07. DOM 활용하기

DOM 트리와 노드 리스트

DOM 트리와 노드

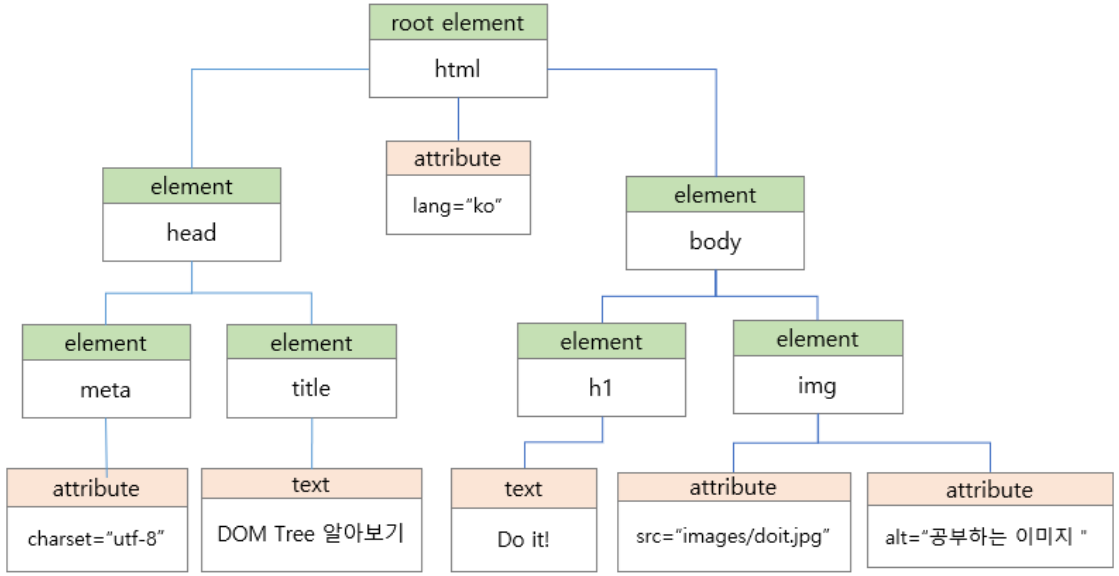
DOM에는 단순히 태그에 해당하는 요소 노드뿐만 아니라 여러 종류의 노드가 있다.
DOM 트리에서 가지가 갈라져 나가는 부분은 노드라고 하고,
DOM 트리의 시작 부분, 즉 html 노드를 나무의 뿌리에 해당하는 루트 노드라고 한다.

노드를 구성하는 원칙

- 모든 HTML 태그는 **요소 노드**가 된다.
- HTML 태그에서 사용하는 텍스트 내용은 자식 노드인 **텍스트 노드**가 된다.
- HTML 태그에 있는 속성은 모두 자식 노드인 **속성 노드**가 된다.
- 주석들은 **주석 노드**가 된다.

DOM 트리와 노드

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset=“utf-8”>
  <title>DOM Tree 알아보기</title>
</head>
<body>
  <h1>Do it!</h1>
  <img src="images/doit.jpg"  alt="공부하는 이미지">
</body>
</html>
```



노드 리스트

- querySelectorAll() 메서드를 사용하면 여러 개의 노드를 한꺼번에 가져올 수 있다.
- 가져온 여러 개의 노드 정보를 저장한 것을 노드 리스트라고 한다.
- 노드 리스트는 배열과 비슷하게 생겼고 배열처럼 사용할 수 있다. (배열은 아님)

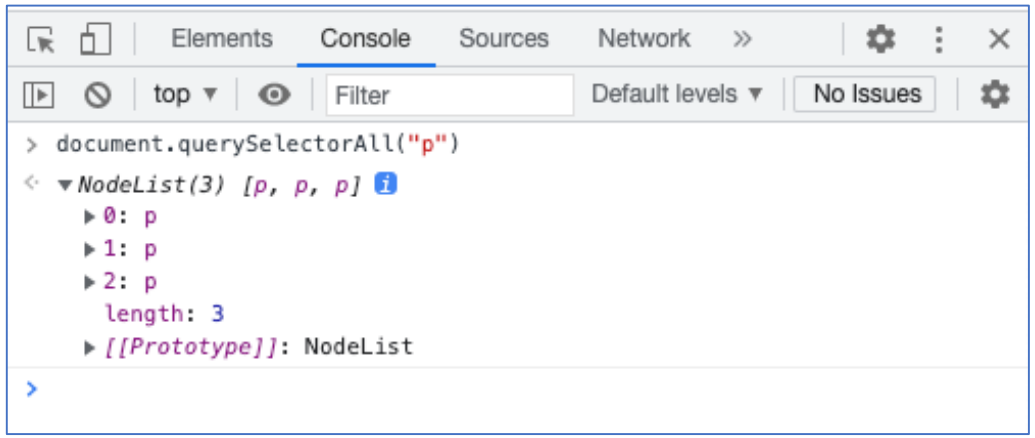
노드 리스트

07₩nodelist.html

```
<h1>노드 리스트 살펴보기</h1>
<p>HTML</p>
<p>CSS</p>
<p>Javscript</p>
```

콘솔 창에 입력하기

```
document.querySelectorAll("p
")
```



p 요소 노드들을 저장한 노드 리스트 중에서

두 번째 노드를 가져오려면?

```
document.querySelectorAll("p")[1]
```

새로운 노드 추가하기

웹 문서에 새로운 노드 추가하기

웹 문서에서 처음에는 화면에 보이지 않다가 이벤트가 발생했을 때 화면에 특정한 내용이 표시되도록 하려면?

→ 특정 이벤트가 발생했을 때 DOM 트리에 새로운 노드를 추가한다.

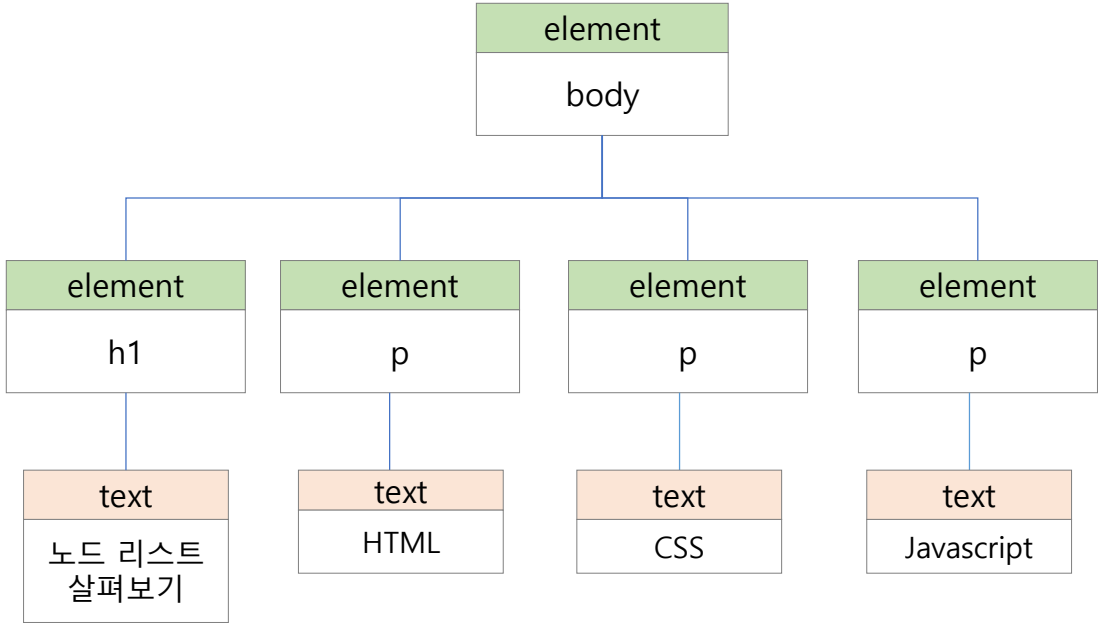
텍스트 내용이 있는 노드 추가하기

새로운 추가할 노드에 텍스트 내용만 있다면

1) 텍스트 노드를 만들고 (텍스트 내용이 있는 노드)
2) 요소 노드를 만들어서 (예를 들어, p 요소 노드)
3) 텍스트 노드를 요소 노드에 연결한 후 (p 요소 노드에 텍스트 내용 연결)
4) 원하는 영역에 추가합니다.

웹 브라우저에 07₩nodelist.html 문서를 열고, 콘솔 창에서 새로운 p 요소 추가하기

```
<body>
   <h1>노드 리스트 살펴보기</h1>
   <p>HTML</p>
   <p>CSS</p>
   <p>Javascript</p>
</body>
```



노드 리스트 살펴보기

HTML

CSS

Javascript

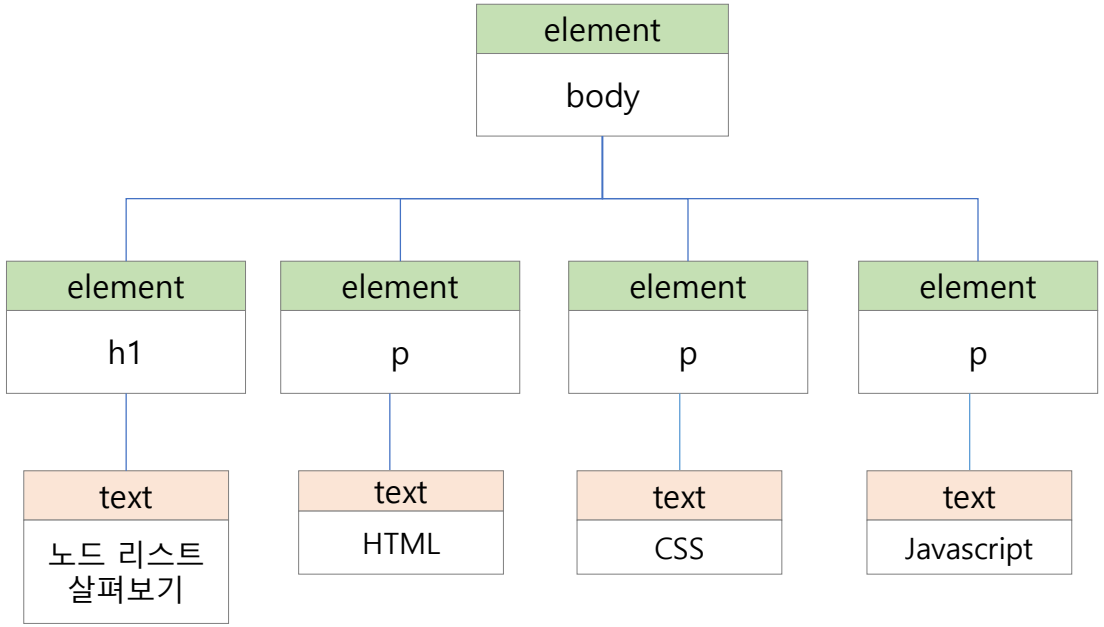
요소 노드 만들기 – createElement()

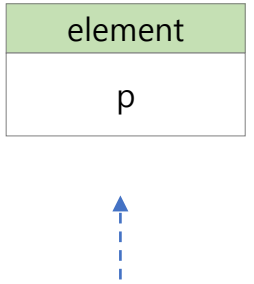
DOM에 새로운 요소를 추가할 때 가장 먼저 요소 노드를 만들어야 한다.

```
document.createElement(요소명)
```

콘솔 창에 입력하기

```
let newP = document.createElement("p")
```





createElement()는 새로운 노드를 만들 뿐, <p> 태그의 내용에 해당하는 텍스트 노드도 만들자.

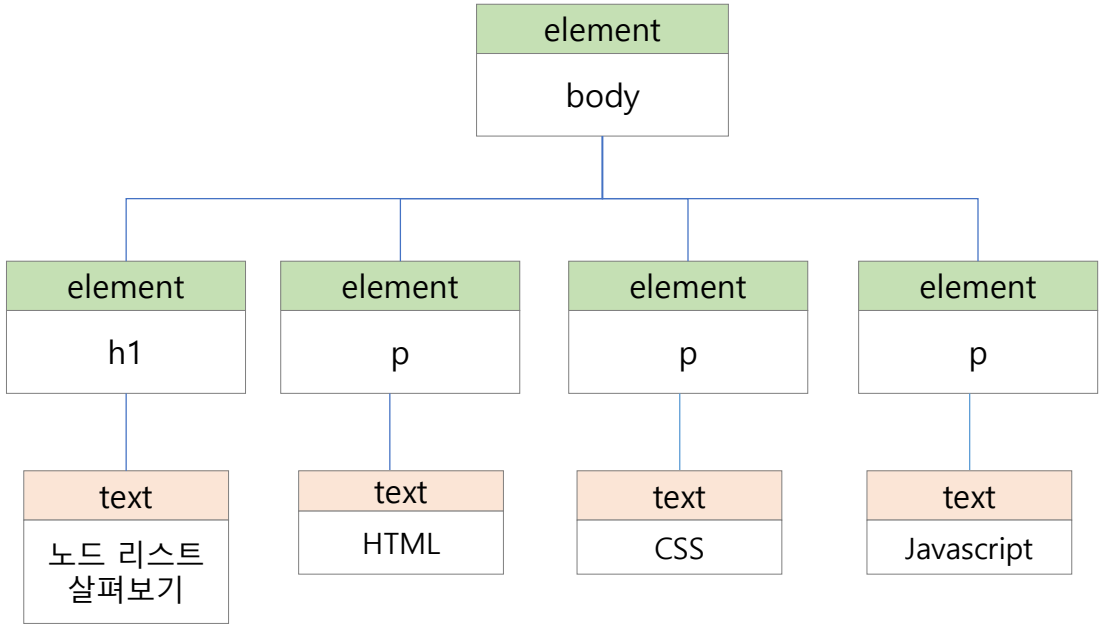
텍스트 노드 만들기 – createTextNode()

텍스트 노드를 만드는 메서드는 createTextNode()

```
document.createTextNode(텍스트)
```

새로운 p 요소에 들어갈 내용을 텍스트 노드로 만들고 textNode라는 변수에 저장

```
let textNode = document.createTextNode("Typescript")
```



element

p

text

Typescript

자식 노드 연결하기 – appendChild()

아직까지는 p 노드와 텍스트 노드가 따로 만들어진 상태

→ 서로 부모 노드와 자식 노드로 연결해야 한다.

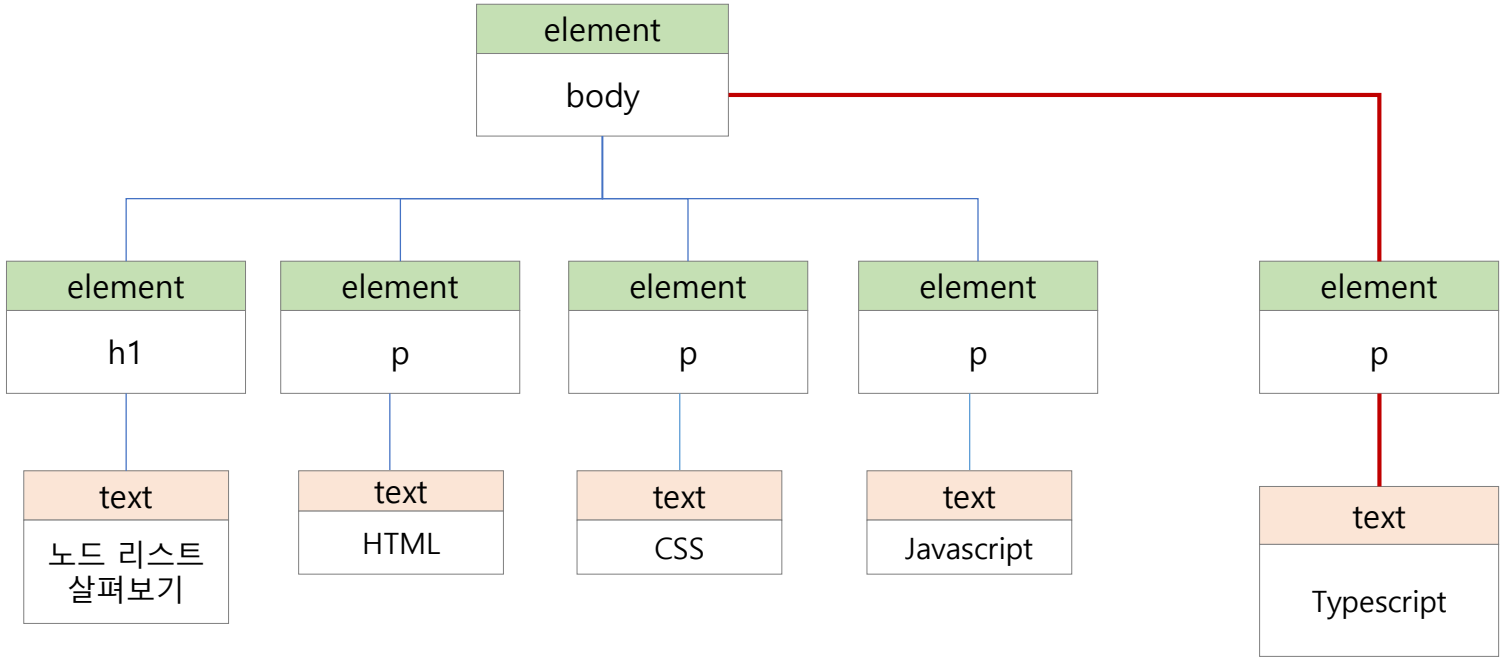
```
부모_노드.appendChild(자식_노드)
```

콘솔 창에 입력하기

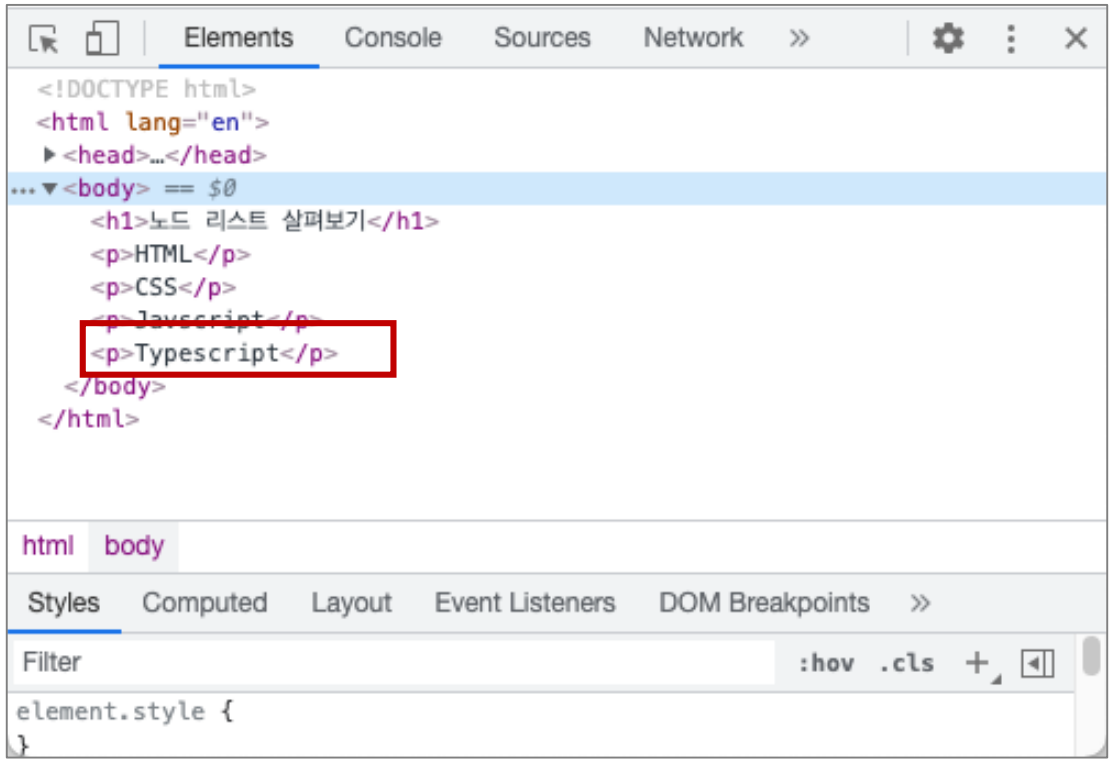
```
newP.appendChild(textNode)
document.body.appendChild(newP)
```

텍스트 노드를 p 요소에 연결

p 요소 노드를 body 노드에 연결

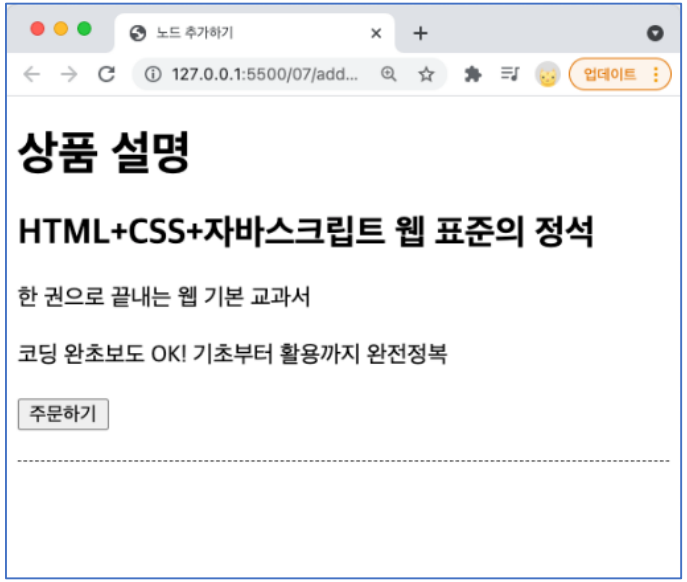

element
body
element
h1
element
p
element
p
element
p
element
p
text
노드 리스트 살펴보기
text
HTML
text
CSS
text
Javascript
text
Typescript

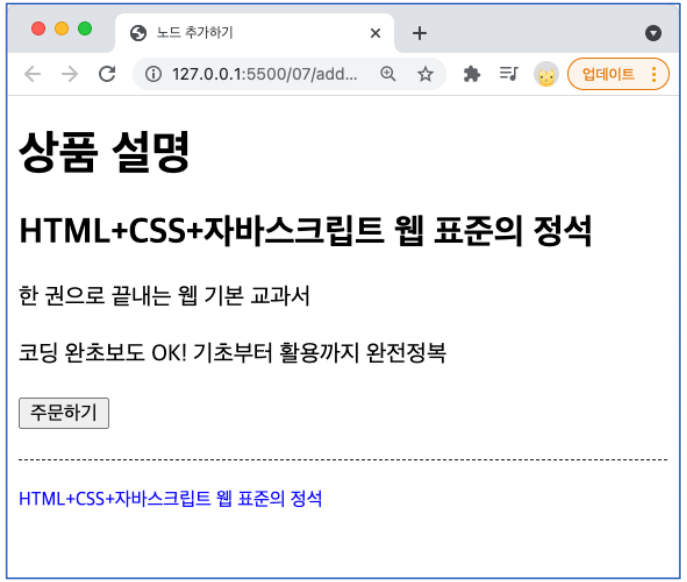

노드 리스트
파일 /Users/funnycom/Deskt...
노드 리스트 살펴보기
HTML
CSS
Javscript
Typescript


Elements
Console
Sources
Network
<!DOCTYPE html>
<html lang="en">
<head>…</head>
<body> == $0
<h1>노드 리스트 살펴보기</h1>
HTML
CSS
Javscript
Typescript
</body>
</html>
html
body
Styles
Computed
Layout
Event Listeners
DOM Breakpoints
Filter
:hov .cls
element.style {
}

[실습] 장바구니에 상품 추가하기

책 소개 화면에서 [주문하기] 버튼을 클릭하면 그 아래 장바구니 영역에 책 제목 텍스트를 추가하기





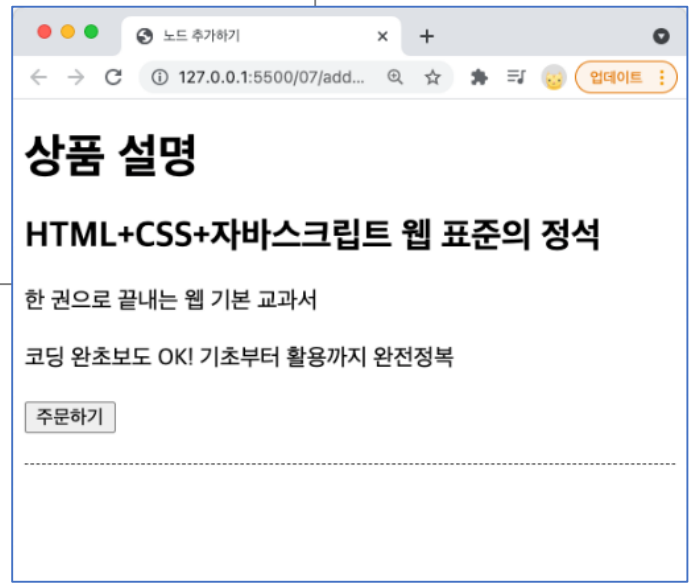
<생각해 보기>

- 클릭하는 버튼과 결과를 표시할 영역을 어떤 방식으로 가져올까?
- 새로 만든 텍스트를 화면에 어떻게 연결할까?

07\addText.html, 07\js\addText.js

```
<div id="container">
  <h1>상품 설명</h1>
  <h2>HTML+CSS+자바스크립트 웹 표준의 정석</h2>
  <p>한 권으로 끝내는 웹 기본 교과서 </p>
  <p>코딩 완초보도 OK! 기초부터 활용까지 완전정복</p>
  <button id="order">주문하기</button>
</div>
<div id="orderInfo"></div>

<script src="js/addText.js"></script>
```



2) 버튼과 주문 정보 영역을 먼저 가져와서 변수로 저장

3) 버튼에 이벤트 리스너를 연결

07\js\addText.js

```
const orderButton = document.querySelector("#order");       // [주문하기] 버튼
const orderInfo = document.querySelector("#orderInfo");     // 주문 정보 영역


orderButton.addEventListener("click", () => {


});
```

4) 장바구니 영역에 책 제목 텍스트를 추가할 것이므로 p 요소 노드를 만들고

5) 책 제목을 내용으로 하는 텍스트 노드를 만든 후 p 요소 노드에 연결

6) 이 소스를 이벤트 리스너에 추가.

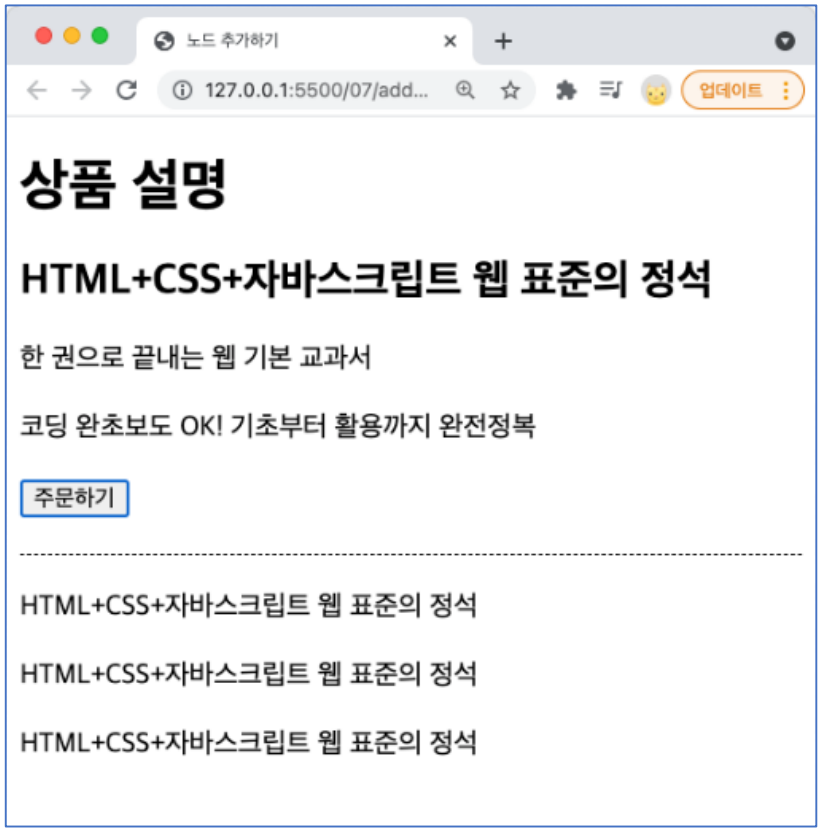
07\js\addText.js

```
orderButton.addEventListener("click", () => {
  let newP = document.createElement("p");      // 새로운 p 요소를 만듭니다.
  let textNode = document.createTextNode(title.innerText);  // 텍스트 노드를 만듭니다.
  newP.appendChild(textNode);     //텍스트 노드를 p 요소에 추가합니다.
  orderInfo.appendChild(newP);    //p 요소를 orderInfo에 추가합니다.
});
```

7) 라이브 서버를 사용해 결과 확인하기

[주문하기] 버튼을 클릭할 때마다 계속 이벤트 리스너가 동작하기 때문에 계속 텍스트 단락이 만들어진다.

→ 한번만 표시하자!



8) 이벤트 리스너가 한번만 동작하고 멈추려면 ‘once: true’ 속성을 추가해야 한다.

9) 수정하는 김에 글자색과 글자 크기 스타일도 지정해 보자.

07\js\addText.js

```
orderButton.addEventListener("click", () => {
  let newP = document.createElement("p");
  let textNode = document.createTextNode(title.innerText);
  newP.appendChild(textNode);
  newP.style.fontSize = "0.8em";
  newP.style.color = "blue";
  orderInfo.appendChild(newP);
}, { once : true });
```

속성 값이 있는 노드 추가하기

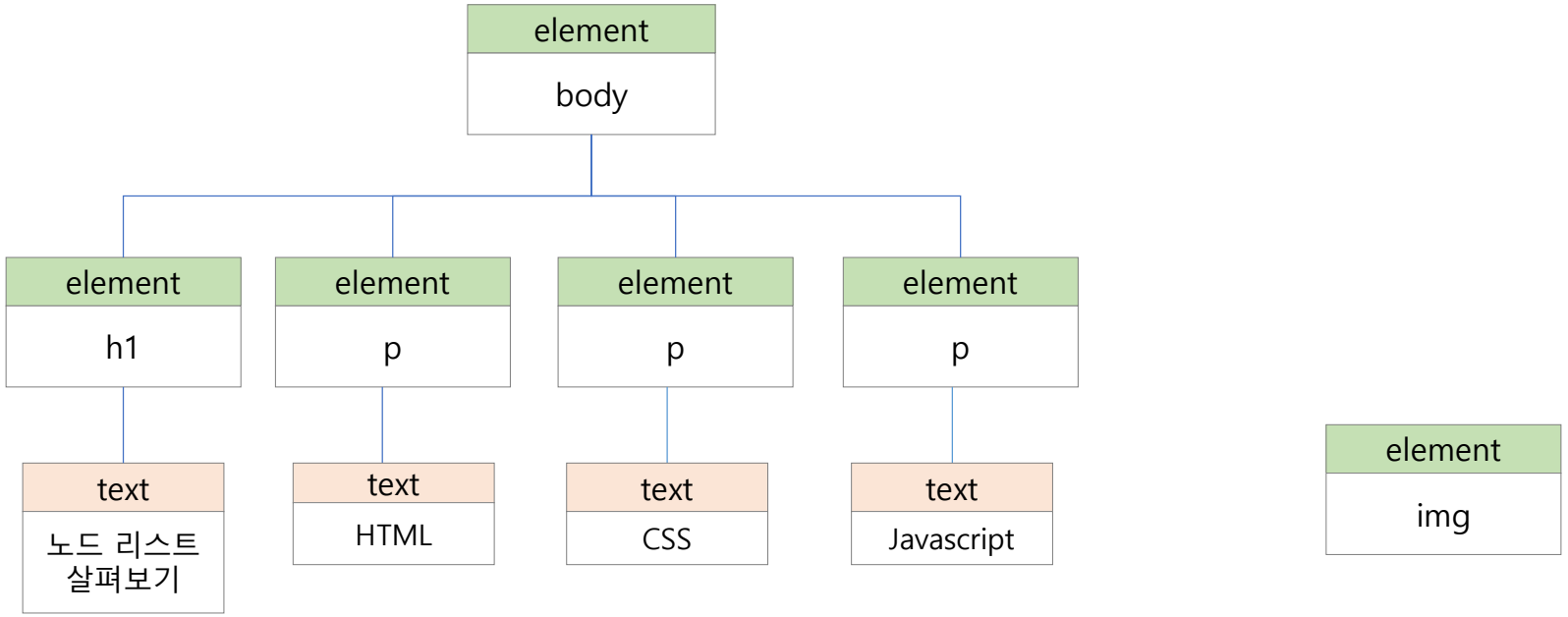
- HTML 태그에서는 여러 가지 속성을 사용해서 웹 요소를 제어한다.
- 속성이 필요한 요소를 추가할 때는 속성 노드도 함께 만들어서 자식 노드로 연결해야 한다.

요소 노드 만들기 – createElement()

웹 브라우저에 07₩nodelist.html 문서를 열고, 콘솔 창에서 새로운 이미지 추가하기

createElement() 메서드를 사용해서 새로운 이미지 노드를 만들기

```
let newImg = document.createElement("img")
```



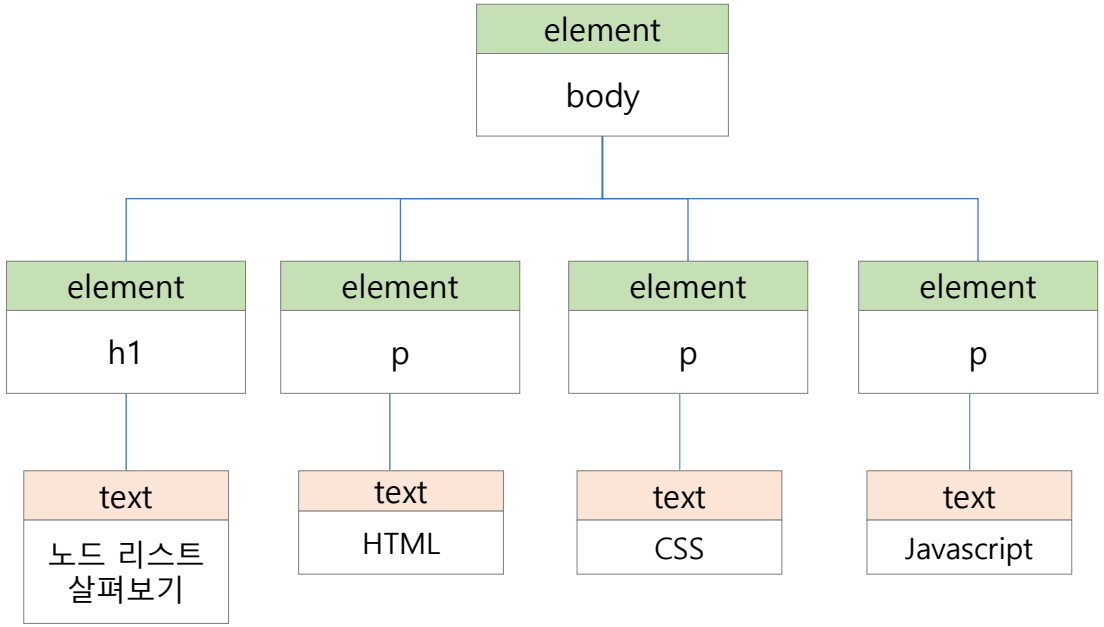
속성 노드 만들기 – createAttribute()

1) createAttribute() 메서드를 사용해서 속성 노드를 만들고,
2) 속성의 값은 value 프로퍼티를 사용해서 지정한다.

document.createAttribute(*속성명*)
노드명.value = *속성값*

콘솔 창에 입력하기

```
let srcNode = document.createAttribute("src")
srcNode.value = "images/wall.jpg"
```



element

img

attribute

src="images/wall.jpg"

속성 노드 연결하기 – setAttributeNode()

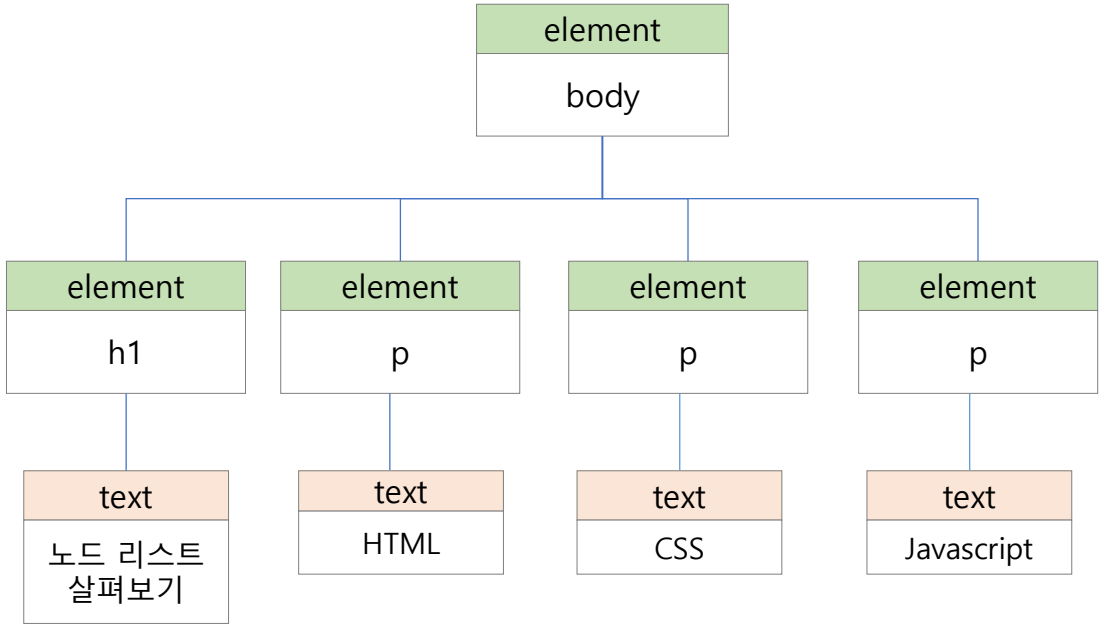
새로 만든 속성 노드를 요소 노드에 추가하려면 setAttributeNode() 메서드 사용

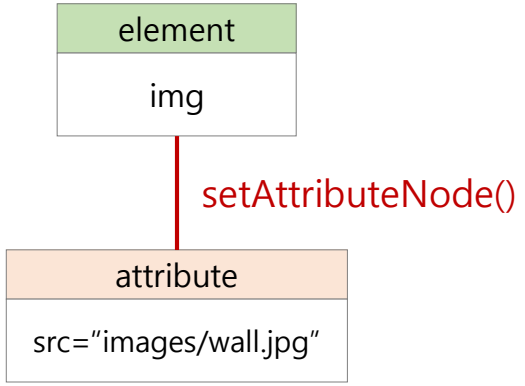
setAttribute()와 다른 메서드이므로 꼭 구별해서 사용하세요

```
요소 노드.setAttributeNode(속성 노드)
```

콘솔 창에 입력하기

```
newImg.setAttributeNode(srcNode)
```


element
body
element
h1
element
p
element
p
element
p
text
노드 리스트 살펴보기
text
HTML
text
CSS
text
Javascript


element
img
setAttributeNode()
attribute
src="images/wall.jpg"

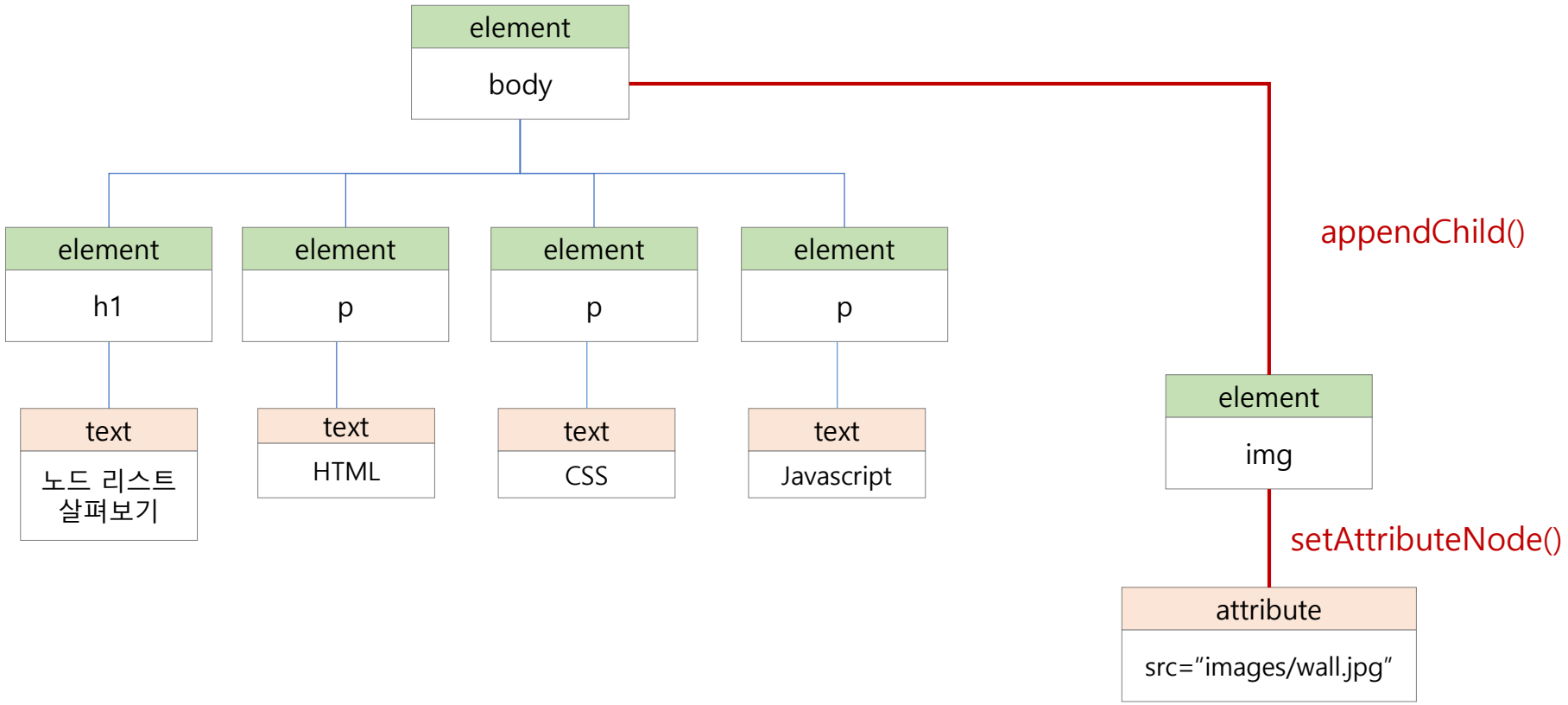
자식 노드 연결하기 – appendChild()

img 요소에 속성 노드가 연결되었지만, img 요소는 아직 DOM에 연결되지 않은 상태

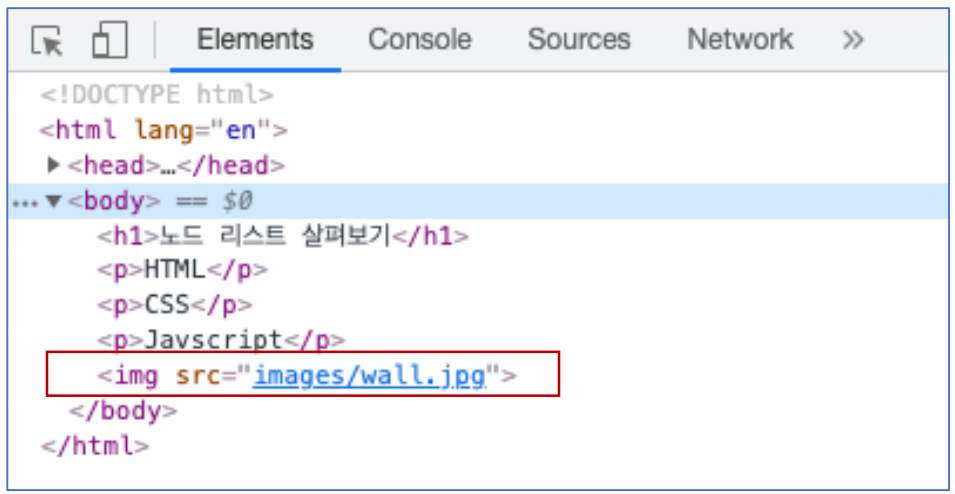
appendChild()를 사용해서 img 요소를 원하는 위치에 연결한다.

콘솔 창에 입력하기

```
document.body.appendChild(newImg)
```


element
body
element
h1
element
p
element
p
element
p
text
노드 리스트 살펴보기
text
HTML
text
CSS
text
Javascript
appendChild()
element
img
setAttributeNode()
attribute
src="images/wall.jpg"


노드 리스트
파일 | /Users/funnycom/Desk...
노드 리스트 살펴보기
HTML
CSS
Javscript


Elements
Console
Sources
Network
<!DOCTYPE html>
<html lang="en">
<head>…</head>
<body> == $0
<h1>노드 리스트 살펴보기</h1>
HTML
CSS
Javscript
<img src="images/wall.jpg">
</body>
</html>

기존 노드 앞에 추가하기 – insertBefore()

지금까지 살펴본 방법은, 새로 만든 요소를 부모 노드의 맨 마지막에 자식 노드로 추가하는 것.
insertBefore() 를 사용하면 특정 노드 앞에 새 요소를 추가할 수 있다.

```
insertBefore(새 노드, 기준 노드)
```

07₩nodelist.html

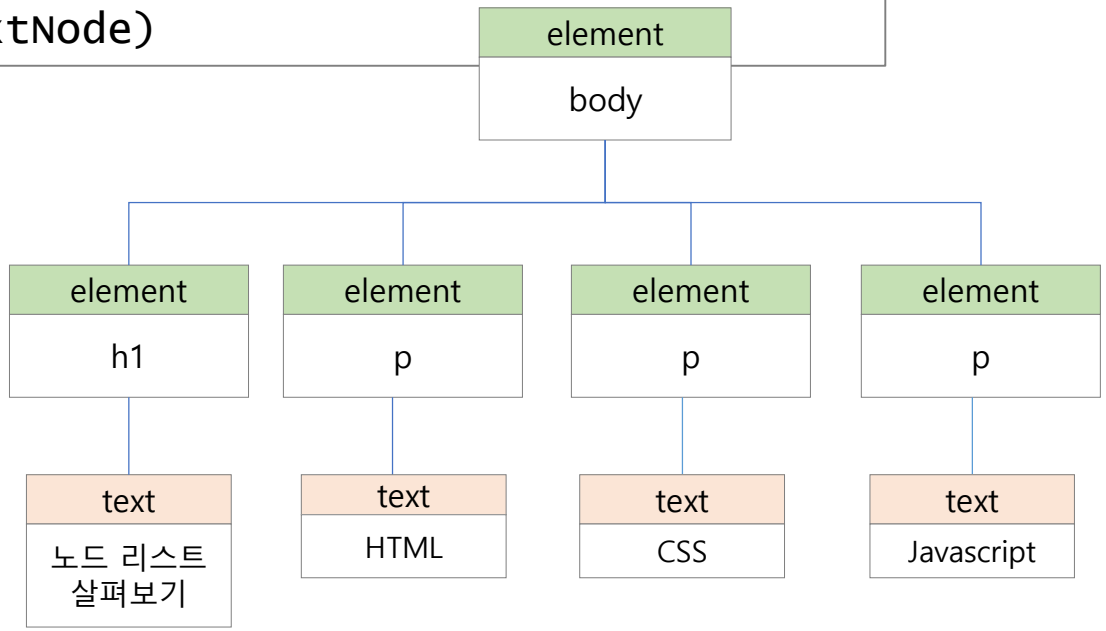
```
<h1>노드 리스트 살펴보기</h1>
<p>HTML</p>
<p>CSS</p>
<p>Javscript</p>
```

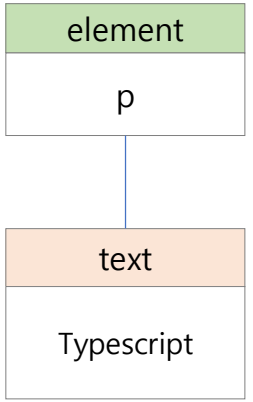
여기에 새로운 p 요소 추가하기

기존 노드 앞에 추가하기 – insertBefore()

콘솔 창에 입력하기

```
let tsNode = document.createElement("p")
let tsTextNode = document.createTextNode("Typescript")
tsNode.appendChild(tsTextNode)
```

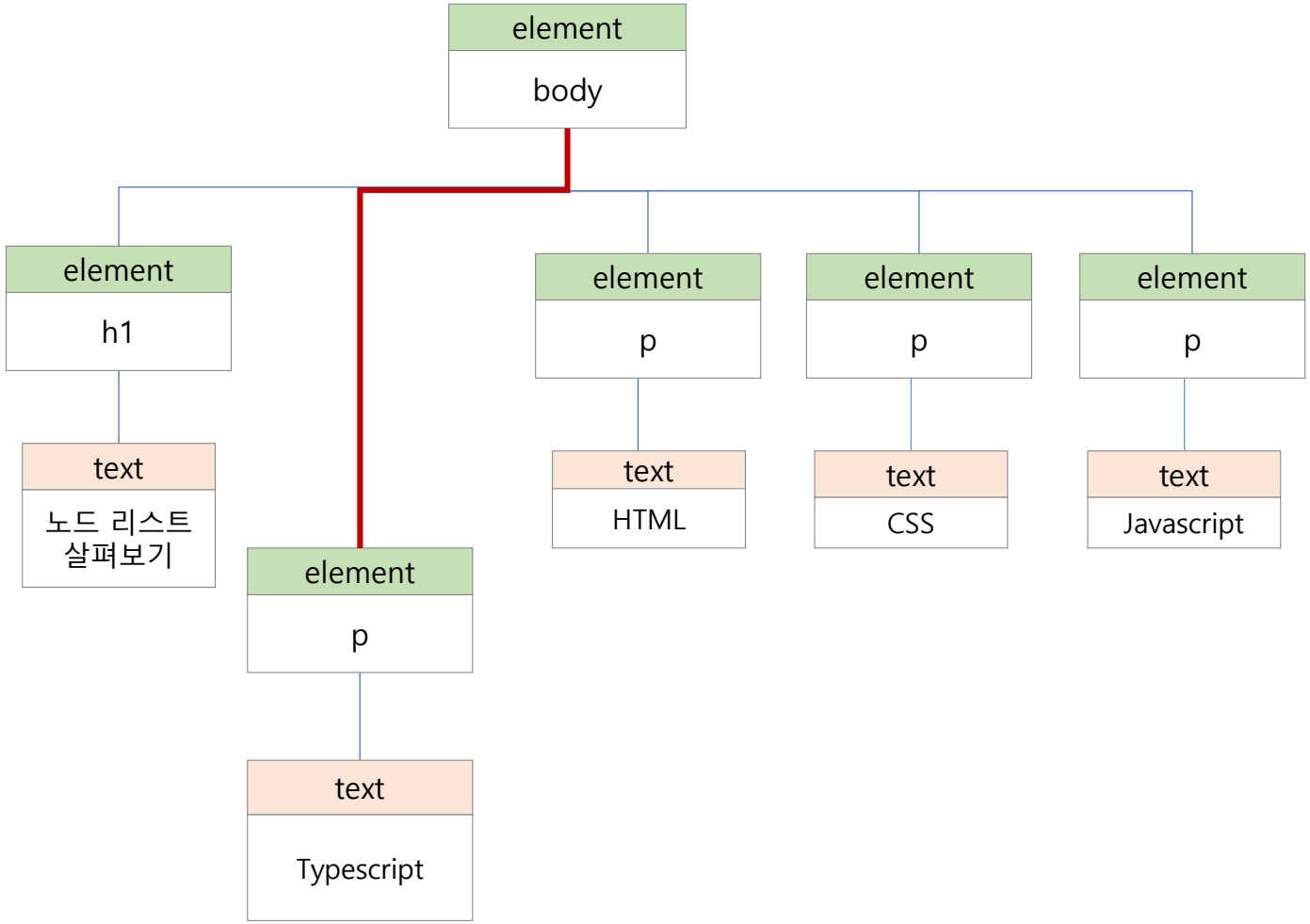


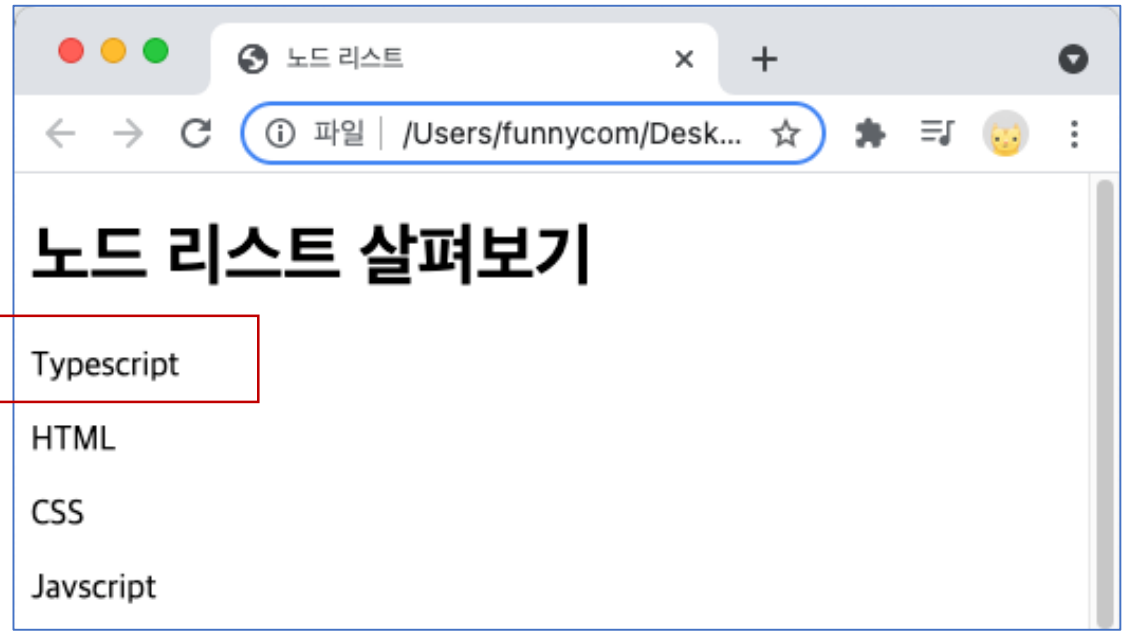


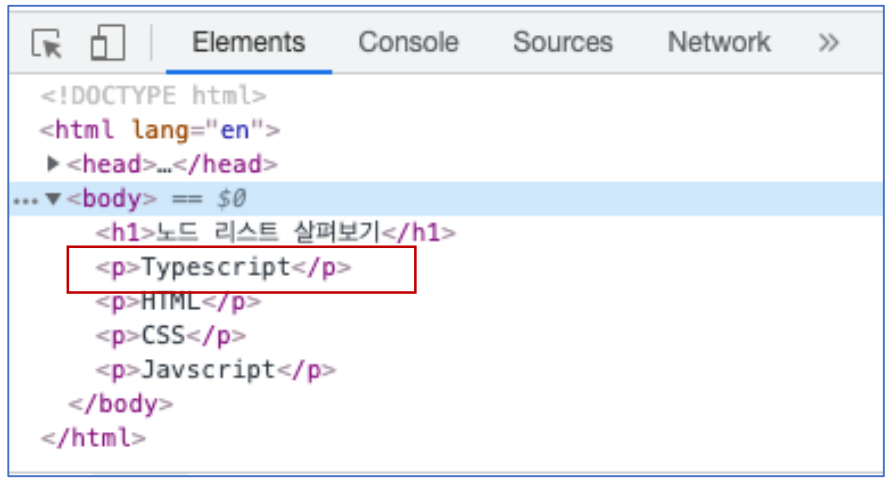
기존 노드 앞에 추가하기 – insertBefore()

1) 기준이 되는 노드를 첫번째 p 노드로 지정하고
2) insertBefore()를 사용해 첫번째 p 노드 앞에 새 노드를 추가한다.

```
let basisNode = document.querySelectorAll("p")[0]   // 첫 번째 p 요소를 기준 노드로 정하기
document.body.insertBefore(tsNode, basisNode)      // 기준 노드 앞에 tsNode 추가하기
```


element
body
element
h1
text
노드 리스트
살펴보기
element
p
text
Typescript
element
p
text
HTML
element
p
text
CSS
element
p
text
Javascript


노드 리스트
파일 /Users/funnycom/Desk...
노드 리스트 살펴보기
Typescript
HTML
CSS
Javscript


Elements
Console
Sources
Network
<!DOCTYPE html>
<html lang="en">
<head>…</head>
<body> == $0
<h1>노드 리스트 살펴보기</h1>
Typescript
HTML
CSS
Javscript
</body>
</html>

노드 삭제하기

노드 삭제하기 – remove()

remove()는 삭제하려는 요소에서 사용하는 메서드

```
요소.remove()
```

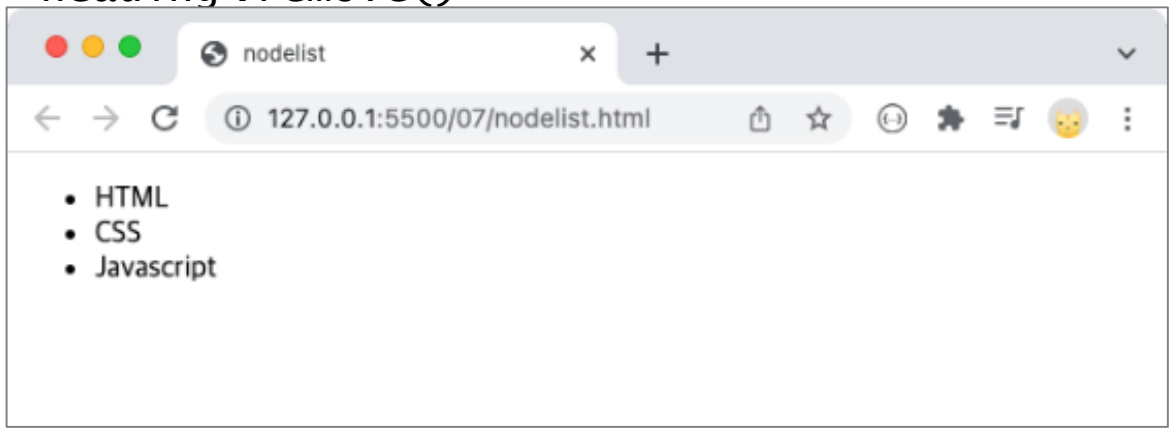
07₩nodelist-2.html

```
<h1>Web Programming</h1>
<ul id="items">
  <li>HTML</li>
  <li>CSS</li>
  <li>Javascript</li>
</ul>
```

이 부분을 없애려면?

콘솔 창에 입력하기

```
heading =
document.querySelector("h1")
heading .remove()
```



노드 삭제하기 – removeChild()

- removeChild()는 자식 노드 삭제
- 이 메서드를 사용하려면 우선 부모 노드를 찾아야 하고 그 후에 자식 노드를 삭제해야 한다.
- 예전에 IE에서는 removeChild() 메서드만 사용할 수 있어서, 주로 removeChild()를 사용했지만, 이제 모던 브라우저에서는 remove() 메서드를 사용해도 된다.

부모 노드를 찾는 parentNode 프로퍼티

```
노드명.parentNode
```

자식 노드를 제거하는 removeChild() 메서드

```
부모 노드.removeChild(자식 노드)
```

노드 삭제하기 – removeChild()

목록에서 각 항목을 클릭했을 때 항목이 삭제되게 하려면?

07₩remove-2.html

```
<h1>Web Programming</h1>
<ul id="items">
  <li>HTML</li>
  <li>CSS</li>
  <li>Javascript</li>
</ul>
```

07₩js₩remove-2.js

```
const items =
document.querySelectorAll("li");
for(let item of items) {
  item.addEventListener("click", function () {
    this.parentNode.removeChild(this);
    // this.remove(this); 도 가능
  });
}
```

이벤트 리스너에서 함수 안에 있는 this

```
item.addEventListener("click", function () {
  this.parentNode.removeChild(this);
});
```

이벤트 리스너에서 function() { ... }에 this를 사용하면 this는 이벤트가 발생한 노드를 가리킨다.

(예) 첫번째 항목을 클릭했다면 첫번째 li 요소가 this가 됩니다.

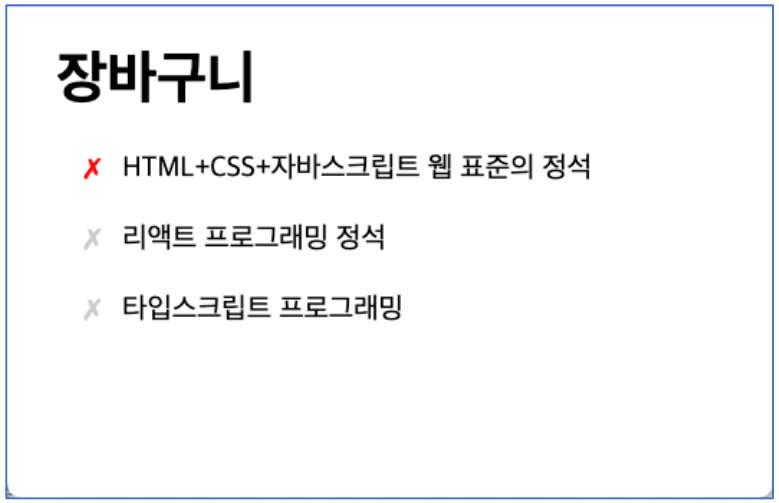
이벤트 리스너에서 화살표 함수를 사용할 경우 this는 최상위 객체 window를 가리킨다.

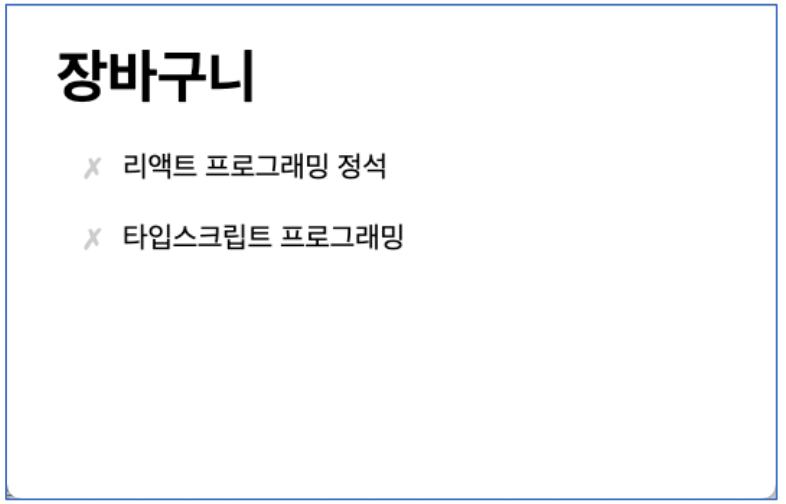
→ 화살표 함수를 사용할 경우 클릭한 요소를 찾을 때 this를 사용할 수 없다.

→ 이벤트가 발생한 요소를 this로 사용하려면 function() { } 를 사용한다.

(예) 삭제 버튼 클릭해서 삭제하기

여러 개의 항목이 있고 각 항목마다 [삭제] 버튼이 있을 경우 버튼을 클릭해서 항목 삭제하기





07\remove-3.html 문서 구조 살펴보기

```
<div id="cart">
  <h1>장바구니</h1>
  <div id="products">
    <p>
      <span>&cross;</span>
      HTML+CSS+자바스크립트 웹 표준의 정석
    </p>
    <p><span>&cross;</span>리액트 프로그래밍 정석</p>
    <p><span>&cross;</span>타입스크립트 프로그래밍</p>
  </div>
</div>
```

1) js\remove-3.js 파일에서 작성
2) 모든 삭제 버튼을 가져온 후
3) for 문을 사용해 삭제 버튼을 들여다 보면서, 어떤게 클릭됐는지 찾아본다.
4) 클릭한 버튼에 이벤트 리스너를 연결한다.

20₩js₩remove-3.js

```
const buttons = document.querySelectorAll("p > span");  // 모든 삭제 버튼을 가져온다.

for(let button of buttons) {
  button.addEventListener("click", function () {      // 항목 클릭했을 때 실행할 함수

  });
}
```

삭제 버튼을 클릭했을 때 어느 부분을 삭제해야 할까?

```
<div id="cart">
  <h1>장바구니</h1>
  <div id="products">
    <p>
      <span>&cross;</span>
      HTML+CSS+자바스크립트 웹 표준의 정석
    </p>
    ……
  </div>
</div>
```

① 삭제 버튼을 클릭하면
② 삭제 버튼의 부모 요소의 부모 요소를 찾아서,
③ 삭제 버튼의 부모 요소를 삭제한다.

20₩js₩remove-3.js

```
const buttons = document.querySelectorAll("p > span");  // 모든 삭제 버튼을 가져옵니다.

for(let button of buttons) {
  button.addEventListener("click", function () {       // 항목 클릭했을 때 실행할 함수
    this.parentNode.parentNode.removeChild(this.parentNode);
  });
}
```

[실습] 나만의 도서 목록 프로그램 만들기

- 구입한 책을 목록 형태로 기록하고 [삭제] 버튼을 클릭하면 목록에서 삭제하기
- 단, 여기에서 만드는 예제는 웹 브라우저 화면을 새로 고침하면 입력한 자료가 사라진다.
- 여기에서는 항목이 정상적으로 추가되고 삭제되는지만 연습해 보자.

마크업하기

07₩booklist.html

- 폼을 사용해 책 정보를 입력한다.
- 목록을 사용해 입력한 책 정보를 표시한다.


```
<div id="container">
  <h1>Book List</h1>
  <form>
    <ul id="bookInfo">
      <li>
        <label for="title">제 목</label>
        <input type="text" id="title">
      </li>
      <li>
        <label for="author">저 자</label>
        <input type="text" id="author">
      </li>
    </ul>

    <button type="reset">취소하기</button>
    <button id="save">저장하기</button>
  </form>

  <ul id="bookList"></ul>
</div>
```

CSS 작성하기

css₩booklist.css 파일 만들고, boolist.html에 연결하기

```
* {
  margin:0;
  padding:0;
  box-sizing: border-box;
}

#container{
  margin:50px auto;
  width:600px;
  padding:10px 20px;
}
ul {
  list-style: none;
}
```

```
#bookInfo {
  margin-top:40px;
}
#bookInfo li {
  font-size:1em;
  line-height: 3;
}
#bookInfo label {
  display:inline-block;
  width:50px;
}
#bookInfo input {
  width:450px;
  padding:5px;
}
```

```
button {
  width:150px;
  margin:20px 55px;
  font-size:1em;
  padding:5px 10px;
}

#bookList {
  width:500px;
  margin-top:80px;
  position:relative;
}
```

js 작성하기

1) 폼에 있는 텍스트 필드 요소를 가져오고 책 정보가 표시될 영역도 가져온다.
2) [save] 버튼에 click을 위한 이벤트 리스너를 연결한다.

```
const title = document.querySelector("#title");           // '제목' 정보
const author = document.querySelector("#author");         // '저자' 정보
const save = document.querySelector("#save");             // [저장하기] 버튼
const bookList = document.querySelector("#bookList");     // 정보가 표시될 영역


save.addEventListener("click", (e) => {      // [저장하기] 버튼을 클릭하면?


});
```

js 작성하기

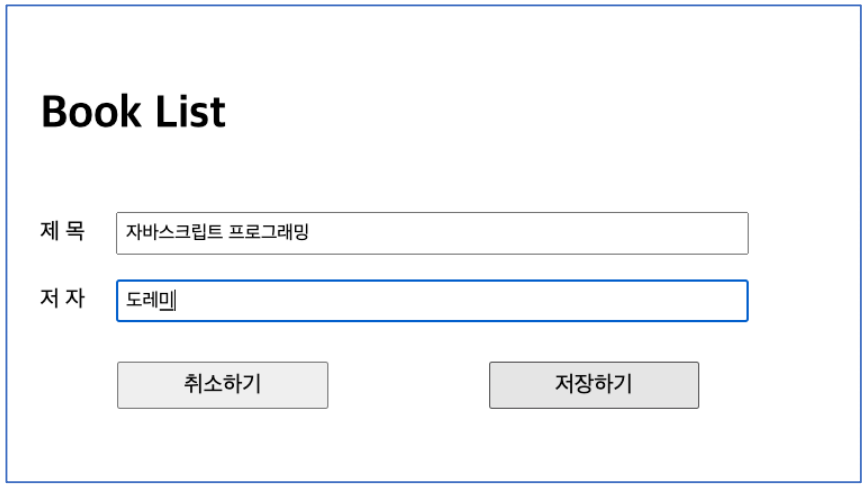
3) [저장하기] 버튼을 클릭하면 '제목'과 '저자' 필드에 입력된 내용을 가져와서

4) bookList 영역에 <li> 태그와 함께 추가한다.

```
save.addEventListener("click", (e) => {
  const item = document.createElement("li");
  item.innerHTML = `
    ${title.value} - ${author.value}
    <span id="delButton">삭제</span>
  `;
  bookList.appendChild(item);
});
```

브라우저로 확인하기

책 제목과 저자를 입력하고 [저장하기] 버튼을 클릭하면 내용이 그대로 사라져 버린다. 왜?



폼에 있는 버튼을 클릭하면 폼의 정보를 서버로 보내는 것이 기본 동작이기 때문에, 그 정보를 서버로 보냈다고 생각하고 화면을 새로 고친다.

→ 기본 동작이 실행되지 않도록 해야 한다.

js 작성하기

버튼을 클릭했을 때의 기본 동작을 취소하기 위해, 이벤트 리스너 안에 소스 추가

```
save.addEventListener("click", (e) => {
  e.preventDefalut();    // 폼의 버튼을 클릭했을 때 서버로 보내지 않도록
  const item = document.createElement("li");
  item.innerHTML = `
    ${title.value} - ${author.value}
    <span id="delButton">삭제</span>
  `;
  bookList.appendChild(item);
});
```

브라우저로 확인하기

책 제목과 저자를 입력하고 [저장하기] 버튼을 클릭한다.

결과 화면에 내용은 나타나는데, 폼 입력 필드에 입력한 내용이 그대로 남아 있다.

js 작성하기

폼의 내용이 화면에 표시된 후 텍스트 필드 안의 내용을 지운다.

```
save.addEventListener("click", (e) => {
  e.preventDefalut();
  ……
  bookList.appendChild(item);
  title.value = "";
  author.value = "";
});
```

CSS 작성하기

결과 화면의 스타일 만들기

```
#bookList li {
  font-size:1em;
  padding:10px;
  border-bottom:1px solid #ccc;
}
#delButton:hover {
  cursor:pointer;
}
#delButton {
  padding:5px 10px;
  font-size:0.8em;
  position:absolute;
  right:10px;
}
```

js 작성하기 – '삭제' 버튼 클릭했을 때 항목 삭제하기

목록을 추가한 후에 삭제할 수 있기 때문에 [저장하기] 버튼과 연결된 이벤트 리스너에 연결해서 작성한다.

화면에 있는 [삭제] 버튼을 모두 가져온 후 반복문을 통해 click 이벤트가 발생한 것이 있는지 확인한다.

```
save.addEventListener("click", (e) => {
  e.preventDefault();
  ......

  const delButtons =
document.querySelectorAll("#delButton");
  for (let delButton of delButtons) {

  }
});
```

js 작성하기 – '삭제' 버튼 클릭했을 때 항목 삭제하기

```
save.addEventListener("click", (e) => {
  e.preventDefault();
  ......

  const delButtons = document.querySelectorAll("#delButton");
  for (let delButton of delButtons) {
    delButton.addEventListener("click", function () {
      this.parentNode.parentNode.removeChild(this.parentNode);
    });
  }
});
```

브라우저로 확인하기

책 제목과 저자를 입력하고 [저장하기] 버튼을 클릭한다.

책 목록에서 '삭제' 버튼을 클릭하고 해당 항목이 삭제되는지 확인한다.