

DOM의 Attribute와 관련 메소드에 대한 상세 설명

Document Object Model (DOM)

DOM의 Attribute 개요

Document Object Model(DOM)은 HTML 및 XML 문서의 구조화된 표현을 제공합니다. DOM은 문서의 구조를 트리 형태로 나타내며, 각 노드는 문서의 일부를 나타냅니다. 이 트리 구조는 프로그래밍 언어를 통해 문서의 내용과 구조를 조작할 수 있는 인터페이스를 제공합니다.

Attribute란?

DOM에서 속성(attribute)은 요소(element)의 특성을 정의하며, 이름(name)과 값(value)으로 구성됩니다. 예를 들어, HTML 요소는 "type"과 "value"라는 속성을 가집니다. 속성은 요소에 추가적인 정보를 제공하며, 스타일링, 동작 제어 및 데이터 저장 등의 역할을 합니다.

DOM에서의 Attribute 관련 메소드

DOM에서는 속성을 다루기 위한 다양한 메소드를 제공합니다. 이들 메소드를 사용하여 속성을 추가, 수정, 제거 및 검색할 수 있습니다.

메소드	설명	사용 예시
getAttribute()	지정된 요소의 특정 속성 값을 반환합니다.	let value = element.getAttribute(attributeName);
setAttribute()	특정 요소에 새 속성을 추가하거나 기존 속성을 수정합니다.	element.setAttribute(attributeName, attributeValue);
removeAttribute()	특정 요소에서 지정된 속성을 제거합니다.	element.removeAttribute(attributeName);
hasAttribute()	특정 요소가 지정된 속성을 가지고 있는지 여부를 확인합니다.	let hasAttr = element.hasAttribute(attributeName);
getAttributeNode()	특정 요소의 속성 노드를 반환합니다.	let attrNode = element.getAttributeNode(attributeName);
setAttributeNode()	특정 요소에 속성 노드를 추가하거나 기존 속성을 변경합니다.	element.setAttributeNode(attributeNode);
removeAttributeNode()	특정 요소에서 속성 노드를 제거합니다.	element.removeAttributeNode(attributeNode);

Attribute 객체

속성을 다루는 또 다른 방법은 Attribute 객체를 사용하는 것입니다. Attribute 객체는 속성의 이름과 값을 포함하는 객체입니다. 이 객체는 `getAttributeNode()`, `setAttributeNode()`, `removeAttributeNode()` 메소드와 함께 사용됩니다.

Attribute 객체의 프로퍼티

Attribute 객체는 다음과 같은 주요 프로퍼티를 가지고 있습니다:

- `name`: 속성의 이름을 나타냅니다.
- `value`: 속성의 값을 나타냅니다.
- `specified`: 속성이 명시적으로 지정되었는지 여부를 나타내는 boolean 값입니다.

classList 객체를 사용한 클래스 조작

DOM의 `classList` 객체를 사용하면 요소의 클래스 속성을 쉽게 조작할 수 있습니다. 다음은 주로 사용되는 메서드와 그 설명입니다:

메서드	설명	예제
<code>classList.add(className)</code>	특정 요소에 새로운 클래스를 추가합니다.	<code>element.classList.add('new-class');</code>
<code>classList.remove(className)</code>	특정 요소에서 클래스를 제거합니다.	<code>element.classList.remove('old-class');</code>
<code>classList.toggle(className)</code>	특정 요소에 클래스가 존재하면 제거하고, 존재하지 않으면 추가합니다.	<code>element.classList.toggle('active-class');</code>
<code>classList.contains(className)</code>	특정 요소가 특정 클래스를 포함하고 있는지 여부를 반환합니다. (boolean 값)	<code>element.classList.contains('some-class');</code> // true 또는 false 반환
<code>classList.replace(oldClassName, newClassName)</code>	특정 요소에서 기존 클래스를 새로운 클래스로 교체합니다.	<code>element.classList.replace('old-class', 'new-class');</code>
<code>classList.item(index)</code>	특정 인덱스 위치의 클래스를 반환합니다.	<code>element.classList.item(0);</code> // 첫 번째 클래스 반환

Dataset에 대한 설명과 관련 메서드

Dataset은 HTMLElement의 속성 중 하나로, 각 요소에 사용자 정의 데이터를 저장할 수 있도록 해줍니다. HTML5에서 도입된 data-* 속성을 통해 요소에 데이터를 추가할 수 있으며, dataset 객체를 사용하여 이러한 데이터를 JavaScript에서 쉽게 접근하고 조작할 수 있습니다.

Dataset 속성

dataset 속성을 통해 data-* 속성에 접근할 수 있습니다. 예를 들어, `` 요소가 있을 때, `element.dataset.user`는 "John"을, `element.dataset.age`는 "30"을 반환합니다.

Dataset 예제

HTML 요소에 data-* 속성을 추가하고 이를 JavaScript에서 접근하는 방법을 예제로 살펴보겠습니다.

JavaScript

// 요소 선택

```
const element = document.getElementById('userProfile');
```

// 데이터 읽기

```
console.log(element.dataset.user); // "John"
```

```
console.log(element.dataset.age); // "30"
```

// 데이터 설정

```
element.dataset.user = 'Jane';
```

```
console.log(element.dataset.user); // "Jane"
```

// 데이터 삭제

```
delete element.dataset.age;
```

```
console.log(element.dataset.age); // undefined
```

// 데이터 존재 여부 확인

```
console.log('user' in element.dataset); // true
```

```
console.log('age' in element.dataset); // false
```

DOM 조작 과제 안내

문제 1. 특정 요소의 속성(attribute) 읽기

- 문제 설명:

HTML 파일 내 `

</div>` 요소에서 `data-info` 속성의 값을 읽어 콘솔에 출력합니다.

- 요구 사항:

1. `document.getElementById`로 요소를 선택합니다.
2. `getAttribute` 메서드를 이용해 `data-info`의 값을 읽습니다.

- 예상 결과: 콘솔에 "Hello, World!"를 출력.

문제 2. 특정 요소의 속성(attribute) 수정하기

- 문제 설명:

HTML 파일 내 `![old image](old.jpg)>` 요소에서 버튼 클릭 시 `src` 속성의 값을 `"new.jpg"`로 변경합니다.

- 요구 사항:

1. 버튼 클릭 이벤트를 설정합니다.
2. `setAttribute` 메서드를 사용하여 `src` 속성을 변경합니다.

- 확장: 변경 후, 이미지가 새 소스 이미지를 정상적으로 로딩하는 지 확인합니다.

문제 3. 여러 요소에 공통 속성(attribute) 추가하기

- 문제 설명:

HTML에 클래스 `"item"`을 가진 여러 `- ` 요소가 있다고 가정합니다. 각 요소에 순차적으로 `data-index` 속성을 부여합니다. (예: 첫 번째 `- `는 `data-index="0"`, 두 번째는 `data-index="1"` 등)

- 요구 사항:

1. `document.querySelectorAll`을 이용하여 `NodeList`를 가져옵니다.
 2. 반복문을 사용해 각 요소마다 `setAttribute`로 `data-index` 추가합니다.
- 확장: 생성된 인덱스를 HTML 내부에 표시하도록 텍스트 노드를 추가해도 좋습니다.

문제 4. 특정 속성(attribute) 삭제하기

- 문제 설명:

HTML 문서 내 클래스 `"tooltip"`을 가진 모든 요소들이 `title` 속성을 가지고 있습니다. 해당 요소들로부터 모두 `title` 속성을 삭제합니다.

- 요구 사항:

1. `document.querySelectorAll`로 대상 요소들을 선택합니다.
 2. 반복문을 돌며 각 요소에 대해 `removeAttribute` 메서드를 호출합니다.
- 확장: 삭제 전후의 요소를 콘솔에 출력하여 변화를 확인합니다.

문제 5. 클래스 추가하기 (`classList.add`)

- 문제 설명:

HTML 파일 내 `<div id="status">상태 메시지</div>` 요소에 `"active"` 클래스를 추가합니다. 단, 해당 클래스가 이미 적용되어 있을 경우 중복 추가하지 않습니다.

- 요구 사항:

1. `document.getElementById`로 요소를 선택합니다.
 2. `classList.contains`로 현재 클래스 포함 여부를 확인 후, 없다면 `classList.add`로 추가합니다.
- 확장: 버튼 클릭 시 이 기능을 실행하게 만들어 동적 반응을 구현합니다.

문제 6. 클래스 제거하기 (`classList.remove`)

- 문제 설명:

HTML 파일 내 ``<p id="message" class="notice inactive">안내 메시지</p>`` 요소에서 `"inactive"` 클래스를 제거합니다.

- 요구 사항:

1. 요소를 선택한 후, `classList.remove`를 호출하여 `"inactive"` 클래스를 삭제합니다.

- 확장: 클래스 제거 전후의 클래스 목록을 콘솔에 출력해 비교합니다.

문제 7. 클래스 토글 기능 구현 (`classList.toggle`)

- 문제 설명:

`<div id="card" class="highlighted"></div>` 요소에 대해 버튼 클릭 시 `"highlighted"` 클래스를 토글합니다.

- 요구 사항:

1. 버튼 클릭 이벤트 핸들러 내에서 `classList.toggle`을 호출합니다.

2. 해당 클래스가 없으면 추가, 있으면 제거하는 로직 확인.

- 확장: 상태 변화에 따라 배경색 등이 바뀌는 CSS 효과를 부여해 시각적으로 확인 가능하도록 합니다.

문제 8. 클래스 포함 여부 확인 (`classList.contains`)

- 문제 설명:

HTML 파일 내 ```` 요소에서 `"feature"` 클래스 포함 여부를 확인합니다.

- 요구 사항:

1. `classList.contains` 메서드를 사용합니다.

2. 포함되어 있으면 콘솔에 "Feature included!"를, 아니면 "Feature not included!"를 출력합니다.

- 확장: 여러 클래스를 대상으로 조건 분기를 확장해보세요.

문제 9. 데이터 속성(data-attributes) 조작하기

- 문제 설명:

HTML 파일 내 ``<button id="dataButton" data-state="off"></button>`` 요소에서 버튼 클릭 시 ``data-state`` 값을 ``"off"``에서 ``"on"``으로, ``"on"`이면 다시 ``"off"``로 변경하는 기능을 구현합니다.

- 요구 사항:

1. ``dataset`` 혹은 ``getAttribute``와 ``setAttribute``를 사용하여 속성 값을 읽고 수정합니다.
 2. 클릭 이벤트에 따라 값이 전환되는 로직을 작성합니다.
- 확장: 전환된 값을 버튼 내부 텍스트로 출력해 상태 변화를 시각적으로 확인합니다.

문제 10. 이벤트를 통한 DOM 속성과 클래스 조작 종합 실습

- 문제 설명:

아래 HTML 요소들을 기반으로 사용자의 입력에 따라 DOM을 조작하는 기능을 구현합니다.

```
```html
```

```
<input type="text" id="nameInput" placeholder="이름을 입력하세요">
```

```
<p id="warning" class="hidden">입력 값이 비어 있습니다!</p>
```

```
<button id="submitBtn">제출</button>
```

```
```
```

- 요구 사항:

1. 버튼 클릭 이벤트 발생 시, ``#nameInput``의 입력 값이 비어 있는지를 확인합니다.
2. 만약 값이 비어 있다면:
 - ``#warning`` 요소에서 ``"hidden"`` 클래스를 제거하여 경고 메시지를 표시합니다.
 - ``#nameInput``에 ``aria-invalid="true"` 속성을 추가합니다.
3. 값이 있다면:
 - ``#warning`` 요소에 ``"hidden"`` 클래스를 추가하여 경고를 감춥니다.
 - ``aria-invalid`` 속성을 제거합니다.

- 확장: 실시간 입력 이벤트(`keyup` 등)를 추가하여 입력 검증을 동적으로 진행할 수 있도록 구현해 보세요.

문제 11. 클릭 카운터 버튼 구현하기

문제 설명

HTML에 다음과 같은 버튼 요소가 존재합니다.

```
```html
```

```
<button id="clickCounter" data-count="0">Click Me!</button>
```

```
```
```

버튼을 클릭할 때마다 `data-count` 속성 값이 1씩 증가하고, 버튼 텍스트가 "Click count: X" 형태로 업데이트되도록 구현하세요.

요구 사항

1. 버튼 요소에 클릭 이벤트 핸들러를 등록합니다.
2. `dataset` 또는 `getAttribute`와 `setAttribute`를 사용해 `data-count`를 읽고 업데이트합니다.
3. 업데이트 후 버튼 텍스트를 동기화하여 표시합니다.

확장

- 페이지에 여러 개의 클릭 카운터 버튼이 있을 경우, 각각 독립적으로 동작하도록 처리해 보세요.

문제 12. 상태 토글 동작 구현하기

문제 설명

HTML에 다음과 같은 요소가 있습니다.


```
```html
```

```
<div id="toggleDiv" data-state="off">상태: off</div>
```

```
```
```

버튼 클릭 시 `data-state` 데이터 속성이 "off"에서 "on" 또는 "on"에서 "off"로 토글되며, 해당 상태에 맞춰 요소의 텍스트도 "상태: on" / "상태: off"로 변경되도록 코드를 작성하세요.

요구 사항

1. 버튼 클릭 이벤트 핸들러를 추가합니다.
2. `dataset`이나 attribute 메서드를 사용해 현재 상태를 읽습니다.
3. 상태를 반전시키고 해당 상태를 요소의 텍스트와 `data-state` 속성에 반영합니다.

확장

- 상태 변경 시, 각 상태에 맞는 배경색 또는 CSS 클래스가 적용되도록 스타일링해 보세요.

```
---
```

문제 13. 데이터 속성 기반 필터링 기능 구현

문제 설명

HTML에 다음과 같이 여러 개의 `

` 요소가 있습니다.

```
```html
```

```
<div class="item" data-category="fruit">Apple</div>
```

```
<div class="item" data-category="vegetable">Carrot</div>
```

```
<div class="item" data-category="fruit">Banana</div>
```

```
<div class="item" data-category="vegetable">Lettuce</div>
```

```
```
```

드롭다운 메뉴를 통해 "fruit" 또는 "vegetable"를 선택하면, 선택한 카테고리에 해당하는 요소들만 화면에 표시되고, 그 외의 요소는 숨겨지도록 코드를 작성하세요.

요구 사항

1. `document.querySelectorAll`로 모든 `.item` 요소들을 선택합니다.
2. `<select>` 요소의 `change` 이벤트를 사용해 선택된 카테고리를 받아옵니다.
3. 각 요소의 `data-category` 값을 확인하여 조건에 맞게 `style.display` 속성을 `"block"` 또는 `"none"`으로 설정합니다.

확장

- 선택된 항목에 대해 시각적인 강조 효과(예: 배경색 변경)를 주는 CSS 클래스를 추가해 보세요.

문제 14. 동적 글꼴 크기 적용하기

문제 설명

다음과 같이 여러 `<p>` 요소가 각각 `data-font-size` 데이터를 가지고 있습니다.

```
```html
```

```
<p class="text" data-font-size="16">Text 1</p>
```

```
<p class="text" data-font-size="20">Text 2</p>
```

```
<p class="text" data-font-size="24">Text 3</p>
```

```
```
```

페이지 로드시 각 요소의 `data-font-size` 값을 읽어 해당하는 글자 크기로 스타일을 적용하는 JavaScript 코드를 작성하세요.

요구 사항

1. `document.querySelectorAll`을 사용해 모든 `.text` 요소들을 선택합니다.
2. 각 요소에서 `data-font-size` 값을 읽어옵니다 (단위 없이 숫자만 있을 경우, `"px"` 등 단위를 붙이세요).
3. 해당 값을 요소의 `style.fontSize` 속성에 적용합니다.

확장

- 이후 동적으로 요소가 추가될 때, 동일한 로직으로 새 요소에 글꼴 크기가 적용되도록 재사용 가능한 함수를 만들어 보세요.

문제 15. 리스트 요소 정렬 기능 구현하기

문제 설명

HTML 내에 아래와 같이 데이터 속성을 활용한 `- ` 요소들이 주어집니다.

```
```html
```

```
<ul id="itemList">
```

```
 <li data-order="3">Item 3
```

```
 <li data-order="1">Item 1
```

```
 <li data-order="2">Item 2
```

```

```

```
```
```

버튼 클릭 시, 각 `- ` 요소의 `data-order` 값을 이용하여 오름차순으로 정렬한 후, 다시 `#itemList`에 순서대로 배치하는 코드를 작성하세요.

요구 사항

1. `document.querySelectorAll`을 활용해 모든 `- ` 요소를 선택합니다.
- 2. 각 요소의 `data-order` 값을 기준으로 오름차순 정렬합니다.
- 3. 정렬된 요소들을 `appendChild` 또는 유사한 DOM 조작 메서드를 사용해 다시 추가합니다.

확장

- 내림차순 정렬 및 정렬 기준을 사용자가 선택할 수 있는 옵션을 추가해 보세요.

정답 확인

문제 1. 특정 요소의 attribute 값 읽기

파일명: `problem1.html`

```
``html

<!DOCTYPE html>

<html lang="ko">

<head>

  <meta charset="UTF-8">

  <title>문제 1: attribute 값 읽기</title>

  <style>

    body { font-family: sans-serif; padding: 20px; }

  </style>

</head>

<body>

  <!-- data-info attribute를 가진 요소 -->

  <div id="myElement" data-info="Hello, World!">이곳의 data-info를 콘솔에 출력합니다.</div>

  <script>

    // 요소 선택 및 data-info attribute 값 읽기

    const myElement = document.getElementById('myElement');

    const info = myElement.getAttribute('data-info');

    console.log("data-info 값:", info);

  </script>

</body>

</html>

``
```

문제 2. 특정 요소의 attribute 수정하기

파일명: `problem2.html`

```
``html

<!DOCTYPE html>

<html lang="ko">

<head>

  <meta charset="UTF-8">

  <title>문제 2: attribute 수정하기</title>

  <style>

    body { font-family: sans-serif; padding: 20px; }

    img { display: block; margin-bottom: 15px; max-width: 300px; }

    button { padding: 8px 16px; cursor: pointer; }

  </style>

</head>

<body>

  <!-- 초기 이미지와 버튼 -->

  

  <button id="changeButton">이미지 변경하기</button>

  <script>

    const image = document.getElementById('image');

    const changeButton = document.getElementById('changeButton');

    changeButton.addEventListener('click', function() {

      // setAttribute를 이용하여 src 변경
```

```
        image.setAttribute('src', 'new.jpg');

        console.log("이미지 src가 new.jpg로 변경되었습니다.");

    });

</script>

</body>

</html>

...

```

> 참고: 실제 이미지 파일이 없다면 브라우저에 broken image 아이콘이 나타날 수 있습니다.

문제 3. 여러 요소에 공통 attribute 추가하기

파일명: `problem3.html`

```
``html

<!DOCTYPE html>

<html lang="ko">

<head>

    <meta charset="UTF-8">

    <title>문제 3: 공통 attribute 추가하기</title>

    <style>

        body { font-family: sans-serif; padding: 20px; }

        ul { list-style: none; padding: 0; }

        li { margin: 5px 0; padding: 5px; border: 1px solid #ccc; }

    </style>

</head>

<body>

```

```
<ul>
```

```
  <li class="item">첫 번째 아이템</li>
```

```
  <li class="item">두 번째 아이템</li>
```

```
  <li class="item">세 번째 아이템</li>
```

```
</ul>
```

```
<script>
```

```
  // 모든 .item 요소 선택 및 인덱스 부여
```

```
  const items = document.querySelectorAll('.item');
```

```
  items.forEach(function(item, index) {
```

```
    item.setAttribute('data-index', index);
```

```
    // 각 아이템에 인덱스 값을 보여주기 위해 텍스트를 추가
```

```
    item.textContent += " (data-index: " + index + ")";
```

```
  });
```

```
</script>
```

```
</body>
```

```
</html>
```

```
---
```

```
---
```

문제 4. 특정 attribute 삭제하기

파일명: `problem4.html`

```
```html
```

```
<!DOCTYPE html>
```

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

```
<head>
```

```
<meta charset="UTF-8">

<title>문제 4: attribute 삭제하기</title>

<style>

 body { font-family: sans-serif; padding: 20px; }

 .tooltip { margin: 5px 0; padding: 5px; border: 1px dashed #888; }

</style>

</head>

<body>

 <div class="tooltip" title="툴팁 1">툴팁 요소 1</div>

 <div class="tooltip" title="툴팁 2">툴팁 요소 2</div>

 <div class="tooltip" title="툴팁 3">툴팁 요소 3</div>

 <script>

 // 모든 .tooltip 요소 선택 후 title attribute 삭제

 const tooltips = document.querySelectorAll('.tooltip');

 tooltips.forEach(function(elem) {

 elem.removeAttribute('title');

 });

 console.log("모든 tooltip 요소에서 title attribute가 삭제되었습니다.");

 </script>

</body>

</html>

...

```

### 문제 5. 클래스 추가하기 (classList.add)

파일명: `problem5.html`



```
``html

<!DOCTYPE html>

<html lang="ko">

<head>

 <meta charset="UTF-8">

 <title>문제 5: 클래스 추가하기</title>

 <style>

 body { font-family: sans-serif; padding: 20px; }

 /* active 클래스가 적용되면 배경색 변화 */

 .active { background-color: #def; }

 #status { padding: 10px; border: 1px solid #ccc; }

 </style>

</head>

<body>

 <div id="status">상태 메시지</div>

 <button id="addClassButton">active 클래스 추가</button>

 <script>

 const statusDiv = document.getElementById('status');

 const addClassButton = document.getElementById('addClassButton');

 addClassButton.addEventListener('click', function() {

 // 이미 active 클래스가 없다면 추가

 if (!statusDiv.classList.contains('active')) {

 statusDiv.classList.add('active');

 console.log("active 클래스가 추가되었습니다.");

 } else {
```

```
 console.log("이미 active 클래스가 존재합니다.");
 }

});

</script>

</body>

</html>

...

```

### ### 문제 6. 클래스 제거하기 (classList.remove)

파일명: `problem6.html`

```
``html

<!DOCTYPE html>

<html lang="ko">

<head>

 <meta charset="UTF-8">

 <title>문제 6: 클래스 제거하기</title>

 <style>

 body { font-family: sans-serif; padding: 20px; }

 #message { padding: 10px; border: 1px solid #ccc; }

 /* inactive 클래스가 적용되면 텍스트 색상 회색 */

 .inactive { color: #999; }

 </style>

</head>

<body>

 <p id="message" class="notice inactive">안내 메시지 (inactive 클래스가 제거될 예정)</p>
```

```
<button id="removeClassButton">inactive 클래스 제거</button>
```

```
<script>
```

```
const message = document.getElementById('message');
```

```
const removeClassButton = document.getElementById('removeClassButton');
```

```
removeClassButton.addEventListener('click', function() {
```

```
 // inactive 클래스를 제거
```

```
 message.classList.remove('inactive');
```

```
 console.log("inactive 클래스가 제거되었습니다.");
```

```
});
```

```
</script>
```

```
</body>
```

```
</html>
```

```
...
```

```

```

### 문제 7. 클래스 토글 기능 구현 (classList.toggle)

파일명: `problem7.html`

```
```html
```

```
<!DOCTYPE html>
```

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

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>문제 7: 클래스 토글</title>
```

```
  <style>
```

```
body { font-family: sans-serif; padding: 20px; }

#card { width: 150px; height: 150px; border: 1px solid #333; line-height: 150px; text-align: center;
margin-bottom: 15px; }

/* highlighted 클래스가 있으면 배경색 적용 */

.highlighted { background-color: #ffd; }

</style>

</head>

<body>

<div id="card" class="highlighted">Card</div>

<button id="toggleButton">클래스 토글</button>


<script>

const card = document.getElementById('card');

const toggleButton = document.getElementById('toggleButton');


toggleButton.addEventListener('click', function() {

    // highlighted 클래스가 있으면 제거, 없으면 추가

    card.classList.toggle('highlighted');

    console.log("highlighted 클래스가 토글되었습니다.");

});

</script>

</body>

</html>

...

---
```

문제 8. 클래스 포함 여부 확인 (classList.contains)

파일명: `problem8.html`

```
``html

<!DOCTYPE html>

<html lang="ko">

<head>

  <meta charset="UTF-8">

  <title>문제 8: 클래스 포함 여부 확인</title>

  <style>

    body { font-family: sans-serif; padding: 20px; }

    #badge { padding: 10px; border: 1px solid #333; display: inline-block; }

  </style>

</head>

<body>

  <span id="badge" class="new feature">Badge</span>

  <button id="checkButton">클래스 확인</button>

  <script>

    const badge = document.getElementById('badge');

    const checkButton = document.getElementById('checkButton');

    checkButton.addEventListener('click', function() {

      // feature 클래스 포함 여부 확인 후 콘솔 출력

      if (badge.classList.contains('feature')) {

        console.log("Feature included!");

      } else {

        console.log("Feature not included!");

      }

    });
```

```
</script>
```

```
</body>
```

```
</html>
```

```
...
```

```
---
```

문제 9. 데이터 속성 (data-attributes) 조작하기

파일명: `problem9.html`

```
```html
```

```
<!DOCTYPE html>
```

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

```
<head>
```

```
 <meta charset="UTF-8">
```

```
 <title>문제 9: 데이터 속성 조작하기</title>
```

```
 <style>
```

```
 body { font-family: sans-serif; padding: 20px; }
```

```
 button { padding: 8px 16px; cursor: pointer; }
```

```
 </style>
```

```
</head>
```

```
<body>
```

```
 <!-- data-state attribute를 가진 버튼 -->
```

```
 <button id="dataButton" data-state="off">State: off</button>
```

```
 <script>
```

```
 const dataButton = document.getElementById('dataButton');
```

```

dataButton.addEventListener('click', function() {

 // getAttribute를 사용하여 현재 data-state 읽기

 let currentState = dataButton.getAttribute('data-state');

 // 토글: 'off'이면 'on', 'on'이면 'off'

 let newState = (currentState === 'off') ? 'on' : 'off';

 // setAttribute로 data-state 업데이트 후 버튼 텍스트 변경

 dataButton.setAttribute('data-state', newState);

 dataButton.textContent = "State: " + newState;

});

</script>

</body>

</html>

...

```

### 문제 10. 이벤트를 통한 DOM 속성과 클래스 조작 종합 실습

파일명: `problem10.html`

```

<<html

<!DOCTYPE html>

<html lang="ko">

<head>

 <meta charset="UTF-8">

 <title>문제 10: DOM 속성과 클래스 조작</title>

 <style>

 body { font-family: sans-serif; padding: 20px; }

 #warning { color: red; margin-top: 10px; }
```

```
.hidden { display: none; }

input { padding: 6px; margin-right: 10px; }

button { padding: 6px 12px; cursor: pointer; }

</style>

</head>

<body>

 <input type="text" id="nameInput" placeholder="이름을 입력하세요">

 <button id="submitBtn">제출</button>

 <p id="warning" class="hidden">입력 값이 비어 있습니다!</p>

 <script>

 const nameInput = document.getElementById('nameInput');

 const warning = document.getElementById('warning');

 const submitBtn = document.getElementById('submitBtn');

 submitBtn.addEventListener('click', function() {

 // 입력 값이 비어있는지 확인 (공백 제거)

 if (nameInput.value.trim() === "") {

 // 경고 메시지를 보이게 하고, aria-invalid 속성 추가

 warning.classList.remove('hidden');

 nameInput.setAttribute('aria-invalid', 'true');

 console.log("입력 값이 없습니다.");

 } else {

 // 경고 메시지 숨기고, aria-invalid 속성 제거

 warning.classList.add('hidden');

 nameInput.removeAttribute('aria-invalid');

 console.log("입력 값이 확인되었습니다: " + nameInput.value);

 }

 });

 </script>

</body>

</html>
```



```
});
</script>
</body>
</html>
```

---

### 문제 11. 클릭 카운터 버튼 구현하기

파일명: `problem11.html`

```
``html
<!DOCTYPE html>
<html lang="ko">
<head>
 <meta charset="UTF-8">
 <title>문제 11: 클릭 카운터 버튼 (dataset)</title>
 <style>
 button {
 padding: 10px 20px;
 font-size: 16px;
 cursor: pointer;
 }
 </style>
</head>
<body>
 <!-- 데이터 속성(data-count)을 dataset으로 활용 -->
 <button id="clickCounter" data-count="0">Click Me!</button>
```

```

<script>

const clickCounter = document.getElementById('clickCounter');

clickCounter.addEventListener('click', function() {

 // dataset을 사용해 data-count 값 읽기 및 정수 변환

 let currentCount = parseInt(this.dataset.count);

 // 클릭 시 1 증가

 currentCount++;

 // dataset의 data-count와 버튼 텍스트 업데이트

 this.dataset.count = currentCount;

 this.textContent = "Click count: " + currentCount;

});

</script>

</body>

</html>

...

```

---

### 문제 12. 상태 토글 동작 구현하기

파일명: `problem12.html`

```

`html

<!DOCTYPE html>

<html lang="ko">

<head>

 <meta charset="UTF-8">

 <title>문제 12: 상태 토글 (dataset)</title>

```

```
<style>
```

```
#toggleDiv {
 padding: 15px;
 font-size: 18px;
 text-align: center;
 border: 1px solid #ccc;
 margin-bottom: 10px;
}

/* 상태에 따른 배경색 스타일 */

.on {
 background-color: lightgreen;
}

.off {
 background-color: lightcoral;
}

button {
 padding: 8px 16px;
 cursor: pointer;
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<!-- dataset을 활용한 data-state -->
```

```
<div id="toggleDiv" data-state="off" class="off">상태: off</div>
```

```
<button id="toggleButton">Toggle 상태</button>
```

```
<script>
```

```
 const toggleDiv = document.getElementById('toggleDiv');
```

```
const toggleButton = document.getElementById('toggleButton');
```

```
toggleButton.addEventListener('click', function() {
 // dataset을 사용하여 현재 상태 읽기
 const currentState = toggleDiv.dataset.state;
 // 토글: 'off'이면 'on', 'on'이면 'off'
 const newState = (currentState === 'off') ? 'on' : 'off';

 // dataset과 텍스트 업데이트
 toggleDiv.dataset.state = newState;
 toggleDiv.textContent = "상태: " + newState;

 // CSS 클래스 업데이트 (배경색 전환)
 toggleDiv.classList.remove('on', 'off');
 toggleDiv.classList.add(newState);
});
```

```
</script>
```

```
</body>
```

```
</html>
```

```
...
```

```

```

### 문제 13. 데이터 속성 기반 필터링 기능 구현

파일명: `problem13.html`

```
```html
```

```
<!DOCTYPE html>
```

```
<html lang="ko">

<head>

  <meta charset="UTF-8">

  <title>문제 13: 데이터 속성 기반 필터링 (dataset)</title>

  <style>

    .item {

      margin: 5px;

      padding: 10px;

      border: 1px solid #333;

    }

    .hidden {

      display: none;

    }

    /* 선택된 항목에 강조 효과 (확장 아이디어) */

    .highlight {

      border-color: blue;

      background-color: #e0f7ff;

    }

  </style>

</head>

<body>

  <h1>아이템 필터링</h1>

  <label for="categorySelect">카테고리 선택: </label>

  <select id="categorySelect">

    <option value="all">모두 보기</option>

    <option value="fruit">과일</option>

    <option value="vegetable">야채</option>

  </select>
```

```
<div class="item" data-category="fruit">Apple</div>
```

```
<div class="item" data-category="vegetable">Carrot</div>
```

```
<div class="item" data-category="fruit">Banana</div>
```

```
<div class="item" data-category="vegetable">Lettuce</div>
```

```
<script>
```

```
const categorySelect = document.getElementById('categorySelect');
```

```
const items = document.querySelectorAll('.item');
```

```
categorySelect.addEventListener('change', function() {
```

```
    const selectedCategory = this.value;
```

```
    items.forEach(function(item) {
```

```
        const itemCategory = item.dataset.category;
```

```
        // 'all' 선택 시 모두 보이고, 선택된 카테고리일 때 보임
```

```
        if (selectedCategory === "all" || itemCategory === selectedCategory) {
```

```
            item.style.display = "block";
```

```
            item.classList.add('highlight');
```

```
        } else {
```

```
            item.style.display = "none";
```

```
            item.classList.remove('highlight');
```

```
        }
```

```
    });
```

```
});
```

```
</script>
```

```
</body>
```

```
</html>
```

```
...
```

문제 14. 동적 글꼴 크기 적용하기

파일명: `problem14.html`

```
```html
<!DOCTYPE html>
<html lang="ko">
<head>
 <meta charset="UTF-8">
 <title>문제 14: 동적 글꼴 크기 적용 (dataset)</title>
 <style>
 p {
 margin: 10px 0;
 }
 </style>
</head>
<body>
 <p class="text" data-font-size="16">Text 1</p>
 <p class="text" data-font-size="20">Text 2</p>
 <p class="text" data-font-size="24">Text 3</p>

 <script>
 // 페이지 로드시 모든 .text 요소에 대해 data-font-size 값을 적용
 const texts = document.querySelectorAll('.text');
 texts.forEach(function(textElement) {
 const fontSize = textElement.dataset.fontSize;
```

```
// 숫자만 있을 경우 "px" 단위를 붙여서 적용
textElement.style.fontSize = fontSize + "px";

});

</script>

</body>

</html>

...

```

### 문제 15. 리스트 요소 정렬 기능 구현하기

파일명: `problem15.html`

```
``html

<!DOCTYPE html>

<html lang="ko">

<head>

 <meta charset="UTF-8">

 <title>문제 15: 리스트 요소 정렬 (dataset)</title>

 <style>

 ul {

 list-style: none;

 padding: 0;

 }

 li {

 padding: 8px;

 border: 1px solid #ccc;

 margin: 3px 0;

 }

 </style>

</head>

<body>

 1. 첫 번째 요소는 정렬되지 않습니다.

 2. 두 번째 요소는 정렬됩니다.

 3. 세 번째 요소는 정렬됩니다.

 4. 네 번째 요소는 정렬됩니다.

 5. 다섯 번째 요소는 정렬됩니다.

</body>

</html>
```



```
}

button {

 padding: 8px 16px;

 cursor: pointer;

 margin-top: 10px;

}

</style>

</head>

<body>

 <ul id="itemList">

 <li data-order="3">Item 3

 <li data-order="1">Item 1

 <li data-order="2">Item 2

 <button id="sortButton">정렬하기</button>

 <script>

 const sortButton = document.getElementById('sortButton');

 sortButton.addEventListener('click', function() {

 const itemList = document.getElementById('itemList');

 // 모든 li 요소를 배열로 변환

 const itemsArray = Array.from(itemList.querySelectorAll('li'));

 // data-order를 dataset으로 읽어 오름차순 정렬

 itemsArray.sort(function(a, b) {

 return parseInt(a.dataset.order) - parseInt(b.dataset.order);

 });

 });

 </script>

</body>

</html>
```

```
// 기존 li 요소들을 지우고 정렬된 순서대로 재삽입
```

```
itemList.innerHTML = '';
```

```
itemsArray.forEach(function(item) {
```

```
 itemList.appendChild(item);
```

```
});
```

```
});
```

```
</script>
```

```
</body>
```

```
</html>
```

```
'''
```