COMP 3612 Assignment #3

Due Friday December 9th at midnight
Version 1, Nov 17

Overview

This assignment provides you with an opportunity to create an API in Node. The assignment APIs will also allow you to make use of the different JavaScript array functions.

Files

You will be able (eventually) to find the three data files (one for artists, one for galleries, one for paintings) at the GitHub repo for the assignment:

`https://github.com/mru-comp3612-archive/f2022-assign3`

Grading

The grade for this assignment will be broken down as follows:

Programming Design and Documentation	15%
Hosting + Correct readme	10%
Functionality (follows requirements)	75%

Recommended Workflow

I recommend you approach this assignment in the following order:

1. Complete the Node Lab
2. Set up github repo for your source code.
3. Implement the artists APIs
4. Implement the galleries APIs.
5. Implement the paintings APIs.
6. Set up the hosting (perhaps by first working through the provided hosting lab).
7. Test the APIs after hosting by constructing readme file with example API request links

Submitting and Hosting

You will be using Node in this assignment. This will mean your assignment will need to reside on a working host server. Static hosts used in the first assignment will not work for this one.

For this assignment, we are going to try glitch.com, which provides a free option for hosting simple Node applications. Do note that glitch projects will go to sleep after 5 minutes, so be aware that the first request of a slept node application will take some time to awaken.

The hosting should be arranged and tested a few days before the assignment is completed!!!

When your glitch hosting is working and the assignment is ready to be marked, then send me an email with the following information:

- The URL of the github repo so that I can mark the source code. If your repo is private, then add me as a collaborator.
- In the `readme.md` file for your repo, provide a link to each of the APIs on glitch so I can test them (see details below).

API Functionality

You must create the following APIs with the specified routes and functionality.

/api/paintings	Returns JSON for all paintings
/api/painting/id	Returns JSON for the single painting whose id matches the provided id.
/api/painting/gallery/id	Returns JSON for the paintings whose gallery id matches the provided gallery id.
/api/painting/artist/id	Returns JSON for the paintings whose artist id matches the provided artist id.
/api/painting/year/min/max	Returns all paintings whose yearOfWork field is between the two supplied values.
/api/painting/title/text	Returns JSON for the paintings whose title contains (somewhere) the provided text. This search should be case insensitive.
/api/painting/color/name	Returns JSON for the paintings that have a color that matches the provided hex value. Each painting has a dominantColors array with the six most common colors in the painting; each of these color values comes with a property named name that contains the name for that color. This should be case insensitive.
/api/artists	Returns JSON for all artists
/api/artists/country	Returns JSON for all artists from the specified country. This should be case insensitive.
/api/galleries	Returns JSON for all galleries
/api/galleries/country	Returns JSON for all galleries from the specified country. This should be case insensitive.

For each of the requests that take parameters, your API needs to handle a Not Found condition. For instance, if an id doesn't exist, return a JSON message that indicates the requested request did not return any data.

Example API Requests

In the `readme.md` file for your assignment repo, you must supply a list of links that allow me to test each of your APIs. Please add the following test links in this file:

`/api/paintings`

`/api/painting/433`

`/api/painting/43374534856`

`/api/painting/gallery/7`

`/api/painting/gallery/43374534856`

`/api/painting/artist/106`

`/api/painting/artist/43374534856`

`/api/painting/year/1850/1900`

`/api/painting/year/2200/2400`

`/api/painting/title/self`

`/api/painting/title/dfjkghdfkgh`

`/api/painting/color/NAPA`

`/api/painting/color/coffee%20bean`

`/api/painting/color/kcvhvxchbkcj`

`/api/artists`

`/api/artists/Netherlands`

`/api/artists/sdfjjsdf`

`/api/galleries`

`/api/galleries/france`

`/api/galleries/kcvhvxchbkcj`

Note: you will need to preface the above URLs with the URL of your host. For instance, if your Glitch URL is `https:/smashing-squirrels.glitch.me`, then the URL for the second test link would be `https:/smashing-squirrels.glitch.me/paintings/433`