02-JavaScript 语言入门

今日任务

1. JavaScript 语言

课程内容

1. 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。

JS 是 Netscape 网景公司的产品，最早取名为 LiveScript;为了吸引更多 java 程序员。更名为 JavaScript。

JS 是弱类型，

Java 是强类型。

特点：

1. 交互性（它可以做的就是信息的动态交互）
2. 安全性（不允许直接访问本地硬盘）
3. 跨平台性（只要是可以解释 JS 的浏览器都可以执行，和平台无关）

2. JavaScript 和 html 代码的结合方式

2.1 第一种方式

只需要在 head 标签中，或者在 body 标签中， 使用 script 标签 来书写 JavaScript 代码

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

    <script type="text/javascript">
        //alter 是 JavaScript 提供的一个提示框函数
        //它可以接收任意类型的参数,参数就是提示框的提示信息.
```

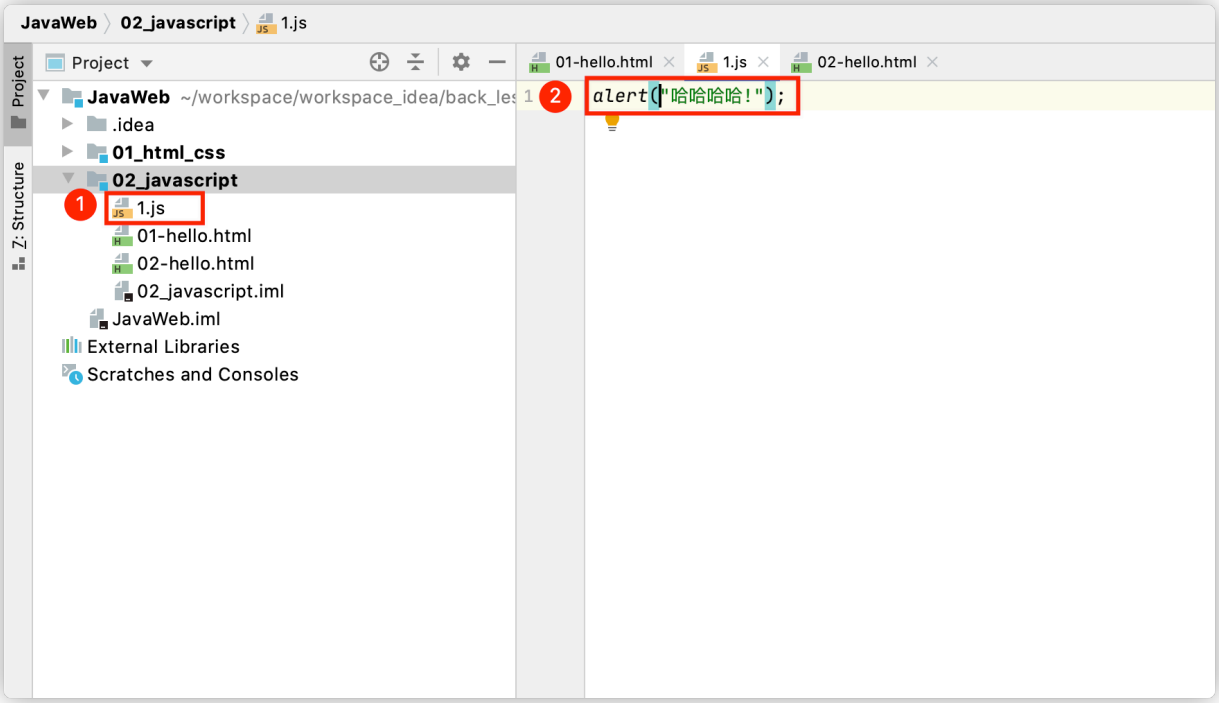
```
        alert("Hello JavaScript")
    </script>

</head>
<body>

</body>
</html>
```

2.2 第二种方式

使用 script 标签引入 单独的 JavaScript 代码文件




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <!--
        现在需要使用 script 引入外部js文件来执行
            src 属性用来指定js文件的路径(可以是相对路径,也可以是绝对路径)

        script 标签既可以用来定义js代码,也可以用来引入js文件
        注意:两个功能只能用一个.
    -->
    <script type="text/javascript" src="1.js"></script>
    <script type="text/javascript">
        alert("白哥现在更帅了!")
    </script>
</head>
```

```
<body>

</body>
</html>
```

3. 变量

什么是变量？变量是可以存放某些值的内存的命名。

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 里特殊的值：
undefined 未定义，所有 js 变量未赋于初始值的时候，默认值都是 undefined.
null 空值
NaN 全称是：Not a Number。非数字。非数值。

JS 中的定义变量格式：
var 变量名;
var 变量名 = 值;

示例代码:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>3.变量</title>

    <script type="text/javascript">
        var i;
        // alert(i);//undefined
        i = 12;
        // typeof() 是JS提供的一个函数
        // typeof() 它可以取变量的数据类型返回
        // alert(typeof(i));//

        i = "abc";
        alert(typeof(i));//

        var a = 10;
        var b = "abc";
```

```
        alert(a*b);// NaN 是非数字,非数值

    </script>

</head>
<body>

</body>
</html>
```

4. 关系（比较）运算

等于：　　==　　等于是简单的做字面值的比较

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

示例代码:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>4.关系比较运算</title>

    <script type="text/javascript">
        var a = 12;
        var b = "12";

        alert(a==b);
        alert(a===b);

    </script>

</head>
<body>

</body>
</html>
```

5. 逻辑运算

且运算：&&
或运算：||
取反运算：！

在 JavaScript 语言中，所有的变量，都可以做为一个 boolean 类型的变量去使用。
0 、null、 undefined、””(空串) 都认为是 false；

```
/*
  && 且运算。
  有两种情况：
    第一种：当表达式全为真的时候。返回最后一个表达式的值。
    第二种：当表达式中，有一个为假的时候。返回第一个为假的表达式的值

  || 或运算
    第一种情况：当表达式全为假时，返回最后一个表达式的值
    第二种情况：只要有一个表达式为真。返回第一个为真的表达式的值

  并且 && 与运算 和 ||或运算 有短路。
  短路就是说，当这个&&或||运算有结果了之后 。后面的表达式不再执行
*/

var a = "abc";
var b = true;
var d = false;
var c = null;
```

示例代码

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>6.逻辑运算符</title>

    <script type="text/javascript">

        /* 在 JavaScript 语言中，所有的变量，都可以做为一个 boolean 类型的变量去使用。
0 、null、 undefined、””(空串) 都认为是 false；*/
        // var a = 0;
        // if (a) {
        //     alert("零为真");
        // } else {
        //     alert("零为假");
        // }
        // var b = null;
        // if (b) {
        //     alert("null 为真");
        // } else {
        //     alert("null 为假");
        // }
        // var c = undefined;
        // if (c) {
```

```
        //     alert("undefined 为真");
        // } else {
        //     alert("undefined 为假");
        // }
        // var d = "";
        // if (d) {
        //     alert("空串为真");
        // } else {
        //     alert("空串为假");
        // }


        var a = "abc";
        var b = true;
        var d = false;
        var c = null;

        /*
            && 且运算。
            有两种情况:
                第一种:当表达式全为真的时候。返回最后一个表达式的值。
                第二种:当表达式中,有一个为假的时候。返回第一个为假的表达式的值
         */
        // alert(a && b);//true
        // alert(b && a);//abc

        // alert(a && d);//false
        // alert(a && c);//null


        /*
            || 或运算
                第一种情况:当表达式全为假时,返回最后一个表达式的值
                第二种情况:只要有一个表达式为真。就会把回第一个为真的表达式的值
         */
        // alert(d || c);//null
        // alert(c || d);//false

        // alert(a || d);//abc
        // alert(b || d);//true


    </script>

</head>
<body>

</body>
</html>
```

6. 数组

6.1数组定义方式

JS 中 数组的定义：
格式：
　var 数组名 = []; // 空数组
　var 数组名 = [1 , 'abc' , true]; // 定义数组同时赋值元素

示例代码:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>6.数组</title>

    <script type="text/javascript">

        var arr = [];

        arr[0] = 10;

        arr[2] = 30;
        arr[3] = "abc"
        // alert(arr[3]);

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

</head>
<body>

</body>
</html>
```

7. 函数

7.1 函数的第一种定义方式

使用的格式如下:

function 函数名(形参列表){
    函数体
}

在 JavaScript 语言中，如何定义带有返回值的函数?
只需要在函数体内直接使用 return 语句返回值即可!

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>7.函数-1</title>

    <script type="text/javascript">
        //定义无参函数
        function fun1() {
            alert("无参函数被调用......")
        }

        //函数被调用才会执行
        // fun1();

        //定义带参数的函数
        function fun2(a,b) {
            alert("有参函数被调用 a=>" + a + ", b=>" + b);
        }
        // fun2(22,"xxx");

        //定义带返回值的函数
        function fun3(num1,num2) {
            var result = num1 + num2;
            return result;
        }

        alert(fun3(10,20));

    </script>

</head>
<body>

</body>
</html>
```

7.1 函数的第二种定义方式

var 函数名 = function(形参列表) { 函数体 }

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>7.函数-2</title>
    <script type="text/javascript">
        var fun = function () {
            alert("无参函数");
        }
        // fun();
        var fun2 = function (a, b) {
            alert("有参函数 a=" + a + ",b=" + b);
        }
        // fun2(1,2);
        var fun3 = function (num1, num2) {
            return num1 + num2;
        }
        alert(fun3(100, 200));
    </script>
</head>
<body>

</body>
</html>
```

> 注：在 Java 中函数允许重载。但是在 JS 中函数的重载会直接覆盖掉上一次的定义

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>8.函数重载</title>

    <script type="text/javascript">

        function fun() {
            alert("无参函数fun()");
        }

        function fun(a,b) {
            alert("有参函数fun()");
        }

        fun();
    </script>
```

```
</head>
<body>

</body>
</html>
```

7.3 函数的 arguments 隐形参数（只在 function 函数内）

就是在 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。我们管它叫隐形参数。

隐形参数特别像 java 基础的可变长参数一样。

```
public void fun( Object ... args ){}
```

可变长参数其实是一个数组。

js 中的隐形参数也跟 java 的可变长参数一样。操作类似数组。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>9-可变长参数arguments</title>
    <script type="text/javascript">
        function fun(a,b) {
            alert(arguments.length);
            // alert(arguments[0]);
            // alert(arguments[1]);
            // alert(arguments[2]);

            alert(a);
            alert(b);


            for(var i = 0; i < arguments.length; i++){
                alert("循环-->" + arguments[i]);
            }
        }
        // fun(11,"abc",true);

        function sum(num1,num2) {
            var result = 0;
            for (var i = 0; i < arguments.length; i++) {
                if (typeof(arguments[i]) == "number"){
                    result+=arguments[i];
                }
```

```
            }
            return result;
        }
        alert(sum(1,2,3,4,"aaa",5,6,7,8,9,10));
    </script>
</head>
<body>

</body>
</html>
```

8. JS 中的自定义对象（扩展内容）

Object形式的自定义对象

对象的定义：

```
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
```

对象的访问：

```
变量名.属性 / 函数名();
```

示例代码:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>10-Object形式的自定义对象</title>
    <script type="text/javascript">
        /*
        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
         */
        var person = new Object();
        // alert(typeof(person));
        person.name = "小白白";
        person.age = 18;
```

```
        person.say = function(){
            alert("姓名:" + this.name + ", 年龄:" + this.age);
        }

        // alert(person.age);
        person.say();

    </script>
</head>
<body>

</body>
</html>
```

{}花括号形式的自定义对象

对象的定义:

```
var 变量名 = { // 空对象

  属性名: 值, // 定义一个属性

  属性名: 值, // 定义一个属性

  函数名: function(){} // 定义一个函数

};
```

对象的访问:

```
变量名.属性 / 函数名();
```

示例代码:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>11-{}形式的自定义对象</title>
    <script type="text/javascript">
        /*
        var 变量名 = { // 空对象
            属性名: 值, // 定义一个属性
            属性名: 值, // 定义一个属性
            函数名: function(){} // 定义一个函数
        };
```

```
         */
        var obj = {
            name:"小花花",
            age:20,

            fun:function () {
                alert("姓名:" + this.name + ", 年龄:" + this.age);
            }
        };
        // alert(typeof(obj));
        obj.fun();


    </script>
</head>
<body>

</body>
</html>
```

9、js 中的事件

什么是事件？事件是电脑输入设备与页面进行交互的响应。我们称之为事件。

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 js 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 内容发生改变事件： 常用于下拉列表和输入框内容发生改变后操作
- 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。

事件的注册又分为静态注册和动态注册两种：

什么是事件的注册（绑定）？

其实就是告诉浏览器，当事件响应后要执行哪些操作代码，叫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于事件响应后的代码，这种方式我们叫静态注册。
- **动态注册事件：**是指先通过 js 代码得到标签的 dom 对象，然后再通过 dom 对象.事件名 = function(){} 这种形式赋于事件

响应后的代码，叫动态注册。

动态注册基本步骤：

1、获取标签对象

2、标签对象.事件名 = fucntion(){}

onload 加载完成事件

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>onload</title>
  </head>
  <!-- onload时间是浏览器在解析完页面之后会自动触发的事件 -->
  <body onload="alert('静态注册onload事件')">
  </body>
</html>
```

通常我们会在页面加载完成之后统一处理.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>onload</title>
    <script type="text/javascript">
      function onLoadFun(){
        alert('静态注册onload事件,还有好多代码');
      }
    </script>
  </head>
  <!-- onload时间是浏览器在解析完页面之后会自动触发的事件 -->
  <body onload="onLoadFun()">
  </body>
</html>
```

onload事件的动态注册

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

```
  <head>
    <meta charset="utf-8">
    <title>onload</title>
    <script type="text/javascript">
      // function onLoadFun(){
      //  alert('静态注册onload事件,还有好多代码');
      // }

      //onload事件的动态注册代码写法的固定的(类似于main方法)
      window.onload = function(){
        alert("动态注册oload事件");
      }
    </script>
  </head>
  <!-- onload时间是浏览器在解析完页面之后会自动触发的事件 -->
  <body>
  </body>
</html>
```

onclick

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

    <script type="text/javascript">
      function onclickFun(){
        alert("静态注册onclick事件");
      }

      window.onload = function(){
        //动态注册onclick事件
        //1.获取标签对象
        //2.通过标签对象.事件名=function(){}
        var btnObj = document.getElementById("btn001");
        btnObj.onclick = function(){
          alert("动态注册的onclick事件");
        }
        alert(btnObj);
      }
    </script>
  </head>
  <body>
    <!-- 静态注册 -->
    <button onclick="onclickFun()">按钮1</button>
```

```
    <button id="btn001">按钮2</button>
  </body>
</html>
```

onblur

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

    <script type="text/javascript">
        function onBlurFunction() {
            //consloe 是控制台对象.JS提供
            console.log("静态注册失去焦点事件");
        }

        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>
    用户账号:<input type="text" onblur="onBlurFunction()"/><br/>
    登陆密码:<input type="password" id="password"/>
</body>
</html>
```

onchange

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

    <script type="text/javascript">
        function onChangeFunction() {
            alert("内容改变了!");
```

```
        }
        window.onload = function (){
            var set = document.getElementById("set01")
            set.onchange = function (){
                alert("男星内容改变了");
            }
        }
    </script>
</head>
<body>
    请选择你喜爱的女星:
    <select onchange="onChangeFunction()">
        <option>-- 请选择 --</option>
        <option>苍井空</option>
        <option>泷泽萝拉</option>
        <option>小泉彩</option>
    </select>
    <br />
    请选择你喜爱的男星:
    <select id="set01">
        <option>-- 请选择 --</option>
        <option>白哥</option>
        <option>小白哥</option>
        <option>白老师</option>
    </select>
</body>
</html>
```

onsubmit

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script type="text/javascript">
        function onsubmitFunction() {
            alert("静态注册表单提交事件");
            return false
        }

        window.onsubmit = function (){
            var btn_submit = document.getElementById("form01");
            btn_submit.onsubmit = function (){
                alert("动态注册表单提交事件")
                return false;
            }
        }
```

```
    </script>

</head>
<body>
    <form action="http://localhost:8080" method="get" onsubmit="return
onsubmitFunction()">
        <input type="submit" value="静态注册">
    </form>
    <form action="http://localhost:8080" method="get" id="form01">
        <input type="submit" value="动态注册">
    </form>
</body>
</html>
```

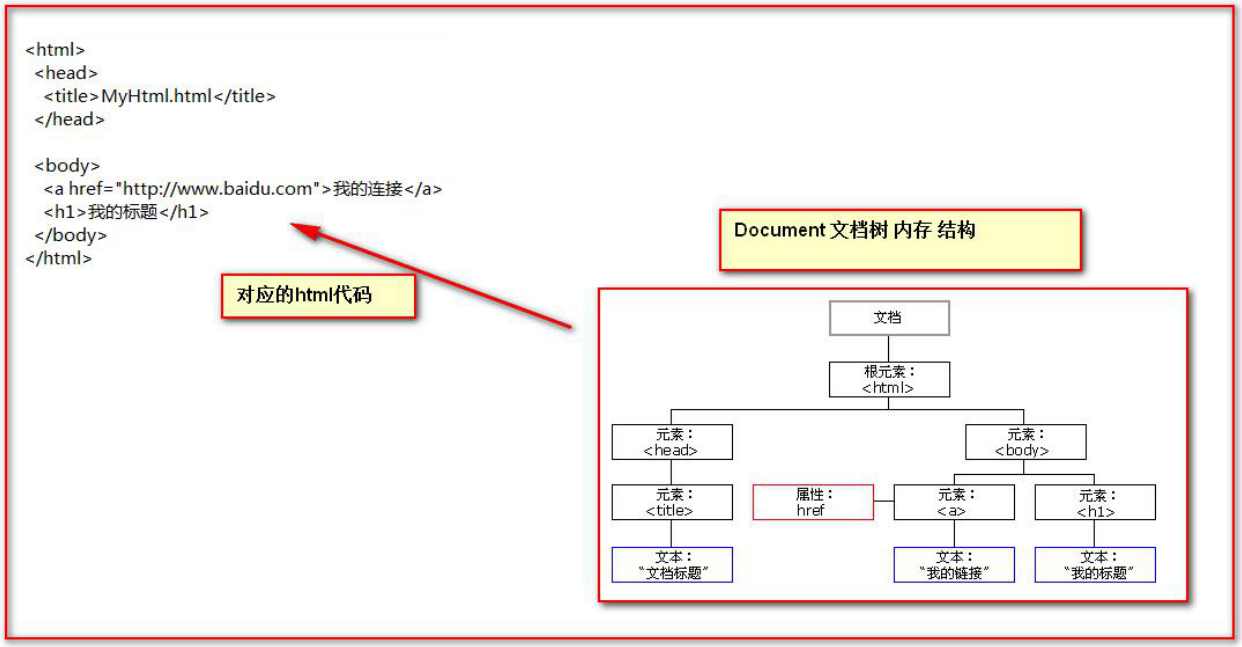
10、DOM 模型

DOM 全称是 Document Object Model 文档对象模型

大白话，就是把文档中的标签，属性，文本，转换成为对象来管理。

那么 它们是如何实现把标签，属性，文本转换成为对象来管理呢。这就是我们马上要学习的重点。

10.1、Document 对象（*重点）



Document 对象的理解：

- 第一点：Document 它管理了所有的 HTML 文档内容。
- 第二点：document 它是一种树结构的文档。有层级关系。
- 第三点：它让我们把所有的标签 都 对象化
- 第四点：我们可以通过 document 访问所有的标签对象。

什么是对象化？？

我们已经学过面向对象。

请问什么是对象化?

举例：

有一个人有年龄：18 岁，性别：女，名字：张某某 我们要把这个人的信息对象化怎么办!

```
Class Person {
  private int age;
  private String sex;
  private String name;
}
```

模拟对象化，相当于：

```
class Dom{
    private String id; // id 属性
    private String tagName; //表示标签名
    private Dom parentNode; //父亲
    private List<Dom> children; // 孩子结点
    private String innerHTML; // 起始标签和结束标签中间的内容
}
```

10.4、Document对象中的方法介绍（重点）

```
document.getElementById(elementId)
```

通过标签的 id 属性查找标签 dom 对象，elementId 是标签的 id 属性值

```
document.getElementsByName(elementName)
```

通过标签的 name 属性查找标签 dom 对象，elementName 标签的 name 属性值

```
document.getElementsByTagName(tagname)
```

通过标签名查找标签 dom 对象。tagname 是标签名

```
document.createElement( tagName)
```

方法，通过给定的标签名，创建一个标签对象。tagName 是要创建的标签名

注：

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** 方法来进行查询
如果 id 属性和 name 属性都没有最后再按标签名查 **getElementsByTagName**

以上三个方法，一定要在页面加载完成之后执行，才能查询到标签对象。

getElementById 方法示例代码：

验证用户名是否合法

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script type="text/javascript">
        /*
        需求:当用户点击了注册按钮.需要获取输入框的内容,然后验证是否合法
        验证规则:必须由字母 数组 下划线组成.并且长度为6-12位
         */
        function onclickFun(){
            var usernameObj = document.getElementById("username");
            var textUserName = usernameObj.value;
            var patt = /^\w{6,12}$/
            if(patt.test(textUserName)){
                alert("用户名合法");
            }else{
                alert("用户名不合法");
            }
        }
    </script>
</head>
<body>
    用户名<input id="username" type="text">
    <button onclick="onclickFun()">注册</button>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script type="text/javascript">
        /*
```

```
        需求:当用户点击了注册按钮.需要获取输入框的内容,然后验证是否合法
        验证规则:必须由字母 数组 下划线组成.并且长度为6-12位
         */
        function onclickFun(){
            var usernameObj = document.getElementById("username");
            var textUserName = usernameObj.value;
            var patt = /^\w{6,12}$/

            var usernameSpanObj = document.getElementById("usernameSpan");
            // alert(usernameSpanObj.innerHTML);

            if(patt.test(textUserName)){
                // alert("用户名合法");
                // usernameSpanObj.innerHTML = "用户名合法"
                usernameSpanObj.innerHTML = "<img src=\"right.png\"
width=\"14px\"  height=\"14px\">"
            }else{
                // usernameSpanObj.innerHTML = "用户名不合法"
                usernameSpanObj.innerHTML = "<img src=\"wrong.png\"
width=\"14px\"  height=\"14px\">"
                // alert("用户名不合法");
            }
        }
    </script>
</head>
<body>
    用户名<input id="username" type="text">
    <span id="usernameSpan" style="color: red">
        <img src="" width="14px"  height="14px">
    </span>
    <button onclick="onclickFun()">注册</button>
</body>
</html>
```

getElementsByName

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

    <script type="text/javascript">
        function checkAll() {
            var hobbies = document.getElementsByName("hobby");
            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].checked){
                    hobbies[i].checked = false;
                }else {
                    hobbies[i].checked = 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="python">Python
    <button onclick="checkAll()">全选</button>
    <button onclick="checkNo()">全不选</button>
    <button onclick="checkReverse()">反选</button>
</body>
</html>
```

getElementsByTagName

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

    <script type="text/javascript">
        function checkAll() {
            var inputs = document.getElementsByTagName("input");
            for(var i = 0; i < inputs.length; i++){
                inputs[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="python">Python
    <button onclick="checkAll()">全选</button>
</body>
</html>
```

10.5、节点的常用属性和方法

节点就是标签对象

方法：

通过具体的元素节点调用 getElementsByTagName() 方法，获取当前节点的指定标签名孩子节点

appendChild(oChildNode) 方法，可以添加一个子节点，oChildNode 是要添加的孩子节点

属性：

childNodes

属性，获取当前节点的所有子节点

firstChild

属性，获取当前节点的第一个子节点

lastChild

属性，获取当前节点的最后一个子节点

parentNode

属性，获取当前节点的父节点

nextSibling

属性，获取当前节点的下一个节点

previousSibling

属性，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

属性，表示获取/设置起始标签和结束标签中的内容

innerText

属性，表示获取/设置起始标签和结束标签中的文本

.DOM 查询练习