一、BOM
1. BOM 介绍
2. window对象常用方法
1）网页弹框
2）窗口的打开和关闭
3）定时器方法
window 对象常用属性
1）history
2）location
3）document
二、DOM节点操作
1）节点对象
2）常用节点分类
3）获取元素节点
4）操作元素内容
5）操作元素属性
6）操作元素样式
7）元素节点的层次属性
8）节点的创建，添加和删除
三、DOM 事件处理
1） 事件函数分类
2）事件绑定方式
3）事件函数使用

hone_work.html 随机生成一个四位验证码，让用户输入，判断用户输入的是否正确。

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <a href="04_history.html">页面0</a>
    <a href="01_window.html">页面二</a>
    <button onclick="console.log(history.length)">
        获取length
    </button>
    <button onclick="history.back()">返回</button>
    <button onclick="history.forward()">前进</button>


    <input type="text" id="uinput">
    <span id="code"></span>
    <button id="btn">验证</button>
    <script>
```

```
        //元素的id属性值作为变量直接操作,存储元素对象本身
        console.log(code);
        //生成四位验证码
        var str = "ABCDEFGHIabcdefghtyuiu0123456789";
        var show = "";
        for(var i = 1; i < 5; i++){
            //生成随机下标
            //Math.floor(n)
            var index = Math.floor(Math.random()*str.length);
            show += str[index];
        }
        //操作元素内容
        code.innerHTML = show;
        //动态绑定事件
        btn.onclick = function (){
            //获取输入框的值
            console.log(uinput.value);
            //验证
            if(uinput.value.toUpperCase() == show.toUpperCase()){
                alert("输入正确")
            }else{
                alert("输入有误");
            }
        };

        /*Math对象*/
        //1.向上取整:舍弃小数位,整数位+1
        console.log(Math.ceil(0.99));
        //2.向下取整:舍弃小数位,保留整数位
        console.log(Math.floor(0.99));
        //3.四舍五入取整
        console.log(Math.round(0.49));
        console.log(Math.round(0.5));
        //4.生成0~1之间的随机小数
        console.log(Math.random());
    </script>
</body>
</html>
```

一、BOM

1. BOM 介绍

BOM全称为“Browser Object Model”，浏览器对象模型。提供一系列操作浏览器的属性和方法。核心对象为window对象，不需要手动创建，跟随网页运行自动产生，直接使用，在使用时可以省略书写。

2. window对象常用方法

1）网页弹框

```
alert()     //警告框
prompt()    //带输入框的弹框
confirm()   //确认框
```

2）窗口的打开和关闭

```
window.open("URL")  //新建窗口访问指定的URL
window.close()      //关闭当前窗口
```

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <script>
        console.log(window);
        //window对象的方法
        //1.网页弹框
        /*
        window.alert("操作无效");//无返回值
        var r = window.confirm("是否确认");//返回布尔值
        console.log(r);
        //prompt可以接收两个参数("提示信息",输入框的默认值)
        var r1 = window.prompt("请输入",0);
        */
        /*
        全局变量和全局函数都是window对象的属性和方法
        */
        var a = 100;
        function a1(){
            console.log("a1被调用");
        }
        console.log(window);
        console.log(a,window.a);
        a1();
        window.a1();
        //2.窗口的打开和关闭
        /*给出确认框,当点击确定时打开目标文件,点击取消时关闭当前窗口*/
        /*
        var r = confirm("");
        if(r){
           window.open("http://www.baidu.com");
        }else{
            window.close();
        }
        */
```

```
    </script>
</head>
<body>
    <a href="04_history.html">页面0</a>
    <a href="00_work.html">页面一</a>
    <button onclick="console.log(history.length)">
        获取length
    </button>
<button onclick="history.back()">返回</button>
    <button onclick="history.forward()">前进</button>
</body>
</html>
```

3）定时器方法

1. 间歇调用(周期性定时器) 作用：每隔一段时间就执行一次代码 开启定时器:

```
var timerID = setInterval(function,interval);
/*
参数 :
 function : 需要执行的代码,可以传入函数名;或匿名函数
 interval : 时间间隔,默认以毫秒为单位 1s = 1000ms
返回值 : 返回定时器的ID,用于关闭定时器
*/
```

关闭定时器：

```
//关闭指定id对应的定时器
clearInterval(timerID);
```

2. 超时调用(一次性定时器) 作用：等待多久之后执行一次代码

```
//开启超时调用:
var timerId = setTimeout(function,timeout);
//关闭超时调用:
clearTimeout(timerId);
```

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <button id="start">开启</button>
```

```
    <button id="stop1">关闭</button>
    <h1 style="text-align:center;" id="show"></h1>
    <script>
        var timer;
        start.onclick = function (){
            //开启定时器
            timer = setInterval(function (){
                //打印日期时间(Date)
                var date = new Date();
                console.log(date);
            },1000);
            console.log("---------");
        };
        stop1.onclick = function (){
            //关闭间歇调用
            clearInterval(timer);
        };
        /*页面输出5秒倒计时*/
        var i = 5;
        show.innerHTML = i;
        var timerId = setInterval(function (){
            i--;
            if(i != 0 ){
                show.innerHTML = i;
            }else{
                show.innerHTML = "倒计时结束!";
                //停止定时器
                clearInterval(timerId);
            }
        },1000);

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

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <button id="start">开启</button>
    <button id="stop1">关闭</button>
    <script>
        start.onclick = function (){
            //开始超时调用
            timer = setTimeout(function (){
                console.log("超时调用");
```

```
            },1000);
        };
        stop1.onclick = function (){
            //关闭超时调用
            clearTimeout(timer);
        };

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

window 对象常用属性

window的大部分属性又是对象类型

1）history

作用：保存当前窗口所访问过的URL 属性 : length 表示当前窗口访问过的URL数量 方法 :

```
history.back() 对应浏览器窗口的后退按钮,访问前一个记录
history.forward() 对应前进按钮,访问记录中的下一个URL
history.go(n) 参数为number值,翻阅几条历史记录，正值表示前进,负值表示后退
```

```
<!--历史记录的长度变化
1.通过超链接在当前窗口访问其他url,会造成历史记录的增加
2.前进和后退不会造成历史记录的变化,只是指针的移动
3.历史记录的进出栈管理-->
```

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <script>
        console.log(history);
    </script>
</head>
<body>
    <a href="00_work.html">页面一</a>
    <a href="01_window.html">页面二</a>
    <button onclick="console.log(history.length)">
        获取length
    </button>
    <button onclick="history.back()">返回</button>
    <button onclick="history.forward()">前进</button>
```

```
    <button onclick="history.go(1)">go(1)</button>
    <button onclick="history.go(-1)">go(-1)</button>
    <button onclick="history.go(1)">go(2)</button>
    <button onclick="history.go(-1)">go(-2)</button>

   <!--历史记录的长度变化:
    1.通过超链接在当前窗口访问其他URL,会造成历史记录增加
    2.前进和后退不会造成历史记录的变化,只是指针的移动
    3.历史记录的进出栈管理:
        A->B->C
        从页面C后退至页面A,在A中通过超链接跳转至页面C
        A->C
    -->

</body>
</html>
```

2）location

作用：保存当前窗口的地址栏信息(URL) 属性 : href 设置或读取当前窗口的地址栏信息 方法 : reload(param) 重载页面(刷新) 参数为布尔值，默认为false，表示从缓存中加载，设置为true,强制从服务器根目录加载

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <!--设置或读取窗口的地址栏信息(URL)-->
    <button onclick="console.log(location.href);">
        获取href
    </button>
    <button onclick="location.href='http://www.baidu.com';">
        设置href</button>
    <!--reload(false)默认从缓存中重载页面,设置为true,表示
    从服务器根目录重载页面-->
    <button onclick="location.reload(true)">刷新</button>
</body>
</html>
```

3）document

提供操作文档HTML 文档的方法，,参见DOM

二、DOM节点操作

DOM全称为“Document Object Model”，文档对象模型，提供操作HTML文档的方法。（注：每个html文件在浏览器中都视为一篇文档,操作文档实际就是操作页面元素。）

1）节点对象

JS 会对html文档中的元素，属性，文本内容甚至注释进行封装，称为节点对象，提供相关的属性和方法。

2）常用节点分类

- 元素节点 (操作标签）
- 属性节点（操作标签属性）
- 文本节点（操作标签的文本内容）

3）获取元素节点

1. 根据标签名获取元素节点列表

```
var elems = document.getElementsByTagName("");
/*
参数 ： 标签名
返回值 ： 节点列表,需要从节点列表中获取具体的元素节点对象
*/
```

2. 根据class属性值获取元素节点列表

```
var elems = document.getElementsByClassName("");
/*
参数 ： 类名(class属性值)
返回值 ： 节点列表
*/
```

3. 根据id属性值取元素节点

```
var elem = document.getElementById("");
/*
参数 ： id属性值
返回值 ： 元素节点
*/
```

4. 根据name属性值取元素列表

```
var elems = document.getElementsByName("");
/*
参数 ： name属性值
返回 ： 节点列表
*/
```

4）操作元素内容

元素节点对象提供了以下属性来操作元素内容

```
innerHTML : 读取或设置元素文本内容,可识别标签语法
innerText : 设置元素文本内容,不能识别标签语法
value : 读取或设置表单控件的值
```

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <div id="d1"></div>
    <input id="d2" type="text">
    <button id="demo">取值</button>
    <script>
        //元素绑定事件,实现了动态的修改
        demo.onclick = function (){
            document.getElementById("d1").innerHTML = "<h1>"+d2.value+"</h1>";
        }
    </script>
</body>
</html>
```

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>

</head>
<body>
    <h1 class="c1" id="d1" style="">Fine,Thank U,and you</h1>
    <h1 class="c1">岁月静好,感恩有你,与你相随</h1>
    <input type="text" name="username" value="">
    <button id="btn">取值</button>
    <div id="show"></div>

    <script>
        //获取元素的操作只能等待标签解析完毕后执行
        //1.根据标签名获取元素列表
        var list1 = document.getElementsByTagName("h1");
        console.log(list1);
```

```
        console.log(list1[0],list1[0].innerHTML);
        //2.根据class属性值获取元素列表
        var list2 = document.getElementsByClassName("c1");
        console.log(list2);
        //3.根据id属性值获取元素
        var elem = document.getElementById("d1");
        console.log(elem);
        //标签属性都是元素节点对象的属性
        console.log(elem.id,elem.className);
        elem.style = "color:red;text-align:center;";
        //4.根据name属性值获取元素列表
        var list3 = document.getElementsByName("username");
        console.log(list3);

        //操作元素内容或值
        elem.innerHTML = "<a href>小泽最帅</a>";
        console.log(elem.innerHTML);
        elem.innerText = "<a href>小泽最帅</a>";
        console.log(elem.innerText);
        btn.onclick = function (){
            //获取表单控件的值
            console.log(list3[0].value);
            show.innerHTML = "<h1>"+list3[0].value+"</h1>";
        };
        /*
        创建输入框,按钮和div
        点击按钮时将输入框中的内容以一级标题的形式显示在div中
        */
    </script>
</body>
</html>
```

5）操作元素属性

1. 通过元素节点对象的方法操作标签属性

```
elem.getAttribute("attrname");//根据指定的属性名返回对应属性值
elem.setAttribute("attrname","value");//为元素添加属性,参数为属性名和属性值
elem.removeAttribute("attrname");//移除指定属性
```

2. 标签属性都是元素节点对象的属性,可以使用点语法访问，例如：

```
h1.id = "d1";        //set 方法
console.log(h1.id);  //get 方法
h1.id = null;       //remove 方法
```

注意：

- 属性值以字符串表示
- class属性需要更名为className,避免与关键字冲突,例如： h1.className = "c1 c2 c3";

6）操作元素样式

1. 为元素添加id，class属性，对应选择器样式
2. 操作元素的行内样式,访问元素节点的style属性，获取样式对象；样式对象中包含CSS属性，使用点语法操作。

```
p.style.color = "white";
p.style.width = "300px";
p.style.fontSize = "20px";
```

注意：

- 属性值以字符串形式给出,单位不能省略
- 如果css属性名包含连接符,使用JS访问时,一律去掉连接符,改为驼峰. font-size -> fontSize

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <script src="index.js"></script>
    <style>
        #main{
            color:red;
        }
        .c1{
            background:green;
        }
    </style>
</head>
<body>
    <h1 id="" style="">小泽真温柔</h1>
    <script>
        //获取元素节点
        var h1 = $("h1");
        //操作标签属性
        h1.setAttribute("id","d1");
        h1.setAttribute("class","c1");
        console.log(h1.getAttribute("class"));
        h1.removeAttribute("id");
        //点语法
        h1.id = "box";
        h1.className = "c1 c2";
        console.log(h1.id,h1.className);
        h1.id = "";
        h1.className = null;

        //操作元素样式
        //1.通过操作元素的id/class属性,对应选择器的样式
```

```
        h1.id = "main";
        h1.className = "c1";

        //2.操作行内样式
        console.log(h1.style);
        //直接赋值
        h1.style = "width:200px;height:200px;";
        //单独调整样式,h1.style返回样式表对象,由CSS的属性组成
        //出现连接符的属性名一律更名为驼峰标识
        h1.style.width = "300px";
        h1.style.textAlign = "center";
        h1.style.lineHeight = "200px";
    </script>
</body>
</html>
```

```
//获取元素
function $(tag,index){ //变量声明未赋值默认为undefined
    var elem;
    if(index){
        elem = document.getElementsByTagName(tag)[index];
    }else{ //index=0或index=undefined
        elem = document.getElementsByTagName(tag)[0];
    }
    return elem;
}
```

7）元素节点的层次属性

1. parentNode 获取父节点
2. childNodes 获取子节点数组,只获取直接子节点(包含文本节点和元素节点)
3. children 获取子节点数组,只获取直接子元素,不包含间接元素和文本节点
4. previousSibling 获取前一个兄弟节点(文本节点也可以是兄弟节点) previousElementSibling 获取前一个元素兄弟节点
5. nextSibling 获取后一个兄弟节点 nextElementSibling 获取下一个元素兄弟节点
6. attributes 获取属性节点的数组

8）节点的创建，添加和删除

1. 创建元素节点

```
var elem = document.createElement("标签名");//返回创建好的元素节点
```

2. 节点的添加 添加和移除操作都必须由父元素执行，方法如下:

- 在父元素的末尾添加子节点

```
parendNode.appendChild(node);
```

- 指定位置添加

```
parendNode.insertBefore(newNode,oldNode);//在oldNode之前添加子节点
```

3. 移除节点

```
parentNode.removeChild(node);//移除指定节点对象
```

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <script src="index.js"></script>
</head>
<body>
    <div>
        <p>参考</p>
        示例
        <h1 id="d1" class="c1">
            <span>h1->span</span>
        </h1>
        <p>参考</p>
    </div>
    <script>
        var h1 = $("h1");
        var div = $("div");
        //获取唯一的父节点
        console.log(h1.parentNode);
        //获取子节点数组(包含文本节点和元素节点)
        console.log(div.childNodes);
        //获取子元素数组(只包含直接子元素)
        console.log(div.children);
        //获取兄弟节点
        console.log(h1.previousSibling);
        console.log(h1.nextSibling);
        //获取元素兄弟节点
        console.log(h1.previousElementSibling);
        console.log(h1.nextElementSibling);
        //获取属性节点数组
        console.log(h1.attributes);

        //节点的创建,添加和移除
        //创建元素节点
        var h2 = document.createElement("h2");
        console.log(h2);
        h2.innerHTML = "动态添加";
        h2.id = "d2";
```

```
        //添加节点,由父节点操作
        document.body.appendChild(h2);//追加至父元素末尾
        /*节点与页面中元素一一对应,想在页面中出现几个元素,
        就需要创建几个节点*/
        var h3 = document.createElement("h3");
        //指定位置添加
        document.body.insertBefore(h3,h2);

        //移除节点
        //document.body.removeChild(div);

        /*
        1.实现网页轮播图
            方式一:控制图片的隐藏与显示
            方式二:切换图片路径
        2.参照效果图(添加元素练习)
        */
    </script>
</body>
</html>
```

三、DOM 事件处理

事件：指用户的行为或元素的状态。由指定元素监听相关的事件，并且绑定事件处理函数。 事件处理函数：元素监听事件，并在事件发生时自动执行的操作。

1） 事件函数分类

1. 鼠标事件

```
onclick     //单击
ondblclick  //双击
onmouseover //鼠标移入
onmouseout  //鼠标移出
onmousemove //鼠标移动
```

2. 键盘事件

```
onkeydown   //键盘按键被按下
onkeyup     //键盘按键被抬起
onkeypress  //字符按键被按下
```

3. 文档或元素加载完毕

```
onload      //元素或文档加载完毕
```

4. 表单控件状态监听

```
onfocus     //文本框获取焦点
onblur      //文本框失去焦点
oninput     //实时监听输入
onchange    //两次输入内容发生变化时触发,或元素状态改变时触发//按钮
onsubmit    //form元素监听,点击提交按钮后触发,通过返回值控制数据是否可以发送给服务器
```

2）事件绑定方式

1. 内联方式 将事件名称作为标签属性绑定到元素上 例：

```
<button onclick="alert()">点击</button>
```

2. 动态绑定 获取元素节点，动态添加事件 例：

```
btn.onclick = function (){

};
```

3）事件函数使用

1. onload 常用于等待文档加载完毕再进行下一步操作
2. 鼠标事件
3. 表单事件 onchange： 监听输入框前后内容是否发生变化;也可以监听按钮的选中状态 onsubmit： 表单元素负责监听,允许返回布尔值,表示数据是否可以发送;返回true,允许发送;返回false,不允许提交
4. 键盘事件
5. www.jquery.123.com

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        div{
            width:200px;
            height:200px;
            background:green;
        }
    </style>
</head>
<body>
    <div id="div"></div>
    <script>
        /*
        事件对象:保存与当前事件相关的所有信息,
        伴随事件发生自动创建,自动作为参数传递到事件处理函数中,
        我们只需要在事件处理函数中定义形参接收即可
```

```
        */

        //1.鼠标事件
        div.onclick = function (e){//event  evt e
            /*
            鼠标事件对象,主要保存鼠标的位置信息
            offsetX offsetY:获取鼠标在元素坐标系中的位置
            */
            console.log("单击",e,e.offsetX,e.offsetY);
        };
        div.ondblclick = function (){
           // console.log("双击");
        };
        div.onmouseover = function (){
            //console.log("鼠标移入");
        };
        div.onmouseout = function (){
            //console.log("鼠标移出");
        };
        div.onmousemove = function (){
           // console.log("鼠标移动");
        };
        //2.键盘事件(了解)
        onkeydown = function (e){
            /*
            键盘事件对象
            key属性返回按键对应的字符
            onkeydown:获取事件对象的which,功能键返回键盘编码;
            字符键一律返回大写字母的ASC码
            onkeypress:获取事件对象的which,返回字符键的ASC码值,
            区分大小写
            */
            console.log("onkeydown",e,e.key,e.which);
        };
        onkeypress = function (e){
            console.log("onkeypress",e,e.key,e.which);
        };
        onkeyup = function (){
            //console.log("onkeyup");
        };
    </script>
</body>
</html>
```

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
```

```
    <script>
        window.onload = function (){
            //窗口加载完毕后执行
            //console.log("window.onload");
            //输入框相关
            uname.onfocus = function (){
                //console.log("获取到焦点");
            };
            uname.onblur = function (){
                //this:指代事件的触发对象或函数的调用者
                //console.log("失去焦点:",this,uname.value,this.value);
            };
            //实时监听输入
            uname.oninput = function (){
                //console.log("oninput:",this.value);
            };
            //监听前后两次输入的内容是否一致
            uname.onchange = function (){
                //只有前后两次输入不一致,并且失去焦点时才触发
                console.log("onchange:",this.value);
            };
            //监听按钮状态的改变
            savePwd.onchange = function (){
                /*按钮有选中和未选中两种状态,
                对应checked属性值为true/false*/
                console.log(this.checked);
            };
            //监听表单中的数据是否可以提交
            /*
            交由form元素监听,在点击提交按钮时触发,允许返回布尔值,
            true表示允许发送,false表示阻止发送
            */
            form.onsubmit = function (){
                //用户名为空时阻止提交
                if(uname.value == ""){
                    return false;
                }else{
                    return true;
                }
            };
        };
        function fn(){
            console.log(this);
        }
        window.fn();
    </script>
</head>
<body>
    <form action="/login" method="get" enctype="" id="form">
        用户姓名:<input type="text" name="uname" id="uname"><br>
        记住密码:<input type="checkbox" id="savePwd" checked="checked"><br>
        <input type="submit">
    </form>
```

```
    <ul>
        <li>
            <span id="city">北京</span>
            <ol id="ol">
                <li>北京</li>
                <li>上海</li>
                <li>广州</li>
                <li>深圳</li>
            </ol>
        </li>
    </ul>
    <script>
        //console.log("测试");
        /*
        实现下拉菜单的点击传值(this练习)
        */
        //获取内层li元素列表
        var list = ol.children;
        for(var i = 0; i < list.length; i++){
            console.log("for:",i);
            list[i].onclick = function (){
                console.log(i);
                //传值
                //this指代事件触发对象
                city.innerHTML = this.innerHTML;
                console.log(this);
            };
        }

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

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        table{
            width:600px;
            border:1px solid #000;
        }
        th,td{
            border:1px solid #000;
        }

    </style>
```

```
</head>
<body>
    <div>
        <input type="text" name="gname" placeholder="商品名称">
        <input type="text" name="gprice" placeholder="商品价格">
        <input type="text" name="gcount" placeholder="商品数量">
        <button onclick="add()">增加</button>
    </div>
    <table>
        <thead>
            <tr>
                <th>商品名称</th>
                <th>商品价格</th>
                <th>商品数量</th>
                <th>
                    <button>操作</button>
                </th>
            </tr>
        </thead>
        <tbody id="tbody">

        </tbody>
    </table>
    <script>
        function add(){
            //1.获取输入框
            var list = document.getElementsByTagName("input");
            //2.获取输入框的值
            var gname = list[0].value;
            var gprice = list[1].value;
            var gcount = list[2].value;
            //3.创建元素节点
            var tr = document.createElement("tr");
            var td1 = document.createElement("td");
            td1.innerHTML = gname;

            var td2 = document.createElement("td");
            td2.innerHTML = gprice;

            var td3 = document.createElement("td");
            td3.innerHTML = gcount;

            var td4 = document.createElement("td");
            td4.innerHTML = "<button>修改</button><button>删除</button>";

            //4.添加显示
            tr.appendChild(td1);
            tr.appendChild(td2);
            tr.appendChild(td3);
            tr.appendChild(td4);
            tbody.appendChild(tr);
```

```
            /*
            var list
            var tr
            for(var i = 0; i < 3;i++){
                var td = document.....;
                td.innerHTML = list[i].value;
                tr.appendChild(td);
            }
            var td4 = ;
            tr.appendChild(td4);
            tbody.appendChild(tr)

            */

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