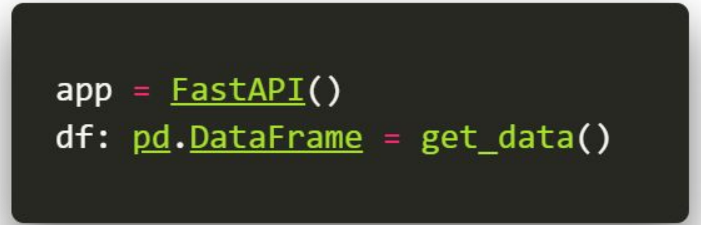
Build a Frontend for Your Web App:

From Basic HTML to JS Frameworks

About me

- Anastasia
- Backend engineer and former data scientist
- Live in Berlin

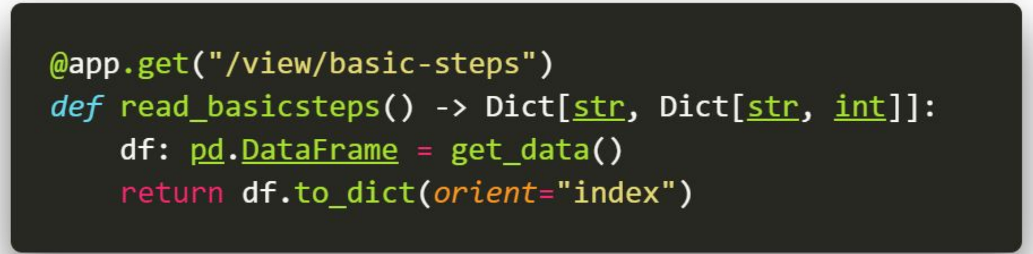
Initial condition



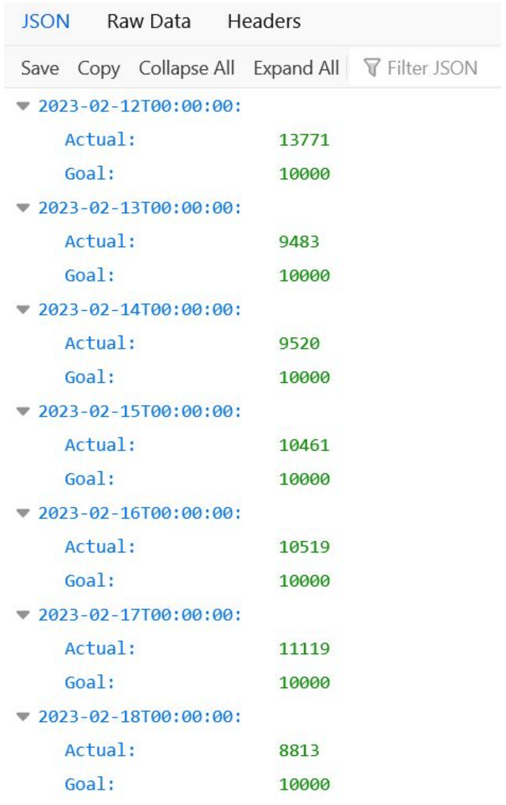
```
app = FastAPI()
df: pd.DataFrame = get_data()
```

Date	Goal	Actual
Feb 12	10000	13771
Feb 13	10000	9483

Quick and dirty

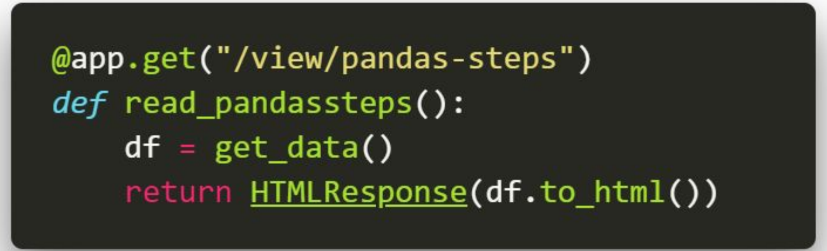


```python
@app.get("/view/basic-steps")
def read_basicsteps() -> Dict[str, Dict[str, int]]:
    df: pd.DataFrame = get_data()
    return df.to_dict(orient="index")
```

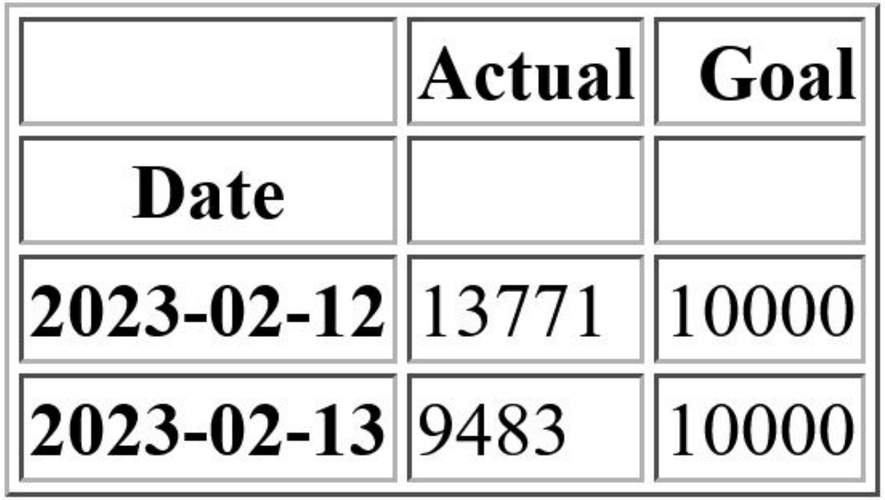


```
▼ 2023-02-12T00:00:00:
    Actual:        13771
    Goal:          10000
▼ 2023-02-13T00:00:00:
    Actual:        9483
    Goal:          10000
▼ 2023-02-14T00:00:00:
    Actual:        9520
    Goal:          10000
▼ 2023-02-15T00:00:00:
    Actual:        10461
    Goal:          10000
▼ 2023-02-16T00:00:00:
    Actual:        10519
    Goal:          10000
▼ 2023-02-17T00:00:00:
    Actual:        11119
    Goal:          10000
▼ 2023-02-18T00:00:00:
    Actual:        8813
    Goal:          10000
```

Pandas to help



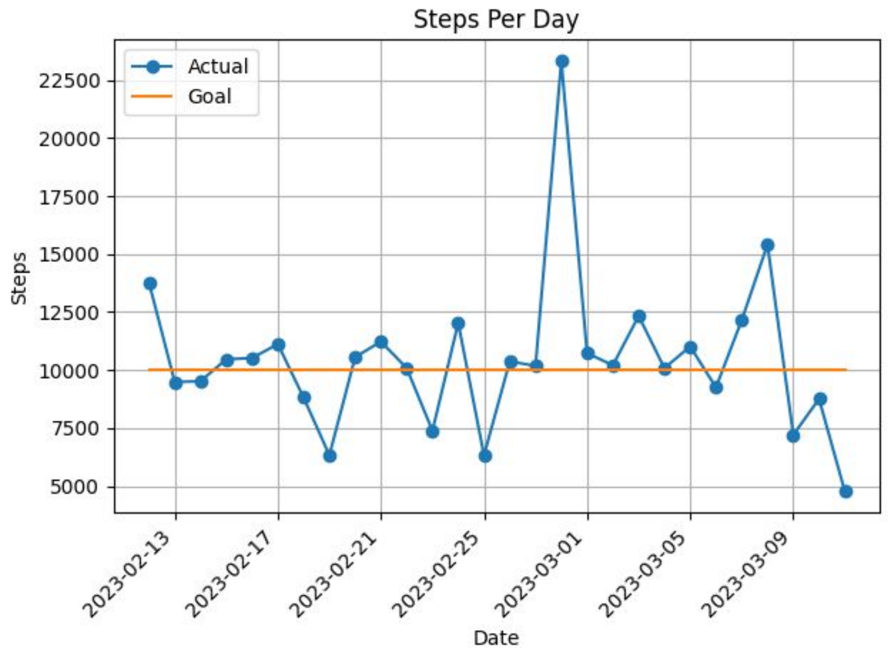
```
@app.get("/view/pandas-steps")
def read_pandassteps():
    df = get_data()
    return HTMLResponse(df.to_html())
```



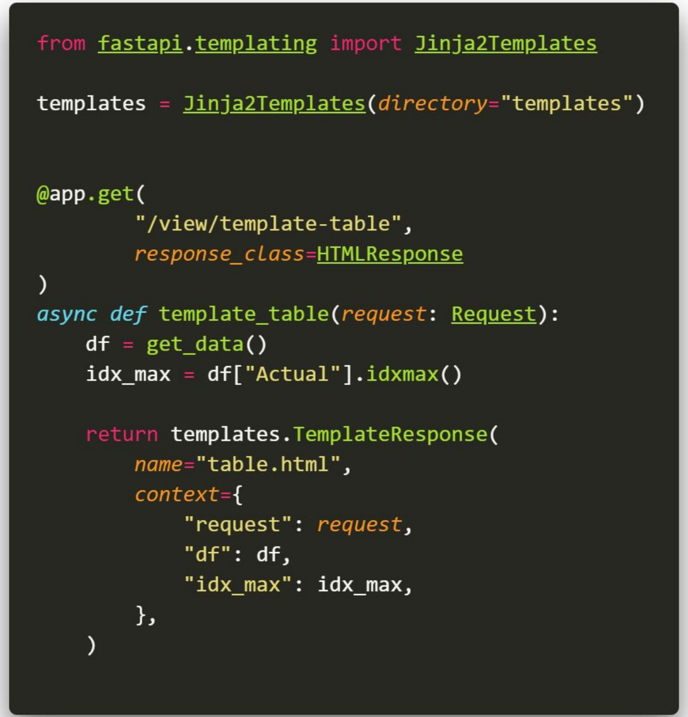
	Actual	Goal
Date		
2023-02-12	13771	10000
2023-02-13	9483	10000

Make a plot

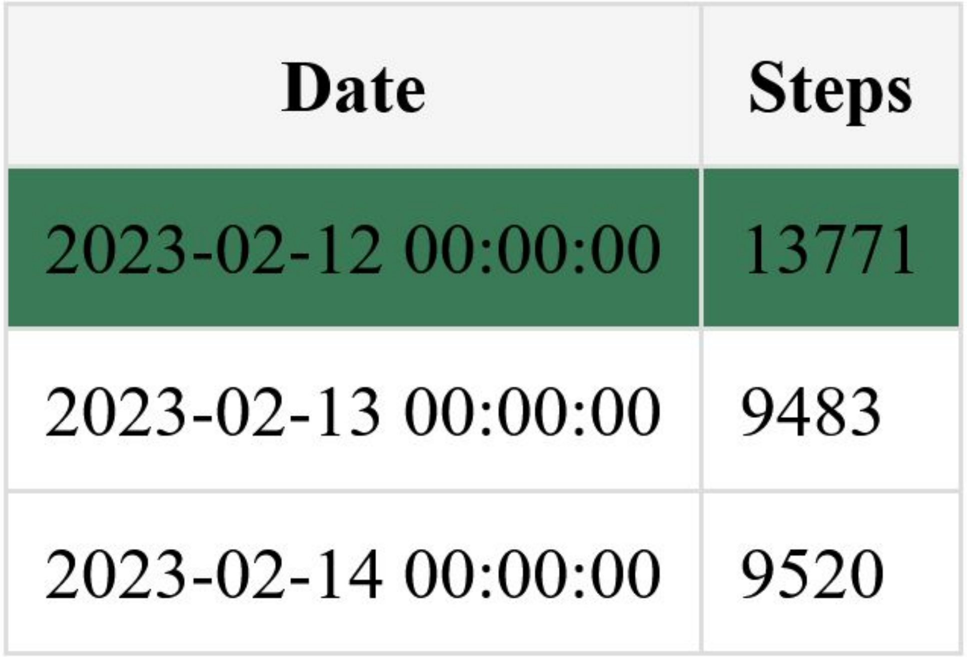
```python
@app.get("/view/matplotlib-plot")
def read_matplotlib_plot():
    df = get_data()
    create_basic_plot(df)

    buffer = io.BytesIO()
    plt.savefig(buffer, format="png")
    buffer.seek(0)

    return Response(
        content=buffer.getvalue(),
        media_type="image/png"
        )
```



HTML + Templates

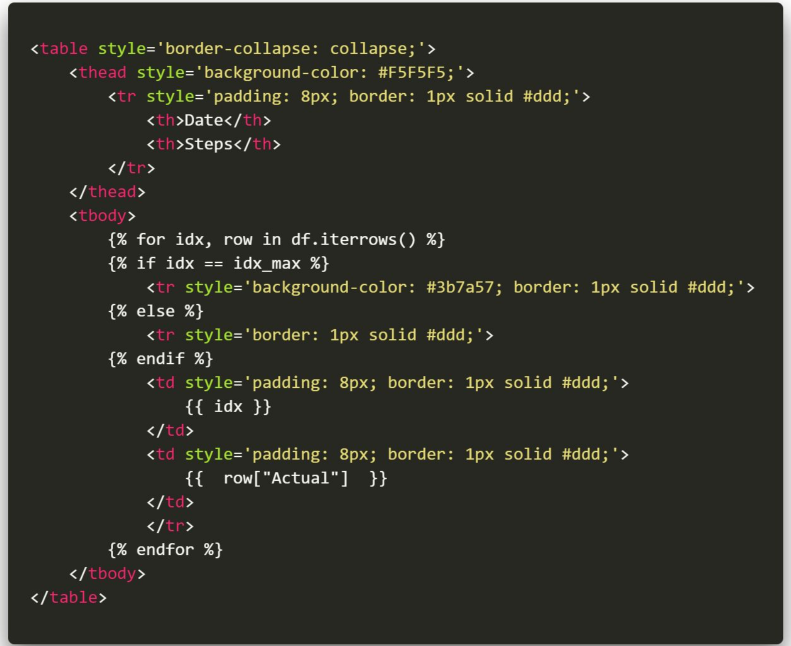


```python
from fastapi.templating import Jinja2Templates

templates = Jinja2Templates(directory="templates")


@app.get(
        "/view/template-table",
        response_class=HTMLResponse
)
async def template_table(request: Request):
    df = get_data()
    idx_max = df["Actual"].idxmax()

    return templates.TemplateResponse(
        name="table.html",
        context={
            "request": request,
            "df": df,
            "idx_max": idx_max,
        },
    )
```

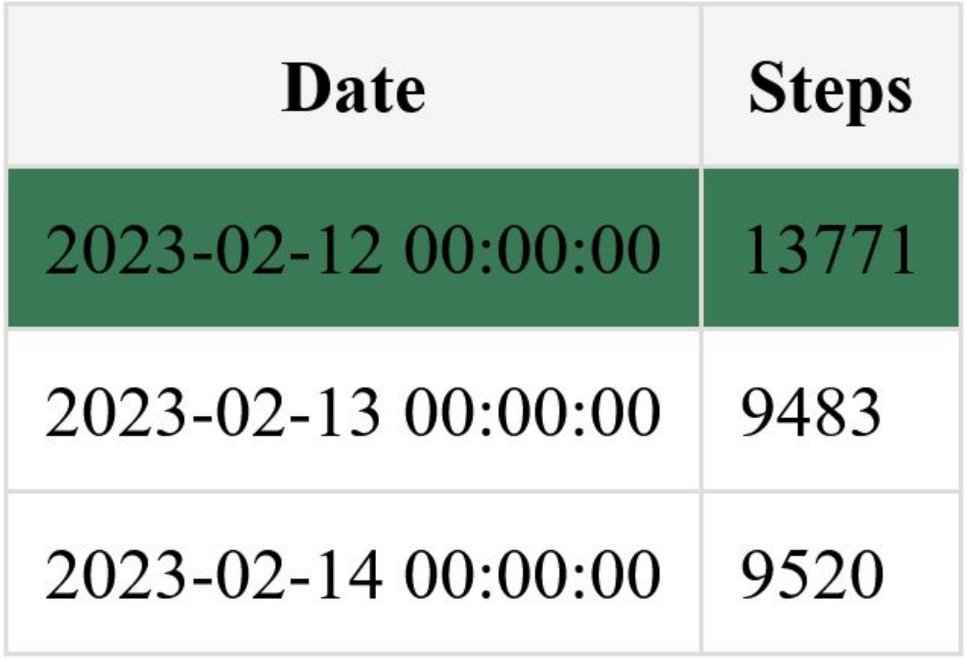


Date	Steps
2023-02-12 00:00:00	13771
2023-02-13 00:00:00	9483
2023-02-14 00:00:00	9520

HTML + Templates

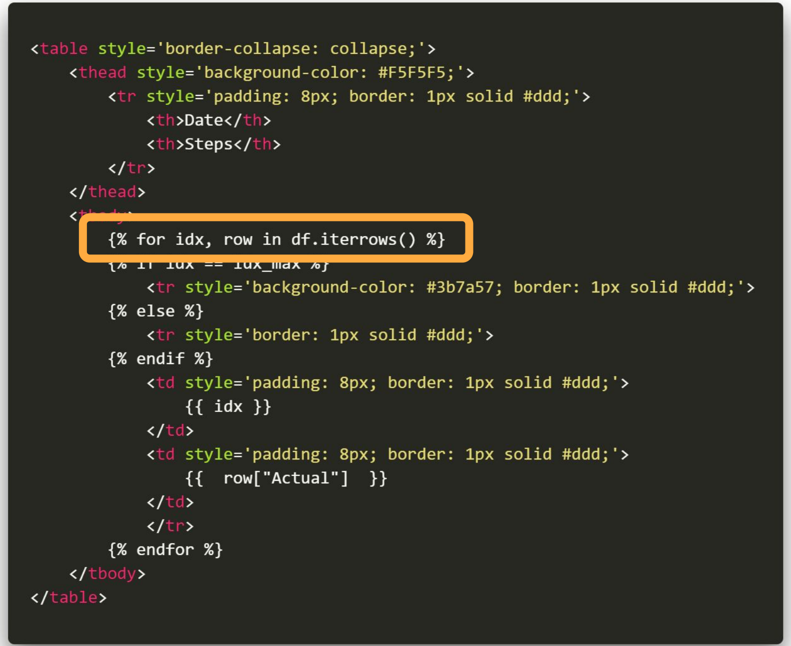


```
<table style='border-collapse: collapse;'>
    <thead style='background-color: #F5F5F5;'>
        <tr style='padding: 8px; border: 1px solid #ddd;'>
            <th>Date</th>
            <th>Steps</th>
        </tr>
    </thead>
    <tbody>
        {% for idx, row in df.iterrows() %}
        {% if idx == idx_max %}
            <tr style='background-color: #3b7a57; border: 1px solid #ddd;'>
        {% else %}
            <tr style='border: 1px solid #ddd;'>
        {% endif %}
            <td style='padding: 8px; border: 1px solid #ddd;'>
                {{ idx }}
            </td>
            <td style='padding: 8px; border: 1px solid #ddd;'>
                {{  row["Actual"]  }}
            </td>
            </tr>
        {% endfor %}
    </tbody>
</table>
```

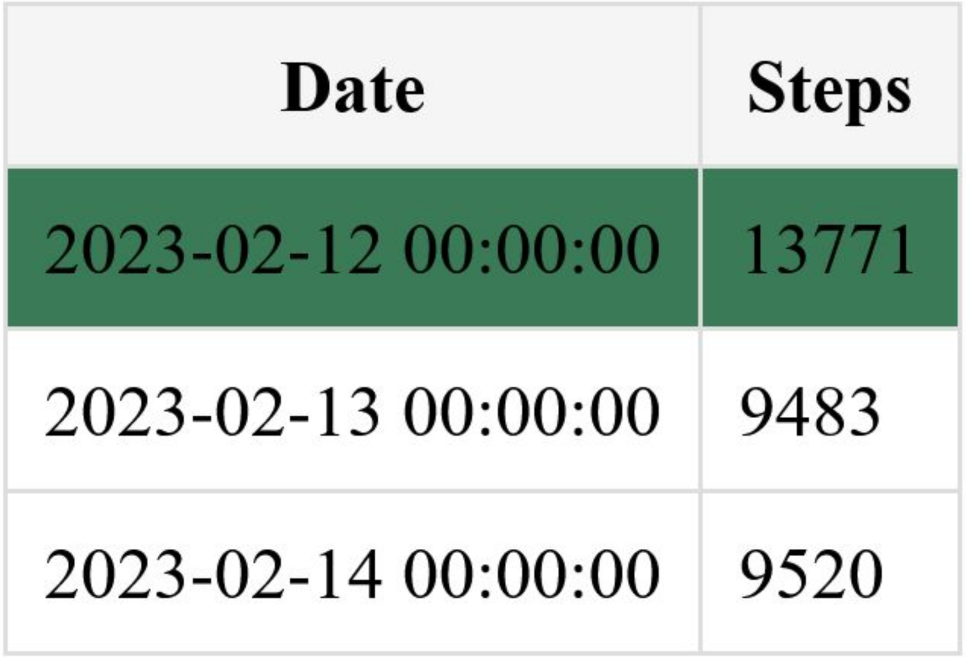


Date	Steps
2023-02-12 00:00:00	13771
2023-02-13 00:00:00	9483
2023-02-14 00:00:00	9520

HTML + Templates

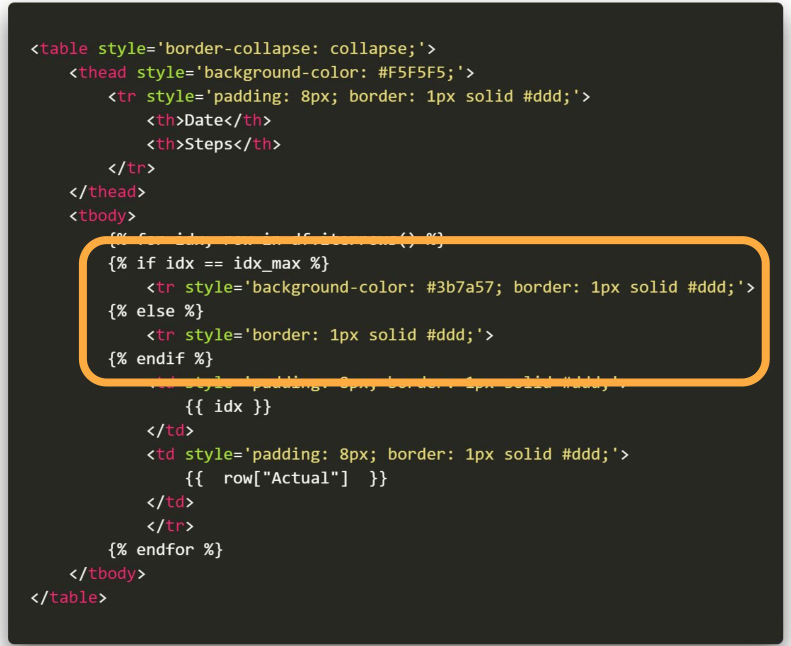


```
<table style='border-collapse: collapse;'>
    <thead style='background-color: #F5F5F5;'>
        <tr style='padding: 8px; border: 1px solid #ddd;'>
            <th>Date</th>
            <th>Steps</th>
        </tr>
    </thead>
    <tbody>
        {% for idx, row in df.iterrows() %}
        {% if idx == idx_max %}
            <tr style='background-color: #3b7a57; border: 1px solid #ddd;'>
        {% else %}
            <tr style='border: 1px solid #ddd;'>
        {% endif %}
            <td style='padding: 8px; border: 1px solid #ddd;'>
                {{ idx }}
            </td>
            <td style='padding: 8px; border: 1px solid #ddd;'>
                {{  row["Actual"]  }}
            </td>
            </tr>
        {% endfor %}
    </tbody>
</table>
```

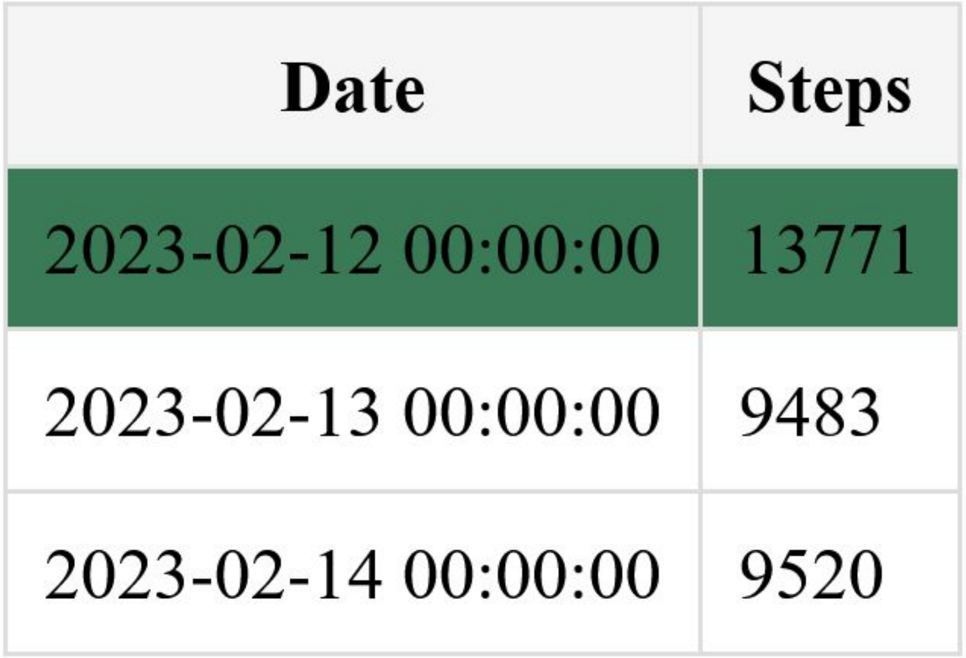


Date	Steps
2023-02-12 00:00:00	13771
2023-02-13 00:00:00	9483
2023-02-14 00:00:00	9520

HTML + Templates



```
<table style='border-collapse: collapse;'>
    <thead style='background-color: #F5F5F5;'>
        <tr style='padding: 8px; border: 1px solid #ddd;'>
            <th>Date</th>
            <th>Steps</th>
        </tr>
    </thead>
    <tbody>
        {% for idx, row in df.iterrows() %}
        {% if idx == idx_max %}
            <tr style='background-color: #3b7a57; border: 1px solid #ddd;'>
        {% else %}
            <tr style='border: 1px solid #ddd;'>
        {% endif %}
            <td style='padding: 8px; border: 1px solid #ddd;'>
                {{ idx }}
            </td>
            <td style='padding: 8px; border: 1px solid #ddd;'>
                {{  row["Actual"]  }}
            </td>
            </tr>
        {% endfor %}
    </tbody>
</table>
```



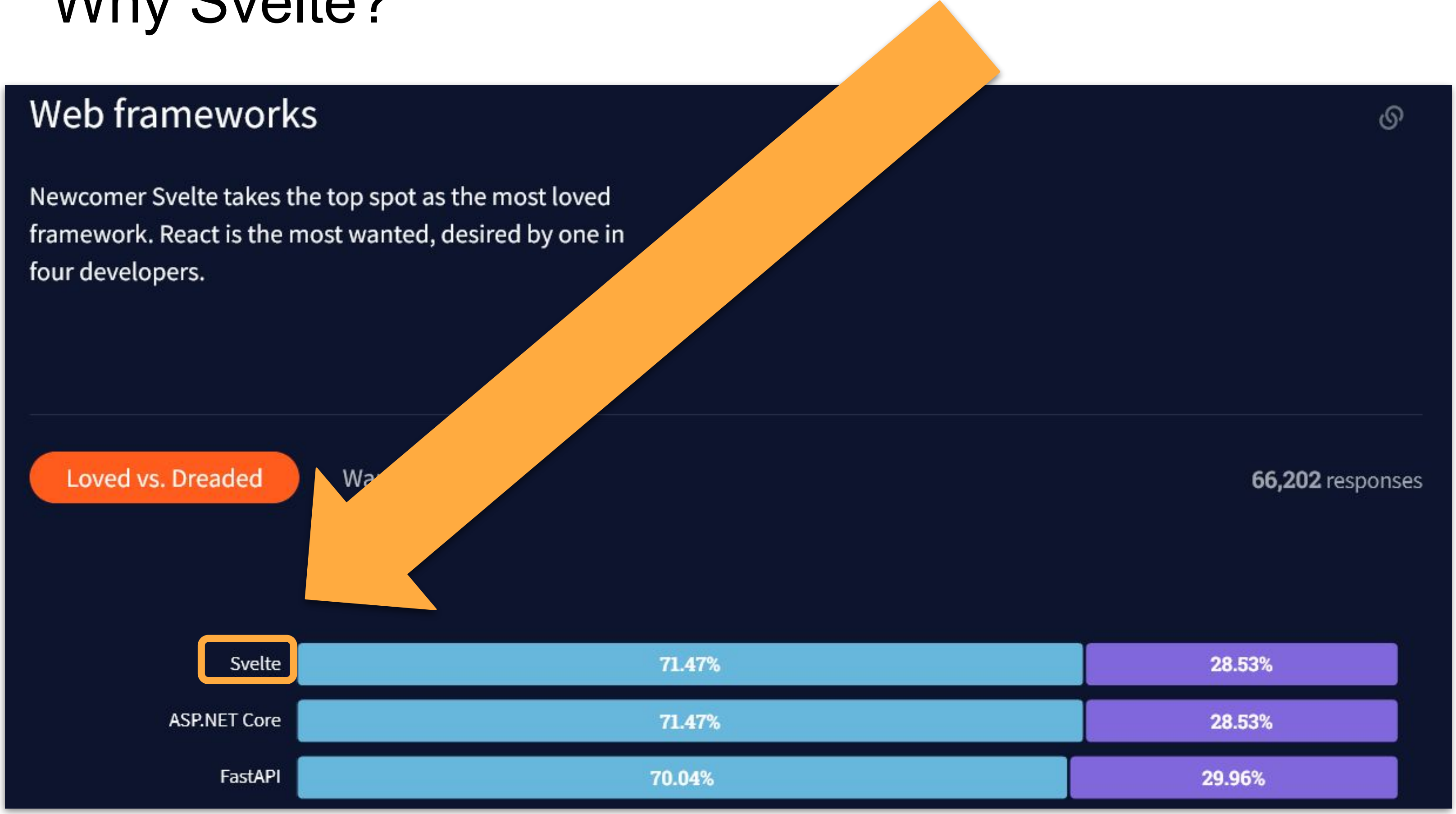
Date	Steps
2023-02-12 00:00:00	13771
2023-02-13 00:00:00	9483
2023-02-14 00:00:00	9520

JS Frameworks

JS Frameworks

- Angular
- Vue
- React
- Svelte

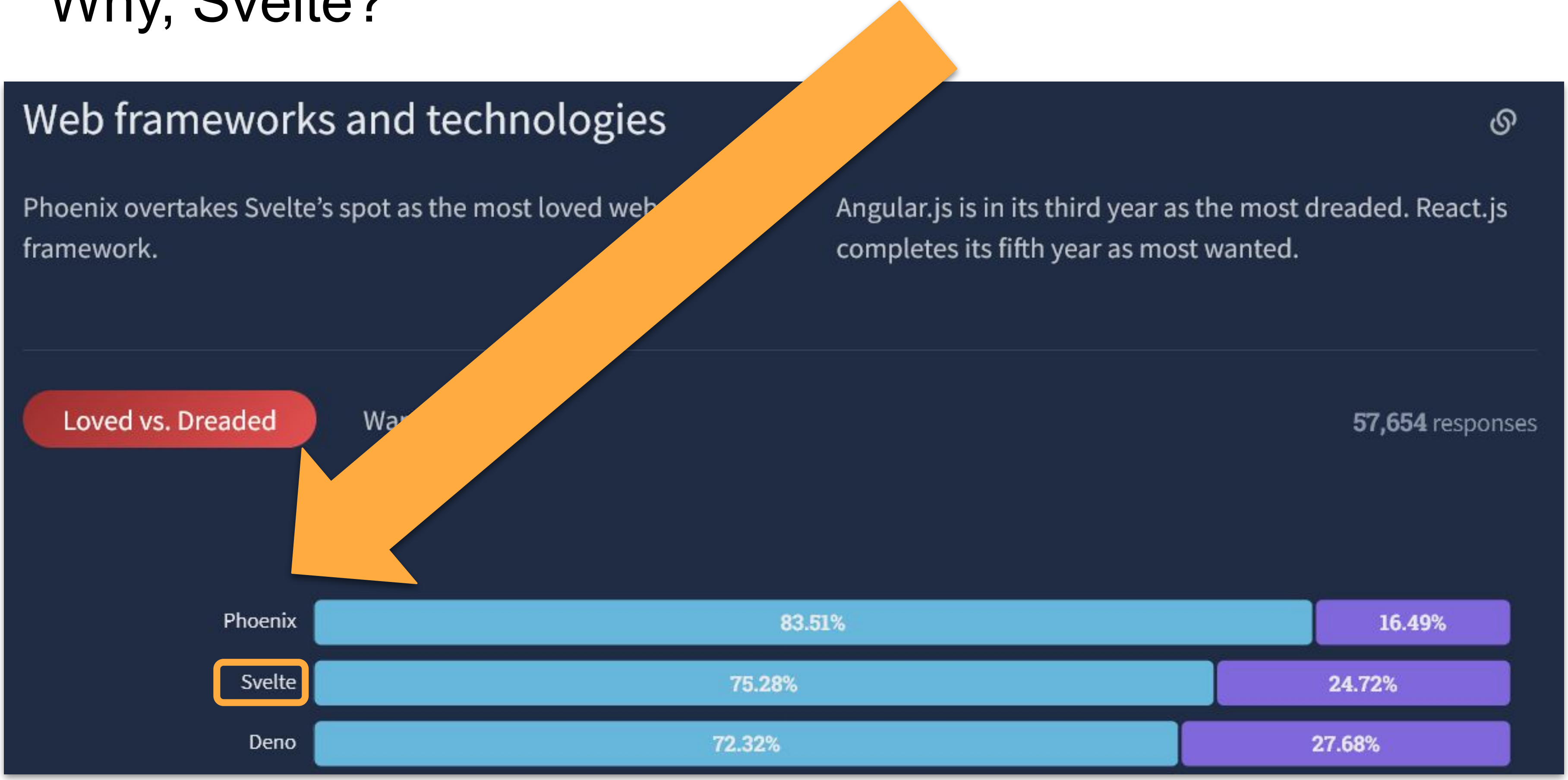
Why Svelte?


Web frameworks

Newcomer Svelte takes the top spot as the most loved framework. React is the most wanted, desired by one in four developers.

Svelte	71.47%	28.53%
ASP.NET Core	71.47%	28.53%
FastAPI	70.04%	29.96%

Why, Svelte?



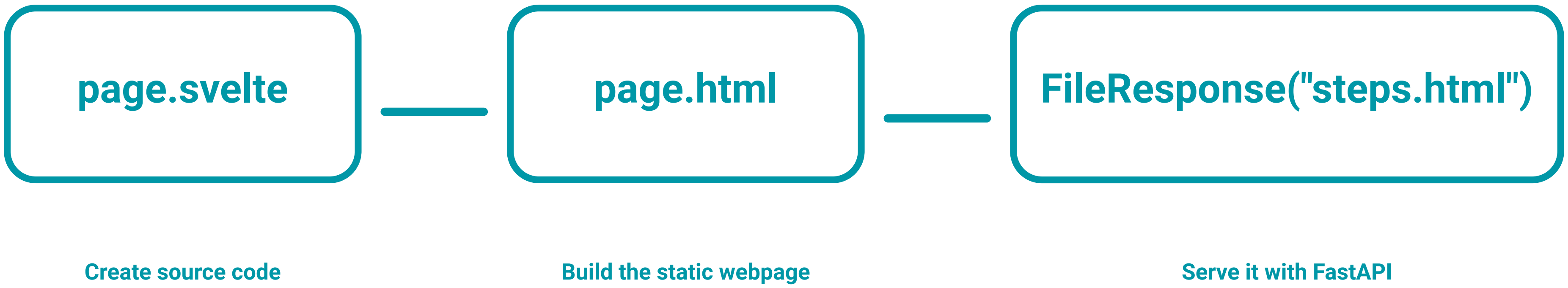
Web frameworks and technologies

Phoenix overtakes Svelte's spot as the most loved web framework.

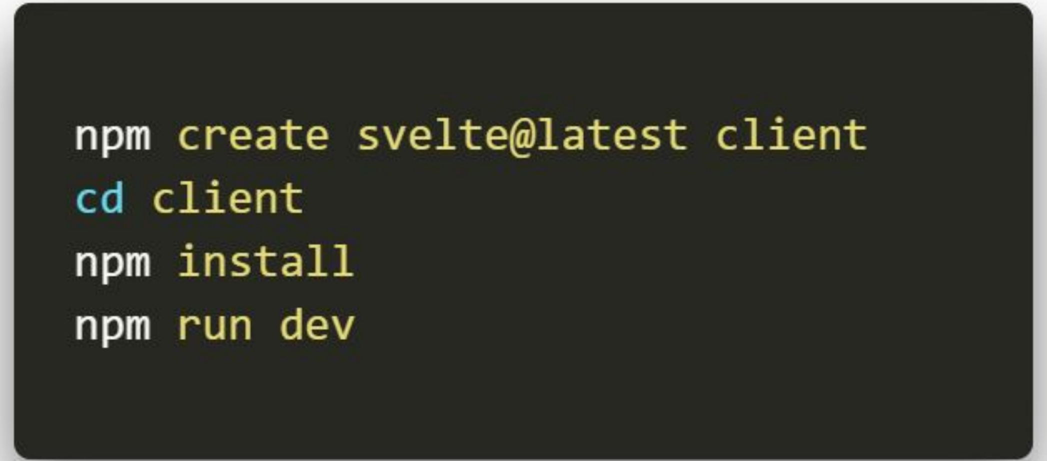
Angular.js is in its third year as the most dreaded. React.js completes its fifth year as most wanted.

Phoenix	83.51%	16.49%
Svelte	75.28%	24.72%
Deno	72.32%	27.68%

Plan


page.svelte
page.html
FileResponse("steps.html")
Create source code
Build the static webpage
Serve it with FastAPI

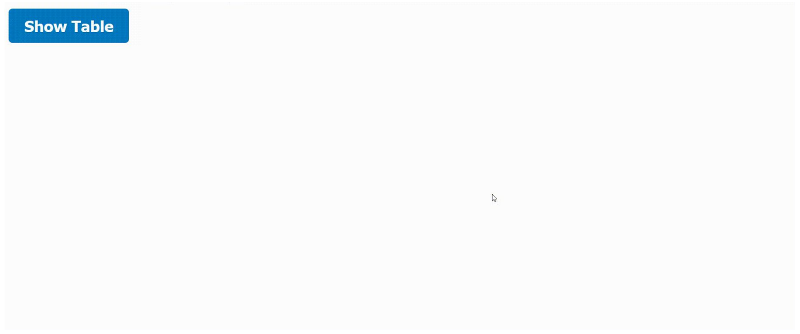
Create Svelte project



```
npm create svelte@latest client
cd client
npm install
npm run dev
```

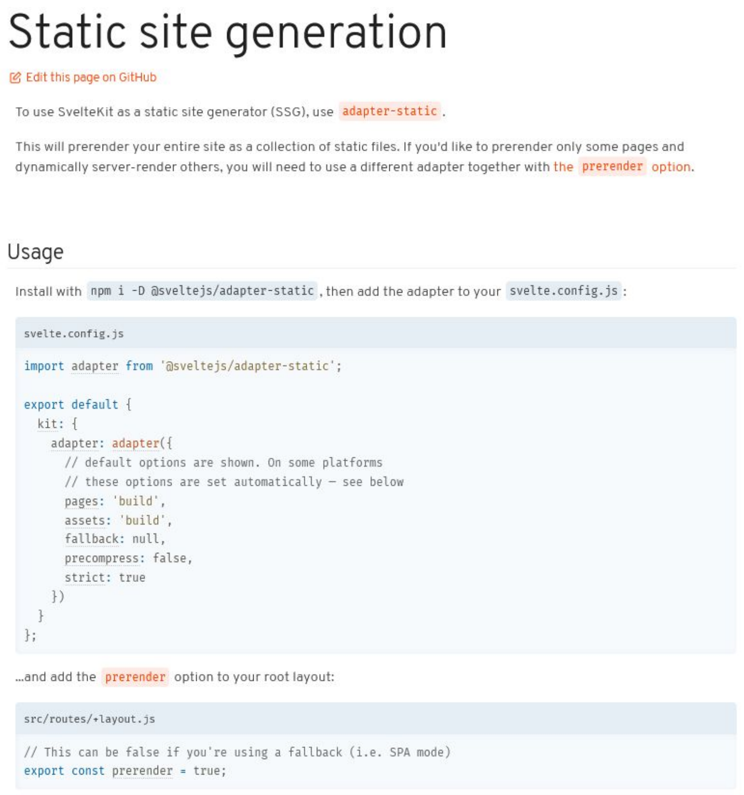
Create a page

```
{#if showTable}
<div class="table-container">
    <h2>Steps Data</h2>
    <table>
        <thead>
            <tr>
                <th>Date</th>
                <th>Actual Steps</th>
                <th>Goal Steps</th>
            </tr>
        </thead>
        <tbody>
            {#each Object.entries(stepsData) as [date, { Actual, Goal }]}
            <tr>
                <td>{date}</td>
                <td>{Actual}</td>
                <td>{Goal}</td>
            </tr>
            {/each}
        </tbody>
    </table>
</div>
{/if}
```



Static site generation

Svelte documentation will guide you through it



Static site generation

Edit this page on GitHub

To use SvelteKit as a static site generator (SSG), use `adapter-static`.

This will prerender your entire site as a collection of static files. If you'd like to prerender only some pages and dynamically server-render others, you will need to use a different adapter together with the `prerender` option.

Usage

Install with `npm i -D @sveltejs/adapter-static`, then add the adapter to your `svelte.config.js`:

svelte.config.js

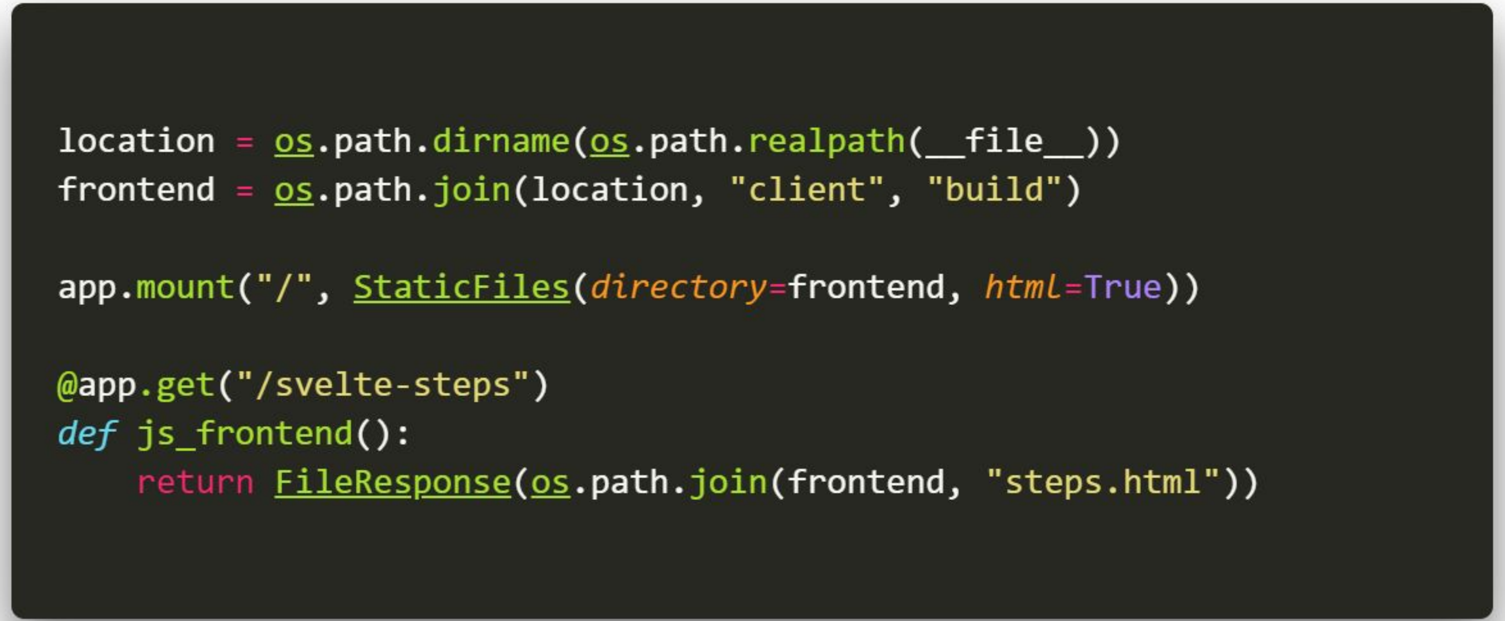
```js
import adapter from '@sveltejs/adapter-static';

export default {
  kit: {
    adapter: adapter({
      // default options are shown. On some platforms
      // these options are set automatically — see below
      pages: 'build',
      assets: 'build',
      fallback: null,
      precompress: false,
      strict: true
    })
  }
};
```

...and add the `prerender` option to your root layout:

src/routes/+layout.js

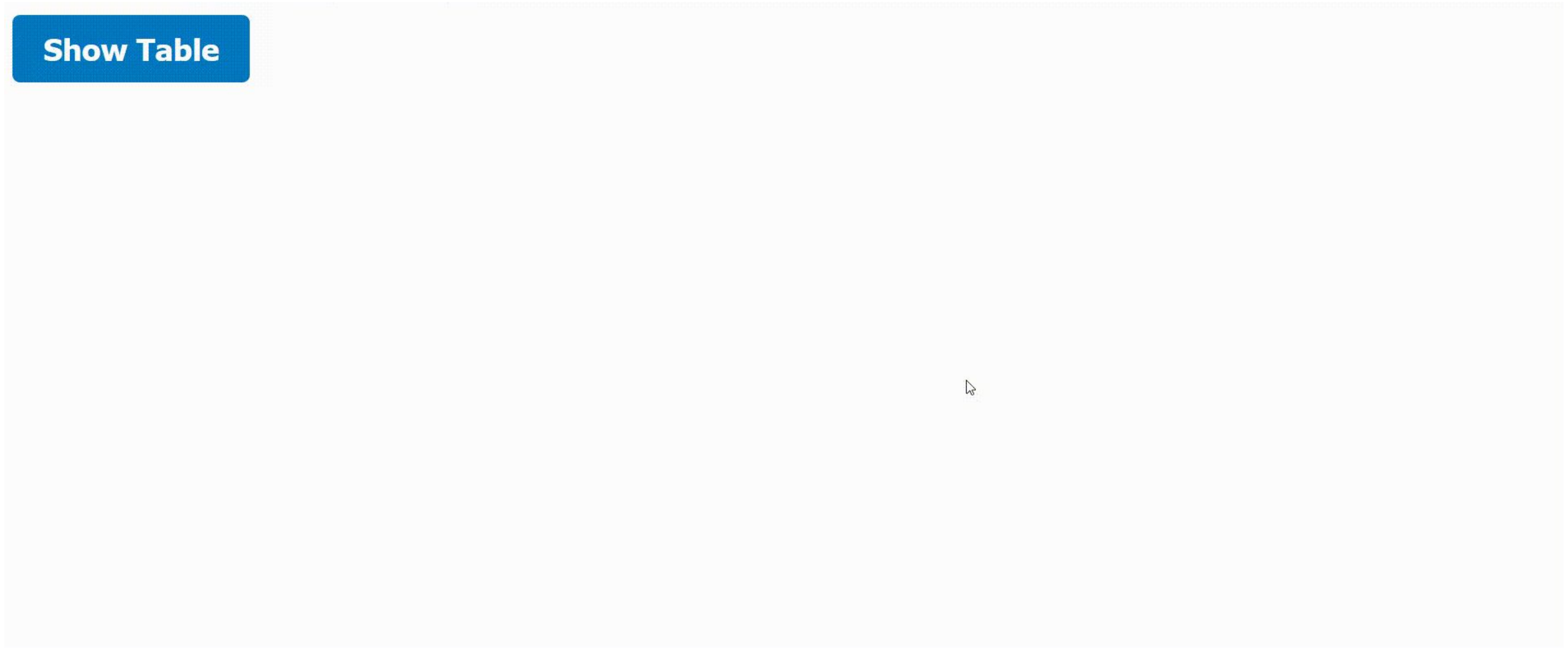
```js
// This can be false if you're using a fallback (i.e. SPA mode)
export const prerender = true;
```

Back to Python

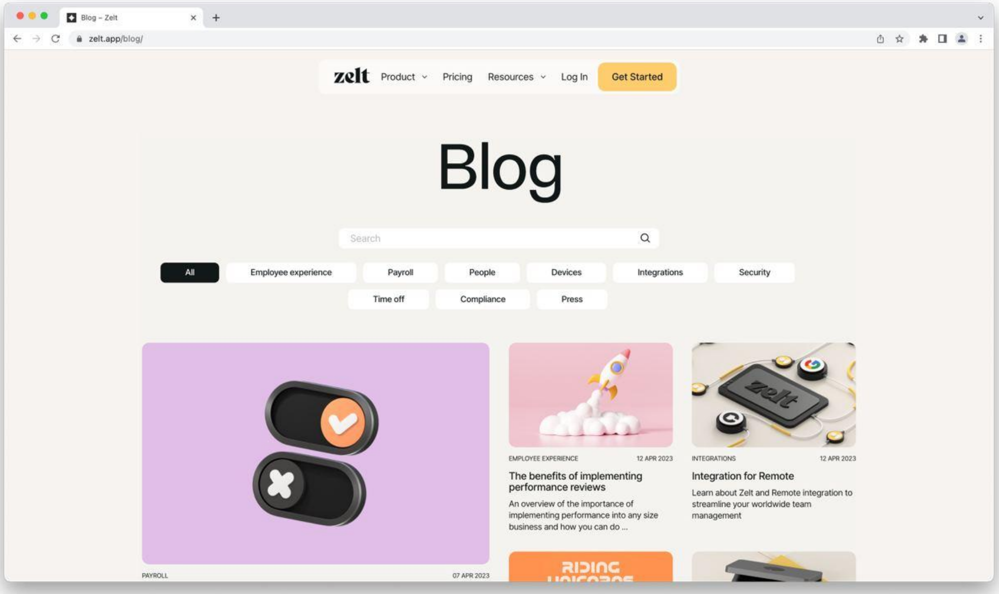


```python
location = os.path.dirname(os.path.realpath(__file__))
frontend = os.path.join(location, "client", "build")

app.mount("/", StaticFiles(directory=frontend, html=True))

@app.get("/svelte-steps")
def js_frontend():
    return FileResponse(os.path.join(frontend, "steps.html"))
```

You are beautiful



People do pretty stuff



EMPLOYEE EXPERIENCE 12 APR 2023

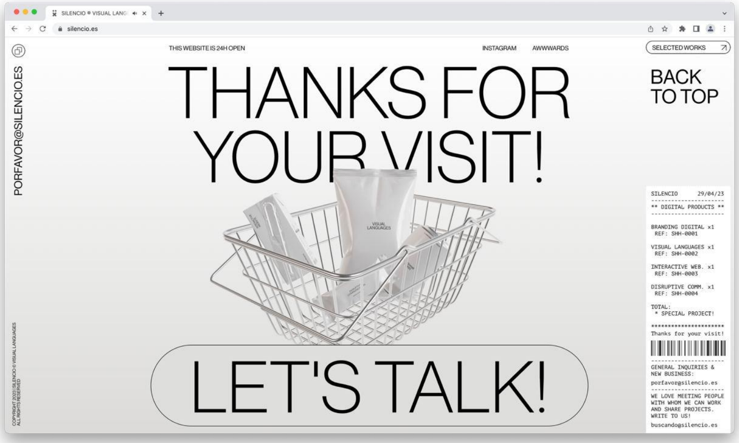
The benefits of implementing performance reviews

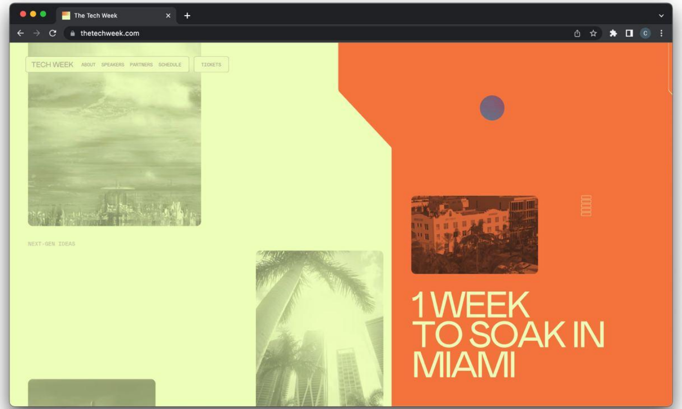
An overview of the importance of implementing performance into any size business and how you can do ...

INTEGRATIONS 12 APR 2023

Integration for Remote

Learn about Zelt and Remote integration to streamline your worldwide team management





Screenshots by https://t.me/nowhow

Wrap up

Pandas df as a table / Matplotlib plots as bytes

⭐ lightning fast

⭐ no learning curve for python devs

🌧️ hard to make sense of data

🌧️ not much power over output

HTML / templating

⭐ simple

🌧️ cumbersome

JS framework

⭐ powerful

🌧️ steep learning curve for python devs

🌧️ time consuming

Reading list

Source code of the talk

MDN Web Docs: Understanding client-side JavaScript frameworks

Stack Overflow Annual Developer Survey

Svelte: Static site generation

React examples: Tech week, silencio.es, Zelt