DAY11 DOM节点的操作

2023年5月5日 18:31

获取元素的尺寸以及偏移量(css有关系)

```
<style>
    * {
      padding: 0;
      margin: 0;
    }
    .box {
      width: 100px;
      height: 100px;
      background-color: red;
      margin: 50px;
      padding: 20px;
      border: 1px solid blue;
      overflow: scroll;
    }
    .father {
      width: 400px;
      height: 400px;
      border: 1px solid red;
      margin: 100px;
      /* position: relative; */
    }
    .child {
      width: 100px;
      height: 100px;
      background-color: orange;
    }
  </style>

  <!-- <div class="box"></div> -->
  <div class="father">
    <div class="child">
    </div>
  </div>
```

1.offsetWidth / offsetHeight 是只读属性，返回一个元素的布局宽高

获取盒模型的尺寸：自身的宽高 + padding的值 + border的值

没有单位

```
// var box = document.querySelector('.box');
// console.log(box.offsetWidth, box.offsetHeight);//142 142
// console.log(parseInt(window.getComputedStyle(box)['width']));
// console.log(window.getComputedStyle(box)['padding-left']);
// console.log(window.getComputedStyle(box)['padding-right']);
// console.log(window.getComputedStyle(box)['border-width']);
```

2.clientWidth / clientHeight

获取盒子的尺寸：自身的宽高 + padding的值

和自身的滚动条尺寸有关系

没有单位

```
// console.log(box.clientWidth, box.clientHeight);//123 123
```

3.offsetLeft / offsetTop 是只读属性，获取元素对象的位置(无论是否存在定位都可以获取)

元素相对于定位父级的距离，但是没有定位父级，以body为基准。

```
// var box = document.querySelector('.box');
// console.log(box.offsetLeft, box.offsetTop);
```

父级存在定位,以父级为基准

```
// var child = document.querySelector('.child');
// console.log(child.offsetLeft, child.offsetTop);//0 0
```

4.clientLeft / clientTop

以父级为基准

```
// var child = document.querySelector('.child');
// console.log(child.offsetLeft, child.offsetTop);
// console.log(child.clientLeft, child.clientTop);
```

5.offsetParent：获取定位父级

如果当前元素的父级元素没有进行CSS定位，offsetParent为body，否则就是定位父级

```
// var child = document.querySelector('.child');
// console.log(child.offsetParent);
```

重点的属性

offsetWidth/offsetHeight 盒模型尺寸

offsetLeft /offsetTop 盒子的位置(无论是否存在定位都可以获取)

offsetParent 获取定位父级，不存在定位父级，就是body

创建、添加、删除

一.创建元素

```
<div class="top">
  <hr>
</div>
```

1.createElement(创建的元素名)，返回创建的元素对象

```
// var cDiv = document.createElement('div');//创建一个元素对象div
// 可以给创建的元素对象添加属性和方法
// 添加属性
// cDiv.innerHTML = '我是创建出来的div';
```

```
// cDiv.className = 'hehe';
// cDiv.style.cssText = 'width:100px;height:100px;background:red';
// 添加事件(被动的方法)
// cDiv.onclick = function () {
//   cDiv.style.backgroundColor = 'orange';
// }
```

二.追加元素，添加元素

1.父元素.appendChild(追加的元素对象)，将追加(创建)的元素添加到某个父元素内部的最后。

```
// document.body.appendChild(cDiv);
// document.querySelector('.top').appendChild(cDiv);
```

三.插入元素

1.父元素.insertBefore(新的元素，存在的元素)，将新的元素插入到存在的元素的前面。

```
// document.querySelector('.top').insertBefore(cDiv,
document.querySelector('hr'));
// document.querySelector('.top').insertBefore(cDiv, null);//如果第二个参数是null，
```

直接追加到最后的位置。

四.删除元素

1.删除子元素：父元素.removeChild(子元素)

```
// document.querySelector('.top').removeChild(document.querySelector('hr'));
```

2.删除自身：元素对象.remove()

```
// document.querySelector('.top').remove();
```

克隆、替换

一.克隆元素

元素节点.cloneNode(true)克隆(复制)元素节点，参数为true，连同元子元素一起克隆.

```
// var box = document.querySelector('.box');
// var cloneElement = box.cloneNode(true);//克隆
// document.body.appendChild(cloneElement);//追加
// document.body.innerHTML += box.outerHTML
```

二.替换元素

父节点.replaceChild(新节点，替换的节点)替换子节点.

```
// var box = document.querySelector('.box');
// var cSpan = document.createElement('span');//创建span
// document.body.replaceChild(cSpan, box);
```

核心关注的方法

createElement
appendChild
insertBefore
remove、removeChild

DOM节点类型

一.DOM节点类型 - 高级DOM操作

1.元素节点:html标签名称
2.属性节点:html标签里面的属性
3.文本节点:html标签里面的文本内容(不包含标签)
4.注释节点:<!---->结构以及结构里面的内容
.....

二.每个节点有三个核心的属性

1.nodeType： 节点的类型，使用数字表示
2.nodeName： 节点的名称
3.nodeValue：节点的值

三.获取节点的方式

```
<!-- 这里是顶部区域 -->
<div class="top" id="header1" name="zhangsan">top</div>
<!-- 这里是头部区域 -->
<div class="header">header</div>
<!-- 这里是JavaScript代码区域 -->
<script src=""></script>
<!-- 这里是底部区域 -->
<div class="footer">footer</div>
```

1.childNodes：获取当前元素节点的所有子节点，这里面包含空白节点，返回类数组
空白节点：换行产生

```
// console.log(document.body.childNodes);
// console.log(document.body.childNodes.length);//18
// console.log(document.body.childNodes[0]);//#text  文本节点
// console.log(document.body.childNodes[1]);//<!-- 这里是顶部区域 -->
// console.log(document.body.childNodes[3]);//<div class="top">top</div>
```

1.1.获取元素节点，测试元素节点下面的三个常用属性

```
// var elementNode = document.body.childNodes[3];//<div class="top">top</div>
// console.log(elementNode.nodeType);// 1 元素节点
```

```
// console.log(elementNode.nodeName);// DIV  大写的标签名
// console.log(elementNode.nodeValue);// null
```

1.2.获取元素节点里面的属性节点

attributes

```
// console.log(elementNode.attributes);//返回类数组
// console.log(elementNode.attributes.length);//3
// console.log(elementNode.attributes[0]);//class="top"
// console.log(elementNode.attributes[1]);//id="header1"
// console.log(elementNode.attributes[2]);//name="zhangsan"

// console.log(elementNode.attributes[1].nodeType);//2 属性节点
// console.log(elementNode.attributes[1].nodeName);// id
// console.log(elementNode.attributes[1].nodeValue);//header1
```

1.3.获取元素节点里面的文本节点

```
// console.log(document.body.childNodes[3].childNodes[0]);//获取文本节点
// console.log(document.body.childNodes[3].childNodes[0].nodeType);//3 文本节点
// console.log(document.body.childNodes[3].childNodes[0].nodeName);//#text
// console.log(document.body.childNodes[3].childNodes[0].nodeValue);//top
```

1.4.获取注释节点

```
// console.log(document.body.childNodes[1]); //< !--这里是顶部区域 -->
// console.log(document.body.childNodes[1].nodeType); //8 注释节点
// console.log(document.body.childNodes[1].nodeName); //#Comment
// console.log(document.body.childNodes[1].nodeValue); //这里是顶部区域
```

2.children： 获取当前元素节点的所有子节点，这里不包含空白节点和注释节点，返回类数组

```
// console.log(document.body.childNodes.length);//18
// console.log(document.body.children.length);//5
// console.log(document.body.children[0]);
// console.log(document.body.children[0].nodeType);
// console.log(document.body.children[0].nodeName);
// console.log(document.body.children[0].nodeValue);
// console.log(document.body.children[0].attributes);
```

3.parentNode：获取当前节点的父节点

```
// console.log(document.body.children[0]);//<div class="top">top</div>
// console.log(document.body.children[0].parentNode);//body
// console.log(document.body.children[0].parentNode.parentNode);//html
//
console.log(document.body.children[0].parentNode.parentNode.parentNode);//document
//
console.log(document.body.children[0].parentNode.parentNode.parentNode.parentNode);
//null
```

了解节点的高级选取

```
  <div class="box">
    <ul>
      <li>11111</li>
      <li>22222</li>
      <li>33333</li>
      <li>44444</li>
      <li>55555</li>
    </ul>
  </div>
</body>
```

```
console.log(document.querySelector('.box ul'));
console.log(document.body.children[0].children[0]);
console.log(document.body.childNodes[1].childNodes[1]);

var ul = document.body.children[0].children[0];
1.firstChild: 第一个子节点(包含空白注释)
  console.log(ul.firstChild);
2.firstElementChild: 第一个子节点(不包含空白注释)
  console.log(ul.firstElementChild);
3.lastChild: 最后一个子节点(包含空白注释)
  console.log(ul.lastChild);
4.lastElementChild: 最后一个子节点(不包含空白注释)
  console.log(ul.lastElementChild);
5.previousSibling: 上一个兄弟节点(包含空白注释)
  console.log(ul.firstElementChild.previousSibling);
6.previousElementSibling: 上一个兄弟节点(不包含空白注释)
  console.log(ul.firstElementChild.previousElementSibling);
7.nextSibling: 下一个兄弟节点(包含空白注释)
  console.log(ul.firstElementChild.nextSibling);
8.nextElementSibling: 下一个兄弟节点(不包含空白注释)
  console.log(ul.firstElementChild.nextElementSibling);
```

有意义的单词

first：第一个

Child：子元素

last：最后

Element：元素

previous：上一个

next：下一个

sibling:兄弟