Fluid Project Recommendation for Use of CSS Frameworks

Summary and Background

Up to version 1.5, Infusion shipped with a built-in CSS framework called the Fluid Skinning System. When the FSS was first designed, there were few viable production-scale CSS frameworks available to developers. As a community, we felt a need to provide something that had been designed with accessibility and flexibility in mind. Since then, we've experienced a massive shift in the way websites are designed and implemented. Responsive Web Design is now widely practiced, enabling web pages to adapt to different screen sizes and device form factors. CSS preprocessors such as Less and Sass have changed the way style sheets are developed and reused. Mature and widely-used CSS frameworks such as Bootstrap and Foundation are available to help ease the effort of styling responsive layouts, grids, and widgets.

Unfortunately, the Fluid Skinning System hasn't kept pace. As a result, we chose to deprecate it in version 1.5. We recommend that users of Infusion choose from any of the available third-party frameworks based on their own needs and requirements. As of Infusion 2.0, we will remove FSS entirely.

Infusion components currently don't rely on any CSS framework, and we will continue to ensure that they are styled in a way that is unlikely to conflict with popular tools like Bootstrap and Foundation. For the foreseeable future, we don't plan to include any third-party CSS framework with Infusion, nor to require that our users select a specific framework.

Who is this document for?

This document reflects the decision-making process we have undertaken as a community to select a CSS framework for some of our own websites, documentation, and demos. It may also be useful when selecting your own CSS framework to use alongside Infusion.

Recommendation Summary

Fluid Infusion components do not use a CSS framework out of the box. The motivation for this is to ensure Infusion components play nicely in whatever application is used. Integrators may choose to add their own CSS framework to Infusion components if they wish.

Using a CSS Framework within an Infusion component

- Components shipped as part of the core Infusion package do not use a CSS framework for their styling.
- Some CSS frameworks don't use namespaced or prefixed class names, which creates the possibility of collisions that can cause styling problems. Infusion component styles are always namespaced, which reduces the potential for Infusion to cause conflicts with non-namespaced frameworks. See Infusion Class Name Conventions for more information.
- Because Infusion components do not use a CSS framework by default, component creators will need to create good default styles and leave it to the integrator to customize with their own CSS framework if they choose to.

Using a CSS Framework within Fluid-related demos, websites, etc.

- A CSS framework is fine to be used for websites, demos, and other integration / non-component scenarios. We recommend Foundation because of its use of REM sizing throughout.

Using a CSS Framework for contrast themes in UI Options and the Preferences Framework

- At this time it is difficult to create contrast themes for CSS frameworks despite the availability of custom builders. Custom builders may not be an appropriate tool as they do not cover all possible styling rules we want to affect.
- Creating a theme manually is non-trivial (lots of values to adapt)
- Themes would have to be tested with each major and minor framework release to ensure compatibility. This adds a maintenance burden.

Future possibilities

- There may be an opportunity to add new features to Learner Preferences that can transform CSS framework components (like navigation bars and button links). Further discussion is encouraged.
- Contrast theme generation using a CSS pre-processor to be investigated. Possible pre-processors include: Less, SASS, and Stylus. This will be a separate discussion.

Table of Contents

Appropriate Use of a CSS Framework

Example: Metadata component

- The Metadata component for the Floe project has two major pieces: the component and the demo.
    - See the project source code here: https://github.com/fluid-project/metadata
    - See the project designs here: FLOE Metadata Authoring Design
- The metadata component is styled using a "good default" style which is intended to be usable in most applications. The metadata component itself does not use a CSS framework.
    - See the component's styling here: https://github.com/fluid-project/metadata/blob/master/src/css/style.css
- The metadata demo itself uses Foundation as a CSS framework as it is an integration scenario.
    - See the demo's styling here: https://github.com/fluid-project/metadata/tree/master/demos/css
- If *both* the metadata component and the metadata demo used the same CSS framework, the styling of the demo would have collided with the styling of the component since the styles would have shared the same namespace and scope. Fixing this issue would have involved adding many styling overrides to restore the Metadata component to its original state.

Choosing a CSS Framework: Why Zurb Foundation?

In deciding which framework to recommend for the Fluid project, many criteria were considered including: documentation, community support, accessibility, and scalability.

- Also see: CSS Framework Research Notes

After researching and comparing 6 different CSS frameworks, Zurb Foundation is the recommended framework for the following reasons:

- Foundation uses REMs as units which ensures scalability across clients and devices.
- Foundation's grid system is flexible - it's fluid width and has a cleaner structure which makes integration and customization easier.
- We typically work in highly customized environments and Foundation is light weight and doesn't get in the way (mostly).

Can I use a different framework?

Yes, you can use any framework you prefer. For Fluid-related websites and demos, we prefer frameworks that are:

- Accessible and scalable
- Mature and widely supported
- Open source

Future Considerations

Better namespacing

- At this time it is very difficult to namespace CSS framework classes. Most solutions are a manual process, or a manual process that is automated by a script. This is fragile and requires constant testing with each new framework release.
- If a CSS framework or tool emerges to support custom name spacing easily, and satisfies our needs (i.e. scalable, accessible, widely supported, etc.) - we should revisit the topic of using a framework for Infusion components.

Add CSS framework support to the Preferences Framework

- Since CSS frameworks share common features (such as navigation bars, link buttons, progress bars, breadcrumbs), this can lead to some new preferences.
- For example:
    - Linearize / stack content by changing the grid layout
    - Navigation bar preferences: make tabs look like pills, make navigation pills stack
    - Pager: size, appearance, spacing
    - Alerts: change the way alerts are positioned or styled (i.e. make them bigger, put them in a floating panel etc.)
- In order to support multiple CSS frameworks, consider using an adapter model where the Preferences framework supports a generic definition and adapters are created to map CSS framework definitions to the framework. i.e. Framework 1 may use divs for breadcrumbs, and Framework 2 may use a list. The adapter would define this configuration so that the framework can use both.