# Scaffolding is my new hobby

John Freeman clearly and concisely details the complexities of scaffolding. A timely article for me as I am a week into a new job where one of our teams responsibilities is owning our project templates.

https://thejohnfreeman.com/blog/2019/05/02/why-is-it-so-hard-to-write-a-scaffolding-tool/

I have a lot to chew on and learn. The conclusion seems roughly correct to introduce composability as a way to reduce complexity and sand down edge cases. It works with classes and functions, why not templates.

I've spent significant portions of my career with Django, jinja, handlebars, rails, React and redux. Some of it as a Front End Engineer. Composable, single purpose was a reliably, prudent choice. A common pattern for example; you have a part of your sight that is either a login box or a link to the users account. One option is to use one template and a conditional. Another option would to have three templates: 'login_or_account.tmpl' 'login.tmpl' and 'account.tmpl'

It costs a little more and you want to be careful introducing unneeded abstractions. My experience is that this becomes quickly worth the abstraction. Perhaps you'll have a login box on another page but instead of showing a log in box, you display a sign up form. So you can reuse 'login.tmpl'

Then it's a quick jump to changing 'login_or_account.tmpl' to a building block like 'if.tmpl'

'{{if user.is_anonymous signup.tmpl login.tmpl}}'

The use case is contrived because most templates will give you an 'if'. My claim is that this holds up in your domain as well.

And I believe this to be a general and guiding principle: avoid complexity, complected-ness. Seek to build small little beautiful parts and a robust coordinator.

Now the downside here is extra time, effort and the danger of introducing poor abstractions, which are immensely expensive. At non trivial scales, the costs and the risk of complex, leaky abstractions become acceptable, if not negligible. To fix too small of piece problem, you… put them together.

For a detailed illustration , check out [Bottoms Up vs Top Down Design (https://youtu.be/Tb823aqgX_0) by Mark Bastian (http://markbastian.github.io).

Now I am skipping over maybe the hardest part: drift and updating templates in the wild. It's low priority for the feature team (likely), they may not have a strong test suite to make it an easy release and then there's the logistics of keeping track of what humans have done. Moved a file here, added comments (which John in his blog points out), etc.

That will deserve more thought. I'm relatively comfortable the hand-waving simplification gets us close enough. Spending some time with this **essential talk** by Rich Hickey will illuminate the power of composability.

The conflicts and drift will be small if all the pieces are small. And that feels in reach of a good test suite, some diff tools and human resolution.