

GABRIELE ROMANATO

Calcolare la distanza in chilometri tra due coordinate con JavaScript

Calcolare la distanza tra due punti sulla superficie terrestre, espressi in coordinate geografiche (latitudine e longitudine), è un problema comune nelle applicazioni che coinvolgono la geolocalizzazione. In questo articolo vedremo come implementare questo calcolo in JavaScript utilizzando la **formula dell'haversine**, un metodo ampiamente utilizzato per determinare la distanza tra due punti su una sfera.

La formula dell'haversine è un'equazione che consente di calcolare la distanza del percorso più breve (la "distanza ortodromica") tra due punti su una sfera, in particolare sulla superficie terrestre, considerando la curvatura del pianeta.

La formula è la seguente:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cos(\phi_2) \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a})$$

$$d = c \cdot R$$

Poiché la formula dell'haversine utilizza i valori in radianti, dobbiamo prima convertire le coordinate (esprese in gradi) in radianti.

```
function toRadians(degrees) {  
    return degrees * (Math.PI / 180);  
}
```

Ecco la funzione principale che implementa la formula dell'haversine.

```

function haversineDistance(lat1, lon1, lat2, lon2) {
  // Raggio medio della Terra in chilometri
  const R = 6371;

  // Convertire le latitudini e le longitudini da gradi a
  radianti
  const  $\phi$ 1 = toRadians(lat1);
  const  $\phi$ 2 = toRadians(lat2);
  const  $\Delta\phi$  = toRadians(lat2 - lat1);
  const  $\Delta\lambda$  = toRadians(lon2 - lon1);

  // Applicare la formula dell'haversine
  const a = Math.sin( $\Delta\phi$  / 2) * Math.sin( $\Delta\phi$  / 2) +
    Math.cos( $\phi$ 1) * Math.cos( $\phi$ 2) *
    Math.sin( $\Delta\lambda$  / 2) * Math.sin( $\Delta\lambda$  / 2);
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

  // Distanza in chilometri
  const distance = R * c;

  return distance;
}

```

Supponiamo di voler calcolare la distanza tra due città, ad esempio Roma (41.9028° N, 12.4964° E) e Milano (45.4642° N, 9.1900° E). Possiamo chiamare la funzione `haversineDistance` passando le rispettive coordinate.

```

const romeLat = 41.9028;
const romeLon = 12.4964;
const milanLat = 45.4642;
const milanLon = 9.1900;

const distance = haversineDistance(romeLat, romeLon,
  milanLat, milanLon);
console.log(`La distanza tra Roma e Milano è di
  ${distance.toFixed(2)} km.`);

```

Eseguendo il codice sopra, otterremo un risultato approssimativo:

```
La distanza tra Roma e Milano è di 477.23 km.
```

Spiegazione del codice:

1. **Conversione in radianti:** poiché la maggior parte delle funzioni trigonometriche di JavaScript come `Math.sin()` e `Math.cos()` accettano i valori in radianti, convertiamo i gradi delle coordinate in radianti.
2. **Calcolo della formula dell'haversine:** applichiamo i passaggi della formula dell'haversine per determinare il valore di a e c .
3. **Calcolo della distanza:** moltiplichiamo il raggio della Terra (R) per l'angolo centrale c , ottenendo così la distanza tra i due punti in chilometri.

Considerazioni

- Il raggio della Terra varia leggermente tra l'equatore e i poli. Tuttavia, l'approssimazione di 6371 km è generalmente abbastanza precisa per la maggior parte delle applicazioni.
- Se si desidera migliorare l'accuratezza in alcuni contesti specifici, si può utilizzare il raggio polare (6356.8 km) o il raggio equatoriale (6378.1 km).

Conclusione

Utilizzare JavaScript per calcolare la distanza tra due coordinate geografiche è piuttosto semplice grazie alla formula dell'haversine. Questa

tecnica è utile in molte applicazioni moderne, come la navigazione, la geolocalizzazione e i servizi basati sulla posizione. Il codice che abbiamo visto può essere facilmente adattato o esteso per includere funzionalità come il calcolo della distanza in miglia o l'uso di diverse approssimazioni del raggio terrestre a seconda della posizione.