



## 03. HTML에서 DOM으로 – 구조가 객체가 되기까지의 여정

### 1. 지난 시간까지 우리는 DOM이 무엇인지, 또 왜 필요한지를 배웠습니다.

HTML은 단지 정적인 정보를 구조화하는 도구에 불과하고, 자바스크립트는 HTML을 직접 조작하지 못하기 때문에 브라우저가 HTML을 객체로 바꿔 제공하는 중간 구조, 즉 DOM(Document Object Model)이 반드시 필요하다는 것을 이해했습니다.

### 2. HTML은 사람이 읽는 설명서, DOM은 브라우저가 조립한 구조물

이번 시간에는 우리가 작성한 HTML이 브라우저에 의해 어떻게 해석되고, DOM이라는 구조로 변환되는 과정을 단계별로 천천히 살펴보겠습니다. DOM의 시작점을 이해하는 데 매우 중요한 내용입니다.

우리가 컴퓨터에 HTML 코드를 적는 건 단순히 웹페이지의 설계를 만드는 것과 같습니다.

→ HTML은 그 자체로 웹의 골격을 정하는 설계도이지만, 그 설계만으로는 아무 기능도 수행되지 않습니다.

그런데 그 설계도는 사람이 보기에만 이해하기 쉬운 글자일 뿐, 컴퓨터는 이것 그냥 문자로밖에 보지 못합니다.

→ 브라우저는 이 텍스트를 문자로 받아들이되, 자동화된 문법 해석기를 통해 구조를 분석해야 합니다.

그래서 브라우저는 이 HTML을 실제 구조물로 바꾸기 위해 읽고 해석해서 하나의 객체 구조로 바꿉니다. 그 구조물이 바로 DOM입니다.

→ 파싱(parser) 과정을 통해 텍스트를 노드 객체로 전환하고, 이들을 계층적으로 연결해 DOM 트리를 구성합니다.

### 3. 비유로 이해하기: 설명서를 읽고 책상을 조립하는 것과 같다

이 과정을 일상에 비유하면 이렇습니다. 우리가 가구를 조립하는 설명서를 받았다고 생각해 보세요. 설명서에는 '다리 4개를 여기 붙이고, 나사를 여기 조여서 책상을 만든다' 같은 내용이 써 있습니다.

그런데 설명서만 봐서는 책상이 만들어지지 않죠. 누군가 이 설명서를 읽고 실제로 조립해야 책상이 생깁니다.

→ 여기서 **설명서 = HTML**, **조립된 책상 = DOM**, **설명서를 읽고 행동하는 사람 = 브라우저**입니다.

이 비유는 매우 정확합니다. 브라우저는 HTML이라는 인스트럭션을 해석해 **구조화된 DOM 객체**를 생성하고, 이후 자바스크립트는 이 DOM 객체를 조작하는 방식으로 동작합니다.

## 4. 브라우저는 HTML을 한 줄씩 읽으며 DOM을 만든다

이제 실제로 브라우저가 어떤 식으로 HTML을 DOM으로 바꾸는지 그 과정을 따라가 보겠습니다.

브라우저는 HTML 문서를 받아오면, 맨 위부터 아래로 한 줄씩 차례로 읽습니다.

→ 브라우저의 렌더링 엔진은 HTML을 위에서부터 순차적으로 해석하며, 읽는 즉시 구조를 생성합니다.

이걸 **스트리밍 파싱(streaming parser)** 라고 하며, 렌더링과 파싱이 동시에 진행됩니다.

그리고 각 태그를 만날 때마다 이를 **토큰(token)** 이라는 작은 단위로 나눕니다. `<body>`, `<h1>`, `</h1>` 같은 것들이 각각 하나의 토큰입니다.

→ HTML 파서는 이 토큰들을 인식해서 **노드 생성 명령**을 실행합니다. 시작 태그 → 노드 생성, 종료 태그 → 닫기 및 부모로 이동.

## 5. 열리는 태그는 새 노드 생성, 닫히는 태그는 부모로 복귀

브라우저는 열리는 태그를 보면 "아, 새 요소가 생겼구나"라고 판단해서 노드를 만들고, 닫히는 태그를 보면 "이제 이 요소는 끝났어"라고 판단해서 부모로 돌아갑니다.

→ 이 로직을 통해 브라우저는 자동으로 **중첩 구조**를 인식하고, 올바른 **트리 구조**를 형성합니다.

이 과정은 마치 괄호가 열리면 중첩되고, 닫히면 바깥으로 나오는 것과 동일한 논리입니다.

이 과정을 반복하면서 하나의 **트리 구조**, 즉 **부모와 자식 관계를 가진 나무 모양의 구조**를 만들어 냅니다.

→ 이 구조가 바로 자바스크립트가 접근하는 DOM 트리입니다. 각 노드는 객체이고, 트리 구조로 연결되어 탐색 가능합니다.

## 6. 예제: HTML → DOM 트리로 변환

예를 들어 다음과 같은 HTML을 작성했다고 해볼게요.

```
<body>
  <h1>제목입니다</h1>
  <p>문단입니다</p>
</body>
```

이 HTML을 브라우저는 이렇게 정리해서 구조화합니다.

```
Document
├── body
│   ├── h1
│   │   └── "제목입니다"
│   └── p
│       └── "문단입니다"
```

📌 `body` 요소가 최상위 노드로 생성되고, 그 아래에 `h1`, `p`가 자식 노드로 배치되며, 각각의 자식 노드로는 **텍스트 노드**가 생성됩니다.

## 7. 이 DOM 구조는 메모리 객체로 구현된다

눈으로 보면 그냥 들여쓰기 된 것처럼 보이지만, 실제로는 메모리 속에서 `body`라는 큰 덩어리가 있고, 그 안에 `h1`과 `p`가 자식으로 있고, 각각 안에 또 텍스트가 들어있는 형태로 연결되어 있습니다.

→ 단순한 들여쓰기가 아니라, **실제 객체의 계층적 연결**을 나타냅니다. 이 객체들은 JavaScript로 탐색 가능한 상태로 저장되어 있습니다.

그리고 이 구조 안에서 각각의 요소는 모두 **\*\*객체(물건처럼 쓸 수 있는 데이터)\*\***로 저장되어 있습니다.

→ 각 노드는 `HTMLElement`, `Text`, `Node` 등으로 구현된 객체이며, `.nodeType`, `.parentNode` 등의 속성을 가집니다.

## 8. 속성도 객체 안에 포함된다

만약 `<p class="text">` 같은 HTML 태그를 작성했다면, 브라우저는 `p` 라는 요소를 만들고, 그 속성인 `class="text"` 도 함께 기억합니다.

→ 속성은 DOM 노드 내부의 `attributes` 속성 또는 `element.className`, `element.getAttribute()` 등을 통해 접근 가능합니다.

이 속성들은 `element.className`, `element.id`, `element.getAttribute()` 같은 자바스크립트 코드로 나중에 접근할 수 있게 됩니다.

→ 자바스크립트를 통해 동적으로 CSS 클래스 변경, 데이터 속성 접근 등이 가능해지는 이유입니다.

## 9. 실습: 브라우저에서 DOM 구조를 직접 확인해보자

이걸 눈으로 확인해보고 싶다면 실제로 아래 HTML을 웹 브라우저에서 열어보고, 개발자 도구 (크롬에서 F12 키) > Console 탭을 열어보세요.

```
<body>
  <h1 class="title">DOM 변환</h1>
  <p id="desc">이것은 설명입니다.</p>
</body>
```


그리고 콘솔에서 아래 코드를 직접 입력해보세요.

```
console.dir(document.body);
```

→ 이 명령은 `document.body` DOM 객체의 모든 속성과 자식 요소들을 **트리 구조로 펼쳐서 출력**합니다.

그러면 `body` 요소가 실제로 어떤 구조를 가지고 있는지, 어떤 자식 요소들이 들어 있는지, 속성은 어떤 게 있는지 전부 펼쳐서 볼 수 있습니다.

특히 `children` 속성에는 `h1` 과 `p` 가 들어 있고, 각각의 텍스트도 포함되어 있음을 볼 수 있습니다.

 DOM 객체의 탐색:

- `document.body.children[0].tagName` → `"H1"`

- `document.body.children[0].textContent` → "DOM 변환"
- `document.body.children[1].id` → "desc"

`console.dir()` 는 자바스크립트 객체를 트리 구조로 펼쳐 보여주는 기능인데, 이걸 통해 우리가 실제로 HTML을 작성할 때 어떤 식으로 브라우저 안에서 DOM으로 바뀌는지를 눈으로 확인할 수 있습니다.

## 10. 정리: HTML은 문자지만, DOM은 객체 트리다

이처럼 HTML은 처음에는 그냥 글자일 뿐이지만, 브라우저가 읽고 해석하면 메모리 안에서 서로 연결된 **객체 트리**, 즉 **DOM 트리**로 바뀝니다.

이 구조는 자바스크립트를 통해 접근할 수 있기 때문에, 어떤 요소를 찾아서 글씨를 바꾸거나, 버튼을 눌렀을 때 새로운 내용을 추가하거나 하는 식으로 동작하게 되는 겁니다.

→ DOM을 조작하면 **화면이 바로 반영되는 이유**는, 브라우저가 DOM을 기반으로 렌더링을 수행하기 때문입니다.

### ✅ 핵심 요약

- HTML은 텍스트지만, 브라우저는 이를 구조화하여 DOM이라는 객체 트리를 만든다.
- DOM 트리는 부모-자식 관계로 연결되며, 각 요소는 자바스크립트로 조작 가능한 객체다.
- 속성(class, id 등)은 객체 내부 속성으로 저장되어 JS 코드로 접근 가능하다.
- `console.dir()` 을 사용하면 DOM의 내부 구조를 직접 확인할 수 있다.