해커뉴스 API 실습(news.html / news.js)을 통한 템플릿 엔진 이해

🎯 1. 학습목표
템플릿 엔진(Handlebars)의 개념과 필요성 이해
순수 JS + innerHTML 템플릿 생성 방식의 한계를 파악하고, Handlebars가 이를 어떻게 해결하는지 이해
Handlebars의 **기본 문법(템플릿 작성, 데이터 바인딩, 반복문 each, 조건문 if)**을 적용할 수 있다.
기존 뉴스 목록/상세보기 SPA 코드에 Handlebars를 적용해 깔끔하고 유지보수 쉬운 구조로 개선할 수 있다.
템플릿 책임 분리(화면 = HTML 템플릿, 데이터 가공 = JS) 구조를 이해하여
실무형 웹 개발 방식에 익숙해진다.

🧩 2. 왜 Handlebars가 필요한가? (필요성 & 역할)

현재 훈련생이 사용 중인 news.js 코드를 보겠습니다.
뉴스 목록을 화면에 그릴 때 아래처럼 문자열을 직접 연결해서 HTML을 만드는 방식을 사용합니다.

📌 현재 코드(문자열 replace 방식)

```
const newsList = [];
let template = `
  <div class="container mx-auto p-4">
    <h1>Hacker News</h1>
    <ul>
      {{__news_feed__}}
    </ul>
  </div>`

template = template.replace('{{__news_feed__}}', newsList.join(''));
container.innerHTML = template;
```

문제점

HTML 문자열 안에 변수를 넣을 때 replace()로 수작업 처리해야 한다.
템플릿 안에 JS 로직이 들어가면 가독성이 떨어지고 유지보수가 어렵다.
태그가 많아질수록 "어디가 HTML이고 어디가 JS인지" 헷갈린다.
같은 문자열이 여러 번 나오면 replace가 정확하지 않을 수 있다.
XSS 보안에 취약해질 가능성도 높다.

🔑 Handlebars 역할
Handlebars는 HTML 템플릿 안에 데이터만 깔끔하게 끼워 넣을 수 있게 도와주는 "템플릿 엔진"이다.
간단히 말하면:
"Handlebars는 HTML 틀(템플릿)과 실제 데이터(JSON)를 합쳐 완성된 HTML을 만들어 주는 도구다."
HTML은 HTML대로 예쁘게 작성하고,
데이터 바인딩은 {{변수명}} 만 써주면 된다.

👍 3. Handlebars의 장점(훈련생 눈높이 버전)
☑ (1) HTML과 JS의 역할 분리

HTML은 화면 구조 담당
JS는 데이터 담당
→ 훨씬 읽기 쉽고 유지보수 쉬움

☑ (2) 문자열 replace 같은 작업이 필요 없음

{{title}}, {{id}} 형식으로 변수만 넣으면 됨.

☑ (3) 반복문·조건문을 HTML 안에서 자연스럽게 사용

예:

{{#each news}}

- {{this.title}}

{{/each}}

☑ (4) 재사용 가능

같은 템플릿을 여러 데이터에 쉽게 적용 가능.

☑ (5) XSS 보안에 상대적으로 안전
데이터가 자동 escape 되어 위험한 script 삽입을 예방.

❗ 4. Handlebars의 단점
훈련생에게도 현실적인 부분을 알려주면 좋습니다.
❌ (1) 런타임에 템플릿을 컴파일해야 하므로 속도가 느릴 수 있음
React / Vue처럼 빌드 단계에서 최적화되는 프레임워크보다 느림.
❌ (2) 기능이 제한적
SPA 전체를 관리하는 React나 Vue처럼 상태관리/라우팅 기능이 없음.
❌ (3) 너무 복잡한 앱에는 적합하지 않을 수 있음
Handlebars는 "정적 템플릿"을 출력하는 데 특화됨.

🛠 5. Handlebars 기본 문법

아주 간단한 예로 설명합니다.

1. 기본 데이터 바인딩

```
<h1>{{title}}</h1>
```

2. each 반복문

```
<ul>
{{#each news}}
  <li>{{this.title}} ({{this.comments_count}})</li>
{{/each}}
</ul>
```

3. if 조건문

```
{{#if isNew}}
  <span>NEW!</span>
{{/if}}
```

🩹 6. 이번 실습 코드에 Handlebars 적용하기

여기서는 훈련생이 업로드한 news.js 구조를 기반으로
템플릿 문자열 대신 Handlebars를 사용하는 버전으로 변환하는 예시를 제공합니다.

실제로 코드에 적용해 보겠습니다.

1. 템플릿 파일 생성
   ▶ 1단계: news.html에 Handlebars CDN 추가

```
news.html
```

```
▶ 2단계: 템플릿 영역 추가<br>
```html
news.html
```

```
<div class="mt-3">
  <a href="#/page/{{prevPage}}">이전 페이지</a>
  <a href="#/page/{{nextPage}}">다음 페이지</a>
</div>
```

▶ 3단계: news.js 변경

```
  news.js
```

기존 replace() 방식 삭제하고 이렇게 변경:

```
function newsFeed() {
  const newsFeed = getData(NEWS_URL);
  const template = Handlebars.compile(
    document.getElementById('news-template').innerHTML
  );

  const totalPages = Math.ceil(newsFeed.length / 10);
  const start = (store.currentPage - 1) * 10;
```
```

```
    const end = store.currentPage * 10;

    const pageData = {
      news: newsFeed.slice(start, end),
      prevPage: store.currentPage > 1 ? store.currentPage - 1 : 1,
      nextPage: store.currentPage < totalPages ? store.currentPage + 1 : totalPages
    };

    container.innerHTML = template(pageData);
  }
```

🔄 7. Handlebars 도입 후 코드가 좋아지는 점 요약

Before (현재 code)	**After (Handlebars 적용)**
HTML + JS가 섞여 구조 파악 어려움	템플릿과 로직이 분리되어 한눈에 읽힘
replace()로 일일이 문자열 치환	{{변수}}로 직관적으로 데이터 바인딩
반복문을 JS에서 직접 push	템플릿에서 {{#each}}로 처리
innerHTML 보안 위험	Handlebars가 자동으로 escape
유지보수가 힘듦	화면 담당 / 로직 담당이 분리되어 확장 쉬움

🏁 8. 정리

“Handlebars는 React나 Vue를 배우기 전에
템플릿과 데이터 바인딩 개념을 익히기 위한 가장 좋은 출발점입니다.”

실제 프로젝트에서도 “서버 템플릿 렌더링(SSR)” 또는 “간단한 정적 페이지”를 만들 때 많이 사용됩니다.