23. 📘 HTML 속성과 DOM 속성의 결정적 차이 – getAttribute vs .value의 진짜 의미

1. 동적 웹페이지를 위한 마지막 관문

이번 시간부터는 지금까지 배운 내용을 바탕으로 실제 웹페이지를 조작하기 위한 핵심 기술들을 하나씩 익혀나가겠습니다. 이제 우리는 선택과 탐색뿐 아니라 요소 자체를 변경하고, 사용자의 인터랙션에 따라 페이지의 상태를 동적으로 제어할 수 있는 단계에 도달했습니다. 하지만 실제 프로젝트를 구현하기에 앞서 반드시 짚고 넘어가야 할 중요한 개념이 남아 있습니다. 바로 DOM 요소의 속성 조작, 클래스 동적 변경, 인라인 스타일 조작, 그리고 Node와 Element의 구조적 차이에 대한 이해입니다.

---

2. 개념 설명: HTML 속성(attribute) vs DOM 속성(property)

HTML 요소는 대부분 여러 개의 속성(attribute)을 가집니다. 예를 들어 `<img>` 태그는 `src`, `alt`, `<input>` 태그는 `value`, `placeholder`, `type` 등의 속성을 가지며, 이러한 속성은 HTML 문서 내에서 태그 안에 직접 정의됩니다.

JavaScript에서는 이 속성들에 두 가지 방식으로 접근할 수 있습니다.

- 하나는 `.getAttribute()`와 `.setAttribute()`를 사용하는 방식입니다.
- 다른 하나는 `.value`, `.checked`, `.href` 등 DOM 속성에 직접 접근하는 방식입니다.

→ 이 둘은 이름은 비슷해 보이지만, **작동 시점과 반영 방식, 영향 범위가 완전히 다릅니다.**

먼저 `getAttribute()`는 해당 요소의 HTML 속성값을 문자열 형태로 가져오는 메서드이며, `setAttribute()`는 그 값을 새로 설정하거나 새로운 속성을 추가할 때 사용합니다. 이 메서드들은 HTML 문서에 명시된 **정적인 속성(attribute)** 에 직접 접근하고 조작할 수 있게 해 주며, 요소의 초기 외형이나 구성 상태를 제어하는 데 주로 사용됩니다.

---

3. 용어의 뿌리: 왜 '속성(attribute)'과 '속성(property)'을 나눠 부를까?

하지만 여기서 중요한 개념 차이가 존재합니다. `attribute`는 HTML 문서에 명시된 초기값으로, DOM이 생성될 때 요소의 기본 상태를 정의합니다. 반면 `property`는 브라우저가 DOM을 구성

하면서 각 요소에 부여하는 동적인 상태값으로, 사용자 입력이나 스크립트 조작을 통해 실시간으로 변합니다.

→ 쉽게 말해 `attribute`는 "태어날 때의 값", `property`는 "현재 상태의 값"이라고 이해하시면 됩니다.

> 🔍 마치 출생신고서에 적힌 이름은 attribute, 지금 옷을 갈아입고 있는 상태는 property라고 생각해보세요.

---

4. 코드 예제: 눈으로 확인해보는 차이

이 차이를 명확히 이해하기 위해 아래와 같은 HTML과 JavaScript 예제를 살펴보겠습니다.

```
<input type="text" id="myInput" value="초기값" />
<button id="changeAttr">setAttribute로 변경</button>
<button id="changeProp">DOM 속성으로 변경</button>
<button id="logValues">현재 값 확인</button>
```

```
const input = document.getElementById("myInput");
const btnAttr = document.getElementById("changeAttr");
const btnProp = document.getElementById("changeProp");
const btnLog = document.getElementById("logValues");

btnAttr.addEventListener("click", () => {
  input.setAttribute("value", "속성으로 변경됨");
  console.log("getAttribute:", input.getAttribute("value"));
  console.log("input.value:", input.value);
});

btnProp.addEventListener("click", () => {
  input.value = "DOM 속성으로 변경됨";
  console.log("getAttribute:", input.getAttribute("value"));
  console.log("input.value:", input.value);
});
```

```
btnLog.addEventListener("click", () => {
  console.log("getAttribute:", input.getAttribute("value"));
  console.log("input.value:", input.value);
});
```

이 예제는 HTML 속성과 DOM 속성의 차이를 **직접 눈으로 확인할 수 있도록** 구성되어 있습니다.

---

5. 실습 결과: 값의 흐름을 시각적으로 비교하기

버튼 1: `setAttribute("value", "...")`

- HTML 속성만 바뀝니다.
- `.value` 값은 아직 변경되지 않음 (즉, 입력창엔 반영되지 않음).
- getAttribute → "속성으로 변경됨"
- input.value → 이전값 그대로

버튼 2: `input.value = "..."`

- 실제 입력창에 값이 즉시 바뀜.
- `getAttribute`는 여전히 예전 HTML 속성값 그대로

버튼 3: `"직접 입력"` 후 `"현재 값 확인"`

- input.value → 사용자가 입력한 텍스트 그대로 나옴
- getAttribute → 여전히 HTML 초기값 또는 setAttribute로 바뀐 값

→ **사용자 입력은 DOM 속성만을 바꾼다는 중요한 사실을 보여줍니다.**

---

6. 실무 팁: 언제 어떤 방식을 써야 하는가?

실무에서는 이러한 차이를 정확히 이해하고 **상황에 맞게 사용해야** 합니다.

- 사용자 입력값을 처리하거나 동적으로 상태를 변경해야 할 때는 반드시 `.value`, `.checked`, `.selectedIndex`와 같은 **DOM 속성**을 사용해야 합니다.
- 예를 들어 로그인 폼, 체크박스, 셀렉트 박스 등 사용자와의 실시간 상호작용이 필요한 경우, **DOM 속성을 통해 현재 상태를 읽고 제어**해야 정확한 동작이 가능합니다.

> ❌ 만약 이때 getAttribute()를 사용하면 입력된 값이 반영되지 않고, 초기 HTML에 설정된 값만 읽히게 됩니다.
>
> 🔥 이는 폼 전송 시 잘못된 데이터를 전송하거나, 화면에 보이는 상태와 내부 로직이 **불일치하는 문제**를 초래할 수 있습니다.

→ 마찬가지로 `setAttribute()`를 사용해 값을 변경하면 **외형만 바뀌고**, 실제 상태는 변하지 않아 **의도치 않은 버그**가 발생할 수 있습니다.

---

7. 요약 정리: 안정적인 DOM 조작을 위한 판단 기준

정리하자면,

- **사용자의 행동에 따라 실시간으로 변하는 값**을 다룰 때는 DOM 속성을 사용해야 합니다.
- 요소의 **초기값이나 외형을 제어**할 때만 HTML 속성 (`get/setAttribute()`)을 다루는 것이 안정적입니다.

→ DOM 속성은 단순한 문자열이 아니라, 브라우저가 실시간으로 상태를 관리하기 위해 제공하는 **동적 인터페이스**입니다. 그 동작 방식을 정확히 이해하는 것이 중요합니다.

✅ 핵심 요약

비교 항목	HTML 속성 (attribute)	DOM 속성 (property)
사용 메서드	`getAttribute()`, `setAttribute()`	`.value`, `.checked` 등
언제 사용?	초기값 설정, 외형 제어	현재 상태 제어, 사용자 입력 대응
값 변경 시 반영 위치	HTML 구조에만 반영됨	실제 DOM 상태와 화면에 반영됨
실무 활용도	템플릿 구조 조작용	사용자 인터랙션 제어에 필수
