Apple Watch Development Tips and Tricks by Jeff Kelley, author of *Developing for Apple Watch, Second Edition*

Perform calculations as lazily as possible.
Use the init() and awakeWithContext(_:) methods to do one-time setup tasks, then defer *all* expensive initialization tasks to willActivate() and didAppear(). If possible, all of the networking your watch app relies on should already have finished by the time the user opens the watch app. Make liberal use of the WatchConnectivity framework to transfer data to the watch from the iPhone; any time you reload your model layer, for instance, transfer the updated data to the watch so it can be processed in the background.

Test your app on both watch sizes.
Use Xcode's built-in Preview mode in the Assistant Editor to see your interfaces in both sizes. When text just won't fit—especially when translated into other languages—use the "Minimum Font Scale" setting on your labels to allow the text to shrink to fit. Just be sure you don't overuse the setting and make the text too small to be readable.

Take advantage of Handoff to allow your users to get to their data more quickly.
Use handoff to get from your Glance and Complication interfaces to the correct place inside of your app's UI, as well as to open the current task on the user's iPhone, iPad, or Mac to finish a task that's too complicated or time-consuming to do on the watch.

Never show a paying client the Watch Simulator screen.
To avoid questions about screen borders with the simulator, use the free Bezel tool (http://infinitapps.com/bezel/) to show the "finished product."

Consider a translation service.
Most watch apps have relatively few strings when compared to full-fledged iOS apps. Per-word pricing for translation services is extremely competitive, and translating your app allows you to reach customers who could never even *read* about your product in your native language.

Practice using your app with _*just*_ VoiceOver.
Not only will this help for users with visual impairment, but it will also force you to evaluate the structure of your user interface in an entirely new light. It will also show you where your current interface is too cumbersome and should be split into a VoiceOver-specific interface instead.

Use animation to show the user when a network request is happening.
And be prepared to resume it. If the user is on a slow connection, it's likely the screen will go to sleep several times during the load, so you should be prepared for willActivate() and other lifecycle methods to run while you're still waiting for the request to load.

Don't create a Complication that's just an image to launch your app.
Users want great complications that bring value to their watch face screen, not your brand everywhere.

Avoid reloading an interface too often.
If you reload a table while the user is scrolling, they'll see a weird animation—or worse, lose their place. If you need to modify the table, insert and remove rows that you know have changed, even if you need to manually compare the old data and new data to make that determination.

If possible, test on a real Apple Watch and iPhone pair.
There are some issues you won't see in the simulator, especially performance issues. Test on the lowest-performance pair you have to ensure your app is fast for everybody. Test on slow Internet connections to see what your app is like for users with them.