CSS, SVG AND HTML CONVERGENCE

A conversation with the CSS Working Group

Author:
Patrick Dengler, Microsoft, Member of W3C SVG Working Group and SVG-CSS Task Force
Contributors:
Sylvain Galineau, Microsoft, Member of W3C CSS and Web Fonts Working Groups

Purpose: To frame the discussion at Technical Plenary Advisory Committee (TPAC) on the subject converging CSS, SVG and HTML in the short term.

SVG in HTML raises issues including:

- How do CSS Transforms align with SVG?
- Can CSS Transitions and Animations with Synchronized Multimedia Integration Language (SMIL) be rationalized?
- How do CSS paint servers (such as gradients) align with SVG?
- Should SVG filters be supported by CSS and apply to HTML?
- Should some SVG attributes be moved into the presentation property category, which would allow for better integration with CSS in the future?
- Can differences be accounted for between existing concepts, without breaking backward-compatibility?
- Should CSS support attributes in addition to styles?
- Should the differences in programming models between the HTML DOM and the SVG DOM be eliminated?
- Should we align all elements and attributes that are virtually identical?
- Should architectural differences, such as document structure and coordinate systems, be normalized between the two technologies?
- Since SVG is an XML format, should it move more toward an HTML model?
- Should SVG be considered foreign content in an HTML document?

APPLYING GUIDING PRINCIPLES TO RATIONALIZE TECHNOLOGIES

The guiding principles proposed in the HTML.next document address how to approach the issues mentioned above. In summary, these are:

- Identifying key (not exhaustive) use cases
- Extracting requirements from use cases
- Scoping the work to meet short-term deliverables

- Focusing efforts on a smaller set of requirements
- Providing tests for each new assertion extrapolated from the requirements
- Modularizing specifications to move them forward quicker
- Establishing a target date up front for Last Call

Based upon customer feedback, integrated support for transforms and animation is a top priority.

The good news is that animations are currently being investigated as a part of 2D Transforms across SVG and CSS. The work done here has been great. The next step is to raise the discussion up a level to find a shorter path to resolution.

By isolating the investigation to 2D transforms (and this is not absolute), bigger issues such as backward compatibility, general CSS and SVG attribute alignment in the naming, values, defaults, data types, functions, and overall integration into HTML are beginning to surface.

In order to resolve the alignment of animation and transforms, the following issues need to be addressed:

- Align CSS Transform functions and OM with SVG (mostly complete)
- Identify how to apply CSS Transitions and Animations ***only*** to SVG Transforms (mostly complete)
- Examine how a ***single*** SVG attribute and one additional attribute should become a presentation property to integrate with CSS Transitions and Animations (scoping)
- Reduce or eliminate backward-compatibility problems
- Align ***some*** core programming concepts between the HTML DOM and the SVG DOM as related to animations

The following proposal demonstrates that this smaller, scoped set of work items does not interfere with future design considerations for the larger set of open issues associated with integration. This is the primary motivation that should be included in the process of creating or modifying existing specifications.

ANIMATION

To provide a richer graphical experience for end users, developers want to take advantage of transforms, transitions, and animations that work in an integrated model using their existing tools, process, and programming experience.

Animation scenarios cross HTML and SVG. CSS 3 introduces two complimentary animation models: transitions and animations. Together they provide moderately complex animations. CSS Animations and Transitions are in high demand by developers and are already implemented in most browsers. SVG currently uses Synchronized Multimedia Integration Language (SMIL) for declarative animation. Though SMIL covers all of the same scenarios as CSS animations, CSS animations do not address animation along a path, nor video and audio synchronization. These additional features can be additive to the CSS specifications at a later date, following the method of stabilizing portions of a module. SMIL is not currently considered for HTML.

SCENARIOS

Nickr, a photograph sharing site, wants to provide visual cues to end users that allow them to have a more instinctive understanding of the interaction model. They need lightweight animations (or transitions) when a user hovers over a picture. They also need to bring the picture to the forefront with a ‘bounce’ when selected.

Politigo, a worldwide election results tracking site, wants to enhance their end user experience by providing interactive geographic maps to navigate the election data. They also want to allow their user to hover over or select a state, country or other municipality that becomes highlighted and then animates to a larger size.

Floeing has used SVG historically for viewing and printing engineering diagrams and has a lot of existing assets. These assets must display and print without modification.

Author's Note: (the above is to reinforce the point that backward compatibility is extremely important, but the scenarios need to be identified)

HomeNeato, a retail store, wants to provide an even deeper interaction for their users who purchase merchandise from their online site. They are eager to create an experience where users can navigate a three-dimensional model view on their products.

RoosterTooths, an animated entertainment site, wants to add enhancements with eye catching animations. They need to integrate documents, vector graphics, and audio concepts together to achieve the desired effect. One specific bit of functionality is to surface an airplane that flies across the screen with a waving banner that announces new content.

PRIORITIZATION

Taking the scenarios above, the following requirements are extracted and prioritized to deliver staged, stabilized and interoperable features for Web developers.

REQUIREMENTS

In Scope

- Declaratively animate SVG and HTML with a single model
- Promote the Transform attribute in SVG into a presentation property
- Add transform-origin to SVG as an attribute and a presentation property
- Ensure existing SVG images and SVG documents continue to function as expected
- Support the more granular transform functions from CSS in SVG
- Provide a consistent programming model across SVG and HTML

Out of scope

- Animating a collection of SVG graphic elements, text, or HTML elements along a path
- 3D Transforms and Animations on 3D Transforms
- Synchronizing animations and transitions with audio and video

HIGH-LEVEL DESIGN

Align the SVG Transform and CSS Transform concepts, expose transform and transform-origin in SVG as a property, and preserve the different defaults of transform-origin property between SVG and HTML.

The assumption is that transform alone will satisfy most scenarios for SVG integrated in HTML based upon the previous scenarios.

Proposal:

- Promote transform and introduce transform-origin as properties for SVG
- Preserve default transform origins for both CSS (50%,50%) and SVG (0,0)
- Apply CSS Transitions and CSS Animations on SVG in HTML (including transforms)
- Preserve SMIL for content not integrated into Web pages or Web applications; investigate in the next phase integration of SMIL and CSS.

This reduces the effort and still enables a significant amount of animation scenarios.

Author's Note: I have not been shy about my position on SMIL and CSS Animations and Transitions. I don't believe we should have two declarative animation models for the Web. However, I have not yet convinced myself. So these proposals are about short term, scoped features.

Proposal: When embedded inline, SMIL is ignored in an HTML context..

Features Reserved for Future Consideration

In future, SVG in HTML should satisfy more support for both motion on a path animations and synchronizing media that need to be added into CSS animations. These are not small features, yet they do not appear to impact the following detailed design.

There have also been discussions about converting more SVG attributes into properties and enabling more CSS styling. Because there are enough variances to believe that this will be a lengthy, arduous process, this should be a long-term goal.

Aligning Transform and transform-origin

The transform and transform-origin discussion is a summary of what has already been detailed in the 2D Transforms across SVG and CSS. It is provided here for completeness.

Simply stated, if transform is a property in SVG, a new property, transform-origin, can be introduced, that allows developers to use the different default behaviors in SVG and CSS. They can also define identical behaviors by including transform-origin into a style. This creates a consistent model of styling CSS across SVG and HTML. The concept of styling SVG with CSS is consistent with HTML and already familiar to developers.

The rotate attribute and function for SVG uses the origin (0,0) while in CSS it uses the default origin 50%/50%. Both make sense for each technology and should not change.

The following is an example of transforms where the transform origin defaults differ, and transform-origin does not exist in SVG (note, the rotate function allows for expressing an origin, but the attribute does not).

Currently in SVG, the following rotates a rectangle 45 degrees at the coordinate 0,0:

```
<rect transform="rotate(45)"/>
```

Whereas in HTML, the following markup rotates a div 45 degrees at the center point:

```
<div style="transform: rotate(45deg);"/>
```

There is the ongoing issue of CSS supporting typeless values, and this property is no exception. It is not unreasonable to assume that the degree unit type would make sense as the default for CSS, but notice that it is not required. It is only when the transform is expressed as a style that the CSS unit types might come into play.

Following the previous design, developers could leverage the benefits of CSS, such as stylesheets, classes, selectors, transitions, and animations on SVG in HTML through the transform property without disturbing content that currently uses transforms as attributes for SVG documents and images.

The following style would affect both <div>s and a <rect>s, rotating them 45 degrees on the center point:

```
div,rect {transform: rotate(45deg);transform-orgin:50% 50%}
```

Aligning Transform Functions

The set of transform functions also requires normalization. The current list expands the transform API in SVG to include the following, more granular, but not conflicting, functions:

- translateX
- translateY
- scaleX
- scaleY
- skew

Again, the 2D Transforms across SVG and CSS specification already address this.

Impact on SVG DOM

The impact of the CSS animation model on SVG also requires analysis of the animated type system in the SVG DOM (some thoughts on the SVG 2.0 DOM have been documented).

While ultimately it will take some time to fully specify the DOM for SVG 2.0, animated types need to be examined in order to provide a consistent programming model across SVG and HTML.

Most data in SVG provide values for initial and current animated value. Each animated data type allows for animVal, and baseVal (which is the initial value). For example:

```
document.getElementById("myRect").height.baseVal.value = "200";
```

is required to set the value of the height of a rectangle. And:

```
document.getElementById("myRect").height = "200";
```

yields `Invalid set operation on read-only property` as specified. Furthermore, setting the baseVal to a value with a unit type requires another unexpected function called valueAsString:

```
document.getElementById("myRect").height.baseVal.valueAsString = "200px";
```

It is generally agreed that the SVG DOM should follow the accepted DOM L2/L3 models. To align program models and achieve backward-compatibility, some changes would need to be made. The following SVGRectElement and underlying SVGAnimatedLength are used as examples.

In the IDL, a combination of [PutForwards] and toString()make each object appear like an attribute, while behind the scenes the sets/gets are forwarded to the right sub-object-property. SVGLength will store the value, but the properties on SVGRectElement will appear to have been simplified as expected. The adjusted interfaces and supporting example code are included here:

```
interface SVGRectElement : SVGElement
 {
  [PutForwards=baseVal] readonly attribute SVGAnimatedLength  x;
  [PutForwards=baseVal] readonly attribute SVGAnimatedLength  y;
  [PutForwards=baseVal] readonly attribute SVGAnimatedLength  width;
  [PutForwards=baseVal] readonly attribute SVGAnimatedLength  height;
  [PutForwards=baseVal] readonly attribute SVGAnimatedLength  rx;
  [PutForwards=baseVal] readonly attribute SVGAnimatedLength  ry;
};
…
```

Then the modification to SVGAnimatedLength is as follows:

```
interface SVGAnimatedLength {
  [PutForwards=valueAsString] readonly attribute SVGLength baseVal;
  readonly attribute SVGLength animVal;
  DOMString toString(); // Returns baseVal.valueAsString
};
```

Then code can be written as expected by Web developers in the following manner:

```
rect.x = 5;
```

OR

```
rect.x = "5px";
```

Since vendors have implemented SMIL, it preserves animVal and baseVal for existing content or content intended for SMIL (that is not under and HTML document).

```
rect.x.baseVal = 5; OR
var animX = rect.x.animVal;
```

Then, code can be written as expected by web developers in the following manner:

```
document.getElementById("myRect").height= "200px";
```

This creates the expected and consistent programming experience for the Web developer.

PROJECT EXECUTION

This proposed high level design for the first phase of SVG 2.0, includes closure on portions that some CSS 3.0 specifications in tandem. To succeed at these in the short term, the following project plan is proposed:

Task	Driver	Due Date
Sign off by the CSS-SVG Task Force on Scenarios and High Level Design	FX	11/15/2010
Prepare 2D Transforms across SVG and CSS for last call	FX	01/15/2010
Scope and prepare CSS 3.0 Transitions Part 1 for last call	CSS	02/15/2010
Scope and prepare CSS 3.0 Animations Part 1 for last call	CSS	02/15/2010
Scope and prepare the SVG DOM 2.0 and Data Types Modules Part 1 for last call	SVG	02/15/2010
Deliver initial and sufficient test cases across specifications	FX	03/15/2010

While this schedule may seem aggressive, by identifying the scenarios up front, scoping the design and providing laser focus to the previous areas, cross technology animations and transitions through transforms can be delivered in a common way. This is the most highly desired feature by Web developers.

SUMMARY

The SVG Working Group is completing the final tasks to prepare the latest SVG 1.1 2nd Edition into a Last Call sometime this year. The SVG Working Group has agreed that all other efforts need to be focused on integrating these technologies in cooperation with CSS.

Because the proposed schedule for alignment is short, a different style in the working groups must be considered. Specifically, the combination of functional area owners (such as animations) who are responsible for scenario development, requirements gathering and specification writing process, combined with specification modularization, will accelerate the process to meet these dates. This will lower the time barrier for introducing the required features. Lastly, conference calls should be limited to tracking the status of work items to ensure the project is on track.