FRONT-END COMPONENT GUIDE

board-page / board-button 컴포넌트 사용 가이드

게시판 페이징 & 하단 버튼 영역 마크업 규칙

본 문서는
게시판 페이징 컴포넌트(`board-page`)와 하단 버튼 레이아웃 컴포넌트(`board-button`)의
마크업 규칙을 정리한 개발 연동 가이드입니다.

문서 버전	v1.0
작성일	2026-08-10
준수 기준	**행정안전부 디지털 정부서비스 UI/UX 가이드라인(KRDS)**

페이징 컴포넌트는 **이전 / 다음 2개 컨트롤 + 첫 페이지 + 9개 번호 블록 + 마지막 페이지** 구조를 표준으로 합니다. 개발 시 판단이 필요한 **5가지 상황**과 각각의 예시 화면은 1-4를 참고하십시오.

목차

> **개발 착수 전 필독 — 2장** `board-button` 은 `:has()` 선택자 기반으로 동작합니다. **자식 요소를** `<div>` **로 감싸는지 여부만으로 정렬 결과가 바뀌므로**, 템플릿 엔진에서 조건부로 버튼을 렌더링할 때 빈 `<div>` 가 남거나 사라지지 않도록 주의해 주십시오. 상세 내용은 2-4를 참고하십시오.

1 board-page · 게시판 페이징

목록형 화면 하단의 페이지 번호 네비게이션 컴포넌트입니다. **마크업 구조 자체가 사용법**이며 별도의 초기화 스크립트나 상태 클래스가 필요하지 않습니다.

1-1. 표준 마크업

아래는 **총 271페이지 중 1페이지를 보고 있는 상태**의 출력 결과입니다. 이 구조를 프로젝트 전체의 페이징 표준으로 사용합니다.

HTML · 표준

```
<div class="board-page">
    <a href="javascript:void(0)" class="dir-prev">이전</a>

    <div>
        <strong><span class="sr-only">현재페이지</span>1</strong>
        <a href="javascript:void(0)" class="page-link">2</a>
        <a href="javascript:void(0)" class="page-link">3</a>
        <a href="javascript:void(0)" class="page-link">4</a>
        <a href="javascript:void(0)" class="page-link">5</a>
        <a href="javascript:void(0)" class="page-link">6</a>
        <a href="javascript:void(0)" class="page-link">7</a>
        <a href="javascript:void(0)" class="page-link">8</a>
        <a href="javascript:void(0)" class="page-link">9</a>
        <span class="page-dot"></span>
        <a class="page-link" href="javascript:void(0)" title="마지막 페이지">271</a>
    </div>

    <a href="javascript:void(0)" class="dir-next">다음</a>
</div>
```

화면 출력

게시판 목록 화면 하단 · 총 271페이지 / 현재 1페이지

2711	2026년 상반기 통계자료 공표 안내	2026-05-29	36
2710	시스템 정기 점검 일정 안내	2026-05-28	112
2709	자료 이용 방법 변경 사항 공지	2026-05-27	87

‹ 이전 1 2 3 4 5 6 7 8 9 … 271 다음 ›

1-2. 구성 요소

요소	역할	비고
`.dir-prev` `.dir-next`	이전 / 다음 블록 이동	**블록 단위(9페이지)로 이동**합니다. 좌우 이동 컨트롤은 이 2개뿐이며 맨처음 / 맨끝 버튼은 사용하지 않습니다.
내부 `<div>`	번호 그룹 컨테이너	**필수 요소입니다.** 번호 간 간격과 화면 크기별 배치가 모두 이 요소에 종속되어 있으므로 생략할 수 없습니다.
`<strong>`	**현재 페이지**	클래스가 아닌 **태그로 구분**합니다. `<a>`가 아니므로 링크로 출력하지 마십시오. 번호 그룹 내에 **항상 정확히 1개**만 존재합니다.
`.page-link`	이동 가능한 페이지	`<a>` 태그를 사용합니다.
`.page-dot`	생략 구간 표시 (…)	내용이 없는 빈 `<span>`입니다. **양옆 번호가 연속되지 않을 때만** 출력하며, 번호 그룹의 **앞·뒤 양쪽**에 나타날 수 있습니다.
첫 페이지 `1`	첫 페이지 바로가기	블록 위치와 무관하게 **항상 노출**합니다.
마지막 `.page-link`	마지막 페이지 바로가기	값은 **전체 페이지 수**이며 **항상 노출**합니다. `title="마지막 페이지"` 속성을 부여합니다.

> **크기 규칙**
>
> `strong`, `.page-link`, `.page-dot`은 모두 동일한 크기로 통일되며, 최소 폭 방식이므로 세 자리 이상 번호(예: 271)가 들어와도 자동으로 확장됩니다.

1-3. 구조 설계의 이치

본 구조가 **이전 / 다음 + 첫 페이지 + 9개 블록 + 마지막 페이지**로 확정된 근거입니다.

1. **첫 페이지와 마지막 페이지를 항상 노출**

   양 끝 페이지가 상시 링크로 열려 있으므로 **맨처음 / 맨끝 버튼이 불필요**합니다. 좌우 컨트롤을 2개로 줄일 수 있어 모바일 가로 폭 부담이 크게 줄고, 사용자는 어느 블록에 있든 한 번의 클릭으로 목록의 처음과 끝으로 이동할 수 있습니다.

2. **이전 / 다음은 블록 단위로 이동**

   현재 블록의 9개 번호는 이미 화면에 노출되어 있으므로, 페이지 단위 이동은 화면상의 번호 링크와 중복됩니다. **이전 / 다음은 9페이지씩 건너뛰는 블록 이동을 담당**하여 번호 링크와 역할이 겹치지 않습니다.

3. **번호 블록을 9개로 고정**

   블록 크기를 고정하면 번호 영역의 폭이 페이지 이동과 무관하게 일정하게 유지되어, **페이지를 넘길 때마다 이전 / 다음 버튼의 위치가 흔들리지 않습니다.** 9개는 세 자리 번호와 양끝 페이지까지 포함해도 한 행에 안정적으로 들어가는 최대치이므로, **블록 크기는 9로 고정하며 임의로 변경하지 마십시오.**

4 **도트는 "번호가 건너뛰었다"는 신호**

양옆 번호가 연속이면 도트를 쓰지 않습니다. 예를 들어 블록이 `2`부터 시작하면 첫 페이지 `1`과 연속이므로 앞쪽 도트를 생략하고, 블록이 `270`으로 끝나고 마지막 페이지가 `271`이면 뒤쪽 도트를 생략합니다. 연속 구간에 점을 찍으면 건너뛴 페이지가 있다는 잘못된 신호를 주게 됩니다.

1-4. 상황별 출력 패턴

개발 시 판단해야 할 상황은 아래 **5가지**가 전부입니다. 각 상황의 출력 형태와 예시 화면을 순서대로 정리합니다.

상황	출력 형태	비고
① 총 9페이지 이하	번호만 출력	**이전 · 다음 · 도트 모두 생략**
② 첫 블록	이전 · `1` ~ `9` · 도트 · 마지막 · 다음	앞쪽 도트 없음 이전 비활성
③ 중간 블록	이전 · `1` · 도트 · 블록 번호 · 도트 · 마지막 · 다음	도트 양쪽 모두 출력
④ 블록 끝이 마지막 직전	이전 · `1` · 도트 · 블록 번호 · 마지막 · 다음	**뒤쪽 도트 생략** (연속)
⑤ 마지막 블록	이전 · `1` · 도트 · 블록 번호 · 다음	마지막 페이지가 블록에 포함 다음 비활성

판단 기준은 하나

다섯 가지 상황이 따로 있는 것이 아니라, **"양옆 번호가 연속인가"** 하나만 보면 됩니다. 첫 페이지 `1`과 마지막 페이지는 항상 출력하고, 블록 번호와 이어붙일 때 **번호가 끊기면 도트를, 이어지면 아무것도 넣지 않습니다.**

상황별 예시 화면

총 271페이지 · 블록 크기 9 기준입니다.

상황 ② 첫 블록 — 현재 1페이지 · 블록 1~9

블록이 **1**부터 시작하므로 첫 페이지가 블록에 포함 → **앞쪽 도트 없음** · `.dir-prev` 비활성

상황 ③ 중간 블록 — 현재 12페이지 · 블록 10~18

‹ 이전 | 1 | … | 10 | 11 | **12** | 13 | 14 | 15 | 16 | 17 | 18 | … | 271 | 다음 ›

1과 **10**, **18**과 **271**이 모두 불연속 → **도트 양쪽 출력**

상황 ④ 블록 끝이 마지막 직전 — 현재 265페이지 · 블록 262~270

‹ 이전 | 1 | … | 262 | 263 | 264 | **265** | 266 | 267 | 268 | 269 | 270 | 271 | 다음 ›

블록 끝 **270**과 마지막 페이지 **271**이 연속 → **뒤쪽 도트만 생략**

상황 ⑤ 마지막 블록 — 현재 271페이지 · 블록 271

‹ 이전 | 1 | … | **271** | 다음 ›

마지막 페이지가 블록에 포함되므로 **별도 출력하지 않음** · `.dir-next` 비활성

상황 ① 총 9페이지 이하 — 현재 3페이지 · 총 5페이지

1 | 2 | **3** | 4 | 5

블록이 1개뿐이며 전체 번호가 한 번에 노출 → **이전 · 다음 · 도트 모두 출력하지 않음**

확인 요청 · 이전 / 다음 비활성 처리

상황 ②의 `.dir-prev`, 상황 ⑤의 `.dir-next`는 이동할 대상이 없습니다. 그러나 현재 **이 두 요소의 비활성 상태 스타일이 정의되어 있지 않습니다.** 개발 착수 전 아래 두 가지 중 처리 방식 확정이 필요합니다.

ⓐ **렌더링 제외** — 해당 `<a>`를 출력하지 않습니다. 추가 작업이 없지만 번호 그룹의 좌우 위치가 미세하게 이동합니다.
ⓑ **비활성 노출** — `<span>`으로 출력하고 흐리게 처리합니다. 레이아웃이 고정되지만 **퍼블리싱 측 스타일 추가가 선행**되어야 합니다.

본 문서의 예시 화면은 ⓑ **기준**으로 표현했습니다. 상황 ①은 요청에 따라 두 컨트롤을 모두 생략(ⓐ)했습니다.

1-5. 필수 준수 사항

1. **번호 그룹 `<div>`를 생략하지 마십시오.**

   번호 간 간격과 화면 크기별 배치가 모두 이 요소에 종속되어 있습니다. 생략 시 좁은 화면에서 이전 / 다음 버튼과 번호가 한 줄에 뒤섞입니다.

2. **현재 페이지는 `<strong>`, 나머지는 `<a class="page-link">`**

   `<a class="page-link is-on">`과 같은 상태 클래스 방식은 **스타일이 적용되지 않습니다.** 현재 페이지를 링크가 아닌 `strong`으로 두는 것은 “현재 위치는 클릭 대상이 아니다”라는 접근성 관점에서도 올바른 처리입니다.

3. **첫 페이지가 현재 페이지인 경우에도 `<strong>`은 1개만**

   상황 ②처럼 첫 페이지가 블록에 포함될 때, **첫 페이지 링크를 블록과 별도로 중복 출력하지 마십시오.** 번호 그룹 안에 같은 번호가 두 번 나타나거나 `<strong>`이 2개가 되면 현재 위치 표시가 깨집니다.

4. **`sr-only`로 현재 페이지를 명시**

   시각적으로는 배경색으로 구분되지만 스크린리더에는 숫자만 전달되므로 아래 텍스트가 필요합니다.

   ```
   <strong><span class="sr-only">현재페이지</span>1</strong>
   ```

5. **마지막 페이지 링크에 `title` 속성 부여**

   번호만으로는 그것이 마지막 페이지라는 사실이 전달되지 않습니다.

   ```
   <a class="page-link" href="javascript:void(0)" title="마지막 페이지">271</a>
   ```

1-6. 접근성 보완 권장

현행 마크업에 아래 속성을 추가하면 접근성이 향상됩니다. **기존 스타일에는 영향이 없습니다.**

HTML · 권장

```
<nav class="board-page" aria-label="페이지 목록">
    <a href="..." class="dir-prev">이전</a>
    <div>
        <a href="..." class="page-link" aria-label="페이지 1">1</a>
        <span class="page-dot" aria-hidden="true"></span>
        <a href="..." class="page-link">10</a>
        ...
        <strong aria-current="true"><span class="sr-only">현재페이지</span>12</strong>
        ...
        <span class="page-dot" aria-hidden="true"></span>
        <a href="..." class="page-link" aria-label="마지막 페이지 271">271</a>
    </div>
    <a href="..." class="dir-next">다음</a>
</nav>
```

- `<div>` → `<nav aria-label="...">` : 랜드마크로 인식되어 건너뛰기 탐색이 가능해집니다.
- `aria-current="true"` : KRDS가 지정한 현재 화면 표기 방식입니다.
- `.page-dot`에 `aria-hidden="true"` : 내용 없는 요소가 읽히는 것을 방지합니다.
- 숫자 링크에 `aria-label="페이지 N"`, 마지막 화면에 `aria-label="마지막 페이지, N"`을 부여합니다.
- 이전 / 다음을 비활성 노출(ⓑ)로 처리하는 경우 `aria-disabled="true"`를 함께 부여합니다.

2 board-button · 하단 버튼 영역

게시판 하단 버튼을 **좌 / 중앙 / 우로 자동 배치**하는 레이아웃 컨테이너입니다. CSS `:has()` 선택자로 자식 구성을 감지해 정렬이 결정되므로, **어떤 자식을 넣느냐가 곧 배치 지정**이 됩니다.

2-1. 동작 원리

자식으로는 아래 두 종류만 사용합니다.

자식 유형	의미	예시
직계 버튼 `.board-button > .button-*`	가운데 정렬 대상	`<button class="button-fill01">`
`<div>` **그룹** `.board-button > div`	좌측 또는 우측 정렬 대상	`<div><a class="button-fill01">...</a></div>`

이 두 가지의 조합에 따라 **5가지 배치 케이스**가 자동 분기됩니다.

2-2. 배치 케이스 5종

CASE 1 `div` 1개만 → 우측 정렬

가장 많이 사용되는 형태입니다. (목록의 “등록”, 상세의 “목록 / 수정 / 삭제”)

```
<div class="board-button">
    <div>
        <a href="javascript:void(0)" class="button-frame02-fill"><i>삭제</i></a>
        <a href="javascript:void(0)" class="button-fill01"><i>수정</i></a>
        <a href="javascript:void(0)" class="button-frame02"><i>목록</i></a>
    </div>
</div>
```

점선 = `.board-button` 컨테이너 · **div 그룹이 우측에 밀착**

조건 — `:has(> div:first-child:last-child)`, 즉 `div`가 **유일한 자식**일 때만 성립합니다. 옆에 직계 버튼을 하나라도 두면 CASE 3으로 전환됩니다.

CASE 2 직계 버튼만 → 중앙 정렬

```
<div class="board-button">
    <a href="javascript:void(0)" class="button-frame02 ico-prev-before"><i>이전</i></a>
    <a href="javascript:void(0)" class="button-fill01 ico-next-after"><i>다음</i></a>
</div>
```

‹ 이전 다음 ›

직계 버튼은 컨테이너 정중앙에 배치

CASE 3 직계 버튼 + `div` 1개 → 중앙 + 우측

```
<div class="board-button">
    <button type="button" class="button-size01-fill02"><i>항목저장</i></button>
    <button type="button" class="button-size01"><i>닫기</i></button>

    <div>
        <button type="button" class="button-size01 ico-down01-after"><i>시계열 연결항목</i></button>
        <button type="button" class="button-size01 ico-down01-after"><i>연도별항목</i></button>
    </div>
</div>
```

::before 균형추 · flex:1 항목저장 닫기 시계열 연결항목 연도별항목

좌측 균형추와 우측 `div`가 **동일한 flex:1**을 가지므로 중앙 그룹이 정중앙에 고정

동작 원리 — `::before`로 보이지 않는 균형추를 생성해 `flex:1`을 부여하고, 우측 `div`에도 동일하게 `flex:1`을 부여합니다. 좌우가 같은 폭을 차지하므로 가운데 버튼이 **컨테이너 정중앙**에 위치하게 됩니다.

CASE 4 직계 버튼 + `div` 2개 → 좌 / 중앙 / 우

```
<div class="board-button">
    <button type="button" class="button-fill01"><i>저장</i></button>

    <div><button type="button" class="button-size01"><i>전체선택</i></button></div>
    <div><button type="button" class="button-size01"><i>엑셀</i></button></div>
</div>
```

화면 순서 **div1 → 버튼 → div2** · 마크업 순서 **버튼 → div1 → div2**

마크업 순서 ≠ 화면 순서

첫 번째 `div`에 `order:-1`이 적용됩니다. 마크업은 `버튼 → div1 → div2` 순이지만, 화면은 `div1 → 버튼 → div2`로 출력됩니다. DOM 순서를 기준으로 하는 키보드 탭 이동은 마크업 순서를 따르므로, 시각적 순서와 탭 순서가 어긋납니다. 접근성이 중요한 화면에서는 CASE 5 사용을 권장합니다.

CASE 5 `div` 2개만 → 양끝 분산

```
<div class="board-button">
    <div><button type="button" class="button-size01"><i>전체선택</i></button></div>
    <div><a href="javascript:void(0)" class="button-fill01"><i>등록</i></a></div>
</div>
```

`justify-content:space-between`으로 **양끝 분산**

2-3. 케이스 선택 요약

원하는 배치	마크업 구성	비고
우측	`div` 1개	가장 일반적. 등록 / 목록 버튼
중앙	직계 버튼만	이전 / 다음, 확인 버튼
중앙 + 우측	직계 버튼 + `div` 1개	주 액션 + 보조 액션
좌 + 중앙 + 우	직계 버튼 + `div` 2개	탭 순서 주의 (CASE 4)
좌 + 우	`div` 2개	—
좌측 단독	**미지원** — 2-4 ① 우회 방법 참고	

2-4. 주의사항

1 좌측 단독 정렬은 지원되지 않습니다.

`div` 1개는 무조건 우측입니다. 좌측에만 배치하려면 우측에 빈 `div`를 넣어 CASE 5로 만들어야 합니다.

```
<div class="board-button">
    <div><button type="button" class="button-size01"><i>전체삭제</i></button></div>
    <div></div><!-- 균형추 -->
</div>
```

2 `div` 감싸기 여부가 배치를 결정합니다.

버튼이 1개뿐이더라도 `div` 유무에 따라 결과가 달라집니다.

```
<!-- 중앙 정렬 -->
<div class="board-button">
    <button type="button" class="button-fill01"><i>확인</i></button>
</div>

<!-- 우측 정렬 -->
<div class="board-button">
    <div><a href="javascript:void(0)" class="button-fill01"><i>등록</i></a></div>
</div>
```

템플릿 엔진에서 권한별로 버튼을 조건부 렌더링할 때 특히 주의가 필요합니다. 조건에 따라 `div` 안의 버튼이 모두 사라지면 빈 `div`가 남아 의도치 않은 케이스로 분기됩니다. 버튼이 없으면 `div`까지 함께 제거하십시오.

3 중첩 `div`를 사용하지 마십시오.

`div` 안에 다시 `div`를 넣으면 `:only-of-type`, `:nth-of-type(2)` 판정이 어긋나 배치가 무너집니다. `div` 내부에는 버튼만 배치하십시오.

4 버튼 텍스트는 반드시 `<i>`로 감싸십시오.

`.board-button` 내부 버튼에는 별도의 줄 높이가 적용되므로, 텍스트를 `<i>`로 감싸지 않으면 버튼 높이가 의도보다 커집니다. 또한 아이콘 클래스(`ico-*`)가 `<i>`를 기준으로 동작하므로 `<i>` 래퍼는 프로젝트 전역의 필수 규칙입니다.

문의 — 본 문서의 내용 중 실제 구현과 상이한 부분이 발견되거나, **확인 요청 1건**(1-4 · 이전 / 다음 비활성 처리 방식)에 대한 결정 사항이 있는 경우 회신 부탁드립니다. 반영본을 제공하겠습니다.