## BIBLIOTEKA CHART.JS

## WPROWADZENIE

Chart.js to biblioteka typu *open-source* na licencji *The MIT License* napisana w języku JavaScript. Służy do wizualizacji danych w postaci różnego rodzaju wykresów.

## SPOSÓB PODŁĄCZENIA BIBLIOTEKI

Istnieją dwa sposoby podłączenia biblioteki Chart.js do skryptu:

* pobranie biblioteki
  + poprzez Node Package Manager (npm)

Wpisujemy komendę npm install chart.js w cmd i w ten sposób biblioteka zostaje pobrana na nasze urządzenie.

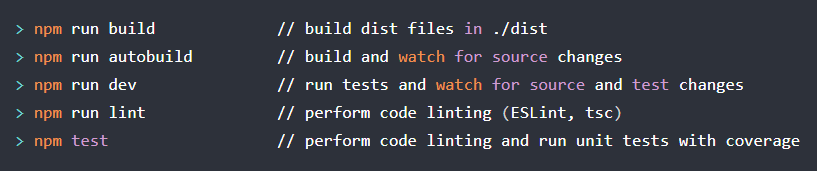


W takim przypadku aby wykorzystać pobraną bibliotekę w skrypcie, należy w sekcji <head> umieścić poniższy wpis:

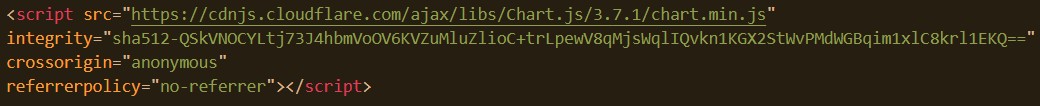


* + z serwisu GitHub

Pobieramy pliki biblioteki na urządzenie bezpośrednio ze strony https://github.com/chartjs/Chart.js, po czym budujemy Chart.js, żeby wygenerować tzw. dist files (distributable files), czyli pliki, które zawierają kod w formie zminifikowanej.

Jest to możliwe przykładowo przy pomocy NPM:

Wysoce zalecane jest jednak skorzystanie z opcji alternatywnych.

* korzystanie z biblioteki za pośrednictwem sieci dostarczania treści
  + CDNJS
  + jsDelivr

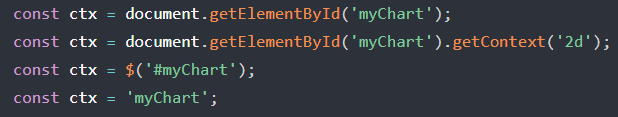


## ZASTOSOWANIE

Po podłączeniu biblioteki Chart.js do skryptu niezbędne jest stworzenie canvasu, czyli obszaru, w którym będzie rysowany wykres.

# Przy tworzeniu canvasu musimy nadać mu atrybut, który pozwoli nam się do niego odwołać, np. id. Możemy też zdefiniować jego wysokość oraz szerokość, jeżeli wykres nie będzie responsywny.

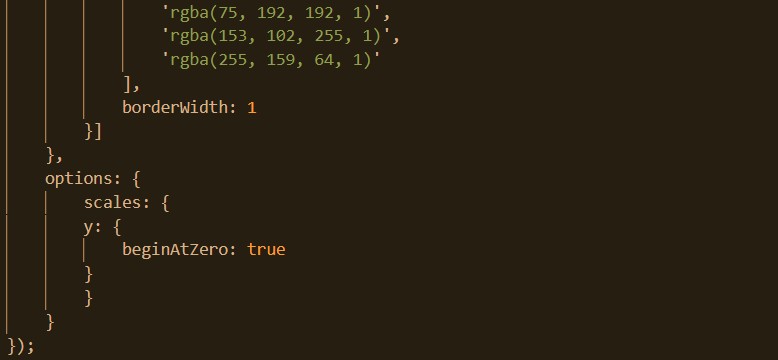
W następnej kolejności potrzebujemy stworzyć zmienną bądź stałą, która będzie się odwoływać do stworzonego canvasu. Dokumentacja wskazuje cztery sposoby, na które można to zrobić – każdy z nich jest poprawny:



# Słowo const do deklaracji stałych może zostać zastąpione słowem var bądź let, stosowanym do deklaracji zmiennych. Sposób trzeci wymaga użycia biblioteki jQuery, a w kodzie użyty został sposób czwarty.

Na koniec musimy stworzyć obiekt Chart, w którym najpierw odwołujemy się do wcześniej utworzonej zmiennej lub stałej, a później podajemy właściwości naszego wykresu: określamy jego typ, etykiety i/bądź skalę osi oraz jego kolorystykę.



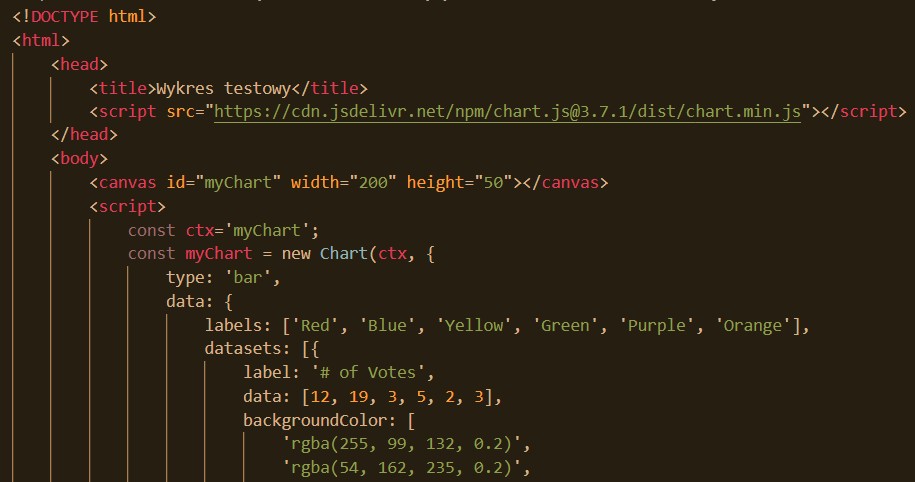


# Zamieszczony kod jest kodem testowym, podanym w dokumentacji.

Jak można zauważyć, Chart.js wprowadza własne właściwości pozwalające nam określić charakterystykę wykresu, m.in.:

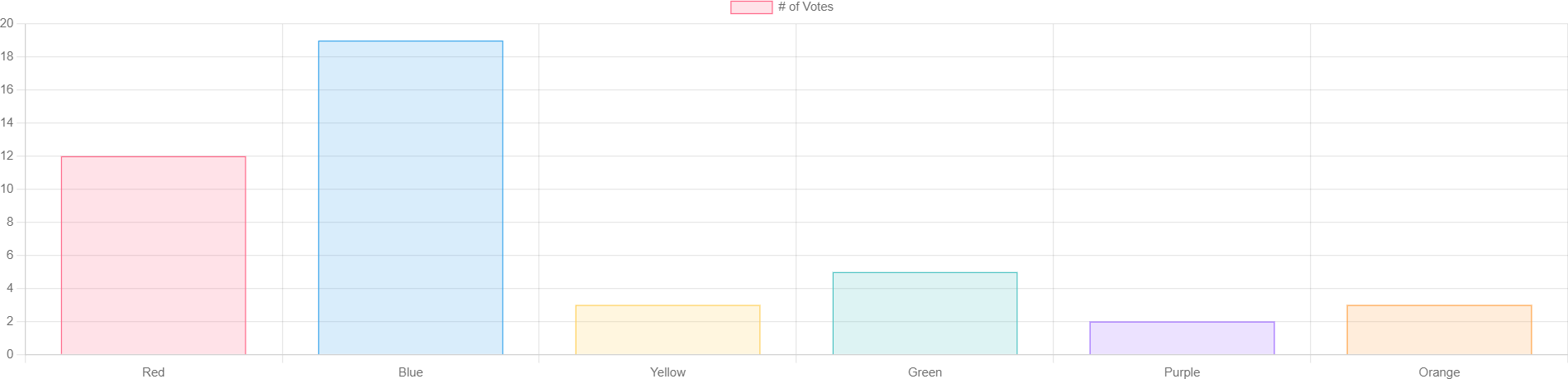
* *type* – pozwala nam określić typ wykresu.
* *data* – stanowi pojemnik dla danych dotyczących wykresu, również służy do określania skali osi bądź przyjmowanych przez nie wartości.
* *labels* – stanowi pojemnik dla etykiet osi.
* *responsive* – decyduje o tym, czy wykres jest responsywny, czy nie.
* *beginAtZero* – sprawia, że bez względu na przedział liczbowy przyjmowanych wartości skala danej osi zaczyna się od zera.

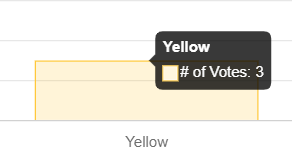
Tak prezentuje się całość kodu:





A tak prezentuje się efekt:

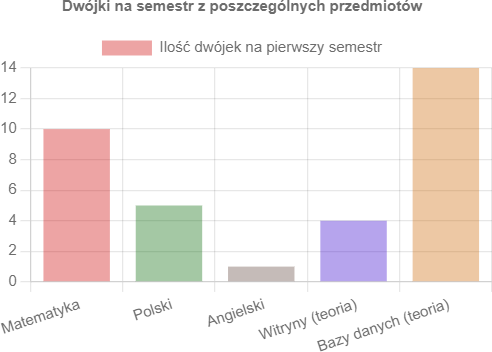


Jak widzimy na obrazku obok, po najechaniu na słupek wyświetla się nam przyjmowana przez niego wartość wraz z treścią legendy.

Korzystając z Chart.js, do wyboru mamy 8 możliwych typów wykresów:

* wykres kolumnowy (słupkowy) Kod:

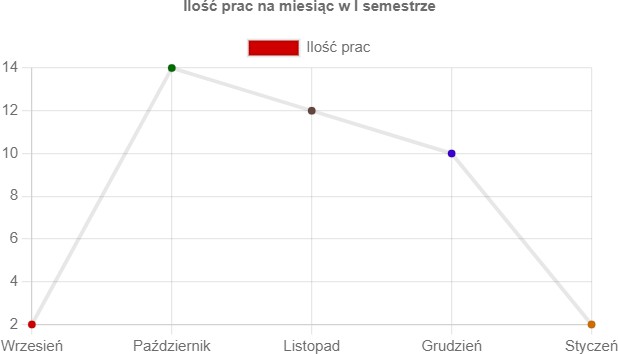
Efekt:



* wykres liniowy Kod:



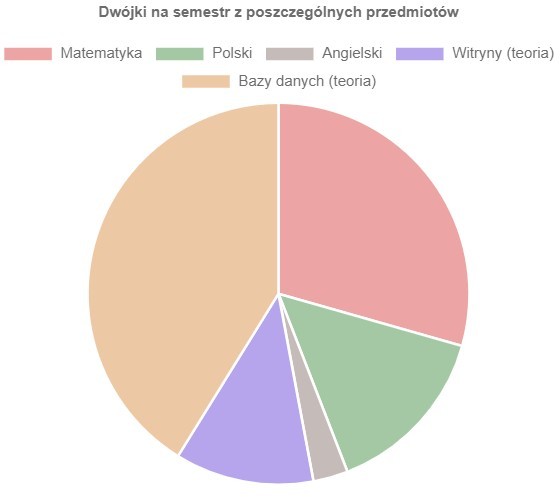
Efekt:



* wykres kołowy Kod (rodzaj 1):



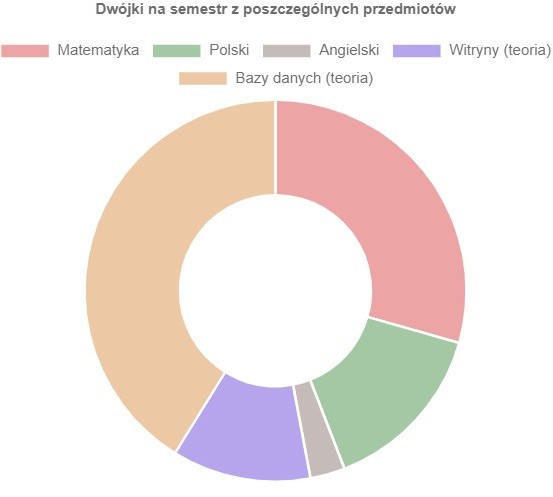
Efekt:



Kod (rodzaj 2):

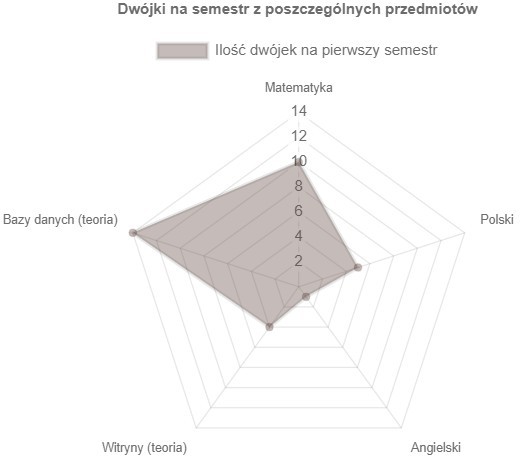


Efekt:



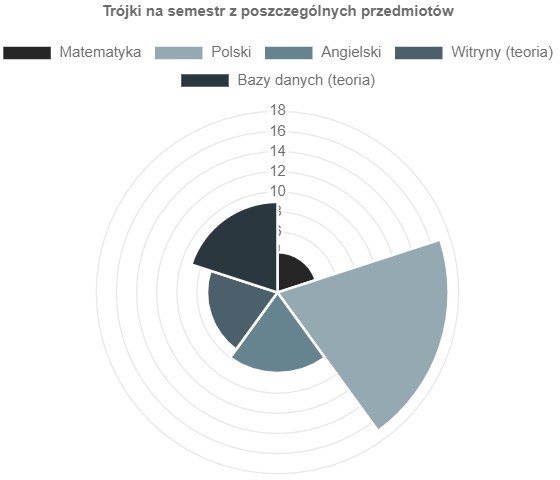
* wykres radarowy Kod:

Efekt:



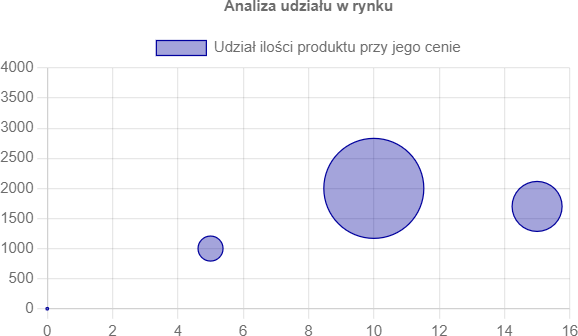
* wykres biegunowy Kod:

Efekt:



* wykres bąbelkowy Kod:

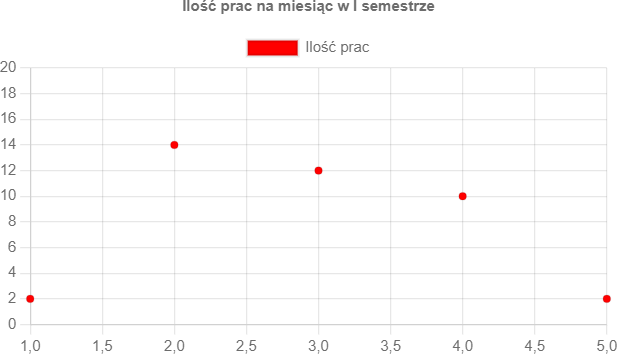
Efekt:



* wykres rozrzutu (punktowy) Kod:



Efekt:



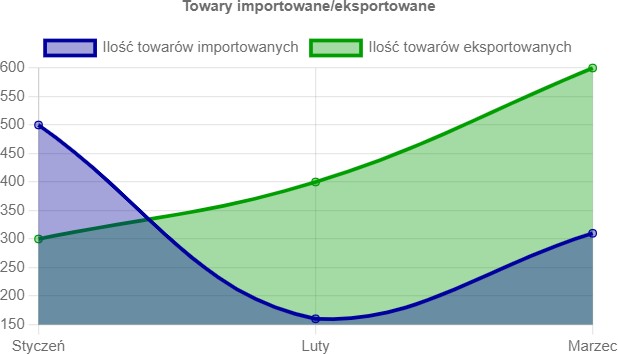
* wykres warstwowy

W przeciwieństwie do reszty przedstawionych wykresów, dla tego wykresu nie ma zdefiniowanego typu. Istnieje jednak możliwość stworzenia takowego wykresu przy użyciu typu liniowego oraz zastosowania właściwości *fill*.

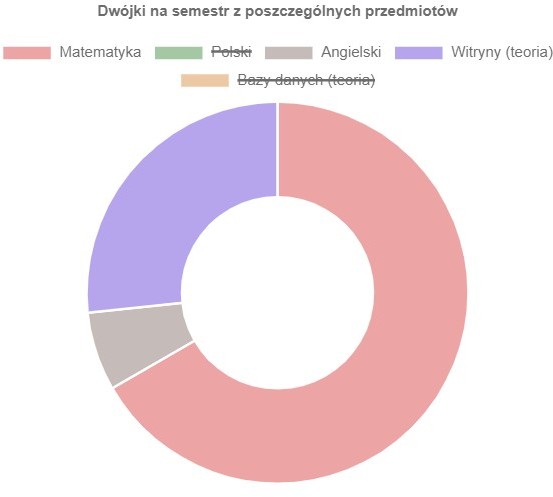
Kod:



Efekt:



Warto wspomnieć, że w przypadku każdego z powyższych wykresów możemy kontrolować, które dane z wykresu zostaną nam wyświetlone. Gdy klikniemy na nazwę elementu legendy, zostanie ona przekreślona, a wykres odnowi się, pokazując dane bez danych dotyczących właśnie tego elementu.

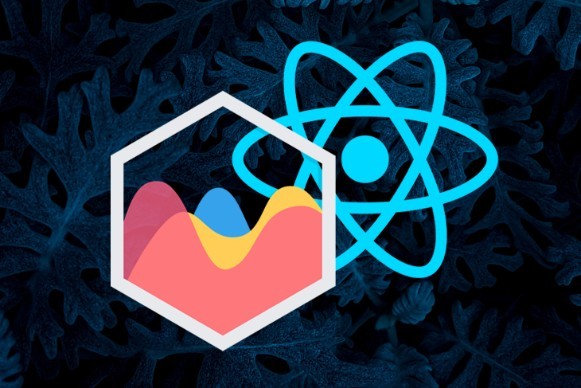


Po kliknięciu nazwy elementu po raz kolejny element zostanie na nowo uwzględniony w wykresie i jego nazwa przestanie być przekreślona.

Najechanie na punkty w wykresach punktowym, liniowym, radarowym i bąbelkowym oraz na przedziały w wykresach kołowym i biegunowym zadziała także tak samo, jak w przykładzie testowym w przypadku najechania na słupek wykresu kolumnowego.

## MOŻLIWOŚĆ WSPÓŁPRACY Z INNYMI BIBLIOTEKAMI

* React.js

Za pomocą wrappera React-chartjs-2, który pozwala nam użyć elementów biblioteki Chart.js jako komponentów React możliwa jest współpraca Chart.js z React.js. Wykorzystanie React-chartjs-2 powoduje, że tworzenie wykresów w React staje się prostsze, aczkolwiek wrapper ten nie zapewnia wszystkich funkcjonalności, jakie oferuje nam sam Chart.js.

Aby zaprezentować efekt współpracy obu tych bibliotek, zaczynamy od utworzenia folderu ReactProjects, w którym zostanie stworzona nasza aplikacja. Przechodzimy do niego poprzez komendę cd i podanie ścieżki dostępu, czyli np.:

* + cd C:\ReactProjects

Następnie tworzymy aplikację poprzez Create React App, wykorzystując npm i wpisując w cmd poniższe polecenie:

* + npx create-react-app wspolpraca\_z\_react

Wraz z utworzeniem aplikacji automatycznie zostaje pobrana biblioteka React.js i związane z nią ReactDOM i ReactScripts.

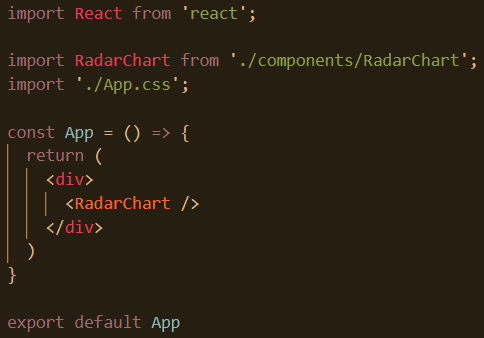
Kolejnym krokiem jest dodanie do naszej aplikacji biblioteki Chart.js i wrappera react-chartjs-2.

Użyjemy do tego yarn, przechodząc wcześniej do folderu aplikacji:

* + cd wspolpraca\_z\_react
  + [yarn add react-chartjs-2@2.9.0](mailto:yarnaddreact-chartjs-2@2.9.0)
  + [yarn add chart.js@2.7.3](mailto:yarnaddchart.js@2.7.3)

Po tych działaniach zajmujemy się edycją plików składających się na aplikację.

1. index.js
2. App.js

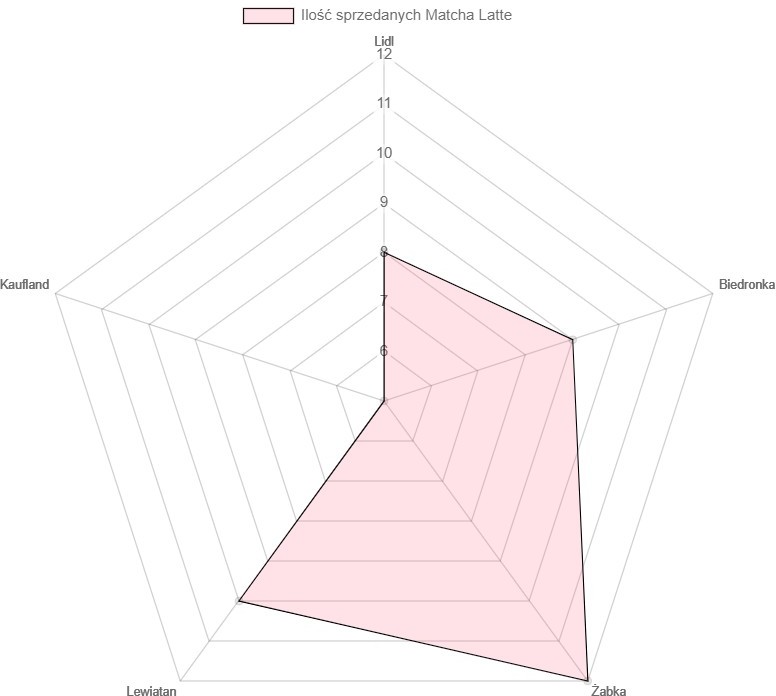


Wtedy tworzymy plik RadarChart.js, wspomniany w App.js, który będzie zawierał informacje dotyczące samego wykresu.

Na koniec uruchamiamy naszą aplikację poleceniem:

* + npm start

W ten sposób otrzymujemy następujący efekt:



* jQuery

Współpraca bibliotek Chart.js i jQuery polega głównie na wykorzystaniu jQuery przy inicjalizacji zmiennej odpowiedzialnej za odwołanie do obszaru wykresu.

W samym wykresie nie zmienia to nic – zmienia się jedynie sposób, w jaki wskażemy obszar, w którym ma być rysowany wykres.

## ŹRÓDŁA

* <https://www.chartjs.org/docs/latest/>
* [https://stackoverflow.com/questions/22842691/what-is-the-meaning-of-the-dist-](https://stackoverflow.com/questions/22842691/what-is-the-meaning-of-the-dist-directory-in-open-source-projects) [directory-in-open-source-projects](https://stackoverflow.com/questions/22842691/what-is-the-meaning-of-the-dist-directory-in-open-source-projects)
* <https://www.educative.io/edpresso/how-to-use-chartjs-to-create-charts-in-react>
* <https://www.npmjs.com/package/react-chartjs-2/v/2.9.0>
* <https://www.youtube.com/watch?v=c_9c5zkfQ3Y>