📘 08. DOM은 왜 트리인가? – 부모, 자식, 형제 노드의 구조 이해

1. 드디어 본격적인 DOM 트리 구조 탐색 시작!

이번 시간부터는 이제 본격적으로 우리가 눈으로 확인하고 조작할 수 있는 **DOM 트리 구조**에 대해 다뤄보겠습니다.

지금까지는 DOM이 무엇인지, 어떻게 만들어지는지, 왜 필요한지를 중심으로 학습해왔죠.

이제부터는 **실제 DOM 객체를 탐색하고 조작하는 실전 편**으로 넘어가는 단계입니다.

---

2. 왜 DOM은 '트리 구조'라고 부를까?

우선 DOM이 왜 '트리(Tree)'라고 불리는지부터 짚고 넘어가야 합니다.

트리 구조는 마치 **뿌리에서 시작해 여러 가지로 퍼지는 나무**를 연상시키죠.

→ 현실에서는 **가계도**, **회사 조직도**, **폴더 구조**가 대표적인 트리 구조 예시입니다.

→ 하나의 기준점(루트)이 있고, 그 아래로 **계층적인 자식들이 연결되는 구조**입니다.

→ HTML도 마찬가지로, `<html>` 태그를 루트로 삼고 그 아래로 `<head>`, `<body>` 같은 요소들이 자식처럼 이어집니다.

→ 다시 `<body>` 아래에는 `<div>`, `<h1>`, `<p>` 같은 요소들이 중첩되어 들어가죠.

→ 이 구조 전체를 브라우저는 **트리 형태의 객체 구조(DOM Tree)** 로 해석합니다.

---

3. DOM이란 문서를 '객체로 다루기 위한 구조'

정리하자면 DOM이란, "문서(Document)를 자바스크립트로 조작할 수 있도록 객체(Object)처럼 만든 트리 구조"입니다.

→ 그래서 이름도 **Document Object Model**.

→ 즉, HTML 문서를 객체처럼 다룰 수 있도록 만든 **모델 구조**입니다.

---

4. 실제 HTML을 트리 구조로 시각화해보기

다음과 같은 HTML을 살펴봅시다:

```html
<!DOCTYPE html>
<html>
  <body>
    <div>
      <h1>Hello</h1>
      <p>World</p>
    </div>
  </body>
</html>
```

이 HTML은 브라우저에서 다음과 같은 DOM 트리로 변환됩니다:

```
document
└── html
    └── body
        └── div
            ├── h1
            │   └── #text("Hello")
            └── p
                └── #text("World")
```

→ 이처럼 DOM 트리는 부모-자식-형제 관계가 명확히 구분되는 **계층적 구조**입니다.

---

5. DOM 탐색의 핵심: 부모, 자식, 형제 관계

이제 DOM 트리에서 반드시 이해해야 할 3가지 기본 관계를 소개합니다.

- **부모 노드 (Parent Node)**: 어떤 노드를 포함하는 상위 노드

  → `<div>`는 `<h1>`, `<p>`의 부모입니다.

- **자식 노드 (Child Node)**: 특정 노드의 내부에 포함된 하위 노드

  → `<h1>`, `<p>`는 `<div>`의 자식입니다.

- **형제 노드 (Sibling Node)**: 같은 부모를 공유하는 노드들

  → `<h1>`과 `<p>`는 서로 형제(sibling)입니다.

---

6. 자바스크립트로 DOM 탐색하기

자바스크립트에서는 DOM 탐색을 위한 다양한 속성을 제공합니다. 가장 기본적인 예제를 실행해보죠:

```
const div = document.querySelector("div");
console.dir(div);
```

→ `console.dir()`은 해당 요소의 내부 속성을 **트리 구조**로 확인할 수 있도록 도와줍니다.

---

7. 부모-자식 탐색 속성들

다음은 핵심 속성과 그 기능입니다:

```
div.parentNode          // <body>
div.childNodes          // NodeList(5) [text, h1, text, p, text]
div.children            // HTMLCollection(2) [h1, p]
div.firstChild          // #text (줄바꿈 포함)
div.firstElementChild   // <h1>
```

속성	설명
`.parentNode`	현재 요소의 부모 노드를 반환
`.childNodes`	모든 자식 노드(요소 + 텍스트 + 주석 등)를 포함한 NodeList
`.children`	**요소 노드만** 포함한 HTMLCollection
`.firstChild`	첫 번째 자식 노드를 반환 (줄바꿈 등 텍스트도 포함됨)
`.firstElementChild`	첫 번째 자식 요소 노드만 반환

→ `childNodes`에는 줄바꿈, 띄어쓰기 등도 포함되어 `#text`로 표시됩니다.

→ 실제 요소만 탐색하고 싶다면 `children` 또는 `firstElementChild`를 사용해야 합니다.

---

8. 형제 노드 탐색하기

이제 형제(sibling) 노드를 탐색하는 방법을 알아봅시다:

```
const h1 = document.querySelector("h1");
h1.nextSibling;          // #text
h1.nextElementSibling;   // <p>
```

속성	설명
`.nextSibling`	다음 노드 (줄바꿈, 주석 포함) 반환
`.nextElementSibling`	다음 요소 노드만 반환
`.previousSibling`	이전 노드 반환
`.previousElementSibling`	이전 요소 노드만 반환

→ 줄바꿈도 하나의 노드로 취급되기 때문에, 일반적으로는 `nextElementSibling`을 사용하는 것이 더 정확합니다.

→ 줄바꿈, 공백 등이 코드 동작을 방해할 수 있으므로, 의도치 않은 오류를 예방하려면 **Element 전용 속성**을 쓰는 습관을 들이는 것이 좋습니다.

✅ 핵심 요약

정리하자면, DOM은 **객체 기반의 트리 구조**이며, 각 노드는 **부모, 자식, 형제**로 연결되어 있습니다.

- `.parentNode` → 부모 노드
- `.childNodes` → 모든 자식 노드 포함
- `.children` → 요소 노드만 포함
- `.firstChild` / `.firstElementChild` → 첫 자식 노드
- `.nextSibling` / `.nextElementSibling` → 다음 형제 노드

→ DOM은 구조와 객체가 결합된 시스템이며, 자바스크립트를 통해 이 구조를 **직관적으로 탐색하고 조작**할 수 있습니다.

→ 이 트리 구조를 제대로 이해하는 것이 곧 **웹 페이지 조작의 시작점**입니다.
