Twitter Thread by Ryan Nystrom

Ryan Nystrom
@_ryannystrom

It makes me smile every time I notice IGListKit at work in Instagram. We put a lot of work into making and improving it. Proud to see it holding up 4 years later.

■ with some memories on how it was built:

At Instagram, product work slowed around the holidays, so we used the time to pay down tech debt. In winter 2014 I reviewed a +12k line refactor from UITableView to UICollectionView.

We recently dropped iOS 5 so we could use the new 6+ APIs without backwards compatibility.

That review taught me a lot about UICV: data sources should mutate in one place else risk out-of-sync exceptions, the first view layout may reloadData, sticky headers make good perf hard, and many more UICV tidbits

We used reloadData for everything and it worked, for the most part.

Except two things bugged the hell out of me:

1. No animated changes (duh)
2. Images could flicker and the like animation could cancel

I dug into the latter and learned that if you reloadData, every cell on screen gets reused. So even if nothing in the UI changes, you still reuse (or init) cells, bind data, set images, etc. that's a lot of work!

When you liked a post, we'd call reloadData. Since all cells are replaced: we had to track the liked row, reload, and THEN perform the like animation. ■

If something triggered another reload after the animation started, the cell cancelled the animation (thus the UI bug)

The image flickering issue was simpler:

When an image cell is reused set the background to grey, async load the image (cache or network), set the image. The async time between reuse and fetch/set from cache is where the flicker came from.

(@ryanolsonk solved the image flicker with an insanely clever “weak image cache” design. It’s a great interview question ■)

This all got me thinking: what if we only updated cells when the data changes? This general idea was gaining traction (ahem, React): bind data to views, and diff the data on change to only rebind changed data+views.

Enter UICV performBatchUpdates

Using the principle of only updating the data source in one place, I could diff the from/to data and use UICVs APIs to insert/delete/reload/move. Anything unchanged wouldn’t get touched!

So how the hell do I write a diffing algorithm? I literally had never done this.

I researched several algos: rsync, Myers, react...

I settled on Paul Heckel’s https://t.co/jpxuHNlpXg

Why?

1. It’s output matched UICV’s APIs: inserts, deletes, updates, and moves
2. There were example implementations I could actually understand

A pivotal decision was how we define identity and equality. I wanted to piggy-back off NSObject’s -hash and -isEqual: methods to spare our engineers having to write/understand diffing concepts.

@ryanolsonk convinced me I was wrong bc writing proper -hash is complex https://t.co/4PgDCykAna

Foundation hashes can collide
https://t.co/h9eVHO342U

Or are too basic for diffing
https://t.co/Pp0eVFRS5e https://t.co/ZNPJnXlXa4

We settled on a hybrid API: a protocol with custom identifier and -isEqual: We later scrapped the NSObject tricks as they were easy to abuse & forget, resulting in crashes.

Fun fact: the library was originally called “IGStoryKit” but my director hated that so we renamed it. Glad we did, guess what launched 6 months later? ■

Over the next several months, @jesse_squires and I refactored probably 20k lines so that we could run our new infra alongside the old one. This let us safely A/B test performance before rolling out.

We started with single-item feeds but quickly moved to support the main feed. Main feed was scary (we were starting to run ads ■) but if we could replace feed, then the rest of the app would fall in line.

@ocrickard and I spent a couple months investigating crashes and tuning performance. I learned a lot about ObjC++ during this time (unordered_map was waaaay faster than NSDictionary for our diffing).

We ended up filing 4 UICV radars from all this investigation! Some have been closed

https://t.co/BACvyI2l08+

When everything was fixed, we launched. I spent a year advocating the eng org help refactor and deprecate the old infra. It came down to the profile view. I rolled up my sleeves and did it myself. It was TOUGH

With profile done, I submitted a -12k change to remove the old infra.

From the start we wanted to open source the library. It ended up being a lot of paper work and internal politics, but we did it. Forever grateful to @jesse_squires for setting the project up for success and @mikeyk for having our back.

We started writing the library in late 2015, launched feed summer of 2016, and deleted the old infra in early 2017.

Throughout this the eng team 10x'd and users 4x'd, we launched tons of products (Stories!), IG opened a NY office, UICV got diffable data sources, and more.

After we launched 4y ago I calc'd our diffing algorithm executes over 40,000 times a second. I can't imagine what that is now!

Building IGListKit was a career's-worth of learning packed into a couple years.

I'm proud that it's still powering Instagram today, but I'm looking forward to learning about what replaces it.