webAPI 第四天

一. 事件的绑定方式 和 组成部分

1.1 事件绑定方式 【回顾】

- 语法：

```
事件目标.事件类型 = 事件处理程序
```

- 代码：

```
var btn = document.getElementById('btn');
btn.onclick = function(){
  alert('hello');
}
```

1.2 事件组成部分 【回顾】

- 事件组成部分

```
事件目标：也被称为事件源，指的是要绑定事件的那个节点对象。
事件类型：指的是什么样的行为，如：鼠标点击、鼠标移入和移出、鼠标移动、键盘按下等等。
事件处理程序：事件触发后要执行的代码，用函数表示。函数体中的代码，在事件触发后执行。
```

二. 事件类型

2.1 鼠标事件【重点】

- onclick

> 作用：鼠标点击事件，鼠标点击某个节点对象的行为。

- onmouseover

> 作用：鼠标移入事件，鼠标移入某个节点对象的行为。

- onmouseout

> 作用：鼠标移出事件，鼠标移出某个节点对象的行为。

- onmouseenter

> 作用：和onmouseover相似，鼠标移入事件，鼠标移入某个节点对象的行为。但这个事件不会产生冒泡。

- onmouseleave

> 作用：和onmouseout相似，鼠标移出事件，鼠标移出某个节点对象的行为。但这个事件不会产生冒泡。

- onmousemove

> 作用：鼠标移动事件，鼠标在某个节点对象上移动时产生的行为。

- onmousedown

> 作用：鼠标按下事件，鼠标在某个节点对象上按下时产生的行为。

- onmouseup

> 作用：鼠标弹起事件，鼠标在某个节点对象上弹起时产生的行为。

- oncontextmenu【拓展】

> 作用：鼠标右键菜单事件，鼠标在页面上右键点击时产生的行为。

2.2 键盘事件 【重点】

- onkeydown

> 作用：键盘键按下事件。

- onkeyup

> 作用：键盘键弹起事件。

2.3 UI相关事件 【了解】

- onload

> 作用：页面加载完后（图片、视频、音频、各种文件等）要执行的程序

2.4 表单事件 【重点】

- onfocus

> 作用：元素获取焦点事件。针对文本框、密码框、文本域

- onblur

> 作用：元素失去焦点事件。针对文本框、密码框、文本域

- onchange

> 作用：元素内容改变事件。针对多选框

- oninput

> 作用：输入事件。针对文本框、密码框、文本域
> 兼容性：IE8及以下版本不支持，可以用onkeyup事件代替

三. 事件对象

3.1 什么是事件对象 【了解】

- 事键对象

> 每种类型的事件，不论是的鼠标点击、鼠标移入和离开、键盘按下等事件都与其对应的事件对象。
>
> 事件对象，是一个小的工具库，工具库中存放了和当前事件相关的各种信息和功能。（比如：鼠标点击时可以通过事件对象获取鼠标在屏幕上的坐标、键盘按下或弹起时可以通过事件对象获取按下的是哪个键、可以通过事件对象获取当前的事件类型是什么等等。）

3.2 如何获取事件对象 【重点】

- 获取方式一：标准方式

  - 语法：

```
事件目标.事件名 = function(形参){
    //这个 形参 就是当前事件相关的  事件对象
}
```

  - 代码：

```
//如：在网页上鼠标移动时，获取当前鼠标在浏览器界面上的坐标（x,y）
document.onmousemove = function(e){
  // e 就是事件对象
  console.log('x:' + e.clientX + ',Y:' + e.clientY);
}
```

- 获取方式二：IE低版本支持的方式
  - 语法：

```
事件目标.事件名 = function(形参){
    //这个 形参 就是当前事件相关的  事件对象
    //缺点：IE低版本（IE8及以下版本）不支持 这个形参 当做 事件对象
    //IE低版本获取事件对象的方式是： window.event;
    window.event;  //获取当前事件的事件对象
}
```

  - 代码：

```
document.onmousemove = function(e){
    // e 就是事件对象,但IE低版本不支持
    //IE低版本支持：window.event
    var _event = window.event;  //获取事件对象
    console.log('x:' + _event.clientX + ',Y:' + _event.clientY);
  }
```

- 兼容写法
  - 语法：

```
事件目标.事件名 = function(e){
    //形参e 标准获取事件对象的方式
    //window.event ie低版本获取事件对象的方式
    var _e = e||window.event; //兼容写法
    console.log('x:' + _e.clientX + ',Y:' + _e.clientY);
  }
```

  - 代码：

```
document.onmousemove = function(e){
    var _e = e||window.event; //兼容写法
    console.log('x:' + _e.clientX + ',Y:' + _e.clientY);
  }
```

3.3 事件对象的公共属性和方法【重点】

- 事件对象的公共属性和方法介绍

  > 公共，就是所有的事件类型的事件对象都可使用的属性和方法。

- 事件对象的公共属性

  - 事件对象.target
    - 作用：获取当前的事件目标节点对象。
    - 代码：

```html
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <button>按钮</button>
  <script>
    //① 获取按钮节点对象
    var btn = document.querySelector('button');
    //② 给按钮绑定点击事件
    btn.onclick = function(e){
      //先获取事件对象
      var _e = e||window.event;
      //获取事件对象中的target属性
      console.dir(_e.target);//获取到事件目标节点对象
      console.log(_e.target.innerHTML); //查看节点对象中的内容
    }
  </script>
</body>
</html>
```

- 兼容性问题:IE低版本（IE8及以下版本）不支持事件 对象的target属性。IE低版本用【事件对象.srcElement】代替target
- 兼容性处理方式：var *target = 事件对象.target||事件对象.srcElement;*

```html
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <button>按钮</button>
  <script>
    //① 获取按钮节点对象
    var btn = document.querySelector('button');
    //② 给按钮绑定点击事件
    btn.onclick = function(e){
      //先获取事件对象
      var _e = e||window.event;
      //获取事件对象中的target属性兼容处理方式
      var _target = _e.target||_e.srcElement;
      console.log(_target);//获取到事件目标节点对象
      console.dir(_target.innerHTML); //查看节点对象中的内容
    }
  </script>
</body>
</html>
```

- 事件对象.type
  - 作用：获取当前事件类型
  - 代码：

```
document.onclick = function(e){
  //先获取事件对象
  var _e = e||window.event;
  //查看当前的事件类型
  console.log(_e.type);  //click
}
```

- 事件对象的公共方法
  - 事件对象.preventDefault();
    - 作用：阻止和默认行为（如：浏览器默认鼠标右键显示菜单）
    - 兼容性：IE8及以下版本不支持，IE低版本用 事件对象.returnValue = false; 代替
    - 兼容性处理方式及代码：

```
document.oncontextmenu = function(e){
  //获取事件对象
  var _e = e||window.event;
  alert('鼠标右键被点击');
  if(_e.preventDefault){ //检测浏览器是否支持此方法
    _e.preventDefault();  //标准方式阻止浏览器的默认行为
  }else{ //不支持
    _e.returnValue = false; //IE低版本方式阻止浏览器的默认行为
  }
 }
```

  - 事件对象.stopPropagation();
    - 作用：阻止事件冒泡

- 兼容性：IE低版本（IE8及以下版本）不支持。 IE低版本中有事件对象.cancelBubble = true代替。
- 兼容性处理方式及代码：

```
1
2  <!doctype html>
3  <html lang="en">
4  <head>
5      <meta charset="UTF-8">
6      <meta name="Generator"
   content="EditPlus®">
7      <meta name="Author" content="">
8      <meta name="Keywords" content="">
9      <meta name="Description" content="">
10     <title>Document</title>
11     <style>
12         *{
13             margin:0;
14             padding:0;
15             line-height:30px;
16             text-align:center;
17             color:#fff;
18
19         }
20         .box0{
21             width:400px;
22             height:400px;
23             background-color:blue;
24             padding:50px;
25         }
26         .box1{
27             width:300px;
28             height:300px;
29             background-color:green;
30             padding:50px;
31         }
32         .box2{
33             width:200px;
```

```
            height:200px;
            background-color:purple;
            padding:50px;
        }
        .box3{
            width:200px;
            height:200px;
            background-color:yellow;
            color:#000;
        }
    </style>
</head>
<body>
<div class="box0">
    box0
    <div class="box1">
        box1
        <div class="box2">
            box2
            <div class="box3">box3</div>
        </div>
    </div>
</div>

<script>
    var divs =
document.getElementsByTagName("div");
    for(var i = 0;i<divs.length;i++){
        divs[i].onclick = function(e){
            alert(this.className);
            //获取事件对象
            var _e = window.event||e;
            //阻止事件冒泡

            if(_e.stopPropagation){ //浏览器是
```

```
否支持该方法
67                  _e.stopPropagation(); //标准
处理方式
68              }else{
69                  _e.cancelBubble = true;//IE低
版本处理方式
70              }
71          }
72      }
73  </script>
74  </body>
75  </html>
```

3.4 键盘事件对象相关的属性和方法【重点】

- 事件对象.altKey
  - 作用：检测是否按下键盘上的 Alt 键。 按下返回 true
  - 代码：

```
1  document.onkeydown = function(e){
2    var _e = window.event||e;
3    alert(_e.altKey);  //按下alt键，返回true
4  }
```

- 事件对象.ctrlKey
  - 作用：检测是否按下键盘上的 Ctrl 键。 按下返回 true
  - 代码：

```
1  document.onkeydown = function(e){
2    var _e = window.event||e;
3    alert(_e.ctrlKey);  //按下Ctrl键，返回true
4  }
```

- 事件对象.shiftKey
  - 作用：检测是否按下键盘上的 Shift 键。 按下返回 true
  - 代码：

```
document.onkeydown = function(e){
  var _e = window.event||e;
  alert(_e.shiftKey);  //按下shift键，返回true
}
```

- 事件对象.keyCode
  - 作用：返回被敲击的键生成的 Unicode 字符码(ascii码)
  - 代码：

```
document.onkeydown = function(e){
  var _e = window.event||e;
  alert(_e.keyCode); //返回ascii码表对应的十进制的数字
}
```

3.5 鼠标事件对象相关的属性和方法【重点】

- 事件对象.clientX / 事件对象.clientY
  - 作用：鼠标在浏览器可视区域中的坐标
  - 代码：

```html
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
  <style>
    body {
      height: 2000px;
    }
  </style>
</head>
<body>
  <script>
    document.onclick = function(e){
      //获取事件对象
      var _e = window.event||e;
      //获取鼠标在浏览器可视区域中的坐标
      alert('x:' + _e.clientX + ',y:' + _e.clientY);
    }
  </script>
</body>
</html>
```

- 事件对象.offsetX / 事件对象.offsetY
  - 作用：获取鼠标在指定的元素的区域中的坐标
  - 代码：

```html
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
  <style>
    div {
      width: 300px;
      height: 300px;
      background-color: red;
      margin:100px auto;
      cursor: default;
    }
  </style>
</head>
<body>
  <div>我第div</div>
  <script>
    var divNode = document.querySelector('div');
    divNode.onclick = function(e){
      //获取事件对象
      var _e = window.event||e;
      //获取鼠标在div中的坐标
      alert('X:' + _e.offsetX + ',Y:' + _e.offsetY);
    }
  </script>
</body>
</html>
```

- 事件对象.pageX / 事件对象.pageY
  - 作用：获取鼠标在整个文档区域中的坐标
  - 代码：

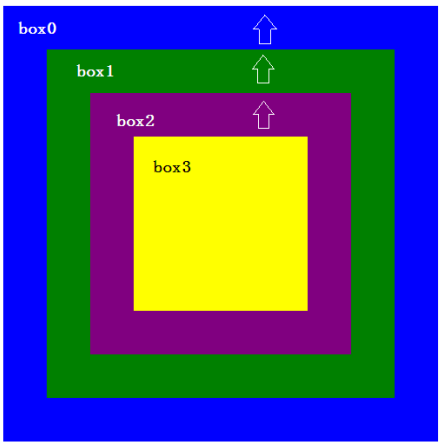
```html
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
  <style>
    body {
      height: 2000px;
    }
  </style>
</head>
<body>
  <script>
    document.onclick = function(e){
      //获取事件对象
      var _e = window.event||e;
      //获取鼠标在整个文档区域中的坐标
      alert('x:' + _e.pageX + ',y:' + _e.pageY);
    }
  </script>
</body>
</html>
```

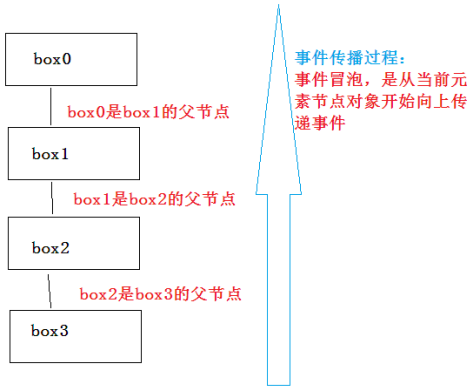
四. 事件流之事件冒泡

4.1 事件流介绍 【了解】

- 事件流
  - 事件流之的是事件的传播方式，传播方式有两种：事件冒泡 和事件捕获

- 事件冒泡的事件传播方式：目标→父级→父级....





- 代码：

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="Generator" content="EditPlus®">
    <meta name="Author" content="">
    <meta name="Keywords" content="">
    <meta name="Description" content="">
    <title>Document</title>
    <style>
        *{
            margin:0;
            padding:0;
            line-height:30px;
            text-align:center;
            color:#fff;

        }
        .box0{
            width:400px;
            height:400px;
            background-color:blue;
            padding:50px;
        }
        .box1{
            width:300px;
            height:300px;
            background-color:green;
            padding:50px;
        }
        .box2{
            width:200px;
            height:200px;
```

```
            background-color:purple;
            padding:50px;
        }
        .box3{
            width:200px;
            height:200px;
            background-color:yellow;
            color:#000;
        }
    </style>
</head>
<body>
<div class="box0">
    box0
    <div class="box1">
        box1
        <div class="box2">
            box2
            <div class="box3">box3</div>
        </div>
    </div>
</div>

<script>
    var divs =
document.getElementsByTagName("div");
    for(var i = 0;i<divs.length;i++){
        divs[i].onclick = function(e){
            alert(this.className);
        }
    }
</script>
</body>

</html>
```

4.2 事件冒泡的作用【了解】

- 可以实现 事件委托（事件代理）

> 事件委托指的是子孙元素的事件绑定，完全交给其上级父元素或祖先元素绑定。

- 为什么学习事件委托（事件代理）

> 在web前端开发中，并不是js事件越多越好，js事件绑定的越多，就越消耗程序的性能。所以，在某些情况下，为了提高程序的性能，应当减少事件的绑定。
>
> 传统的事件处理中，需要为每个元素添加事件处理器。js事件代理则是一种简单有效的技巧，通过它可以把事件处理器添加到一个父级元素上，从而避免把事件处理器添加到多个子级元素上。

- 事件委托（事件代理）的原理：

> 事件代理的原理用到的就是事件冒泡和目标元素，把事件处理器添加到父元素，等待子元素事件冒泡，并且父元素能够通过target（IE为srcElement）判断是哪个子元素，从而做相应处理。

- 事件委托的优点:

> ① 可以提高程序的性能。
> ② 可以为未来新添加的元素绑定事件。

- 代码：

```
//需求：点击一个div中的p标签时，弹出p标签里的内容
```

- 代码：

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
  <style>
    p {
      background-color: blue;;
    }
  </style>
</head>
<body>
  <div>
    <h2>标题</h2>
    <p>段落1</p>
    <p>段落2</p>
    <h2>标题</h2>
    <p>段落3</p>
    <h2>标题</h2>
    <p>段落4</p>
    <p>段落5</p>
    <h2>标题</h2>
    <p>段落6</p>
    <p>段落7</p>
    <h2>标题</h2>
  </div>
  <script>
    var divNode = document.querySelector('div');
    divNode.onclick = function(e){
      //获取事件对象
      var _e = window.event||e;
      //获取目标（在冒泡）
      var _target = _e.target||_e.srcElement;
      //console.log(_target);
```

```
35        //节点对象.tagName  获取节点对象对应的标签名返回的是大写
36        if(_target.tagName.toLowerCase()=='p'){
37          alert(_target.innerHTML);
38        }
39      }
40    </script>
41  </body>
42  </html>
```