Web API第三天

一. 操作节点对象

1.1 创建节点对象 【重点】

• 方式一【了解】

```
语法: document.write('内容');缺点: 会覆盖整个网页的内容代码:
```

```
1 <!DOCTYPE html>
 2 <html>
3 <head lang="en">
 4 <meta charset="UTF-8">
   <title></title>
 6 </head>
 7 <body>
    <button id="btn">添加</button>
    <script>
      var btn = document.getElementById('btn');
10
       btn.onclick = function(){
11
        //缺点:覆盖了网页的内容,之前的按钮元素会消
12
   失。
         document.write('<h1>标题</h1>');
13
14
      };
15 </script>
16 </body>
17 </html>
```

• 方式二【重点】

。 语法: 节点对象.innerHTML = '内容';

○ 优点:对于添加多嵌套的内容操作方便。

○ 缺点: 会覆盖部分网页元素以及事件。

o 代码:

```
<!DOCTYPE html>
2
   <html>
  <head lang="en">
3
     <meta charset="UTF-8">
4
5
     <title></title>
6
   </head>
7
   <body>
     <button id="btn">添加</button>
8
9
     <1i>>
10
        <i>\i>张三</i>
11
12
      <
13
14
        <i>>李四</i>
      15
   16
   <script>
17
      //按钮节点对象
18
      var btn = document.getElementById('btn');
19
      //ul节点对象
20
21
      var nameList =
   document.getElementById('nameList');
      //给ul中的每一个li添加点击事件,获取姓名
22
23
      for(var i =
   0;i<nameList.children.length;i++){</pre>
        nameList.children[i].onclick = function(){
24
25
          alert(this.innerText);
26
        }
27
      //点击按钮添加一个li到ul中
28
      btn.onclick = function(){
29
        //会覆盖掉ul中之前所有的li
30
        nameList.innerHTML = '<i>王五</i>
31
```

```
';
      //解决覆盖的问题: 在添加新的内容之前, 把旧的内
32
  容获取到一起添加。但会之前旧的内容的事件就会消失。
33
       nameList.innerHTML = nameList.innerHTML +
34
  '<i>王五</i>';
      */
35
      //解决事件消失:事件委托【后面讲到】
36
37
38 };
39 </script>
40 </body>
41 </html>
```

• 方式三【重点】

。 语法: document.createElement('标签名');

o 优点: 不会覆盖原有的元素的事件

。 缺点: 对于添加嵌套多的内容操作麻烦。

o 代码:

```
<!DOCTYPE html>
2 <html>
3 <head lang="en">
     <meta charset="UTF-8">
4
5
     <title></title>
6
   </head>
7
   <body>
8 <button id="btn">添加</button>
   <1i>>
10
      <i>\i>张三</i>
11
12
    <1i>>
13
14
      <i>> 李四</i>
     15
16 
   <script>
17
   //按钮节点对象
18
   var btn = document.getElementById('btn');
19
    //ul节点对象
20
21
     var nameList =
   document.getElementById('nameList');
     //给ul中的每一个li添加点击事件,获取姓名
22
23
     for(var i = 0;i<nameList.children.length;i++){</pre>
24
       nameList.children[i].onclick = function(){
        alert(this.innerText);
25
26
      }
27
28
     //点击按钮添加一个li到ul中
     btn.onclick = function(){
29
      //创建新的li节点对象
30
      var newLi = document.createElement('li');
31
      //创建新的i节点对象
32
      var i = document.createElement('i');
33
```

```
//给i节点对象添加文本内容
34
35
      i.innerHTML = '\pm\Xi';
   //把i放到newLi中
36
37
      newLi.appendChild(i);
     //把newLi放到ul中
38
      nameList.appendChild(newLi);
39
40
    };
41 </script>
42 </body>
43 </html>
```

- innerHTML创建节点对象 和 document.createElement('标签') 的效率问题 【了解】
 - o innerHTML 会产生字符串解析,由于字符串的不可变性,尽量避免大量的拼接,否则消耗内存,影响性能。
 - o document.createElement('标签')创建的性能要比innerHTML要高,但是若涉及到多层嵌套内容时,代码操作麻烦。
 - 。 所以,一般情况下,两者配合使用较多

1.2 添加节点对象 【重点】

- 方式一:
 - 。 语法: 父节点对象.appendChild(新的子节点对象);
 - 。 作用: 把一个新的子节点对象追加到父节点对象中的最后。
 - 0 代码:

```
1
   <!DOCTYPE html>
2
   <html>
   <head lang="en">
3
     <meta charset="UTF-8">
4
5
     <title></title>
6
     <style>
7
       div {
         border:1px solid red;
8
9
10
       p{
         background-color: blue;
11
12
         color:gold
13
       }
     </style>
14
   </head>
15
   <body>
16
     <button id="btn">添加</button>
17
     <div id="box">
18
19
       \段落
     </div>
20
21
     <script>
       //按钮节点对象
22
       var btn = document.getElementById('btn');
23
24
       //div节点对象
       var box = document.getElementById('box');
25
       //点击按钮添加p节点对象到div中
26
27
       btn.onclick = function(){
         //创建P节点对象
28
29
         var p = document.createElement('p');
         //设置p节点对象中的内容
30
         p.innerHTML = '段落';
31
         //把p追加到div中
32
         box.appendChild(p);
33
34
```

```
35 </script>
36 </body>
37 </html>
```

• 方式二:

- 。 语法: 父节点对象.insertBefore(新的子节点对象,旧的子节点对象);
- 作用:把新的子节点对象追加到父节点对象中的旧的子节点对象之前。
- 0 代码:

```
1
   <!DOCTYPE html>
2
   <html>
   <head lang="en">
3
     <meta charset="UTF-8">
4
5
     <title></title>
6
     <style>
7
       div {
        border:1px solid red;
8
9
      }
10
       p{
        background-color: blue;
11
12
        color:gold
13
       }
14
     </style>
15 </head>
16 <body>
   <button id="btn">添加</button>
17
18 <div id="box">
19
     段落
   </div>
20
   <script>
21
    //按钮节点对象
22
23
   var btn = document.getElementById('btn');
24
   //div节点对象
    var box = document.getElementById('box');
25
    //旧的p节点对象
26
27
     var p = document.getElementById('p');
     //点击按钮添加h1标签节点对象到div中的p节点对象之前
28
29
     btn.onclick = function(){
      //创建h1节点对象
30
      var h1 = document.createElement('h1');
31
      //设置h1节点对象中的内容
32
      h1.innerHTML = '标题';
33
      //把h1追加到div中
34
```

```
box.insertBefore(h1,p);

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

1.3 删除节点对象 【重点】

• 方式一:

。 语法: 父节点对象.removeChild(子节点对象);

。 作用: 删除父节点对象中的某一个子节点对象

0 代码:

```
<!DOCTYPE html>
2
   <html>
3 <head lang="en">
     <meta charset="UTF-8">
4
5
     <title></title>
6
   </head>
7
   <body>
     <button id="btn">删除</button>
8
9
     <u1>
       *社大
10
       *li>老二
11
12
    <script>
13
14
      //按钮节点对象
      var btn = document.getElementById('btn');
15
      //第一个li节点对象
16
       var li1 =
17
   document.getElementsByTagName('li')[0];
      //点击按钮删除ul中的第一个li
18
       btn.onclick = function(){
19
        li1.parentNode.removeChild(li1);
20
21
      };
    </script>
22
23 </body>
24 </html>
```

• 方式二: 【拓展,了解】

○ 语法: 节点对象.remove();

○ 作用: 删除某一个节点对象

。 缺点: 有兼容性问题, ie低版本浏览器不支持

0 代码:

```
<!DOCTYPE html>
2 <html>
3 <head lang="en">
     <meta charset="UTF-8">
4
5
    <title></title>
   </head>
7
   <body>
     <button id="btn">删除</button>
8
9
     <u1>
      老大
10
      *li>老二
11
12
   <script>
13
14
      //按钮节点对象
      var btn = document.getElementById('btn');
15
      //第一个li节点对象
16
      var li1 =
17
   document.getElementsByTagName('li')[0];
      //点击按钮删除ul中的第一个li
18
       btn.onclick = function(){
19
        li1.remove();
20
21
      };
22 </script>
23 </body>
24 </html>
```

1.4 复制节点对象 【拓展,了解】

- 语法: 节点对象.cloneNode(boolean);
 - 节点对象.cloneNode();

默认为false,浅拷贝,只复制外层的标签,不复制里面的内容;注意:无法复制事件

。 节点对象.cloneNode(true);

深拷贝,复制该节点对象的所有内容(包含外层的标签和里面的内

容);

注意: 无法复制事件

• 代码:

```
1
   <!DOCTYPE html>
 2
   <html>
   <head lang="en">
 3
     <meta charset="UTF-8">
 4
 5
     <title></title>
 6
     <style>
 7
       #box {
         border:1px solid red;
 8
 9
       }
       .item {
10
11
         width: 200px;
12
         height: 200px;
         background-color: #000;
13
         color:gold;
14
15
         margin: 10px;
16
       }
17
     </style>
   </head>
18
19
   <body>
     <button id="add">添加</button>
20
21
    <div id="box">
       <div class="item">
22
         <h2>标题</h2>
23
24
         \段落
         <i>>斜体</i>
25
       </div>
26
27
     </div>
     <script>
28
29
       //按钮节点对象
30
       var add = document.getElementById('add');
       //box节点对象
31
32
       var box = document.getElementById('box');
       //item节对象
33
       var item = box.children[0];
34
```

```
//点击item, 更改item背景色
35
       item.onclick = function(){
36
         item.style.backgroundColor='blue';
37
38
       };
       //点按钮,复制一份item并追加到box中
39
       add.onclick = function(){
40
         //克隆item返回一个新的item
41
         var newItem = item.cloneNode(true);
42
43
         box.appendChild(newItem);
       }
44
     </script>
45
46 </body>
47 </html>
```

1.5 替换节点对象 【拓展,了解】

- 语法: 父节点对象.replaceChild(newNode,oldNode);
- 作用: 替换父节点对象中的某个子节点对象
- 代码:

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
    <meta charset="UTF-8">
4
5
    <title></title>
6
  </head>
7
  <body>
     <button id="btn">替换</button>
8
    <l
9
      ti>我是老大
10
      tli>我是老二
11
      tli>我是老三
12
13
    14
    <script>
      //按钮节点对象
15
      var btn = document.getElementById('btn');
16
      //ul节点对象
17
      var ul = document.querySelector('ul');
18
      //点击按钮替换第二li节点对象
19
      btn.onclick = function(){
20
21
        //创建一个新的节对象
        var newLi = document.createElement('li');
22
        newLi.innerHTML = '<i>我是two</i>';
23
        //替换
24
25
        ul.replaceChild(newLi,ul.children[1]);
26
      }
27
28 </script>
29 </body>
30 </html>
```