Other features of `expandChain`

- Can render multiple `meta` objects, by passing multiple arguments

- Complex graphs of meta-reactive dependencies are automatically turned into linear code, in the correct order; each dependency object is inserted above the *first* object that needed it

- Specify which `metaReactive` objects have their values printed to the console/report (versus just being invisibly assigned)

- Easily split code into distinct code chunks, so you can divide your app's logic among multiple R Markdown code chunks

Other features of `expandChain`

- Can render multiple `meta` objects, by passing multiple arguments
- Complex graphs of meta-reactive dependencies are automatically turned into linear code, in the correct order; each dependency object is inserted above the *first* object that needed it
- Specify which `metaReactive` objects have their values printed to the console/report (versus just being invisibly assigned)
- Easily split code into distinct code chunks, so you can divide your app's logic among multiple R Markdown code chunks

Using shinymeta

1. You (the app author) **identify the domain logic in your app code** so we can separate it from the reactive structure
2. Within that domain logic, you **identify references to reactive values and reactive expressions** that need to be replaced with static values and static code, respectively
3. At runtime, **choose which pieces** of domain logic to export, and in what order
4. **Present the code** to the user (in a window, as a downloadable script or report, etc.)