Abstract

My idea was to select a few regions and show off some of the flavors that the regions had.
Flavors meaning, most picked, highest winrate and top champions that get Pentakills. I wanted to keep it simple as I realised that I can't possibly request the API every time I go to the page, so I would have to fetch the data, process it in a way that reduces the byte amount and store it on the server so that I can then work with it, without worrying about further API calls. One other reason for finalizing with 3 regions is that I got rate limited, but not by Riot, by my own host as I requested too many times in an N amount of time. So I didn't want to push it.
The final regions I worked with were EUNE, EUW and NA. I play in EUNE and wanted to see what differences there are between the EU neighbors and to see how different NA is compared to EU.
My data does not deduce that EU > NA or that EU < NA!

Getting started

/data folder

Contains the 3 regions (NA,EUW,EUNE) .json files for the match ID's that Riot provided and .php files that are explained below.

api.php

Setup api key.

ajax.php

Contains the functions to send a curl to the API server and return the data for formatting

$region variable for determining the API server where the request is made to.

$file variable to determine where all the data will be written after all 10 000 matches have been processed.

The php will work upon getting a GET request (from ajax.html) and then given the Match ID from the GET, it will send a curl request to the API. Once the data is received fromt he curl it will be decoded to an array and processed further.

All the data that is returned will be stored client-side as a session cookie (**$_SESSION['champs']**) in the browser. After all the data has been processed and the GET method sends 'w' as one of the headers, the PHP will write the data into the specified **$file.**

E.g. to process 5 regions, you'll need to change **$region** and **$file** variables for each region, and also some modifying in ajax.html (line 148), though, since the data fetching takes quite some time, it isn't that demanding.

ajax.html

Contains ajax functions to start getting all the needed data from the API.
ajax.html will be communicating with the **ajax.php** and **champs.php** file(line 80 and line 102, respectively). When running in the browser, one button is for the regions, the other is for champion data.

var json_region will be initialized as an array of all the match IDs from a specified region.
var json_champions is for initializing champions and their ID's (this is irrelevant if you haven't processed the game datas first).

lines **171**,**177** are the initializing functions to load in the target .json files.
loadJSON(callback) : is for loading the bare match ID data (that was provided by Riot).
loadJSON2(callback): is for loading the processed data and get all of the champions ID's.

Firstly, to start the data fetching, the **function fetch()** on line 64 will start **function apiCall()** with a specified interval in milliseconds.
The faster the interval is, the higher the failure rate (not all curls will return a json, might be my coding inexperience in this line of work) and if it's faster than 1300'ish you can or will get rate-limited by the API server.
The longer the interval is, like 3-5 seconds, the higher the success chance and no risk of getting rate-limited from the API.

apiCall() works with the **json_region** array that was initiliazed when you opened the html in a browser. Likewise **apiCall2()** is just a paste of the first one, but can only be used after you've gotten the 10k games processed and want to get the champions names/titles/keys and save them locally (the 'key' gives acces to their .png icons and splashes etc, since you can use it with ddragon)

As I mentioned **failure**, there is a failsafe within the **apiCall()** functions that listens to what the PHP file says in response.

- 'done' - the var **count** variable will be increased and the next element in the array will be used to send another request, later.
- '!=done' - If there's **any other response**, the counter will not be increased and it will be counted as a failure, the next request will be sent with the same match id, so we don't skip data.

Once all the match ID's have been requested (**count > json_region.length**) , line 118 **function print()** will trigger and send the ajax.php a request that will trigger it to write all the session cookie data into file. I had some weird problems with ending the intevals, so there are many variables that just exist to ensure that the intervals are stopped and that printing is only done once.

champs.php
Used to verify a champions name, title, key according to the ID that we get after having processed a regions games.
It works basically in the same way as does ajax.php. The data will be stored as a session cookie variable **($_SESSION['champdata']**) and the data in the cookie will be printed into a file.

Working with the data

data.php
Contains functions that work with the already processed champion info. data.php is required for index.php and champions.php to work.

function champions($region) - Reads all the stats for a champion and returns them.
function champItems($region, $champId) - Based on inputs, returns items that the champion bought and their stats. There are 2 versions of this, the second one is used for the champion-specific page.

index.php

Building the regional overviews. The basic of it is a for-loop that builds all the necessary divs/tables and javascript before the page is even loaded.

> $script - contains the data structure that is to be built for the bar charts and doughnuts to work. Chart.js was used for the bar & doughnut charts.
> $onload - is another needed aspect to render the Chart.js charts
> $table1 and $table2 are the containers of the tables that are alongside the bar charts.
> $body is the HTML structure that is to be echo'd into the HTML.

The identifications for each bar/doughnut chart is simple, using the for cycles index to be written after the name.

In a nutshell, the data is built, reading from the $regions array and then calling the champions() function of a specific region. To sort the data as "most played" or "highest winrate", usort is used and then foreach cycles through the sorted data and find the appropriate champions by their id.
There are many variables like $c0, $cc0 etc that are used to cached the data and use it meaningfully after a foreach loop. For and foreach might be easy to use but it'snot really good to go 1 layer deeper and foreach(foreach(foreach)) yourself into a state where you timeout on your response : D, that's also a reason for so many "chache" variables.

Also, if one has more than 3 regions datasets available and wants to use them, then the only needed changes to do are:

> **index.php line 5** - add more regions
> **data.php** - add more file urls **lines 2-4**, and **lines 14, 52, 91**, update the logic accordingly.

champions.php

Displays the champion roster. If an id is specified it will show the stats of that champion for each region and top items as well.

The html is built by the PHP and then echo'd, much like in the index page. data.php is required as well to render the items and stats of each champion.