Django + htmx

Front end slickness with less learning

Why ?

- Users love ‘no-flash’ screen refreshes
- Bringing the user experience closer to a native app
- Django developers know lots of what necessary already
- Django developers don’t (necessarily) know much about frontend libraries

… the same page

What’s Django ?

- Full-Stack Framework
- Batteries Included
- Scalability and Security
- Rapid Development

What’s htmx ?

- A small js library
- Looks like a superset of HTML
- Custom attributes added to, eg <form>, support interaction with API
- Allows most logic to be defined on the server
- Maintains compatibility with ‘normal’ HTML

What’s the alternative ?

- A ‘traditional’ Javascript framework, such as React.js
- Very powerful, but requires a lot of non-Python knowledge
- Lots of dependencies means lots of mandatory maintenance
- Large payload passed to the browser

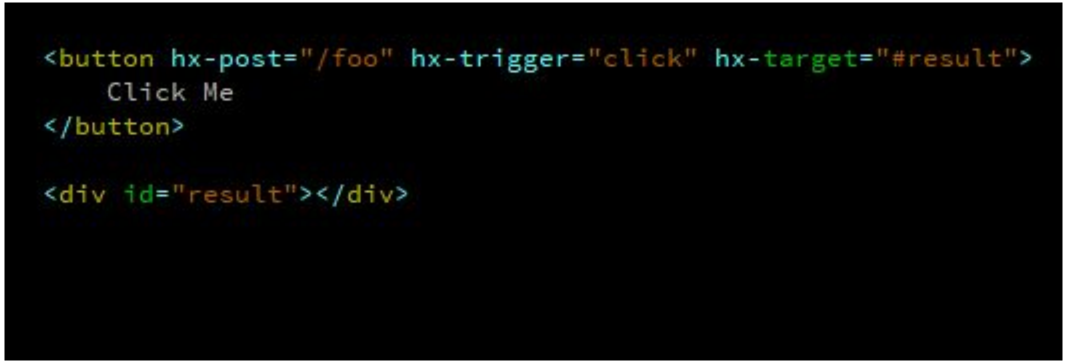
… let’s talk about htmx

htmx relies on html custom attributes

The ‘hx-’ attributes used to supplement the standard HTML behaviour

```
hx-get

hx-post

hx-put

hx-delete

hx-patch
```

Here’s a very simple sample



```
<button hx-post="/foo" hx-trigger="click" hx-target="#result">
    Click Me
</button>

<div id="result"></div>
```

htmx custom attributes and what they do

hx-get

hx-post

hx-put

hx-delete

hx-patch

triggered by DOM events

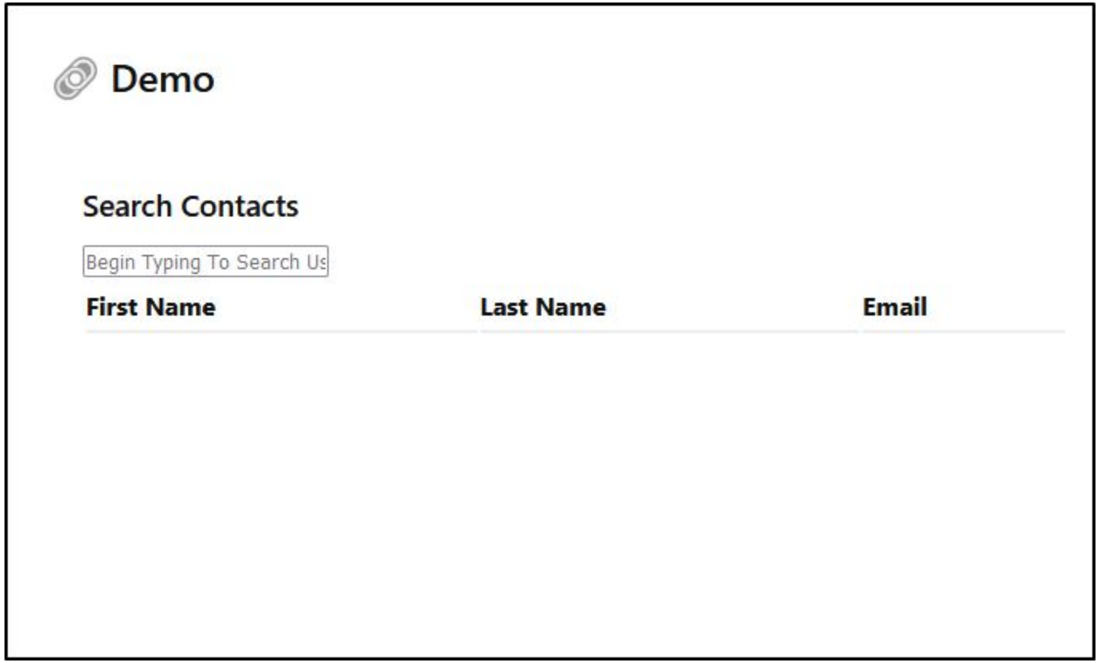
change

submit

click

… a htmx demo

What allows this to work ?

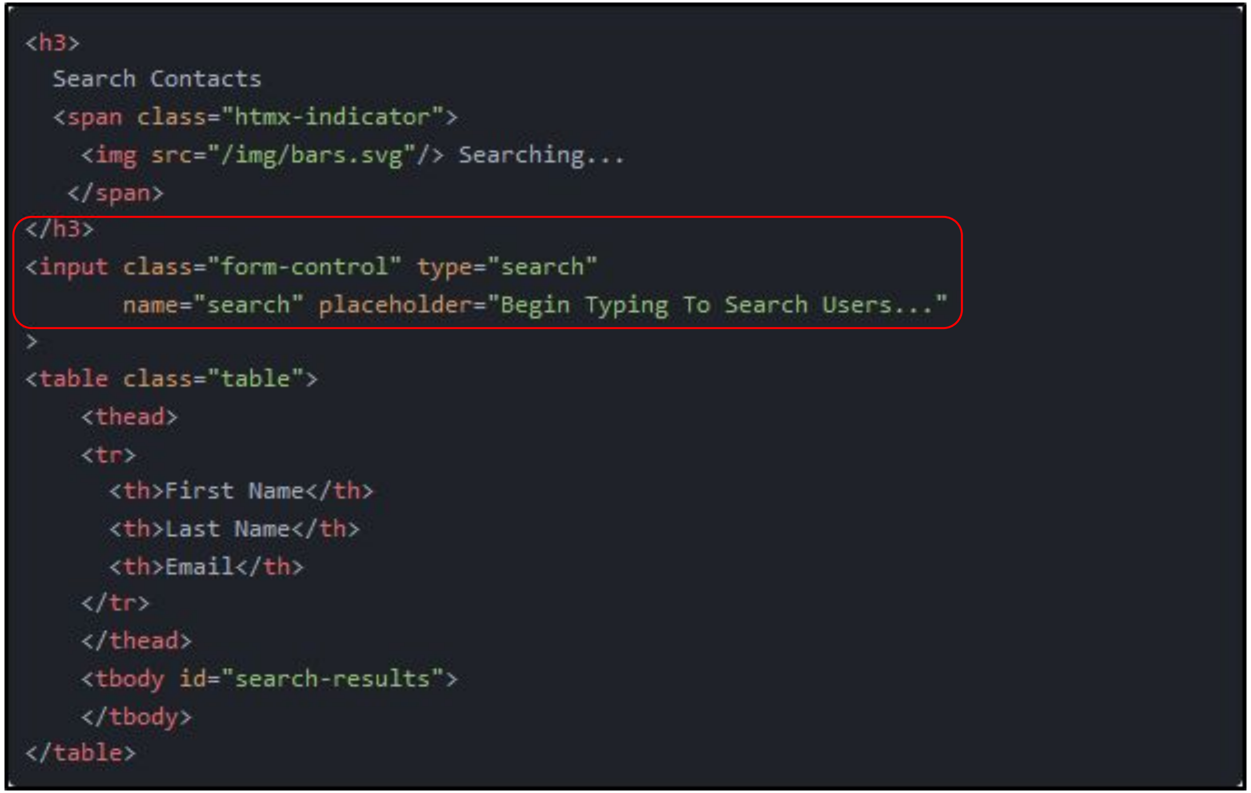

Demo

Search Contacts

Begin Typing To Search Us

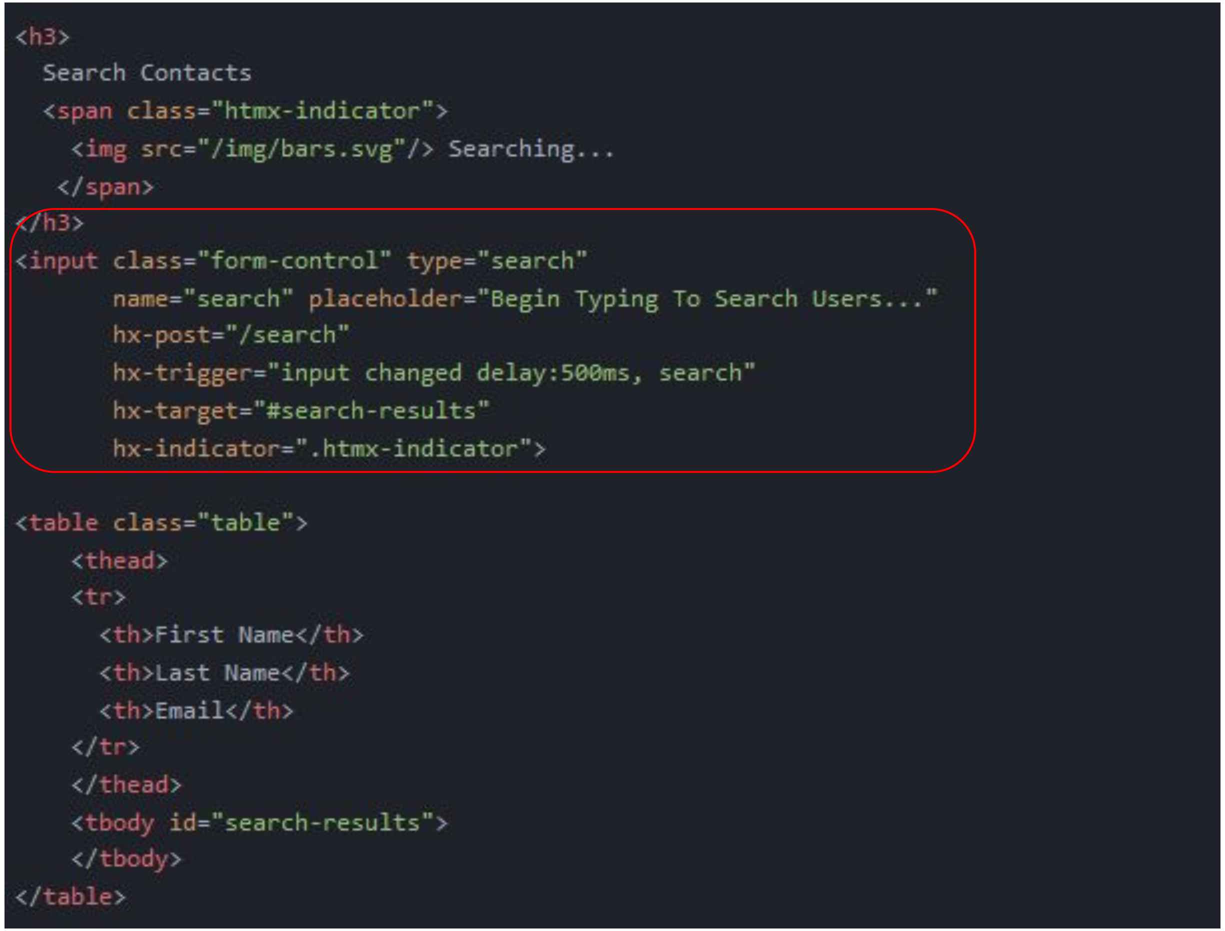
First Name	Last Name	Email

Markup of demo before htmx custom attributes added



```
<h3>
  Search Contacts
  <span class="htmx-indicator">
    <img src="/img/bars.svg"/> Searching...
   </span>
</h3>
<input class="form-control" type="search"
       name="search" placeholder="Begin Typing To Search Users..."
>
<table class="table">
    <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
    </thead>
    <tbody id="search-results">
    </tbody>
</table>
```

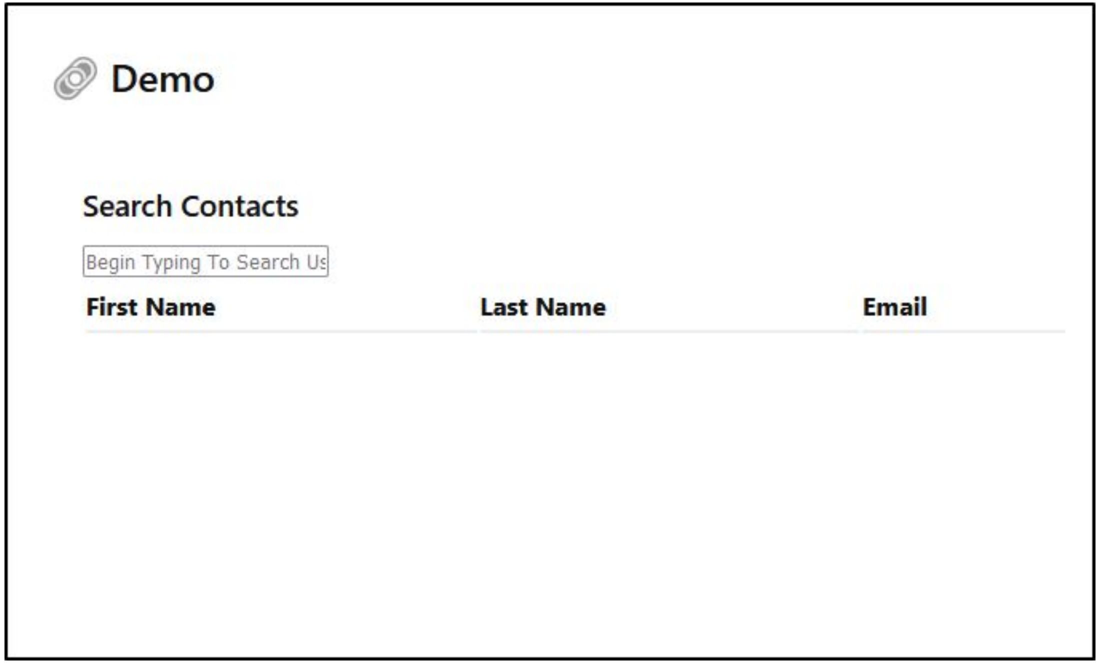
Markup of demo after htmx custom attributes added



```html
<h3>
  Search Contacts
  <span class="htmx-indicator">
    <img src="/img/bars.svg"/> Searching...
   </span>
</h3>
<input class="form-control" type="search"
       name="search" placeholder="Begin Typing To Search Users..."
       hx-post="/search"
       hx-trigger="input changed delay:500ms, search"
       hx-target="#search-results"
       hx-indicator=".htmx-indicator">

<table class="table">
    <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
    </thead>
    <tbody id="search-results">
    </tbody>
</table>
```

Demo

Here’s that markup in action. https://htmx.org/examples/active-search/#demo



Demo

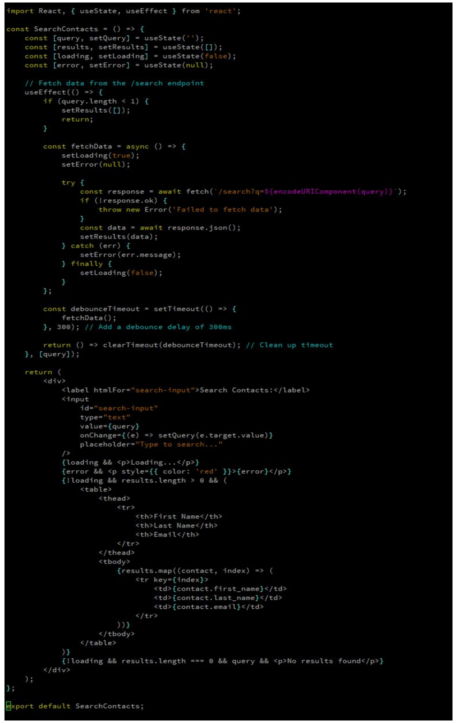
Search Contacts

Begin Typing To Search Us

First Name	Last Name	Email

The Alternative

- So what we just saw was four extra attributes on a pre-existing HTML element
- A corresponding React component is 70 lines long
- https://bit.ly/dj-hx-react



```
import React, { useState, useEffect } from 'react';

const SearchContacts = () => {
    const [query, setQuery] = useState('');
    const [results, setResults] = useState([]);
    const [loading, setLoading] = useState(false);
    const [error, setError] = useState(null);

    // Fetch data from the /search endpoint
    useEffect(() => {
        if (query.length < 1) {
            setResults([]);
            return;
        }

        const fetchData = async () => {
            setLoading(true);
            setError(null);

            try {
                const response = await fetch(`/search?q=${encodeURIComponent(query)}`);
                if (!response.ok) {
                    throw new Error('Failed to fetch data');
                }
                const data = await response.json();
                setResults(data);
            } catch (err) {
                setError(err.message);
            } finally {
                setLoading(false);
            }
        };

        const debounceTimeout = setTimeout(() => {
            fetchData();
        }, 300); // Add a debounce delay of 300ms

        return () => clearTimeout(debounceTimeout); // Clean up timeout
    }, [query]);

    return (
        <div>
            <label htmlFor="search-input">Search Contacts:</label>
            <input
                id="search-input"
                type="text"
                value={query}
                onChange={(e) => setQuery(e.target.value)}
                placeholder="Type to search..."
            />
            {loading && <p>Loading...</p>}
            {error && <p style={{ color: 'red' }}>{error}</p>}
            {!loading && results.length > 0 && (
                <table>
                    <thead>
                        <tr>
                            <th>First Name</th>
                            <th>Last Name</th>
                            <th>Email</th>
                        </tr>
                    </thead>
                    <tbody>
                        {results.map((contact, index) => (
                            <tr key={index}>
                                <td>{contact.first_name}</td>
                                <td>{contact.last_name}</td>
                                <td>{contact.email}</td>
                            </tr>
                        ))}
                    </tbody>
                </table>
            )}
            {!loading && results.length === 0 && query && <p>No results found</p>}
        </div>
    );
};

export default SearchContacts;
```

… let’s talk about django (and htmx)

django-htmx

- django-htmx is a python package developed by Adam Johnson to facilitate using htmx with Django
- Installed like any other Django extension package
    - ‘pip install -m pip django-htmx’
    - Add to INSTALLED_APPS
    - Add to MIDDLEWARE
- Place the htmx project’s htmx.min.js into STATICFILES_DIR
- Add a script reference to the base template

django-htmx

- The `django_htmx.middleware.HtmxMiddleware` attaches a `htmx` property to the incoming `request` object
- `request.htmx` contains the htmx-specific request headers and so makes them easily available in the view servicing the request.
- Examples of properties available to the view. Some key ones are:
    - react.htmx.target
    - react.htmx.trigger
    - react.htmx.trigger_name
    - react.htmx.triggering_event

django-htmx - an example in use

This example shows you how you can do partial rendering for htmx requests using **django-template-partials**. The view renders only the content of the table section partial for requests made with htmx, saving time and bandwidth. Paginate through the below list of randomly generated people to see this in action, and study the view and template.

See more in the docs.

id	name
1	Casey Black
2	Kimberly Myers
3	Ms. Stephanie Meza
4	Ms. Vickie Moore
5	Zachary Barry
6	Tina Pham
7	Jason Davis
8	Susan Wiley MD
9	Erin Freeman
10	Jonathon Smith

1 **2** **» Last**

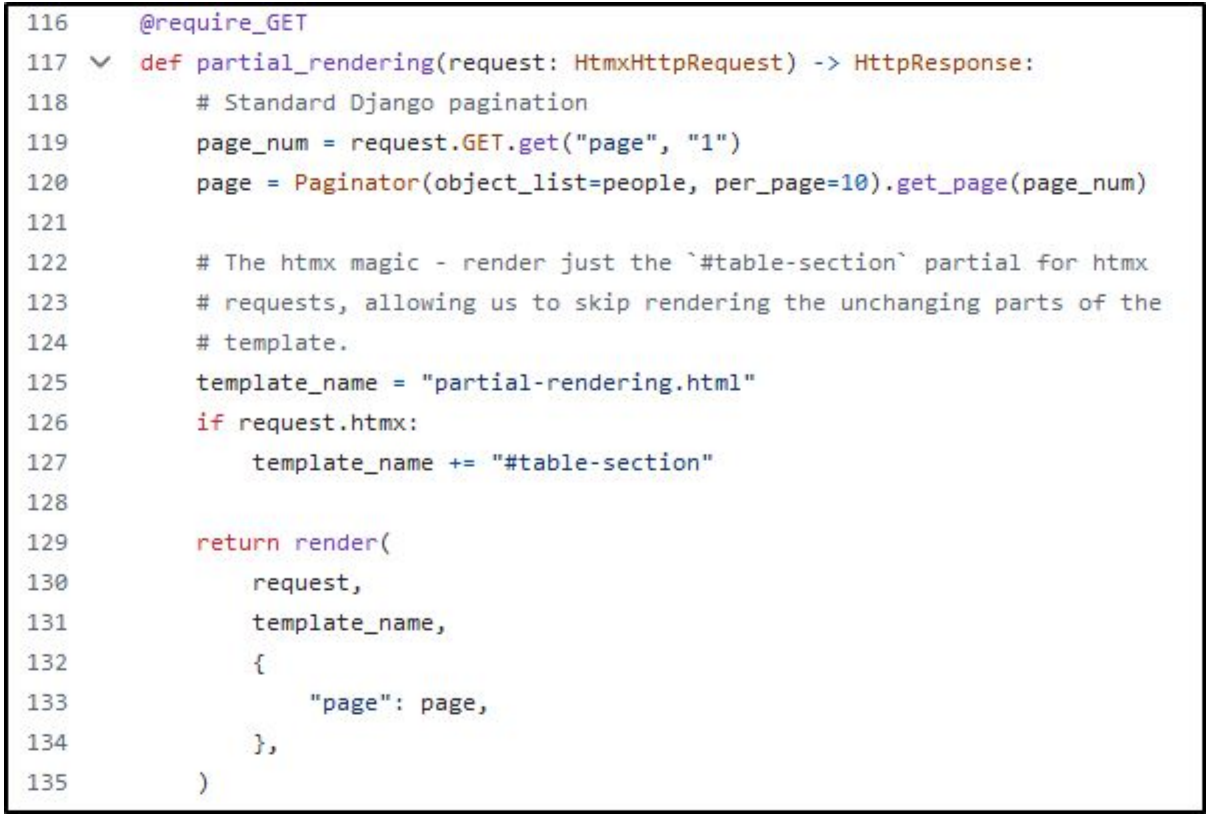
This example shows you how you can do partial rendering for htmx requests using **django-template-partials**. The view renders only the content of the table section partial for requests made with htmx, saving time and bandwidth. Paginate through the below list of randomly generated people to see this in action, and study the view and template.

See more in the docs.

id	name
11	Joseph Allen
12	Henry Powell
13	Micheal Hill
14	Taylor Andrews
15	Gabriel Duarte
16	Jennifer Strong
17	Sherri Haley
18	Alex Fuller
19	Adrian Briggs
20	Justin Snyder

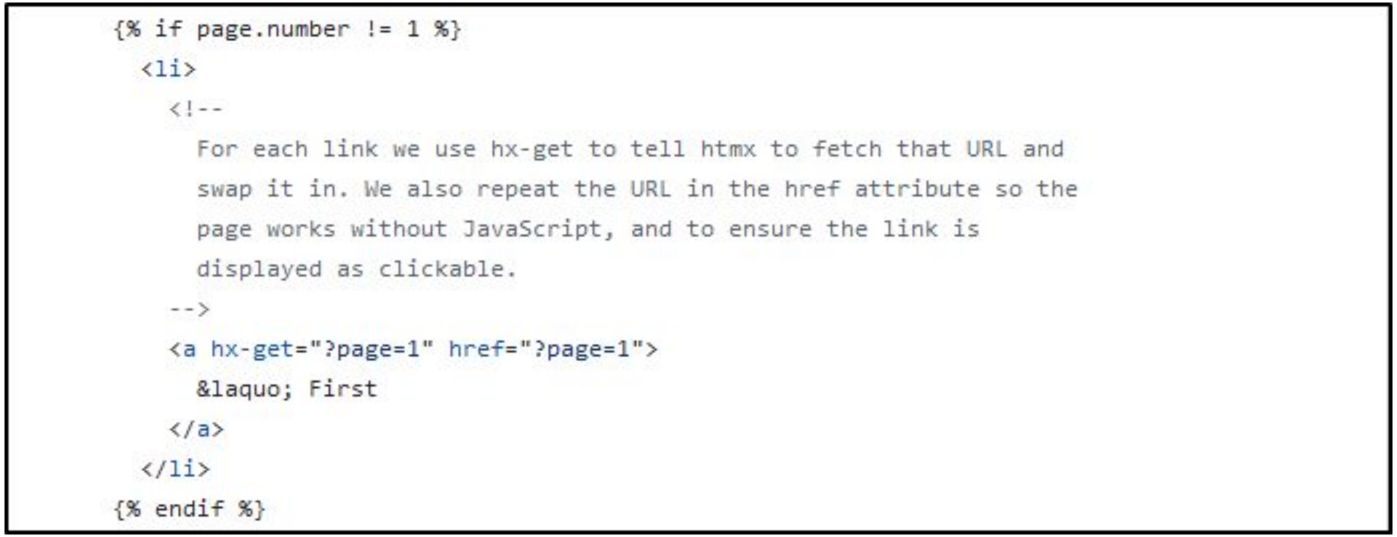
« First **1** **2** **3** **» Last**

The view function which populates the table

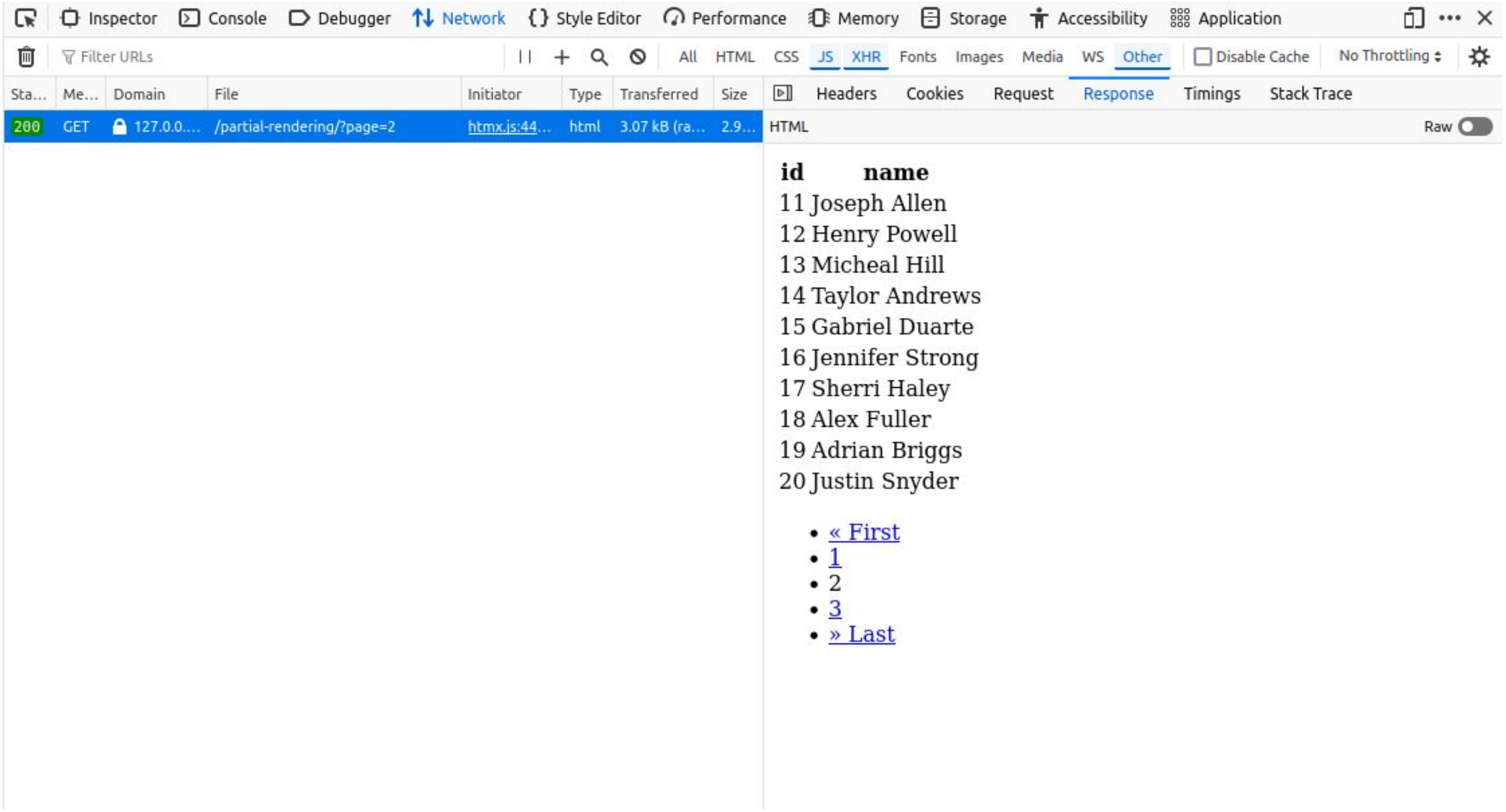


```python
@require_GET
def partial_rendering(request: HtmxHttpRequest) -> HttpResponse:
    # Standard Django pagination
    page_num = request.GET.get("page", "1")
    page = Paginator(object_list=people, per_page=10).get_page(page_num)

    # The htmx magic - render just the `#table-section` partial for htmx
    # requests, allowing us to skip rendering the unchanging parts of the
    # template.
    template_name = "partial-rendering.html"
    if request.htmx:
        template_name += "#table-section"

    return render(
        request,
        template_name,
        {
            "page": page,
        },
    )
```

The anchor link from the template which links back to the view seen on the previous screen.



```
{% if page.number != 1 %}
  <li>
    <!--
      For each link we use hx-get to tell htmx to fetch that URL and
      swap it in. We also repeat the URL in the href attribute so the
      page works without JavaScript, and to ensure the link is
      displayed as clickable.
    -->
    <a hx-get="?page=1" href="?page=1">
      &laquo; First
    </a>
  </li>
{% endif %}
```

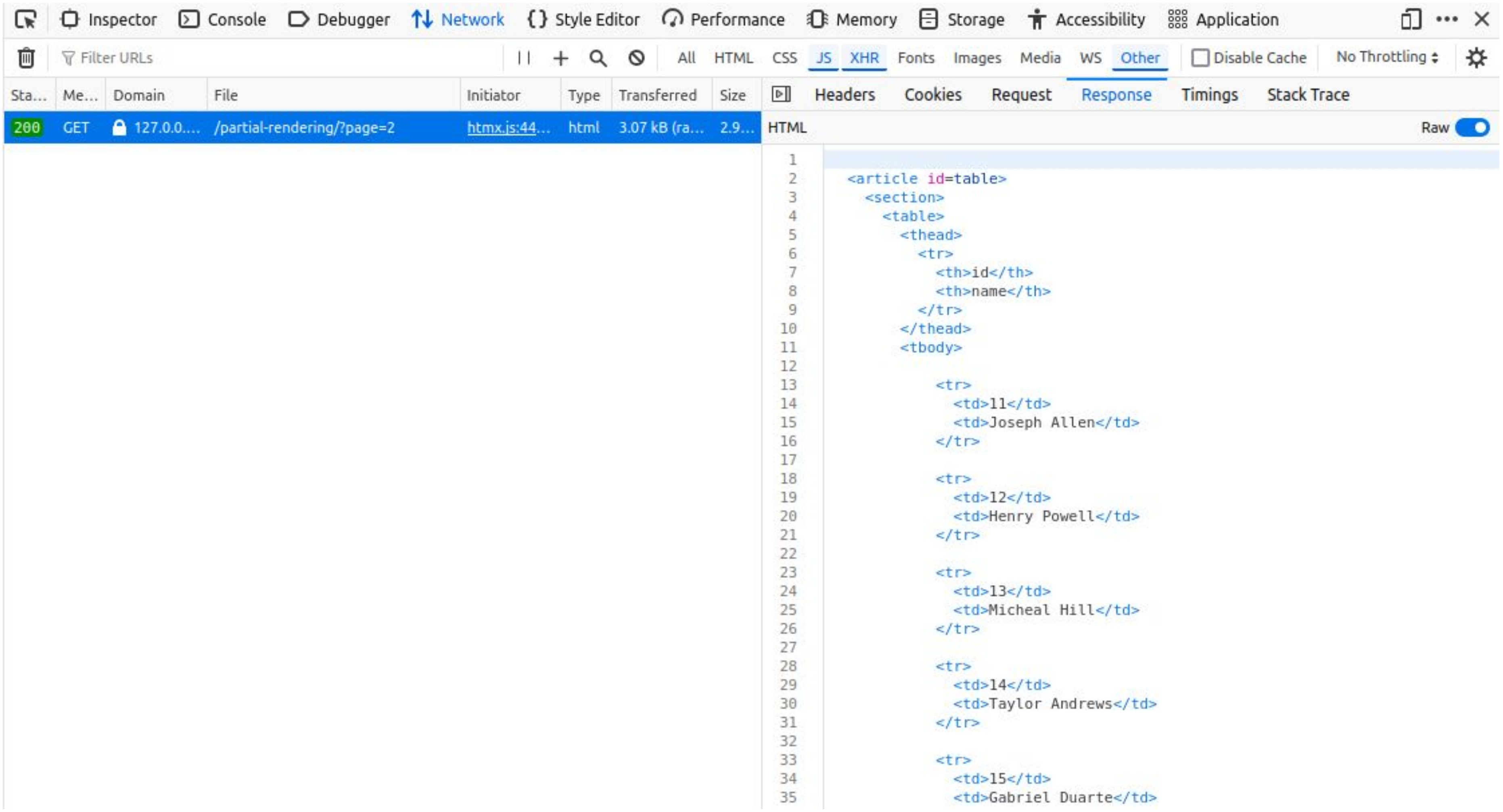


```
Sta...  Me...  Domain     File                           Initiator    Type  Transferred    Size
200     GET    127.0.0....  /partial-rendering/?page=2   htmx.js:44...  html  3.07 kB (ra...  2.9...
```

HTML

id	name
11	Joseph Allen
12	Henry Powell
13	Micheal Hill
14	Taylor Andrews
15	Gabriel Duarte
16	Jennifer Strong
17	Sherri Haley
18	Alex Fuller
19	Adrian Briggs
20	Justin Snyder

- « First
- 1
- 2
- 3
- » Last



```
200  GET  127.0.0.…  /partial-rendering/?page=2  htmx.js:44…  html  3.07 kB (ra…  2.9…
```

```

<article id=table>
  <section>
    <table>
      <thead>
        <tr>
          <th>id</th>
          <th>name</th>
        </tr>
      </thead>
      <tbody>

          <tr>
            <td>11</td>
            <td>Joseph Allen</td>
          </tr>

          <tr>
            <td>12</td>
            <td>Henry Powell</td>
          </tr>

          <tr>
            <td>13</td>
            <td>Micheal Hill</td>
          </tr>

          <tr>
            <td>14</td>
            <td>Taylor Andrews</td>
          </tr>

          <tr>
            <td>15</td>
            <td>Gabriel Duarte</td>
```

Final Thoughts

Photo by @derekthomson on Unsplash

Thanks for listening

Richard Shea
@shearichard
rshea@thecubagroup.com