학습결과서

이름 : 이채미

날짜 : 2020.07.03(목)

오늘 학습 한 내용

* Element 객체 및 관련 API, Node 객체 및 관련 API

오늘 알게된 사실

* JS에서 모든 Element의 속성은 attribute와 property를 통해 제어 가능하며 두 방식의 리턴값이 다를 수 있다. property 방식을 통한 속성 제어가 더 간편하고 속도도 빠르지만 실제 속성 이름과 다른 경우가 있으므로 주의해야 한다.

<a id="target" href="./demo1.html">ot</a>

<script>

//현재 웹페이지가 http://localhost/webjs/Element/attribute\_api/demo3.html 일 때

var target = document.getElementById('target');

console.log('target.href', target.href);

// http://localhost/webjs/Element/attribute\_api/demo1.html

console.log('target.getAttribute("href")', target.getAttribute("href"));

// ./demo1.html

</script>

* jQuery에서도 Element의 속성은 attr(attribute)와 prop(props) 두 가지 방식으로 제어할 수 있고 JS와 마찬가지로 다른 결과값을 리턴한다.

예 1)

<a id="t1" href="./demo.html">opentutorials</a>

<input id="t2" type="checkbox" checked="checked" />

<script src="//code.jquery.com/jquery-1.11.0.min.js"></script>

<script>

// 현재 문서의 URL이 아래와 같다고 했을 때

// http://localhost/jQuery\_attribute\_api/demo2.html

var t1 = $('#t1');

console.log(t1.attr('href')); // ./demo.html

console.log(t1.prop('href')); // http://localhost/jQuery\_attribute\_api/demo.html

var t2 = $('#t2');

console.log(t2.attr('checked')); // checked

console.log(t2.prop('checked')); // true

</script>

* node에는 HTML 타입 노드 외에도 Text 타입 노드 등 다양한 형의 자식 객체가 포함되어 있다. 이에 따라 nextSibling 프로퍼티를 사용했을 때 의도와 다른 대상을 지정할 수 있다.

<body id="start">

<ul>

<li><a href="./532">html</a></li>

<li><a href="./533">css</a></li>

<li><a href="./534">JavaScript</a>

<ul>

<li><a href="./535">JavaScript Core</a></li>

<li><a href="./536">DOM</a></li>

<li><a href="./537">BOM</a></li>

</ul>

</li>

</ul>

</body>

위 코드에서 id가 start인 body HTML Element를 잡고 nextSibling 프로퍼티를 사용하면 ul Element가 아니라 body와 ul 사이에 있는 공백 문자를 text 타입 노드로 리턴한다. 만일, body와 ul을 ‘<body id="start"><ul>’처럼 붙여 쓸 경우 nextSibling의 리턴값은 ul Element가 된다.

* node를 추가할 때는 node를 추가할 element를 먼저 생성해야 하고 이것은 document 객체의 기능이다.

<ul id="target">

<li>HTML</li>

<li>CSS</li>

</ul>

<script>

function callAppendChild(){

var target = document.getElementById('target');

var li = document.createElement('li'); // 1) 새 element 생성

var text = document.createTextNode('JavaScript'); // 2) 새 text node 생성

li.appendChild(text); // 3) 새 element에 새 text node 추가

target.appendChild(li); // 4\_1) 새 element document에 추가(append: 제일 뒤에)

}

function callInsertBefore(){

var target = document.getElementById('target');

var li = document.createElement('li');

var text = document.createTextNode('jQuery');

li.appendChild(text);

target.insertBefore(li, target.firstChild); // 4\_2) 새 element document에 추가(inserBefore: 지정한 요소 앞에)

}

</script>

* node를 제거할 때는 제거하려는 node의 부모 객체가 가지고 있는 removeNode 메소드를 실행해야 한다.

<ul>

<li>HTML</li>

<li>CSS</li>

<li id="target">JavaScript</li>

</ul>

<script>

function callRemoveChild(){

var target = document.getElementById('target');

target.parentNode.removeChild(target);

}

</script>

* jQuery에서 node를 추가할 때 사용하는 메소드에는



1. before: 타겟 요소 앞
2. prepend: 타겟 요소 하위 > 제일 앞
3. append: 타겟 요소 하위 > 제일 뒤
4. after: 타겟 요소 뒤

가 있는데 메소드에 전달하는 인자로 특정 노드를 선택하면 선택한 노드를 지정한 위치로 이동할 수 있다.

<div class="target" id="target1">

target 1

</div>

<div id="source">source</div>

<input type="button" value="append source to target 1" id="btn1" />

<script src="//code.jquery.com/jquery-1.11.0.min.js"></script>

<script>

$('#btn1').click(function(){

$('#target1').append($('#source')); // id가 source인 요소를 id가 target1인 요소 하위 제일 앞으로 이동

})

</script>

* insertAdjacentHTML(): innerHTML + 위치 지정

1. beforebegin: 타겟 요소 앞
2. afterbegin: 타겟 요소 하위 > 제일 앞
3. beforeend: 타겟 요소 하위 > 제일 뒤
4. afterend: 타겟 요소 뒤

내가 부족하다고 느낀 부분

* JavaScript 기초 지식
* React 기초 지식

수업에 바라는 점