📘 05. DOM은 자바스크립트일까? 브라우저 안의 동작 원리

1. DOM은 자바스크립트인가요? – 가장 많이 헷갈리는 질문

이번 강의에서는 많은 분들이 DOM을 처음 배울 때 가장 많이 헷갈려 하는 아주 중요한 질문 하나를 해결하고 넘어가려고 합니다. 바로 이런 질문입니다.

> “DOM은 자바스크립트인가요?”

정답은 단호하게 말씀드리자면 **“아닙니다.”**

→ DOM은 자바스크립트의 일부가 아닌, 브라우저(Web Platform)가 제공하는 별도의 기능입니다.

이 문장을 정확히 이해하지 못하면, 자바스크립트와 브라우저 환경을 구분하지 못하고 혼동하게 됩니다. 그래서 실제 에러가 발생했을 때도 문제의 원인을 파악하기 어려워질 수 있습니다.

→ 자바스크립트 언어의 문법 오류인지, 브라우저 환경에서만 동작하는 DOM API 호출 오류인지 구분이 필요합니다.

---

2. 왜 헷갈릴까? – 자바스크립트로 DOM을 다루기 때문

DOM은 우리가 웹 브라우저에서 HTML 문서를 조작하거나, 버튼을 클릭했을 때 어떤 반응을 만들고 싶을 때 가장 자주 사용하는 구조입니다.

→ 사용자가 보는 웹페이지는 사실상 HTML을 기반으로 만들어진 DOM 구조입니다.

그런데 이 구조를 **자바스크립트로 조작**하기 때문에, 많은 분들이 DOM을 자바스크립트의 일부로 착각합니다.

특히 처음 프로그래밍을 배우는 분들은 이렇게 생각할 수 있습니다.

> “나는 HTML로 뼈대를 만들고, 자바스크립트로 DOM을 바꿔. 그럼 DOM은 자바스크립트에 포함된 거 아니야?”

→ 조작 도구로 사용한다고 해서, 그것이 그 도구의 일부는 아닙니다.

---

3. DOM은 웹 브라우저가 제공하는 구조다

하지만 실제로 DOM은 **자바스크립트 언어 자체에 포함된 것이 아니라**, **웹 브라우저(Web Platform)** 가 제공하는 기능입니다.

→ 자바스크립트는 단지 그 기능을 사용할 수 있도록 허용된 "사용자"일 뿐입니다.

즉, 자바스크립트는 DOM이라는 문서 객체 구조를 **직접 생성하거나 소유하지 않습니다.**

→ 브라우저가 HTML을 해석해 DOM 구조를 만들고, 자바스크립트는 그 위에서 조작만 할 수 있습니다.

---

4. 비유로 이해하기 – DOM은 건물, 자바스크립트는 공구

이 관계를 현실에서 비유하자면, DOM은 '건물'이고 자바스크립트는 '공구'입니다.

→ HTML이라는 설계도를 기반으로 브라우저가 건물을 짓고(DOM 생성), 자바스크립트는 그 건물 내부를 공구처럼 조작합니다.

- DOM = 이미 존재하는 구조화된 건물
- 자바스크립트 = 그 건물을 수리하거나 꾸미는 도구

→ 즉, 자바스크립트는 **이미 지어진 DOM이라는 건물을 조작하고 활용하는 수단**에 불과합니다.

---

5. 좀 더 정확한 기술적 정의

기술적으로 조금 더 설명하자면, 자바스크립트는 **프로그래밍 언어**이고, DOM은 **브라우저에서 제공하는 객체 구조이자 API 집합**입니다.

→ 자바스크립트 자체에는 DOM이라는 기능이 내장되어 있지 않습니다.

브라우저는 HTML 문서를 파싱한 뒤 `document`, `window` 같은 **특수한 전역 객체**들을 만들어 자바스크립트에게 제공합니다.

→ 이들은 자바스크립트가 스스로 만들 수 없으며, 브라우저가 생성하여 연결해주는 객체입니다.

이런 객체들을 보통 **Web API**, 또는 **Host Objects**라고 부릅니다.

---

6. 코드로 확인하는 DOM API의 정체

예를 들어 다음과 같은 자바스크립트 코드를 보겠습니다:

```
const h1 = document.querySelector("h1");
h1.textContent = "바뀐 제목입니다!";
```

이 예시에서 `document`, `querySelector`, `textContent`는 자바스크립트 문법으로 접근하고 조작할 수 있습니다.

→ 하지만 이 기능들의 실체는 **브라우저가 만들어 놓은 DOM 구조를 자바스크립트가 다룰 수 있도록 연결한 API입니다.**

즉, 자바스크립트는 DOM을 인식하거나 해석할 능력이 원래 있는 것이 아니라, **브라우저가 제공한 다리(API)**를 통해 DOM을 사용할 수 있을 뿐입니다.

---

7. Node.js에서는 DOM이 없다?

이 개념을 완전히 이해하기 위해, 브라우저가 아닌 자바스크립트 환경인 Node.js를 떠올려 봅시다.

Node.js는 **브라우저가 아닌 서버 환경**에서 자바스크립트를 실행할 수 있게 만든 런타임입니다.

→ 이 환경에는 HTML 문서도 없고, DOM도 없습니다.

따라서 다음과 같은 코드는 작동하지 않습니다:

```
document.querySelector("h1");
```

→ `document is not defined` 오류가 발생합니다.

이는 DOM이 **자바스크립트 언어의 일부가 아니라, 웹 브라우저라는 특정 환경에서만 존재하는 기능**이라는 사실을 명확하게 보여줍니다.

---

8. 구조 vs API – 함께 쓰이지만 완전히 다른 존재

결국 DOM은 **브라우저가 HTML 문서를 해석해 구성한 구조화된 문서 객체 트리**입니다.

그리고 자바스크립트는 그 구조를 조작할 수 있도록 **브라우저가 연결해준 API를 사용하는 언어**입니다.

→ 둘은 함께 사용되지만, 각각의 출처와 역할이 명확히 다릅니다.

이 개념을 혼동하면 실무에서도 다음과 같은 오류 판단을 하게 됩니다:

- "왜 document가 undefined야?" → 브라우저 환경이 아님

- "왜 querySelector가 작동 안 해?" → DOM API가 제공되지 않은 환경

✅ 핵심 요약: DOM은 자바스크립트가 아니다

> DOM은 브라우저가 만들어주는 웹 문서의 객체 모델이고,
>
> 자바스크립트는 그것을 조작하는 데 사용하는 **도구**입니다.

→ DOM은 자바스크립트의 일부가 아니라, `웹 환경(Web Platform)`이 제공하는 구조입니다.

→ 자바스크립트는 DOM을 생성하지 않으며, **조작만 가능**합니다.

이제 이 구분을 명확히 정리해두면, 앞으로 DOM을 배울 때 구조와 API를 구분하여 정확히 이해할 수 있습니다.
