CODE TEST FRONT END DEVELOPMENT (VUE.JS)

This test includes the front end development of a photo gallery that show images of paintings from the Rijksmuseum, the Dutch national museum for arts and history.

The assignment

You're about to create a photo gallery, written in Vue.js. This gallery must retrieve the photos of the paintings from the Rijksmuseum API in order to show them in the browser.

The Rijksmuseum offers their entire art and history collection through their API. This includes paintings, drawings, books, papers, coins and more. In this assignment, we focus ourselves on paintings.

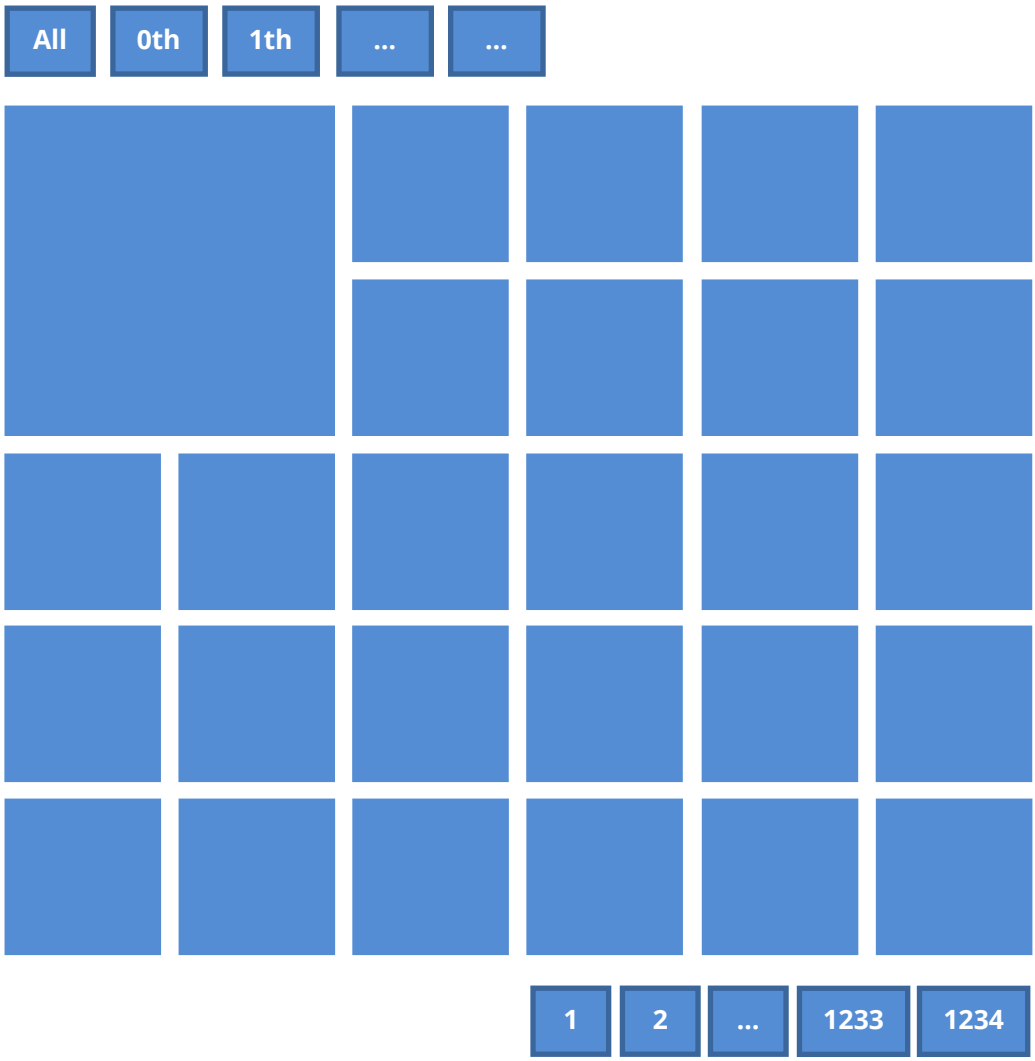
For this assignment, you'll need to connect to their API to retrieve and show the paintings returned by the API. The pictures of the paintings need to be shown in a grid of square thumbnails, where the first picture should always be twice the size of the rest of the pictures. The thumbnails should always be filled with a centered (not stretched) portion of the painting.

Above the grid, there needs to be an option to filter the pictures based on period (the century in which the painting was made). There should also be an option "All", which selects all paintings from all periods. Below the grid, you need to show pagination so that the user can go through all pictures specified by the period filter above the grid. Each page should list 22 paintings.

On desktop screen sizes, you'll get a result that looks like the "wireframe" on the next page. You may use a max width of 1200 pixels if that gives a nice result, but feel free to experiment with a full width lay-out. Make sure to scale this lay-out down in a logical way on smaller screen sizes, such as tablet or mobile phone screen sizes.

When a user clicks on of the pictures, a modal should pop up with a bigger version of the picture along with some details about the selected painting: its title, object number, description, maker(s), material(s) and technique(s). For this modal, no example/wireframe is give, so use your creativity and imagination. The purpose of this modal is to provide details about the painting.

Wireframe of the desktop grid lay-out



API instructions

In order to connect to the Rijksmuseum API, you need an API key and some API documentation.

- Here is a link to the documentation: https://data.rijksmuseum.nl/object-metadata/api/
- This is the API key you can use: `eQs2JtwU`
- We only want to list paintings that have images
- The API has no limitations, but does have a fair use policy. If there's anything wrong with the API key, please let us know asap.

Guidelines

The following guidelines apply to this code test:

- We expect a solution in Vue.js. The code you deliver should consist of one or more components, whatever you would do in a real project.
- You must set up a Vue.js project yourself. Tip: use the Vue CLI tool to do this.
- You may use existing 3rd party JavaScript packages.
- You may use a frontend toolkit such as Tailwind CSS or Bootstrap.
- SASS or LESS may be used for CSS.
- Quality and loading time of the photos is beyond the scope of this test.
- Docker may be used to speed up setting up the toolkit. We have a docker image ready to use; a `docker-compose.yml` file is included.

We look at the following parts of the test, in this order of importance:

1. Overall impression of the Vue.js code and the choices made within the code;
2. Functioning and appearance of the complete gallery (including the modal on desktop sizes;
3. Functioning of the periods filter and pagination;
4. Layout on desktop screen sizes;
5. Responsiveness and layout on mobile devices (04.png);
   - Make sure the gallery uses the full screen width for screen widths below 768px.
6. Cross browser functioning;
   - Make sure the gallery works well on at least Chrome (latest), Firefox (latest), and Chrome on mobile.

When assessing this test, we mainly look at the following aspects, not necessarily in this order:

- Functioning of the end product from the end user's point of view;
- Development speed;
- Quality of code and (potential) bugs in the code;
- Readability and elegancy of the code.

Finally, you must register what time you started the test and what time you have completed it.