Shared Element Transitions
Rendering Model

Documents - old and new

Old DOM (SPA or MPA, doesn't matter)

```
<style>
#my-shared-element {
   width: 100px; height: 100px;
   page-transition-tag: foo;
   position: absolute;
   top: 300px;
   left: 100px;
}

html { page-transition-tag: root; }

body { background-color: lightblue; }
</style>

<div id=”my-shared-element”>Text</div>
```

New DOM

```
<style>
#my-shared-element {
   width: 100px; height: 100px;
   page-transition-tag: foo;
   position: absolute;
   top: 150px;
   left: 400px;
}

html { page-transition-tag: root; }

body { background-color: pink; }
</style>

<div id=”my-shared-element”>Text</div>
```

Layout Box Tree Representation

uber-root
- root
  - html
    - div
      - text
  - top layer
- **::transition**
  - ::container(root)
    - ::wrapper(root)
      - :: old-content(foo)
      - :: new-content(foo)
  - ::container(foo)
    - ::wrapper(foo)
      - :: old-content(foo)
      - :: new-content(foo)

UA Styling

uber-root
- |_ root
  - |_ html
    - |_ div
      - |_ text
  - |_ top layer
  - |_ ::transition
    - |_ ::container(root)
      - |_ ::wrapper(root)
        - |_ :: old-content(foo)
        - |_ :: new-content(foo)
      - |_ ::container(foo)
        - |_ ::wrapper(foo)
          - |_ :: old-content(foo)
          - |_ :: new-content(foo)

```
::transition {
  position: fixed; top: 0; left: 0; width: 100vw; height: 100vh;
}
::container(root) {
  width: 100vw; height: 100vh;
  z-index: 0;
}
::container(foo) {
  position: absolute; width: 100px; height:100px;
  transform: translate(300px, 100px);
  z-index: 1;
}
::wrapper(foo) {
  inset 0; isolation: isolate; mix-blend-mode: plus-lighter;
}
::old-content(*)  {
  position: absolute; top: 0px; left: 0px;
  width: 100%; height: 100%;
}
::old-content(root) {
  content: cached-element(root);
}
::new-content(foo) {
  content: element(foo);
}
shared-root {
  visibility:hidden;
  contain:paint !important;
  break-inside: avoid  !important;
}
shared-foo {
   visibility:hidden;
  contain:paint !important;
  break-inside: avoid  !important;
}
```

UA Animations

```
uber-root
    |_ root
    |  |_ html
    |  |  |_ div
    |  |     |_ text
    |  |_ top layer
    |_ ::transition
       |_ ::container(root)
       |  |_ ::wrapper(root)
       |     |_ :: old-content(root)
       |     |_ :: new-content(root)
       |_ ::container(foo)
          |_ ::wrapper(foo)
             |_ :: old-content(foo)
             |_ :: new-content(foo)
```

```
@keyframes foo-keyframes {
  from {
    width: 100px; height: 100px;
    transform: translate(300px, 100px);
  }
}

::container(foo) {
  animation: foo-keyframes 0.25s;
  transform: translate(150px, 400px);
}

@keyframes fade-in {
  from { opacity: 0; }
}

::new-content(*) {
  animation: fade-in 0.25s;
}

@keyframes fade-out {
  to { opacity: 0; }
}

::old-content(*) {
  animation: fade-out 0.25s;
}
```

What won’t look the same as the real documents?

- Inherited Properties
  - Effects such as clipping, filters, and opacity between a shared element and its ancestors are not preserved.
  - Positioning and screen-space transform are preserved.
- Paint order
  - Shared elements always paint on top of other content. (But relative paint order of two shared elements is preserved.)
- Live animations of DOM
  - All UA style rules/animation state are live-updated if the developer changes anything in the new document’s DOM.
  - Contents of the element() function for the new document live-update.
  - Contents of the cached-element() function do not live-update.

Spec changes needed

1. Updates to the element() spec when used with this feature:
    a. Increasing image size to include self ink overflow.
    b. Do not support paint sources, each shared element must generate a box.
2. Define element() for the DOM's root element
3. Define cached-element() to refer to a cached output of element().
4. Define the transition stacking context.
5. Define box tree generated using pseudo elements and selectors for targeting them in UA and developer stylesheets.
6. Define default UA stylesheet animations (duration, easing and clipping for changes in box-size).
7. Define a replaced element which overflows its box (for ::old-content/::new-content).
8. Functions to target shared elements in UA stylesheet.
9. Plus-lighter/lighter mix-blend-mode (proposal).
10. An easing function (platform-ease) which is platform dependent to match animations with OS aesthetics.