The Backdrop Root

Note: this argument/discussion is very similar in rationale and conclusions to the discussion (see here and here) about the conditions that cause 3d transform flattening. We might be able to combine these concepts to avoid defining a new term/concept.

Web operations that blend elements with their ancestors can cause certain ambiguities. For example, because the DOM is defined hierarchically, with filters and opacity applying atomically to all descendants, it becomes unclear what the proper order of operations should be, when descendant elements need to access the “backdrop” behind them. See below, in the section called Motivation, for more details.

To clarify this ambiguity, in particular for mix-blend-mode and backdrop-filter, we propose to define a new term, called the Backdrop Root. All such blending operations applied to an element (call it B) would apply to all content painted between the Backdrop Root element and element B. No content that is an ancestor of the Backdrop Root would be filtered/blended by element B.

There are currently three other related, but distinct, concepts in the web platform:

- The Stacking Context. A Stacking Context is induced by many different types of element properties, and is primarily related to defining the **painting (Z) order** of elements.
- The Containing Block. A Containing Block is also induced by several types of element properties, and is primarily related to defining how **layout (X/Y)** is performed for elements.
- The 3D Rendering Context. A 3D Rendering Context is induced by several properties, mainly transform-style, but also any of the grouping properties, and is primarily related to defining the relevant coordinate space for **3D transforms**.

To this list, we would like to add a fourth concept:

- The **Backdrop Root**. A Backdrop Root is induced by several properties, and is primarily related to defining the elements that contribute to the “**backdrop**” of an element.

A Backdrop Root is formed, anywhere in the document, by any element in the following scenarios:

- The root element of the document (HTML).
- An element with a filter property other than "none".
- An element with an opacity value less than 1.
- An element with mask, mask-image, mask-border, clip, clip-path properties with values other than “none”.

- An element with an overflow property other than “visible”, **and** that is not axis-aligned with its parent. For example, an element with style=”transform: rotate(45deg); overflow: hidden;“.
- An element with a transform-style value of “preserve-3d”.
- An element with a backdrop-filter value other than "none".
- An element with a mix-blend-mode value other than "normal".
- An element with a will-change value specifying any property that would create a Backdrop Root on non-initial value.

Note that this definition encompases fewer element types than the definition for a Stacking Context. In particular, a Backdrop Root is **not** formed by elements with z-index applied, fixed or sticky-positioned elements, and elements with most normal transforms. This allows elements with backdrop-filter or mix-blend-mode to apply to elements higher up the DOM tree than would otherwise apply if they stopped at the parent Stacking Context. For example, a container can be used to contain elements with backdrop-filter applied, and that container can use transforms for animation or positioning, while still allowing the backdrop-filter to apply to the background behind the container.

Motivation

There is an important motivation for the creation of yet another web platform concept for Backdrop Root. Earlier proposals have either called for backdrop-filter (and to some extent mix-blend-mode) to filter either 1) “everything that painted before, up to the root node”, or 2) “everything up to the parent stacking context”. However, there are ambiguities with definition #1, and excessive limitations with definition #2. Therefore, we are proposing an intermediate definition for “what comes before” a backdrop-filter or mix-blend-mode.

In particular, it is important to note that including “everything that paints before” an element is ambiguous in the case where an ancestor of the element contains filtering and/or opacity. Because those effects are inherited by descendant elements, including the element containing backdrop-filter or mix-blend-mode, it is not clear “when” to apply the effect. Filters and opacity create a stacking context, and the css filter-effect specification states that “All the [Stacking Context] descendants are rendered together as a group with the filter effect applied to the group as a whole.” If, somewhere inside that stacking context, an element contains a backdrop-filter property other than “none”, then it is technically impossible to honor the preceding sentence. At the point of the backdrop-filtered element, all of the (partially-painted) contents of the stacking context need to be fully rendered, including applied filters, opacity, and blending with the backdrop, and the resulting image needs to be used as an input to the backdrop-filtered element. That breaks the atomicity of the stacking context. And the ambiguity arises from the need, at the end of rendering the fully-completed stacking context, to apply those filters and opacity **again** to the completed rendering. The filters and opacity will, at this point, be applied

twice to the portion of the image that has been backdrop-filtered. This situation grows exponentially worse if backdrop-filters are nested inside each other. (As a side-note, nested backdrop-filters would also cause exponential performance degradation in this case.) It is therefore necessary to not allow the backdrop to be “seen” by a backdrop-filtered element above nodes that have filters, opacity, and the other conditions listed above in the definition of the Backdrop Root.

It is also important to note that the most straightforward definition of backdrop-filter or mix-blend-mode would clearly be for these effects to filter “everything behind them”, without having to understand the nuances of the Backdrop Root. It would be much easier, and more powerful, for developers to be able to apply the full set of web platform tools available, and still allow the backdrop-filtered element to “see” everything on the page. However, for the reasons listed above, it is important to protect the developer from very counterintuitive and/or poorly-defined (and therefore variable among browser implementations) effects and behavior. It is for this reason that we would like to define this new Backdrop Root concept, to strike the proper balance. If we instead just use the Stacking Context as the boundary for backdrop-filter, then developers are prevented from using many common tools such as transforms, which pose no equivalent ambiguity for backdrop-filter.