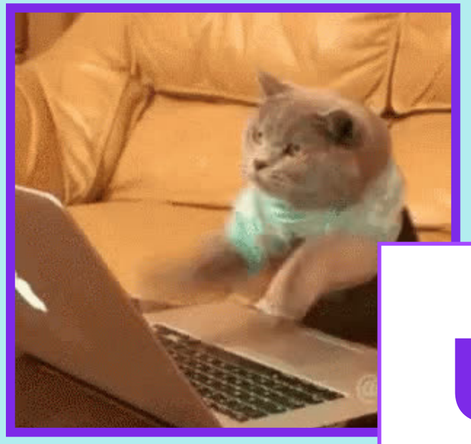
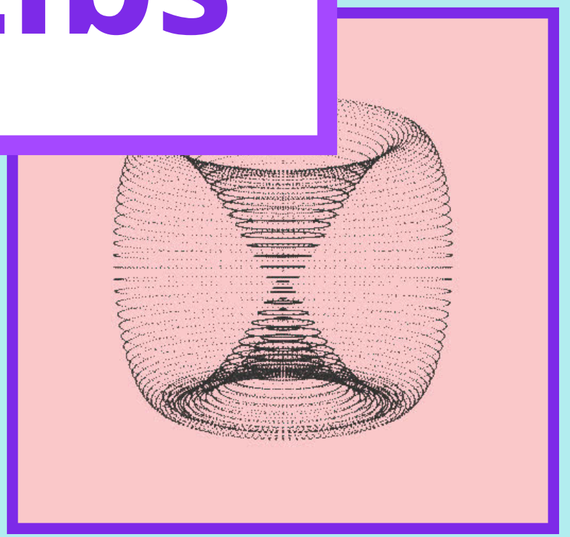


<>

# UI/UX #3

# Component Libs

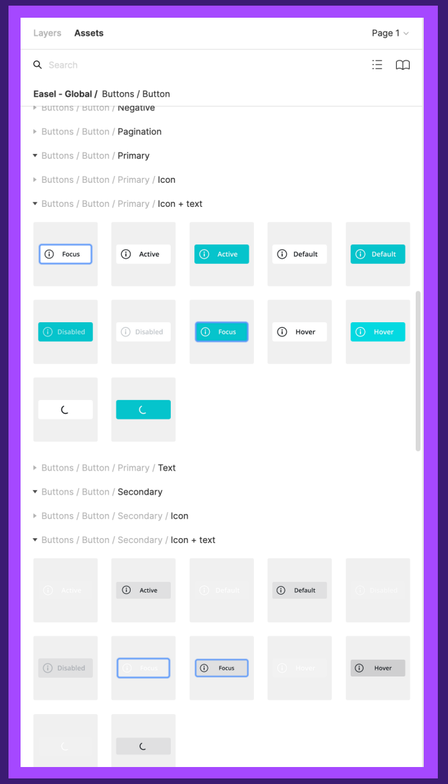


Presented by
**Sam Parkinson**

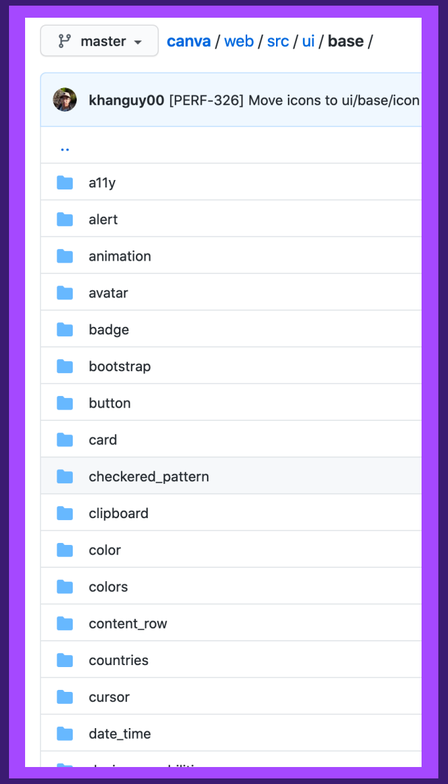
Canva

# What is a component library?

Library of bits of the UI, that are later put together to build an app.
Different things to different people:

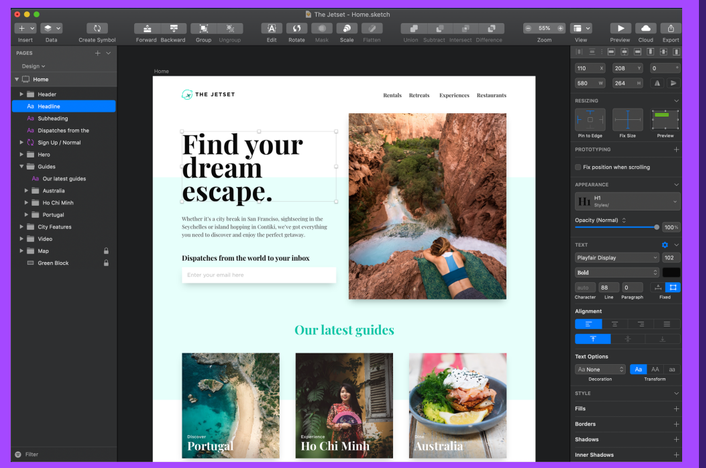



Designers see graphical elements they can re-use

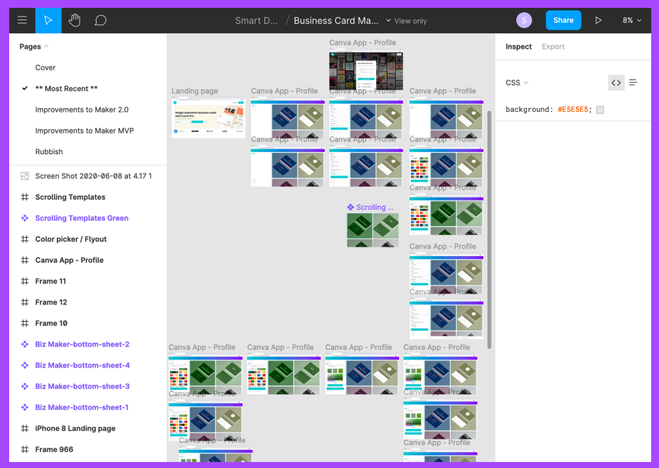



Frontenders see shared react components or css classes

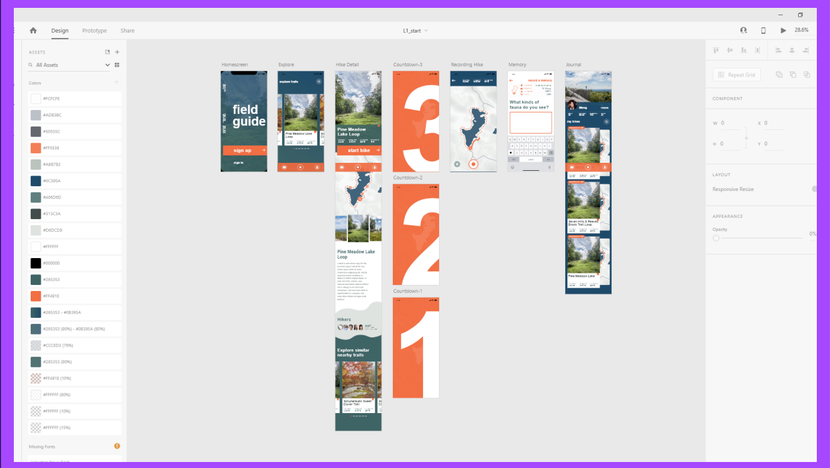
# Popular UI design applications include:




Sketch




Figma




Adobe XD

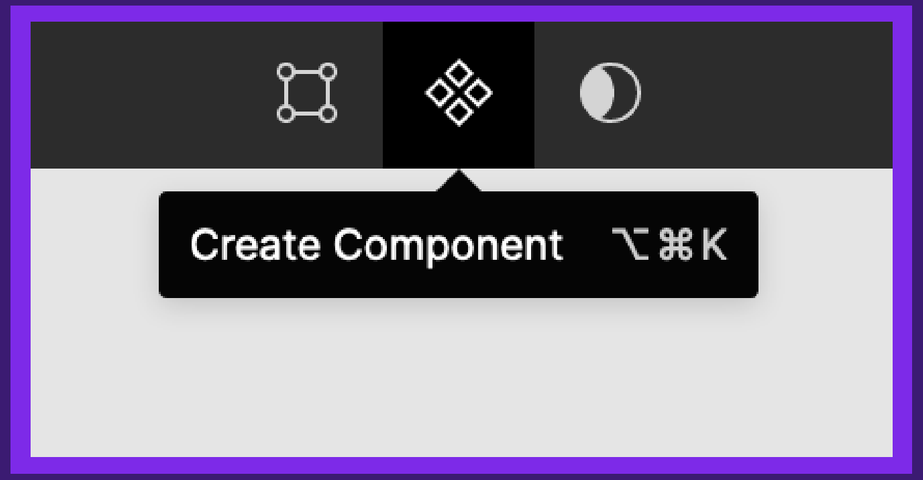
(we'll use Figma today)

# Figma "Component"

Conceptually similar to a programmer's component - it is a group of elements.

Unlike a group, when duplicated it remains linked to the original.

Component instances can be still be individually changed. Only the parts that aren't modified remained linked.



There are broadly 2 types of frontend component libraries:

**Vanilla (CSS)**

E.g. bootstrap, materialize.io

Provides CSS class for each component, and sometimes JS runs based on the classes

**React / framework**

E.g. material ui, Microsoft UI Fabric

Provides React components for each UI component

# Vanilla Component Libraries

- easy to use, including on static non-react pages
- some components require js for interactivity, can be messy to have non-react js interacting with the DOM
- lack of type safety in class names
- poor encapsulation makes upgrading components hard

e.g. bootstrap alert:

```
<div class="alert alert-primary" role="alert">
    This is a primary alert with
    <a href="#" class="alert-link">
      an example link
    </a>.
    Give it a click if you like.
</div>
```

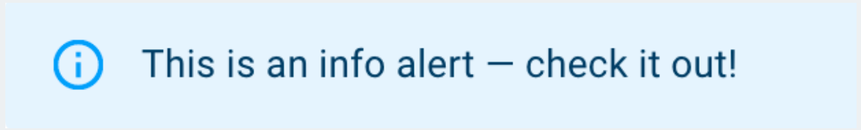
This is a primary alert with **an example link**. Give it a click if you like.

# React Component Libraries

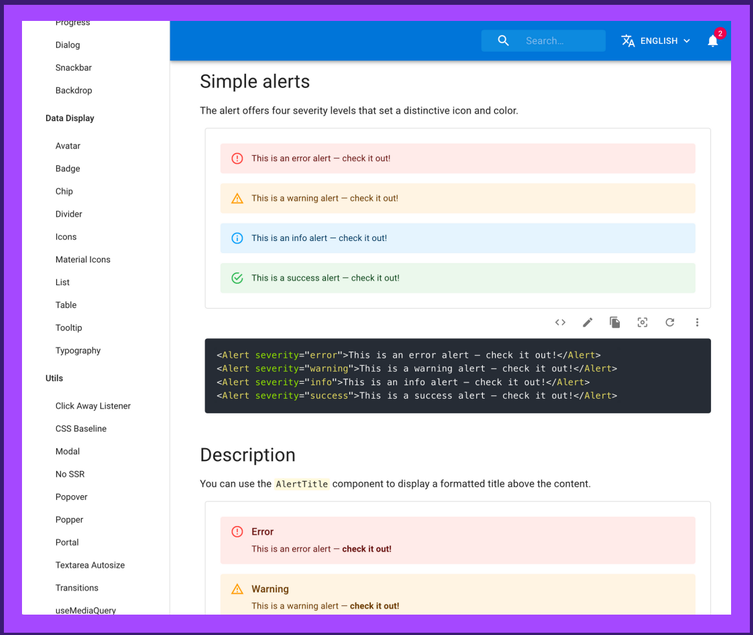
- UI system components implemented as react components
- each component defines an api -> more type safe
- each component has it's own dom node -> more encapsulated

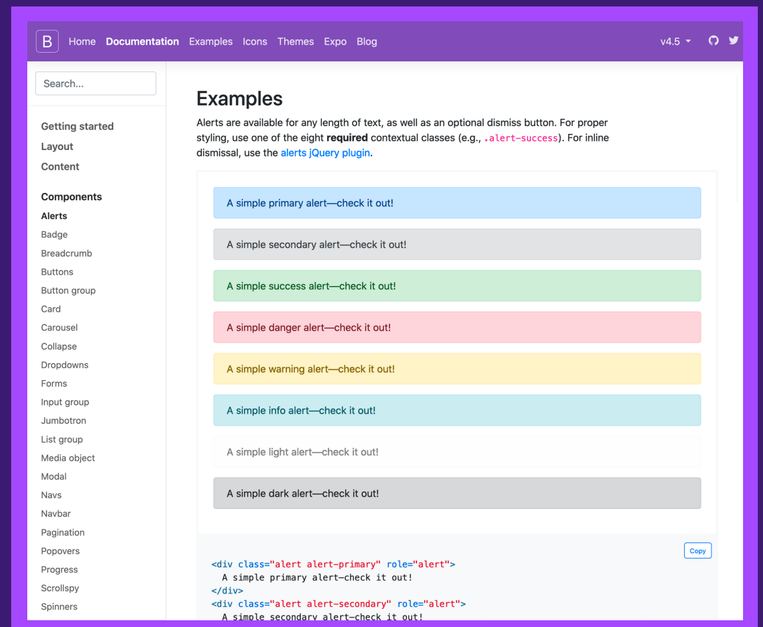
e.g. bootstrap alert:

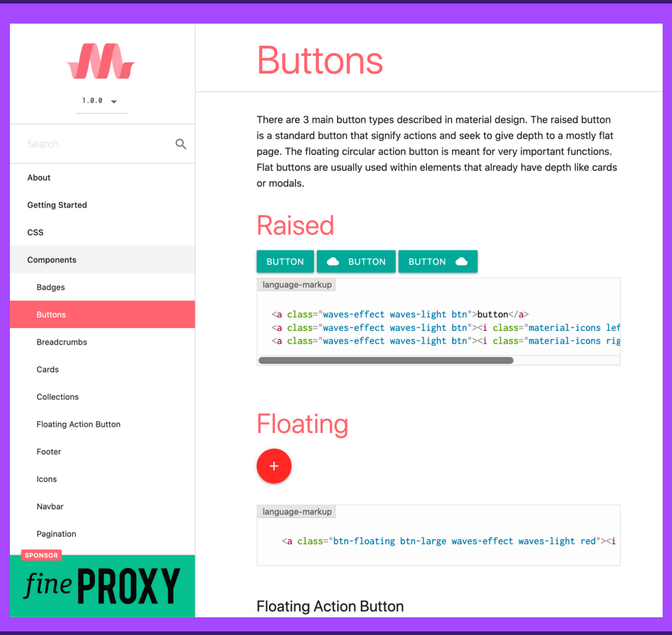
```
<Alert severity="info">
   This is an info alert — check it out!
</Alert>
```



When developing a component library, it's really helpful to have a harness or storybook page to test all variants of a component. All the open source libraries do it, for example:




We'll look at at Storybook, which is a tool for creating such a page

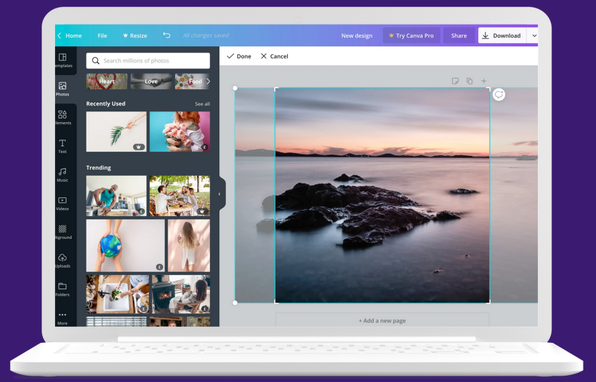
# Why create a storybook / harness?

- build components before the app they belong in
- see all states of component next to each other
- preview component without nagivating through app
- confidently change core components after you've built the app

# A Storybook creates a separate entrypoint

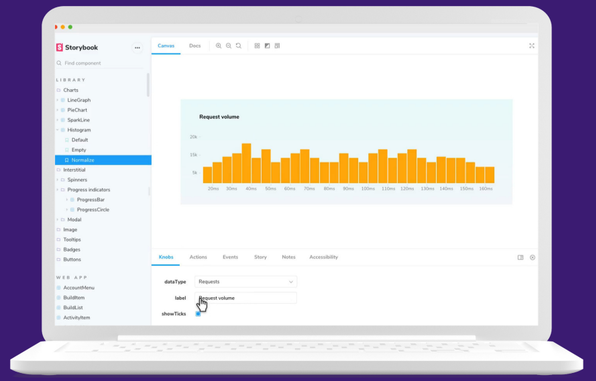
your (single)
codebase

"yarn run start"
build normal entrypoint



your app (e.g. canva)

"yarn run storybook"
build storybook entrypoint



your component library storybook

# Demo: Using Storybook

- I'm using a "create-react-app" project
- I've prepared an Alert component
- I'll demonstrate setting up storybook
- I'll demonstrate creating a story for the Alert
- See the storybook.js.org for documentation

# Extensions: storybooks for "visual regression" (visreg) testing

You can screenshot the storybook and see if the screenshots change.  Like a test, this can be automated and detected unintended changes.  You can use a tool like Percy to do this