01_JavaScript函数

- 函数就是方法,它们都是完成一段特定功能的代码块。对于函数必须调用它才可以执行。
- A.创建函数
  - 第一种方式:使用function关键字

```
function 函数名(形式参数列表){
    函数体;
}
```

  - 第二种方式

```
var 方法名=function(形式参数列表){
    函数体;
};
```

  - 第三种方式:使用javascript对象Function

```
var 函数名称=new Function(参数,函数体);
```

- B.注意事项
  - a.参数
    - js中的方法在定义时，

02_JavaScript事件

- A.基本概念：
  - Javascript中事件一般是与函数配合使用，当事件产生时，相对应的函数可以执行。
  - 事件：就是一件事情，例如点击按钮。
  - 事件源:它是事件产生的源头。按钮是事件源
  - 监听器:它是用于监听特定事件的组件，当事件产生时，监听器就可以执行。
  - 注册监听(绑定监听):是将事件源与监听器绑定到一起，当事件源产生了这个事件，监听器就可以知道 并执行相应的操作。
- B.事件注册(绑定事件)方式:
  - a.元素属性绑定

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>

        function  fn1() {
           console.log("点击了");
        }

    </script>
```

```
</head>
<body>
    <button id="btn" onclick="fn1"></button>
</body>
```

- b.dom分配事件

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        //监听页面加载完成
        window.onload = function () {
            document.getElementById("btn").onclick = function () {
                console.log("点击了");
            }
        }

    </script>
</head>
<body>
    <button id="btn" ></button>
</body>
```

- C.常见事件
  - onclick 鼠标点击某个元素
  - onload 代表页面或图片加载完成后
  - 焦点操作
    - onfocus 元素获取焦点
    - onblur 元素失去焦点

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function fn1() {
            console.log("1获取焦点")
        }
        function fn2() {
            console.log("1失去焦点")
        }

        function fn3() {
            console.log("2获取焦点")
        }
        function fn4() {
            console.log("2失去焦点")
        }
    </script>
</head>
<body>
    <input type="text" onfocus="fn1()" onblur="fn2()"/>
```

```
    <input type="text" onfocus="fn3()" onblur="fn4()" />
</body>
```

- 关于键盘操作
onkeydown	某个键盘的键被按下
onkeypress	某个键盘的键被按下或按住
onkeyup	某个键盘的键被松开

```
<head>
    <script>
        function fn1(){
            console.log(event.keyCode);
        }
    </script>
</head>
<body>
    <input onkeydown="fn1()" />
</body>
```

- 关于鼠标操作
onmousedown	某个鼠标按键被按下
onmousemove	鼠标被移动
onmouseout	鼠标从某元素移开
onmouseover	鼠标被移到某元素之上
onmouseup	某个鼠标按键被松开
- onchange 代表用户改变域的内容
  - 比如:下拉框、checkbox等等
- onsubmit 监听

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>

        function fn1() {
            console.log("提交了...")
//          return false;//拦截表单内容。
            return true;//不拦截表单内容
        }
    </script>
</head>
<body>
<form onsubmit="return fn1()">
    用户名：<input type="text" name="username"/><br/>
    <button type="submit" id="btn" >提交</button>
</form>
```

03_JavaScript中的bom及Window对象

- browser object model:浏览器对象模型
- A.bom对象
  - Window对象
  - Navigator对象
  - Screen对象
  - History对象
  - Location对象
- B.Window对象
  - a.常用属性
    - 可以通过Window对象获取其他四个bom对象
  - b.常用方法

Window 对象方法

方法	描述	IE	F	O
alert()	显示带有一段消息和一个确认按钮的警告框。	4	1	9
blur()	把键盘焦点从顶层窗口移开。	4	1	9
clearInterval()	取消由 setInterval() 设置的 timeout。	4	1	9
clearTimeout()	取消由 setTimeout() 方法设置的 timeout。	4	1	9
close()	关闭浏览器窗口。	4	1	9
confirm()	显示带有一段消息以及确认按钮和取消按钮的对话框。	4	1	9
createPopup()	创建一个 pop-up 窗口。	4	No	No
focus()	把键盘焦点给予一个窗口。	4	1	9
moveBy()	可相对窗口的当前坐标把它移动指定的像素。	4	1	9
moveTo()	把窗口的左上角移动到一个指定的坐标。	4	1	9
open()	打开一个新的浏览器窗口或查找一个已命名的窗口。	4	1	9
print()	打印当前窗口的内容。	5	1	9
prompt()	显示可提示用户输入的对话框。	4	1	9
resizeBy()	按照指定的像素调整窗口的大小。	4	1	9
resizeTo()	把窗口的大小调整到指定的宽度和高度。	4	1.5	9
scrollBy()	按照指定的像素值来滚动内容。	4	1	9
scrollTo()	把内容滚动到指定的坐标。	4	1	9
setInterval()	按照指定的周期（以毫秒计）来调用函数或计算表达式。	4	1	9
setTimeout()	在指定的毫秒数后调用函数或计算表达式。	4	1	9

  - c.练习
    - setInterval方法、clearInterval方法

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script type="application/javascript">
        var inter;
        window.onload = function () {
            inter = window.setInterval("showTime1()",1000);
        }
        function showTime1() {
            var time = document.getElementById("span1");
            time.innerHTML = new Date().toLocaleString();
        }

        function stop1() {
            window.clearInterval(inter);
        }
    </script>
</head>
<body>
    当前时间为:<span id="span1"></span><br/>
    <button onclick="stop1()">停止</button>
</body>
```

- setTimeout方法、clearTimeout方法

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <script>
        var timeOut;
        window.onload = function () {
            window.setTimeout("showTime1()",1000)
        }
        function showTime1() {
            var span1 = document.getElementById("span1");
            span1.innerHTML = new Date().toLocaleString();
            timeOut = window.setTimeout("showTime1()",1000);
        }
        function stop1(){
            window.clearTimeout(timeOut);
        }
    </script>
</head>
<body>
    当前时间为:<span id="span1"></span><br/>
    <button onclick="stop1()">停止</button>
</body>
```

04_Location对象

- A.Location对象
  - Location 对象包含有关当前 URL 的信息。

- a.常用属性

  Location 对象属性

属性	描述	IE	F	O
hash	设置或返回从井号 (#) 开始的 URL（锚）。	4	1	9
host	设置或返回主机名和当前 URL 的端口号。	4	1	9
hostname	设置或返回当前 URL 的主机名。	4	1	9
href	设置或返回完整的 URL。	4	1	9
pathname	设置或返回当前 URL 的路径部分。	4	1	9
port	设置或返回当前 URL 的端口号。	4	1	9
protocol	设置或返回当前 URL 的协议。	4	1	9
search	设置或返回从问号 (?) 开始的 URL（查询部分）。	4	1	9

- b.常用方法

  Location 对象方法

属性	描述	IE	F	O
assign()	加载新的文档。	4	1	9
reload()	重新加载当前文档。	4	1	9
replace()	用新的文档替换当前文档。	4	1	9

- c.练习

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function fn1() {
            //跳转到index.html
            location.href = "index.html";
        }
    </script>
</head>
<body>
    <button onclick="fn1()"></button>
</body>
```

05_History对象

- History 对象包含用户（在浏览器窗口中）访问过的 URL。
- 代码实现
  - demo02.jsp

```
<head>
    <title>demo02</title>
    <script>
```

```
        function forward() {
            history.forward();
        }

    </script>
</head>
<body>
    <a href="demo03.jsp">跳转到demo03</a>
    <button onclick="forward()">下一个</button>
</body>
```

- demo03.jsp

```
<head>
    <title>demo03</title>
    <script>

        function back() {
            history.back();
        }

        function forward() {
            history.forward();
        }

    </script>
</head>
<body>
    <button onclick="back()">上一个</button>
    <a href="demo04.jsp">跳转到demo04</a>
    <button onclick="forward()">下一个</button>
</body>
```

- demo04.jsp

```
<head>
    <title>demo04</title>
    <script>

        function back() {
            history.back();
        }

    </script>
</head>
<body>
    <button onclick="back()">上一个</button>
</body>
```

06_Xml Dom

- A.核心dom模型

    - Document:文档对象
    - Element:元素对象
    - Node:节点对象
- B.Document文档对象
    - a.创建(获取)：可以使用window对象来获取
        - window.document
        - 直接document
    - b.方法：
        - 获取Element对象：
            - getElementById():
                - 根据id属性值获取元素对象。id属性值一般唯一
            - getElementsByTagName()：
                - 根据元素名称获取元素对象们。返回值是一个数组
        - 创建其他DOM对象：
            - createAttribute(name)
                - 创建属性对象
            - createElement()
                - 创建元素对象
            - createTextNode()
                - 创建文本对象
    - c.练习
        - 演示获取Element对象
- C.Element元素对象
    - a.常用方法：
        - getAttribute():获取属性值
        - removeAttribute()：删除属性
        - setAttribute()：设置属性
    - b.练习
        - 需求1:获取输入框中的name属性值
        - 需求2:设置输入框中的name属性值
        - 需求3:移除输入框中的name属性
    - c.注意事项
        - 使用getAttribute获取属性值时,该属性必须显式的定义在标签上。

```
<head>
    <meta charset="UTF-8">
    <title>Element对象</title>
    <script>

        function fn1() {
            var ele = document.getElementById("username");
            var username1 = ele.getAttribute("name");
            console.log("username1:"+username1);
```

```
        }

        function fn2() {
            var ele = document.getElementById("username");
            ele.setAttribute("name","username1");
        }

        function fn3() {
            var ele = document.getElementById("username");
            ele.removeAttribute("name");
        }

    </script>
</head>
<body>
    <input value="默认" type="text" id="username"  name="username"/>
    <button onclick="fn1()">获取</button>
    <button onclick="fn2()">设置</button>
    <button onclick="fn3()">移除</button>
</body>
```

- D.Node节点对象
- 节点对象，可以是元素节点、属性节点、文本节点
  - a.特点：所有dom对象都可以被认为是一个节点
  - b.方法:
    - appendChild()：
      - 向节点的子节点列表的结尾添加新的子节点。
    - removeChild()
      - 删除（并返回）当前节点的指定子节点。
    - 练习
      - 需求1:给父div中添加一个子div，子div中的内容为”学好java”；
      - 需求2:移除父div中指定的子元素

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>

        function fn1() {
            let parentDiv = document.getElementById("parent");
            //创建子div
            let childDiv = document.createElement("div");
            //给子div设置id属性
            childDiv.setAttribute("id","child");
            //创建文本
            let text = document.createTextNode("学好java");
            //往子div中添加文本
            childDiv.appendChild(text);
            //往父div中添加子div
            parentDiv.appendChild(childDiv);
        }

        function fn2() {
```

```
            let parentDiv = document.getElementById("parent");
            let childDiv = document.getElementById("child");
            console.log(childDiv);
            if(null != childDiv){
                parentDiv.removeChild(childDiv);
            }
        }
    </script>
</head>
<body>
    <div id="parent"></div>
    <button onclick="fn1()">添加</button>
    <button onclick="fn2()">删除</button>
</body>
```

- 属性：
  - parentNode 返回节点的父节点。
  - nodeType 节点的类型

    最重要的节点类型是：

元素类型	节点类型
元素	1
属性	2
文本	3
注释	8
文档	9

  - 练习：
    - 需求1：如下代码，在id为"child1"的子div的父div中添加其他的子div.

```
<body>
    <div >
        <div id="child1">子div1</div>
    </div>
    <button onclick="fn1()">添加</button>
</body>
```

    - 实现1:

```
<script>
    function fn1() {
        //1,创建子div
        let child2 = document.createElement("div");
        //2,创建文本节点,内容为"子div2"
        let textNode = document.createTextNode("子div2");
        //3,将文本节点添加到创建的子div中
```

```
        child2.appendChild(textNode);
        //4,获取id为"child1"的子div
        let child1 = document.getElementById("child1");
        //5,获取到子div对应的父节点
        let parent = child1.parentNode;
        //6,将新创建div添加到父div中
        parent.appendChild(child2);
    }
</script>
```

- 需求2:如下代码，获取父div中的子元素的类型

```
<div id="parent">
    这是一个div1
    <div>这是一个div2</div>
</div>
```

- 实现2:

```
<script>
    function fn1() {
        var parent = document.getElementById("parent");
        var childNodes = parent.childNodes;
        for (var i = 0; i < childNodes.length; i++) {
            var childNode = childNodes[i];
            console.log(childNode.nodeType);
        }
    }
</script>
```

07_Html Dom

- HTML DOM 定义了访问和操作HTML文档的标准方法。HTML DOM 把 HTML 文档呈现为带有元素、属性和文本的树结构（节点树）。
- A.innerHTML和innerText
- B.使用html元素对象的属性
  - 需求:动态获取文本输入框中的内容
  - 实现:

```
function fn1() {
    var ele = document.getElementsByTagName("input")[0];
    var value = ele.value;
    console.log(value);
}
```

- C.控制元素样式
  - 需求:修改div元素的边框、宽度、内容字体大小.
  - 实现:

```
function fn1() {
    let element = document.getElementsByTagName("div")[0];
    element.style.width = "100px";
    element.style.border="1px solid red";
    element.style.fontSize = "20px";
}
```

08_Js综合案例1

- A.需求:得到select下的所有的option中的文本信息

```
<select>
    <option value="xx">小学</option>
    <option value="cz">初中</option>
    <option value="gz">高中</option>
    <option value="dx">大学</option>
</select>
```

B.实现:

```
//xml dom方式
function fn1() {
    var selectElement = document.getElementsByTagName("select")[0];
    var childNodes = selectElement.childNodes;
    for (var i = 0; i < childNodes.length; i++) {
        var childNode = childNodes[i];
        if(childNode.nodeType == 1){
            var nodeValue = childNode.firstChild.nodeValue;
            console.log(nodeValue);
        }
    }
}
//html dom方式
function fn2() {
    var selectElement = document.getElementsByTagName("select")[0];
    var options = selectElement.options;
    for (var i = 0; i < options.length; i++) {
        var option = options[i];
        var val = option.value;
        console.log(val);
    }
}
```

09_Js综合案例2

- A.需求:得到select下的选中的option中的文本信息和value属性的值

```
function fn3() {
    var selectElement = document.getElementsByTagName("select")[0];
    var options = selectElement.options;
    var selectedIndex = selectElement.selectedIndex;
    console.log(options[selectedIndex].text+","+options[selectedIndex].value)
}
```

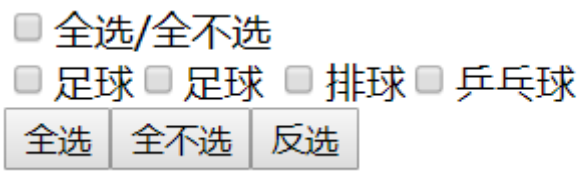
10_Js综合案例3

- A.需求:在select下增加一个选项"硕士"
- B.实现

```
 function fn4() {
    var option = document.createElement("option");
    option.value = "ss";
    option.text="硕士";
    var selectElement = document.getElementsByTagName("select")[0];
    selectElement.add(option);

}
```

11_Js综合案例4

- A.需求:如下图所示
- 

    - 当选择 全选/全不选时，下面的四个爱好，都会与我们全选/全不选一样。
    - 当选择全选按钮时，要求四个爱好项全都选择。
    - 当选择全不选按钮时，要求四个爱好项全都不选择。
    - 当选择反选时，要求四个爱好项，选择的取消，没有选择的选中。

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function fn1() {
            console.log("全选");
            //全选
            var eles = document.getElementsByTagName("input");
            for (var i = 0; i < eles.length; i++) {
                if(eles[i].id != "all"){
                    eles[i].checked = true;
                }
```

```
            }
        }

        function fn2() {
            //全不选
            var eles = document.getElementsByTagName("input");
            for (var i = 0; i < eles.length; i++) {
                if(eles[i].id != "all") {
                    eles[i].checked = false;
                }
            }
        }

        function fn3() {
            //反选
            var eles = document.getElementsByTagName("input");
            for (var i = 0; i < eles.length; i++) {
                if(eles[i].id != "all") {
                    eles[i].checked = !eles[i].checked;
                }
            }
        }

        function fn4() {
            var eles = document.getElementsByTagName("input");
            var firstCheckBox = eles[0];
            for (var i = 1; i < eles.length; i++) {
                eles[i].checked = firstCheckBox.checked;
            }
        }
    </script>
</head>
<body>
    <input id="all" type="checkbox" onchange="fn4()"/>全选/全不选<br/>
    <input id="football" type="checkbox"/>足球<input id="basketball"
type="checkbox"/>足球
    <input id="volleyball" type="checkbox"/>排球<input id="pingpong"
type="checkbox"/>乒乓球<br/>
    <button onclick="fn1()">全选</button>
    <button onclick="fn2()">全不选</button>
    <button onclick="fn3()">反选</button>
</body>
```

12_Js综合案例5

- A.案例：
  - a.完成如下图表单

邮箱:
用户名:
密码
重复密码:
提交

- b.要求:
  - 所有内容不可以为空
  - 邮箱必须邮箱的规则
  - 用户名与密码长度必须6位以上
  - 密码与重复密码必须一致
- c.正则表达式
  - 为空 /^\s*$/
  - 邮箱格式 /^(\w)+@(\w)+(.\w+)+$/
  - 长度必须6位以上 /^.{6,}$/
- d.代码实现
  - 页面搭建

```
<form onsubmit="return checkInfo()">
    账户:<input type="text" name="username" /><span id="username_msg"></span>
<br/>
    密码:<input type="text" name="password"/><span id="password_msg"></span><br/>
    确认密码:<input type="text" name="repassword"/><span id="repassword_msg">
</span><br/>
    邮箱:<input type="text" name="email"/><span id="email_msg"></span><br/>
    <button type="submit">提交</button>
</form>
```

  - 校验内容是否为空

```
function checkNull(fieldName) {
    console.log(fieldName)
    var ele = document.getElementsByName(fieldName)[0];
    var reg =/^\s*$/;
    if (reg.test(ele.value)) {
        var spanEle = document.getElementById(fieldName+"_msg");
        spanEle.innerHTML="<font color='red'>"+fieldName+"不能为空</font>";
        return false;
    } else {
        return true;

    }
}
```

  - 清空之前的警告

```
function clearSpan() {
    var spans = document.getElementsByTagName("span");
```

```
    for (var i = 0; i < spans.length; i++) {
        spans[i].innerHTML = "";
    }
}
```

- 校验账户和密码长度

```
function checkLength(fieldName) {
    var ele = document.getElementsByName(fieldName)[0];
    if(/^.{6,}$/.test(ele.value)){
        return true;
    }else {
        var spanEle = document.getElementById(fieldName+"_msg");
        spanEle.innerHTML="<font color='red'>"+fieldName+"长度不对</font>";
        return false;
    }
}
```

- 校验密码是否一致

```
function checkPassword(){
    var ele1 = document.getElementsByName("password")[0];
    var ele2 = document.getElementsByName("repassword")[0];
    if(ele1.value == ele2.value){
        return true;
    } else {
        var span = document.getElementById("repassword_msg");
        span.innerHTML = "<font color='red'>密码不一致!</font>";
        return false;
    }
}
```

- 校验邮箱格式

```
function checkEmail() {
    var ele = document.getElementsByName("email")[0];
    if(/^(\w)+@(\w)+(.\w+)+$/.test(ele.value)){
        return true;
    }else {
        let span = document.getElementById("email_msg");
        span.innerHTML = "<font color='red'>邮箱格式不对!</font>";
        return false;
    }
}
```

- checkInfo方法

```
function checkInfo() {
    clearSpan();
    return checkNull("username") && checkLength("username") &&
        checkNull("password") && checkLength("password") &&
        checkNull("repassword") && checkLength("repassword") && checkPassword()
&&
```

```
        checkNull("email") && checkEmail();
}
```