Rick & Morty Api

Bernat Ferrer

1 Requirements

This project consists of a website that enables the user to search information about the series ***Rick and Morty*** . The information is obtained by making queries to a **third party Api** and using **Axios**, a wrapper library for Ajax. The project is developed using **jQuery** and it is **responsive** to changes in height or width. Moreover, a **postman collection** with some query examples is included in the project folder .

The way the user can navigate through the website is by selecting a given chapter within the navbar located on one side, which will cause the website to query the characters appearing in the episode and displaying their **image, name, status and species** on the main container **.** It will also display the **episode name** and the **episode number**.

When clicking on top of a character image, more information will be added, namely its **gender, origin location** and **list of episodes** where it appears. If the user wants more information, he/she can click on top of the location label, and the website will query more details regarding the character origin location. It will display the **type of place, name, dimension and residents list.**

The way the website is structured is the following. It is divided into three parts. A **header,** where the title of the series is displayed and also the current title section the user finds himself at. Then a **sidebar** located in the left part, which enables the user to load more pages of episodes and can also be hidden for a better visualization of the data being displayed in the main container. Finally, a **main container** where the information is being displayed.

2 Api documentation

General info

The rick and morty api is a REST api with the following base url :
https://rickandmortyapi.com/api/

This url has an object with the three available data resources in form of links for the client to make further queries to.

```
{
  "characters": "https://rickandmortyapi.com/api/character",
  "locations": "https://rickandmortyapi.com/api/location",
  "episodes": "https://rickandmortyapi.com/api/episode"
}
```

- “Characters” returns an array of all characters in the series.
- “Locations” returns an array of all the locations
- “Episodes” returns an array of all the episodes

The info that the api gives to the client is always paginated, with up to 20 documents per page. Also, each resource contains an **info** object with information about the response. In the table below we can find some of the keys we can add to our query to make it more complete.

Key	Type	Description
count	int	The length of the response
pages	int	The amount of pages
next	string (url)	Link to the next page (if it exists)
prev	string (url)	Link to the previous page (if it exists)

Example

As a sample request to the following url, we obtain the object below.

https://rickandmortyapi.com/api/characters

```
{
  "info": {
    "count": 591,
    "pages": 20,
    "next": "https://rickandmortyapi.com/api/character/?page=2",
    "prev": null
  },
  "results": [
    // ...
  ]
}
```

Pages

If no page is specified, the first page will be shown. In order to request the page n , the client will append “/?page=n”

Multiple characters

Moreover, we can also get multiple characters by adding an array of ids as a parameter :   `/character/[1,2,3]` or `/character/1,2,3`

Filter characters

We can also filter our query by including additional parameters, for instance, by adding , `name=rick&status=alive,` we would get a list of the characters named rick who also are alive.

Available parameters:

- name: filter by the given name.
- status: filter by the given status (alive, dead or unknown).
- species: filter by the given species.
- type: filter by the given type.
- gender: filter by the given gender (female, male, genderless or unknown

Similarly, we can access a single location, multiple locations, a single episode, multiple episodes or filter with the aforementioned parameters.

For more information regarding this api, please visit

https://rickandmortyapi.com/api/documentation

Making queries with javascript

In Javascript we can use axios , a library used to make HTTP requests from **node. js** For making a query, he have to code the following:

axios.get(https://rickandmortyapi.com/api/characters/1)

In order to use the data obtained we have to either set this line of code to a variable, or using by promises, for instance

axios.get(https://rickandmortyapi.com/api/characters/1).then(

```
function(data) {
var response = data.data;
displayCharacter(data.data);
}
)
```

If we want to perform multiple queries and use the data to make an action only when all of the data has been obtained, we can use axios.all(), with an array of queries inside of it, followed by the "then()" promise. Please be aware that the axios queries are performed asynchronously, and hence, lines of code outside of "then()" will be executed immediately after. Moreover, it may be convenient for the user if we show a loading icon in a form of animation while our code is performing the queries .

3 Incident record

The incidents encountered during this project are listed below.

- Sidebar that can be hidden. Some issues with element positioning when the sidebar is shown. Fixed by adding or subtracting the same amount of width to those elements as the sidebar gains or loses when growing / shrinking.

- When the main has overflow, the button for showing or hiding the sidebar disappears if we scroll down. Solved by adding a position absolute to the button.

- The elements inside the main are not evenly distributed and not being responsive. Both problems are solved by making the characters container flex , with “justify-content : space-between”, “flex-wrap:wrap “ and adding some margins.

- A scrollbar always appears even if we do not have overflow right now. Solved by changing the property “ overflow: scroll” to “overflow:auto”.

- Some code duplicity encountered. Solved by making the code more modular and using the same functions for similar tasks.

4 Lessons

Some of the lessons acquired during the development of this project are the ones that follow.

- Usage of svg files for animation purposes .
- Usage axios.all() for multiple queries .
- Code optimization by not making queries of information already used .
- More confidence when working with apis as well as using promises .

Author: Bernat Ferrer Mundó