JavaScript

JavaScript介绍

> *JavaScript语言诞生主要是完成页面的数据验证。因此它运行在客户端，需要运行浏览器来解析执行JavaScript代码。*

JavaScript是弱类型，Java是强类型。弱类型指类型可变。强类型指定义变量时，类型已确定，且不可变。

特点

- 交互性（信息交互）
- 安全性（不允许直接访问本地硬盘）
- 跨平台性（只要是可以解释JavaScript的浏览器都可以执行，与平台无关）

JavaScript和HTML代码的结合方式

第一种方式

在head标签中，或者在body标签中，使用script标签来书写JavaScript代码。

```
<script type = "text/javascript">
    alert("Hello JavaScript!");
</script>
```

第二种方式

使用script标签引入单独的JavaScript代码文件

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

JavaScript的变量和数据类型

JavaScript的变量类型

- 数值类型 number
- 字符串类型 string
- 对象类型 object
- 布尔类型 boolean
- 函数类型 function

JavaScript里的特殊值

- undefined：未定义，所有JavaScript变量为赋予初始值时，默认值为undefined
- null
- NAN：Not a Number - 非数值

```
var a = 12
var b = "abc";
alert(a * b); //NaN 非数值 非数字
```

JavaScript定义变量格式

```
var 变量名 = 值;
```

JavaScript的运算

JavaScript的关系（比较）运算

- 等于： == 等于是简单的做字面的比较
- 全等于： === 除了做字面值的比较之外，还会比较两个变量的数据类型

```
var a = "12"
var b = 12;
alert(a == b); //true
alert(a === b); //false
```

JavaScript的逻辑运算

在JavaScript中，所有变量都可以作为一个Boolean类型的变量去使用，且0、null、undefined、""（空串）被认为是false。

&& 与运算

- 第一，当表达式全为真时，返回最后一个表达式的值。
- 第二，当表达式有一个为假时，返回第一个为假的表达式的值。

|| 或运算

- 第一，当表达式全为假时，返回最后一个表达式的值
- 第二，只要有一个表达式为真。就会把第一个为真的表达式的值返回

JavaScript的数组

数组定义方式

```
var 数组名 = [];
var 数组名 = [1, 'abc', true];
```

注意：JavaScript中的数组，只要我们通过数组下标赋值，那么最大的下标值，就会自动的给数组做扩容操作。

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

函数

函数的两种定义方式

第一种

可以使用function关键字定义函数

```
function 函数名(形参列表){
    函数体
}
```

```
function fun(a, b){
    alert(a + b);
}
fun("Hello", "JavaScript");
```

注意：JavaScript中定义带有返回值的函数，直接在函数体内加上return语句即可。

第二种

```
var 函数名 = function(形参列表){函数体}
```

函数不允许重载

> *Javascript 中的函数重载Javascript 中没有真正意义上的函数重载，因为javascript 中同一作用域下的同名函数，前者会被后者覆盖，但是可通过其他方法间接实现重载同样的效果，Javascript中的函数没有签名，它的参数是由包含零的多个数组来表示的。*

函数的**arguments**隐形参数

- 只在function函数内
- 定义：在function函数中不需要定义，但却可以直接用来获取所有参数的变量。称为隐形参数。
- 隐形参数特别像Java中的可变长参数
- public void function(Object... args)
- JavaScript中隐形参数与Java的可变长参数一样，操作类似数组。

```
function fun(){
    alert(arguments.length);

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

需求

要求编写一个函数，用于计算所有参数相加的和并返回、

```
function sum(num1, num2){
    var result = 0;
    for(var i = 0; i < arguments.length; i++){
        //避免字符串或者其他非number类型造成除了参数相加的其他操作
        if(typeof(arguments[i]) == "number"){
            result += arguments[i];
        }
    }
    return result;
}
alert(sum(num1, num2, ...));
```

JavaScript中的自定义对象

Object形式的自定义对象

```
var 变量名 = new Object();//对象实例（空对象）
变量名.属性名 = 值;//定义一个属性
变量名.函数名 = function(){}//定义一个函数
```

{}花括号形式的自定义对象

```
var 变量名 = {
    属性名:值,
    属性名:值,
    函数名:function(){}//最后一个不需要加逗号
};
```

JavaScript的事件

> *事件是电脑输入设备与页面进行交互的响应*

常用的事件

- onload 加载完成事件：页面加载完成之后，常用于做页面JavaScript代码初始化操作。

- onclick 单击事件：常用于按钮的点击响应操作。
- onblur 失去焦点事件：常用于输入框失去焦点后验证其输入内容是否合法。
- onchange 内容发送改变事件：常用于下拉列表和输入框发送改变后操作。
- onsubmit 表单提交事件：常用于表单提交前，验证所有表单项是否合法。

事件的注册

> *告诉浏览器，当事件响应后要执行哪些操作代码，称为事件注册或事件绑定。*

静态注册事件

> *通过HTML标签的事件属性直接赋予事件响应后的代码，这种方式被称为静态注册。*

动态注册事件

> *是指先通过JavaScript代码得到标签的dom对象.事件名=function(){}，这种形式赋予事件响应后的代码，被称为动态注册。*

动态注册基本步骤：

1. 获取标签对象
2. 标签对象.事件名 = function(){}

onload事件

静态注册**onload**事件

> *onload事件是浏览器解析完页面就会自动触发的事件*

onload事件的静态注册是在标签中

```
<body onload = "alert('静态注册onload事件')">
```

动态注册**onload**事件

```
<script type = "text/javascript">
    //onload事件动态注册，固定写法
    window.onload = function(){
        alert("动态注册的onload事件");
    }
</script>
```

onclick事件

静态注册**onclick**事件

```
<button onclick = "onclickFun();"></button>
```

动态注册**onclick**事件

```
<script type = "text/javascript">
    //动态注册onclick事件
    //1.获取标签对象
    //document对象是文档的根节点
    var btnObj = document.getElementById("btn01");
    //2.通过标签对象.事件名 = finction(){}
    btnObj.onclick = function(){
        alert("动态注册onclick事件");
    }
</script>
</head>
<body>
    <button id = "btn01">按钮</button>
```

onblur失去焦点事件

静态注册失去焦点事件

```
<script type = "text/javascript">
    function onblurFun(){
        //console是控制台对象，是由JavaScript语言提供，专门用来向浏览器的控制器打印输出，用于测试使用
        //log()是打印的方法
        console.log("静态注册失去焦点事件");
    }
</script>
</head>
<body>
    用户名：<input type="text" onblur="onblurFun();"><br/>
    密码：<input type="text"><br/>
```

动态失去焦点事件

```
<script type = "text/javascript">
    window.onload = function(){
        var passwordObj = document.getElementById("password");
        passwordObj.onblur = function(){
            console.log("动态注册失去焦点事件");
        }
}
</script>
</head>
<body>
    用户名：<input type="text" onblur="onblurFun();"><br/>
    密码：<input id="password" type="text"><br/>
```

onchange事件

静态注册**onchange事件**

```
<script type="text/javascript">
    function onchangeFun(){
    alert("Number is changed");
}
</script>
</head>
<body>
    <select onchange="onchangeFun();">
        <option>Number</option>
        ...
```

动态注册**onchange**事件

```
<script type="text/javascript">
    //1.获取标签对象
    window.onload = function(){
    var selObj = document.getElemenrById("sel01"){
    //2.通过标签对象.事件名 = function(){}
    selObj.onchange = function(){
        alert("Number is changed");
    }
}
</script>
</head>
<body>
    <select id = "sel01">
        <option>Number</option>
        ...
```

onsubmit表单提交事件

静态注册**onsubmit**事件

```
<script type = "text/javascript">
    function onsubmitFun(){
        if( ){//验证其合法性，如表单项不合法，则阻止提交
            alert("静态注册表单提交事件---不合法");
            return false;
        }
    }
</script>
</head>
<body>
    <form action = "http://localhost:8080" method = "post" onsubmit = "return onsubmitFun()">
        <input type = "submit" value = "静态注册"/>
    </form>
```

动态注册**onsubmit**事件

```
<script type = "text/javascript">
    window.onload = function(){
        var formObj = document.getElementById("form01");
        formObj.onsubmit = function(){
            alert("动态注册onsubmit事件");
            return false;//阻止
        }
}
</script>
</head>
<body>
    <form action = "http://localhost:8080" method = "post" id = "form01">
        <input type = "submit" value = "静态注册"/>
    </form>
```

DOM模型

> *DOM全称是Document Object Model 文档对象模型。是把文档中的标签，属性，文本，转换成为对象来管理。*

Document对象

Document文档树内存结构及使用方法的参考网址（点击可跳转）

Document对象的理解

- Document管理了所有的HTML文档内容
- document是一种树结构文档，有层级关系
- document让所有标签都对象化
- 可以通过document访问所有的标签对象

Document对象中的方法介绍

- document.getElementById(elementId)
- document.getElementByName(elementName)
- document.getElementByTagName(tagname)
- document.createElement(tagName)

需求

当用户点击了校验按钮，要获取输出框中的内容。然后验证其是否合法。

验证的规则是，必须有字母、数字、下划线组成，并且长度为5到12位

```
<script type="text/javascript">
    function onclickFun(){
        //获取标签对象
        var usernameObj = document.getElementById("username");
        var usernameText = usernameObj.value;
        var patt = /^\w{5, 12}$/;
        //test()方法用于测试输入字符串是否符合输入规则
        if(patt.test(usernameText)){
            alert("用户名合法");
        }else{
            alert("用户名不合法");
        }
    }
</script>
```

```
</head>
<body>
    用户名：<input type="text" id="username" value="username"/>
    <button onclick="onclickFun()">校验</button>
```

正则表达式对象

参考资料菜鸟教程-正则表达式（点击可跳转）

两种常见的验证提示效果

第一种

```
<script type="text/javascript">
    function onclickFun(){
        //获取标签对象
        var usernameObj = document.getElementById("username");
        var usernameText = usernameObj.value;
        var patt = /^\w{5, 12}$/;
        //test()方法用于测试输入字符串是否符合输入规则
        var usernameSpanObj =
document.getElementById("usernameSpan");
        //innerHTML 表示起始标签和结束标签中的内容
        //innerHTML 此属性可读，可写
        if(patt.test(usernameText)){
            usernameSpanObj.innerHTML = "用户名合法";
        }else{
            usernameSpanObj.innerHTML = "用户名不合法";
        }
    }
</script>
</head>
<body>
    用户名：<input type="text" id="username" value="username"/>
        <span id="usernameSpan style ="color:red;"></span>
    <button onclick="onclickFun()">校验</button>
```

第二种

```
在第一种的情况下，将 usernameSpanObj.innerHTML = "<img src = \"正确的/错误的.jpg\">"
```

getElementsByName方法

> *返回带有指定名称的对象集合*

```
    <script type="text/javascript">
        function checkALL(){
            //document.getElementsByName();根据指定的name属性查询返回多个标签对象集合
            //集合的操作与数组一样
            //集合中每个元素都为dom对象
            //集合中元素顺序位HTML页面中从上到下的顺序
            var hobbies = document.getElementsByName("hobby");
            //checked表示复选框的选中状态，选中则为true，反之则为false
            //checked属性可读，可写
            for(var i = 0; i < hobbies.length; i++){
                hobbies[i].checked = true;
            }
        }
        function checkNo(){
            var hobbies = document.getElementsByName("hobby");
            for(var i = 0; i < hobbies.length; i++){
                hobbies[i].checked = false;
            }
        }
        function checkReverse(){
            var hobbies = document.getElementsByName("hobby");
            for(var i =0; i < hobbies.length; i++){
                if(hobbies[i] == true)
                    hobbies[i] = false;
                if(hobbies[i] == false)
                    hobbies[i] = true;
            }
        }
    </script>
```

```
</head>
<body>
        兴趣爱好：
        <input type="checkbox" name="hobby" value="cpp">C++
        <input type="checkbox" name="hobby" value="java">Java
        <input type="checkbox" name="hobby"
value="js">JavaScript
        <button onclick="checkALL">全选</button>
        <button onclick="checkNo">全不选</button>
        <button onclick="checkReverse">反选</button>
```

getElementsByTagName方法

> *返回带有指定签名的对象集合*

```
<script type="text/javascript">
        function checkALL(){
            //document.getElementsByTagName();根据指定标签名来进行查
询并返回集合
            //集合的操作与数组一样
            //集合中每个元素都为dom对象
            //集合中元素顺序位HTML页面中从上到下的顺序
            var inputs = document.getElementsByTagName("input");
            //checked表示复选框的选中状态，选中则为true，反之则为false
            //checked属性可读，可写
            for(var i = 0; i < inputs.length; i++){
                inpurs[i].checked = true;
            }
        }
    </script>
</head>
<body>
        兴趣爱好：
        <input type="checkbox" value="cpp">C++
        <input type="checkbox" value="java">Java
        <input type="checkbox" value="js">JavaScript
        <button onclick="checkALL">全选</button>
```

document对象三个查询方法的使用注意事项

- document对象的三个查询方法，如果有id属性，优先使用**getElementById**方法来进行查询。如果没有id属性，则优先使用**getElementsByName**。如果id属性和name属性都没有最后再按标签名查**getElementsByTagName**。
- 三个方法都一定要在页面加载完成之后执行，才能查询到标签对象。

节点的常用属性和方法

节点是标签对象

方法

getElementsByTagName()

获取当前节点的指定标签名孩子节点

appendChild(oChildNode)

可以添加一个子节点，oChildNode是要添加的孩子节点

属性

childNodes

获取当前节点的所有节点

firstNode

获取当前节点的第一个子节点

lastNode

获取当前节点的最后一个子节点

parentNode

获取当前节点的父节点

nextSibling

获取当前节点的下一个节点

previousSibling

获取当前节点的上一个节点

className

用于获取或设置标签的class属性值

innerHTML

获取/设置起始标签和结束标签中的内容

innerText

获取/设置起始标签和结束标签中的文本

DOM查询练习

createElement() && appendChild()

```
<script type="text/javascript">
    window.onload = function(){
        //使用JavaScript代码创建HTML标签，并显示在页面上
        var divObj = document.createElement("div");
        var divObj.innerHTML = "创建div标签";
        document.body.appendChild(divObj);
    }
</script>
```

文末[1]

返回文首

---

1. 点击到达文末。 ↩