Infusion Versioning and Release Strategy

There are a lot of JavaScript frameworks available to choose from. Infusion was one of the first, and it has continued to evolve and grow over the last seven years. As a community of innovation-focused designers and developers, we don't spend much time advertising or evangelizing our work. Yet despite the proliferation of frameworks in recent years, Infusion continues to stand out for its unique goals and features.

The basic premise of Infusion is that current industry software development practices and tools are ill-suited to creating adaptable and configurable applications that take diverse user needs into account. Specifically, Infusion's goal is to support the creation of:

- adaptable software that allows users to customize its appearance and behaviour to suit their personal preferences
- assistive technologies that are integrated directly into mainstream user interfaces and that can access richer information than provided by current assistive technology APIs
- authoring tools that allow users to configure, redesign, or develop a piece of software without having to be expert programmers

Without development tools to support this level of adaptability and customizability, inclusive designers are faced with the challenge of creating "universal" user interfaces that attempt to shoehorn all user needs into one design. It's a time-consuming and costly prospect. As result, accessibility features are routinely left out or bolted on afterwards. Until we have new software development techniques and frameworks that support adaptability by default, the accessibility field will continue to be largely reactive and remedial, trailing behind mainstream innovations.

This is a big task. Changing the foundations of software development practices takes time and necessarily requires a degree of experimentation, speculation, and willingness to make mistakes. We don't have all the answers ahead of time; as we gather new insights and devise new techniques, Infusion (and its API) changes. These insights are inevitably gained by building real, large-scale applications with Infusion and putting them into production. We call this *production-scale research*. Where traditional software research projects typically emphasize highly constrained laboratory prototypes that are intended to illustrate a particular research hypothesis, Infusion needs to leave the laboratory in order to be collaboratively developed, evaluated, and iterated upon.

Production-scale research involves pursuing innovation while also supporting stability for developers who are using the system to build actual software. The Fluid community has devised a set of techniques and practices that are critical to supporting production-scale research, which are based on well-established open source and agile development practices:

- Comprehensive unit tests for all code
- Code review for all changes and new contributions
- Quality assurance testing prior to major releases
- Nightly builds
- A-Grade browser support
- Designer-led user interface implementation
- Extensive documentation and tutorials

In the past, Infusion followed a relatively slow release schedule. Major releases were rare, and the incubation time between releases was fairly long. However, rapid releases are becoming the norm for many open source projects. To this end, the Fluid community has adopted the Semver 2.0.0 standard, and will release and publish versions of Infusion to npm more frequently.

As per Semver, all changes that affect Infusion's public API will be marked as major revisions. Since Fluid is currently in a period of change and responsiveness to user needs, we anticipate that Infusion's major version number will increase regularly as we add new features and improvements. This is consistent with many other open source projects such as Node.js. We will, however, identify particular versions that may need long-term support based on feedback and requests from our in-production users. This will help to ensure that the rate of change is balanced with the need for stability by production-scale implementers.