Derivative statistics

Background

Application is presenting page view statistics from a time period on a chart. The chart always displays one month in a form of one or more series (each one containing data from separate website).

Series contains one entry for each day of a month in following format:

```
[
      {“date”: new Date(2020, 4, 1), “visits”: 32},
      {“date”: new Date(2020, 4, 2), “visits”: 75},
      {“date”: new Date(2020, 4, 3), “visits”: 96},
      …
      {“date”: new Date(2020, 4, 31), “visits”: 73},
]
```

Series may be missing entries for some days of a month (if website tracking was set up or if user disabled it during that month).

Task

We want to display the average number of visits per day and per each weekday in a month based on available data.

Requirements

1. Write implementation of a “getAverage” function that accepts a series and returns average daily visits.

   ```
   > getAverage(series1);
   {
         “averageVisits”: 55,
   }
   ```

2. The `function should` also accept optional parameter and calculate average number of visits per each week day when set to true.

   ```
   > getAverage(series1, true);
   {
         “Monday”: {
               “averageVisits”: 32,
   ```

```
        },
        “Tuesday”: {
              “averageVisits”: 82,
        },
        “Wednesday”: {
              “averageVisits”: 74,
        },
        “Thursday”: {
              “averageVisits”: 35,
        },
        “Friday”: {
              “averageVisits”: 54,
        },
        “Saturday”: {
              “averageVisits”: 64,
        },
        “Sunday”: {
              “averageVisits”: 44,
        }
  }
```

3. The function should also let user calculate averages from “total series” (series containing data entries from all website series displayed on the chart). Such series will be created by concatenating all entries from multiple series into a single array. This means that total series may contain multiple entries stating the same date, but different visits values. Order of the entries in total series isn’t defined.

   ```
   let totalSeries = series1.concat(series2);
   ```

4. Code should be able to run on nodejs (v8.0+). You can check node EcmaScript compatibility here.
5. The function should be implemented in provided file structure (download) and pass basic tests (more about them in “Tests” section).

Tests

We have prepared basic setup for the package containing a placeholder function declaration awaiting you implementation and rudimentary unit tests. Feel free to expand those tests during development. Task assessment will include use of more extensive battery of tests to determine if your code has any errors.

How to setup environment

1. Install nodejs
2. Copy project files (from zip file) on your disk
3. Go to the package directory (contains “package.json”)
4. Run “npm install”

How to run tests

1. Go to the package directory (contains “package.json”)
2. Run “npm test”

Scoring criteria

List of criteria (most important at the top):

1. Code readability
2. Elegance of solution (it should be concise)
3. Test coverage (not necessary but encouraged)
4. Solution provided as git repository (not necessary but encouraged). Examples: bitbucket or github.