📘 DOM은 무엇인가? – 02. 브라우저가 HTML을 해석하는 진짜 방식

1. 지난 시간 복습: HTML만으로는 왜 부족한가?

우리는 지난 강의에서 HTML이 단순한 정적 구조만을 표현하는 언어이기 때문에, 현대 웹의 동적 상호작용을 위해서는 자바스크립트와의 연결이 반드시 필요하다는 점을 배웠습니다.

그리고 자바스크립트가 HTML을 직접 조작할 수 없기 때문에, 브라우저가 HTML을 파싱해 구성한 **DOM(Document Object Model)** 이라는 구조가 반드시 존재해야 한다는 것도 확인했습니다.

즉, 자바스크립트는 브라우저가 HTML을 '객체화'해 제공하는 DOM 구조 없이는 **웹 페이지의 어떤 부분도 탐색하거나 제어할 수 없습니다.**

---

2. DOM은 단순한 '도구'가 아니다 – 브라우저와 자바스크립트의 공식 인터페이스

많은 초보 개발자들은 DOM을 단지 HTML 요소를 자바스크립트로 다루기 위한 수단쯤으로 생각하기 쉽습니다.

하지만 DOM은 그 이상입니다. DOM은 **브라우저가 HTML을 구조화해서 생성한 객체 트리이자**, 자바스크립트가 **브라우저 내부 구조에 접근할 수 있도록 해주는 유일한 공식 인터페이스**입니다.

즉, DOM은 단순한 '제어용 객체'가 아니라, **브라우저가 내부적으로 HTML을 해석하여 생성한 진짜 메모리 구조이며**, 자바스크립트는 이 구조를 경유하지 않고는 브라우저와 소통할 수 없습니다.

---

3. 브라우저는 HTML을 어떻게 DOM으로 바꾸는가?

HTML은 네트워크를 통해 **단순한 문자열**로 브라우저에 전달됩니다.

이때 브라우저는 **HTML 파서(parser)** 라는 내부 프로그램을 통해 이 문자열을 다음 단계로 구조화합니다:

- HTML 문자열 → **토큰(token)** 으로 분할

→ 예: `<h1>`, `</h1>`, `<body>` 같은 단위로 자른다

- 토큰 → **노드(node)** 로 변환

  → 각 토큰을 기반으로 객체를 생성한다 (예: `HTMLHeadingElement`)

- 노드 → **계층 구조의 트리**로 연결

  → 부모-자식 관계로 엮어 트리로 구성한다

이 과정을 통해 브라우저는 HTML을 DOM 트리로 만들어 메모리에 저장합니다.

이것이 바로 자바스크립트가 읽고 쓰는 **문서 객체 모델(Document Object Model)** 입니다.

---

4. 예제: 간단한 HTML이 DOM으로 바뀌는 과정

다음은 우리가 작성할 수 있는 매우 기본적인 HTML입니다:

```
<html>
  <head>
    <title>예제</title>
  </head>
  <body>
    <h1>안녕하세요</h1>
    <p>DOM이란 무엇인가?</p>
  </body>
</html>
```

이 HTML이 브라우저에 의해 DOM으로 해석되면, 브라우저 메모리 안에는 다음과 같은 **트리 구조**가 생성됩니다:

```
Document
└── html
    ├── head
    │   └── title
    │       └── "예제"
    └── body
        ├── h1
        │   └── "안녕하세요"
```

```
        └── p
            └── "DOM이란 무엇인가?"
```

📌 여기서 중요한 점은 이 구조가 단순한 들여쓰기 스타일이 아니라, 실제로 **브라우저 메모리 내 객체들이 부모-자식 관계로 연결되어 존재**한다는 사실입니다.

이 트리는 HTML의 구조를 그대로 반영하면서도, 각 요소가 **Node 객체로 구현된 살아 있는 구조**입니다.

---

5. DOM 구조를 콘솔에서 직접 확인해보자

DOM이 실제로 어떤 형태로 존재하는지를 확인하려면, 브라우저의 **개발자 도구(Console 탭)**를 활용하는 것이 가장 좋습니다.

대표적인 명령어는 다음과 같습니다:

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

- `console.log(document)`는 HTML 코드처럼 렌더링된 모습을 출력합니다.
- `console.dir(document)`는 **객체 트리 구조로서의 속성과 메서드**를 탐색할 수 있도록 펼쳐 보여줍니다.

📌 예: `console.dir(document.body)`를 입력하면, 브라우저는 다음과 유사한 구조를 보여줍니다:

```
HTMLBodyElement
 ├── accessKey: ""
 ├── attributes: NamedNodeMap {...}
 ├── childNodes: NodeList(3)
 ├── firstChild: Text
 ├── innerHTML: "..."
 ├── tagName: "BODY"
 └── ...
```

이 결과를 통해 우리는 다음을 알 수 있습니다:

- `<body>`는 단순 문자열이 아니라 `HTMLBodyElement`라는 객체이다
- 다양한 속성(`accessKey`, `innerHTML`)과 **자식 노드 리스트(childNodes)**를 포함한다

- 자바스크립트는 이러한 객체의 속성을 수정하거나 새로운 노드를 추가하여 페이지를 바꾼다

---

6. DOM 객체의 실제 형태

하나의 HTML 요소가 메모리 상에서 어떤 객체로 바뀌는지를 살펴보겠습니다.

예를 들어 다음과 같은 코드를 DOM 구조로 변환해보면:

```
<h1>안녕하세요</h1>
```

브라우저 메모리에서는 다음과 같은 형태의 객체로 표현됩니다:

```
{
  nodeName: "H1",
  nodeType: 1,
  childNodes: [
    {
      nodeType: 3,
      nodeValue: "안녕하세요"
    }
  ]
}
```

📌 해설:

- `nodeName: "H1"` → 이 노드는 `<h1>` 태그임을 의미
- `nodeType: 1` → 1은 **요소 노드(Element Node)**
- 내부에는 또 다른 노드 `nodeType: 3`, 즉 **텍스트 노드(Text Node)** 가 포함되어 있음
- 이 텍스트 노드의 `nodeValue`는 실제 화면에 보이는 `"안녕하세요"`

즉, `<h1>` 하나만 있어도 브라우저는 이를 **요소 노드 + 텍스트 노드** 두 개의 객체로 구성하여 DOM 트리를 만든다는 사실을 알 수 있습니다.

---

7. DOM의 기반은 'Node' – 프로토타입 체인 이해

DOM의 모든 구성 요소는 공통적으로 `Node`라는 **최상위 프로토타입 객체**에서 파생됩니다.

즉, DOM 구조는 클래스 기반이 아니라 **프로토타입 기반의 상속 체계**를 따릅니다.

구체적으로는 다음과 같은 타입들이 존재합니다:

- `Document` → 문서 전체를 나타냄 (`document` 객체)
- `Element` → HTML 요소 노드 (`div`, `p`, `h1` 등)
- `Text` → 요소 내부의 텍스트를 나타냄
- `Comment` → 주석 노드 (`<!-- 주석 -->`)

이들 모두는 `Node`를 기반으로 하며, 다음과 같은 공통 속성과 메서드를 가집니다:

- `nodeType` – 노드 종류 구분 (1: 요소, 3: 텍스트 등)
- `childNodes` – 자식 노드 목록
- `parentNode` – 부모 노드
- `appendChild()` – 자식 노드 추가 등

따라서 DOM은 단순히 HTML 구조를 표현하는 트리를 넘어서, **자바스크립트로 조작 가능한 객체 지향 시스템**이라고 할 수 있습니다.

---

8. 실습: 브라우저에서 직접 DOM 탐색해보기

아래 절차대로 직접 DOM 구조를 탐색해보면, 지금까지 배운 이론이 실제 브라우저 안에서 어떻게 구현되어 있는지를 체험할 수 있습니다.

1. 크롬 브라우저에서 아무 웹페이지나 열기
2. `F12` 또는 오른쪽 클릭 → **검사(Inspect)** 클릭
3. 개발자 도구의 **Console 탭**에서 다음 명령어 실행:

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

- 첫 번째 명령어는 `<body>` 객체의 전체 구조, 속성, 메서드를 펼쳐서 보여줍니다.
- 두 번째 명령어는 `<body>`의 자식 노드를 리스트(NodeList) 형태로 보여줍니다.

→ 여기에 포함된 노드에는 **실제 요소 노드** 외에도, 줄바꿈이나 공백 같은 **텍스트 노드**까지 포함되어 있다는 점을 눈여겨봐야 합니다.

---

9. 비유로 이해하기: DOM은 무대, 자바스크립트는 연출자

복잡한 개념일수록 직관적인 비유가 이해를 돕습니다.

> HTML: 공연 설계도 (스크립트, 무대 배치도)
> DOM: 설계도를 바탕으로 실제로 지어진 무대
> JS: 연출자 (조명, 소품, 배우 이동을 제어)

- 설계도만으로는 아무것도 동작하지 않습니다 – HTML만 있는 상태
- 설계를 바탕으로 무대를 지어야 연출이 가능합니다 – DOM
- 연출자는 무대가 있어야 조명을 켜고 배우를 움직일 수 있습니다 – JavaScript

즉, **DOM은 HTML을 실제로 동작 가능한 상태로 만든 구조**이며, 자바스크립트는 이 DOM을 조작함으로써 **웹을 동적으로 변화시키는 연출자** 역할을 합니다.

✅ 핵심 요약

요소	역할
HTML	정적인 구조와 내용 작성
DOM	HTML을 객체로 구조화한 트리
JavaScript	DOM 트리를 읽고 조작하여 화면을 바꿈

🎯 오늘의 목표 정리

- DOM은 HTML을 객체화한 트리 구조이다.
- 브라우저는 HTML 문자열을 파싱해 DOM 트리를 만든다.
- DOM은 자바스크립트가 브라우저를 제어할 수 있도록 만들어주는 **공식 인터페이스**이다.
- DOM 탐색 도구로는 `console.dir()`, `console.log()`, `.childNodes`, `.nodeType`, `.nodeName` 등이 있다.
- DOM은 단순 트리가 아니라 **프로토타입 기반의 객체 시스템**이다.