General Assignment

What is it?

In this exercise, you'll deploy a Cloudflare Workers project in order to build a linktree-style website. Below is a list of requirements and success criteria for your finished project.

Requirements

Deploy a JSON API
Create a new Workers project using Wrangler. This project will respond to two kinds of requests, one to generate a JSON API (defined below), and second, to serve an HTML page (see "Set up an HTML page")

To begin, you should define an array of links. **The links should be a JavaScript array, with a number of link objects, each with a name and URL string.** See the below example for one of these link objects:

```json
{ "name": "Link Name", "url": "https://linkurl" }
```

You should define at least three link objects inside of your array.

Once you've defined this array, set up a request handler to respond to the path `/links`, and return the array itself as the root of your JSON response.

In addition, you should ensure that the API response has the correct `Content-Type` header to indicate to clients that it is a JSON response.

You should be able to test that this works as expect by running `wrangler dev` to access your Workers application locally. Visit `localhost:8000/links` to confirm that your application returns the link array as a JSON response.

Set up an HTML page
With your API deployed, you can flesh out the rest of your application. If the path requested _is not_ `/links`, your application should render a static HTML page, by doing the following steps:

1. Retrieve a static HTML page
2. Get the links from your previously deployed JSON response
3. Use HTMLRewriter to add these links to the static HTML page
4. Return the transformed HTML page from the Worker

Retrieve a static HTML page
Your Worker should begin by making a `fetch` request to `https://static-links-page.signalnerve.workers.dev`.

The response from this URL will be a static HTML page that you can enhance using HTMLRewriter.

Note that you need to make the request to this HTML page _in_ your Worker. The URL will return multiple static HTML pages randomly, so you should not copy-paste the HTML into your Worker, as you will not complete the exercise correctly.

Use HTMLRewriter to update the page
Using HTMLRewriter, you should transform the static HTML response from $URL, and pass in the links array you previously defined into the page. Note that this means that your links array should be available as a variable for both your previous `/links` endpoint, as well as this static HTML section. Target the `div#links` selector, and add in a new `a` for each link in your API using HTMLRewriter.

In order to use the links inside of your HTMLRewriter handler, you can use a custom class to pass in arguments, for instance:

```js
class LinksTransformer {
```

~~~
  constructor(links) {
    this.links = links
  }

  async element(element) {
    // Your code
  }
}
```

Here's what the output of the `div#links` section should look like:

```html
<div id="links">
  <a href="https://asampleurl.com">A sample URL</a>
  <a href="https://anothersampleurl.com">Another sample URL</a>
  <a href="https://afinalsampleurl.com">A final sample URL</a>
</div>
```

You should also remove the `display: none` from the `div#profile` container, and inside of it, update
the two child elements to show your user avatar and name.

To do this, target `img#avatar` and set the `src` attribute to a profile image.

Do the same for `h1#name`, setting the text to your username.

#### Return the transformed HTML page from the Worker

Once you've passed the static HTML response through your HTMLRewriter instance, you can return it as
the response from the Worker.

Ensure that the correct `Content-type` header is set for this page, allowing it to render as HTML in
a browser.

Once you've completed the code for this project, you can deploy it using `wrangler publish`. Follow
our Quick Start guide to learn how to deploy Workers applications. You should submit both the JSON
API URL and your deployed webpage for grading.

#### Add a link to your deployed worker to your repo

Create a URL.txt that contains one line in it with the URL of your page.  For example if my
assignment was deployed at https://myassignment.mydomain.com/ I would put the following in URL.txt:

```
https://myassignment.mydomain.com/
```

We will test your code by visiting the URL in that file and the /links endpoint as well.  In the
above example we would test https://myassignment.mydomain.com/ and
https://myassignment.mydomain.com/links .

### Extra Credit
1. Provide social links

Remove the `display: none` style from `div#social`, and add any number of social links as children to
the container. The children of this container should be `a` links, with `svg` icons as the children
of those links. For example SVGs, use https://simpleicons.org.

```js
<div id="links">
  <a href="someurl.com">
    <svg></svg>
  </a>
</div>
~~~

~~~
```

2. Update the title field

Using the HTMLRewriter, target the `title` tag and update the text inside to your name.

3. Change the background color:

Using HTMLRewriter, target the `body` element and change the background color. You can simply swap
out the `bg-gray-900` class with another class from [Tailwind CSS's color palette]
(https://tailwindcss.com/docs/customizing-colors), or set a `background-color` style to a color,
gradient, or image of your choice.

4. Do the [Systems Assignment](https://github.com/cloudflare-hiring/cloudflare-2020-systems-
engineering-assignment) that depends on the work you've done here.
~~~