CMPS 356 – Fall 2022
Web Applications Design and Development

Lab 07
Rendering with Next

Objective

1. Using Next's SWR hook for data fetching
2. Rendering using client-side rendering (CSR)
3. Rendering using server-side rendering (SSR) with `getServerSideProps`
4. Rendering using static-site generation (SSG) with `getStaticProps` and `getStaticPaths`
5. Rendering using incremental-site generation (ISG)
6. Pre-rendering with default data
7. Mixing and using multiple rendering modes

Prerequisites

1. Next tutorial: https://nextjs.org/learn/foundations/about-nextjs
2. Data fetching overview: https://nextjs.org/docs/basic-features/data-fetching
3. SWR hook: https://swr.vercel.app

1. Client-side Rendering (CSR)

1. Create a new directory `01-csr` and a Next application under it.
2. Create a new page and use SWR with a mount effect hook to fetch data on the client-side from PokéAPI (https://pokeapi.co/docs/v2).
3. Update your implementation to use Suspense for showing the status of the request.

2. Server-side Rendering (SSR)

1. Create a new directory `02-ssr` and reuse the code from the previous exercise.
2. Create a new page and use SWR with `getServerSideProps` to fetch data on the server-side from PokéAPI: https://pokeapi.co/docs/v2.
3. Create a navigation bar that provides links to the two pages you have created so far.

3. Static-site Generation (SSG)

1. Create a new directory `03-ssg` and reuse the code from the previous exercise.
2. Create a new page and use SWR with `getStaticProps` and `getStaticPaths` to fetch data on the server-side from PokéAPI: https://pokeapi.co/docs/v2.
3. Update your navigation bar with a link to the newly created page.

4. Incremental Static Regeneration (ISR)

1. Create a new directory `04-isr` and reuse the code from the previous exercise.

2. Regenerate your pages automatically at regular intervals. This can be achieved by adding a field "`revalidate: 30`" to update your content every 30 seconds, for example, regardless of whether it has changed or not.
3. Regenerate a certain page on-demand using an API endpoint: "`/api/revalidate`" and test it with new content. Use a secret token stored in an environment variable to protect the API endpoint.
4. How can we use this endpoint to revalidate any given page in our application?