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：

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <button id="btn">添加</button>
  <script>
    var btn = document.getElementById('btn');
    btn.onclick = function(){
      //缺点：覆盖了网页的内容，之前的按钮元素会消失。
      document.write('<h1>标题</h1>');
    };
  </script>
</body>
</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。
    - 代码：

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <button id="btn">添加</button>
  <ul id="nameList">
    <li>
      <i>张三</i>
    </li>
    <li>
      <i>李四</i>
    </li>
  </ul>
  <script>
    //按钮节点对象
    var btn = document.getElementById('btn');
    //ul节点对象
    var nameList =
document.getElementById('nameList');
    //给ul中的每一个li添加点击事件，获取姓名
    for(var i =
0;i<nameList.children.length;i++){
      nameList.children[i].onclick = function(){
        alert(this.innerText);
      }
    }
    //点击按钮添加一个li到ul中
    btn.onclick = function(){
      //会覆盖掉ul中之前所有的li


      nameList.innerHTML = '<li><i>王五</i>
```

```
</li>';
      //解决覆盖的问题：在添加新的内容之前，把旧的内容获取到一起添加。但会之前旧的内容的事件就会消失。
      /*
      nameList.innerHTML = nameList.innerHTML + '<li><i>王五</i></li>';
      */
      //解决事件消失：事件委托【后面讲到】

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

- 方式三【重点】
  - 语法：document.createElement('标签名');
  - 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。
  - 代码：

```html
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
<button id="btn">添加</button>
<ul id="nameList">
  <li>
    <i>张三</i>
  </li>
  <li>
    <i>李四</i>
  </li>
</ul>
<script>
  //按钮节点对象
  var btn = document.getElementById('btn');
  //ul节点对象
  var nameList =
document.getElementById('nameList');
  //给ul中的每一个li添加点击事件，获取姓名
  for(var i = 0;i<nameList.children.length;i++){
    nameList.children[i].onclick = function(){
      alert(this.innerText);
    }
  }
  //点击按钮添加一个li到ul中
  btn.onclick = function(){
    //创建新的li节点对象
    var newLi = document.createElement('li');
    //创建新的i节点对象
    var i = document.createElement('i');
```

```
    //给i节点对象添加文本内容
    i.innerHTML = '王五';
    //把i放到newLi中
    newLi.appendChild(i);
    //把newLi放到ul中
    nameList.appendChild(newLi);
  };
</script>
</body>
</html>
```

- innerHTML创建节点对象 和 document.createElement('标签') 的效率问题 【了解】
  - 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。
  - 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。
  - 代码：

```html
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
  <style>
    div {
      border:1px solid red;
    }
    p{
      background-color: blue;
      color:gold
    }
  </style>
</head>
<body>
  <button id="btn">添加</button>
  <div id="box">
    <p>段落</p>
  </div>
  <script>
    //按钮节点对象
    var btn = document.getElementById('btn');
    //div节点对象
    var box = document.getElementById('box');
    //点击按钮添加p节点对象到div中
    btn.onclick = function(){
      //创建P节点对象
      var p = document.createElement('p');
      //设置p节点对象中的内容
      p.innerHTML = '段落';
      //把p追加到div中
      box.appendChild(p);
    }
```

```
  </script>
</body>
</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。
  - 代码：

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
  <style>
    div {
      border:1px solid red;
    }
    p{
      background-color: blue;
      color:gold
    }
  </style>
</head>
<body>
<button id="btn">添加</button>
<div id="box">
  <p id="p">段落</p>
</div>
<script>
  //按钮节点对象
  var btn = document.getElementById('btn');
  //div节点对象
  var box = document.getElementById('box');
  //旧的p节点对象
  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
  btn.onclick = function(){
    //创建h1节点对象
    var h1 = document.createElement('h1');
    //设置h1节点对象中的内容
    h1.innerHTML = '标题';
    //把h1追加到div中
```

```
    box.insertBefore(h1,p);
  }
</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
  - 代码：

```html
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <button id="btn">删除</button>
  <ul>
    <li>老大</li>
    <li>老二</li>
  </ul>
  <script>
    //按钮节点对象
    var btn = document.getElementById('btn');
    //第一个li节点对象
    var li1 =
document.getElementsByTagName('li')[0];
    //点击按钮删除ul中的第一个li
    btn.onclick = function(){
      li1.parentNode.removeChild(li1);
    };
  </script>
</body>
</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
  - 代码：

```html
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <button id="btn">删除</button>
  <ul>
    <li>老大</li>
    <li>老二</li>
  </ul>
  <script>
    //按钮节点对象
    var btn = document.getElementById('btn');
    //第一个li节点对象
    var li1 =
document.getElementsByTagName('li')[0];
    //点击按钮删除ul中的第一个li
    btn.onclick = function(){
      li1.remove();
    };
  </script>
</body>
</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：

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
  <style>
    #box {
      border:1px solid red;
    }
    .item {
      width: 200px;
      height: 200px;
      background-color: #000;
      color:gold;
      margin: 10px;
    }
  </style>
</head>
<body>
  <button id="add">添加</button>
  <div id="box">
    <div class="item">
      <h2>标题</h2>
      <p>段落</p>
      <i>斜体</i>
    </div>
  </div>
  <script>
    //按钮节点对象
    var add = document.getElementById('add');
    //box节点对象
    var box = document.getElementById('box');
    //item节对象
    var item = box.children[0];
```

```
    //点击item，更改item背景色
    item.onclick = function(){
      item.style.backgroundColor='blue';
    };
    //点按钮，复制一份item并追加到box中
    add.onclick = function(){
      //克隆item返回一个新的item
      var newItem = item.cloneNode(true);
      box.appendChild(newItem);
    }
  </script>
</body>
</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：

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <button id="btn">替换</button>
  <ul>
    <li>我是老大</li>
    <li>我是老二</li>
    <li>我是老三</li>
  </ul>
  <script>
    //按钮节点对象
    var btn = document.getElementById('btn');
    //ul节点对象
    var ul = document.querySelector('ul');
    //点击按钮替换第二li节点对象
    btn.onclick = function(){
      //创建一个新的节对象
      var newLi = document.createElement('li');
      newLi.innerHTML = '<i>我是two</i>';
      //替换
      ul.replaceChild(newLi,ul.children[1]);
    }

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