事件三要素：事件源、事件类行、事件处理函数

```
onclick
元素.addEventListener('事件类型',function(){},布尔值)
```

一、正则表达式

1.什么是正则表达式?

规则表达式,是可以对字符串操作的一种逻辑公式

2.为什么要使用正则表达式?

1.使用简单的代码，去匹配字符串
2.速度快，代码少
3.在复杂的字符串中快速精确的匹配的想要的字符

3.正则表达式的创建

字面量：

```
var reg = /规则/修饰符;
```

实例化：

```
var reg = new RegExp(规则,修饰符)
```

4.修饰符

i	忽略大小写
g	全部匹配
m	多行匹配

```
var  reg =/at/gi;//定义正则表达式
var str = "cATuATPOatjjkAT"//字符串
console.log(str.replace(reg,"**"));
```

5.元字符

5.1概念

正则表达式中具有特殊含义的字符

\d	**匹配数字**
\D	
\w	匹配数字、字母、下划线
\W	匹配非数字、字母、下划线
\s	匹配空字符
\S	匹配非空字符
\b	匹配单词边界
\B	匹配非单词边界
[\u4e00-\u9fa5] 有事100 有酒发我	

5.2特殊字符

****	**转义字符，将在正则中具有特殊含义的字符转化为普通字符**
[]	中括号中的字符匹配任意一个
^	以什么什么开始
$	以什么什么结束
[^]	取反、除了中括号里以外的字符
.	除了换行以外的任意字符
|	
()	

5.3示例

```
手机号
 var telReg = /^1[3-9]\d{9}$/;
 var tel  = '13856239848';
 console.log(telReg.test(tel));
```

5.4验证

```
test格式:
    正则表达式.test(字符串)
返回值：布尔值  true  代表正确匹配  false 匹配失败
```

二、面向对象