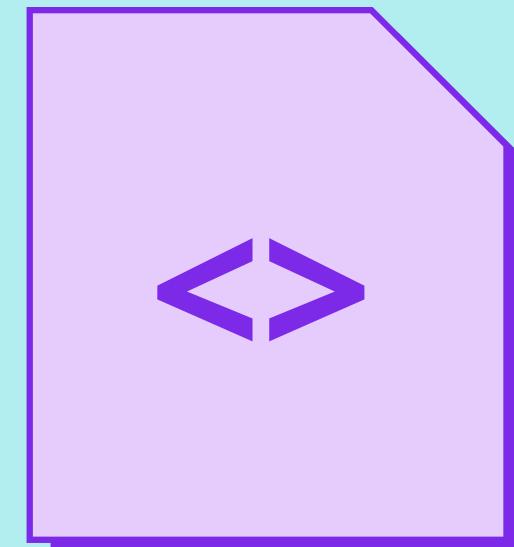
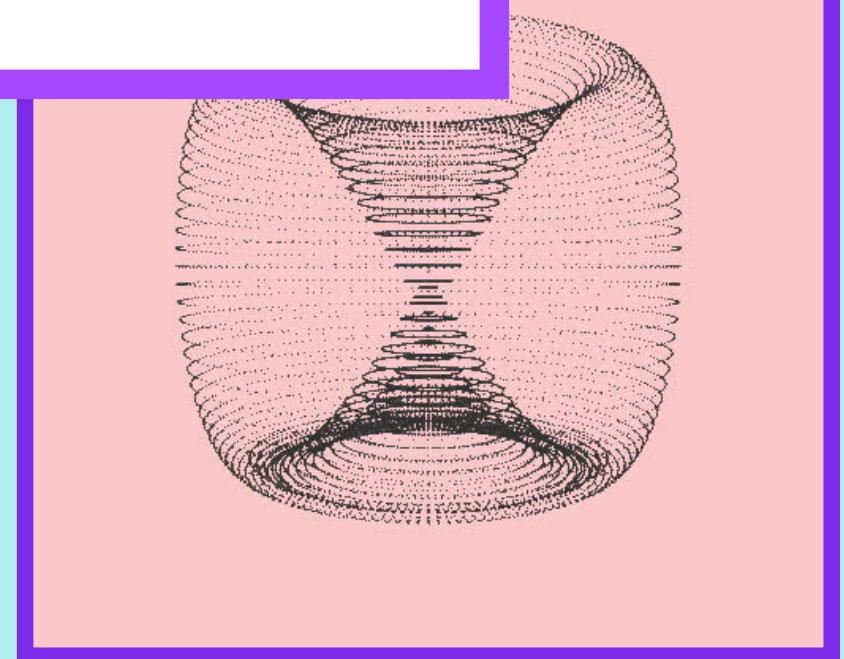


UI/UX #3



# Component Libs

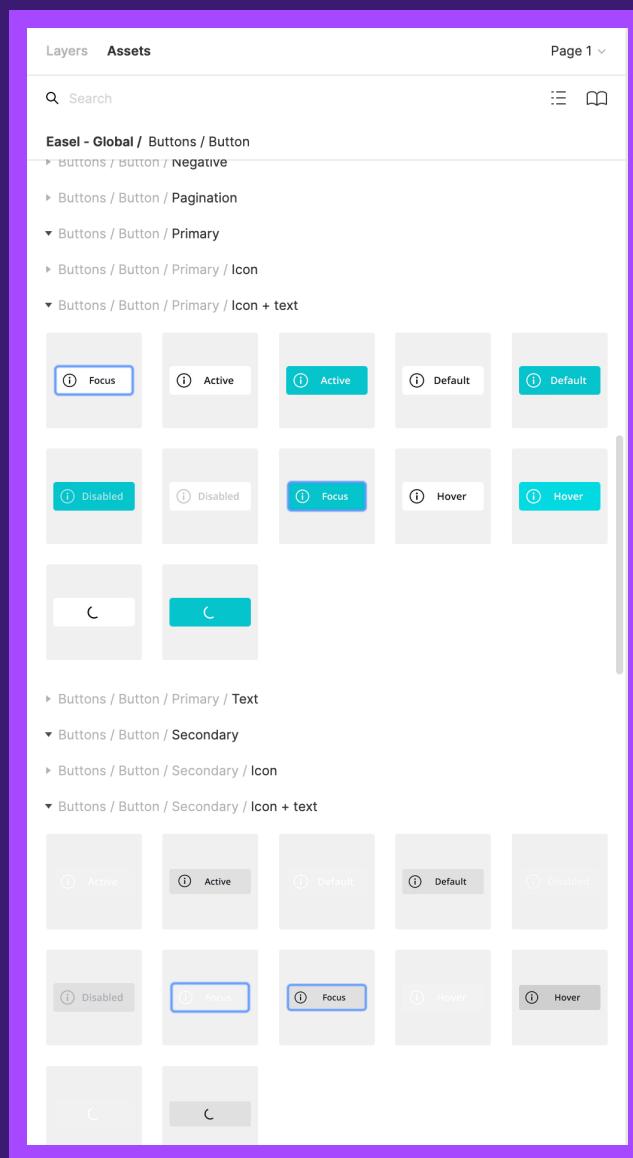
Presented by  
**Sam Parkinson**



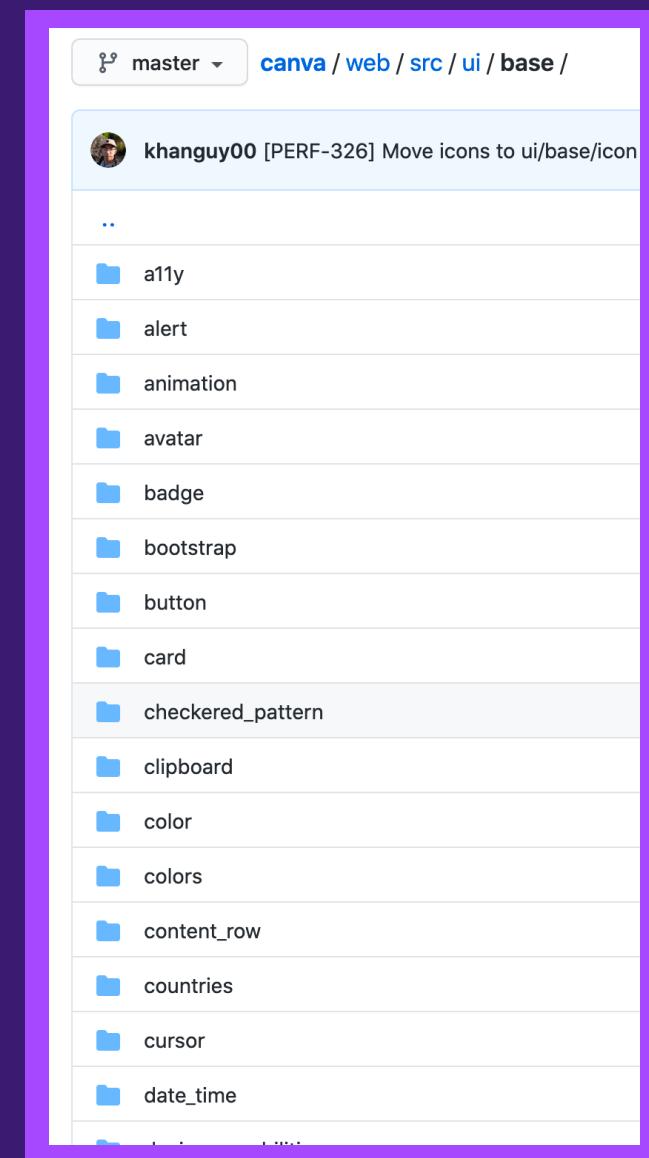
# What is a component library?

Library of bits of the UI, that are later put together to build an app.

Different things to different people:

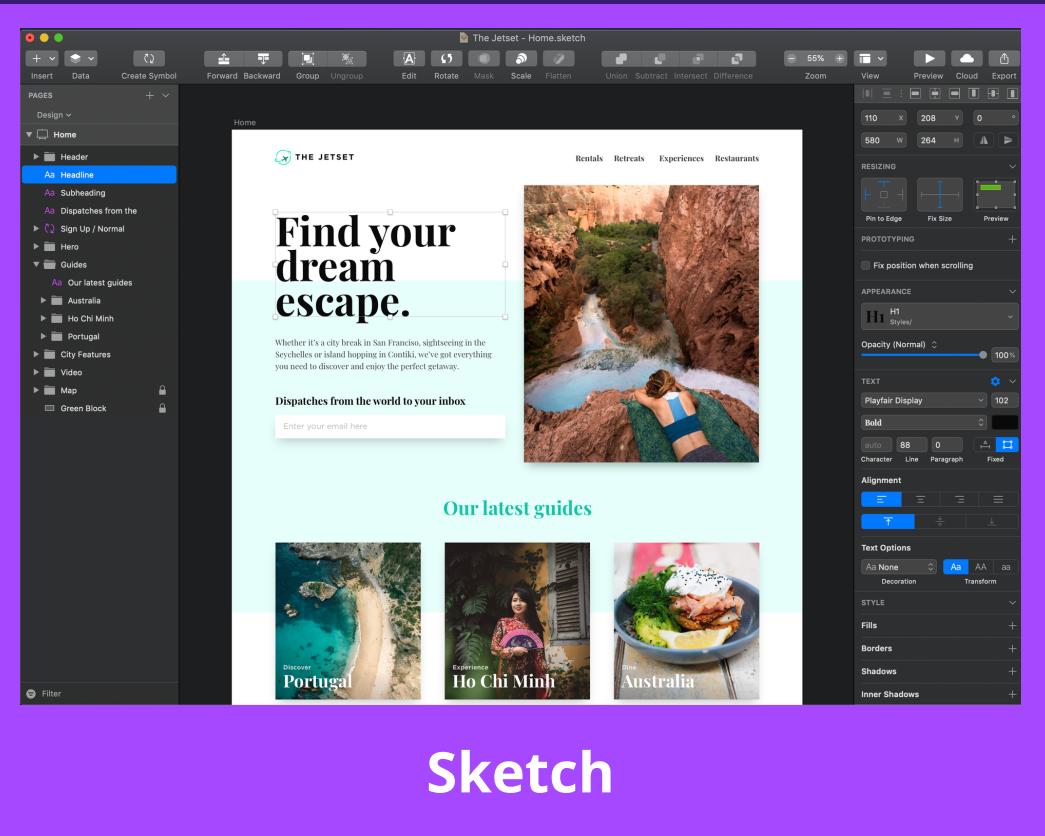


Designers see  
graphical elements  
they can re-use

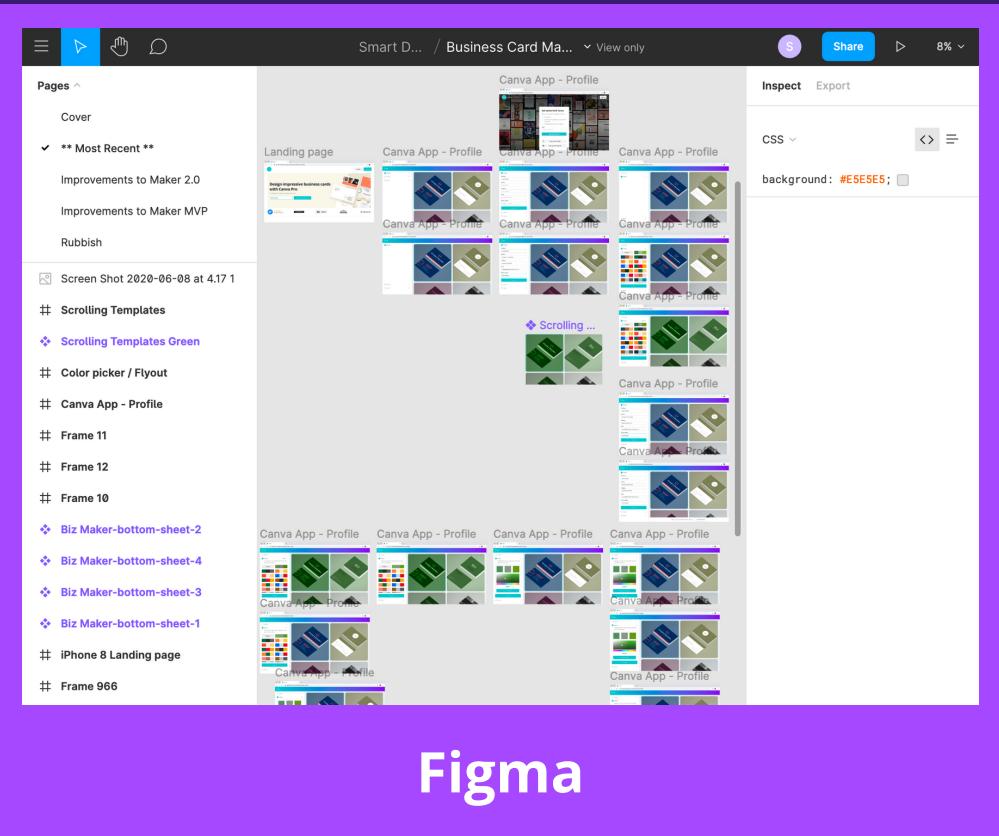


Frontenders  
see shared react  
components or  
css classes

# 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.



Create Component ⌘K



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>
```



This is an info alert – check it out!

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:

The image displays three screenshots of open-source component libraries, illustrating how they provide comprehensive documentation and testing environments for their components.

- Left Screenshot (Material-UI Components):** Shows a "Simple alerts" section. It displays four severity levels: error (red), warning (orange), info (blue), and success (green). Each alert contains a dismiss button. Below the examples is a code snippet for creating alerts with different severities. The sidebar lists various components like Progress, Dialog, Snackbar, Backdrop, etc.
- Middle Screenshot (Bootstrap 4 Documentation):** Shows an "Examples" section for alerts. It displays nine alerts of different types (primary, secondary, success, danger, warning, info, light, dark) with their respective descriptions. A "language-markup" button is present at the bottom. The sidebar lists components such as Alerts, Badge, Breadcrumb, Buttons, etc.
- Right Screenshot (Material Design Components):** Shows a "Buttons" section. It displays three types of buttons: Raised, Floating, and Flat. Each type has a description and a "language-markup" button. The sidebar lists components like Badges, Breadcrumbs, Cards, Collections, etc.

We'll look 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 navigating through app
- confidently change core components after you've built the app

# A Storybook creates a separate entrypoint



## 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

