William Austin

07/05/2023

WEB 420 - Discussion 7.1 - Semantic Versioning

On any given day if you go to your app store you can see what apps can be updated (there are usually plenty). In each box they have the update icon, the app icon, and brief description of what the update is doing. Below this description Apple shows you the version number to which you are updating. It is usually two or three digits separated by periods. We’ve seen this format all over our screens for years. It is called Semantic Versioning. It is a standardized system of “using numbers to name a software component version” (Lauret, 2019). The number consists of three parts. Each part corresponds with the type of update, Major, Minor and Patch.

When we first release a brand new API or App we would label it 1.0.0. We quickly see that we’ve got a few bugs in our code. Something as simple as a typo. We fix this and release version 1.0.1. That last number is incremented anytime our app gets ‘patched’. A patch is any update where you are simply fixing broken functionality. It cannot add new functionality, or create any changes to the current functionality. Everything in version 1.0.1 is backwards compatible. Generally, patch versions are released to fix bugs quickly between minor releases. (WebDev, 2020).

The second column in our semantic versioning number model represents minor versions. Let’s say we released our app, and released a couple patches to fix any bugs in our release. Now we want to introduce new functionality to our app. This would come in the form of a minor patch. We would release version 1.1.0. Our patch number resets when we increment our minor number. In order for an update to be minor, all new functionality must be backwards compatible with the older versions. Minor updates won’t break the old version, just enhance it.

Now, let’s say our app is growing, we want it to do new stuff, and in so doing, we have to change how we are doing the original stuff. For example, if you’re a bank, and want to start doing international banking. Before, you never had to worry about currency and exchange rates, but as you go international, you will. You need to change how your account data is store to include currency. This is a major change, and would involve introducing an update that would crash the old version of the app by “adding a new mandatory parameter” (Lauret, 2019). This release would be version 2.0.0. Here both the minor and the patch numbers reset to zero. It is important to know that it doesn’t matter how big or how small the breaking change, if the update breaks the old version it is considered major. (WebDev, 2020)

Semantic versioning is a well-known and widely used system to let developers know what types of changes have been made in an update. The key thing to remember, especially for API development is “Breaking” vs “Non-Breaking”. Most users might not notice new features and bug fixes, but everyone will know when the app stops working. When doing a major update it is important to inform your users and possibly even support the old version for some time while encouraging users to transition.

SOURCES

Lauret, A. (2019). *The design of web apis*. Manning Publications Co.

WebDev Simplified. (2020, January 13). Semantic Versioning Explained. https://blog.webdevsimplified.com/2020-01/semantic-versioning/