Overlay API: Use-cases

Angular 2 should provide official APIs for creating overlays in applications. This document lists various types of overlays we should support.

# Z-indexing

Overlays are pieces of UI that appear on top of other content. It is important that we support precise control over z-index of overlays.

Angular 2 should support two modes:

* Mode 1: this is used when overlays appear on top of elements inside a scrollable area.
* Mode 2: this is used when overlay appears on top of everything, ignoring scrolling/overflow-hidden settings.



Unfortunately, the normal CSS z-index is not sufficient to achieve both modes\*. When “overflow” property is set to scroll/hidden z-index cannot override that property; the content would still appear hidden. The element needs to appear outside the scroll area in the DOM.

# Use-cases

## Tooltip

A tooltip is triggered by hovering the mouse pointer over or clicking a target element on the UI. It appears either relative to the pointer location or relative to the location of the target. If it is the latter, scrolling the content must adjust the location of the tooltip:



## Multi-level popup menu

This type of overlay is similar to tooltip, but usually triggered via a click and has multiple levels, each appearing at a higher z-index than the last:



One important aspect of multi-level popup menus like this is that it requires precise control over keyboard focus. Once a menu is invoked, the user should be able to use keyboard keys UP/DOWN/LEFT/RIGHT/ENTER to navigate it and select menu items.

## Drawer

This is extremely common in mobile apps but recently has been used in desktop web-apps as well. It’s a type of sidebar that is hidden and invoked by clicking (touching) on the “hamburger” icon in the corner. When invoked it either shifts the page contents or appears on top of them:



## Auto-hide header

A page header that appears when the user scrolls *up* and disappears when the user scrolls *down*.



## Dialog

A dialog is a popup with fixed positioning on the screen. It is not tied to any specific element on the UI:

