17. 📘 getElementsByClassName() 완전 정복 – 클래스 선택, 실시간 반영, 배열 변환까지

1. 클래스(class)란 무엇인가? 그리고 왜 중요한가?

DOM 요소를 선택하는 방식은 다양하지만, 그 중에서도 가장 현실적으로 자주 쓰이는 기준이 바로 `클래스(class)`입니다.

클래스는 CSS에서 스타일을 부여할 때도, 자바스크립트에서 여러 요소를 그룹화할 때도 필수로 사용됩니다.

→ HTML에서 `class="menu-item"`과 같이 정의된 클래스는 말 그대로 **의미를 공유하는 요소들의 이름표**입니다.

→ 실제로 웹페이지의 수많은 버튼, 목록, 섹션들은 거의 항상 클래스를 통해 스타일과 기능을 공유합니다.

---

2. `getElementsByClassName()` – 클래스 이름으로 한꺼번에 선택하는 메서드

`getElementsByClassName()`은 특정 클래스명을 가진 HTML 요소들을 **전부 한꺼번에 가져오는 DOM 메서드**입니다.

```
<ul>
  <li class="menu-item">Home</li>
  <li class="menu-item">About</li>
  <li class="menu-item">Contact</li>
</ul>
```

위 구조에서 `menu-item` 클래스를 가진 모든 `<li>` 요소를 자바스크립트로 가져오고 싶다면 다음처럼 작성할 수 있습니다.

```
const items = document.getElementsByClassName("menu-item");
```

→ 반환 결과는 **HTMLCollection**이며, 겉보기에는 배열처럼 생겼지만 사실은 다른 특징을 지닙니다.

→ 무엇보다 중요한 특징은 **live 상태**, 즉 DOM 구조가 바뀌면 결과도 즉시 반영된다는 점입니다.

---

3. 실시간으로 DOM이 반영되는 HTMLCollection의 동작 방식

```
const items = document.getElementsByClassName("menu-item");

const newItem = document.createElement("li");
newItem.textContent = "Blog";
newItem.className = "menu-item";

document.querySelector("ul").appendChild(newItem);

console.log(items.length); // 자동으로 4개로 증가됨
```

→ 새로운 요소를 DOM에 추가하자마자, `items`의 길이도 자동으로 4로 증가합니다.

→ 이것은 `HTMLCollection`이 DOM과 실시간 연결되어 있다는 증거이며, 반복문 처리나 조작 시 주의를 요합니다.

즉, 자바스크립트의 일반 배열처럼 "고정된 데이터"가 아니라, **HTML 문서와 동기화된 참조형 리스트**라고 이해해야 합니다.

---

4. 특정 영역만 선택하고 싶을 때 – 부모 요소 기준으로 좁히기

`getElementsByClassName()`은 문서 전체가 아닌 **특정 DOM 요소 내부에서만** 검색하도록 제한할 수 있습니다.

```
const menu = document.querySelector("ul");
const items = menu.getElementsByClassName("menu-item");
```

→ 이렇게 하면 해당 `ul` 태그 내부에 있는 `menu-item`만 가져오게 되며, 다른 `ul`이나 외부의 같은 클래스는 무시됩니다.

💡 이처럼 **선택 범위를 좁히는 습관**은 DOM 조작의 **정확도**와 **성능**을 모두 향상시켜 줍니다.

---

5. 공백으로 구분된 복합 클래스도 전달 가능 – 단, 완벽히 일치해야 함

클래스 속성은 여러 개를 공백으로 나열할 수 있습니다. 예:

```
<li class="menu-item active">Home</li>
<li class="menu-item">About</li>
```

이때 두 클래스를 **모두 가진 요소만** 선택하려면 다음처럼 사용합니다.

```
const activeItem = document.getElementsByClassName("menu-item active");
console.log(activeItem.length); // 결과는 1
```

여기서 중요한 점은:

→ `"menu-item active"`는 **하나의 인자**이며, 모든 클래스명이 **완벽히 일치해야만** 선택된다는 것입니다.

→ CSS 선택자 `.menu-item.active`처럼 "둘 다 포함" 조건과는 다릅니다.

---

6. 배열이 아닌 HTMLCollection – forEach는 동작하지 않음

`HTMLCollection`은 배열처럼 인덱스로 접근은 가능하지만, **`forEach()`나 `map()` 같은 메서드는 없습니다.**

```
const items = document.getElementsByClassName("menu-item");

// ❌ 에러 발생
// items.forEach((el) => ... );
```

따라서 반복 작업을 하기 위해선 먼저 배열로 복사해주는 것이 가장 안전합니다.

```
const itemsArray = Array.from(items);
itemsArray.forEach((el) => console.log(el.textContent));
```

💡 `Array.from()`은 iterable 객체를 배열로 바꿔주는 매우 강력한 도구입니다.

---

7. 실무 팁과 유의사항 요약

- `getElementsByClassName()`은 **클래스를 기준으로 여러 요소를 한꺼번에** 가져오는 간편한 메서드입니다.
- 반환값은 배열이 아닌 `HTMLCollection`이며, DOM 변경사항을 **실시간으로 반영**합니다.
- **부모 요소를 기준으로 범위를 좁혀 사용하는 것**이 바람직합니다.
- 클래스명을 여러 개 주면 **모두 일치하는 요소만** 반환합니다.
- `forEach()`는 사용할 수 없으므로, 배열로 변환해서 처리해야 안전합니다.

✅ 핵심 요약

- `getElementsByClassName()`은 DOM에서 특정 클래스명을 가진 요소들을 **빠르게 선택**할 수 있는 강력한 도구입니다.
- 반환값은 live HTMLCollection이며, DOM 변경에 **자동 반응**한다는 점에서 일반 배열과 다릅니다.
- 클래스명이 여러 개인 경우, 모든 클래스명이 정확히 일치해야 선택됩니다.
- 반복문 처리를 위해선 `Array.from()`으로 정적 배열로 변환하는 것이 안전합니다.
