value = i;
        } else {
            opt.innerHTML = i;
            opt.value = i;
        }
        month.appendChild(opt);
    }
    month.onchange = myDay;
}

function myDay() {
    clear();
    //创建天
    // 大月1 3 5 7 8 10 12 ,小月4 6 9 11    闰年2月 非闰年的2月
    //获取年
    var year = dom.getElementById("year").value;
    //获取月
    var month = dom.getElementById("month").value;
    if (year == "") {
        alert("请选择年");
    }
}

```

```

        return;
    }
    if (month == "") {
        alert("请选择月");
        return;
    }
    //获取天select
    var day = dom.getElementById("day");
    //一个月至少有28天.
    for (var i = 1; i <= 28; i++) {
        var opt = dom.createElement("option");
        if (i < 10) {
            opt.innerHTML = "0" + i;
            opt.value = "0" + i;
        } else {
            opt.innerHTML = i;
            opt.value = i;
        }
        day.appendChild(opt);
    }
    //大月
    var isBigMonth = month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12;
    //小月
    var isSmallMonth = month == 4 || month == 6 || month == 9 || month == 11;
    //闰年    可以整除4但不能整除100 或者 年份可以整除400.
    var isLeapYear = (year % 4 == 0 && year % 100 != 0) || year % 400 == 0;

    //判断,如果是大月,添加3天
    if (isBigMonth) {
        //添加3天
        for (var i = 29; i <= 31; i++) {
            var opt = dom.createElement("option");
            opt.innerHTML = i;
            opt.value = i;
            day.appendChild(opt);
        }
    } else if (isSmallMonth) {
        //添加2天
        for (var i = 29; i <= 30; i++) {
            var opt = dom.createElement("option");
            opt.innerHTML = i;
            opt.value = i;
        }
    }
}

```

```

        day.appendChild(opt);
    }
} else if (isLeapYear) {
    //如果是闰年,添加一天.专门处理闰年2月.
    var opt = dom.createElement("option");
    opt.innerHTML = 29;
    opt.value = 29;
    day.appendChild(opt);
}
}
function clear() {
    var day = dom.getElementById("day");
    var optArr = day.childNodes;
    for (var i = optArr.length - 1; i >= 0; i--) {
        day.removeChild(optArr[i]);
    }
}
function getBirthday() {
    //获取Dom
    var dom = window.document;
    myYear();
    myMonth();
}
getBirthday();
</script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>javascript</title>
</head>
<body>
    生日:
        <select id="year">
            <option>年</option>
        </select>
        <select id="month">
            <option>月</option>
        </select>
        <select id="day">
            <option>日</option>
        </select>

</body>
</html>

```

### 5.3.3. 操作元素的 CSS 样式

Style 对象代表一个单独的样式声明，可以通过元素访问 Style 对象。

1. 方式一：设置元素的 CSS 样式：

```
document.getElementById("id").style.property="值"
```

2. 方式二：设置元素的 Class 属性：

```
document.getElementById("id").className = "值";
```

🧩 练习：随机生产一个四位数的验证码。



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script type="text/javascript">
/**
 * @author Administrator
 */
//验证码 ,4位的,由字符,数字组成.
function createCode() {
    var datas = ["A", "B", "C", "D", "E", "F", "G", "H", "Z", "0", "1",
"2", "3", "4", "5", "6", "7", "8", "9"];
    //随机的从数组中取出4个元素.
    var mess = "";
    var index = 0;
    for (var i = 0; i < 4; i++) {
        //生成随机数.而且是在数组的长度范围内.
        //0-9之间的随机数. Math.floor(Math.random()*10)
        //0到数组长度(不包含)之间的浮点数.,向下取整,
        var index = Math.floor(Math.random() * datas.length);
        mess += datas[index];
    };
    //
    var codeSpan = window.document.getElementById("codeSpan");
    codeSpan.style.color = "red";
    codeSpan.style.fontSize = "20px";
    codeSpan.style.background = "gray";
    codeSpan.style.fontWeight = "900";
    codeSpan.style.fontStyle = "italic";
    codeSpan.style.textDecoration = "line-through";
    codeSpan.innerHTML = mess;
    codeSpan.value = mess;
}
createCode();
</script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>javascript</title>
</head>
<body>
    <span id="codeSpan"></span><a href="#" onclick="createCode()">看不
清楚</a>

</body>
</html>
```

## 6. 正则表达式

### 6.1. 创建正则表达式的方式

- ✓ 方式一：/正则表达式/模式
- ✓ 方式二：new RegExp(“正则表达式”, “模式”);

### 6.2. 正则表达式的方法

test(): 使用正则表达式去匹配字符串，如果匹配成功返回 true，否则返回 false;

注意：在 js 中写正则表达式，建议加上边界匹配器。

exec(): 根据正则表达式去查找字符串中符合规则的内容;

### 6.3. 模式

i 模式：忽略大小写;

g 模式：全文查找出现的所有 pattern;

🔗 练习：使用 js 实现表单验证的功能。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script>

//用户名的规则： 第一位是字母，只有由数字与字母 组成，6位。
function checkName(){
    //获取到了用户名的值
    var userName = document.getElementById("username").value;
    var userSpan = document.getElementById("userId");
    var reg = /^[a-z][a-z0-9]{5}$/i;
    if(reg.test(userName)){
        //符合规则
        userSpan.innerHTML="正确".fontcolor("green");
        return true;
    }else{
        //不符合规则
        userSpan.innerHTML="错误".fontcolor("red");
    }
}
```

```

        return false;
    }
}

//校验密码 6位
function checkPass(){
    var password = document.getElementById("pwd").value;
    if(password.length>0){
        var reg = /^\\w{6}$\\//;
        var passSPan = document.getElementById("passId");
        if(reg.test(password)){
            //符合规则
            passSPan.innerHTML="正确".fontcolor("green");
            return true;
        }else{
            //不符合规则
            passSPan.innerHTML="错误".fontcolor("red");
            return false;
        }
    }
}

//检验密码是否正确
function ensurepass(){
    var password1 = document.getElementById("pwd").value; //第一次
    输入的密码
    var password2 = document.getElementById("ensurepwd").value;
    if(password2.length>0){
        var enSpan = document.getElementById("ensure");
        if(password1.valueOf()==password2.valueOf()){
            enSpan.innerHTML="正确".fontcolor("green");
            return true;
        }else{
            enSpan.innerHTML="错误".fontcolor("red");
            return false;
        }
    }
}

//校验邮箱

```

```

function checkEmail(){
    var email = document.getElementById("email").value;
    var reg = /^[a-z0-9]\w+@[a-z0-9]{2,3}(\.[a-z]{2,3}){1,2}$/i;
// .com .com.cn
    var emailspan = document.getElementById("emailspan");
    alert(reg.test(email));
    if(reg.test(email)){
        //符合规则
        emailspan.innerHTML="正确".fontcolor("green");

        return true;
    }else{
        //不符合规则
        emailspan.innerHTML="错误".fontcolor("red");
        return false;
    }
}

//校验兴趣爱好： 至少要算中其中 的一个。
function checkHoby(){
    var likes = document.getElementsByName("like");
    var hobySpan =document.getElementById("hobbySpan")
    var flag = false;
    for(var i = 0 ; i<likes.length ; i++){
        if(likes[i].checked){
            flag =true;
            break;
        }
    }

    if(flag){
        //符合规则
        hobySpan.innerHTML="正确".fontcolor("green");
        return true;
    }else{
        //不符合规则
        hobySpan.innerHTML="错误".fontcolor("red");
        return false;
    }
}

```

//总体校验表单是否可以提交了 如果返回的true表单才可以提交。 上面的表单项必须要每个都填写正确。

```
function checkForm(){
    var userName = checkName();
    var pass = checkPass();
    var ensure = ensurepass();
    var email = checkEmail();
    var hobby = checkHoby();
    if(userName&&pass&&ensure&&email&&hoby){
        return true;
    }else{
        return false;
    }
}

}

</script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>正则表达式</title>
</head>
<body>
<form action="success.html" method="get" onsubmit="return checkForm()">
<!--如果表单提交时候触发的方法返回是false,那么该表单不允许提交, 如果返回的是true
允许提交 -->
    <table border="1px" width="50%" align="center"
cellspacing="0px" cellpadding="3px">
        <tr>
            <td width="25%">姓名:</td>
            <td>
                <input type="text" name="username" id="username"
onblur="checkName()" />
                <span id="userId"></span>
            </td>
        </tr>
        <tr>
            <td>密码:</td><td>
                <input type="password" name="pwd" id="pwd"
onblur="checkPass()" />
                <span id="passId"></span>
            </td>
        </tr>
        <tr>
            <td>确认密码:</td><td>
                <input type="password" name="ensurepwd" id="ensurepwd">
            </td>
        </tr>
    </table>
</body>
</html>
```

```

onblur="ensurepass()" />                                <span id="ensure"></span>
        </td>
    </tr>
    <tr>
        <td>邮箱</td><td>
            <input type="text" name="email" id="email"
onblur="checkEmail()" />
            <span id="emailspan"></span>

        </td>
    </tr>
    <tr>
        <td>性别</td><td>
            <input type="radio" checked="ture" name="gender"
id="male" value="male" />
            男
            <input type="radio" name="gender"
value="female" />

            女</td>
    </tr>

    <tr>
        <td>爱好:</td><td>
            <input type="checkbox" checked="checked"
name="like" />
            eat
            <input type="checkbox" name="like" />
            sleep
            <input type="checkbox" name="like" />
            play
            <span id="hobbySpan"></span>
        </td>
    </tr>
    <tr>
        <td>城市</td><td>
            <select name="city" id="city">
                <option value=""> 请选择</option>
                <option value="bj"> 北京 </option>
                <option value="gz"> 广州 </option>
                <option value="sh"> 上海 </option>
            </select>

        </td>
    </tr>

```

```
        </tr>
        <tr>
            <td>自我介绍</td><td>
                <textarea
cols="15" rows="5" name="myInfo" id="myInfo"></textarea></td>
            </tr>
            <tr align="center">
                <td colspan="2"><!--提交按钮-->
                    <input type="submit"/>
                </td>
            </tr>
        </table>
    </form>
</body>
</html>
```