JavaScript基础

回顾

```
CSS概述：Cascading Style Sheets 层叠样式表，控制网页样式
CSS作用:控制网页样式，美化网页，实现内容和样式的分离  版本 CSS3
CSS语法：
    选择器：网页中的html元素
    选择器{
        声明;
        声明;
    }
使用方式
    三种：
        内联  样式和页面元素通过style属性关联在一起
        内部  在head标签中 <style type="text/css"> </style>
        外部  独立的css文件 多个页面使用同一个样式  链接式link（推荐）  导入式
选择器
    基本选择器 ：  标签选择器、id选择器、class选择器
    属性选择器：  input[type]  a[href]
    层级选择器：后代  div p   子代  div>p  紧邻兄台 div+p   通用兄弟 div~p
    伪元素选择器  a:link  a:visited  a:hover a：active
    其他选择器  *   h2,h3,h4
CSS属性
    文本字体
        字体：font-size 大小  font-weight粗细   font-style 正常 斜体 italic  font-family
字体类型
            font: swsf
        文本：color 颜色  text-indent 缩进   text-decoration 文本修饰  text-align  word-
spacing
        letter-spacing
    背景
        background-color 背景颜色
        background-image 背景图片
        background-repeat 平铺方式
        background-size 背景大小
        background-position 背景位置
        background:
    列表
        list-style-type
        list-style-position
        list-style-image
        list-style:
    尺寸显示和轮廓
        width
        height
        display :none  block  inline  inline-block
        outline
    浮动
        float :left  right
        clear: 清除浮动影响
    定位
        position:  static静态  relative相对  absolute 绝对 fixed 固定
    盒子模型
        border 边框
```

```
        padding  内边距
        margin  外边距

        宽度=width+ 2*margin+2*padding+2*border
        高度=height+ 2*margin+2*padding+2*border

    CSS3扩展
        border-radius 圆角
        box-shadow 盒子阴影
        background-size
        background-image
        text-shadow
```

今日内容

```
1、JavaScript概述
2、JavaScript数据类型
3、JavaScript运算符
4、JavaScript条件语句
5、JavaScript循环语句
6、JavaScript函数
```

教学目标

```
1、了解JavaScript概述
2、掌握JavaScript数据类型
3、掌握JavaScript运算符
4、掌握JavaScript条件语句
5、掌握JavaScript循环语句
6、掌握JavaScript函数
```

第一节 JavaScript概述

1.1 JavaScript简介

JavaScript(LiveScript)一种解释性脚本语言，是一种动态类型、弱类型、基于原型继承的语言，内置支持类型。它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML（标准通用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。

1.2 JavaScript发展史

它最初由Netscape的Brendan Eich设计。JavaScript是甲骨文公司的注册商标。Ecma国际以JavaScript为基础制定了ECMAScript标准。JavaScript也可以用于其他场合，如服务器端编程。**完整的JavaScript实现包含三个部分：ECMAScript，文档对象模型（DOM Document Object Model），浏览器对象模型(BOM Browser Object Model)。**

Netscape在最初将其脚本语言命名为LiveScript，后来Netscape在与Sun合作之后将其改名为JavaScript。JavaScript最初受Java启发而开始设计的，目的之一就是“看上去像Java”，因此语法上有类似之处，一些名称和命名规范也借自Java。但JavaScript的主要设计原则源自Self和Scheme。JavaScript与Java名称上的近似，是当时Netscape为了营销考虑与Sun微系统达成协议的结果。为了取得技术优势，微软推出了JScript来迎战JavaScript的脚本语言。为了互用性，Ecma国际（前身为欧洲计算机制造商协会）创建了ECMA-262标准（ECMAScript）。两者都属于ECMAScript的实现。尽管JavaScript作为给非程序人员的脚本语言，而非作为给程序人员的脚本语言来推广和宣传，但是JavaScript具有非常丰富的特性。

发展初期，JavaScript的标准并未确定，同期有Netscape的JavaScript，微软的JScript和CEnvi的ScriptEase三足鼎立。1997年，在ECMA（欧洲计算机制造商协会）的协调下，由Netscape、Sun、微软、Borland组成的工作组确定统一标准：ECMA-262。

第二节 JavaScript基本语法

入门程序

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>javascrit入门</title>
    </head>
    <body>
        <script type="text/javascript">
            //alert("hello world");//弹出窗口显示
            document.write("hello world");//页面中显示
        </script>
     </body>
</html>
```

2.1 变量声明

在JavaScript中，任何变量都用var关键字来声明，var是variable的缩写。

```
var num; //声明变量
num=10;//赋值

var s="zhangsan";//声明的同时赋值
```

var是声明关键字，a是变量名，语句以分号结尾，分号可以省略。

这里值得注意的是，JavaScript中的关键字，不可以作为变量名。就像在Java中你不可以写"int int=1;"一样。

JavaScript的关键字和保留字：

```
abstract、else、instanceof、super、boolean、enum、int、switch、break、export、interface、
synchronized、byte、extends、let、this、case、false、long、throw、catch、final、native、
throws、char、finally、new、transient、class、float、null、true、const、for、package、
try、continue、function、private、typeof、debugger、goto、protected、var、default、if、
public、void、delete、implements、return、volatile、do、import、short、while、double、in、
static、with。
```

之后的课程还会有文档对象相关的关键字等。

JavaScript中的注释有两种：

```
单行注释//
多行注释/* ... */
```

2.2 数据类型

2.2.1 基本（原始）类型

变量的基本类型有number、string、boolean、undefined、null五种。 u n n s b

你可以使用:

```
var a=1;
```

来声明一个数字Number类型。

你可以使用:

```
var a="1";
```

来声明一个字符串String类型。

你可以使用

```
var a=false;
```

来声明一个布尔Boolean类型。

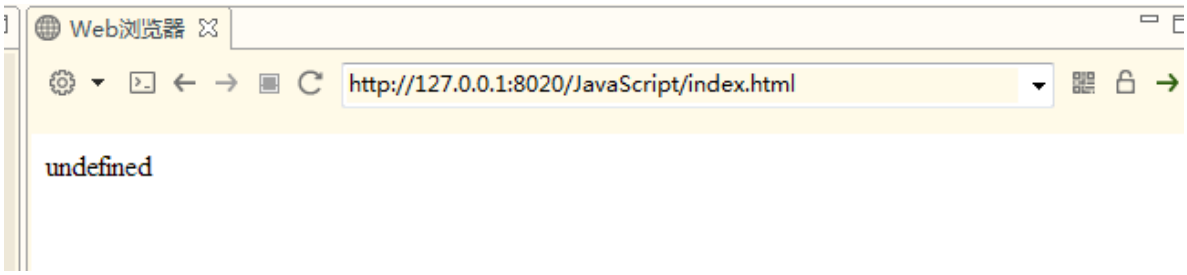
在Java中，当一个变量未被初始化的时候，Java中是null或者基本数据类型的默认值。

在JavaScript中，当一个变量未被初始化的时候，它的值为undefined。

下面是演示undefined的情况:

```
var a;
document.write(a);
```

运行结果为:



最后，当一个引用不存在时，它为Null。这个现象我们在之后的引用类型时再详细探讨。

2.2.2 引用（对象）类型

在Java中需要类定义，然后在实例对象:

```
public class Student{
  public int id;
  public String name;
  public int age;
}
public class Test{
    public static void main(String [] args){
        Student student=new Student();
        student.id=1;
        student.name="张三";
        student.age=18;
    }
}
```

在JavaScript中对象创建

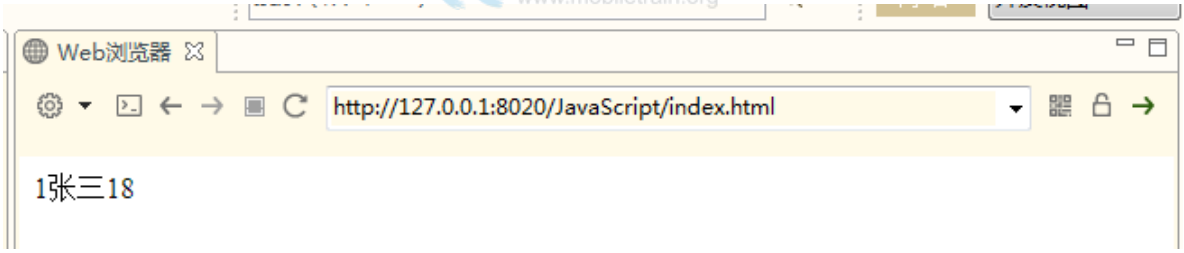
1 创建对象方式

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>javascript引用(对象)类型</title>
        <script type="text/javascript">
            //1创建(类)对象定义
            function Person(){
                //属性
                this.age;
                this.name;
                this.address;
                //函数(匿名函数)
                this.show=function(){
                    document.write(this.age+"...."+this.name+"....."+this.address);

                }
            }

            /*function Student(){

            }
            Student.prototype=new Person();*/
            //2创建对象
            var zhangsan=new Person();
            zhangsan.age=20;
            zhangsan.name="张三";
            zhangsan.address="北京";
            zhangsan.show();

            /*var lisi=new Student();
            lisi.age=30;
            lisi.name="李四";
            lisi.address="上海";
            lisi.show();*/




        </script>
    </head>
    <body>
    </body>
</html>
```

2 使用json对象直接写出来：

```
var student={id:1,name:"张三",age:18};
document.write(student.id);
document.write(student.name);
document.write(student.age);
```

运行结果：



事实上，student被赋值为了一个JSON，JSON就是我们在Java基础阶段学过的，全称是JavaScript Object Notation，叫做JavaScript对象标记，也就是说，在JavaScript中，JSON是用于标记一个对象的。

数组类型

数组就是和我们之前理解的数组概念一致，而在JavaScript中成为Array类型。

```
var cities=new Array(3);
document.write(cities.length);
//添加数据
cities[0]=10;
cities[1]=20;
cities[2]=30;
cities[3]=40;
cities[4]=50;
document.write("<br/>");
document.write(cities.length);
```

我们说JSON可以标记一个对象，那么它同样可以标记一个数组，就是Java基础时我们学过的JSONArray。

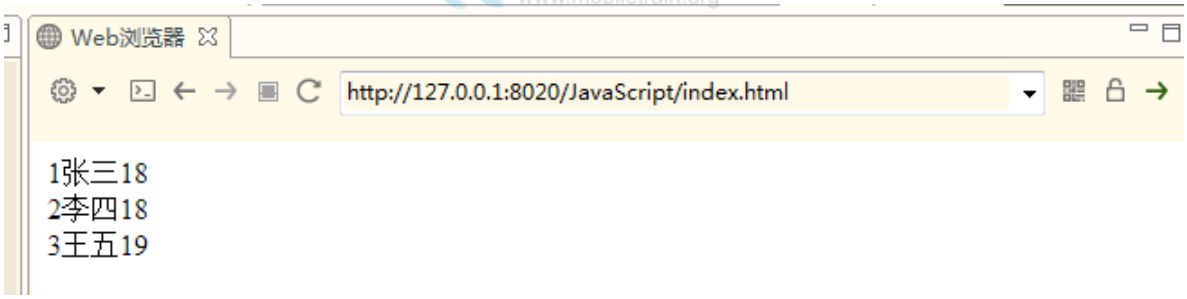
```
var a=[1,2,3,4];
```

上述代码，我们说a是一个数组，在a中角标为0的元素是1。可以说这很简单。

接下来我们来尝试把之前的JSON放入数组中：

```
    //我是注释
    /*我也是注释*/
    var students = [
    {id: 1,name: "张三",age: 18},
    {id: 2,name: "李四",age: 18},
    {id: 3,name: "王五",age: 19}
    ];
    document.write(students[0].id);
    document.write(students[0].name);
    document.write(students[0].age);
    document.write("<br>");//这个是html的换行的意思
    document.write(students[1].id);
    document.write(students[1].name);
    document.write(students[1].age);
    document.write("<br>");
    document.write(students[2].id);
    document.write(students[2].name);
    document.write(students[2].age);
```

运行结果：



我们看到，访问students这个数组，第0个，第1个，第2个元素，都可以。

2.2.3 JavaScript的三种使用方式

第一种方式：在标签中，script可以放在网页中任何位置。

```
<script type="text/javascript">
    var num=10;
    var d=new Date();
    document.write(num);
</script>
```

第二种方式：使用外部JavaScript文件，把js代码放入单独的文件中 ，这个文件的扩展名.js

```
<script type="text/javascript" src="js/myjs.js"></script>
```

第三种方式： 放在标签中的事件属性中,常见事件 ,onclick

```
<input type="button" value="你点我啊" onclick="alert('你点我干嘛')" />
```

2.3 运算符

2.3.1 算术运算符

+、-、*、/、%、++、--

```
var a=1;
var b=2;
a+b;//相加
a-b;//相减
a*b;//相乘
a/b;//相除
a%b;//求余

a++;//自增
b--;//自减
//上述规则和Java一样。
```

2.3.2 赋值运算符

=、+=、-=、/=、*=、%=

```
var a=10;
var b=20;
a=b;//赋值
a+=b;//相加后赋值
a-=b;//相减后赋值
a/=b;//相除后赋值
a*=b;//相乘后赋值
a%=b;//求余后赋值
```

2.3.3 逻辑运算符

与、或、非

&&、||、！

这个就是像我们Java一样，对于真假的boolean值可以进行三种逻辑命题的运算。

```
var a=false;
var b=true;
//非的逻辑
//!a->true;
//!b->false;
//与的逻辑
//a&&a->false;
//a&&b->false;
//b&&a->false;
//b&&b->true;
//或的逻辑
//a||a->false;
//a||b->true;
//b||a->true;
//b||b->true;
```

2.3.4 关系运算符

==相等

<小于

<=小于或等于

>大于

>=大于或等于

!=不等于

===全相等

```
var a=1;
var b=2;
//a==a->true
//a==b->false
//a<b->false
//a<=b->false
//a>b->true
//a>=b->true
//a!=b->true
//a===b->false
```

```
//这里三个等于“===”和两个等于“==”区别：
//前者相当于比较两个引用，后者相当于比较两个值。
//当比较两个值得时候，不考虑数据类型。
//也就是说1=="1"是true的。
```

2.3.5 字符串连接运算符

+

和java一样

2.3.6 三目运算符

?:

和java一样

2.4 分支结构

2.4.1 if-else

```
var a=1;
var b=1;
if(a==b){
    document.write("相等");
}else{
    document.write("不相等");
}
//很明显，运行结果是相等
//这就是if-else的结构，和Java语言是一样的。

// 1 if(条件表达式) 关系运算符  逻辑运算符
// 2 if(变量) 直接写变量，不用运算符，如果变量值为undefined,null,0 表示false，否则true
// 3 如果变量没有定义,则出现异常。zhangsan.name ,给zhangsan添加name属性，没有赋值
```

2.4.2 switch

```
var a=2;
switch(a){
  case 1:
    document.write("值为1");
    break;
  case 2:
    document.write("值为2");
    break;
  case 3:
    document.write("值为3");
    break;
  default:
     document.write("值不是3也不是2也不是1");
}
```

2.5 循环结构

2.5.1 for循环

```
var a=0;
for(var i=1;i<=100;i++){
    a+=i;
}
document.write(a);
//上述代码是对1~100求和。
```

2.5.2 while循环

```
var a=0;
var i=1;
while(i<=100){
   a+=i;
   i++;
}
document.write(a);
//上述代码是对1~100求和。
```

2.5.3 do-while循环

```
var a=0;
var i=1;
do{
    a+=i;
    i++;
}while(i<=100);
document.write(a);
//上述代码是对1~100求和。
```

2.5.4 break与continue

和Java一样，在JavaScript中，你也可以使用break结束循环，用continue来结束本次循环。

2.5.5 增强for循环

for(var i in arr){ //i不是arr中的元素，而是下标

}

```
var arr=[10,20,30,40];
for(var i in arr){
  document.write(arr[i]);
  document.write("<br/>")
}

document.write("<br/>-----------------for in-----------------<br/>")
var map=new Array();
map['cn']="中国";
map['us']="美国";
map['jp']="日本";
map['kor']="韩国";

for(var k in map){
  document.write(k+"...."+map[k]);
  document.write("<br/>")
}
```

第三节 JavaScript函数和事件

3.1 函数

包含一段功能的代码。目的：重复使用

3.1.1 函数定义

用function关键字来声明，后面是函数名字，参数列表里不写var。整个方法不写返回值类型。

```
function functionName(parameters){
    //执行的代码
}
```

下面是一个方法的定义与调用：

```
function add(a,b){
    return a+b;
}
var c=1;
var d=2;
var e=add(1,2);
document.write(e);
//上述代码运行结果是3
//这里定义了一个add方法，参数是两个，与Java不同，参数的数据类型并没有。
//因为就算是写，全都是var，为了保证语法的简洁性，全写var索性就设计成全都不用写了。
//返回值也是同样的道理，区别是，如果你写了返回值，那么有返回值，如果没写return，就没有返回值。
```

3.1.2 匿名函数

```
/*匿名函数*/
var method1=function(){
document.write("这是一个匿名函数");
}

method1();
/*匿名函数(自执行匿名函数)*/
(function(s){
document.write("这是一个自执行匿名函数"+s);
})("hahaha");
```

3.1.3 全局变量和局部变量

```
函数内部声明的变量是局部变量,所以只能在函数内部访问它。
在函数外声明的变量是全局变量，网页上的所有脚本和函数都能访问它
```

3.1.4 闭包 （Closure）

闭包就是能够读取其他函数内部局部变量的函数; 闭包可以理解成“定义在一个函数内部的函数“。

闭包三个条件：

1 闭包是一个内部函数

2 闭包能够读取其他（外部）函数的局部变量

3 闭包和局部变量在同一个作用域。

使用形式：1 闭包作为函数的返回值；2闭包作为函数的参数。

案例：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>闭包 closure</title>
        <script type="text/javascript">
            /*定义一个函数*/
            function  a(){
                var a=10;
                document.write(a);
                document.write("<br/>");
            }
//          a();
//          a();
//          a();
            function b(){
                var num=10;//没有释放
                /*函数包含函数*/
                function c(){
                    num++;
                    document.write(num);
                    document.write("<br />")
                }
                return c;
            }

            var f=b();//f是一个函数：闭包
            f();
            f();
            f();

        </script>
    </head>
    <body>
  </body>
    </body>
</html>
```

闭包好处：1 使局部变量常驻内存，2 避免污染全局变量 3 .提高封装性保护局部变量

3.2 系统函数

3.2.1 弹框函数

提示框 alert();

```
alert("你好");
这是一个只能点击确定的弹窗
```

运行结果：

alert方法没有返回值，也就是说如果用一个变量去接受返回值，将会得到undefined。无论你点击“确定”还是右上角的那个“X“关闭。

确认框 confirm();

这是一个你可以点击确定或者取消的弹窗

```
confirm("你好");
```

运行结果：

confirm方法与alert不同，他的返回值是boolean，当你点击“确定”时，返回true，无论你点击“取消”还是右上角的那个“X“关闭，都返回false。

输入框 prompt()

这是一个你可以输入文本内容的弹窗

```
prompt("你爱学习吗？","爱");
```

第一个参数是提示信息，第二个参数是用户输入的默认值。

运行结果：

当你点击确定的时候，返回用户输入的内容。当你点击取消或者关闭的时候，返回null。

3.2.4其他系统函数

parseInt(); 字符串转换整数

parseFloat(); 字符串转成小数

isNaN(); 判断数字是不是不是一个数字。

3.3 事件

事件	描述
onchange	HTML 元素内容改变(离开光标触发)
onclick	用户点击 HTML 元素
onmouseover	光标移动到HTML元素
onmouseout	光标离开HTML元素
onkeydown	用户按下键盘按键
onload	浏览器已完成页面的加载

3.4 字符串、正则表达式

3.4.1 字符串对象String

常用函数演示

```
var str="hello,我爱java";//原始类型
var str2=new String("hello");//引用类型对象类型
```

```
var str3=String("hello");//原始类型
document.write(str==str2);
document.write("<br/>");
document.write(str===str2);
document.write("<br/>");
document.write(typeof(str2));
document.write("<br/>");
//document.write(str instanceof String);

document.write("长度:"+str.length);
document.write("<br/>");
document.write("指定位置的字符:"+str.charAt(0));
document.write("<br/>");
document.write("indexOf:"+str.indexOf("java"));
document.write("<br/>");
var ss=str.split(",");
document.write(ss.length);
document.write("<br/>");
var s1=str.substr(6,6);
document.write(s1);
var s2=str.substring(6,12);
document.write(s2);
```

3.4.2 RegExp 对象

Regular Expression 正则表达式

正则表达式是描述字符模式的对象。

正则表达式用于对字符串模式匹配及检索替换，是对字符串执行模式匹配的强大工具。

语法：

var patt=new RegExp(pattern,modifiers);

或者更简单的方式:

var patt=/pattern/modifiers;

如：

```
var re1 = new RegExp("^1[3589]\\d{9}$");
var re2 = /^1[3589]\d{9}$/;
```

修饰符

修饰符用于执行区分大小写和全局匹配:

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。

方括号

方括号用于查找某个范围内的字符：

表达式	描述
[abc]	查找方括号之间的任何字符。
[^abc]	查找任何不在方括号之间的字符。
[0-9]	查找任何从 0 至 9 的数字。
[a-z]	查找任何从小写 a 到小写 z 的字符。
[A-Z]	查找任何从大写 A 到大写 Z 的字符。
[A-z]	查找任何从大写 A 到小写 z 的字符。
(red|blue|green)	查找任何指定的选项。

元字符

元字符（Metacharacter）是拥有特殊含义的字符：

元字符	描述
.	查找单个字符，除了换行和行结束符。
\w	查找单词字符。[a-zA-Z0-9_]
\W	查找非单词字符。
\d	查找数字。[0-9]
\D	查找非数字字符。
\s	查找空白字符。
\S	查找非空白字符。
\b	匹配单词边界。

量词

量词	描述
n+	匹配任何包含至少一个 n 的字符串。例如，/a+/ 匹配 "candy" 中的 "a"，"caaaaaaandy" 中所有的 "a"。
n*	匹配任何包含零个或多个 n 的字符串。例如，/bo*/ 匹配 "A ghost booooed" 中的 "boooo"，"A bird warbled" 中的 "b"，但是不匹配 "A goat grunted"。
n?	匹配任何包含零个或一个 n 的字符串。例如，/e?le?/ 匹配 "angel" 中的 "el"，"angle" 中的 "le"。
n{X}	匹配包含 X 个 n 的序列的字符串。例如，/a{2}/ 不匹配 "candy," 中的 "a"，但是匹配 "caandy," 中的两个 "a"，且匹配 "caaandy." 中的前两个 "a"。
n{X,}	X 是一个正整数。前面的模式 n 连续出现至少 X 次时匹配。例如，/a{2,}/ 不匹配 "candy" 中的 "a"，但是匹配 "caandy" 和 "caaaaaaandy." 中所有的 "a"。
n{X,Y}	X 和 Y 为正整数。前面的模式 n 连续出现至少 X 次，至多 Y 次时匹配。例如，/a{1,3}/ 不匹配 "cndy"，匹配 "candy," 中的 "a"，"caandy," 中的两个 "a"，匹配 "caaaaaaandy" 中的前

量词	描述
	面三个 "a"。注意，当匹配 "caaaaaaandy" 时，即使原始字符串拥有更多的 "a"，匹配项也是 "aaa"。
n$	匹配任何结尾为 n 的字符串。
^n	匹配任何开头为 n 的字符串。

3.4.2 RegExp 对象方法

方法	描述	FF	IE
exec	检索字符串中指定的值。返回找到的值，并确定其位置。	1	4
test	检索字符串中指定的值。返回 true 或 false。	1	4

支持正则表达式的 String 对象的方法

方法	描述	FF	IE
match	找到一个或多个正则表达式的匹配。	1	4
replace	替换与正则表达式匹配的子串。	1	4
split	把字符串分割为字符串数组。	1	4

3.4.3 正则表达式的使用

test()方法：

test()方法搜索字符串指定的值，根据结果并返回真或假。

```
var patt1=new RegExp("^1[3589]\\d{9}$");
document.write(patt1.test("13688889999"));//true
```

exec() 方法：

exec() 方法检索字符串中的指定值。返回值是被找到的值。如果没有发现匹配，则返回 null。

```
var reg=/java/ig;
var str="hello java,java是最好的编程语言，我爱Java";
var s=null;
while(s=reg.exec(str)){
  document.write(s);
  document.write("<br/>")
}
```

字符串match()方法

```
var reg=/java/ig;
var str="hello java,java是最后的语言，我爱Java";
var arr=str.match(reg);
for(var i=0;i<arr.length;i++){
    document.write(arr[i]+"<br/>");
}
```

字符串replace()方法

```
var reg=/java/ig;
var str="hello java,java是最后的语言，我爱Java";
var str2=str.replace(reg,"JAVA");
document.write(str2+"<br/>");
```

字符串split()方法

```
var str="hello world,java best language";
var arr=str.split(/[ ,]/);
for(var i=0;i<arr.length;i++){
document.write(arr[i]+"<br/>");
}
```

总结

（1）javascript是一门解释性脚本语言，动态添加，弱类型语言。

（2）定义变量

var n;

n=10;

var name="xxx";

（3）javascript包括基本（原始）类型,引用（对象）类型

原始类型：number boolean string undefined null

引用类型：Object（Person）、Array、String、Number、Boolean、RegExp

（4）运算符

===

（5）选择（if（条件））、switch 循环 for while do while for in

（6）函数和事件

自定义函数

function

匿名函数

系统函数

alert() confirm() prompt(); parseInt() parseFloat() isNaN();

事件: onchange onclick onmouseover onmouseout onkeydown onload

（7）字符串（记住）

（8）正则表达式

作业题

```
1.对数组进行排序冒泡或者选择算法
    var arr = {7,3,4,1,16,8};
```

2.打印99乘法表 要求 有格式 必须整齐工整-->把99乘法表嵌套到table中
document.write();
3.文本框输入一个年份，判断是否是闰年（能被4整除却不能被100整除的年份。世纪年份能被400整除的是闰年）将结果在弹出窗口中显示

4.完成页面表单验证
邮箱验证 包含 @ 和 . @在.前面
用户名 必须是字母(大小写)开头,可以有数字和下划线 限制5-8位 必填
密码 必填
手机号: 11位 数字
身份证号: 18位 最后一位可以是X
点击提交按钮 完成表单验证 验证若不符合条件 就弹出提示窗口
5.
请输入性别:
请输入身高:
请输入体重:
男性标准体重=（身高cm－80）×70％
女性标准体重=（身高cm－70）×60％
评估标准
标准体重正负10%为正常体重
标准体重正负10%-20%为体重过重或过轻
标准体重正负20%以上为肥胖或体重不足

轻度肥胖:超过标准体重 20% -30%
中度肥胖:超过标准体重 40%-50%
重度肥胖:超过标准体重 50%以上
要求:
1.判断文本框是否为空 如果为空 需要提示
2.弹出窗口显示评估结果 输出 性别 身高 体重 标准体重
评估结果
轻:输出体重偏轻 多吃点
正常:继续保持
重:轻度/中度/重度肥胖 体重过重 多运动

面试题