25. 📘 Node vs Element – childNodes와 children의 진짜 차이

1. 직관적인 개념 설명

지금까지 우리는 HTML 요소들을 자바스크립트로 선택하고, 그 요소들의 부모나 자식, 형제 노드를 탐색하면서 DOM을 구성하는 구조를 이해해왔습니다. 이 과정에서 DOM 트리의 구조와 탐색 방식에 대해 많은 기초를 쌓아왔죠.

그런데 DOM을 조금 더 깊이 다루다 보면 반드시 마주치게 되는 개념이 있습니다. 바로 **Node와 Element의 차이**, 그리고 `childNodes`, `children`, `firstChild`, `firstElementChild`처럼 비슷하게 생겼지만 **내부 동작과 반환 결과가 전혀 다른 속성들**입니다.

이 속성들은 초보자뿐 아니라 중급 개발자도 헷갈리기 쉬운데, 이 차이를 제대로 이해하지 못하면 DOM을 조작할 때 **예상치 못한 오류**나 **의도하지 않은 결과**를 만나게 됩니다. 이번 시간에는 이 차이를 정확하게, 그리고 근본적인 구조부터 상세하게 이해해보겠습니다.

---

2. DOM의 기본 단위 – Node란 무엇인가?

DOM(Document Object Model)은 HTML 문서를 자바스크립트가 조작할 수 있도록 만든 **트리 구조 기반의 객체 모델**입니다. 이 구조에서 가장 기본적인 단위는 `Node(노드)`입니다.

Node는 여러 종류가 있으며, 다음과 같은 하위 타입들을 포함합니다:

- **Element Node**: 실제 태그 요소 (`<div>`, `<ul>`, `<li>` 등)
- **Text Node**: 요소 사이의 줄바꿈, 공백, 텍스트
- **Comment Node**: `<!-- 주석 -->` 형태의 주석
- **Attribute Node**: 요소에 부여된 속성 (단, 최신 DOM에서는 별도로 분리해 다루지 않음)

이 중에서 우리가 가장 자주 사용하는 것은 `Element Node`입니다. 왜냐하면 화면에 직접 렌더링되는 구조이자, 시각적인 웹페이지의 구성 요소이기 때문입니다.

---

3. 코드 예제와 출력 결과 분석

다음과 같은 HTML 구조를 살펴봅시다:

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```

자바스크립트로 해당 `ul` 요소를 선택한 뒤 자식 노드를 확인해봅시다.

```
const ul = document.querySelector("ul");
console.log(ul.childNodes);
console.log(ul.children);
```

이때 출력되는 결과는 아래와 같이 다릅니다:

```
// 모든 노드 포함 (줄바꿈 포함)
NodeList(5) [#text, <li>, #text, <li>, #text]

// 요소 노드만 포함
HTMLCollection(2) [<li>, <li>]
```

여기서 중요한 차이점은 다음과 같습니다:

- `childNodes`는 줄바꿈과 공백까지 포함된 **모든 자식 노드**를 반환합니다.
- `children`은 오직 **Element Node만 필터링**하여 반환합니다.

---

4. 내부 구조 시각화 – 텍스트 노드 포함 구조

브라우저는 HTML을 파싱할 때, 줄바꿈이나 들여쓰기까지도 **텍스트 노드**로 취급합니다. 눈에는 보이지 않지만 실제로 존재하는 구조입니다.

```
<ul>              ← Element Node
  \n              ← Text Node (줄바꿈)
  <li>Item 1</li>     ← Element Node
  \n              ← Text Node (줄바꿈)
  <li>Item 2</li>     ← Element Node
```

```
  \n              ← Text Node (줄바꿈)
</ul>
```

→ `ul.childNodes.length === 5`

→ `ul.children.length === 2`

이처럼 `childNodes`는 전체 구조를 포함하고, `children`은 태그만 필터링해서 반환합니다.

---

5. 실무에서의 활용 예시

실제 DOM 조작을 하는 상황에서는 **불필요한 Text Node를 무시하고 태그 요소만 다루는** 경우가 많습니다.

예를 들어, 자식 요소들을 반복문으로 순회하며 class를 추가하려면 다음과 같이 `children`을 사용하는 것이 안정적입니다:

```
for (const li of ul.children) {
  li.classList.add("active");
}
```

하지만 만약 `childNodes`를 쓴다면 다음과 같은 문제에 부딪힐 수 있습니다:

```
for (const node of ul.childNodes) {
  node.classList.add("active"); // ❌ Text Node에서 오류 발생
}
```

→ `Text Node`는 `classList` 속성을 가지지 않기 때문에, 에러가 발생합니다.

따라서 실무에서는 자식 요소들을 안전하게 처리할 때는 항상 `children`을 사용하는 것이 좋습니다.

---

6. firstChild vs firstElementChild

`childNodes`와 `children`의 차이는 `firstChild`와 `firstElementChild`에서도 동일하게 적용됩니다.

```
console.log(ul.firstChild);        // #text (줄바꿈 텍스트 노드)
console.log(ul.firstElementChild);  // <li>Item 1</li>
```

→ `firstChild`는 줄바꿈이 먼저라면 그것을 반환하고

→ `firstElementChild`는 그다음 실제 태그를 반환합니다.

---

7. 노드 타입 확인하기 – nodeType

모든 노드는 `nodeType`이라는 속성을 가지고 있으며, 이 값은 노드의 종류를 나타내는 숫자입니다:

nodeType	의미
`1`	Element Node
`3`	Text Node
`8`	Comment Node

예를 들어 다음과 같이 노드를 순회하며 노드 타입을 확인할 수 있습니다.

```
for (const node of ul.childNodes) {
  console.log(node.nodeType); // 3, 1, 3, 1, 3
}
```

→ 이 코드는 Text와 Element가 교차로 등장하는 구조임을 보여줍니다.

---

8. 실무 팁 – Element Node만 필터링하기

```
for (const node of ul.childNodes) {
  if (node.nodeType === 1) {
    console.log("Element:", node);
  }
}
```

또는 더 가독성 있게 상수를 사용할 수도 있습니다.

```
if (node.nodeType === Node.ELEMENT_NODE) {
  // 안전하게 요소만 처리
}
```

---

9. nodeName과 nodeValue 속성

각 노드는 `nodeName`, `nodeValue` 같은 속성으로도 정보를 제공합니다.

```
console.log(ul.firstChild.nodeName);  // #text
console.log(ul.firstChild.nodeValue); // \n
```

- Element Node의 `nodeName` → 대문자 태그 이름 (`LI`, `DIV`)
- Text Node의 `nodeValue` → 실제 텍스트 값

---

10. 개념 정리 비유

> DOM 구조는 마치 출판된 책입니다.
>
> 책에는 본문(태그 요소)뿐만 아니라, 줄바꿈, 공백, 각주, 주석 같은 **보이지 않지만 존재하는 요소**도 함께 들어 있습니다.

- `childNodes`는 책의 **모든 페이지 구성 요소**를 보여주는 원본 원고
- `children`은 **본문만 정리된 교정본**이라고 볼 수 있습니다

✅ 핵심 요약 표

속성	포함 대상	반환 타입	특징 및 용도
`childNodes`	모든 노드 (Text 포함)	`NodeList`	전체 구조 파악, 디버깅에 유리
`children`	오직 Element 노드	`HTMLCollection`	실무에서 반복문 조작에 적합
`firstChild`	첫 번째 노드 (줄바꿈 가능)	`Node`	예상치 못한 Text Node 반환 가능
`firstElementChild`	첫 번째 HTML 요소	`Element`	안전한 요소 접근
`nodeType`	노드 종류 숫자	`Number`	1: Element, 3: Text, 8: Comment
`nodeName`, `nodeValue`	이름 / 값	`String` / `null`	노드 정보 조회