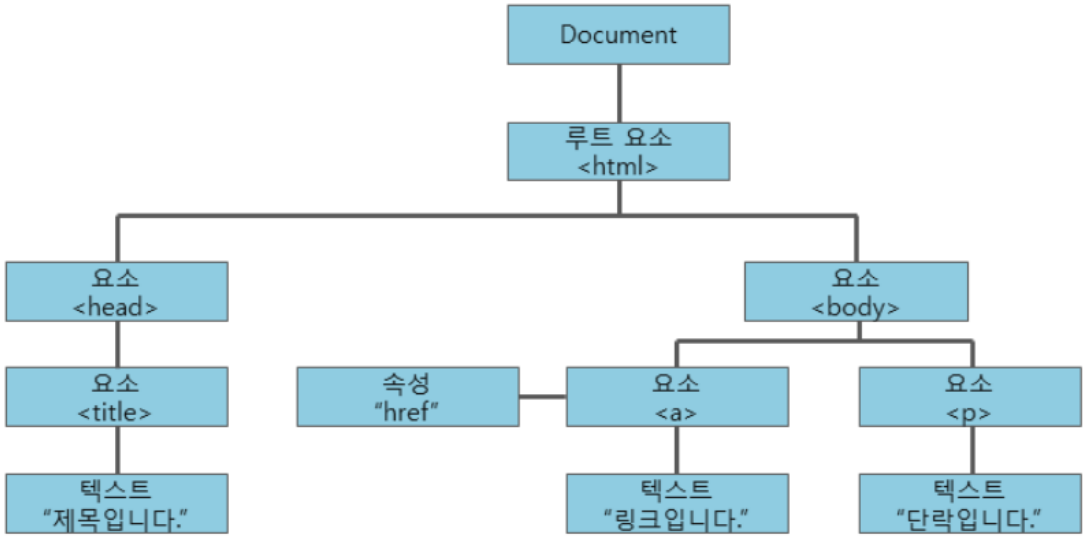
노드

HTML DOM은 노드(node)라고 불리는 계층적 단위에 정보를 저장하고 있습니다.

HTML DOM은 이러한 노드들을 정의하고, 그들 사이의 관계를 설명해 주는 역할을 합니다.

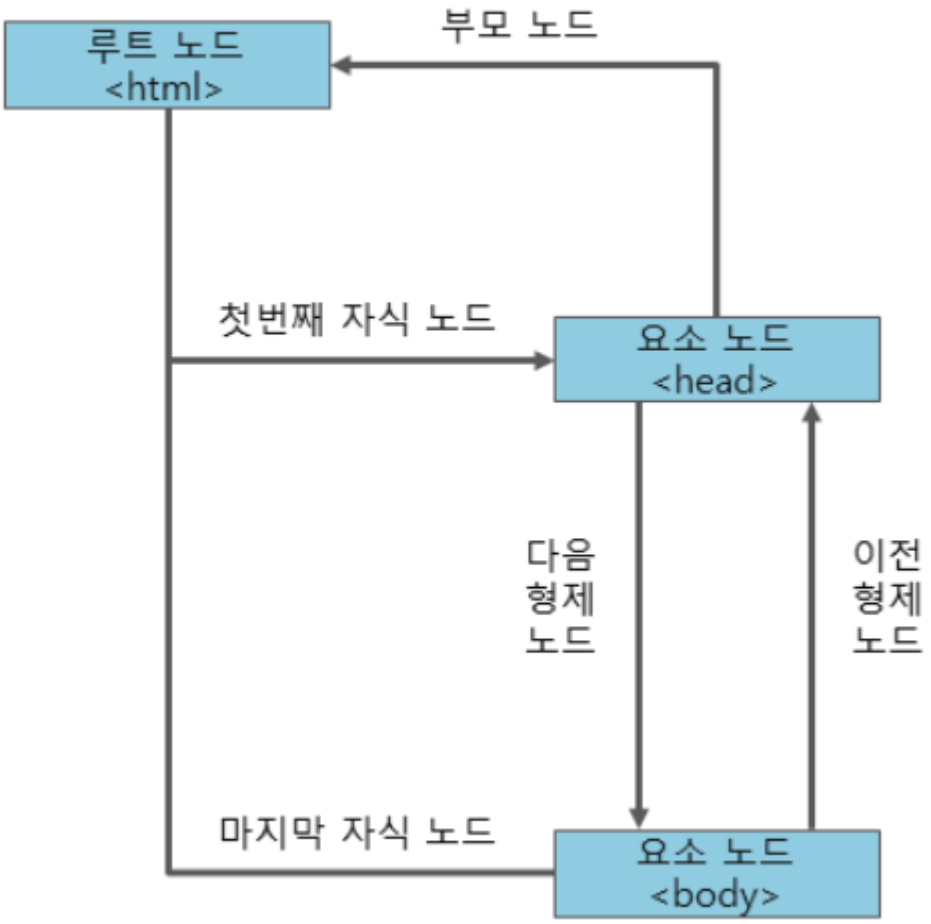


노드의 종류

노드	설명
문서 노드 (document node)	HTML 문서 전체를 나타내는 노드임.
요소 노드 (element node)	모든 HTML 요소는 요소 노드이며, 속성 노드를 가질 수 있는 유일한 노드임.
속성 노드 (attribute node)	모든 HTML 요소의 속성은 속성 노드이며, 요소 노드에 관한 정보를 가지고 있음. 하지만 해당 요소 노드의 자식 노드(child node)에는 포함되지 않음.
텍스트 노드(text node)	HTML 문서의 모든 텍스트는 텍스트 노드임.
주석 노드 (comment node)	HTML 문서의 모든 주석은 주석 노드임.

노드 간의 관계

노드 트리의 모든 노드는 서로 계층적 관계를 맺고 있습니다.



- 노드 트리의 가장 상위에는 단 하나의 루트 노드(root node)가 존재합니다.
- 루트 노드를 제외한 모든 노드는 단 하나의 부모 노드(parent node)만을 가집니다.
- 모든 요소 노드는 자식 노드(child node)를 가질 수 있습니다.
- 형제 노드(sibling node)란 같은 부모 노드를 가지는 모든 노드를 가리킵니다.
- 조상 노드(ancestor node)란 부모 노드를 포함해 계층적으로 현재 노드보다 상위에 존재하는 모든 노드를 가리킵니다.
- 자손 노드(descendant node)란 자식 노드를 포함해 계층적으로 현재 노드보다 하위에 존재하는 모든 노드를 가리킵니다.

노드로의 접근

노드 간의 관계를 이용하여 접근하는 방법

HTML DOM에서 노드 간의 관계는 다음과 같은 프로퍼티로 정의됩니다.

1. **parentNode :** 부모 노드 (부모 노드가 텍스트 노드인 경우는 없다)
2. **childNodes** : 자식 노드를 모두 탐색하여 NodeList에 담아 반환한다.(텍스트 노드 포함)
3. **children** : 자식 노드 중에서 요소 노드만 모두 탐색하여 HTMLCollection에 담아 반환한다.(요소만 반환)
4. **firstChild** : 첫 번째 자식 노드 (텍스트 노드 또는 요소)
5. **firstElementChild** : 첫 번째 자식 노드 (요소만 반환)
6. **lastChild** : 마지막 자식 노드 (텍스트 노드 또는 요소)
7. **lastElementChild** : 마지막 자식 노드 (요소만 반환)
8. **previousSibling** : 이전 형제 노드 (텍스트 노드 또는 요소)
9. **previousElementSibling** : 이전 형제 노드(요소만 반환)
10. **nextSibling** : 다음 형제 노드 (텍스트 노드 또는 요소)
11. **nextElementSibling** : 다음 형제 노드(요소만 반환)

노드에 대한 정보

노드에 대한 정보는 다음과 같은 프로퍼티를 통해 접근할 수 있습니다.

1. nodeName
2. nodeValue
3. nodeType

이 프로퍼티들은 특별히 다른 인터페이스를 이용하지 않고도, 해당 노드의 정보에 직접 접근할 수 있는 방법을 제공합니다.

nodeName

nodeName 프로퍼티는 노드 고유의 이름을 저장하므로, 수정할 수 없는 읽기 전용 프로퍼티입니다.

요소 노드의 nodeName 프로퍼티는 언제나 해당 HTML 요소의 태그 이름을 대문자로 저장합니다

노드	프로퍼티 값
문서 노드(document node)	#document
요소 노드(element node)	태그 이름 (영문자로 대문자)
속성 노드(attribute node)	속성 이름
텍스트 노드(text node)	#text

```
<h1>nodeName 프로퍼티</h1>
<p id="document"></p>
<p id="html"></p>

<script>
// HTML 문서의 모든 자식 노드 중에서 두 번째 노드의 이름을 선택함.
document.getElementById("document").innerHTML = document.ch
ildNodes[1].nodeName;            // HTML

// html 노드의 모든 자식 노드 중에서 첫 번째 노드의 이름을 선택함.
document.getElementById("html").innerHTML = document.childN
odes[1].childNodes[0].nodeName; // HEAD
</script>
```

nodeValue

nodeValue 프로퍼티는 노드의 값을 저장합니다.

노드	프로퍼티 값
요소 노드(element node)	undefined
속성 노드(attribute node)	해당 속성의 속성값

노드	프로퍼티 값
텍스트 노드(text node)	해당 텍스트 문자열

```
<h1 id="heading">nodeValue 프로퍼티</h1>
<p id="text1">텍스트</p>
<p id="text2">텍스트</p>

<script>
// 아이디가 "heading"인 요소의 첫 번째 자식 노드의 노드값을 선택함.
let headingText = document.getElementById("heading").firstC
hild.nodeValue;
 document.getElementById("text1").innerHTML = headingText;
document.getElementById("text1").firstChild.nodeValue = hea
dingText;
 //innerHTML 프로퍼티 대신에 firstChild.nodeValue를 사용해도 같은
결과를 얻을 수 있습니다.
</script>
```

nodeType

nodeType 프로퍼티는 노드 고유의 타입을 저장하므로, 수정할 수 없는 읽기 전용 프로퍼티입니다

노드	프로퍼티 값
요소 노드(element node)	1
속성 노드(attribute node)	2
텍스트 노드(text node)	3
주석 노드(comment node)	8
문서 노드(document node)	9

```
<h1 id="heading">nodeType 프로퍼티</h1>
<p id="head"></p>
<p id="document"></p>
```

```
<script>
// 아이디가 "heading"인 요소의 첫 번째 자식 노드의 타입을 선택함.
var headingType = document.getElementById("heading").firstC
hild.nodeType;
document.getElementById("head").innerHTML = headingType;
// 3
document.getElementById("document").innerHTML = document.no
deType; // 9
</script>
```

노드 리스트(node list)

노드 리스트는 getElementsByTagName() 메소드나 childNodes 프로퍼티의 값으로 반환되는 객체입니다.

이 객체는 HTML 문서와 같은 순서로 문서 내의 모든 노드를 리스트 형태로 저장하고 있습니다.

리스트의 각 노드는 0부터 시작하는 인덱스를 이용하여 접근할 수 있습니다.

```
<h1>노드 리스트</h1>
<ul id="list">
    <li>첫 번째 아이템이에요!</li>
    <li>두 번째 아이템이에요!</li>
    <li>세 번째 아이템이에요!</li>
</ul>

<script>
// 아이디가 "list"인 요소의 모든 자식 노드들을 선택함.
const listItems = document.getElementById("list").childNode
s;
// 자식 노드들 중 첫 번째 li 요소의 내용을 변경함.
listItems[1].firstChild.nodeValue = "HTML 요소의 내용을 변경했
어요!";
</script>
```

1. 첫 번째 노드에는 아이디가 "list"인 요소 다음에 존재하는 텍스트 노드가 저장됩니다.

2. 그래서 첫 번째 요소를 선택할 때 인덱스로 0이 아닌 1을 사용합니다.

---

textContent

요소 노드의 텍스트와 모든 자손 노드의 텍스트를 모두 취득하거나 변경한다.

```
<!-- 요소 노드의 콘텐츠 영역에 다른 요소 노드가 없고 텍스트만 존재 -->
<div id="foo">Hello</div>

<script>
const $foo = document.getElementById('foo');

// #foo 요소 노드의 모든 자식 노드가 제거되고 할당한 문자열이 텍스트로 추가된다.
// 이때 HTML 마크업이 파싱되지 않는다.
$foo.textContent = 'Hi <span>there!</span>';
</script>
```

innerHTML

요소 노드의 HTML 마크업을 취득하거나 변경한다.

```
<div id="bar">Hello <span>world!</span></div>

<script>
// #foo 요소의 콘텐츠 영역 내의 HTML 마크업을 문자열로 취득한다.
console.log(document.getElementById('bar').innerHTML);
// "Hello <span>world!</span>"

// HTML 마크업이 파싱되어 요소 노드의 자식 노드로 DOM에 반영된다.
 document.getElementById('bar').innerHTML = 'Hi <span>there!</span>';
</script>
```

노드의 생성 및 추가

appendChild() 메소드

`appendChild()` 메소드는 한 노드를 특정 부모 노드의 자식 노드 리스트 중 마지막 자식으로 붙입니다

```
<ul id="fruits">
  <li>Apple</li>
</ul>
<script>
const $fruits = document.getElementById('fruits');
// 1. 요소 노드 생성
const $li = document.createElement('li');
// 2. 텍스트 노드 생성
const textNode = document.createTextNode('Banana');
// 3. 텍스트 노드를 $li 요소 노드의 자식 노드로 추가
$li.appendChild(textNode);
// 4. $li 요소 노드를 #fruits 요소 노드의 마지막 자식 노드로 추가
$fruits.appendChild($li);
</script>
```

복수의 노드 생성과 추가

- createDocumentFragment 노드를 통해 DOM을 한번만 변경하여 복수노드를 추가하는 것이 가능하다

```
<script>
const $fruits = document.getElementById('fruits');

// DocumentFragment 노드 생성
const $fragment = document.createDocumentFragment();

['Apple', 'Banana', 'Orange'].forEach(text => {
  // 1. 요소 노드 생성
  const $li = document.createElement('li');
  // 2. 텍스트 노드 생성
```

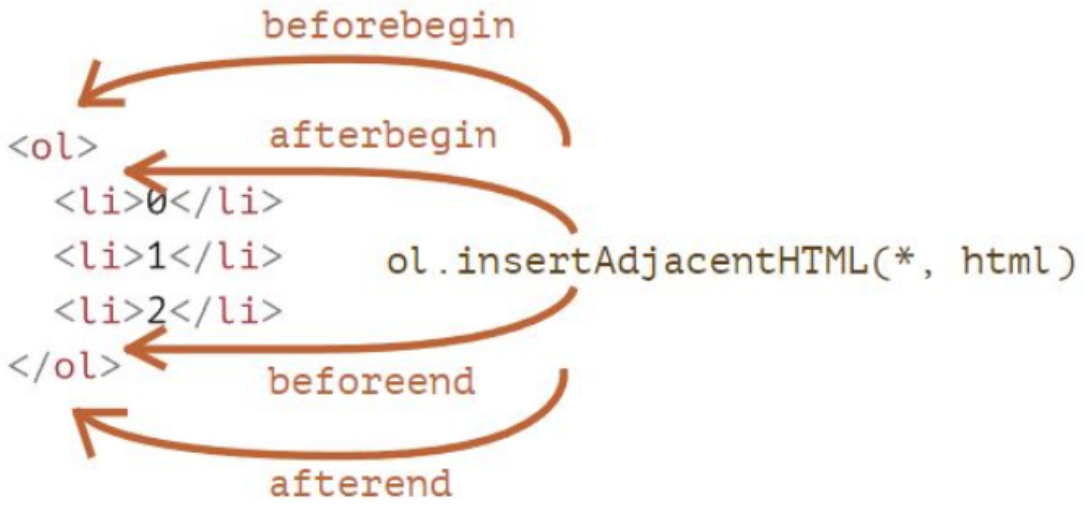
```
  const textNode = document.createTextNode(text);
  // 3. 텍스트 노드를 $li 요소 노드의 자식 노드로 추가
  $li.appendChild(textNode);
  // 4. $li 요소 노드를 DocumentFragment 노드의 마지막 자식 노드로 추가
  $fragment.appendChild($li);
});
// 5. DocumentFragment 노드를 #fruits 요소 노드의 마지막 자식 노드로 추가
$fruits.appendChild($fragment);
</script>
```

---

insertAdjacentHTML

기존 요소를 제거하지 않으면서 위치를 지정해 새로운 요소를 삽입한다.

첫 번째 인수로 전달할 수 있는 문자열은 총 4가지다.




```
<!-- beforebegin -->
<div id="foo">
  <!-- afterbegin -->
  text
```

```
  <!-- beforeend -->
</div>
<!-- afterend -->

<script>
const $foo = document.getElementById('foo');

$foo.insertAdjacentHTML('beforebegin', '<p>beforebegin</p
>');
$foo.insertAdjacentHTML('afterbegin', '<p>afterbegin</p>');
$foo.insertAdjacentHTML('beforeend', '<p>beforeend</p>');
$foo.insertAdjacentHTML('afterend', '<p>afterend</p>');
</script>
```

insertBefore() 메소드

insertBefore() 메소드는 새로운 노드를 특정 자식 노드 바로 앞에 추가합니다.

insertBefore() 메소드의 원형은 다음과 같습니다.

부모노드.insertBefore(새로운자식노드, 기준자식노드);

1. 새로운 자식 노드 : 자식 노드 리스트(child node list)에 새롭게 추가할 자식 노드를 전달합니다.

2. 기준 자식 노드 : 새로운 노드를 삽입할 때 기준이 되는 노드로, 이 노드 바로 앞에 새로운 노드가 추가됩니다.

```
// 요소 노드 생성
const $li2 = document.createElement('li');
// 텍스트 노드를 $li 요소 노드의 마지막 자식 노드로 추가
$li2.appendChild(document.createTextNode('pineapple'));
// $li 요소 노드를 #fruits 요소 노드의 마지막 자식 요소 앞에 삽입
$fruits.insertBefore($li2, $fruits.lastElementChild);
```

노드 이동

이미 존재하는 노드를 appendChild 또는 insertBefore 메서드를 사용하여 DOM에 다시 추가하면 현재 위치에서 노드를 제거하고 새로운 위치에 노드를 추가한다.(노드 이동)

```
// 이미 존재하는 요소 노드를 취득
const [$apple, $banana, ] = $fruits.children;

// 이미 존재하는 $apple 요소 노드를 #fruits 요소 노드의 마지막 노드로 이동
$fruits.appendChild($apple); // Banana - Orange - Apple

// 이미 존재하는 $banana 요소 노드를 #fruits 요소의 마지막 자식 노드 앞으로 이동
$fruits.insertBefore($banana, $fruits.lastElementChild);
```

파일 새로 생성

cloneNode() 메소드

cloneNode() 메소드는 기존의 존재하는 노드와 똑같은 새로운 노드를 생성하여 반환합니다.

cloneNode() 메소드의 원형은 다음과 같습니다.

복제할노드.cloneNode(자식 노드 복제 여부);

자식 노드 복제 여부 : 전달된 값이 true이면 복제되는 노드의 모든 속성 노드와 자식 노드도 같이 복제하며, false이면 속성 노드만 복제하고 자식 노드는 복제하지 않습니다. (텍스트 노드도 없다)

```
<h1>cloneNode() 메소드</h1>
<button onclick="cloneElement()">노드 복제!</button>
<h2 id="item">JavaScript</h2>
<div id="list">
    <p>HTML</p>
    <p>CSS</p>
    <p>JQuery</p>
```

```
</div>

<script>
function cloneElement() {
    var parent = document.getElementById("list");
// 아이디가 "list"인 요소를 선택함.
    var originItem = document.getElementById("item");
// 아이디가 "item"인 요소를 선택함.
    parent.appendChild(originItem.cloneNode(true)); // 해당
노드를 복제하여 리스트의 맨 마지막에 추가함.
}
</script>
```

요소 노드의 교체 (replaceChild)

replaceChild() 메소드를 사용하면 기존의 요소 노드를 새로운 요소 노드로 교체할 수 있습니다.

replaceChild() 메소드의 원형은 다음과 같습니다.

교체할노드 = 부모노드.replaceChild(새로운자식노드, 기존자식노드);

1. 새로운 자식 노드 : 자식 노드 리스트에 새롭게 추가할 요소 노드를 전달합니다.

2. 기존 자식 노드 : 자식 노드 리스트에서 제거할 요소 노드를 전달합니다.

예제는 자식 노드 중에서 첫 번째 요소를 삭제하고, 그 대신에 세 번째 요소를 첫 번째 요소 자리에 삽입하는 예제입니다.

```
<h1>요소 노드의 교체</h1>
<div id="parent">
    <p id="first">첫 번째 요소입니다.</p>
    <p>두 번째 요소입니다.</p>
</div>
<p id="third">세 번째 요소입니다.</p>
<button onclick="changeNode()">요소 노드 교체!</button>
```

```
<script>
var parent = document.getElementById("parent"); // 부모 노드를 선택함.
var first = document.getElementById("first");
var third = document.getElementById("third");
function changeNode() {
    parent.replaceChild(third, first);          // first 요소를 삭제하고, 그 대신 third 요소를 삽입함.
}
</script>
```

노드의 제거

removeChild() 메소드

removeChild() 메소드는 자식 노드 리스트에서 특정 자식 노드를 제거합니다.

노드가 제거될 때에는 제거되는 노드의 모든 자식 노드들도 다 같이 제거됩니다

```
<h1>removeChild() 메소드</h1>
<button onclick="remove()">요소 노드 삭제!</button>
<div id="list">
    <p>HTML</p>
    <p id="item">CSS</p>
    <p>JavaScript</p>
</div>

<script>
function remove() {
    var parent = document.getElementById("list");  // 아이디가 "list"인 요소를 선택함.
    var removedItem = document.getElementById("item");
// 아이디가 "item"인 요소를 선택함.
    parent.removeChild(removedItem);   // 지정된 요소를 삭제함.
}
</script>
```