目录

作业讲评

```
<script>
    // 2、(Date)计算自己出生到现在活了多少天？
    // 思路:是使用现在时间戳-出生时间戳  最后时间戳单位是毫秒数,需要转成天数
    var now = new Date();
    var birthday = new Date("1999-9-9");
    var millseconds = now.getTime() - birthday.getTime();
    // 毫秒数需要转成天数
    var days = Math.ceil( millseconds/1000/60/60/24 );
    console.log("出生到现在活了" + days +"天" );

</script>
```

```
<script>
    // (Array)把下面数组的首尾两个元素互换
    // var arr = ["鹿晗","王俊凯","蔡徐坤","彭于晏","周杰伦","刘德华","赵本山"];
    var arr = ["鹿晗","王俊凯","蔡徐坤","彭于晏","周杰伦","刘德华","赵本山"];

    // 从头部取出第一个元素
    var first = arr.shift();
    // 从尾部取出最后一个元素
    var last = arr.pop();
    // console.log( first ,last );
    // console.log( arr );

    // 在头部添加刚取出来的最后一个元素
    arr.unshift( last );

    // 在尾部添加刚取出来的第一个元素
    arr.push( first );

    console.log( arr );
</script>
```

```
<script>
    /* 进阶作业—随机选学员
    从以下学员名单中随机选出4个学员：
    var arr = ["鹿晗","王俊凯","蔡徐坤","彭于晏","周杰伦","刘德华","赵本山"];
    注意：不要有重复的学员
    效果图：  每次刷新结果都会不一样 */

    var arr = ["鹿晗","王俊凯","蔡徐坤","彭于晏","周杰伦","刘德华","赵本山"];
    // 思路：  每次随机取出一个数组元素,再把这个数组元素从数组中删除,下次再取,就不会取到重复的学员了
    // console.log( arr );// 输出原数组

    // 定义一个随机函数
    function getRandom(min,max){
        return Math.floor( Math.random()*(max-min+1) + min );
    }

    // 定义一个新的空数组
    var newArr = [];
    for(var i = 1;i<= 4; i++){
        // 先取一个,需要使用到随机数,随机数的范围就是数组的下标    0 到 数组长度-1
        var index = getRandom(0, arr.length-1 );
        // console.log( arr[index] );// 输出取到的元素

        // 把取出来的数组元素放进新数组中
        newArr.push( arr[index] );

        // 取到的元素,还需要从原数组中移除    splice(start,length)
        arr.splice( index, 1);
        // console.log( arr );// 输出移除元素以后的数组
    }
    console.log( newArr );
</script>
```

```
var arr = ["鹿晗","王俊凯","蔡徐坤","彭于晏","周杰伦","刘德华","赵本山"];
// 使用sort排序函数  sort的回调函数需要返回三种值  正数 负数 0
arr.sort(function(){
    // Math.random()的取值范围是 0~1之间的所有小数,0可以取到
    // 如果Math.random()得到的数大于0.5      -   0.5    =>  正数
    // 如果Math.random()得到的数小于0.5      -   0.5    =>  负数
    // 如果Math.random()得到的数等于0.5      -   0.5    =>  0
    return Math.random()-0.5;
});
console.log( arr );
```

```
<script>
        /* 3、进阶题 (Date) 制作一个函数,getDayNum( 年,月,日 ),可以返回指定日期是当前年的第几天
        说明:
        闰年一年有366天    平年一年有365天   如果闰年,2月份有29天;如果平年,2月份只有28天

        判断闰年的条件是
        1.能被4整除并且不能被100整除的年份
        2.能被400整除的年份
        比如:   getDayNum( 2019,1,2)    返回值为:2 */


        // 先写一个函数判断是否为闰年
        function isLeapYear(year){
            if( (year%4==0 && year%100 != 0) || year%400 ==0 ){
                return true;
            }


            return false;
        }

        function getDayNum( year,month,day ){
            // 如果是1月,直接返回day的天数即可
            if(month == 1){
                return day;
            }


            // 如果是其他月份呢,就是大于1月的月份, 2 3 4....12月份
            // 如果闰年,2月份有29天;如果平年,2月份只有28天
            // 先假设是平年
            var arrDays = [31,28,31,30,31,30,31,31,30,31,30];


            // 如果现在的月份是2月,那么就需要加上整个1月的天数
            // 31+5 = 36


            // 如果现在的月份是2月,那么就需要加上整个1月的天数,再加上整个2月的天数
            // 31+28+5=64


            // 把大于2月份之前的月份整月天数累加起来
            var days = 0;
            for(var i = 0; i < month-1;i++){
                days = days + arrDays[i];
            }
            days = days + day;


            // 如果传进来的月份大于2月,因为大于2月才需要加上整个2月的天数
            if( month >2 ){
                // 判断年份是否为闰年
                if( isLeapYear(year) ){
                    days++;
                }
            }

            return days;
        }
        console.log( getDayNum(2020,2,5) );
        console.log( getDayNum(2020,3,5) );
    </script>
```

学习目标

- 能够通过ID来获取元素
- 能够通过标签名来获取元素
- 能够通过class来获取元素
- 能够通过选择器来获取元素
- 能够获取body和html元素
- 能够给元素注册事件(绑定事件)
- 能够修改元素的内容
- 能够区分innerText和innerHTML的区别
- 能够修改像div这类普通元素的属性
- 能够修改表单元素的属性
- 能够修改元素的样式属性

学习目标

- 能够通过ID来获取元素
- 能够通过标签名来获取元素
- 能够通过class来获取元素
- 能够通过选择器来获取元素
- 能够获取body和html元素

类似CSS选择器,选择标签用的

- 能够给元素注册事件(绑定事件)

鼠标单击,鼠标移上等事件

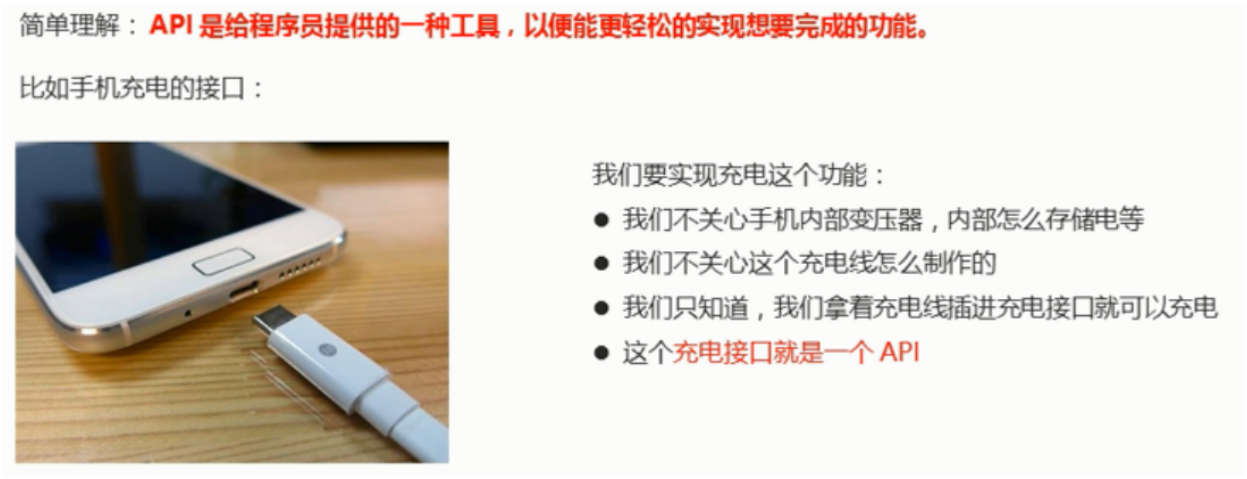
- 能够修改元素的内容
- 能够区分innerText和innerHTML的区别
- 能够修改像div这类普通元素的属性
- 能够修改表单元素的属性
- 能够修改元素的样式属性

使用js可以改变HTML的属性,改变HTML标签的内容,改变CSS样式

Web API介绍

API的概念

API（Application Programming Interface，应用程序编程接口）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，无需理解其内部工作机制细节，只需直接调用使用即可。



举例解释什么是API。

例如，

C语言中有一个函数 fopen()可以打开硬盘上的文件，这个函数对于我们来说，就是一个C语言提供的打开文件的工具。

javascript中有一个函数alert()可以在页面弹一个提示框，这个函数就是js提供的一个弹框工具。

这些工具（函数）由编程语言提供，内部的实现已经封装好了，我们只要学会灵活的使用这些工具即可。

Web API的概念

Web API 是浏览器提供的一套操作浏览器功能和页面元素的 API (BOM 和 DOM)。

现阶段我们主要针对于浏览器讲解常用的 API , 主要针对浏览器做交互效果。比如我们想要浏览器弹出一个警示框， 直接使用 alert('弹出')

MDN 详细 API : https://developer.mozilla.org/zh-CN/docs/Web/API

因为 Web API 很多，所以我们将这个阶段称为 Web APIs。

此处的 Web API 特指浏览器提供的一系列API(很多函数或对象方法)，即操作网页的一系列工具。例如:操作html标签、操作页面地址的方法。

API 和 Web API 总结

1. API 是为我们程序员提供的一个接口，帮助我们实现某种功能，我们会使用就可以了，不必纠结内部如何实现
2. Web API 主要是针对于浏览器提供的接口，主要针对于浏览器做交互效果。
3. Web API 一般都有输入和输出（函数的传参和返回值），Web API 很多都是方法（函数）
4. 学习 Web API 可以结合前面学习内置对象方法的思路学习

DOM 介绍

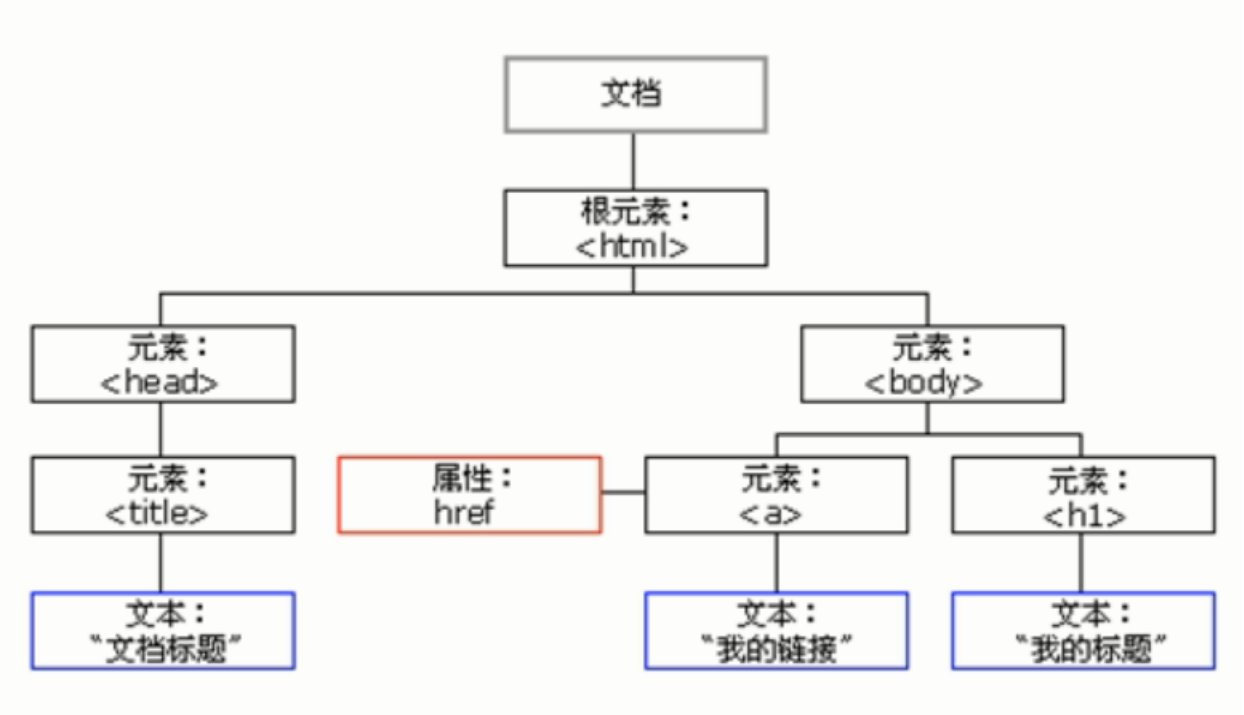
什么是DOM

文档对象模型（Document Object Model，简称DOM），是 W3C 组织推荐的处理可扩展标记语言（html或者xhtml）的标准编程接口。

W3C 已经定义了一系列的 DOM 接口，通过这些 DOM 接口可以改变网页的内容、结构和样式。

> DOM是W3C组织制定的一套处理 html和xml文档的规范，所有的浏览器都遵循了这套标准。

DOM树



DOM树 又称为文档树模型，把文档映射成树形结构，通过节点对象对其处理，处理的结果可以加入到当前的页面。

- 文档:一个页面就是一个文档，DOM中使用document表示
- 节点:网页中的所有内容，在文档树中都是节点（标签、属性、文本、注释等），使用node表示
- 标签节点:网页中的所有标签，通常称为元素节点，又简称为“元素”，使用element表示

DOM把以上内容都看做是对象

获取元素 重点

为什么要获取页面元素?

例如:我们想要操作页面上的某部分(显示/隐藏，动画)，需要先获取到该部分对应的元素，再对其进行操作。

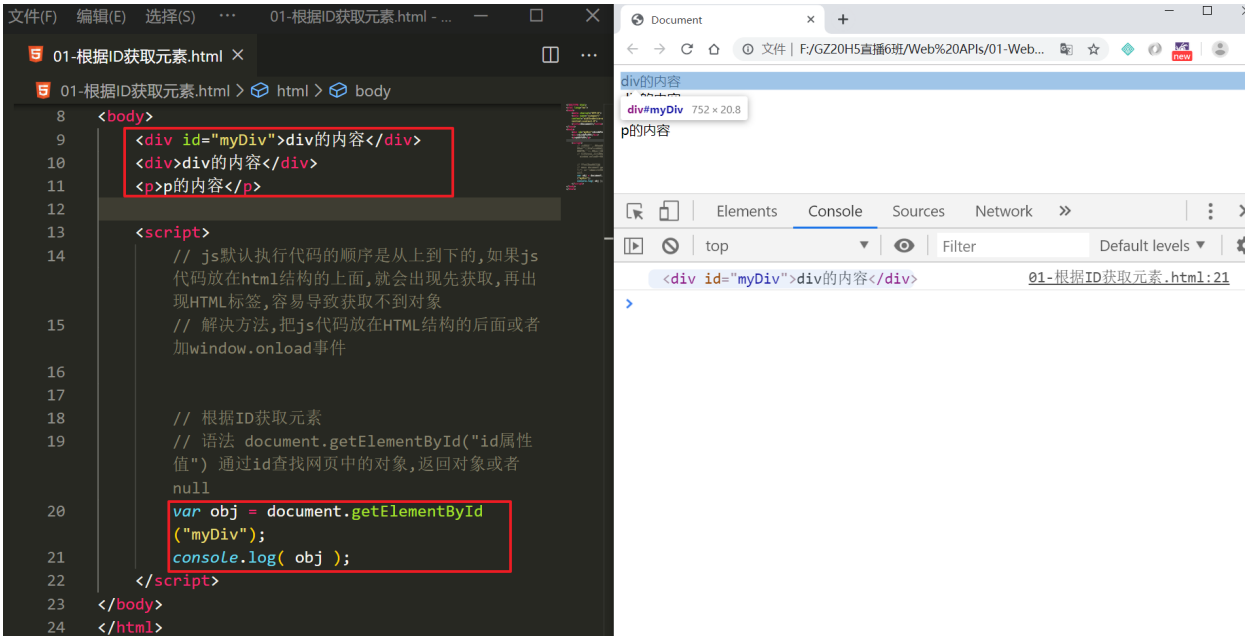
根据ID获取元素

```
语法:document.getElementById(id)
作用:根据ID获取元素对象
参数:id值，区分大小写的字符串
返回值:元素对象 或 null
```

补充一个方法,之前我们学过console.log可以在控制台中打印内容

console.dir(变量名) 可以用于打印内容或者对象; 如果打印的是对象的话,可以显示一个对象所有的属性和方法

举例



根据标签名获取元素

语法:

document.getElementsByTagName('标签名')

或者

父元素对象.getElementsByTagName('标签名') 可以获取到这个父元素对象里面的某些标签

作用:根据标签名获取元素对象

参数:标签名

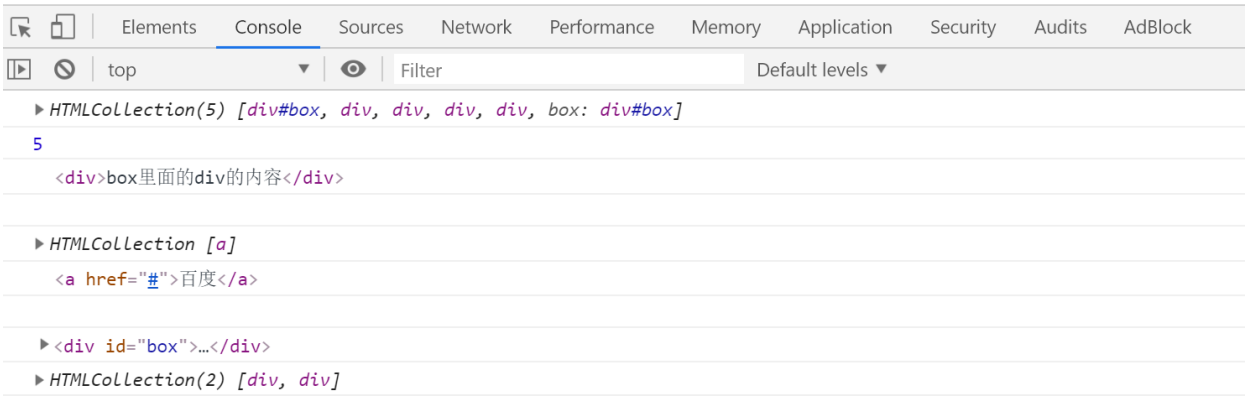
返回值:元素对象集合（伪数组，数组元素是元素对象）

代码:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <div id="box">
        <div>box里面的div的内容</div>
        <div>box里面的div的内容</div>
    </div>


    <div>div的内容</div>
    <div>div的内容</div>


    <a href="#">百度</a>


    <script>
        // 根据标签名获取元素  有点类似CSS标签选择器
        /* 语法:
        document.getElementsByTagName('标签名')
        或者
        父元素对象.getElementsByTagName('标签名')   可以获取到这个父元素对象里面的某些标签

        作用:根据标签名获取元素对象
        参数:标签名
        返回值:元素对象集合（伪数组，数组元素是元素对象） */


        var objDivs = document.getElementsByTagName("div");
        console.log( objDivs );
        console.log( objDivs.length );
        console.log( objDivs[2] );
        console.log("");


        // 注意:document.getElementsByTagNament即使获取出来只有一个元素,也会放在伪数组(集合)中
        var objA = document.getElementsByTagName("a");
        console.log( objA );
        console.log( objA[0] );
        console.log("");


        // 父元素对象.getElementsByTagName('标签名')   可以获取到这个父元素对象里面的某些标签   有点类似CSS后代选择器


        // 先通过id获取p段落标签
        var objP = document.getElementById("box");
        console.log( objP );


        var Pdivs = objP.getElementsByTagName("div");
        console.log( Pdivs );
    </script>
</body>
</html>
```

效果图:

box里面的div的内容
box里面的div的内容
div的内容
div的内容
百度



注意:

- 因为得到的是一个对象的集合,所以我们想要操作里面的元素就需要遍历
- getElementsByTagName()获取到是动态集合，即:当页面增加了标签，这个集合中也就增加了元素。

H5新增获取元素方式

方法名	功能
document.getElementsByClassName("类名")	通过标签的class属性值来获取元素 它返回是一个数组 如果要访问其中的某一个标签对象 要使用下标来进行访问！ 就算这个数组里面的元素只有一个 那么也是要使用数组下标的方式来进行访问！类似css中的类选择器
document.querySelector("css选择器")	**html5新增,**该方法返回满足CSS选择器的单个元素。如果找到多个满足的元素，则返回第一个满足条件的元素。
document.querySelectorAll("css选择器")	**html5新增,**该方法返回满足CSS选择器的所有元素，结果是一个数组集合 如果要访问其中的某一个标签对象 要使用下标来进行访问！ 就算这个数组里面的元素只有一个 那么也是要使用数组下标的方式来进行访问！

注意: querySelector和querySelectorAll里面的选择器需要加符号, 比如:
document.querySelector("#nav") document.querySelectorAll(".box")

举例

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .redFont{
            color:red;
        }
        #myDiv{
            background: skyblue;
        }
    </style>
</head>
<body>
    <div class="redFont">one</div>
    <div class="redFont">one</div>
    <div>div的内容</div>
    <div id="myDiv">div的内容</div>
    <div>div的内容</div>


    <script>
        // document.getElementsByClassName("类名") 通过标签的class属性值来获取元素 它返回是一个伪数组 如果要访问其中的某一个标签对象 要使用下标来进行访问！ 就算这个数组里面的元素只有一个 那么也是要使用数组下标的方式来进行访问！类似css中的类选择器


        // 通过类名获取标签
        var redFonts = document.getElementsByClassName("redFont");
        console.log( redFonts );
        console.log("");


        // document.querySelector("css选择器") html5新增,该方法返回满足CSS选择器的单个元素。如果找到多个满足的元素，则返回第一个满足条件的元素。


        // 通过CSS选择器获取"第一个"满足条件的元素
        // 注意:document.querySelector()里面的字符串需要加上css选择器符号,比如# .
        var myDiv = document.querySelector("#myDiv");
        console.log( myDiv );
        console.log("");


        var divs = document.querySelector(".redFont");
        console.log( divs );
        console.log("");


        // document.querySelectorAll("css选择器") html5新增,该方法返回满足CSS选择器的所有元素，结果是一个"数组集合" 如果要访问其中的某一个标签对象 要使用下标来进行访问！ 就算这个数组里面的元素只有一个 那么也是要使用数组下标的方式来进行访问！
        var objDivs = document.querySelectorAll(".redFont");
        console.log( objDivs );
    </script>


</body>
</html>
```

04-能否使用伪类选择器.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <ul>
        <li>1111</li>
        <li>2222</li>
        <li>3333</li>
        <li>4444</li>
        <li>5555</li>
    </ul>

    <script>
        var li = document.querySelectorAll("ul li:nth-child(even)");
        console.log( li );
    </script>
</body>
</html>
```

获取特殊元素(body,html)

```
获取body元素
document.body
```

```
获取html元素
document.documentElement
```

举例

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        // 获取特殊元素  html元素  body元素
        // document.body获取body元素
        console.log( document.body );

        // document.documentElement获取html标签
        console.log( document.documentElement );
    </script>
</body>
</html>
```

事件基础

事件概述

JavaScript 使我们有能力创建动态页面，而事件是可以被 JavaScript 侦测到的行为。

简单理解: **触发--- 响应机制。**

网页中的每个元素都可以产生某些可以触发 JavaScript 的事件，例如，我们可以在用户点击某按钮时产生一个 事件，然后去执行某些操作。

事件三要素 重点

- 事件源（谁）:触发事件的元素
- 事件类型（做什么）: 例如 onclick 点击事件
- 事件处理程序（怎么做）:事件触发后要执行的代码(函数形式)，事件处理函数

事件的绑定方式 重点

事件的绑定方式有二种:**行内绑定与动态绑定**

行内绑定事件

说明:所谓的行内绑定事件 是将事件写在HTML标签里面

格式:

```
<标签名 事件名="函数名()" />
```

动态绑定事件 重点

说明:是将事件名写在JS代码里面

动态事件绑定的步骤如下:

```
第一步:获取事件源(获取对象)
第二步:注册事件(绑定事件)
第三步:添加事件处理程序(一般是一个匿名函数)
```

格式:

```
通过JS获取标签对象.事件名 = 事件的处理程序     事件的处理程序一般是一个匿名函数
```

举例:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <button onclick="fn()">唐伯虎</button>
    <button onclick="alert('获取验证码成功')">获取验证码</button>


    <div>div内容</div>


    <script>
        // 事件的绑定方式,其实就是给网页中的元素绑定事件
        // 事件的绑定方式的方法分为两种


        // 一种是行内绑定(比较少用)
        // 说明:所谓的行内绑定事件 是将"事件"写在"HTML标签"里面
        // 格式:
        // <标签名 事件名="函数名()" />
```

```
        // 或者
        // <标签名 事件名="js代码" />


        function fn(){
            alert("点秋香");
        }



        // 一种是动态绑定(常用)
        // 动态绑定事件    重点
        // 说明:是将"事件名写在JS代码"里面

        // 动态事件绑定的步骤如下:
        // 第一步:获取事件源(获取对象)
        // 第二步:注册事件(绑定事件)
        // 第三步:添加事件处理程序(一般是一个匿名函数)


        // 1.获取对象
        var div = document.querySelector("div");
        // 建议初学者,获取对象以后,输出对象看看是否获取正确
        // console.log( div );


        // 格式:
        // 通过JS获取标签对象.事件名 = 事件的处理程序    事件的处理程序一般是一个"匿名函数"


        // 2.注册事件(绑定事件)
        div.onclick = function(){
            // 具体事件处理程序


            // 改变div的背景颜色
            div.style.backgroundColor = "pink";
        }
    </script>
</body>
</html>
```

常见的鼠标事件

鼠标事件	触发条件
onclick	鼠标点击左键触发
onmouseover	鼠标经过触发
onmouseout	鼠标离开触发
onfocus	获得鼠标焦点触发
onblur	失去鼠标焦点触发
onmousemove	鼠标移动触发

onmouseup	鼠标弹起触发
onmousedown	鼠标按下触发

获取元素练习

第一题:

```
HTML页面，有一个ul，ul中有3个li标签:
<ul>
  <li>苹果</li>
  <li>香蕉</li>
  <li>葡萄</li>
</ul>

请分别获取第一个li和最后一个li，并打印到控制台!
```

第二题:

```
创建一个ul和多个li
使用不同的方式获取ul中所有li标签

var ul = document.getElementsByTagName("ul")[0];
var lis = ul.getElementsByTagName("li");

var lis = document.querySelectorAll("ul li")

还可以给ul设置id或者给li设置类名
```

事件练习

问答题

```
1.事件三要素是哪些?
事件源,事件类型, 事件处理程序

2.动态事件绑定的步骤是什么?
第一步:获取对象
第二步:给对象注册事件
第三步:写具体事件驱动函数,一般是一个匿名函数
```

代码题:

```
页面中有一个按钮，当鼠标点击按钮的时候，弹出"你好"警示框。

<button id="btn">按钮</button>

document.getElementById("btn").onclick=function(){
 alert("你好");
}
```

操作元素

JavaScript的 DOM 操作可以改变网页内容、结构和样式，我们可以利用 DOM 操作元素来改变元素里面的内容、属性等。（注意:这些操作都是通过元素对象的属性实现的）

改变元素内容（获取或设置）

属性名	功能
双标签对象.innerHTML	获取或者设置**双边标签**里面的所有内容
双标签对象.innerText	获取或者设置**双边标签**里面的文本内容

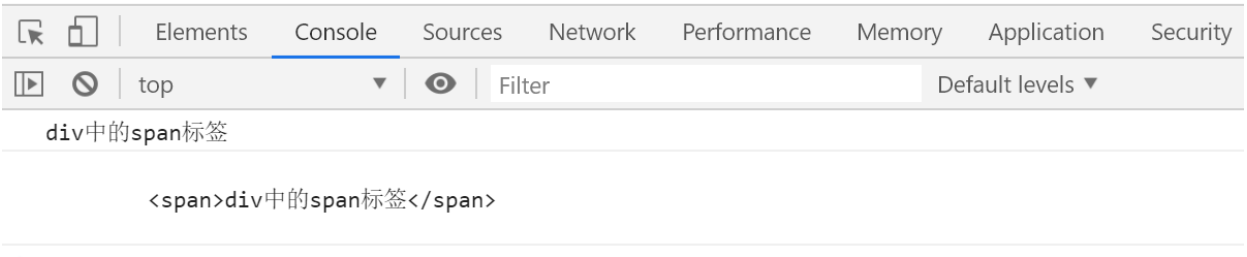
innerHTML跟innerText 这两个属性是可读写的 可以获取元素里面的内容, 也可以设置元素里面的内容

innerText和innerHTML的区别

- 获取内容时的区别:innerText会去除空格和换行，而innerHTML会保留空格和换行
- 设置内容时的区别:innerText不会解析html标签，而innerHTML会解析

举例:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div id="myDiv1">
        <span>div中的span标签</span>
    </div>

    <button id="btn1">innerText获取内容</button>
    <button id="btn2">innerHTML获取内容</button>
    <button id="btn3">innerText设置内容</button>
    <button id="btn4">innerHTML设置内容</button>

    <div id="myDiv2"></div>


    <script>
        // 操作元素
        // JavaScript的 DOM 操作可以改变"网页内容"、"结构"和"样式"，我们可以利用 DOM 操作元素来改变元素里面的内容、属性等。（注意:这些操作都是通过元素对象的属性实现的）


        // 因为双标签才可以设置标签内容
        // 改变元素内容（获取或设置）
        // 双标签对象.innerHTML "获取"或者"设置"双边标签里面的"所有内容"
        // 双标签对象.innerText "获取"或者"设置"双边标签里面的"文本"内容


        // innerHTML跟innerText 这两个属性是可读写的  可以获取元素里面的内容，也可以设置元素里面的内容


        // innerText和innerHTML的区别
        // 获取内容时的区别:innerText会去除空格和换行，而innerHTML会保留空格和换行
        // 设置内容时的区别:innerText不会解析html标签，而innerHTML会解析



        // 绑定事件的三步骤
        // 第一步:获取对象
        var btn1 = document.getElementById("btn1");
        // 第二步:注册事件(绑定事件)
        btn1.onclick = function(){
            // 第三步:事件处理程序(具体做什么)
            // 获取#myDiv1中的文本内容
            var myDiv1 = document.getElementById("myDiv1");
            // 获取双标签对象中"文本内容"  语法:   对象.innerText
            console.log( myDiv1.innerText );
        }


        var btn2 = document.getElementById("btn2");
        btn2.onclick=function(){
            // 获取#myDiv1中的所有内容
            var myDiv1 = document.getElementById("myDiv1");
            // 获取双标签对象中"所有内容"  语法:   对象.innerHTML
            console.log( myDiv1.innerHTML );
        }


        var btn3 = document.getElementById("btn3");
        btn3.onclick=function(){
            // 我们给myDiv2的标签设置内容
            var myDiv2 = document.getElementById("myDiv2");


            // 使用innerText设置内容  语法:  对象.innerText = "内容";
            myDiv2.innerText = "<h1>我是标题标签</h1>";
        }


        var btn4 = document.getElementById("btn4");
        btn4.onclick=function(){
            // 我们给myDiv2的标签设置内容
            var myDiv2 = document.getElementById("myDiv2");


            // 使用innerHTML设置内容  语法:  对象.innerHTML = "内容";
            myDiv2.innerHTML = "<h1>我是标题标签</h1>";
        }


        // innerText和innerHTML的区别
        // 获取内容时的区别:innerText会去除空格和换行，而innerHTML会保留空格和换行
        // 设置内容时的区别:innerText不会解析html标签，而innerHTML会解析


        // 总结:innerText跟innerHTML两个用法很相近,但是平时我们使用innerHTML会更多
    </script>
</body>
</html>
```



常用元素的属性操作(获取和设置标签设置属性)

属性名	功能
innerText	获取或者设置元素中文本内容
innerHTML	获取或者设置元素中所有内容
src	获取或者设置元素中src属性值
href	获取或者设置元素中href属性值
id	获取或者设置元素中id属性值
alt	获取或者设置元素中alt属性值
title	获取或者设置元素中title属性值

获取属性的值

```
元素对象.属性名
```

设置属性的值

```
元素对象.属性名 = 值
```

举例:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <a href="https://www.baidu.com">百度</a>


    <button>获取标签内容和href属性值</button>
    <button>设置标签内容和href属性值</button>
    <button>给a标签添加新的自带属性</button>
    <button>获取a标签没有设置的自带属性</button>


    <script>
        // 我们的DOM可以操作网页的内容,网页的结构,网页的样式,还可以操作标签的自带属性,什么时候自带属性,比如img标签自带src属性,a标签自带href属性...


        // 获取按钮对象
        var btns = document.querySelectorAll("button");
        // console.log( btns );

        // 获取a标签对象
        var a = document.querySelector("a");


        // 给按钮绑定单击事件
        btns[0].onclick=function(){
            // 获取a标签里面的内容  双标签对象.innerHTML或者双标签对象.innerText
            console.log( a.innerHTML );


            // 获取a标签里面href属性值 语法 对象.自带属性名
            console.log( a.href );
        }


        btns[1].onclick=function(){
            // 设置a标签里面的内容  双标签对象.innerHTML = "内容" 或者双标签对象.innerText = "内容";
            a.innerHTML = "腾讯";


            // 设置a标签里面href属性值 语法 对象.自带属性名 = 新属性值
            a.href = "https://www.qq.com";
        }


        btns[2].onclick=function(){
            // 给a标签添加一个target属性
            a.target = "_blank";
        }


        btns[3].onclick=function(){
            // 获取a标签的title属性
            console.log( a.title );
        }
    </script>
</body>
</html>
```

练习: 小居佩奇和多啦A梦切换

小居佩奇 多啦A梦

小居佩奇 多啦A梦

练习:分时问候

亲，晚上好，好好写代码

要求:

```
根据不同时间，页面显示不同图片，同时显示不同的问候语。
如果上午时间打开页面，显示问候语上午好，显示上午的图片。
如果下午时间打开页面，显示问候语下午好，显示下午的图片。
如果晚上时间打开页面，显示问候语晚上好，显示晚上的图片。
```

分析:

```
1. 根据系统不同事件来判断,所以需要用来日期内置对象
2. 利用多分支语句设置不同的图片
3. 需要一个图片,并且根据时间修改图片,那么修改图片就要用到图片的src属性
4. 需要一个div元素,显示不同问候语,修改元素内容即可
```

表单元素的属性操作

利用DOM可以操作一下表单元素属性

```
常见表单元素属性: type,value,checked,selected,disabled
```

获取属性的值

```
元素对象.属性名
```

设置属性的值

```
元素对象.属性名 = 值
```

注意:表单元素中有一些属性如:`disabled`、`checked`、`selected`，元素对象的这些属性的值是布尔型。

举例: 点击按钮弹窗提示发送验证码成功,并把按钮状态改为禁用,按钮内容显示为60s重新获取验证码

```
<input type="text" placeholder="请输入手机验证码">
<button>获取验证码</button>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <p>
        <input type="radio" checked="checked"/>男
    </p>


    <input type="text" placeholder="请输入手机验证码">
    <button>获取验证码</button>


    <script>
        // 利用DOM可以操作一下表单元素属性
        // 常见表单元素属性: type,value,checked,selected,disabled


        // 注意:表单元素中有一些属性如:disabled、checked、selected，元素对象的这些属性的值是"布尔型",所以 在给表单元素设置 disabled、checked、selected这几个属性的时候,也需要使用"布尔型"


        // var input = document.querySelector("input");
        // console.log( input.checked );// true


        // 点击按钮弹窗提示发送验证码成功,并把按钮状态改为禁用,按钮内容显示为60s重新获取验证码
        var btn = document.querySelector("button");
        btn.onclick=function(){
            alert("发送验证码成功");
            // 设置按钮的禁用状态为true
            btn.disabled = true;

            // 改变按钮的内容
            btn.innerHTML = "60s重新获取验证码";
        }
    </script>
</body>
</html>
```

案例:仿京东显示密码

京东登录

用户名/邮箱/已验证手机

••••••••• | 忘记密码

登录

短信验证码登录　　手机快速注册

京东登录

点击以后

用户名/邮箱/已验证手机

123456789 | 忘记密码

登录

短信验证码登录　　手机快速注册

要求:

```
1 点击按钮将密码框切换为文本框,并可以查看明文密码
```

分析:

```
1 1.点击闭眼图片,可以把图片变成眼睛图片,同时密码框类型改为文本框,这样就可以看到里面的密码了
2 2.再次点击眼睛图片,可以把图片变成闭眼图片,同时文本框类型改回密码框,这样密码又看不见了
3 3.因为一个图片要实现两种状态的切换,所以需要先判断当前密码框的type是text还是password
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        input{
            border:none;
            outline: none;
        }
        a{
            text-decoration: none;
            color:#999999;
        }
        a:hover{
            color:#c81623;
        }


        body{
            font-size:12px;
        }
        .fl{
            float: left;
        }
        .fr{
            float: right;
        }
```

```
        img{
            vertical-align: bottom;
        }
        .box{
            width: 300px;
            margin:100px auto;
            position: relative;
        }
        .box .title{
            text-align: center;
            font-weight: bold;
            margin-bottom: 20px;
            font-size: 18px;
        }
        .box input{
            border-bottom: 1px solid #cccccc;
            width: 100%;
            height: 40px;
        }
        .box input[type="submit"]{
            border-bottom: none;
            background: #e1251b;
            border-radius: 20px;
            color:white;
            text-align: center;
            line-height: 40px;
            cursor: pointer;
            margin-bottom: 20px;
            margin-top: 20px;
        }
        .box .forgetPassword{
            position: absolute;
            right: 0;
            bottom: 88px;
            color:#757575;
        }
        .box .forgetPassword img{
            width: 24px;
            vertical-align: middle;
            margin-right: 5px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <div class="box">
        <div class="title">京东登录</div>
        <input type="text" placeholder="用户名/邮箱/已验证手机"/>
        <input type="password" class="password"/>

        <div class="forgetPassword">
            <img src="images/close.png"/>|  <a href="#">忘记密码</a>
        </div>

        <input type="submit" value="登录">
        <a href="#" class="fl">短信验证码登录</a>
        <a href="#" class="fr">手机快速注册</a>
    </div>
    <script>
        // 获取所需对象
        // 眼睛图片对象
        var img = document.querySelector("img");


        // 密码框对象
        var password = document.querySelector(".password");


        // console.log( img );
        // console.log( password );


        // 给眼睛图片对象绑定事件
        img.onclick = function(){
            if(password.type == "password"){// 如果当前的密码框的type是password,怎么获取当前密码框的type属性值
                // 改变图片
                img.src = "images/open.png";
```

```
                // 改变密码框的type属性值
                password.type = "text";
            }else{
                // 改变图片
                img.src = "images/close.png";
                // 改变密码框的type属性值
                password.type = "password";
            }
        }

    </script>
</body>
</html>
```

样式属性操作

我们可以通过 JS 修改元素的大小、颜色、位置等样式。

常用方式

样式属性	功能
元素.style	行内样式操作
元素.className	类名样式操作

方式1:通过操作style属性

```
元素对象的style属性也是一个对象!
元素对象.style.样式属性 = 值;
```

注意:

- JS修改style样式操作,操作的是行内样式,CSS优先级比较高
- JS里面的样式采取驼峰法命名, 如果CSS属性中有中划线要去掉 并且将中划线后面的单词首字母大写 驼峰法 比如:background-color => backgroundColor

定义和用法

background-color 属性设置元素的背景颜色。

元素背景的范围

background-color 属性为元素设置一种纯色。这种颜色会填充元素的内容、内边距和边框区域，扩展到元素边框的外边界（但不包括外边距）。如果边框有透明部分（如虚线边框），会透过这些透明部分显示出背景色。

transparent 值

尽管在大多数情况下，没有必要使用 transparent。不过如果您不希望某元素拥有背景色，同时又不希望用户对浏览器的颜色设置影响到您的设计，那么设置 transparent 值还是有必要的。

默认值：	transparent
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.backgroundColor="#00FF00"

举例:点击按钮以后,改变div的宽度高度和背景颜色

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div>div的内容</div>
    <button>设置style属性值</button>
    <button>获取style属性值</button>


    <script>
        // 样式属性操作
        // 我们可以通过 JS 修改元素的大小、颜色、位置等样式。

        // 常用方式
        // 样式属性 功能
        // 元素.style 其实通过js操作标签的行内样式
        // 元素.className 类名样式操作

        // 方式1:通过操作style属性
        // 元素对象的style属性也是一个对象!
        // 元素对象.style.样式属性 = 值;


        // 要求点击按钮以后,改变div的宽度高度和背景颜色
        // 获取所需对象
        var btns = document.querySelectorAll("button");
        var div = document.querySelector("div");

        btns[0].onclick=function(){
            // 通过操作标签的style属性,从而改变标签的CSS样式
            div.style.width = "100px";
            div.style.height="200px";
            div.style.backgroundColor = "pink";
            div.style.fontSize = "30px";
        }


        /* 注意:
        JS修改style样式操作,操作的是行内样式,CSS优先级比较高
        JS里面的样式采取驼峰法命名, 如果CSS属性中有中划线要去掉 并且将中划线后面的单词首字母大写  驼峰法     比如:background-color   =>   backgroundColor  */


        btns[1].onclick=function(){
            // 如果使用的是div.style得到的是所有的css属性
            console.log( div.style );
            // 可以使用div.style.css属性名得到具体某个属性值
            console.log( div.style.backgroundColor );
        }
    </script>
</body>
</html>
```

案例:淘宝点击关闭二维码

要求:

```
当鼠标点击二维码关闭按钮,则关闭整个二维码(隐藏整个二维码)
```

分析:

```
1.利用样式的显示和隐藏完成,    隐藏元素display:none    显示元素display:block
2.点击按钮,让二维码盒子隐藏即可
```

```
<!DOCTYPE html>
<html lang="en">


<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        body{
            background: #fafbfc;
        }
        .box {
            position: relative;
            width: 74px;
            height: 88px;
            border: 1px solid #ccc;
            margin: 100px auto;
            font-size: 12px;
            text-align: center;
            color: #f40;
        }

        .box img {
```

```
            width: 60px;
            margin-top: 5px;
        }

        .close-btn {
            position: absolute;
            top: -1px;
            left: -16px;
            width: 14px;
            height: 14px;
            border: 1px solid #ccc;
            line-height: 14px;
            cursor: pointer;
            font-style: normal;
        }
    </style>
</head>


<body>
    <div class="box">
        淘宝二维码
        <img src="images/tao.png" alt="">
        <i class="close-btn">x</i>
    </div>


    <script>
        // 获取所需对象
        var closeBtn = document.querySelector(".close-btn");
        var box = document.querySelector(".box");
        /* console.log( closeBtn);
        console.log( box); */


        // 给关闭按钮绑定单击事件
        closeBtn.onclick=function(){
            // 隐藏整个大盒子
            box.style.display = "none";
        }
    </script>
</body>


</html>
```

案例:循环精灵图背景

效果图:

图片素材:

要求:

```
可以使用for循环设置一组元素的精灵图背景
```

分析:

```
1. 首先精灵图图片排列是有规律的
2. 利用for循环  修改图片的背景定位属性 background-position
3. 让循环里面的i索引号*44就是每个图片的y坐标
```

```
<!DOCTYPE html>
<html lang="en">


<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        * {
            margin: 0;
            padding: 0;
        }

        li {
            list-style-type: none;
        }

        .box {
            width: 250px;
            margin: 100px auto;
        }

        .box li {
            float: left;
            width: 24px;
            height: 24px;
            margin: 15px;
            background: url(images/sprite.png) no-repeat;
        }
    </style>
</head>


<body>
    <div class="box">
        <ul>
            <li></li>
            <li></li>
```

```
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
        </ul>
    </div>


    <script>
        // 获取.box ul中所有的li标签
        var lis = document.querySelectorAll(".box ul li");
        // console.log( lis );
        // console.log( lis.length );


        for(var i=0;i<lis.length;i++){
            var posY = -44*i;
            lis[i].style.backgroundPosition = "0px "+posY+"px";
        }
    </script>
</body>


</html>
```

案例:显示隐藏文本框内容

要求:

```
1 当鼠标点击文本框获取焦点时,里面的默认文字隐藏,当文本框失去焦点时,里面的文字显示
```

分析:

```
1 1. 首页文本框需要绑定两个事件, 获取焦点 onfocus 失去焦点onblur
2 2. 如果获得焦点,判断文本框里面内容是否为默认文字,如果是默认文字,则清空文本框内容
3 3. 如果失去焦点,判断文本框里面内容是否为空,如果为空,则文本框内容设置为默认文字
```

```html
<!DOCTYPE html>
<html lang="en">


<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        input{
            border:none;
            outline: none;
        }
        .box{
            width: 500px;
            margin:100px auto;
        }
        .box input{
            width: 430px;
            height: 30px;
            border:2px solid #e2231a;
            text-indent: 1em;
            float: left;
        }
        .box span{
            float: left;
            background: url(images/fangdajing.png) #e2231a no-repeat center center;
            width: 58px;
            height: 34px;
            cursor: pointer;
        }
        input {
            color: #999;
        }
    </style>
</head>


<body>
    <div class="box">
        <input type="text" value="手机"/>
        <span></span>
    </div>


    <script>
        // 获取input对象
        var input = document.querySelector("input");
        // 给input绑定获取焦点事件
        input.onfocus = function(){
            if(input.value == "手机"){
                // 清空input的内容
                input.value = "";
                // 改变input中内容的字体颜色
```

```
                input.style.color = "#333333";
            }

        }


        // 给input绑定失去焦点事件
        input.onblur = function(){
            // 判断内容不为空,就赋值默认值手机
            if(input.value == ""){
                input.value = "手机";
            }
            input.style.color = "#999999";
        }
    </script>
</body>


</html>
```

今日总结

xmind要做,今天案例比较多,最好把案例写一遍

今日作业

看作业文件夹