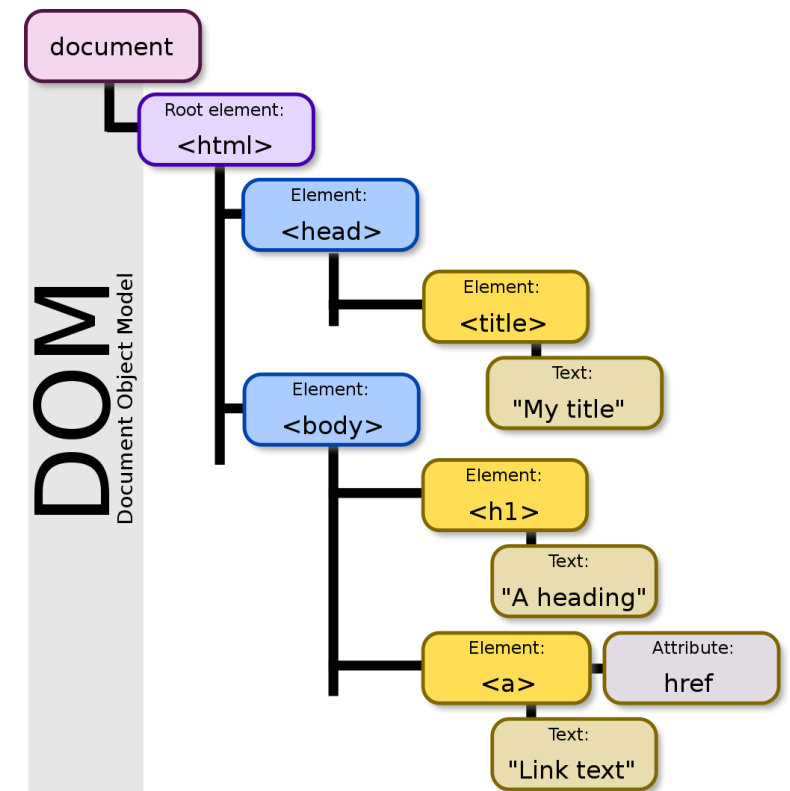


JAVASCRIPT

DOM

❖ Document Object Model

- HTML문서를 이루는 각 요소에 접근할 수 있도록 지원하는 인터페이스 역할
- 화면에 어떤 요소를 렌더링할 것인지 결정하기 위해 사용
- 자바스크립트에서 문서 구조 및 내용을 조작하기 위해 사용
- DOM과 HTML문서의 요소가 반드시 일치하지 않을 수 있음

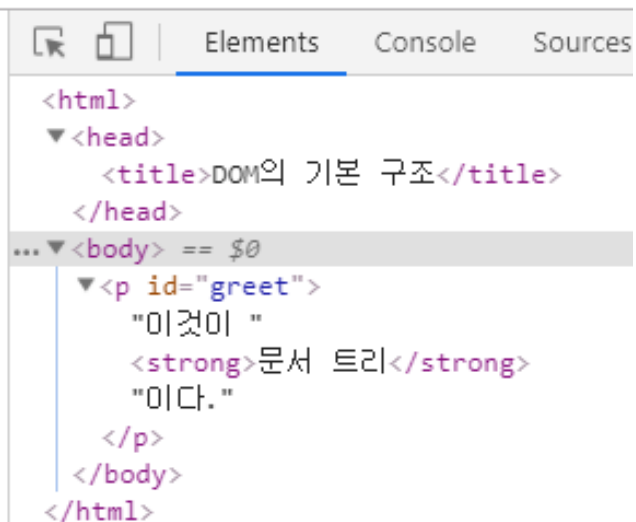


DOM

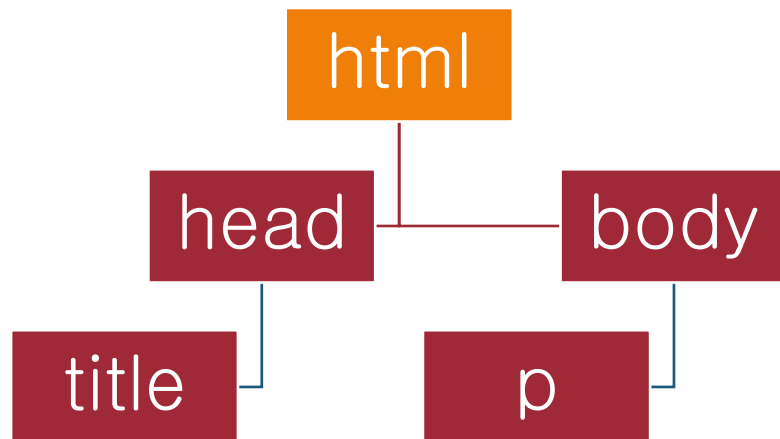
❖ 개념 이해

```
<html>
<head>
<title>DOM의 기본 구조</title>
</head>
<body>
  <p id="greet">이것이 <strong>문서 트리</strong>이다.</p>
</body>
</html>
```

이것이 문서 트리이다.



The screenshot shows the 'Elements' tab of a browser's developer tools. It displays the DOM tree for the provided HTML code. The root node is 'html', which has two children: 'head' and 'body'. The 'head' node contains a 'title' node with the text 'DOM의 기본 구조'. The 'body' node contains a 'p' node with the text '이것이 문서 트리이다.'. The 'p' node's text is split into three parts: '이것이 ', '문서 트리', and '이다.'.



DOM

- ❖ 문서에 포함되는 요소나 속성, 텍스트를 각각의 객체로 본다.(객체의 집합)
- ❖ 문서 구성하는 각 요소 객체를 노드라고 한다.
- ❖ 객체의 종류에 따라 요소노드, 속성노드, 텍스트 노드 등으로 부름.
- ❖ DOM은 노드들을 추출/추가/치환/삭제를 위한 범용적인 함수를 제공하는 API
- ❖ DOM은 웹 표준화 단체 W3C에서 표준화 진행 중인 레벨이 있다.

레벨	권고 시기	개요
Level 1	1998/01	노드의 참조/추가/치환/삭제 등의 기본 기능을 정의
Level 2	2000/11	Level 1에 스타일 조작이나 이벤트 모델, 노드의 범위 지정 등의 기능을 추가
Level 3	2004/04(일부)	Level 2에 XPath, 타당성 검증, 읽기/보존 등의 기능을 추가

출처: <https://www.w3.org/DOM/DOMTR>

DOM – 특정 노드 참조하기

- ❖ 문서에서 어떤 문자열이나 특정 요소를 컨트롤 하고자 하는 경우에 노드를 참조하는 방법
 - 직접 접근(Direct Access)
 - 노드의 id값이나 name값 또는 태그명을 이용하여 직접 노드를 참조(성능 저하 고려)
 - 노드 워킹(Node Walking)
 - 특정 노드를 기준으로 자식, 또는 부모, 형제 노드에 접근하여 참조(절차 복잡도 고려)

DOM – 특정 노드 참조하기

❖ 1. 노드의 id값을 이용하여 노드를 참조하기

- getElementById()
- HTML문서에서 요소에 부여하는 id속성을 문서내에서 고유함.
- id값을 이용하여 특정 노드를 자바스크립트로 참조하여 조작이 가능

```
<script>
function btn_click() {
    let result = document.getElementById("result");
    result.innerHTML = "안녕하세요," + document.fm.name.value + "님!";
    //result.innerHTML = '안녕하세요,' + document.getElementById('name').value + '님!';
}
</script>
</head>
```

innerHTML : 현재 요소의 내용을 가져오거나 설정할 수 있는 프로퍼티

```
<body>
<form name="fm">
    <label>이름 : <input type="text" id="name" name="name" size="15"/></label>
    <input type="button" value="전송" onclick="btn_click()" />
    <div id="result"></div>
</form>
</body>
```

이름 :

안녕하세요,홍길동씨 !

DOM – 특정 노드 참조하기

❖ 2. 노드의 태그 이름을 이용하여 노드를 참조하기

- `getElementsByTagName()`
- 동일한 이름의 태그가 여러 개 있을 경우 집합을 반환한다.(select, list 등)

```
<script>
  window.onload = function () {
    let result = [];
    let list = document.getElementsByTagName('a');
    for (let i = 0; i < list.length; i++) {
      result.push(list.item(i).href);
    }
    window.alert(result.join('\n'));
  }
</script>
</head>

<body>
  <a href="http://www.google.co.kr">외국친구</a><br />
  <a href="http://www.naver.com">한국친구</a>
</body>
```

이 페이지 내용:

<http://www.google.co.kr/>

<http://www.naver.com/>

확인

DOM – 특정 노드 참조하기

❖ 3. 노드의 클래스 속성을 이용하여 노드 참조하기

- `getElementsByClassName()`
- 동일한 클래스 이름을 가진 요소를 반환

```
<script>
  window.onload = function () {
    let result = [];
    let list = document.getElementsByClassName('my');
    for (let i = 0; i < list.length; i++) {
      result.push(list.item(i).href);
    }
    window.alert(result.join('\n'));
  }
</script>
</head>

<body>
  <a href="http://www.google.co.kr" class="my">외국친구</a><br />
  <a href="http://www.naver.com" class="my">한국친구</a>
</body>
```

이 페이지 내용:

<http://www.google.co.kr/>
<http://www.naver.com/>

확인

DOM – 특정 노드 참조하기

❖ 4. 노드의 이름 속성을 이용하여 노드 참조하기

- `getElementsByName()`
- 앞의 예제에서 `getElementsByClassName`을 `getElementsByName`으로 변경
- `a`태그에 지정된 `class`속성을 `name`속성으로 변경

```
<script>
  window.onload = function () {
    let result = [];
    let list = document.getElementsByName('my');
    for (let i = 0; i < list.length; i++) {
      result.push(list.item(i).href);
    }
    window.alert(result.join('\n'));
  }
</script>
</head>

<body>
  <a href="http://www.google.co.kr" name="my">외국친구</a><br />
  <a href="http://www.naver.com" name="my">한국친구</a>
</body>
```

이 페이지 내용:

<http://www.google.co.kr/>

<http://www.naver.com/>

확인

DOM – 특정 노드 참조하기

❖ 상대위치로 노드 참조하기

```
<script>
  window.onload = function () {
    let result = [];
    let s = document.getElementById("food");
    let opts = s.childNodes;
    for (let i = 0; i < opts.length; i++) {
      let opt = opts.item(i);
      if (opt.nodeType == 1) {
        result.push(opts.item(i).value);
      }
    }
    window.alert(result.join(", "));
  };
</script>
```

```
<body>
  <form name="fm">
    가장 먹고 싶은 음식은 ? :
    <select id="food" name="food">
      <option value="라면">라면</option>
      <option value="만두">만두</option>
      <option value="불고기">불고기</option>
    </select>
    <input type="submit" value="제출" />
  </form>
</body>
```

이 페이지 내용:

라면,만두,불고기

확인

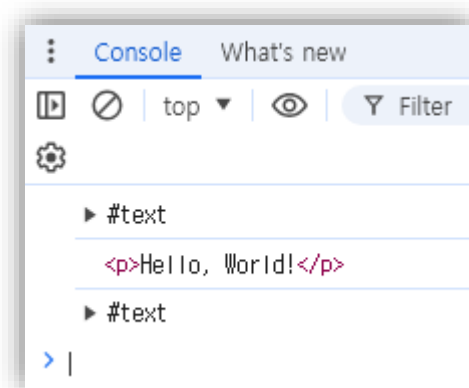
DOM – 특정 노드 참조하기

- ❖ 앞의 예제는 food id를 가진 노드를 찾아 하위 노드 집합을 가져온다.
 - 하위 노드 집합에 공백이나 개행이 텍스트 노드의 형태로 인식될 수 있다.(브라우저마다 다름)
 - option태그만 가져오게 하려는 경우는 nodeType을 활용한다.

반환값	개요
1	요소 노드
2	속성 노드
3	텍스트 노드
4	CDATA 섹션
5	실제 참조 노드
6	실제 선언 노드
7	처리 명령 노드
8	코멘트 노드
9	문서 노드
10	문서형 선언 노드
11	문서의 단편(fragment)
12	기법 선언 노드

```
<script>
  window.onload = function () {
    let div = document.querySelector('div');
    div.childNodes.forEach(node => {
      console.log(node);
    });
  };
</script>
</head>

<body>
  <div>
    <p>Hello, World!</p>
  </div>
</body>
```



요소간의 공백이나 개행은 브라우저마다 다르므로 크로스브라우징을 고려한다면 노드타입을 고려하여 작성하도록 하자.

DOM – 속성 값 가져오기

- ❖ 특정 요소에 접근할 수 있다면 해당 요소의 속성에도 접근이 가능
- ❖ 접근 시 거의 대부분 속성이름과 동일한 프로퍼티로 접근
- ❖ 예외 적인 속성 이름들이 존재한다. (ex. HTML class 속성은 DOM에서 className으로 접근)
- ❖ 만약 HTML속성이름과 DOM에서의 이름이 다른 경우 setAttribute() / getAttribute() 메서드를 사용하면 된다.

```
var url = link.href;  
link.href = 'http://www.naver.com/';
```

```
<p class="sumary">샘플</p> HTML  
node.className = 'sumary'; DOM
```

```
요소 노드.getAttribute(속성명)  
요소 노드.setAttnibute(속성명, 속성값)
```

DOM – 속성 값 가져오기

- ❖ 특정 노드에 속하는 모든 속성을 가져오려는 경우 attributes 프로퍼티를 사용
- ❖ 예제

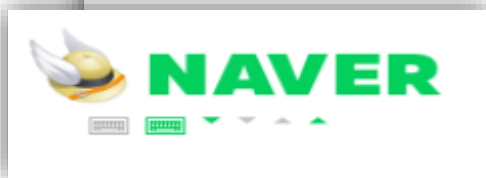
```
<script>
  window.onload = function () {
    // Array.from 사용
    const logo = document.getElementById("logo");
    const attrs = Array.from(logo.attributes);
    const result = attrs.map((attr) => `${attr.name}: ${attr.value}`);
    window.alert(result.join("\r\n"));
  };
</script>
</head>

<body>
  
</body>
```

이 페이지 내용:

id:logo
src:https://s.pstatic.net/static/www/img/uit/2019/sp_search.png
height:67
width:215
border:0
alt:Naver

확인



- attributes는 노드의 전체 속성을 NamedNodeMap타입으로 저장하고 있음 (이름이나 인덱스 번호로 접근이 가능)
- 각 속성은 nodeName과 nodeValue로 꺼낼 수 있다.

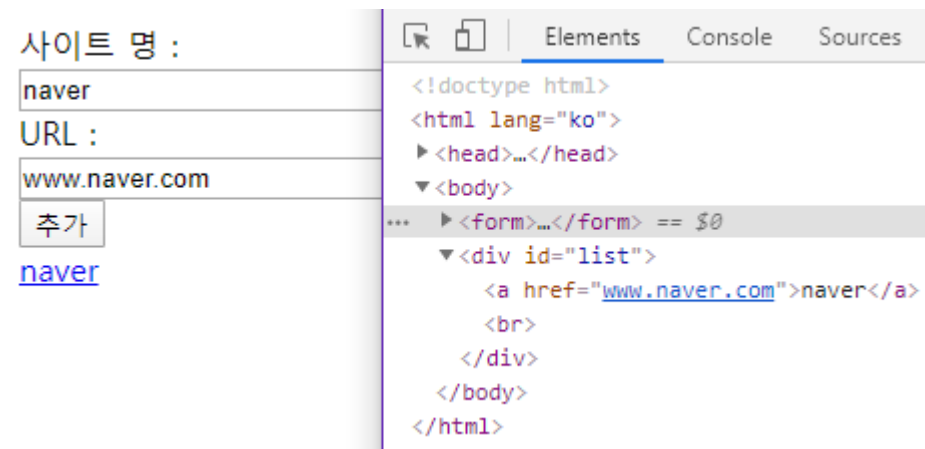
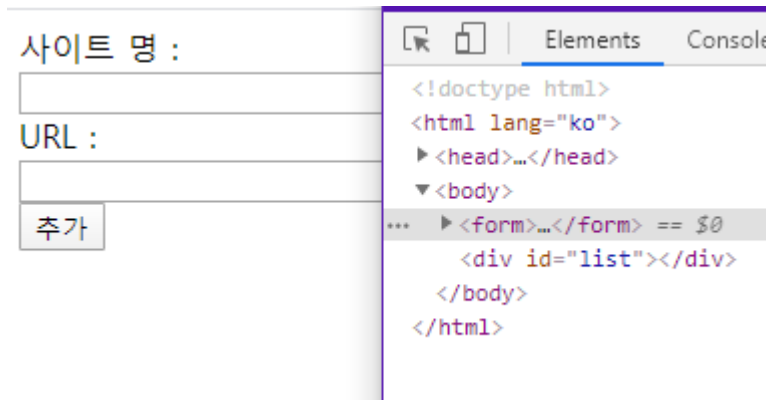
DOM - 노드 치환

❖ 문서 트리에 새 노드를 추가 하거나 기존 노드를 삭제할 수 있다.

- 간단한 컨텐츠 편집은 innerHTML을 사용
- 복잡한 컨텐츠 변경은 다음 예제의 접근 방법을 활용한다.

```
<script>
  function add(f) {
    let anchor = document.createElement("a"); //태그 생성
    anchor.href = f.url.value; //태그에 속성 지정
    let name = document.createTextNode(f.name.value); //텍스트 노드 생성
    anchor.appendChild(name); //anchor의 맨 마지막 자식으로 추가
    let br = document.createElement("br"); //태그 생성
    let list = document.getElementById("list");
    list.appendChild(anchor); //리스트 노드에 자식으로 추가
    list.appendChild(br); //리스트 노드에 자식으로 추가
  }
</script>
</head>

<body>
  <form>
    <label>사이트 명: <input type="text" name="name" size="30" /></label><br />
    <label>URL: <input type="text" name="url" size="50" /></label><br />
    <input type="button" value="추가" onclick="add(this.form)" />
  </form>
  <div id="list"></div>
</body>
```



DOM - 노드 치환

❖ 동일한 결과의 다른 형태 예제

```
<script>
function add(f) {
  let anchor = document.createElement("a"); //태그 생성
  let href = document.createAttribute("href"); //속성 생성
  href.nodeValue = f.url.value;
  anchor.setAttributeNode(href);
  let name = document.createTextNode(f.name.value); //텍스트노드 생성
  anchor.appendChild(name);
  let br = document.createElement("br"); //속성 생성
  let list = document.getElementById("list");
  list.insertBefore(br, null); //기준이 없으므로 마지막에 추가
  list.insertBefore(anchor, br); //br요소 앞에 anchor삽입
}
</script>
```

- node.appendChild(추가할노드)
=> 자식으로 추가한다. 기존의 자식 노드가 있다면 제일 마지막에 추가
- node.insertBefore(추가할 노드, 기준 노드)
=> 추가할 노드를 기준노드 앞에 삽입한다.
=> 마지막 노드로 추가하고 싶다면 두 번째 인자를 null로 한다.

사이트 명 :

URL :

사이트 명 :

URL :

[daum](#)

Elements

Console

Sources

```
<!doctype html>
<html lang="ko">
  <head>...</head>
  ...<body> == $0
    <form>...</form>
    <div id="list"></div>
  </body>
</html>
```

```
<!doctype html>
<html lang="ko">
  <head>...</head>
  ...<body> == $0
    <form>...</form>
    <div id="list">
      <a href="www.daum.net">daum</a>
      <br>
    </div>
  </body>
</html>
```

DOM – 노드 치환

❖ 속성 노드 치환하기

- 태그 요소에 적용될 속성 노드도 추가가 가능하다.
- 속성과 동일한 이름의 프로퍼티를 설정하면 된다.
- 속성을 문자열로 지정할 수 있으므로 "스크립트로부터 동적으로 속성 이름을 변경할 수 있다.

예제)

```
anchor.href= f.url.value;  
위 속성 추가 코드를 다음과 같이 변경할 수 있음.
```

```
var href = document.createAttribute("href");  
href.nodeValue = f.url.value;    //속성 노드의 값을 지정  
anchor. setAttributeNode(href); //노드에 속성 지정
```


DOM – 노드 치환

❖ 기존 노드 치환/삭제하기

- 이미지 파일 준비 – 예제코드에서 파일이름 활용(파일이름에 따라 코드에서 파일이름 수정)
- 다른 이미지를 사용해도 무관



978-4-7741-407
6-6.jpg



978-4-7741-422
3-4.jpg



978-4-7981-220
5-2.jpg



978-4-8443-286
5-0.jpg



978-4-8443-287
9-7.jpg

DOM - 노드 치환

❖ 예제

```
<script>
function show(e, isbn) {
    let img = document.createElement("img");
    img.src = isbn + ".jpg";
    img.alt = e.innerHTML;
    img.height = 150;
    img.width = 108;
    let pic = document.getElementById("pic");
    if (pic.getElementsByTagName("img").length > 0) {
        pic.replaceChild(img, pic.lastChild);
    } else {
        document.getElementById("del").disabled = false;
        pic.appendChild(img);
    }
}

function del() {
    let pic = document.getElementById("pic");
    pic.removeChild(pic.lastChild);
    document.getElementById("del").disabled = true;
}
</script>
```

```
<body>
<ul>
    <li>
        <a href="JavaScript:void(0)" onclick="show(this, '978-4-7741-4223-4')">
            Apache 포켓 레퍼런스</a>
        </li>
    <li>
        <a href="JavaScript:void(0)" onclick="show(this, '978-4-7741-4076-6')">
            3단계로 확실히 배우는 MySQL 입문</a>
        </li>
    <li>
        <a href="JavaScript:void(0)" onclick="show(this, '978-4-8443-2879-7')">
            개정3판 기초PHP</a>
        </li>
    <li>
        <a href="JavaScript:void(0)" onclick="show(this, '978-4-7981-2205-2')">
            Visual Studio 2010 스타트 업 가이드</a>
        </li>
    <li>
        <a href="JavaScript:void(0)" onclick="show(this, '978-4-8443-2865-0')">
            Perl 프레임워크 Catalyst 완전 입문</a>
        </li>
</ul>
<input type="button" id="del" value="삭제" disabled="disabled" onclick="del()"/>
<div id="pic"></div>
</body>
```

DOM - 노드 치환

❖ 실행 확인

- [Apache 포켓 레퍼런스](#)
- [3단계로 확실히 배우는 MySQL입문](#)
- [개정3판 기초PHP](#)
- [Visual Studio 2010 스타트 업 가이드](#)
- [Perl 프레임워크 Catalyst 완전 입문](#)

삭제

```
Elements Console
<!doctype html>
<html lang="ko">
  <head>...</head>
  <body> == $0
    <ul>...</ul>
    <input type="button" id="del" value="삭제" onclick="del()" />
    <div id="pic"></div>
  </body>
</html>
```

- [Apache 포켓 레퍼런스](#)
- [3단계로 확실히 배우는 MySQL입문](#)
- [개정3판 기초PHP](#)
- [Visual Studio 2010 스타트 업 가이드](#)
- [Perl 프레임워크 Catalyst 완전 입문](#)

삭제



```
Elements Console Sources Network Performance Me
<!doctype html>
<html lang="ko">
  <head>...</head>
  <body> == $0
    <ul>...</ul>
    <input type="button" id="del" value="삭제" onclick="del()" />
    <div id="pic">
      
    </div>
  </body>
</html>
```

- [Apache 포켓 레퍼런스](#)
- [3단계로 확실히 배우는 MySQL입문](#)
- [개정3판 기초PHP](#)
- [Visual Studio 2010 스타트 업 가이드](#)
- [Perl 프레임워크 Catalyst 완전 입문](#)

삭제

```
Elements Console
<!doctype html>
<html lang="ko">
  <head>...</head>
  <body> == $0
    <ul>...</ul>
    <input type="button" id="del" value="삭제" onclick="del()" />
    <div id="pic"></div>
  </body>
</html>
```

- [Apache 포켓 레퍼런스](#)
- [3단계로 확실히 배우는 MySQL입문](#)
- [개정3판 기초PHP](#)
- [Visual Studio 2010 스타트 업 가이드](#)
- [Perl 프레임워크 Catalyst 완전 입문](#)

삭제



```
Elements Console Sources Network
<!doctype html>
<html lang="ko">
  <head>...</head>
  <body> == $0
    <ul>...</ul>
    <input type="button" id="del" value="삭제" onclick="del()" />
    <div id="pic">
      
    </div>
  </body>
</html>
```

DOM – 노드 치환

❖ 노드 치환하기

- 자식 노드를 옮긴다.
- 치환될 대상 노드는 현재 노드에 대한 자식 노드여야 한다.

```
node.replaceChild(치환에 사용될 노드, 치환될 대상 노드)  
pic.replaceChild(img, pic.lastChild)
```

❖ 노드 삭제하기

- 삭제될 대상 노드는 현재 노드에 대한 자식 노드여야 한다.

```
pic.removeChild(pic.lastChild)
```

DOM – StyleSheet 조작

❖ HTML5 에서 중요한 개념

- HTML로 문서의 구조를 정하는 것
- CSS로 문서의 디자인을 하도록 나누는 것
- HTML요소에 디자인을 적용하는 부분에서 DOM을 활용하게 된다.

❖ DOM으로 스타일시트를 조작하는 방법

- 인라인 스타일에 접근(style 프로퍼티 사용)
- 외부 스타일시트를 적용(className 프로퍼티 사용)

❖ 다음과 같은 속성들에 대한 조작이 가능

- 테두리, 배경, 텍스트 표시, 폰트, 표시와 배치, 리스트. 여백(마진), 여백(패딩), 커서 등

DOM – StyleSheet 조작

❖ 1. 인라인 스타일에 접근

- 인라인 스타일이란 객체의 노드에 직접 설정된 스타일
- `<div style='color:Red' > 붉은 문자입니다.</div>`

```
<script>
  function changeStyle(elem, color) {
    elem.style.backgroundColor = color;
  }
</script>
</head>
```

```
<body>
  <div
    onmouseover="changeStyle(this, 'Pink')"
    onmouseout="changeStyle(this, 'White')"
  >
    마우스를 올려놓으면 색이 변합니다.
  </div>
</body>
```

마우스를 올려놓으면 색이 변합니다.

마우스를 올려놓으면 색이 변합니다.

- 위와 같이 인라인 스타일 접근 시 프로퍼티명에 대해 주의
- CSS속성 이름 중에 하이픈을 포함하는 것(background-color)과 같은 프로퍼티이름은 자바스크립트에서 직접 접근 시 하이픈을 제거하고 카멜 표기법으로 고쳐야 한다.

background-color -> backgroundColor
border-top-style -> borderTopStyle
단. float 속성만 예외로 styleFloat으로 사용한다.

DOM – StyleSheet 조작

❖ 외부 스타일시트 적용하기 className

- 인라인 스타일 적용은 간단하지만 스타일 속성이 많으면 소스가 혼란스러워진다.
- 스타일 변경에도 유연하기 어렵다.
- 따라서 스타일시트는 별도의 파일에서 관리하도록 하는 것이 일반적이다.
- 이때 외부에 정의된 스타일시트에 접근하기 위해 className속성을 사용하게 된다.
- className 에는 복수의 클래스와도 연결 지을 수 있다.

DOM – StyleSheet 조작

❖ 예제

```
<!DOCTYPE html>
<html lang="ko">

<head>
  <meta charset="utf-8" />
  <title>DOM의 기본 구조</title>

  <link rel="stylesheet" type="text/css" href="style.css" />
  <script>
    function changeStyle(elem, clazz) {
      elem.className = clazz;
    }
  </script>
</head>

<body>
  <div onmouseover="changeStyle(this, 'highlight')"
    onmouseout="changeStyle(this, 'normal')">
    마우스를 올려 놓으면 색이 변한다.</div>
</body>
</html>
```

예제용 css파일 (style.css)

```
.highlight {
  background-color:   Pink;
}

.normal {
  background-color:   White;
}
```

마우스를 올려놓으면 색이 변합니다.

마우스를 올려놓으면 색이 변합니다.

DOM – Event

❖ Event처리

- 일반적인 이벤트처리는 동일한 요소/이벤트에 대해 복수의 이벤트 핸들러를 정의할 수 없다.
- 여러 라이브러리를 조합하여 이용하고 있는 경우에는 어떤 라이브러리가 어떤 요소의 특정 이벤트를 사용해버리면 다른 라이브러리에서 동일한 요소에 다른 이벤트를 등록할 수가 없다.
- 위와 같은 경우에 사용할 수 있는 방법이 이벤트리스너이다.

❖ EventListener

- 이벤트 리스너는 이벤트 핸들러와 같이 발생한 이벤트에 대응하는 처리를 정의하는 것은 같지만 다음과 같은 부분이 다르다.
- 동일 요소의 동일 이벤트에 대해서 복수의 이벤트 리스너를 등록할 수 있다.
- 먼저 설정한 이벤트 리스너를 삭제할 수 있다.
- 단점으로는 브라우저를 의식하고 작성해야 한다.
 - IE – attachEvent메서드 사용
 - 그 외 – addEventListener 메서드 사용

DOM – Event

❖ 이벤트 등록

addEventListener/attachEvent 메소드

요소 노드. attachEvent (이벤트명, 이벤트 핸들러)

요소 노드. addEventListener (이벤트명, 이벤트 핸들러, 이벤트의 전달 방향)

❖ 이벤트 삭제

detachEvent/removeEventListener 메소드

요소 노드.detachEvent (이벤트명, 이벤트 핸들러명)

요소 노드.removeEventListene (이벤트 명, 이벤트 핸들러 명, 이벤트의 전달 방향)

DOM – Event

❖ 1. 이벤트 리스너 등록하기(예전)

```
<script>
  function addListener(elem, ev, listener) {
    if (elem.addEventListener) { //IE가 아닐 경우
      elem.addEventListener(ev, listener, false);
    } else if (elem.attachEvent) { //IE일 경우
      elem.attachEvent('on' + ev, listener);
    } else {
      throw new Error('이벤트 리스너에 미대응입니다.');
```

```
    }
  }
  addListener(window, 'load', btn_click);

  function btn_click() {
    addListener(document.getElementById('btn'), 'click', function () {
      window.alert('버튼이 클릭되었습니다.');
```

```
    });
  }
</script>
</head>

<body>
  <input type="button" id='btn' value="다이얼로그표시" />
</body>
```

다이얼로그표시

이 페이지 내용:
버튼이 클릭되었습니다.

확인

크로스 브라우징을 위한 형태의 코드
페이지 로드 시 addListener()호출
window객체의 load시 init메서드를 핸들러로 등록

DOM – Event

❖ 1. 이벤트 리스너 등록하기(요즘)

```
<script>
  document.addEventListener('DOMContentLoaded', () => {
    const button = document.getElementById('btn');
    button.addEventListener('click', () => {
      window.alert('버튼이 클릭되었습니다.');
    });
  });
</script>
</head>
<body>
  <input type="button" id="btn" value="다이얼로그 표시" />
</body>
```

다이얼로그 표시

이 페이지 내용:
버튼이 클릭되었습니다.

확인

69px

DOM – Event

❖ 2. 이벤트 리스너 삭제하기(예전)

```
function removeListener(elem, ev, listener) {  
  if (elem.removeEventListener) {  
    elem.removeEventListener(ev, listener, false);  
  } else if (elem.detachEvent) {  
    elem.detachEvent('on' + ev, listener);  
  } else {  
    throw new Error('이벤트 리스너에 미대응입니다.');  }  
}
```

```
function btn_click() {  
  window.alert('버튼이 클릭되었습니다.');}
```

```
addListener(window, 'load', init);
```

```
function init() {  
  addListener(document.getElementById('btn'), 'click', btn_click);  
  addListener(document.getElementById('btnRemove'), 'click', btnRemove_click);  
}
```

```
function btnRemove_click() {  
  removeListener(document.getElementById('btn'), 'click', btn_click);  
}
```

```
</script>
```

```
<body>  
  <input type="button" id='btn' value="다이얼로그 표시" />  
  <input type="button" id='btnRemove' value="이벤트 리스너 삭제" />  
</body>
```

다이얼로그 표시

이벤트 리스너 삭제

다

이 페이지 내용:

버튼이 클릭되었습니다.

확인

다이얼로그 표시

이벤트 리스너 삭제

다이얼로그 표시

이벤트 리스너 삭제

DOM – Event

❖ 2. 이벤트 리스너 삭제하기(요즘)

```
<script>
  document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('btn');
    const btnRemove = document.getElementById('btnRemove');

    function btn_click() {
      window.alert('버튼이 클릭되었습니다.');
```

```
    }

    function btnRemove_click() {
      btn.removeEventListener('click', btn_click);
    }

    // 이벤트 리스너 추가
    btn.addEventListener('click', btn_click);
    btnRemove.addEventListener('click', btnRemove_click);
  });
</script>
```

```
<body>
  <input type="button" id="btn" value="다이얼로그 표시" />
  <input type="button" id="btnRemove" value="이벤트 리스너 삭제" />
</body>
```

다이얼로그 표시

이벤트 리스너 삭제

다

이 페이지 내용:

버튼이 클릭되었습니다.

확인

다이얼로그 표시

이벤트 리스너 삭제

다이얼로그 표시

이벤트 리스너 삭제

DOM – Event

❖ 이벤트 관련 정보 취득하기

- Event객체는 이벤트를 발생시킨 객체에 대한 정보를 참조하고 있다.
- 테스트를 위해 앞에서 정의한 이벤트 함수를 외부 파일로 저장한다.
- listener.js, listener2.js

```
function addListener(elem, ev, listener) {  
  if(elem.addEventListener) {  
    elem.addEventListener(ev, listener, false);  
  } else if(elem.attachEvent) {  
    elem.attachEvent('on' + ev, listener);  
  } else {  
    throw new Error('이벤트 리스너에 미대응입니다.');
```

```
}  
  
function removeListener(elem, ev, listener) {  
  elem.removeEventListener(ev, listener, false);  
}
```

DOM – Event

❖ 예제

```
<script type="text/javascript" src="listener.js"></script>
<script>
```

```
  addListener(window, 'load', init);

  function init() {
    addListener(document.getElementById('title'), 'mousedown',
      function (e) {
        var result = [];
        result.push('발생원 : ' + getSource(e).id);
        result.push('이벤트 : ' + e.type);
        result.push('버튼 : ' + getButtonType(e));
        result.push('X좌표 : ' + e.clientX);
        result.push('Y좌표 : ' + e.clientY);
        window.alert(result.join('\n'));
      }
    );
  }

  function getSource(e) {
    if (e.target) {
      return e.target;
    } else if (window.event) {
      return window.event.srcElement;
    }
  }
}
```

```
function getButtonType(e) {
  if (e.target) {
    switch (e.button) {
      case 0: return 'left';
      case 1: return 'center';
      case 2: return 'right';
    }
  } else if (window.event) {
    switch (window.event.button) {
      case 1: return 'left';
      case 2: return 'right';
      case 4: return 'center';
    }
  }
}
```

```
</script>
</head>

<body>
  <h1 id="title">[마우스로 클릭해보세요]</h1>
</body>
```

이 페이지 내용:

발생원 : title
이벤트 : mousedown
버튼 : left
X좌표 : 296
Y좌표 : 40

확인

DOM – Event

❖ 동작 확인 후 Event객체를 취급하는 경우의 주의사항을 보자.

- Event객체를 얻으려면 브라우저들 간의 동작 차이를 알고 있어야 한다.
- 기본적으로 대부분의 브라우저에서 이벤트 발생 시 이벤트 핸들러 함수가 등록되어 있다면 발생한 이벤트 객체를 이벤트리스너 함수의 첫 번째 인수로 전달받는다.
- 이 때 주의해야 하는 것은 이렇게 이벤트를 전달받은 핸들러 내에서 Event객체를 이용하려는 경우이다.
- 역시 대부분의 브라우저에서 이벤트 핸들러 안에서 핸들러를 정의하는 경우 이벤트 객체를 첫 번째 인자로 전달받지만 Internet Explorer는 이벤트 객체를 전달하지 않는다.
- 이런 경우 window.event 프로퍼티를 경유하여 명시적으로 Event객체를 취할 필요가 있다.
- 일반적인 상황에서는 고려하지 않겠지만 만약 이벤트핸들러에서 이벤트 객체를 취급하려는 경우에는 크로스 브라우징을 고려해야할 필요가 있으며 특정한 형태의 코드가 필요하다.

DOM – Event

❖ 이벤트 핸들러에서 이벤트 객체에 접근해야 하는 경우 사용되는 코드 형태(이전)

```
window.onload = function(){
    document.getElementById('btn').onmousedown = function(e){
        if(!e) { e = window.event; } //브라우저에서 자동으로 이벤트가 발생된 객체를 인자로 넘기지 않는 경우
        //event 객체에 의해 읽어들이는 코드
    }
}
```

- 위 코드와 같이 이벤트가 전달되지 않은 경우 window.event 프로퍼티로부터 이벤트 객체를 명시하여 가져온다. 이를 활용하면 어떤 브라우저든 변수 e를 이용하여 이벤트 객체에 접근할 수 있다.

DOM – Event

❖ 이벤트 핸들러에서 이벤트 객체에 접근해야 하는 경우 사용되는 코드 형태

```
document.addEventListener('DOMContentLoaded', () => {  
  const btn = document.getElementById('btn');  
  btn.addEventListener('mousedown', (e) => {  
    // event 객체는 모든 주요 브라우저에서 자동으로 전달  
    console.log('마우스 버튼이 눌렸습니다.', e);  
  });  
});
```

- 어떤 브라우저든 이벤트가 발생된 객체는 자동으로 callback 함수로 전달됨
- 변수 e를 이용하여 이벤트 객체에 접근 가능

DOM – Event

❖ Event객체에서 사용 가능한 프로퍼티

분류	프로퍼티	설명
일반	srcElement	이벤트를 발생시킨 요소(IE 브라우저)
	target	이벤트를 발생시킨 요소(IE 이외 브라우저)
	type	이벤트 종류(click ,mouseover 등)
좌표	clientX	이벤트 발생 좌표(브라우저에서의 좌표)
	clientY	이벤트 발생 좌표(브라우저에서의 좌표)
	screenX	이벤트 발생 좌표(화면에서의 좌표)
	screenY	이벤트 발생 좌표(화면에서의 좌표)
키보드/마우스	button	마우스 버튼 구분(좌/우/중앙)
	keyCode	키보드 키 입력 코드 값
	altKey	알트키 눌림 상태
	ctrlKey	컨트롤키 눌림 상태
	shiftKey	시프트키 눌림 상태

DOM – Event

❖ 이벤트 객체 프로퍼티에 대한 주요 사항

- 이벤트를 발생시킨 요소를 취득하는 것은 srcElement/target 프로퍼티이다.

➤ ~~IE -> srcElement 프로퍼티~~

➤ 그 외 -> target 프로퍼티(**현재**)

➤ ~~크로스브라우저에 대응하기 위해 이벤트 발생 요소를 얻기 위해서 함수를 사용~~

➤ getSource() 함수에서 다음과 같은 처리를 확인할 수 있다.

Event.target 프로퍼티 존재 확인

위 성공 여부에 따라 srcElement/target 프로퍼티 중 하나를 호출

- button 프로퍼티의 값은 브라우저에 따라 다름

종류	IE	그 외
마우스 왼쪽 버튼	1	0
마우스 오른쪽 버튼	2	2
중앙 버튼	4	1

DOM – Event

❖ 상위 요소로 이벤트 전달(Bubbling) 확인

- 기본적으로 페이지상에서 발생한 이벤트는 상위 요소에도 전달된다.

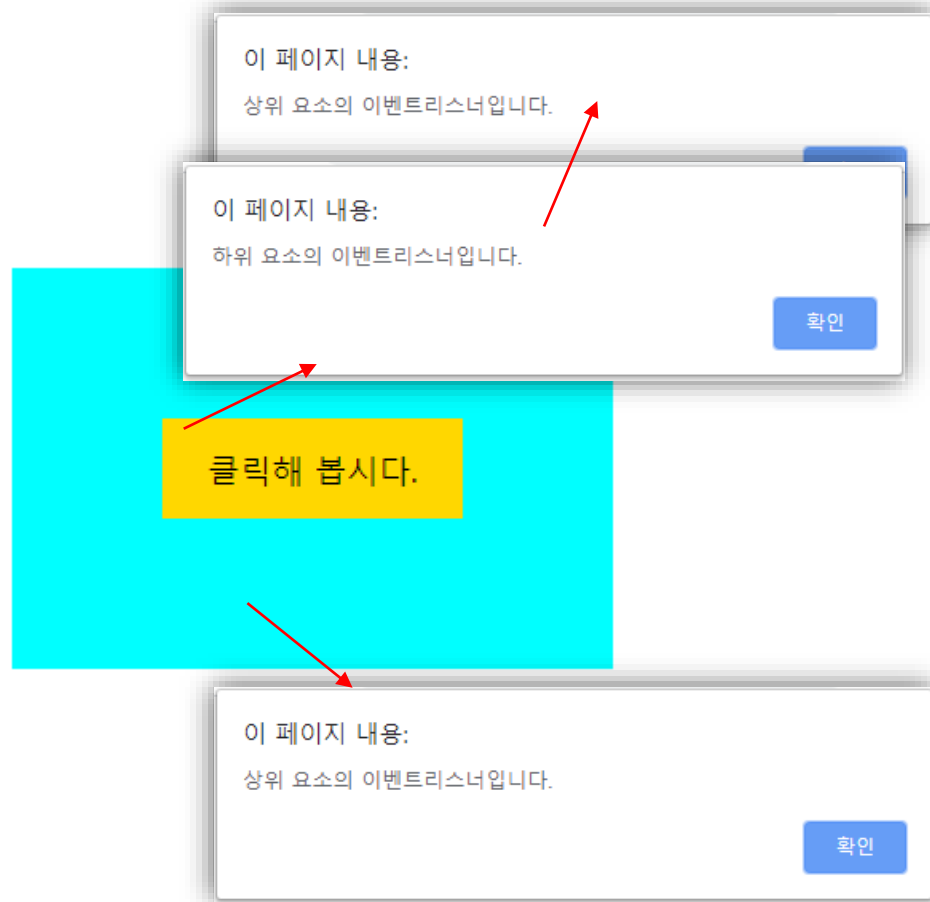
```
<script>
  document.addEventListener("DOMContentLoaded", () => {
    document.getElementById("outer").addEventListener("click", (e) => {
      window.alert("상위 요소의 이벤트 리스너입니다.");
    });

    document.getElementById("inner").addEventListener("click", (e) => {
      window.alert("하위 요소의 이벤트 리스너입니다.");
      // 이벤트 전파 중지
      // e.stopPropagation();
    });
  });
</script>
</head>

<body>
  <div id="outer">
    <div id="inner">클릭해 봅시다.</div>
  </div>
</body>
```

```
<style>
#outer {
  background-color: aqua;
  width: 300px;
  height: 200px;
  display: flex;
  justify-content: center;
  align-items: center;
}

#inner {
  background-color: gold;
  width: 150px;
  height: 50px;
  display: flex;
  justify-content: center;
  align-items: center;
}
```



DOM - Event

❖ 상위 요소로 이벤트 전달 막기

- 앞의 예제에서 상위요소로의 이벤트 전달을 확인할 수 있음
- 이러한 동작의 개념을 거품이 올라가는 것에 빗대어 버블링(Bubbling)이라 한다.
- 버블링은 경우에 따라서 막아야 할 필요가 있는데 이런 경우 명시적으로 억제가 가능하다.

```
<script>
  document.addEventListener('DOMContentLoaded', () => {
    function cancelBubbling(e) {
      e.stopPropagation();
    }

    document.getElementById('outer').addEventListener('click', (e) => {
      window.alert("상위 요소의 이벤트리스너입니다.");
    });

    document.getElementById('inner').addEventListener('click', (e) => {
      window.alert("하위 요소의 이벤트리스너입니다.");
      cancelBubbling(e);
    });
  });
</script>
```

표시된 코드 추가 후 테스트



DOM – Event

❖ 이벤트 상위 전달을 막는 방법은 브라우저에 따라 다를 수 있음

- IE → `cancelBubble` 프로퍼티 사용
- 그 외 → `stopPropagation` 메서드 사용

❖ 동작 과정

- `Event.stopPropagation` 프로퍼티가 존재하는지 확인
- 위 확인 결과에 따라 `stopPropagation` 메서드 또는 `cancelBubble` 프로퍼티를 사용

DOM – Event

❖ 이벤트 디폴트의 동작 막기

- 개발을 하다 보면 submit과 같은 버튼의 기본 이벤트 동작을 제거해야 할 경우도 있다.
- submit이나 a태그와 같이 기본적인 동작이 있는 요소에 대해 이벤트를 제거하는 방법이 있다.
- 이벤트 핸들러는 기본 반환값으로 false를 반환하지만 기본 동작은 취소되지 않는다.

❖ 이를 해결하기 위해 다음과 같은 작업이 필요하다.

- ~~IE -> returnValue 프로퍼티(현재는 의미 없음)~~
- 그 외 -> PreventDefault 메서드

DOM – Event

❖ 예제(이전)

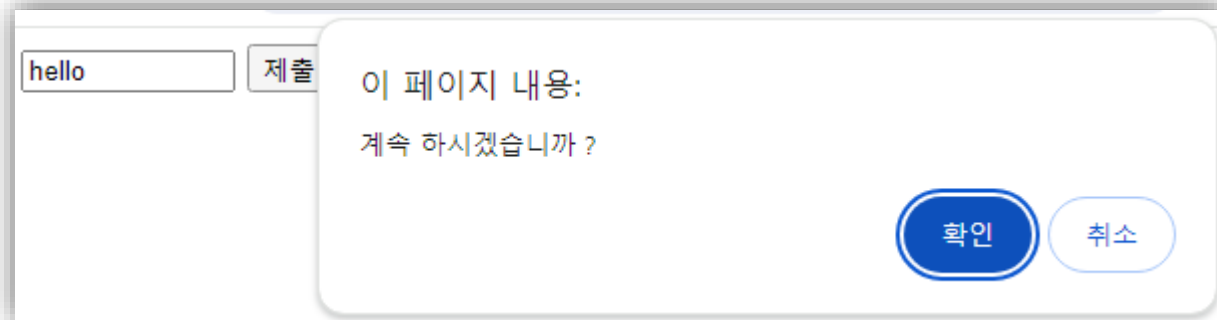
```
<script type="text/javascript" src="listener.js"></script>
<script>
function cancelEvent(e) {
    if (e.preventDefault) {
        alert("c");
        e.preventDefault();
    } else if (window.event) {
        alert("I");

        window.event.returnValue = false;
    }
}

addListener(window, "load", init);

function init() {
    addListener(document.getElementById("fm"), "submit", function (e) {
        if (!window.confirm("계속 하시겠습니까?")) {
            cancelEvent(e);
        }
    });
}
</script>
```

```
<body>
    <form id="fm">
        <input type="text" id="name" size="10" />
        <input type="submit" value="제출" />
    </form>
</body>
```



- 위와 같이 크로스브라우저에 대응하기 위한 함수를 정의하여 이벤트 취소 처리를 외부에 정의하였다.
- 처리 설명
 - Event.preventDefault 프로퍼티 존재여부 확인
 - 위 성공여부에 따라 preventDefault 메서드 호출 또는 returnValue 프로퍼티 사용하여 처리

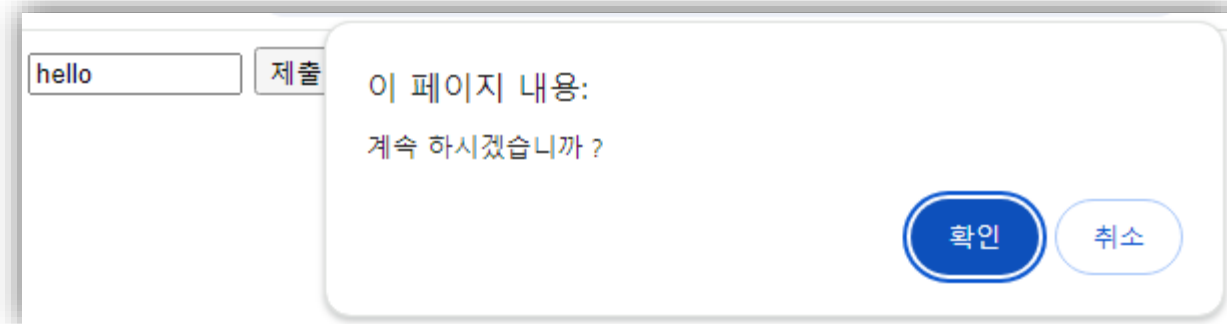
DOM – Event

❖ 예제(현재)

```
<script>
  document.addEventListener("DOMContentLoaded", () => {
    function cancelEvent(e) {
      e.preventDefault();
    }

    document.getElementById("fm").addEventListener("submit", function (e) {
      if (!window.confirm("계속 하시겠습니까?")) {
        cancelEvent(e);
      }
    });
  });
</script>
```

```
<body>
  <form id="fm">
    <input type="text" id="name" size="10" />
    <input type="submit" value="제출" />
  </form>
</body>
```



The screenshot shows a web form with a text input field containing the text "hello" and a submit button labeled "제출". A confirmation dialog box is displayed over the form, asking "이 페이지 내용: 계속 하시겠습니까?" (This page content: Continue?). The dialog has two buttons: "확인" (Confirm) and "취소" (Cancel).

- 모든 주요 브라우저에서 이벤트 취소를 위해 `preventDefault()` 메서드를 사용
- `preventDefault()` 메서드는 이벤트의 기본 동작(여기서는 폼 제출)을 취소하는 역할
- 더 이상 브라우저 호환성을 위해 조건문을 사용할 필요가 없음

e.stopPropagation() vs e.preventDefault()

❖ 비교

메서드	목적	동작	사용 사례
preventDefault()	기본 동작 취소	브라우저의 기본 동작 취소	링크 클릭 시 페이지 이동 취소 폼 제출 시 새로 고침 취소
stopPropagation()	이벤트 전파 중지	이벤트가 부모 요소로 전파 중지	이벤트 버블링/캡처링 중지