# Testing Document for the Metro Art program.

I wanted to have a go at API interaction as well, as it’s something I hadn’t done, and I’ve had an idea for a random painting generator. Sort of a ‘daily dose of culture’ webpage. So combining that with a Typescript program, something I also want experience in, seemed like a good idea.

I found that the The Metropolitan Museum of Art Collection API (<https://metmuseum.github.io/>) is completely open to the public, not even requiring a login or password, and seemed exactly what I was looking for.

The code is available on my github: <https://github.com/yumyab/metro-art>

And it’s live at <https://mattt.net.au/paintings/>

In Backend Web development I learned to use the Windows Subsystem for Linux (WSL) as a development environment, and I continued to use it here.

I followed these steps to build it:

1. Install Node.js and npm (the Node.js package manager) in your WSL environment, if they are not already installed. You can do this by running the following commands:

Update the package manager

sudo apt update

Install Node.js and npm

sudo apt install nodejs npm

1. Install the TypeScript compiler globally using npm:

sudo npm install -g typescript

1. Navigate to the directory containing your TypeScript file using the cd command. For example:

cd path/to/directory

1. Compile the TypeScript file to a JavaScript file using the tsc command. For example, if your TypeScript file is named index.ts, you can compile it to a JavaScript file named script.js using the following command:

tsc index.ts

This also validates the javascript file for errors!

# The steps and testing of this program:

I used <https://gist.github.com/rsalzer/598e9c59e3911fe02871e2be515e589b#file-paintings-js> as a reference, and the documentation available here <https://metmuseum.github.io/> to understand the Objects available.

The first step was to generate a random image from the [Metropolitan Museum of Art Collection](https://www.metmuseum.org/art/the-collection), in this case the European Classical Collection, which is [department #11.](https://www.metmuseum.org/art/collection/search?showOnly=withImage&department=11)

async function fetchRandomImage() {

// *Make a request to the Met Museum API to search for paintings that have images, with the department ID set to 11 (paintings).*

const url = '<https://collectionapi.metmuseum.org/public/collection/v1/search?hasImages=true&medium=Paintings&departmentId=11&q=Painting>';

const response = await fetch(url);

const res = await response.json();

// *Generate a random number between 0 and the total number of results minus 1.*

const max = res.total - 1;

const min = 0;

const random = Math.floor(Math.random() \* (max - min + 1)) + min;

// *Use the random number to select an object ID from the list of object IDs in the response.*

const picked = res.objectIDs[random];

// *Make another request to the API using the selected object ID to retrieve more information about the object.*

const url2 = `https://collectionapi.metmuseum.org/public/collection/v1/objects/${picked}`;

const response2 = await fetch(url2);

const res2 = await response2.json();

// *Retrieve the image URL from the response.*

const imgUrl = res2.primaryImageSmall;

// *Fetch the image and create an object URL for it.*

const imageResponse = await fetch(imgUrl);

const imageBlob = await imageResponse.blob();

const imageUrl = URL.createObjectURL(imageBlob);

// *Create an image element and set its src to the object URL.*

const imgElement = document.createElement('img');

imgElement.src = imageUrl;

// *Add the image element to the document.*

document.body.appendChild(imgElement);

The Museum requests that usages of the API link back and credit their website

So I added a link back to the Metropolitan Museum of Art URL.

// *Create a link to the object's page on the Met Museum website.*

const anchorElement = document.createElement('a');

anchorElement.href = res2.objectURL;

anchorElement.textContent = 'View on Met Museum website';

// *Add the link to the document.*

document.body.appendChild(anchorElement);

Display title, artist, date, medium, dimensions and credit line of the painting

// *Create an element to display the title of the painting.*

const titleElement = document.createElement('p');

titleElement.textContent = `Title: ${res2.title}`;

document.body.appendChild(titleElement);

// *Create an element to display the artist's name.*

const artistElement = document.createElement('p');

artistElement.textContent = `Artist: ${res2.artistDisplayName}`;

document.body.appendChild(artistElement);

// *Date*

const date = res2.objectDate;

const dateElement = document.createElement('p');

dateElement.textContent = `Date: ${date}`;

document.body.appendChild(dateElement);

// *Medium*

const medium = res2.medium;

const mediumElement = document.createElement('p');

mediumElement.textContent = `Medium: ${medium}`;

document.body.appendChild(mediumElement);

// *Dimensions*

const dimensions = res2.dimensions;

const dimensionsElement = document.createElement('p');

dimensionsElement.textContent = `Dimensions: ${dimensions}`;

document.body.appendChild(dimensionsElement);

// *Credit Line*

const creditLine = res2.creditLine;

const creditLineElement = document.createElement('p');

creditLineElement.textContent = `Credit Line: ${creditLine}`;

document.body.appendChild(creditLineElement);

I wanted to create the ability to regenerate paintings continually so I added a button to call the function repeatedly..

// *Create a button to generate a new painting image.*

const button = document.createElement('button');

button.textContent = 'New Painting';

button.addEventListener('click'**,** fetchRandomImage);

document.body.appendChild(button);

I found that sometimes the images error and don’t show (usually a licensing issue of the original painting, and the agreements on reproductions etc) so I developed an error function to delete the created html info and regenerate.

imgElement.onerror = () => {

// *When the image fails to load, remove the elements from the page and call the `fetchRandomImage` function again to try again.*

document.body.removeChild(imgElement);

document.body.removeChild(anchorElement);

document.body.removeChild(titleElement);

document.body.removeChild(artistElement);

document.body.removeChild(dateElement);

document.body.removeChild(mediumElement);

document.body.removeChild(dimensionsElement);

document.body.removeChild(creditLineElement);

document.body.removeChild(button);

fetchRandomImage();

};

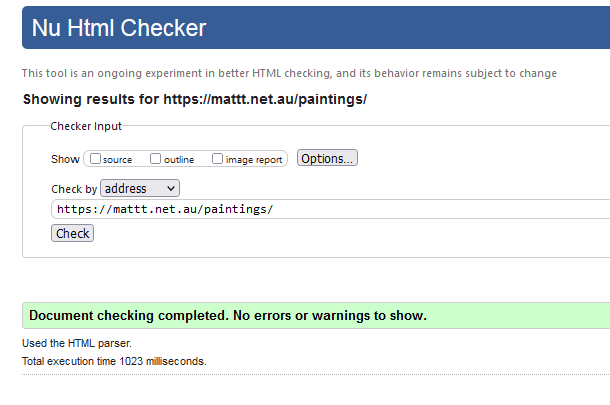
}

Lastly, call the function to generate.

// *Call the function to fetch and display a random image.*

fetchRandomImage();

I then created an html file and styled it with CSS, to then validate.



The final form looks something like this!

