

一、JavaScript概述

1.1 JavaScript简介

JavaScript是一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，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，浏览器对象模型。
- 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作为给非程序员的脚本语言，而非作为给程序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CEnvir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。

二、JavaScript基本语法

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是声明关键字，a是变量名，语句以分号结尾。
- 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。

```
var a;
```

javascript

JavaScript的部分关键字：

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。

2.2基本类型

- 变量的基本类型又有Number、String、Boolean、Undefined、Null五种。
- 来声明一个数字Number类型，如下：

```
var a=1;
```

- 来声明一个字符串String类型。
- 你可以使用：

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

- 来声明一个布尔Boolean类型。
- 你可以使用：

```
var a=false;
```

- 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：(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)

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

2.3引用类型

在Java中需要类定义，然后再实例对象：

```
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中对象可以直接写出来：

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

2.4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。
- 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];
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>Document</title>
9 </head>
10
11 <body>
12   <script type="text/javascript">
13     //例子1
14     var n = [100, true, "hello", n2 = {
15       name: "list",
16       age: 12,
17       sex: "man"
18     ];
19     // alert(n.length); //弹出框
20     // alert(n[0]);
21     // alert(n.length - 1);
22     document.write(n[n.length - 1].name + "<br>");
23     //例子2
24     var student = [
25       {
26         id: 1,
27         name: "张三",
28         age: 18
29       },
30       {
31         id: 2,
32         name: "李四",
33         age: 18,
34       },
35       {
36         id: 3,
37         name: "王五",
38         age: 19
39     }];
40     document.write(student[0].id);
41     document.write(student[0].name);
42     document.write(student[0].age);
43     document.write("<br>");
44     document.write(student[1].id);
45     document.write(student[1].name);
46     document.write(student[1].age);
47     document.write("<br>");
48     document.write(student[2].id);
49     document.write(student[2].name);
```

```
47         document.write(student[2].age);
48         document.write("<br>");
49     </script>
50 </body>
51
52 </html>
```

2.5运算符

逻辑运算

名称	运算符	描述
与	&&	要求表达式左右两边的表达式同为true，整体结果才为true
或		要求表达式左右两边的表达式只要有一个为true，整体结果就为true
非	!	将布尔值取反操作

```
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;
```

激活 Windo
在图标上单击右键

关系运算

名称	运算符
等于	==
小于	<
小于或等于	<=
大于	>
大于或等于	>=
不等于	!=
值和类型相同	===

单目运算：自增自减

名称	运算符	描述
自增	++	变量的值每次加1，再赋给变量
自减	--	变量的值每次减1，再赋给变量

双目运算符

名称	运算符
加	+
减	-
乘	*
除	/
求余	l%
赋值	=
加等	+=
减等	-=
除等	/=
乘等	*=
求余等	%=

2.6条件分支结构

和java基本语法差不多

2.7循环结构

和java基本语法差不多

2.8函数

函数定义：用function关键字来声明，后面是方法名字，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  
}
```

方法的定义与调用举例：

```
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。
```

2.9常见弹窗函数

- alert弹框：这是一个只能点击确定按钮的弹窗
- 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。

```
alert("你好");
```



- confirm弹框：这是一个你可以点击确定或者取消的弹窗
- 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。

```
confirm("你好");
```

- prompt弹框：这是一个你可以输入文本内容的弹窗
 - 第一个参数是提示信息，第二个参数是用户输入的默认值。
- 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。

```
prompt("你爱学习吗？", "爱");
```



2.10事件

事件名称	描述
onchange	HTML元素内容改变
onclick	用户点击HTML元素
onmouseover	用户将鼠标移入一个HTML元素中
onmousemove	用户在一个HTML元素上移动鼠标
onmouseout	用户从一个HTML元素上移开鼠标
onkeyup	键盘
onkeydown	用户按下键盘按键
onload	浏览器已完成页面的加载
onsubmit	表单提交

2.11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);
 - | var patt=/pattern/modifiers;

```
var re = new RegExp("\\w+");
var re = /\w+/;
```

修饰符：用于执行区分大小写和全局匹配：

i g m

更多

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：用于查找某个范围内的字符

i g m

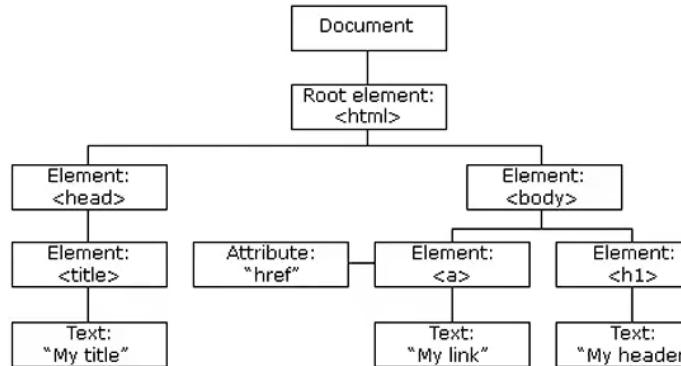
更多

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。
[adgk]	查找给定集合内的任何字符。
[^adgk]	查找给定集合外的任何字符。
(red blue green)	查找任何指定的选项。

三、JavaScript的DOM

3.1概述

- 通过 HTML DOM，可访问 JavaScript HTML 文档的所有元素。
- 当网页被加载时，浏览器会创建页面的文档对象模型（Document Object Model）。
- HTML DOM 模型被构造为对象的树：



© 2023 出版人

- 通过可编程的对象模型，JavaScript 获得了足够的能力来创建动态的 HTML。
 - JavaScript 能够改变页面中的所有 HTML 元素。
 - JavaScript 能够改变页面中的所有 HTML 属性。
 - JavaScript 能够改变页面中的所有 CSS 样式。
 - JavaScript 能够对页面中的所有事件做出反应。

3.2查找HTML元素

- 通常，通过 JavaScript，您需要操作 HTML 元素。
- 为了做到这件事情，您必须首先找到该元素。有三种方法来做这件事：
 - 通过 id 找到 HTML 元素
 - 在 DOM 中查找 HTML 元素的最简单的方法，是通过使用元素的 id。
 - 方法：`document.getElementById("id属性值");`
 - 如果找到该元素，则该方法将以对象（在 x 中）的形式返回该元素。
 - 如果未找到该元素，则 x 将包含 null。
 - 通过标签名找到 HTML 元素
 - 方法：`getElementsByName("合法的元素名");`
 - 通过类名找到 HTML 元素
 - 方法：`getElementsByClassName("class属性的值")`

3.3改变HTML

改变HTML输出流: document.write() 可用于直接向 HTML 输出流写内容

```
<!DOCTYPE html>
<html>
<body>
<script>
document.write("Hello world,I'm JavaScript");
</script>
</body>
</html>
```



改变HTML内容: 使用 innerHTML 属性

```
<html>
<body>
<p id="p1">Hello World!</p>
<script>
document.getElementById("p1").innerHTML="abcd";
</script>
</body>
</html>
```



- 改变HTML属性: document.getElementById(*id*).attribute=新属性值
- 将attribute替换为真实的属性名

```
<!DOCTYPE html>
<html>
<body>

<script>
document.getElementById("image").src="2.jpg";
</script>
</body>
</html>
```

3.4 CSS变化+

- 对象.style.property=新样式
- 将property替换成真实的css属性名

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>

<p id="p1">Hello World!</p>
<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color="blue";
document.getElementById("p2").style.fontFamily="Arial";
document.getElementById("p2").style.fontSize="larger";
</script>
<p>以上段落通过脚本修改。</p>

</body>
</html>
```

3.5DOM事件

- HTML DOM 允许我们通过触发事件来执行代码。
- 比如以下事件：
 - 元素被点击。
 - 页面加载完成。
 - 输入框被修改。
- 本例改变了 id="id1" 的 HTML 元素的样式，当用户点击按钮时：

```
<!DOCTYPE html>
<html>
<body>

<h1 id="id1">myH1</h1>
<button type="button"
onclick="document.getElementById('id1').style.color='red'>
button</button>

</body>
</html>
```

3.6EventListener

- addEventListener() 方法
- 在用户点击按钮时触发监听事件：

```
document.getElementById("myBtn").addEventListener("click", displayDate);
```

- addEventListener() 方法用于向指定元素添加事件句柄。
- addEventListener() 方法添加的事件句柄不会覆盖已存在的事件句柄。
- 你可以向一个元素添加多个事件句柄。
- 你可以向同个元素添加多个同类型的事件句柄，如：两个 "click" 事件。
- 你可以向任何 DOM 对象添加事件监听，不仅仅是 HTML 元素。如： window 对象。
- addEventListener() 方法可以更简单的控制事件（冒泡与捕获）。
- 当你使用 addEventListener() 方法时，JavaScript 从 HTML 标记中分离开来，可读性更强，在没有控制 HTML 标记时也可以添加事件监听。|

你可以使用 removeEventListener() 方法来移除事件的监听。

```
element.addEventListener(event, function, useCapture);
```

- 事件传递有两种方式：冒泡与捕获。
- 事件传递定义了元素事件触发的顺序。如果你将 <p> 元素插入到 <div> 元素中，用户点击 <p> 元素，哪个元素的 "click" 事件先被触发呢？
 - 在 *冒泡* 中，内部元素的事件会先被触发，然后再触发外部元素，即：<p> 元素的点击事件先触发，然后会触发 <div> 元素的点击事件。
 - 在 *捕获* 中，外部元素的事件会先被触发，然后才会触发内部元素的事件，即：<div> 元素的点击事件先触发，然后再触发 <p> 元素的点击事件。
- addEventListener() 方法可以指定 "useCapture" 参数来设置传递类型：

```
addEventListerner(event, function, useCapture);
```

默认值为 false，即冒泡传递，当值为 true 时，事件使用捕获传递。

```
document.getElementById("myDiv").addEventListener("click", myFunction, true);
```

removeEventListener() 方法移除由 addEventListener() 方法添加的事件句柄：

```
element.removeEventListener("mousemove", myFunction);
```

3.7 操作元素

- 如需向 HTML DOM 添加新元素，您必须首先创建该元素（元素节点），然后向一个已存在的元素追加该元素。
 - 创建元素: `document.createElement()`
 - 追加元素: `appendChild()`

```
<div id="div1">
<p id="p1">这是一个段落。</p>
<p id="p2">这是另一个段落。</p>
</div>

<script>
var para=document.createElement("p");
var node=document.createTextNode("这是一个新段落。");
para.appendChild(node);

var element=document.getElementById("div1");
element.appendChild(para);
</script>
```

四、浏览器DOM

- 浏览器对象模型 (BOM-Browser Object Model) 使 JavaScript 有能力与浏览器“对话”。
- 由于现代浏览器已经（几乎）实现了 JavaScript 交互性方面的相同方法和属性，因此常被认为是 BOM 的方法和属性。

4.1 window

- 所有浏览器都支持 `window` 对象。它表示浏览器窗口。
- 所有 JavaScript 全局对象、函数以及变量均自动成为 `window` 对象的成员。
- 全局变量是 `window` 对象的属性。
- 全局函数是 `window` 对象的方法。
- 甚至 HTML DOM 的 `document` 也是 `window` 对象的属性之一：

• window的尺寸

- 对于 Internet Explorer、Chrome、Firefox、Opera 以及 Safari:
 - `window.innerHeight` - 浏览器窗口的内部高度(包括滚动条)
 - `window.innerWidth` - 浏览器窗口的内部宽度(包括滚动条)
- 对于 Internet Explorer 8、7、6、5:
 - `document.documentElement.clientHeight`
 - `document.documentElement.clientWidth`
 - 或者
 - `document.body.clientHeight`
 - `document.body.clientWidth`

激活 Window
转到“设置”以激活

```
var w=window.innerWidth||document.documentElement.clientWidth||document.body.clientWidth;
var h=window.innerHeight||document.documentElement.clientHeight||document.body.clientHeight;
```

- Window Screen

- 可用宽度: screen.availWidth 属性返回访问者屏幕的宽度, 以像素计, 减去界面特性, 比如窗口任务栏。
- 可用高度: screen.availHeight 属性返回访问者屏幕的高度, 以像素计, 减去界面特性, 比如窗口任务栏。

```
document.write("可用宽度: " + screen.availWidth);
document.write("可用高度: " + screen.availHeight);
```

- Window Location

- window.location 对象用于获得当前页面的地址 (URL), 并把浏览器重定向到新的页面。
- window.location 对象在编写时可不使用 window 这个前缀。一些例子:
 - location.hostname 返回 web 主机的域名
 - location.pathname 返回当前页面的路径和文件名
 - location.port 返回 web 主机的端口 (80 或 443)
 - location.protocol 返回所使用的 web 协议 (http:// 或 https://)
 - location.href 属性返回当前页面的 URL
 - location.assign() 方法加载新的文档

激活 Windows
转到“设置”以激活

- Window History

- window.history 对象包含浏览器的历史。
- window.history 对象在编写时可不使用 window 这个前缀。
 - history.back()
 - history.forward()
- 一些方法示例如下:

history.back() - 与在浏览器点击后退按钮相同

history.forward() - 与在浏览器中点击按钮向前相同

```
<html>
<head>
<script>
function goForward()
{
    window.history.forward()
}
</script>
</head>
<body>

<input type="button" value="Forward" onclick="goForward()">

</body>
</html>
```

- Window Navigator

- window.navigator 对象在编写时可不使用 window 这个前缀。

```
<div id="example"></div>
<script>
txt = "<p>浏览器代号: " + navigator.appCodeName + "</p>";
txt+= "<p>浏览器名称: " + navigator.appName + "</p>";
txt+= "<p>浏览器版本: " + navigator.appVersion + "</p>";
txt+= "<p>启用Cookies: " + navigator.cookieEnabled + "</p>";
txt+= "<p>硬件平台: " + navigator.platform + "</p>";
txt+= "<p>用户代理: " + navigator.userAgent + "</p>";
txt+= "<p>用户代理语言: " + navigator.systemLanguage + "</p>";
document.getElementById("example").innerHTML=txt;
</script>
```

4.2 JavaScript 定时器

- 定义定时器：

- setInterval('调用函数',毫秒时间)：每间隔固定毫秒值就执行一次函数
 - setTimeout('调用函数',毫秒时间)：在固定时间之后执行一次调用函数

- 关闭定时器：

- clearInterval(定时器名称)
 - clearTimeout(定时器名称)