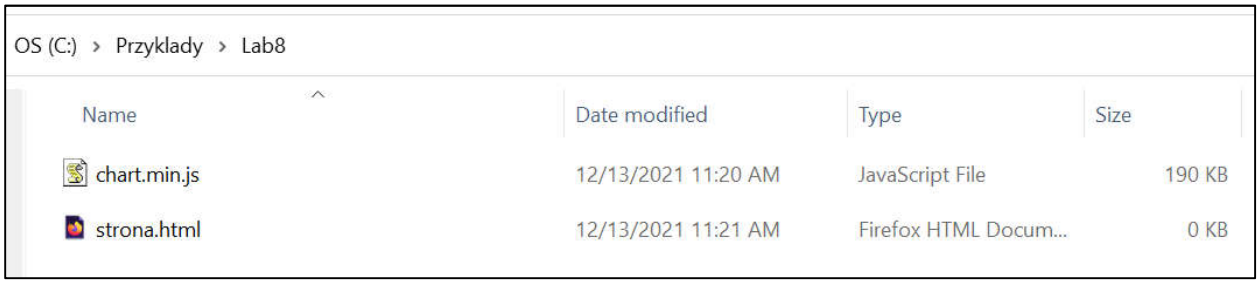
Tworzenie Aplikacji Internetowych
Laboratorium 9

Wykorzystanie bibliotek JavaScript – wykresy na przykładzie biblioteki Chart.js

Biblioteka Chart.js jest darmowa i jej dokumentację można znaleźć pod adresem: https://www.chartjs.org/. Najprostszy sposób wykorzystania biblioteki to pobranie pliku **chart.min.js** ze strony https://www.jsdelivr.com/package/npm/chart.js i umieszczenie go bezpośrednio w folderze tworzonej strony internetowej (aktualny link: https://cdn.jsdelivr.net/npm/chart.js@3.6.2/dist/chart.min.js):



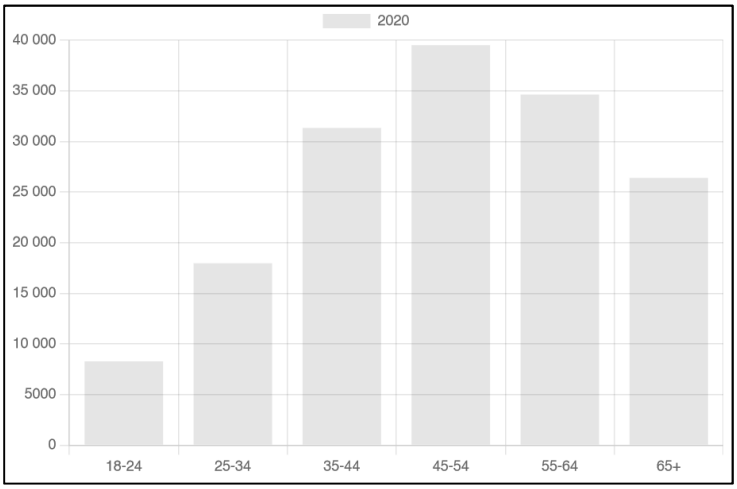
Aby wykorzystać bibliotekę należy w sekcji **head** załączyć bibliotekę **chart.min.js**, w **body** dodać znacznik **div** (o rozmiarze określonym za pomocą **CSS**) **wewnątrz** którego umieścić **canvas** w którym będzie umieszczony wykres (znacznikowi **canvas** należy nadać wybrane przez siebie **id** oraz wielkość i szerokość). Następnie w skrypcie JS należy wywołać konstruktor new Chart, zawierający jako pierwszy parametr referencję do kontektu canvasu, a jako drugi parametr dane, które powinny zostać wyświetlone na wykresie.

W danych należy podać typ wykresu, oraz dane źródłowe. Dane źródłowe są złożone z etykiet (labels) dla każdej kolumny danych, oraz datasetów zawierających wiersze danych. W każdym wierszu powinno być tyle danych ile jest głównych etykiet kolumn, oraz opcjonalnie etykieta danego wiersza (label). Podstawowa struktura wygląda zatem następująco (trzy kolumny, jeden wiersz danych o tytule "etykieta"):

Przykład (średnie zadłużenie Polaków w roku 2020 – źródło danych https://media.big.pl/publikacje/650730/infodlug-ogolnopolski-raport-o-zaleglym-zadluzeniu-i-niesolidnych-dluznikach-marzec-2021-41-edycja):

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <script src="chart.min.js"></script>
    </head>
    <body>
        <div style="width: 600px; height: 400px;">
            <canvas id="wykres" width="600" height="400"></canvas>
        </div>
        <script>
            const dane = {
                labels: [ "18-24", "25-34", "35-44", "45-54", "55-64", "65+" ],
                datasets: [
                    { label: "2020",
                      data: [ 8280, 17965, 31335, 39511, 34626, 26399 ] }
                ]
            }
            const ctx = document.getElementById("wykres").getContext("2d");
            const wykres = new Chart(ctx, { type: "bar", data: dane });
        </script>
    </body>
</html>
```

Wynik działania powyższego kodu:



Na wykresie widać sześć etykiet zdefiniowanych dla wszystkich danych (18-24, 25-34, 35-44, 45-54, 55-64, 65+) oraz jeden wiersz danych (2020) zawierających sześć danych: 8280, 17965, 31335, 39511, 34626, 26399.

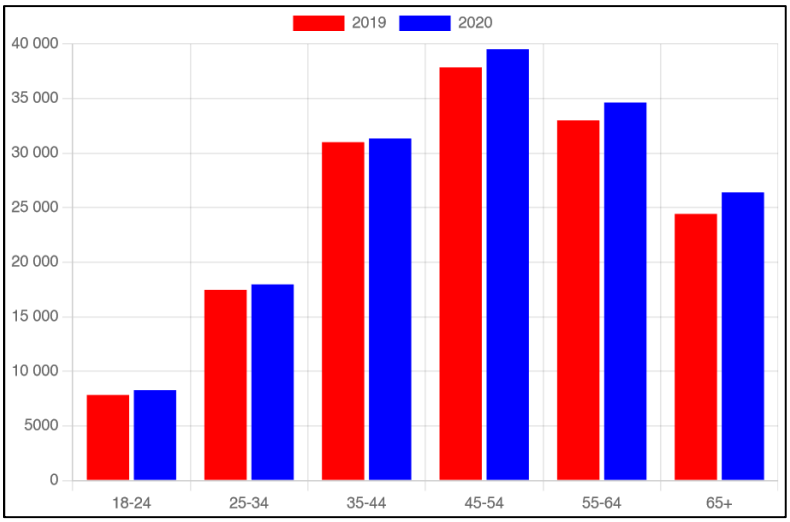
Aby nadać identyczny kolor całemu wierszowi danych należy w wierszu dodać parametr:
`backgroundColor: "rgb(50, 100, 200)"`
(rgba(r, g, b, a) nada kolor z przeźroczystością, gdzie a oznacza kanał alpha i jest w zakresie 0 do 1)

Aby nadać kolor każdej danej liczbowej osobno należy w tym parametrze przypisać tablicę kolorów (ma to znaczenie np. przy wykresie kołowym):
`backgroundColor: [ "red", "green", "blue", "orange", "purple", "brown" ]`

Wszystkie parametry należy oddzielać znakiem przecinka, dotyczy to także wstawiania wierszy danych:

```
                datasets: [
                    {   label: "2019",
                        data: [ 7835, 17465, 30997, 37846, 32995, 24425 ],
                        backgroundColor: "red"     },
                    {   label: "2020",
                        data: [ 8280, 17965, 31335, 39511, 34626, 26399 ],
                        backgroundColor: "blue"    }
                ]
```

Wynik dla powyższego przykładu:



Typy wykresów:

- Słupkowy: bar
- Liniowy: line
- Kołowy: pie
- Pączkowy: doughnut
- Radarowy: radar
- Biegunowy: polarArea
- Punktowy: scatter (zamiast danych liczbowych należy podać dane w postaci struktur { x: wartość, y: wartość }, etykiety labels nie są wymagane w przypadku takiego wykresu, punkty można połączyć linią dodając właściwość showLine: true oraz ustawiając kolor linii borderColor: kolor,
- Bąbelkowy: bubble (j.w. ale w strukturze należy podać trzecią wartość: r: wartość, oznaczającą wielkość danego punktu na wykresie)

Opcje wykresu

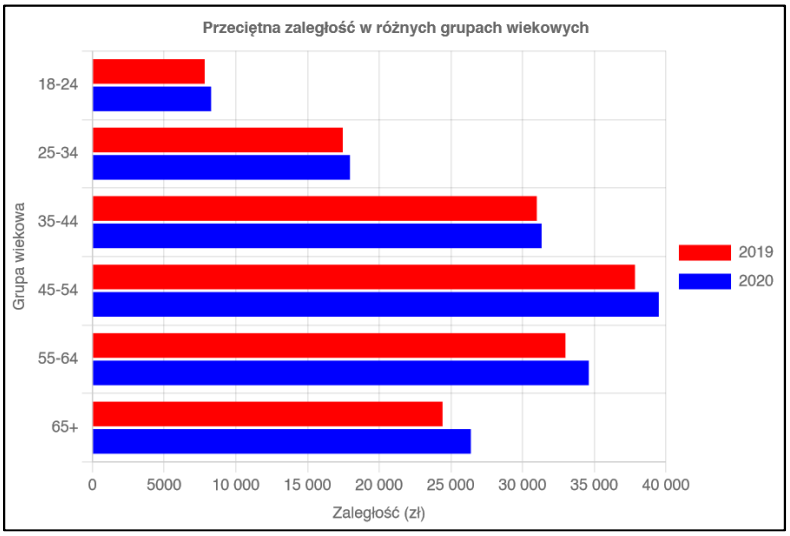
Opcje wykresu podaje się w postaci struktury jako parametr options. Przykładowe opcje:

- indexAxis: "y" – zmiana bazowej osi na poziomą (dla niektórych wykresów)
- scales { } – opcje skali
    - x { } lub y { } – wybór skali
        - min: wartość – ustawienie minimum na skali
        - max: wartość – ustawienie maksimum na skali
        - stacked: true – ustawienie nakładania się danych z wierszy
        - reverse: true – odwrócenie osi
        - type: "logarithimic" – oś w skali logarytmicznej
        - title: { } – nadanie tytułu osi
            - display: true – wyświetlenie tytułu
            - text: "nazwa" – podpis osi
- plugins { } – opcje pluginów
    - legend { } – legenda
        - display: false – wyłączenie legendy
        - position: "left"/"right"/"top"/"bottom" – ustawienie pozycji legendy
    - title { } – tytuł wykresu
        - display: true – włączenie tytułu
        - text: "tytuł" – tytuł wykresu

Przykład ustawienia opcji:

```
const opcje = {
    indexAxis: "y",
    scales: {
        y: { title: {
                display: true,
                text: "Grupa wiekowa" } },
        x: { title: {
                display: true,
                text: "Zaległość (zł)" } } },
    plugins: {
        title: { display: true,
                 text: "Przeciętna zaległość w różnych grupach wiekowych" },
        legend: { position: "right" }
    }
}
const ctx = document.getElementById("wykres").getContext("2d");
const wykres = new Chart(ctx, { type: "bar", data: dane, options: opcje });
```

Wynik dla powyższego przykładu:



Zadanie 1

- Pobrać bibliotekę Chart.js (plik: chart.min.js) i uruchomić przykład z pierwszej strony tej instrukcji.
- Nadać każdemu słupkowi na wykresie inny kolor (kolory dobrać samodzielnie).
- Spróbować ustawić kolory z przeźroczystością.
- Zmienić typ wykresu na kołowy, pączkowy, radarowy lub biegunowy (samodzielnie określić najlepszy typ wykresu dla takich danych).
- W przypadku problemów ze skalowaniem wykresu (np. kołowy może nie być dobrze rozciągnięty) spróbować ustawić wielkość **diva** i **canvasu** tak, aby wykres nie wychodził poza div (w celach pomocniczych można ustawić kolor tła znacznika div).
- Dodać tytuł wykresu "Przeciętna zaległość w różnych grupach wiekowych".
- Ustawić legendę pod wykresem.

Aktualizacja wykresu

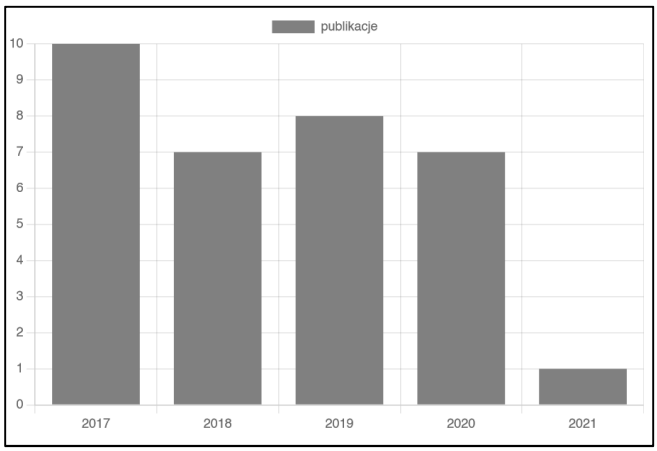
Wszystkie dane wyświetlane na wykresie można **dynamicznie modyfikować** (oprócz typu wykresu). Dane będą się wyświetlały poprawnie tylko jeżeli **liczba etykiet** będzie odpowiadała **liczbie danych** w wierszach (nie dotyczy to wykresów scatter oraz bubble). Po aktualizacji danych należy wywołać metodę update dla modyfikowanego wykresu. Pokazuje to następujący przykład:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <script src="chart.min.js"></script>
    </head>
    <body>
        <div style="width: 600px; height: 400px;">
            <canvas id="wykres" width="600" height="400"></canvas>
        </div>
        <script>
            const dane = {
                labels: [ "2017", "2018", "2019", "2020" ],
                datasets: [
                    {
                        label: "publikacje",
                        data: [ 10, 7, 8, 7 ],
                        backgroundColor: "gray"
```

```
                    }
                ]
            }
            const ctx = document.getElementById("wykres").getContext("2d");
            const wykres = new Chart(ctx, { type: "bar", data: dane });

            // dynamiczne dodanie danych do wykresu
            wykres.data.labels.push("2021"); // dodanie etykiety
            wykres.data.datasets[0].data.push(1); // dodanie danych w pierwszym wierszu
            wykres.update(); // aktualizacja wykresu
        </script>
    </body>
</html>
```

Wynik dla powyższego przykładu:



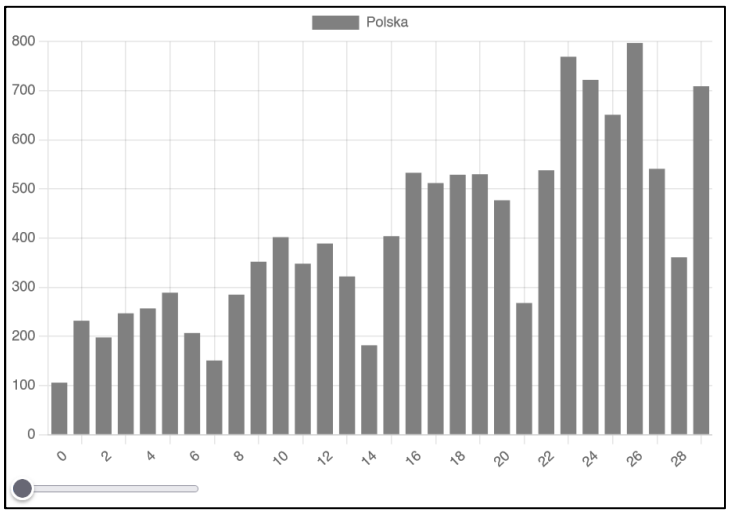
Dane można oczywiście modyfikować w dowolny sposób (usuwać, podmieniać, przesuwać za pomocą dowolnych metod działających na tablicach). Czynności te mogą zostać wykonane w dowolnym czasie (np. cyklicznie za pomocą setInterval, czy po kliknięciu przez użytkownika w przycisk).

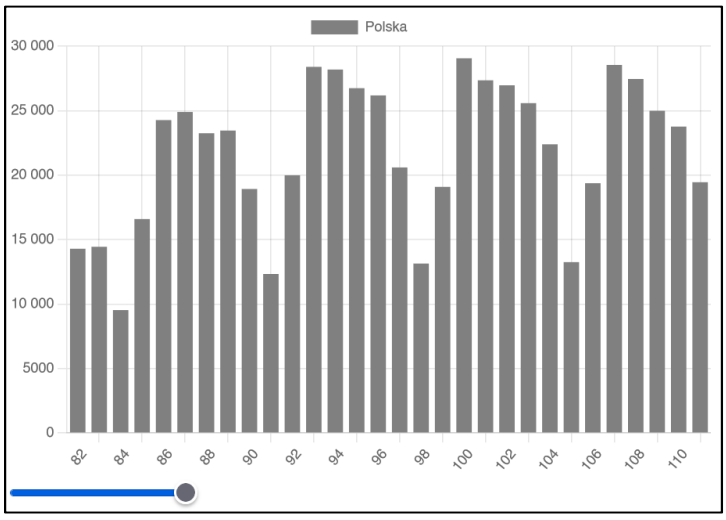
Poniższy przykład pokazuje dane, które zmieniają się przy zmianie wartości slajdera umieszczonego pod wykresem. W przykładzie umieszczono dodatkową tablicę (112 wartości – jest to liczba zakażeń covid w Polsce – źródło danych: gov.pl) oraz slajder (wartości od 0 do 82). Funkcja **pokazdane** pokazuje na wykresie zawsze 30 danych zaczynając od wartości slajdera (np. 40-69)

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8"><script src="chart.min.js"></script>
    </head>
    <body>
        <div style="width: 600px; height: 400px;">
            <canvas id="wykres" width="600" height="400"></canvas>
        </div>
        <input type="range" min="0" max="82" value="0" id="slajder" onchange="pokazdane()">
        <script>
            const cvpl = [
106,232,198,247,257,289,207,151,285,352,402,348,389,322,182,404,533,512,529,530,477,268,538,76
9,722,651,797,541,361,709,882,973,813,917,643,422,976,1231,1207,1362,1344,1089,686,1326,2085,2
006,1894,2012,1527,906,2118,2639,2999,2770,3234,2522,1536,3932,5558,5602,5709,6273,4731,2948,6
270,8365,8394,9395,9800,7155,4897,4519,10425,15516,15909,15204,12491,7317,13640,18555,19074,12
971,14292,14444,9533,16592,24269,24899,23246,23455,18924,12333,19987,28395,28187,26741,26180,2
0594,13141,19086,29062,27352,26964,25581,22389,13253,19373,28543,27455,24989,23762,19450 ];
```

```
            const dane = {
                labels: [ ],
                datasets: [ { label: "Polska", data: [ ], backgroundColor: "gray" } ]
            }
            const ctx = document.getElementById("wykres").getContext("2d");
            const wykres = new Chart(ctx, { type: "bar", data: dane });
            function pokazdane() {
                let value = parseInt(document.getElementById("slajder").value);
                let dowykresu = cvpl.slice(value, value + 30);
                let etykiety = [];
                for (let i = 0; i < 30; i++) etykiety.push(value + i);
                wykres.data.datasets[0].data = dowykresu;
                wykres.data.labels = etykiety;
                wykres.update();
            }
            pokazdane();
        </script>
    </body>
</html>
```

Wynik dla powyższego przykładu:





Zadanie 2

- Utworzyć wykres typu **scatter**, w którym będą umieszczone trzy punkty o współrzędnych: 0;0, 1;0, 1,1 (cały kod na dole poniższej strony).
- W **datasecie** z punktami zmienić sposób wyświetlania, tak aby punkty były połączone linią w kolorze niebieskim (opis na początku trzeciej strony przy typach wykresów). Właściwości wyświetlania ustawia się bezpośrednio w zbiorze danych – np. pod pointRadius w poniższym kodzie - nie trzeba ustawiać dodatkowych opcji w całym wykresie).
- Na stronie umieścić dwa pola **input** pozwalające użytkownikowi wpisać współrzędne nowego punktu (x i y) oraz przycisk **button**, po którego wciśnięciu punkt o podanych współrzędnych powinien zostać dodany do wykresu. Dla przypomnienia: w wykresie typu scatter nie trzeba dodawać etykiet, wystarczy do wykres.data.datasets[0].data dodać strukturę w postaci { x: wartość, y: wartość }, a wartości pobrać z pól input. Należy pamiętać o wywołaniu wykres.update() po dodaniu punktu.
- Przetestować działanie dodawania punktów, sprwadzić co się stanie w przypadku wpisania niepoprawnych danych (np. tekstu), a następnie podaniu poprawnych danych. Sprawdzić co się stanie w przypadku dodaniu kilku punktów o tych samych współrzędnych i najechaniu kursorem na te punkty
- Dodać przycisk, po którego kliknięciu pierwszy punkt ze zbioru punktów zostanie usunięty (metoda shift() bez parametrów dla tablicy punktów)
- **(opcjonalnie)** dodać użytkownikowi możliwość usunięcia punktu o wybranym indeksie

Kod do wykorzystania w zadaniu:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8"><script src="chart.min.js"></script>
    </head>
    <body>
        <div style="width: 600px; height: 400px;">
            <canvas id="wykres" width="600" height="400"></canvas>
        </div>
        <script>
            const dane = {
                datasets: [ {
                    label: "punkty",
                    data: [
                        { x: 0, y: 0 },
                        { x: 1, y: 0 },
                        { x: 1, y: 1 }
                    ],
                    backgroundColor: "violet",
                    pointRadius: 10
                } ]
            }
            const ctx = document.getElementById("wykres").getContext("2d");
            const wykres = new Chart(ctx, { type: "scatter", data: dane });
        </script>
    </body>
</html>
```

Zadanie 3 (opcjonalne)

- Poprawić przykład ze strony 6 w taki sposób, aby przy przesuwania suwaka wartości na wykresie płynnie się przesuwały. Aby to osiągnąć należy najpierw dodać brakujące dane na wykresie, a potem usunąć dane niepotrzebne. Tzn. jeżeli wyświetlane są dane od 30 do 59, a użytkownik przesunął suwak na wartość 40, najpierw należy dodać dane 60-69, a później usunąć dane 30-39. Na początku należy dodać dane bez sprawdzania tego warunku. Pod koniec zmienić także onchange na oninput.