High Charts Demo (Pokémon Dataset)

Understanding the dataset

Format: JSON

Source: Data is stored in separate .JSON file, inside angular project, no separated server/API is used.

Structure:


```
[
  {
    "#": 1,
    "Name": "Bulbasaur",
    "Type 1": "Grass",
    "Type 2": "Poison",
    "Total": 318,
    "HP": 45,
    "Attack": 49,
    "Defense": 49,
    "Sp. Atk": 65,
    "Sp. Def": 65,
    "Speed": 45,
    "Generation": 1,
    "Legendary": "False"
  }
]
```


Pokémon is a game where player creates a team of 6 Pokémon's, main each Pokémon has type, 6 base stats, evolution line and abilities and player can hand them special items to improve the stats of a Pokémon, well the game is still more complex than this, but we are only considering the most important property of a Pokémon that is its type & these 6 base stats (Hit Points [HP], Attack [ATK], Defense [Def], Special Attack [SP. ATK], Special Defense [SP. DEF], Speed [SPD]).

Goal:

To visualize the data of all Pokémon in general to find their weak spots and strong points, compared to others in general and versus 1 specific Pokémon.

Project Outline:

Build on Angular 14, Bootstrap 5, HighCharts. Uses a central ChartService to distribute the data across the components. tsconfig.json was modified to accept imports of json files.

1. We use the Pie chart to show distribution of Pokémon having a specific type.
2. We use a box-plot to compare the general stats of different types of Pokémon.
3. We can compare the Pokémon directly and check the difference in stats using the bar chart.

Project Structure:

- src/
  - app/
    - bar-chart/
      - bar-chart.component.html
      - bar-chart.component.ts
    - box-chart/
      - box-chart.component.html
      - box-chart.component.ts
    - chart-service/
      - chart.service.ts
    - common/
      - filter-pipe/
        - filter.pipe.ts
      - select-dropdown/
        - select-dropdown.component.html
        - select-dropdown.component.ts
      - common.type.ts
    - pie-chart/
      - pie-chart.component.html
      - pie-chart.component.html
    - app.component.html
    - app.component.ts
    - app.module.ts
  - assets/
    - POKEMON_DATA.json
  - index.html
  - main.ts

main.ts

driver file, nothing changed from the one generated by angular-cli.

index.html

template modified to add bootstrap css & js.

app/app.module.ts

added imports FormsModule to add support for directive NgModel & HighchartsChartModule to add support for HighCharts chart-rendering components.

app/app.component.html,

app/app.component.ts

uses bootstrap grid to build a basic layout & footer.

maintains 3 variable, selectedPokemon1 & selectedPokemon2 for comparing in bar-chart, and pokemonOptions an Array that contains list of all the Pokémon to feed in the dropdown input that is used to select which 2 Pokémon are to be compared.

Communicates with ChartService to inform it about the selected Pokémon's to compare.

app/common/fitler-pipe/filter.pipe.ts

name: 'filter'

transforms: Option []

parameter: search: string

output: Option []

custom dropdown component <app-select-dropdown> used in the project, requires to filter list of options according to search input, this pipe is used to filter an Option []. Accepts a string of searched input as a pipe parameter.

app/common/select-dropdown/select-dropdown.component.html,

app/common/select-dropdown/select-dropdown.component.ts

Selector: 'app-select-dropdown'

Input:

placeholder: string

options: Option []

Output:

select: Event<any>

uses 2 input types select & input[type=”text”] and bootstrap classes to construct a component that will open the select on focusing & filter options of select when input of textbox is changed.

Accepts placeholder to display inside textbox

Accepts an option array for select (dropdown & to be filtered by search)

Emits a select event that is emitted when an option is selected, emitted value is Option.value of that selected option.

Note: There is more to the component’s select > options storing JSON string and not atomic value, check the select-dropdown.component.ts for detail.

app/common/common.type.ts

stores custom type object which are commonly used in the project.

List of custom types:

1. Option

app/chart-service/chart.service.ts

Provided in Root.

All the data is supplied from here, imported from JSON file.

Data members:

identifier	type	description
typeColors	Object	an object denoting the color assigned to specific Pokémon type (key-value pair).
selectedPokemon1	BehaviorSubject <number>	Contains the ID of selected Pokémon to compare in Bar Chart
selectedPokemon2	BehaviorSubject <number>	Contains the ID of selected Pokémon to compare in Bar Chart
selectedTypes	BehaviorSubject <Set<string>>	Maintains the selected Pokémon types of compare in Box-Plot & Pie-Chart

Member Functions:

1. changeSelectedPokemon (p1: number, p2: number)
    a. Changes the data members *selectedPokemon1* & *selectedPokemon2* according to passed parameters.
    b. Returns null

2. getPokemonById (index: number)
    a. Finds & returns the Pokémon object from Pokémon array by its ID.

3. getAllPokemons ()
    a. Return all the Pokémon from the array.

4. getTypeCompareBoxChart ()
    a. fetch all Pokémon
    b. initialize a container to store all 12 types of Pokémon in a bucket.
    c. parse all Pokémon
        i. check If Pokémon is of the *selectedType* If not ignore, Else proceed.
        ii. Create the type bucket if not exists, there maintain object with 6 arrays as properties [for HP, ATK, DEF, SP. ATK, SP. DEF, SPD] and push the stats of the Pokémon into respective array.

d. Initialize seriesData
e. For each *selectedType* find the following from container
    i. Min, Lower Quartile, Median, Upper Quartile, Max
    ii. Do this for each stat [HP, ATK, DEF, SP. ATK, SP. DEF, SPD]
    iii. Add them to seriesData
f. Fit this into *Highcharts.Options* Object & return it.

5. getTypeChart()
    a. get *selectedTypes*.
    b. Get all pokemons
    c. Count the pokemon's population based on the type.
        i. If the pokemon's type is in *selectedTypes*, set property "sliced" - true, (making that slice extruded from center when rendered).
        ii. Modify the default click event to enable multiple selected slices according to *selectedType* set.
    d. Fit that into *Highchart.Options* Object & return it.

6. getComparePokemonData()
    a. get *selectedPokemon1* data
    b. get *selectedPokemon2* data
    c. fit the data into the *Highchart.Options* Object & return it.

app/bar-chart/bar-chart.component.html,

app/bar-chart/bar-chart.component.ts

Wraps the HighCharts libraries component to a controlled bar-chart that is used to compare 2 pokemon's 6 base stats.

Uses ChartService and subscribes to BehaviorSubjects *selectedPokemon1* & *selectedPokemon2* and updates the chart on changes.

app/box-chart/box-chart.component.html,

app/box-chart/box-chart.component.ts

Wraps the HighCharts library 'highcharts-chart' component to a controlled box-plot that provides detailed comparison of general base stat distribution of a type of Pokémon.

Updates when different types are selected from the Pie Chart

Uses ChartService and subscribe to BehaviorSubjects *selectedTypes* and updates when a new type is added or existing type is removed from the set.

app/pie-chart/pie-chart.component.html,

app/pie-chart/pie-chart.component.ts

Wraps the HighCharts library 'highcharts-chart' component to a controlled pie-chart that provides population distribution graphic of type of Pokémon.

Uses ChartService to fetch the population data once.