21. 📘 다양한 위치에 요소 삽입하기 – insertBefore, insertAdjacentHTML

1. 단순한 삽입을 넘어서 – 다양한 구조에 요소를 삽입해야 하는 실전 상황

우리가 지금까지 배운 `appendChild()`와 `prepend()`는 새로운 DOM 요소를 특정 부모 요소의 **안쪽**, 즉 **자식 요소로 삽입**할 때 가장 기본적으로 사용되는 방법입니다. 그러나 실제 웹사이트를 만들다 보면 이런 단순한 삽입만으로는 해결되지 않는 상황들이 무척 많습니다.

→ 예: 리스트 중간에 항목 삽입, 요소 바깥쪽에 새 구조 삽입, 서버 HTML 문자열 즉시 렌더링

이런 복잡한 삽입 요구에 대응하기 위해 DOM은 매우 다양한 삽입 메서드를 제공합니다.

---

2. 공구 상자처럼 다양한 삽입 메서드가 존재하는 이유

삽입 대상이...

- 요소의 **안쪽인지 바깥쪽인지**
- **맨 앞인지 중간인지 뒤인지**
- 삽입할 대상이 **DOM 노드인지 HTML 문자열인지**

...에 따라 정확히 대응할 수 있도록, 다양한 방식이 준비되어 있습니다.

→ 마치 드라이버, 망치, 펜치처럼 역할이 다른 공구들이 필요하듯,

→ DOM 메서드도 정확한 용도별로 세밀하게 나뉘어 있는 것입니다.

이번 시간에는 그중에서도 `insertBefore`, `append`, `insertAdjacentHTML` 이 세 가지 메서드의 의미, 사용법, 실무 예시, 성능 및 보안 측면까지 모두 다룹니다.

---

3. `insertBefore()` – 기준 요소 앞에 새 노드 삽입

✅ 개념

`insertBefore`는 DOM 구조 내에서 **기존 요소의 바로 앞에 새로운 요소를 삽입**하고 싶을 때 사용합니다.

```
parentNode.insertBefore(newNode, referenceNode);
```

- `parentNode` → 삽입이 이루어질 부모 요소
- `newNode` → 새로 삽입할 요소
- `referenceNode` → 기준이 되는 기존 자식 노드

✅ 예제

```
<ul id="fruits">
  <li>🍎 Apple</li>
  <li>🍌 Banana</li>
</ul>
```

```
const fruits = document.getElementById("fruits");
const banana = fruits.children[1];

const newItem = document.createElement("li");
newItem.textContent = "🍊 Orange";

fruits.insertBefore(newItem, banana);
```

→ 실행 결과:

```
<ul id="fruits">
  <li>🍎 Apple</li>
  <li>🍊 Orange</li> <!-- 중간 삽입 -->
  <li>🍌 Banana</li>
</ul>
```

✅ 실전 사용처

- 상품 리스트 중간에 배너 삽입
- 댓글 중간에 고정 공지 넣기

- 요소가 특정 순서로 정렬된 경우 중간 위치 삽입

✅ 성능 고려사항 – Reflow 발생

DOM을 변경하면 브라우저는 레이아웃을 재계산하게 되는데, 이를 **리플로우(Reflow)**라고 합니다.

→ 위치/크기/배치 변경이 일어나므로 비용이 큽니다.

→ 반복적인 삽입 시 `DocumentFragment`와 함께 사용해 Reflow를 최소화해야 합니다.

---

4. `append()` – 텍스트와 노드를 함께 유연하게 추가

✅ 차이점

`appendChild()`는:

- 단 하나의 노드만 인자로 받음
- 반드시 DOM 노드여야 함

반면 `append()`는:

- 여러 개의 인자를 동시에 받을 수 있음
- 텍스트, DOM 노드 섞어서 사용 가능

✅ 예제

```
const box = document.createElement("div");
box.append("🌟", document.createElement("span"), "✨");
```

→ `box` 내부에: 텍스트("🌟") → `<span>` → 텍스트("✨") 순으로 삽입됩니다.

✅ 실전 사용처

- 버튼 옆에 아이콘 + 텍스트 추가
- 피드백 메시지를 구성할 때 문자열과 노드를 같이 붙이기
- 간단한 구성요소를 순식간에 붙이고 싶을 때

✅ 성능 측면

- 복수 노드 및 텍스트 동시 삽입 → 코드가 훨씬 간결해짐

- 메서드 체이닝 불가 (값을 반환하지 않음) → 가볍게 사용할 때 유용

---

5. `insertAdjacentHTML()` – HTML 문자열을 빠르게 삽입하는 최강 도구

✅ 개념

`insertAdjacentHTML`은 이름 그대로:

- `insert` → 삽입하다
- `adjacent` → 인접한 위치
- `HTML` → HTML 문자열

HTML 코드를 요소의 인접한 위치에 삽입합니다.

```
element.insertAdjacentHTML(position, htmlString);
```

- `position`: `"beforebegin"`, `"afterbegin"`, `"beforeend"`, `"afterend"` 중 하나
- `htmlString`: 삽입할 HTML 구조

✅ 예제

```
<div id="box">Hello</div>
```

```
const box = document.getElementById("box");
box.insertAdjacentHTML("beforebegin", "<p>1. 앞에 추가</p>");
box.insertAdjacentHTML("afterbegin", "<p>2. 안에 맨 앞</p>");
box.insertAdjacentHTML("beforeend", "<p>3. 안에 맨 뒤</p>");
box.insertAdjacentHTML("afterend", "<p>4. 뒤에 추가</p>");
```

→ 결과 구조:

```
<p>1. 앞에 추가</p>
<div id="box">
  <p>2. 안에 맨 앞</p>
  Hello
  <p>3. 안에 맨 뒤</p>
```

```
</div>
<p>4. 뒤에 추가</p>
```

✅ 실전 사용처

- 서버로부터 받아온 HTML 구조를 한 번에 렌더링
- 무한 스크롤 댓글, 실시간 피드 추가
- `createElement()` 없이 빠르게 DOM 구성

---

6. 보안 경고 – XSS(Cross-Site Scripting) 위험

`insertAdjacentHTML()`은 HTML 문자열을 직접 삽입하기 때문에 **XSS 공격에 취약**합니다.

→ 예: 사용자가 `<script>alert("해킹")</script>` 같은 코드를 넣으면 그대로 실행됨

✅ 해결법

- 사용자 입력은 반드시 **escape 처리**하거나
- DOMPurify 등 **전용 필터링 라이브러리** 사용
- 단순한 문자열 출력은 `textContent` 사용

---

7. 성능 비교

메서드	텍스트/노드	위치 제어	HTML 문자열	보안
appendChild	노드만	맨 뒤	❌	안전
append	텍스트+노드	맨 뒤	❌	안전
insertBefore	노드만	기준 앞	❌	안전
insertAdjacentHTML	문자열	정확한 위치	✅	위험 (주의 필요)

✅ 핵심 요약

- `insertBefore()` → 기준 노드 앞에 새 노드 삽입
- `append()` → 여러 노드/문자열을 한꺼번에 마지막에 삽입
- `insertAdjacentHTML()` → HTML 문자열을 위치 지정하여 삽입 (성능 우수, 보안 취약)

- 반복 삽입 시 `DocumentFragment`로 리플로우 최소화
- 사용자 입력이 HTML에 들어갈 경우 **XSS 필터링 필수**
