```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
```

简介:
JAVASCRIPT是互联网上最流行的脚本语言 由于是网页的组成部分 所以它广泛应用与 PC 笔记本电脑
以及移动设备

用法
```
<!--HTML 中的脚本必须位于 <script> </script> 标签之间 标签会告诉 JavaScript 在何处开始和结束。 。
    脚本可被放置在 HTML 页面的 <body> 和 <head> 部分中。
-->

<script>
      document.write("<h1>第一行</h1>");
    document.write("<h1>第二行</h1>");


    </script>
```

核心数据类型

数字: 1,2,3,4, 1.1, 2.2
字符串: "hello!!" 'world' 用单引号或者双引号把字符包起来
表达式: 1+1 2*2
数组: [1,2,3,3,4]
字典(对象): {"a":1, "b": 2}
函数: function f1(){}

JavaScript 语句
1:分号用于分隔 JavaScript 语句。
2:通常我们在每条可执行的语句结尾添加分号。
3:使用分号的另一用处是在一行中编写多条语句。
4.JavaScript的代码是 JavaScript 语句的序列。
5.浏览器按照编写顺序依次执行每条语句。

变量:
变量是用于存储信息的"容器"。 以下为变量的定义
javascript是一门动态类型语言

```
<script>
    var x = 1;
    var y = 2;
    console.log(x + y);

    var z= "hello";
    var z = 1.1;
    var z = [1,2,23];

    //字符串
    var x = "apple";

    //布尔
    var x = true;
    var y = false;

    //数组
    var x = [1,2,3,4,5]; //第一种方法
    var y = new Array(); //第二种方法
    y[0] = "a";
    y[1] = 2;
    y[2] = true;

    //对象(字典)

    var person = {"a": 1, "b": 2, "c": 3};

    //undefined 与 null
    //undefined #表示变量不含有值
    //null 可以通过将一个变量的值设置为null来清空变量

    var person = {"name": "liushuo", "action": function () {
            return "object person.action"
        }};


    alert(person.name);
    alert(person.action);  //不带()表示不调用方法，而是返回该方法的代码
    alert(person.action());  //表示调用对象的方法 返回函数的return内容
```

</script>

javascript 字符串
字符串可以存储一系列字符,这些字符用 "" 或者 ''包起来
字符串索引从0开始 意味着字符串的第一个元素的索引值为0
第二个索引为1, 以此类推, 这跟Python一致

字符串的方法

split() 字符起付出切割
trim() #移除字符串的收尾空白类似Python strip()
slice() 字符串的分片
replace() 替换字符串中的元素
concat() ##拼接字符串
match() 找到一个或多个正则表达式的匹配
search() 检索与正则表达式相匹配的值
substr() 从起始索引号提取字符串中指定数目的字符
substring() #提取字符串中两个指定索引号之间的字符
toLowerCase() ##字符串转换为小写
toUpperCase() ##字符串转换为大写
toString() ##返回字符串对象值
valueOf() #返回字符串对象的原始值

```
<script>
var name = "liushuo";
var school = 'togogo';

console.log(name[0]);
alert(name.length); //返回字符串的长度

</script>
```

```
//数组的常用方法

var z = x.concat(y); //合并2个数组
console.log(z);

var w = x.concat(y, z); //合并3个数组
console.log(w);
```

```
var p = x.join("|");  //列表拼接成字符串
console.log(p);

console.log(x.pop()); //从列表中删除元素

x.push("www"); //向列表中追加元素
console.log(x);
x.shift();  //删除列表中的第一个元素
console.log(x);

console.log(y.slice(1, 4));  //列表分片

var s = y.toString(); //列表转换成字符串
console.log(s);


var s1 = ["banana", "apple", "orange"];
s1.unshift("dog", "cat"); //从列表的前面添加元素
console.log(s1)
```

javascript 条件控制

结构

```
if(condition){

}


if(condition){

}else if(condition){

}else{
}


if(condition){

}else{

}
```

```
switch(n){

  case 1:
    xx
    break;

  case 2:
     yy
     break;
  default:
     print zz  匹配不存在的时候做的事情
}
```

```
<script>

   var s = "apple";
   switch (s){

       case "apple":
          alert("甜的!!");
          break;
       case "banana":
          alert("黄的");
          break;
       default:
          alert("水果")
   }
```

```
var x = 80;

     switch (true){  //如果case是表达式的话  switch(true) 这里要用bool型

          case x > 60 && x < 70:
             console.log("good!!");
             break;
          case x >= 70 && x < 80:
             console.log("better!");
             break;
          case x >= 80:
```

```
            console.log("best!")
            break;
        default:
            console.log("xxxyyyy")

    }

</script>
```

javascript 循环控制

for - 循环代码块一定的次数
for / in 循环遍历对象属性
while 当指定条件为true时循环指定代码
do/while 同样当指定的条件为true时循环指定的代码块 该循环至少会执行一次 即使条件为假也会执行一次, do下的代码会在条件被测试前执行, 每次循环之前都会执行do下面的语句

关键字break continue 与Python中的相同

```
<script>

    var i = 5;
    while (i > 0){
      alert("this is " + i);
      i--
    }


    var i = 1;
    while (i > 2){
       alert("hahahaa"); //这里是不打印的
    }
```

在实际的编程练习中循环语句的使用频率是：for >while() >do....while()

```
var x = 5;
var text = '';

do{
    var l  ='this is do....' + x + "<br/>";
    console.log(l);
    text += l ;
    x = x - 1;

}while(x >= 0){
    console.log(text);
}
```

##首先执行一次DO中的语句, 然后判断while中的条件, 如果成立则返回头部再去执行do中的语句 如果不成立则执行while
下的语句.

执行结果:

this is do....5
this is do....4
this is do....3
this is do....2
this is do....1
his is do....0
this is do....5
this is do....4
this is do....3
this is do....2
this is do....1
this is do....0

```
    var s = "apple";

    for(var i =0; i<s.length; i++){
        alert(s[i])
    }


    var l = [1,2,3,4];
    for(var i =0; i<l.length; i++){
        alert(l[i])
```

```
    }

    for(var i in l){
        alert(i); //i为索引
    }


    var d = {"name": "liushuo", "age": 20};
    for (var i in d){
        alert(i + ":" + d[i])
    }


</script>
```

typeof 与 类型转换

可以使用typeof操作符来检测变量的数据类型

```
<script>

    alert(typeof "togogo");
    alert(typeof 3.14);
    alert(typeof false);
    alert(typeof [1,2,3,4]);
    alert(typeof {name: "john", age:34});

    //null是一个特殊的类型 表示一个空对象引用 typeof检测null 返回object


    var person = null;
    alert(typeof person);

    //undefined 是一个没有设置值的变量 任何变量都可以设置值为
    //undefined来清空 类型为undefined

    var person = undefined;

    alert(typeof undefined);
```

```
    alert(typeof null);


</script>
```

JavaScript 函数和事件
上面例子中的 JavaScript 语句，会在页面加载时执行。

通常，我们需要在某个事件发生时执行代码，比如当用户点击按钮时。

如果我们把 JavaScript 代码放入函数中，就可以在事件发生时调用该函数。

JavaScript 显示数据
JavaScript 可以通过不同的方式来输出数据：

使用 window.alert() 弹出警告框。
使用 document.write() 方法将内容写到 HTML 文档中。
使用 innerHTML 写入到 HTML 元素。
使用 console.log() 写入到浏览器的控制台。

```
<p1 id="p1">原来的文字</p1><br/>
<button id="b1" onclick="f5()">改变这里的文字</button>
<script>
    window.alert("呜呜呜呜我");

    function f5(){
        document.getElementById("p1").innerText = "改变后的文字@@"
    }

    console.log("togogo")

</script>
```

事件与函数

HTML事件可以是用户的行为也可以是浏览器的行为, 以下是HTML事件实例
1. HTML页面完成加载时
2. HTML input字段改变时
3. HTML 中的按钮被点击时

常见的HTML事件
onclick 按钮点击
onmouseover 鼠标移动到元素上时
onmouseout 鼠标从元素上移开时
onload 页面加载完成时

函数是由事件驱动或者当它它被调用的时候可重复执行的代码块

函数的定义
function name(x, y){
    函数的内容
}

javascript 对大小写敏感， 定义函数的关键字function必须是小写的

<button onclick="f1()">执行函数f1</button>
<button onclick="f2(1, 2)">执行函数f2</button>

结果是:
结果是:

<button onmouseover="f1()" onmouseout="f2(5,6)">鼠标移动到这里</button><br/
>

s

<script>

    window.onload(alert("页面加载完毕")); //当窗口加载完毕时
    //1. 一个基本的函数

```
function f1(){

    alert("hello world!")

}

//2. 带有参数的函数
function f2(x, y){
    alert(x + y)
}


//3. 带有返回值的参数

function f3(x=3, y=4){
   return x + y
}

   document.getElementById("res").innerHTML = f3();
   document.getElementById("res1").innerHTML = f3(1, 2);

   </script>
```

代码异常检测
try 语句允许我们定义在执行时进行错误检测的代码块
catch 语句允许我们定义当try执行错误的时候所执行的代码块

```
try{
  这里运行代码
}catch(err){
  处理错误用的代码
}
```

Throw语句允许我们创建自定义错误 类似Python的raise

```
<script>

 try{
    alert("这里是错误检测代码")
 }catch(err){
    alert("代码发生了错误")
 }
```

```
  try{
     addlert("这里是执行的代码 报错")
  }catch(err){
     alert("代码发生错误在这里处理")
  }


try{
  throw "报错了！！！！"
  }catch(err){
     alert("throw" + err)
}


</script>
```

JSON处理

1.轻量级的数据交换格式
2.是一门独立的语言
3.便于理解书写格式相当于python中的字典

在javascript中JSON的转换

```
var t = {"a": 1, "b": 2, "c": 3};
console.log(typeof t);

var w = JSON.stringify(t);
console.log(typeof w);
console.log(w);

var z = JSON.parse(w);
console.log(z);
console.log(typeof z);
```

javascript void:0含义
这其中最关键的就是void关键字, 代表要计算一个表达式
但是不返回值

```
<a href="javascript:void(alert('waring!!'))">点我</a><br/>
<a href="javascript:void(0)">定义一个死连接</a>


</body>
</html>
```