Here's a **complete, runnable Angular example** showing how to add a **custom control** to an `ngx-leaflet` map.
We'll create a Leaflet control (e.g., a button) and integrate it into an Angular component.

1️⃣ Install Dependencies

```
npm install leaflet ngx-leaflet
```

Also, in `angular.json`, add Leaflet CSS:

```
"styles": [
  "node_modules/leaflet/dist/leaflet.css",
  "src/styles.css"
]
```

2️⃣ Angular Component with Custom Control

```
// map.component.ts
import { Component, AfterViewInit } from '@angular/core';
import * as L from 'leaflet';

@Component({
  selector: 'app-map',
  template: `<div style="height: 500px;" leaflet
[leafletOptions]="options"></div>`,
})
export class MapComponent implements AfterViewInit {
  map!: L.Map;

  // Leaflet map options
  options: L.MapOptions = {
    center: L.latLng(46.0669, 11.1217), // Example coords
    zoom: 13,
    zoomControl: true
  };
```

```
  ngAfterViewInit(): void {
    // Wait until map is ready
    setTimeout(() => {
      if (this.map) {
        this.addCustomControl();
      }
    });
  }

  // Called by ngx-leaflet when map is ready
  onMapReady(map: L.Map) {
    this.map = map;
  }

  // Create and add a custom control
  private addCustomControl() {
    const customControl = L.Control.extend({
      options: { position: 'topright' }, // 'topleft', 'topright',
'bottomleft', 'bottomright'

      onAdd: (map: L.Map) => {
        const container = L.DomUtil.create('div', 'leaflet-bar
leaflet-control leaflet-control-custom');

        container.style.backgroundColor = 'white';
        container.style.width = '30px';
        container.style.height = '30px';
        container.style.cursor = 'pointer';
        container.style.display = 'flex';
        container.style.alignItems = 'center';
        container.style.justifyContent = 'center';
        container.innerHTML = '📍';

        // Prevent map drag when clicking control
        L.DomEvent.disableClickPropagation(container);
```

```
        container.onclick = () => {
          map.setView([46.0669, 11.1217], 13); // Example: reset view
        };

        return container;
      }
    });

    this.map.addControl(new customControl());
  }
}
```

3️⃣ Template Binding for `onMapReady`

Update the template to bind onMapReady:

```
<!-- map.component.html -->
<div
  style="height: 500px;"
  leaflet
  [leafletOptions]="options"
  (leafletMapReady)="onMapReady($event)">
</div>
```

4️⃣ Styling (Optional)

```
/* styles.css */
.leaflet-control-custom:hover {
  background-color: #f0f0f0;
}
```

How It Works

- `L.Control.extend` creates a new control class.

- onAdd returns a DOM element that Leaflet will place in the map.
- `L.DomEvent.disableClickPropagation` prevents map panning when clicking the control.
- The control is added with `map.addControl(new customControl())`.

✅ This approach works with **ngx-leaflet** because it still uses the underlying Leaflet API.
You can replace the button with any HTML (e.g., legend, toggle, search bar).

If you want, I can also show you **how to make this control an Angular component** so it can have its own bindings and lifecycle.
Do you want me to prepare that version?