WeHelp
Guide - Part 2

This week, we have to complete a RWD layout for the website homepage, where users can browse attractions and search them by keyword. **In this project, don't use third-party libraries for front-end development, e.g. jQuery, Bootstrap, and etc.**

For separation of the front-end and back-end architecture, never modify any code marked as # Static Pages.

Part 2-1: RWD Layout

Complete a RWD layout for the website homepage based on the design in Figma, excluding pop-up dialog for member sign up/sign in.

Part 2-2: Fetch Attraction API

Right after page load, fetch Attraction API (/api/attractions) without the keyword for the first page data and render them by JavaScript. We should render the attraction name, category and MRT for each attraction data.

Use the first image of each attraction for the list in the homepage.

Part 2-3: Fetch Attraction API for More Data Automatically

When users scroll down to the bottom of the page, our JavaScript code should load the next page of data and render them on the bottom of the list. Refer to the steps below:

1. After the first page data is loaded, we save the `nextPage` field, which is included in the response from the server, in a variable for the next page load.
2. Detect if users scroll down to the bottom of the page: use IntersectionObserver object, or use window scroll event with getBoundingClientRect method defined in HTML Element to do detection.
3. If users scroll down to the bottom of the page, fetch Attraction API to load the next page. **Because network speed is much slower than CPU operations, you should test the ONLINE version of your page carefully, preventing fetch API multiple times in an instant.**
4. After step 3, render attraction data on the bottom of the list, save the `nextPage` field, which is from the latest response of API, to the same variable in step 1. Repeat steps 1 to 4 until the `nextPage` field is null.

Part 2-4: Keyword Search Feature

When users enter a keyword in the text input element on the top and click the search button, our code should fetch Attraction API (/api/attractions) with the entered keyword for filtered attraction data **without refreshing the page.**

After fetching, we render the attraction list based on filtered data, replacing an existing list on the page. Follow Part 2-3 to complete the same feature which can load more pages of data and render them, if it exists.

Part 2-5: MRT Name List and Filtered by MRT Name

We should support MRT related features as described below:

1. Right after page load, fetch MRT Station API (/api/mrts) for MRT Station names which should be rendered in a scrollable interface based on the design in Figma.
2. Users can click left/right arrows to scroll for more MRT Station names.
3. If users click one of the MRT Station names, our code will fill the clicked name into the text input element on the top, and fetch Attraction API (/api/attractions) with MRT Station name as keyword.
4. After fetching, we render the attraction list based on filtered data, replacing an existing list on the page. Follow Part 2-3 to complete the same feature which can load more pages of data and render them, if it exists.

How to Submit Task:

Following the Git, GitHub workflow we have done in Part 0. Use the develop branch to do any development, deployment, and testing. When you want to submit your task, create a Pull Request in the GitHub website to request merging from develop to main branch. Invite Chao-Wei Peng (cwpeng) as code reviewer, wait for the feedback.

Write down your name and online API URLs in the Pull Request comment.
For example:
彭兆蔚
http://140.112.3.5:8000/