



TODO 만들기 with **HTMX**

양다영

CONTENTS

1. HTMX 소개
2. HTMX 특징
3. HTMX 간단한 문법 소개
4. 데모
5. TODO 만들기
6. 실제 활용 사례

1. HTMX 소개

HTMX?

HTMX는 웹 개발에서 HTML을 기반 동적인 웹 애플리케이션을 쉽게 만들 수 있도록 돋는 JavaScript 라이브러리.

HTMX는 기존의 HTML, CSS, 서버 측 템플릿을 활용하면서 복잡한 클라이언트 사이드 JavaScript 프레임워크를 사용하지 않고도 동적인 기능을 구현할 수 있게 해줌.

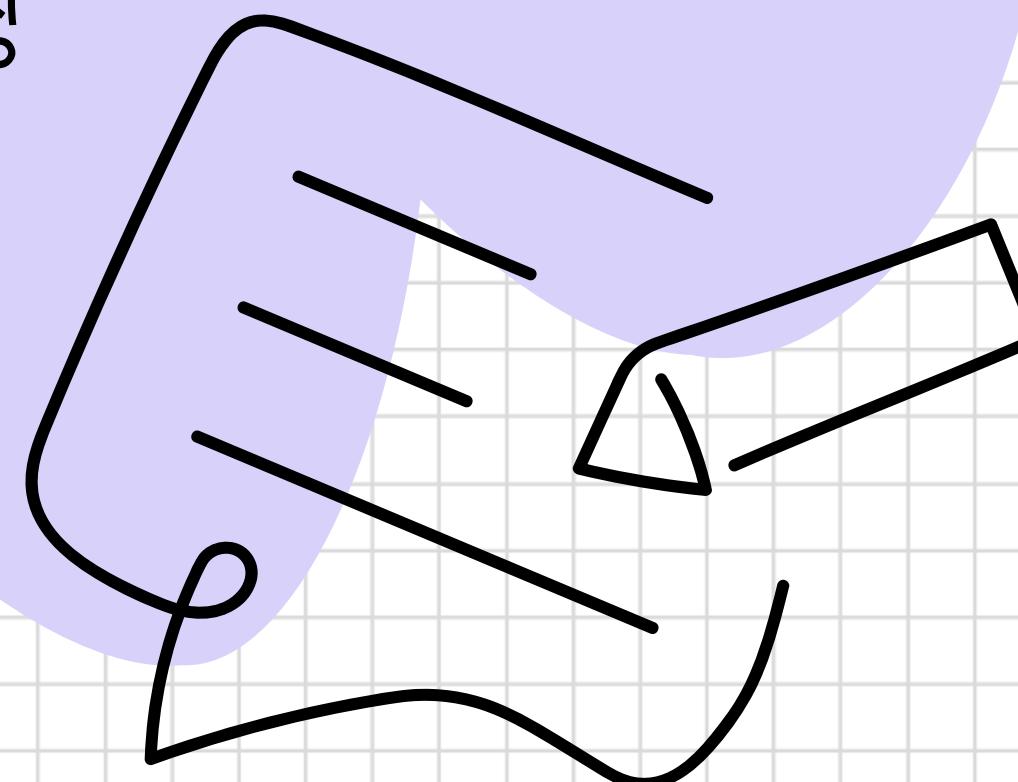
서버사이드 렌더링과는 달리 인터페이스 일부만 업데이트 가능. 페이지 깜빡임 없이 데 이터 표기 가능!

2. HTMX 특징



htmx의 CARSON 피셀

주요 특징



HTMX는 하이퍼미디어 지향 프론트엔드 라이브러리

HTML을 향상시키는 것에 중점을 두며 대체하지 않음

JS로 만들어 줬지만, JS를 많이 작성할 필요 없음

전체 크기는 약 14KB이며 의존성이 없음

HTMX는 반JavaScript가 아님!!
스크립팅언어로 사용 및 베이스에 집중!



2. HTMX 특징

htmx의 주요 특징 정리!

HTML 기반의 동적 웹 애플리케이션

서버 측 렌더링과의 통합

부분 업데이트와 페이지 변경

AJAX, CSS 전환, 이벤트 처리 지원

RESTful 인터페이스와 쉬운 통합

간결한 사용법

작은 의존성

3. HTMX 간단한 문법 소개

hx-{속성}="{값}"

기본 속성 : hx-{속성}="{값}"

```
<button hx-get="/api/data" hx-target="#result">  
    데이터 가져오기  
</button>
```

헤더와 파라미터

```
<!-- 헤더 추가 -->  
<div hx-get="/api/data"  
    hx-headers='{"Authorization": "Bearer token"}'>  
  
<!-- URL 파라미터 -->  
<div hx-get="/api/search?term=검색어">  
  
<!-- form 데이터 전송 -->  
<form hx-post="/api/submit"  
    hx-include="[name='extra']">
```

주요 속성

- GET 요청

```
<div hx-get="/api/users" hx-trigger="click">  
    사용자 목록 조회  
</div>
```

- POST 요청

```
<form hx-post="/api/submit">  
    <input name="email">  
    <button type="submit">전송</button>  
</form>
```

- hx-trigger: 이벤트 트리거 정의

```
<!-- 클릭 시 실행 -->  
<div hx-get="/api/data" hx-trigger="click">  
  
<!-- 3초마다 자동 실행 -->  
<div hx-get="/api/data" hx-trigger="every 3s">  
  
<!-- 여러 이벤트 -->  
<div hx-get="/api/data" hx-trigger="click, keyup[key=='Enter']">
```

3. HTMX 간단한 문법 소개

hx-{속성}="{값}"

- hx-target: 응답을 삽입할 대상 요소

```
<!-- id로 대상 지정 -->
<button hx-get="/api/data" hx-target="#result">

<!-- CSS 선택자 사용 -->
<button hx-get="/api/data" hx-target=".content">

<!-- this는 현재 요소 -->
<button hx-get="/api/data" hx-target="this">
```

- hx-swap: 응답 삽입 방식

```
<!-- 대상 내부를 교체 (기본값) -->
<div hx-swap="innerHTML">

<!-- 대상 앞에 추가 -->
<div hx-swap="beforebegin">

<!-- 대상 뒤에 추가 -->
<div hx-swap="afterend">

<!-- 대상의 처음에 추가 -->
<div hx-swap="prepend">

<!-- 대상의 마지막에 추가 -->
<div hx-swap="append">
```

이벤트 핸들링

```
<!-- 요청 전 확인 -->
<button hx-confirm="정말 삭제하시겠습니까?"
         hx-delete="/api/delete">
    삭제
</button>

<!-- 로딩 표시 -->
<button hx-get="/api/data"
         hx-indicator="#loading">
    데이터 로드
</button>
<div id="loading" class="htmx-indicator">
    로딩중...
</div>
```

3. HTMX 간단한 문법 소개

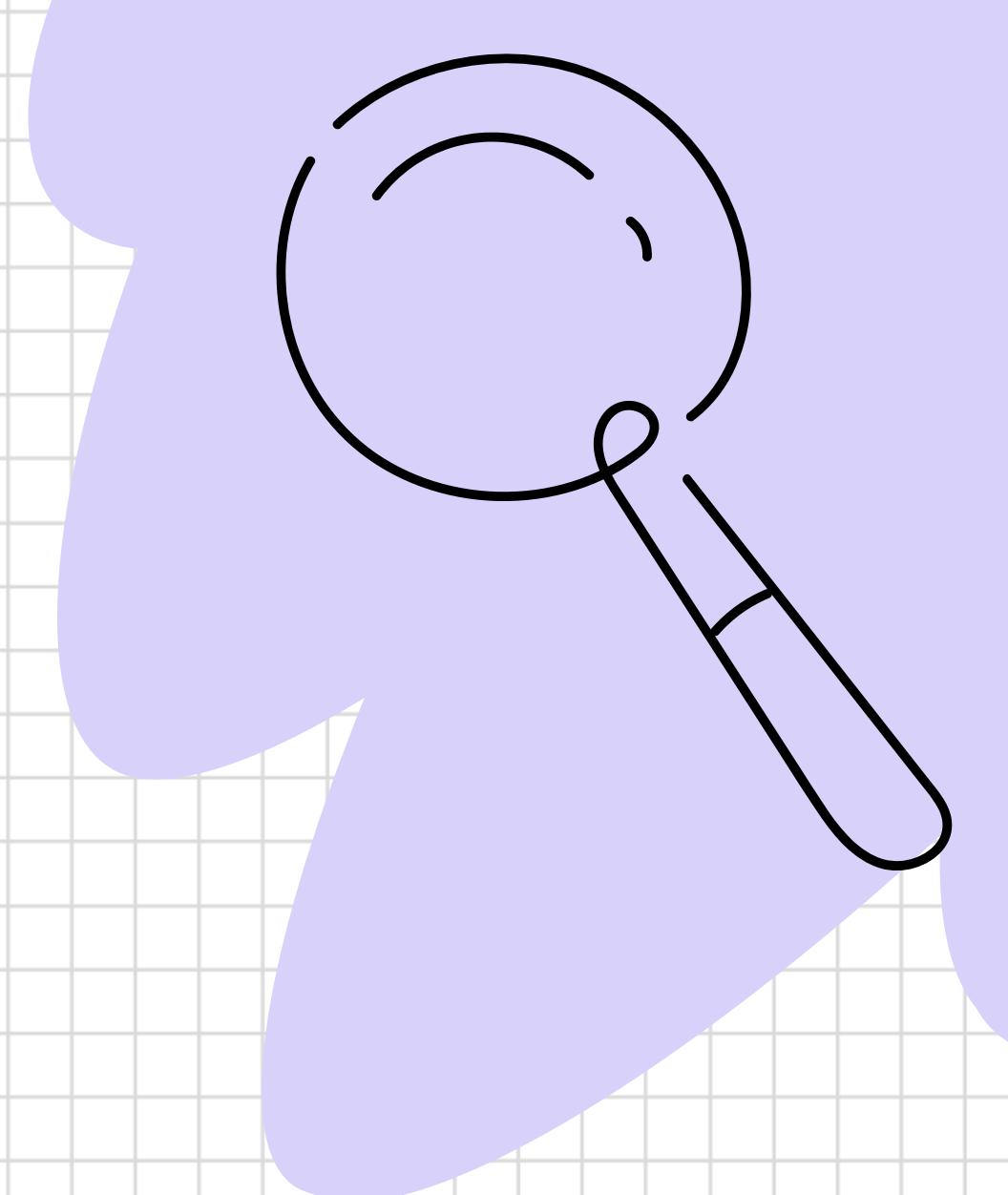
hx-{속성}="{값}"

부가 기능

```
<!-- 히스토리 관리 -->
<div hx-push-url="true">

<!-- 부스트 (전체 페이지 새로고침 방지) -->
<a hx-boost="true" href="/page">페이지 이동</a>

<!-- CSS 클래스 토글 -->
<div hx-get="/api/data"
    hx-classes-add="active"
    hx-classes-remove="inactive">
```



DEMO

todos

h

할일1

할일이 1개 남았습니다.

전체 진행중 완료 완료 삭제

Double-click to edit a todo
Written with HTMX

5. todo 만들기

TODO 만들기

```
<header class="header">
  <h1>todos</h1>
  <form hx-post="/todos"
    hx-target="#todo-list"
    hx-swap="afterbegin"
    hx-on::after-request="this.reset()">
    <input class="new-todo"
      name="todo"
      placeholder="할 일을 입력해 주세요."
      autofocus>
  </form>
</header>
```

hx-get, hx-post, hx-patch, hx-delete:

HTTP 메서드를 지정

hx-target: 응답을 어디에 넣을지 지정
(CSS 선택자 사용)

hx-swap: 응답을 어떻게 DOM에 삽입할지
지정 (예: outerHTML은 요소 전체를 교체)

5. todo 만들기

TODO 만들기

```
// Todo item template
const todoItemTemplate = (todo) => `
<li id="todo-${todo.id}" class="${todo.done ? 'completed' : ''}>
  <div class="view">
    <input
      class="toggle"
      type="checkbox"
      ${todo.done ? 'checked' : ''}
      hx-patch="/todos/${todo.id}"
      hx-target="#todo-${todo.id}"
      hx-swap="outerHTML">
    <label
      hx-get="/todos/edit/${todo.id}"
      hx-target="#todo-${todo.id}"
      hx-swap="outerHTML"
      hx-trigger="dblclick">
      ${todo.name}
    </label>
    <button
      class="destroy"
      hx-delete="/todos/${todo.id}"
      hx-target="#todo-${todo.id}"
      hx-swap="outerHTML">
    </button>
  </div>
</li>
`;
```

```
// Edit form template
const editItemTemplate = (todo) => `
<li id="todo-${todo.id}" class="editing">
  <form hx-post="/todos/update/${todo.id}"
        hx-target="#todo-${todo.id}"
        hx-swap="outerHTML">
    <input
      class="edit"
      type="text"
      name="name"
      value="${todo.name}"
      autocomplete="off"
      autofocus
      hx-trigger="blur from:input">
  </form>
</li>
`;
```

hx-trigger: 이벤트 트리거, 더블클릭으로 편집모드 진입,

blur from:input: blur 이벤트로 자동 제출, 특정 요소에서 발생한 이벤트만 감지

5. todo 만들기

TODO 만들기

```
<input  
  class="toggle"  
  type="checkbox"  
  ${todo.done ? 'checked' : ''}  
  hx-patch="/todos/${todo.id}"  
  hx-target="#todo-${todo.id}"  
  hx-swap="outerHTML">
```

```
app.post('/todos/clear-completed', (req, res) => {  
  todos = todos.filter(t => !t.done);  
  const remainingTodos = getFilteredTodos(req.query.filter || 'all');  
  res.header("HX-Trigger", "todosChanged");  
  res.send(remainingTodos.map(todo => todoItemTemplate(todo)).join(''));  
});
```

동적 UI 업데이트

- 서버에서 HTML을 직접 반환하고, HMTLX가 이를 자동으로 페이지에 반영
- 부분적 업데이트로 전체 페이지 새로고침 없이 UI갱신

5. todo 만들기

TODO 만들기

```
// Todo item template
const todoItemTemplate = (todo) => `
<li id="todo-${todo.id}" class="${todo.done ? 'completed' : ''}>
  <div class="view">
    <input
      class="toggle"
      type="checkbox"
      ${todo.done ? 'checked' : ''}
      hx-patch="/todos/${todo.id}"
      hx-target="#todo-${todo.id}"
      hx-swap="outerHTML">
    <label
      hx-get="/todos/edit/${todo.id}"
      hx-target="#todo-${todo.id}"
      hx-swap="outerHTML"
      hx-trigger="dblclick">
      ${todo.name}
    </label>
    <button
      class="destroy"
      hx-delete="/todos/${todo.id}"
      hx-target="#todo-${todo.id}"
      hx-swap="outerHTML">
    </button>
  </div>
</li>
`;
```

server.js

템플릿 시스템

- HTML 속성들로 동적 기능 구현 가능
- 동적 데이터와 HTML 속성 결합

서버에서
HTML 템플릿을
생성하여 클라
이언트에 전송

```
<h1>todos</h1>
<form hx-post="/todos"
       hx-target="#todo-list"
       hx-swap="afterbegin"
       hx-on::after-request="this.reset()">
  <input class="new-todo"
        name="todo"
        placeholder="할 일을 입력해 주세요."
        autofocus>
</form>
```

public/index.html

5. todo 만들기

TODO 만들기

```
<a hx-get="/todos?filter=all" hx-target="#todo-list">전체</a>
<a hx-get="/todos?filter=active" hx-target="#todo-list">진행중</a>
<a hx-get="/todos?filter=completed" hx-target="#todo-list">완료</a>
```

필터링 가능

- 쿠리 파라미터를 사용한 서버 사이드 필터링
- 필터링된 결과를 특정 타겟에 자동 업데이트

6. 실제 사용 사례

BERKELEY NOBEL PRIZE

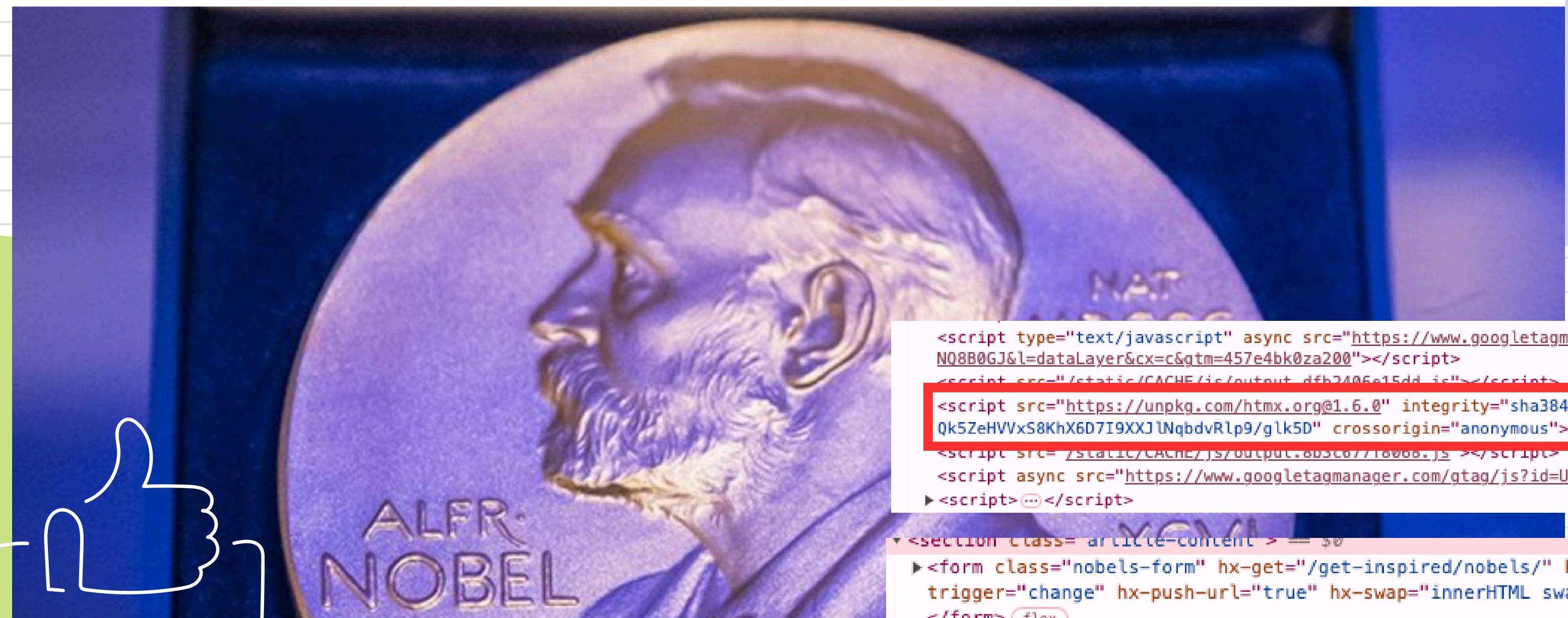
UC Berkeley Inspire

MAKE AN IMPACT ▾

GET INSPIRED ▾

GET TO KNOW US ▾

GIVE NOW



```
<script type="text/javascript" async src="https://www.googletagmanager.com/gtag/js?id=G-1JQNQ8B0GJ&l=dataLayer&cx=c&gtm=457e4bk0za200"></script>
<script src="/static/CACHE/1c/output_dfb2406e15dd.js"></script>
<script src="https://unpkg.com/htmx.org@1.6.0" integrity="sha384-G4dtlRlMBrk5fEiRXDsLjriPo8Qk5ZeHVVxS8KhX6D7I9XXJ1NqbdvRlp9/glk5D" crossorigin="anonymous"></script>
<script src="/static/CACHE/1c/output_e03ec07710000.js"></script>
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-31194529-3"></script>
▶ <script>...</script>

* <section class="article-content"> == $0
▶ <form class="nobels-form" hx-get="/get-inspired/nobels/" hx-target="#winners" hx-trigger="change" hx-push-url="true" hx-swap="innerHTML swap:.15s settle:.15s">
</form> flex
└ article id="nobel-prize" => $0
```

감사합니다...

- <https://htmx.org/>
- https://youtu.be/LriHRa9t1fQ?si=zSwW1S-ss_m9BGEo
- https://www.youtube.com/watch?v=vZ_oTop113I
- https://www.youtube.com/watch?v=te_IYPEDycc&t=46s
- <https://www.youtube.com/watch?v=r-GSGH2RxJs>
- <https://github.com/rajasegar/todomvc-htmx>