

### **Esercizio 08 : chart js**

Scrivere una applicazione che:

- Richieda al servizio **/api** di randomuser.me un elenco di **n** persone con il valore di n impostato all'interno di una apposita finestra sweetAlert2



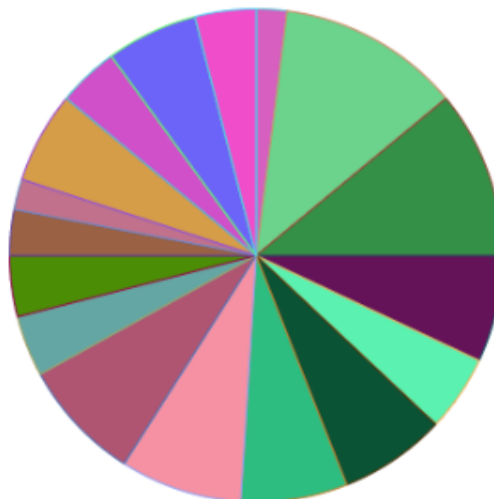
- Visualizzi sotto forma di tabella (creata dinamicamente) il numero di persone appartenenti a ciascuna nazionalità
- Un apposito pulsante consente di salvare l'intero json sul disco del PC locale
- Riporti la stessa informazione sotto forma di grafico
- Un apposito pulsante consente di salvare l'immagine sul disco del PC locale.

# Chart

invia richiesta

country	people
Turkey	2
Canada	12
New Zealand	11
Netherlands	7
Switzerland	5
Iran	7
Denmark	7
United States	8
Norway	8
France	4
Spain	4
Germany	3
Brazil	2
Australia	6
Ireland	4
United Kingdom	6
Finland	4

Turkey Canada New Zealand Netherlands Switzerland Iran Denmark  
United States Norway France Spain Germany Brazil Australia  
Ireland United Kingdom Finland



*I valori numerici sono visualizzati in corrispondenza del mouse over sulla torta*

[salva immagine](#)

## Note

- 1) Attenzione che l'istruzione **new chart()**, per quanto possa sembrare strano, è ASINCRONA, per cui il tracciamento del grafico viene avviato e poi l'elaborazione continua. Se subito dopo il **new chart()** accedo al canvas lo troverò vuoto. **await** NON è utilizzabile perché funziona SOLO con le **promise**.
- 2) Tra una chiamata e l'altra, anche se viene reistaziato, l'oggetto **Chart** mantiene i valori precedenti, per cui prima di istanziare il chart occorre necessariamente 'ripulirlo' con un **destroy**  

```
if(chart) // La variabile chart deve Necessariamente essere dichiarata globale  
chart.destroy();
```
- 3) Per salvare il canvas su disco, all'interno dell'onClick() si potrebbe settare l'attributo href del tag **<a>** utilizzando il seguente metodo dell'oggetto chart:  

```
this.href = myChart.toBase64Image();
```

dove myChart è il nome assegnato all'oggetto Chart.