✔ [3장] 바닐라 JS로 크롬앱 만들기

1.document 객체

- console.log(document) → 작성한 HTML 전체 구조 확인 가능
- document는 브라우저에 존재하는 object
- HTML은 JS에서 **object(document)**로 제공됨 → JS가 접근/변경 가능

```
console.dir(document);
console.log(document.title);   // HTML <title> 내용
document.title = "HO";         // JS로 title 변경
document.body;                 // body 항목만 가져오기
```

2.HTML 요소 선택하기

- document.getElementById("title") → 특정 id 불러오기
- document.getElementsByClassName("class") → 여러 개 가져오기 (HTMLCollection 반환)
- document.getElementsByTagName("tag") → 여러 개 가져오기 (HTMLCollection 반환)
- document.querySelector("css-selector") → CSS selector 방식, **첫 번째 요소만 반환**
- document.querySelectorAll("css-selector") → 조건에 맞는 **모든 요소(NodeList)**

예시:

```
document.querySelector("h1");
document.querySelector("div.hello:first-child h1");
```

3. 요소 조작
innerText : 요소 내부 텍스트 변경
title.innerText = "got you";
style : 인라인 스타일 변경

```
title.style.color = "blue";
```

className : class 전체 교체 (⚠️ 기존 class 사라짐)

classList : class 추가/삭제/확인/토글 (기존 class 유지)

```
h1.classList.add("clicked");

h1.classList.remove("clicked");

h1.classList.contains("clicked"); // 포함 여부 확인

h1.classList.toggle("clicked");   // 있으면 제거, 없으면 추가
```

4. 이벤트(Event)와 이벤트 리스너

- **event** : 브라우저에서 발생하는 행위 (click, mouseenter 등)
- **event listener** : 특정 event를 감지하고 실행할 코드 등록

기본 예시

```
const title = document.querySelector("div.hello:first-child h1");

function handleTitleClick(){
  title.style.color = "blue";
}
title.addEventListener("click", handleTitleClick);
```

handleTitleClick()처럼 () 붙이지 않음 → JS가 직접 실행하지 않고, **이벤트 발생 시 브라우저가 대신 실행**

5. 다양한 이벤트 활용

```
function handleMouseEnter() {
  title.innerText = "Mouse is here!";
}
function handleMouseLeave() {
  title.innerText = "Mouse is gone!";
}

title.addEventListener("mouseenter", handleMouseEnter);
title.addEventListener("mouseleave", handleMouseLeave);
```

- element.onmouseenter = handleMouseEnter; 와 element.addEventListener("mouseenter", handleMouseEnter); 는 동일하지만, addEventListener를 선호하는 이유 → removeEventListener로 제거 가능하기 때문.

6. window 객체

- window는 브라우저 자체를 의미
- 다양한 이벤트 지원 (resize, copy, paste, online/offline 등)

예시:

```
function handleWindowResize(){
  document.body.style.backgroundColor = "tomato";
}
function handleWindowCopy(){
  alert("copier");
}

window.addEventListener("resize", handleWindowResize);
window.addEventListener("copy", handleWindowCopy);
```

7. 변수 사용 (getter & setter 개념)

- currentColor : getter (현재 값 복사) → const 사용 적절
- newColor : setter (값 대입 예정) → let 사용 적절

코드 흐름 예시

1. click 이벤트 발생 → 함수 실행
2. currentColor에 h1.style.color 복사
3. newColor 선언
4. 조건문으로 색상 값 결정 ("tomato" 또는 "blue")
5. h1.style.color = newColor; 대입

👉 "="는 **오른쪽 값을 왼쪽에 대입**한다는 의미

8. JS + CSS 협력

- **스타일은 CSS가 담당**, JS는 HTML 변경만 담당

CSS:

```
h1 {

  color: cornflowerblue;
}
.clicked {
  color: tomato;
}
```

JS:

```
const h1 = document.querySelector("div.hello:first-child h1");
function handleTitleClick() {
  if(h1.className === "clicked") {
    h1.className = "";
  } else {
    h1.className = "clicked";
  }
}
h1.addEventListener("click", handleTitleClick);
```

⚠️ className 사용 시 기존 class가 사라짐 → 유지하고 싶다면 classList 사용 권장.

9. className vs classList

- className : 모든 class를 교체 (다른 class 삭제 위험 있음)
- classList : class들의 목록을 조작 (안전하게 유지 가능)

예시

```
function handleTitleClick() {
  h1.classList.toggle("clicked"); // sexy-font 같은 기존 class 보존
}
```