Decoration ordering

Status

Accepted

Summary

Order block decoration items in the DOM in a deterministic and controllable way.

Motivation

When multiple block decorations are created at the same screen line, they are inserted into the DOM in an order determined by the sequence of their creation; from oldest to newest when `position` is set to `"before"`, from newest to oldest when `position` is set to `"after"`. While this is deterministic, it is limited: it isn't possible to insert decorations within a sequence of existing ones, and it's difficult to control the order of decorations when creating and destroying and moving markers around an editor.

We hit the need for this in atom/github#1913 when we have a block decoration for multiple consecutive collapsed file patches.

Explanation

TextEditor::decorateMarker() accepts an additional `order` parameter in its `decorationParams` argument when `type` is "block". When multiple block or overlay decorations occur at the same screen line, they are ordered within the DOM in increasing "order" value.

Block decorations with the same `order` property are rendered in the order they were created, oldest to newest. Block decorations with no `order` property are rendered after those with one, in the order in which they were created, oldest to newest.

Drawbacks

This is a breaking change for co-located block decorations created with an "after" position - they'll now appear in the reverse order.

When multiple packages create block decorations at the same screen line, they'll need to coordinate their `order` values to have expected behavior. There may not even be a clear, universal answer about how block decorations from distinct packages *should* be ordered.

This adds another situational parameter to `TextEditor::decorationMarker()`, which already has complicated arguments.

Rationale and alternatives

Originally I wanted to address the package coordination problem with a similar approach to the way context menu items are ordered, by allowing individual decorations to specify constraints: "before this block," "after this block," "next to this block" and so forth. I ultimately chose to write up the simpler proposal because:

- Block decoration collisions among packages seem much less likely than context menu collisions.

- Constraint satisfaction problems are complex. There would be a relatively high chance of introducing bugs and performance regressions.
- The order number approach is similar to the APIs already offered to order status bar tiles and custom gutters.

The alternative to having an explicit API for this is at all is to create and destroy decorations to achieve the desired order. That's possible, but requires a great deal of bookkeeping on the package's side to accomplish, especially as decorations are added and removed and text is edited.

Unresolved questions

- Should overlay decorations respect an `order` parameter in a similar fashion?
- Should screen column effect decoration ordering at all?