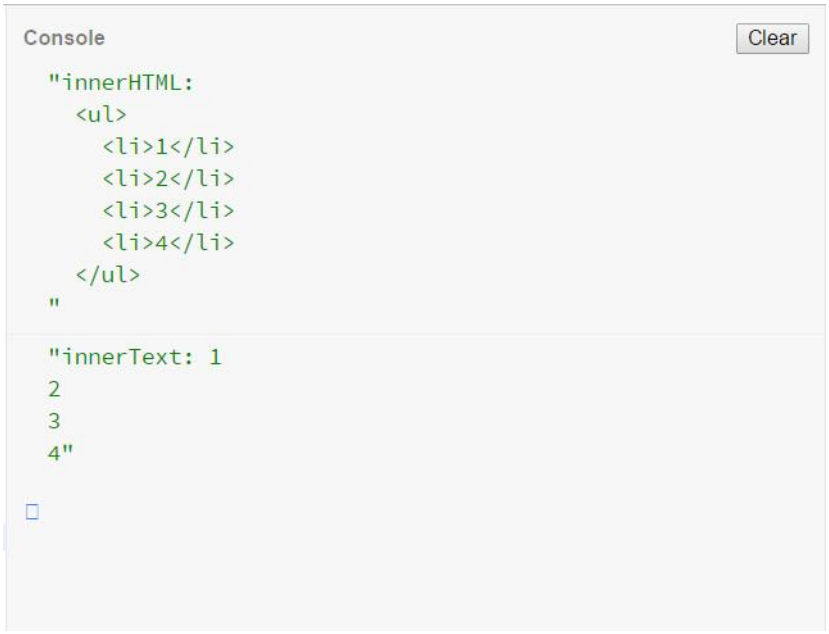
# task22

## 问答

### 1.dom对象的`innerText`和`innerHTML`有什么区别？

两者都是获取范围内的所有内容，但后者包含html标签，前者不包含。

```
<html>
<head><title>innerHTML</title></head>
<body>
<div id="header">
  <ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
  </ul>
</div>
<script>
    var content = document.getElementById("header");
    console.log('innerHTML: ' + content.innerHTML);
    console.log('innerText: ' + content.innerText);
</script>
</body>
</html>
```




### 2.`elem.children`和`elem.childNodes`的区别？

`elem.children`：返回指定元素子元素HTML节点的集合（只读），不包含文本节点。
`elem.childNodes`：返回指定元素子元素所有节点集合，包含文本节点、HTML节点和属性节点

### 3.查询元素有几种常见的方法？

1.通过document节点获取：`document.getElementById、document.getElementsByClassName、document.getElementsByTagName、document.querySele`
2.通过父节点获取：`parentObj.childNodes、parentObj.firstChild、parentObj.lastChild`
3.通过兄弟节点获取：`neighbourNode.nextSibling`
4.通过子节点获取：`childNode.parentNode`

### 4.如何创建一个元素？如何给元素设置属性？

`document.creatElement`:创建元素节点
`document.creatTextNode`:创建文本节点
`setAttribute`:设置属性

```
document.creatElement('ul');//创建ul元素
```

```
document.creatTextNode('表格');//创建文本'表格'
object.setAttribute(attribute,value);//设置object元素的attribute属性值为value
```

## 5.元素的添加、删除？

`appendChild`：将一个节点添加到父元素末尾处，该节点成为父元素最后一个节点。
`insertBefore`：在当前节点的某个子节点之前插入指定节点。
`replaceChild`：用指定节点替换当前节点的一个子节点。
`removeChild`：从某个父节点中删除指定子节点。

```
<html>
<head><title>innerHTML</title></head>
<body>
<div id="content">
    <p id="one">1</p>
    <p class="two">2</p>
    <p id="three">3</p>
</div>
<script>
    var content = document.getElementById('content');
    var one = document.getElementById('one');
    var two = document.getElementsByClassName('two')[0];
    var three = document.getElementById('three');
    var txt = document.createTextNode('txt');
    var txt1 = document.createTextNode('txt1');
    var txt2 = document.createTextNode('txt2');
    content.appendChild(txt);
    content.insertBefore(txt1,two);
    content.replaceChild(txt2,txt);
    content.removeChild(three);
</script>
</body>
</html>
```

## 6.DOM0 事件和DOM2级在事件监听使用方式上有什么区别？

DOM0级事件处理程序可以使用HTML元素属性来指定，也可以通过JavaScript来添加。其优点在于兼容性强、操作简单，而问题在于无法给一个事件添加多个事件处理程序，后添加的会覆盖先前的。

```
<button onclick = "console.log('点我')" id="btn">输出无数次</button>

<script>
var btn = document.getElementById("btn");
btn.onclick = function(){
  console.log("只输出一次");
  btn.onclick = null;
}


</script>
```

`addEventListener()`和`removeEventListener()`用于添加和删除事件处理程序，3个参数为事件名、事件函数和布尔值。布尔值默认为`false`，表示在冒泡阶段调用时间处理程序；为`true`则表示在捕获阶段调用时间处理函数。需要注意的是，在使用`removeEventListener()`时，参数必须和添加时的参数完全相同，这也意味着通过`addEventListener()`添加匿名函数无法被清除。另外，`addEventListener()`可以对一个事件同时添加多个事件处理程序。

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

<button  id="btn">点击就送</button>

<script>
var btn = document.querySelector("#btn")
var hello = function(){
  console.log('hello')
```

```
}
btn.addEventListener("click",hello,false);//添加事件处理程序
btn.removeEvenListener("click",hello,false);//删除事件处理程序
btn.addEventListener("click",function(){
 console.log('error');
},false);//添加事件处理程序
btn.removeEventListener("click",function(){
 console.log('error');
},false);//未删除事件处理程序

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

## 7.`attachEvent`与`addEventListener`的区别？

适用的浏览器不同：
`addEventListener`适用于现代浏览器
`attachEvent`是低版本IE浏览器的私有方法
参数和触发阶段不同:
`addEventListener`有三个参数，最后一个参数可以定义处理程序在捕获/冒泡阶段触发
`attachEvent`只有两个传递参数，只能在冒泡阶段触发
第一个参数的形式不同：
`addEventListener`的第一个参数是`click`
而`attachEvent`的第一个参数是`onclick`
`this`不同:
`addEventListener`中的`this`是触发事件的元素
`attachEvent`的`this`指代的是`window`
在同一个事件上绑定多个事件处理程序时的执行顺序不同`addEventListener`会按添加顺序并按照第三个参数执行
而`attachEvent`的执行是无序的

## 8.解释IE事件冒泡和DOM2事件传播机制？

IE事件冒泡是事件由第一个被触发的元素接收，然后逐级向上传播.
DOM2事件传播，事件由最外层元素接收，然后逐层向内传播，这个过程为捕获阶段，当达到目标元素时，处于目标阶段，然后事件由目标元素向最外层开始传递，这个过程称之为冒泡阶段.

## 9.如何阻止事件冒泡？ 如何阻止默认事件？

`e.stopPropagation()`;阻止事件冒泡,IE中则用`cancelBubble()`
`e.preventDefault()`;阻止默认事件,IE中需要把`returnValue`设为`false`

# 代码

## 1.有如下代码，要求当点击每一个元素`li`时控制台展示该元素的文本内容。不考虑兼容

```
<ul class="ct">
    <li>这里是</li>
    <li>饥人谷</li>
    <li>前端6班</li>
</ul>
<script>
        var point = function () {
            console.log(this.innerText);//定义事件处理函数，输出为打印元素的innerTe
xt
        }
        function a() {
            var li = document.getElementsByTagName('li');//获取li
            for (var i = 0; i < li.length; i++) {
                li[i].addEventListener('click',point,false);//遍历li，为每个li添
加事件处理程序
            }
        }
        a();
</script>
```

## 2.补全代码，要求：

当点击按钮开头添加时在 `<li>` 这里是 `</li>` 元素前添加一个新元素，内容为用户输入的非空字符串；当点击结尾添加时在 `<li>` 前端6班 `</li>` 后添加用户输入的非空字符串.
当点击每一个元素 `li` 时控制台展示该元素的文本内容。

```
<ul class="box">
    <li>这里是</li>
    <li>饥人谷</li>
    <li>前端6班</li>
</ul>
<input class="ipt-add-content" placeholder="添加内容"/>
<button id="btn-add-beginning">开头添加</button>
<button id="btn-add-ending">结尾添加</button>
<script>
       function beginning() {
            var atthebeginning = document.getElementById("btn-add-beginning");
            var pointbeginning = function () {
                var addstart = document.createElement("li");
                var txt = document.getElementsByClassName("ipt-add-content")[0]
.value;
                addstart.innerText = txt;
                var box = document.getElementsByClassName("box")[0];
                box.insertBefore(addstart,box.firstChild);
            }
            atthebeginning.addEventListener("click",pointbeginning,false);
        }
        function ending() {
            var attheending = document.getElementById("btn-add-ending");
            var pointending = function () {
                var addend = document.createElement("li");
                var txt = document.getElementsByClassName("ipt-add-content")[0]
.value;
                addend.innerText = txt;
                var box = document.getElementsByClassName("box")[0];
                box.appendChild(addend);
            }
            attheending.addEventListener("click",pointending,false);
        }
        function clicktoshow() {
            var point = function (a) {
                console.log(a.target.innerText);
            }
            var box = document.getElementsByClassName("box")[0];
            box.addEventListener("click",point,false);
        }
        beginning();
        ending();
        clicktoshow();
</script>
```

## 3.补全代码，要求：当鼠标放置在 `li` 元素上，会在 `img-preview` 里展示当前 `li` 元素的 `data-img` 对应的图片。

```
    <ul class="ct">
        <li data-img="http://img0.imgtn.bdimg.com/it/u=1727737719,4275581402&fm
=214&gp=0.jpg">鼠标放置查看图片1</li>
        <li data-img="http://img2.everychina.com/img/79/70/784edf4d977c13421cfd
dd605a69-600x400c1-743d/gas_powered_airsoft_pistol_bb_gun.jpg">鼠标放置查看图片2<
/li>
        <li data-img="http://img2.everychina.com/img/20/68/a20fbbd52253a6a504a8
9dff50fa-600x400c1-6ce3/1911_replica_gas_powered_airsoft_pistol_bb_gun.jpg">鼠
标放置查看图片3</li>
    </ul>
    <div class="img-preview">

    </div>

<script>
var ct = document.querySelector('.ct'),
    childs = ct.querySelectorAll('li'),
    preview = document.querySelector('.img-preview');

for(var i=0; i<childs.length; i++){
    childs[i].addEventListener('mouseenter', function(){
```

```
        var dataImg = this.getAttribute('data-img');
        preview.innerHTML = '<img src="' + dataImg + '">'
    });
}
</script>
```

4.实现如下图Tab切换的功能

5.实现下图的模态框功能