Can AI helps us to complete repetitive tasks (and make design fun again?)

Mike Kamminga

Creative Technologist / CEO in stealth-mode

About this talk

Topics List

1. Context for Today’s talk
2. AI used in other areas of design
3. One approach towards making Design System design more efficient.

1. Context

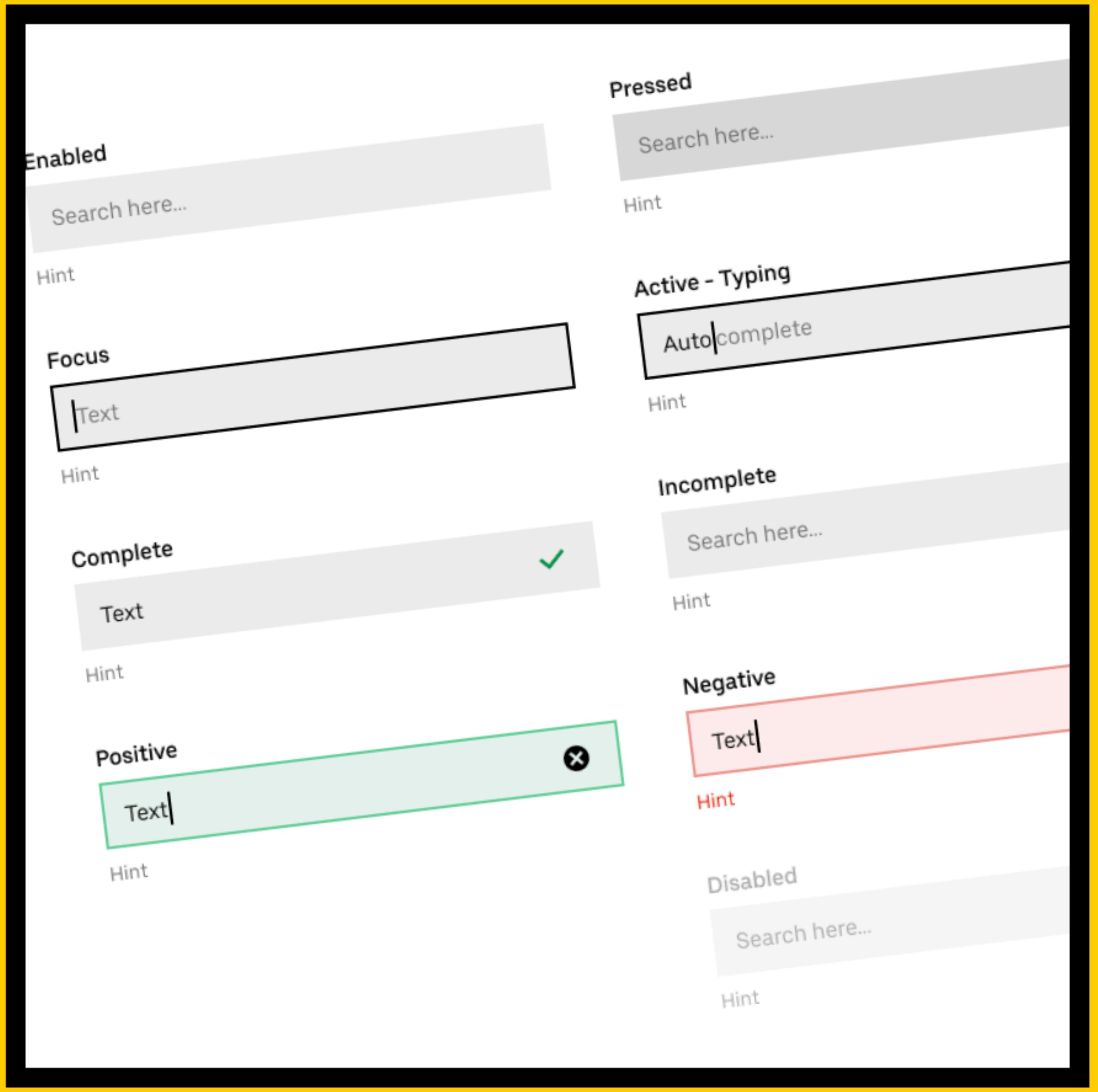
Building design systems is a laborious job with many repetitive tasks.

2. Goal

Take advantage of AI to improve the design workflow.

3. How

There isn’t one answer, but we can start by making design more efficient.



“Artificial Intelligence is the broader concept of machines being able to carry out tasks in a way that we would consider “smart”.”

Bernard Marr

Some characteristics

1. It mimics humans
2. It saves time
3. Taking manual tasks and doing it very quickly (automation)

Generative Design

Generative design is an iterative design exploration process that uses an AI-driven software program to generate a range of design solutions that meet a set of constraints.

~ image by autodesk

Real life applications

1. Architecture & Engineering

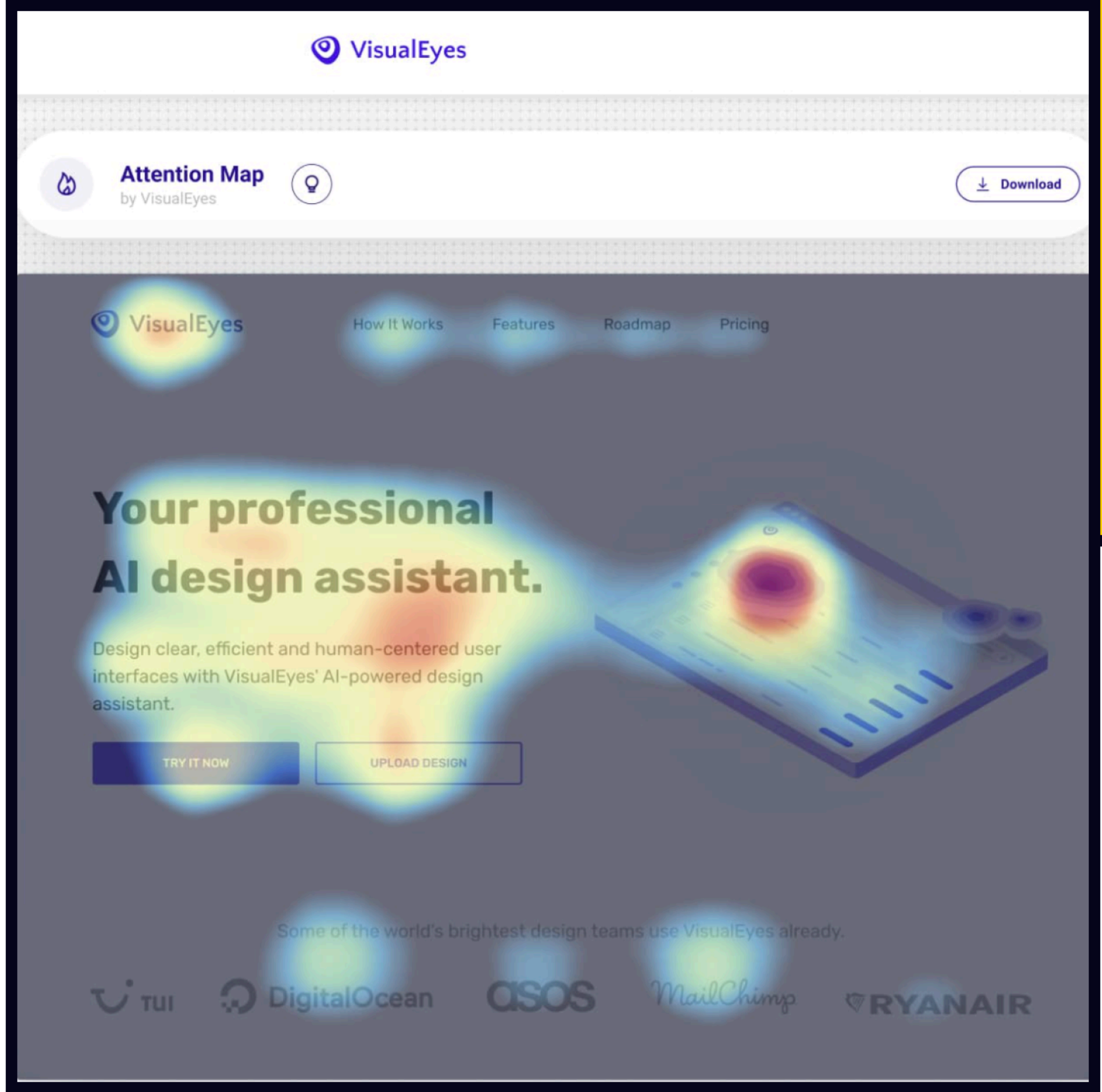
2. Automotive

3. Aerospace

3. Industrial Machinery

VisualEyes

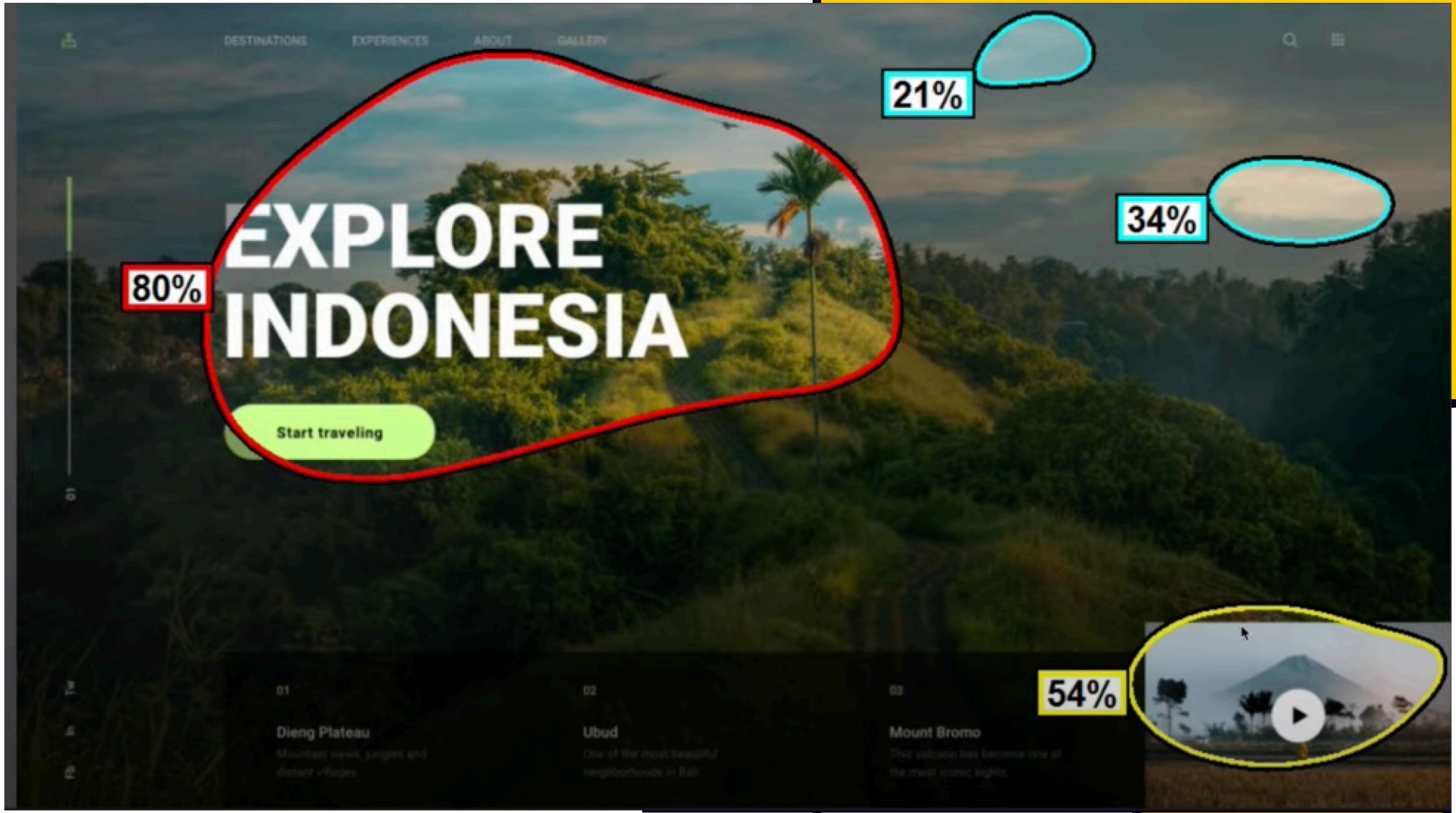
VisualEyes simulates eye-tracking studies and preference tests with a 93% accurate predictive technology

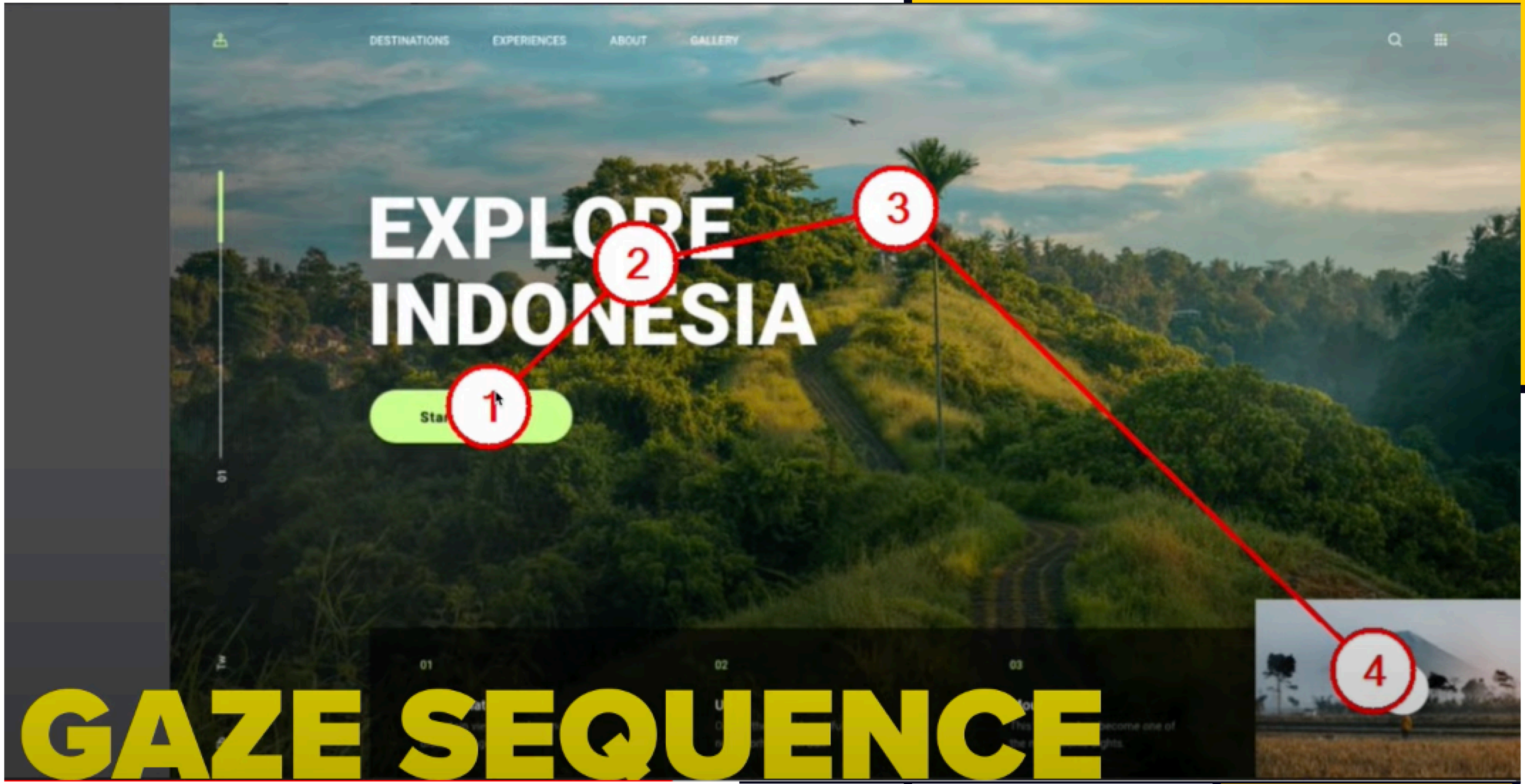


Your professional AI design assistant.

Design clear, efficient and human-centered user interfaces with VisualEyes' AI-powered design assistant.

Some of the world's brightest design teams use VisualEyes already.

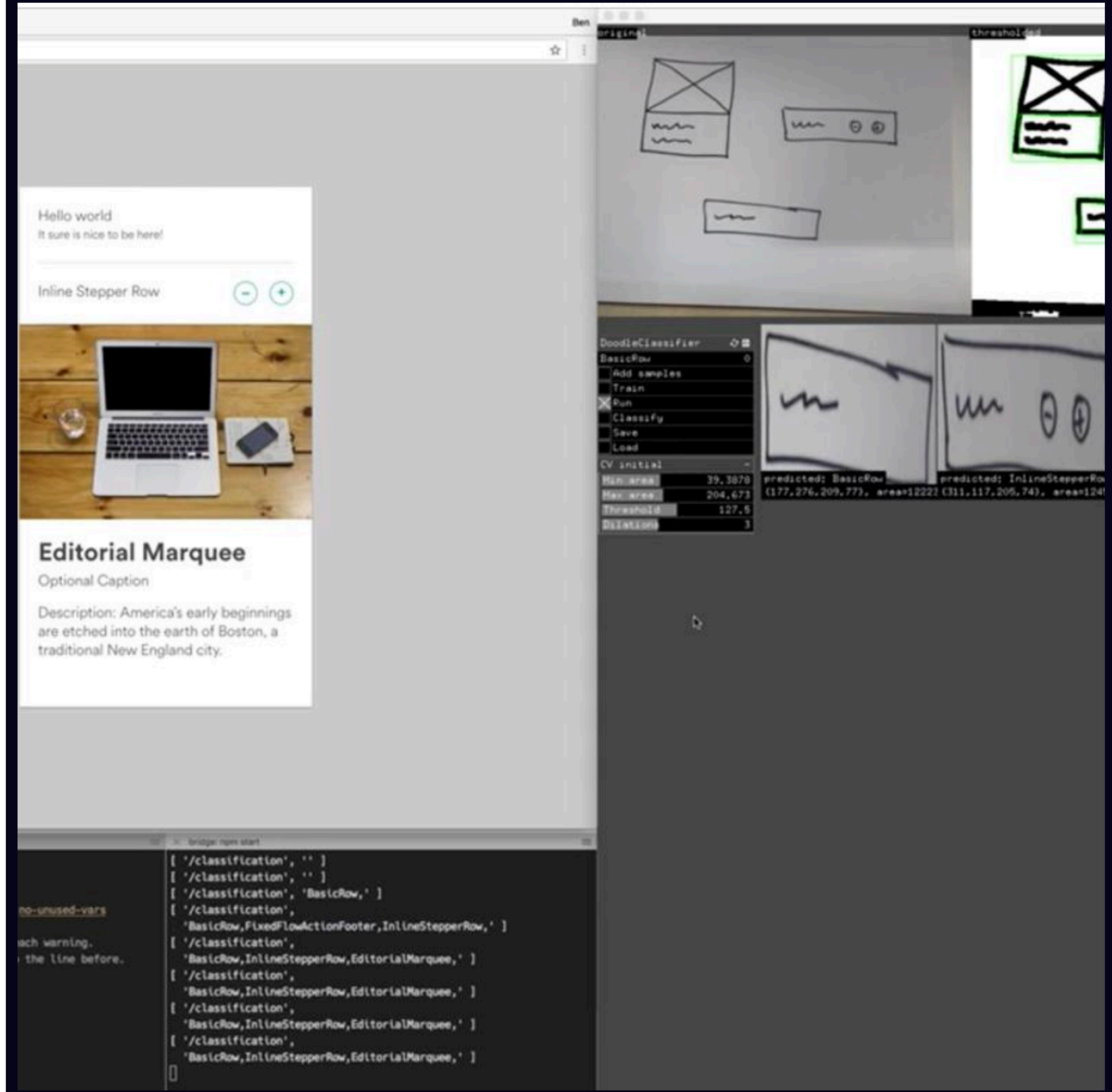

DESTINATIONS
EXPERIENCES
ABOUT
GALLERY
21%
34%
80%
EXPLORE INDONESIA
Start traveling
01
Dieng Plateau
02
Ubud
03
Mount Bromo
54%



GAZE SEQUENCE

Airbnb's wireframe to code

Converting low fidelity wireframes to code by identifying the components from the component library


Hello world
It sure is nice to be here!
Inline Stepper Row
Editorial Marquee
Optional Caption
Description: America's early beginnings are etched into the earth of Boston, a traditional New England city.
DoodleClassifier
Add samples
Train
Run
Classify
Save
Load

```
[ '/classification', '' ]
[ '/classification', '' ]
[ '/classification', 'BasicRow,' ]
[ '/classification',
  'BasicRow,FixedFlowActionFooter,InlineStepperRow,' ]
[ '/classification',
  'BasicRow,InlineStepperRow,EditorialMarquee,' ]
[ '/classification',
  'BasicRow,InlineStepperRow,EditorialMarquee,' ]
[ '/classification',
  'BasicRow,InlineStepperRow,EditorialMarquee,' ]
[ '/classification',
  'BasicRow,InlineStepperRow,EditorialMarquee,' ]
```

React App
localhost:3000
Ben
original
thresholded
merged

Hello world
It sure is nice to be here!

Editorial Marquee
Optional Caption
Description: America's early beginnings are etched into the earth of Boston, a traditional New England city.

Hello world
It sure is nice to be here!

```
DoodleClassifier
BasicRow            0
[ ] Add samples
[ ] Train
[X] Run
[ ] Classify
[ ] Save
[ ] Load
CV initial
Min area      39.3878
Max area      204.673
Threshold     127.5
Dilations     3
```

```
predicted: BasicRow
(79,388,180,90), area=7875
predicted: EditorialMarquee
(75,181,263,198), area=37034
predicted: BasicRow
(126,97,209,84), area=13267.
predicted: BasicRow
(156,13,177,65), area=7590.5
```

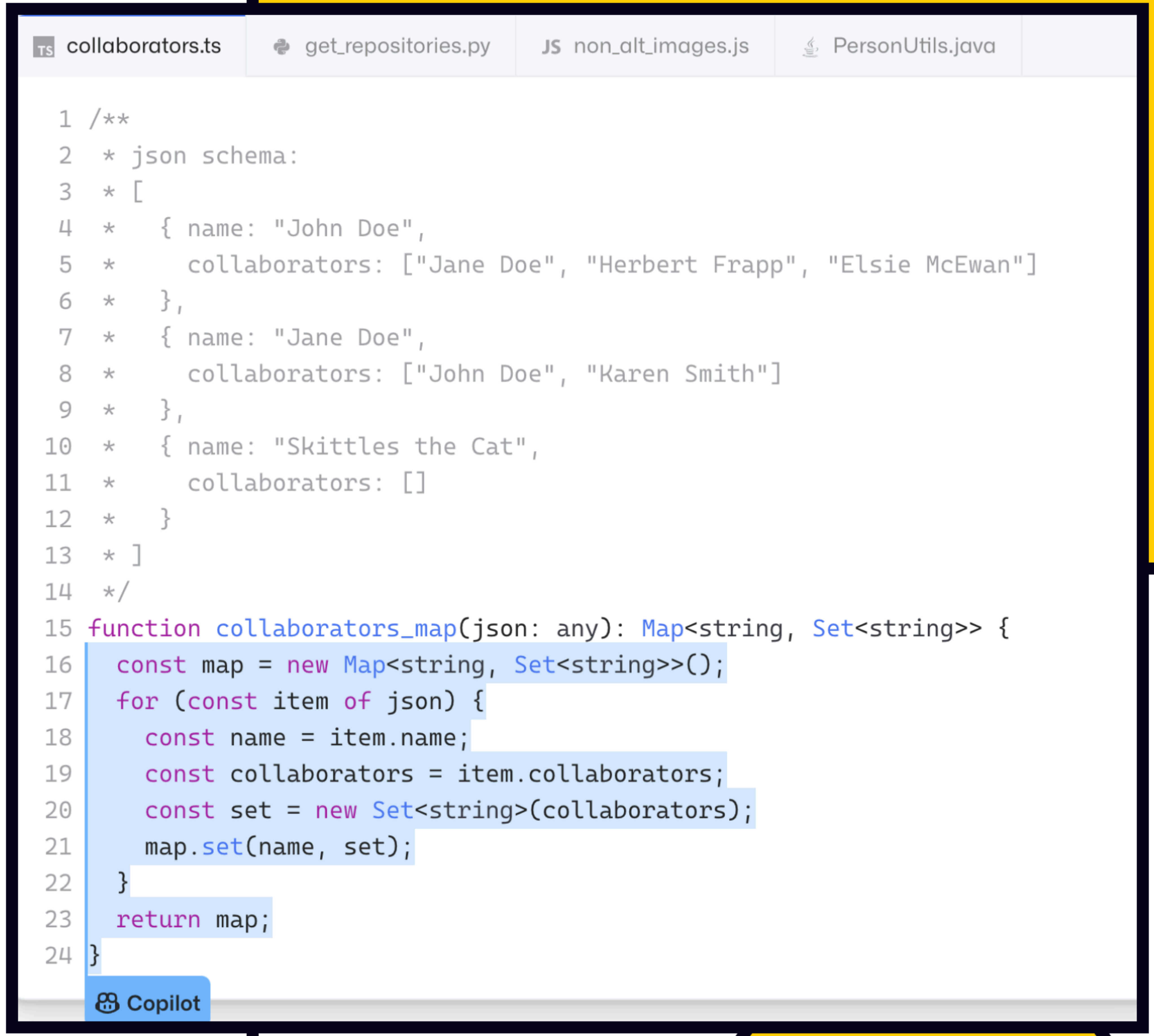
```
Compiled with warnings.

./src/App.js
  Line 2:  'logo' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

```
  'BasicRow,EditorialMarquee,BasicRow,BasicRow,' ]
[ '/classification',
  'BasicRow,EditorialMarquee,BasicRow,BasicRow,' ]
[ '/classification',
  'BasicRow,EditorialMarquee,FixedFlowActionFooter,BasicRow,' ]
[ '/classification',
  'BasicRow,EditorialMarquee,BasicRow,BasicRow,' ]
[ '/classification',
  'BasicRow,EditorialMarquee,FixedFlowActionFooter,BasicRow,' ]
[ '/classification',
  'BasicRow,EditorialMarquee,BasicRow,BasicRow,' ]
[ '/classification',
  'BasicRow,EditorialMarquee,BasicRow,BasicRow,' ]
```

Github-copilot

GitHub Copilot draws context from comments and code, and suggests individual lines and whole functions instantly.

GitHub Copilot is powered by OpenAI Codex, a new AI system created by OpenAI.



```
/**
 * json schema:
 * [
 *   { name: "John Doe",
 *     collaborators: ["Jane Doe", "Herbert Frapp", "Elsie McEwan"]
 *   },
 *   { name: "Jane Doe",
 *     collaborators: ["John Doe", "Karen Smith"]
 *   },
 *   { name: "Skittles the Cat",
 *     collaborators: []
 *   }
 * ]
 */
function collaborators_map(json: any): Map<string, Set<string>> {
  const map = new Map<string, Set<string>>();
  for (const item of json) {
    const name = item.name;
    const collaborators = item.collaborators;
    const set = new Set<string>(collaborators);
    map.set(name, set);
  }
  return map;
}
```

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      // create a div with an h1 and an input inside it
        var div = document.createElement('div');
        var h1 = document.createElement('h1');
        var input = document.createElement('input');
        div.appendChild(h1);

      // update the h1 text to say "Enter a color"

      // add 100px padding to the div

      // give the input a placeholder color

      // add in input event listener to the input

    </script>
  </body>
</html>
```

Some thoughts on the ‘How’

A use case I think we’re all familiar with: the **button**

Designing a button

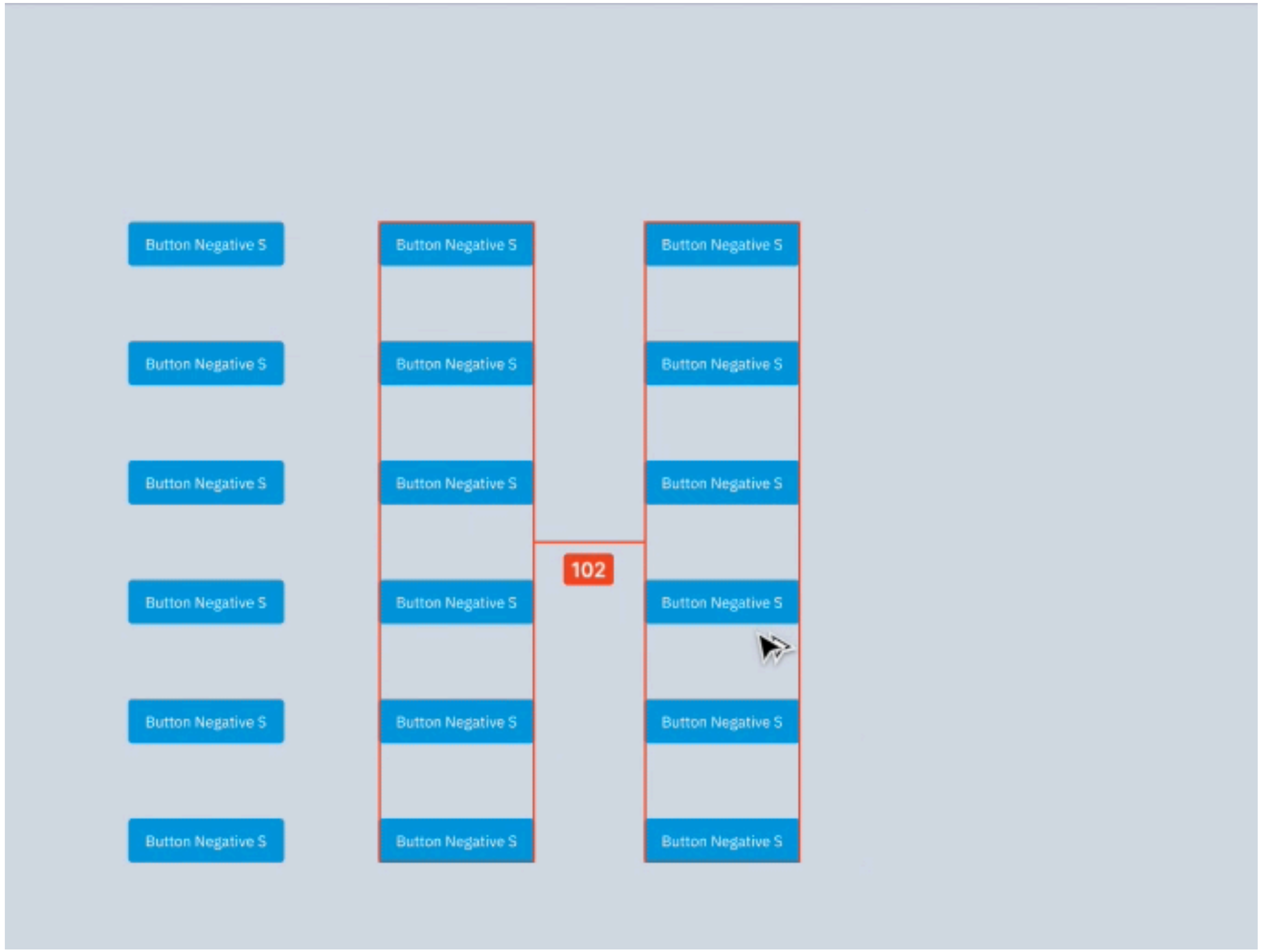
Create a base button

Duplicate

Update styles

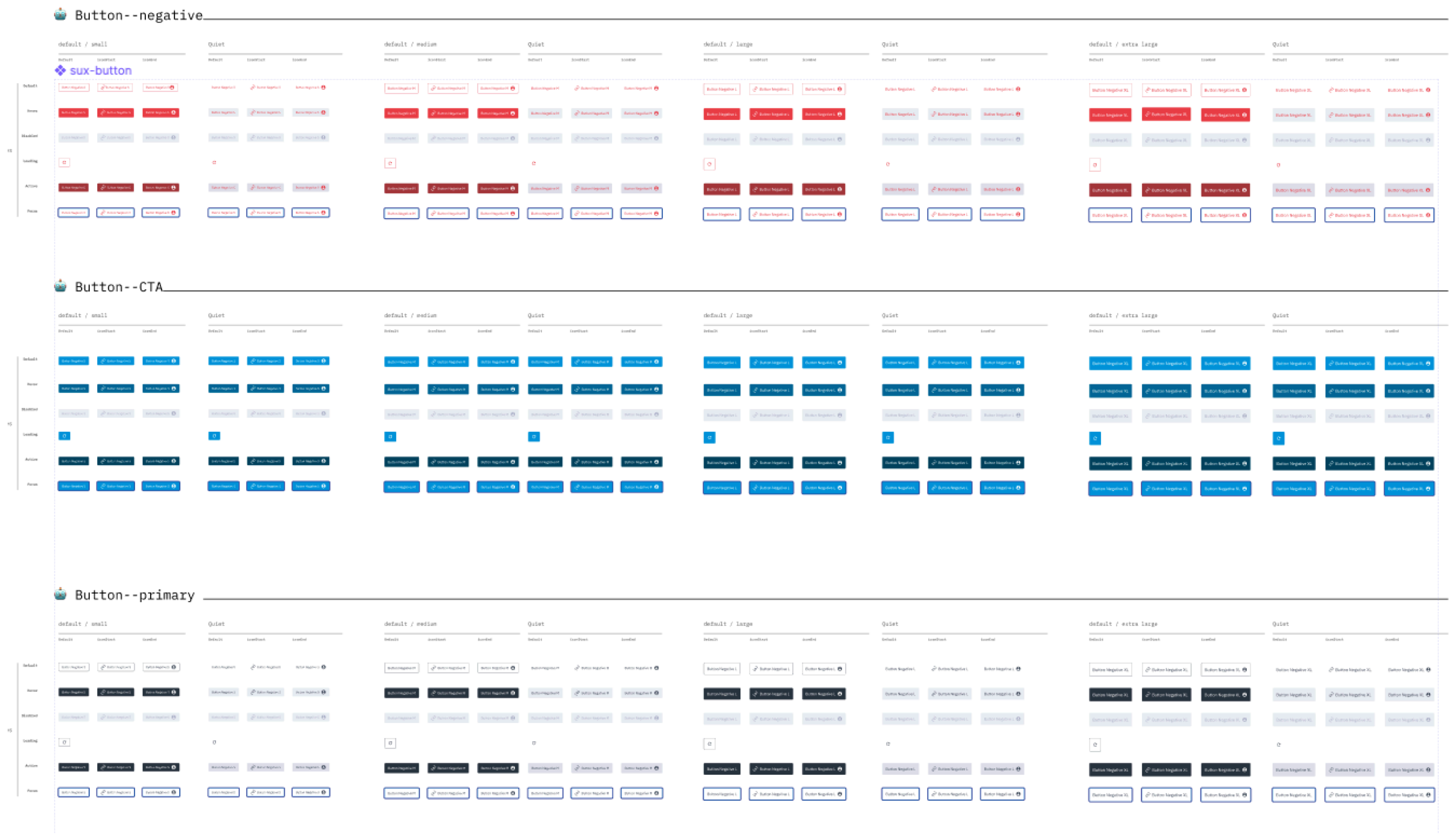
Componentise & Name

Combine as variant

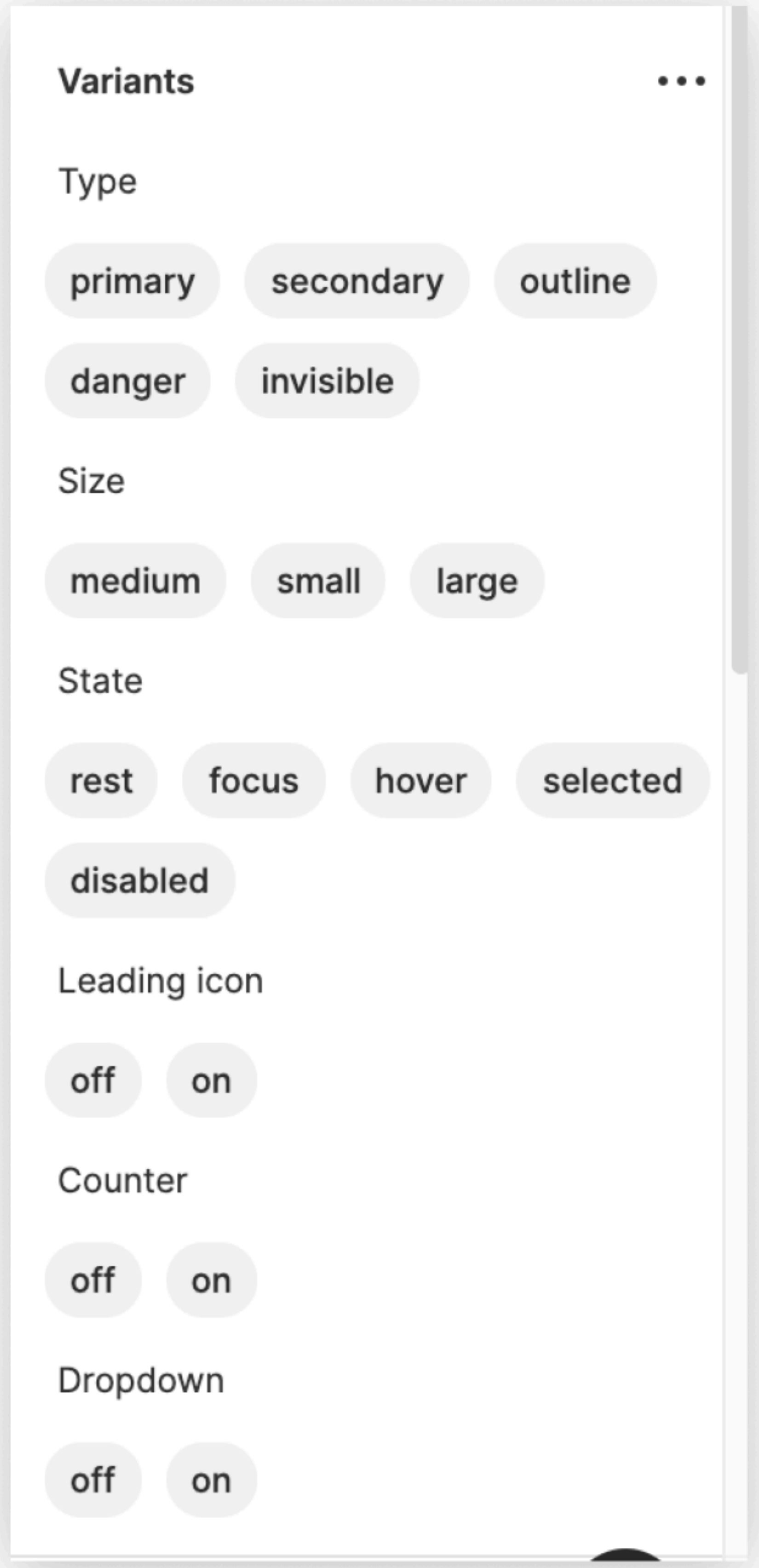


Designing a button

Once we're done, we usually end up with something that looks like this



Component API



Variants

Type
primary secondary outline
danger invisible

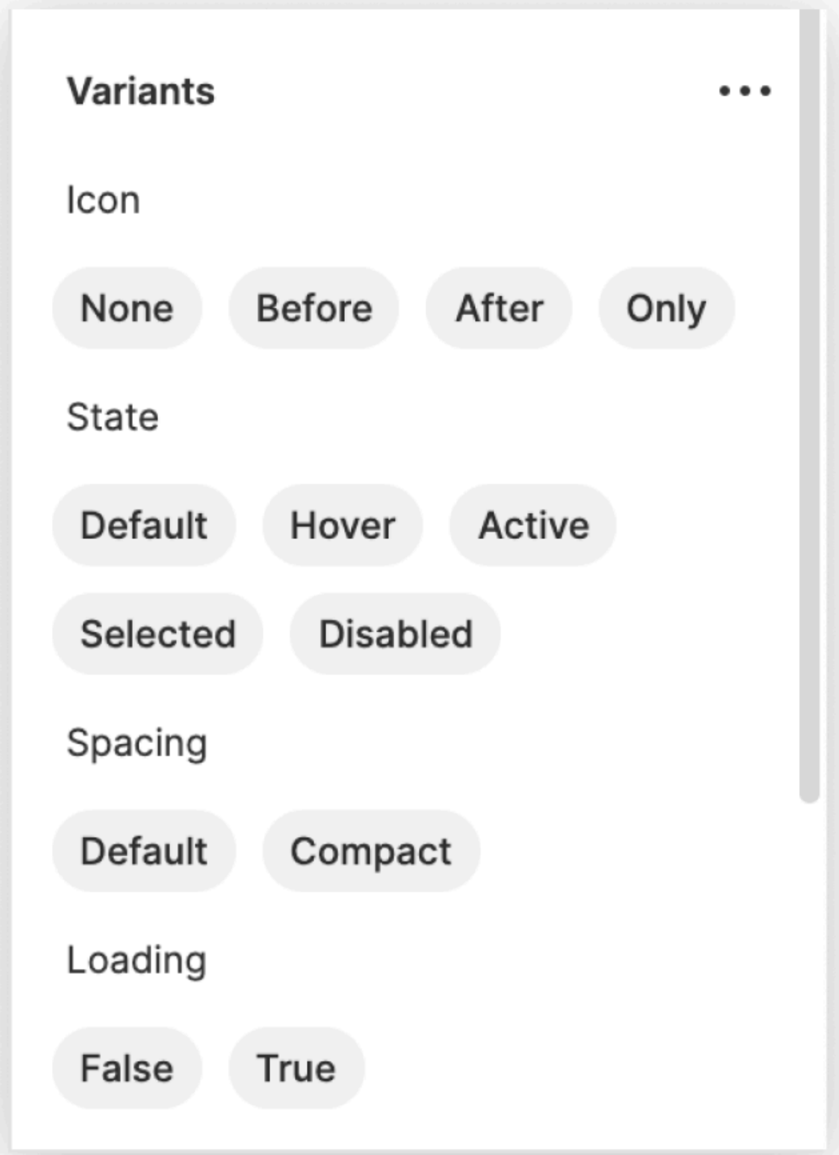
Size
medium small large

State
rest focus hover selected
disabled

Leading icon
off on

Counter
off on

Dropdown
off on



Variants

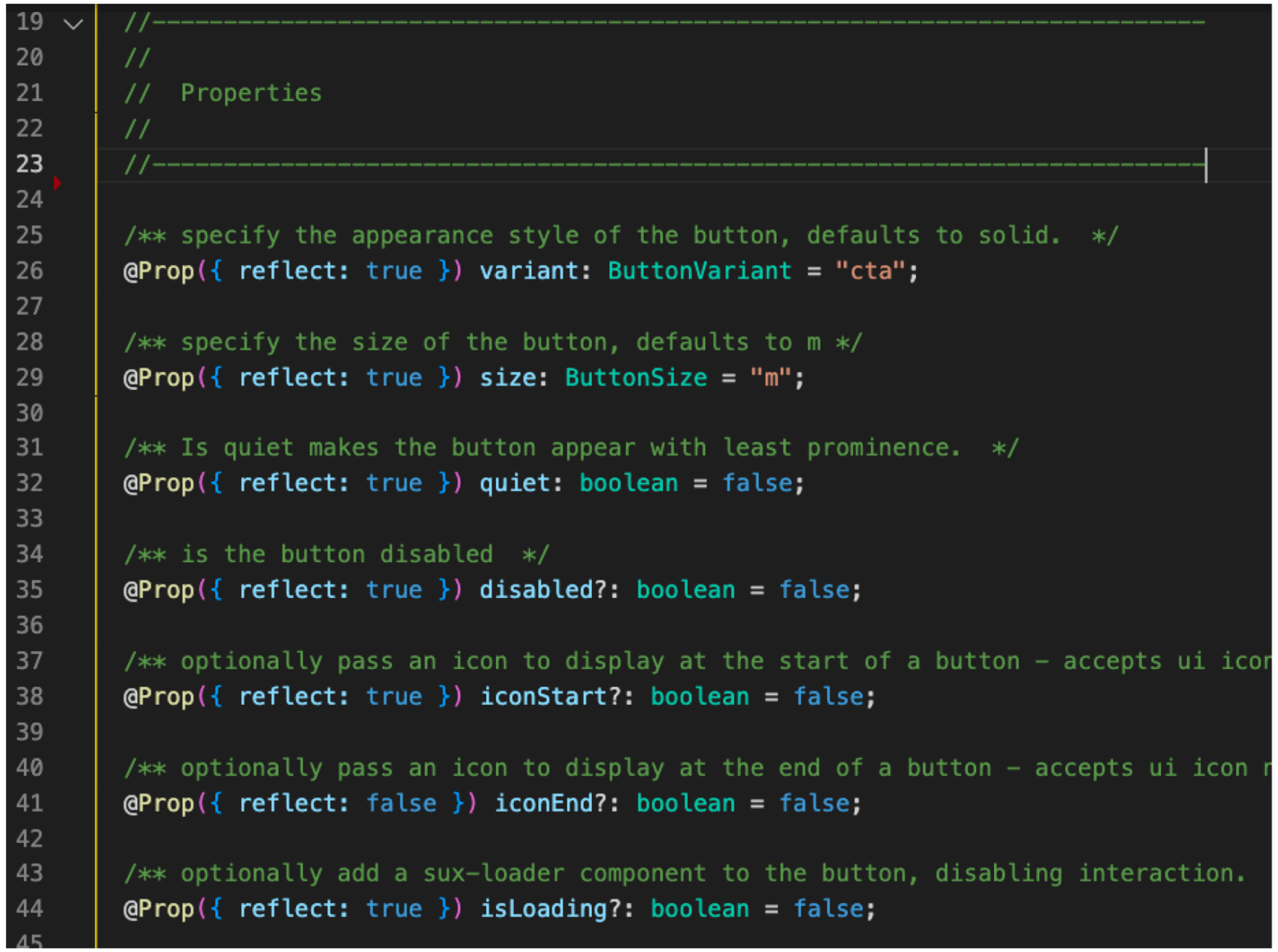
Icon
None Before After Only

State
Default Hover Active
Selected Disabled

Spacing
Default Compact

Loading
False True

Component API in Figma



```
//----------------------------------------------------------------------------
//
//  Properties
//
//----------------------------------------------------------------------------

/** specify the appearance style of the button, defaults to solid.  */
@Prop({ reflect: true }) variant: ButtonVariant = "cta";

/** specify the size of the button, defaults to m */
@Prop({ reflect: true }) size: ButtonSize = "m";

/** Is quiet makes the button appear with least prominence.  */
@Prop({ reflect: true }) quiet: boolean = false;

/** is the button disabled  */
@Prop({ reflect: true }) disabled?: boolean = false;

/** optionally pass an icon to display at the start of a button - accepts ui icon
@Prop({ reflect: true }) iconStart?: boolean = false;

/** optionally pass an icon to display at the end of a button - accepts ui icon r
@Prop({ reflect: false }) iconEnd?: boolean = false;

/** optionally add a sux-loader component to the button, disabling interaction.
@Prop({ reflect: true }) isLoading?: boolean = false;
```

Component API in Code

This is great, but...

Changes & new variants
are a big headache!

How can we make changes more manageable?

Use design tokens in Design

In Figma:
Using Figma Tokens

Add granular tokens

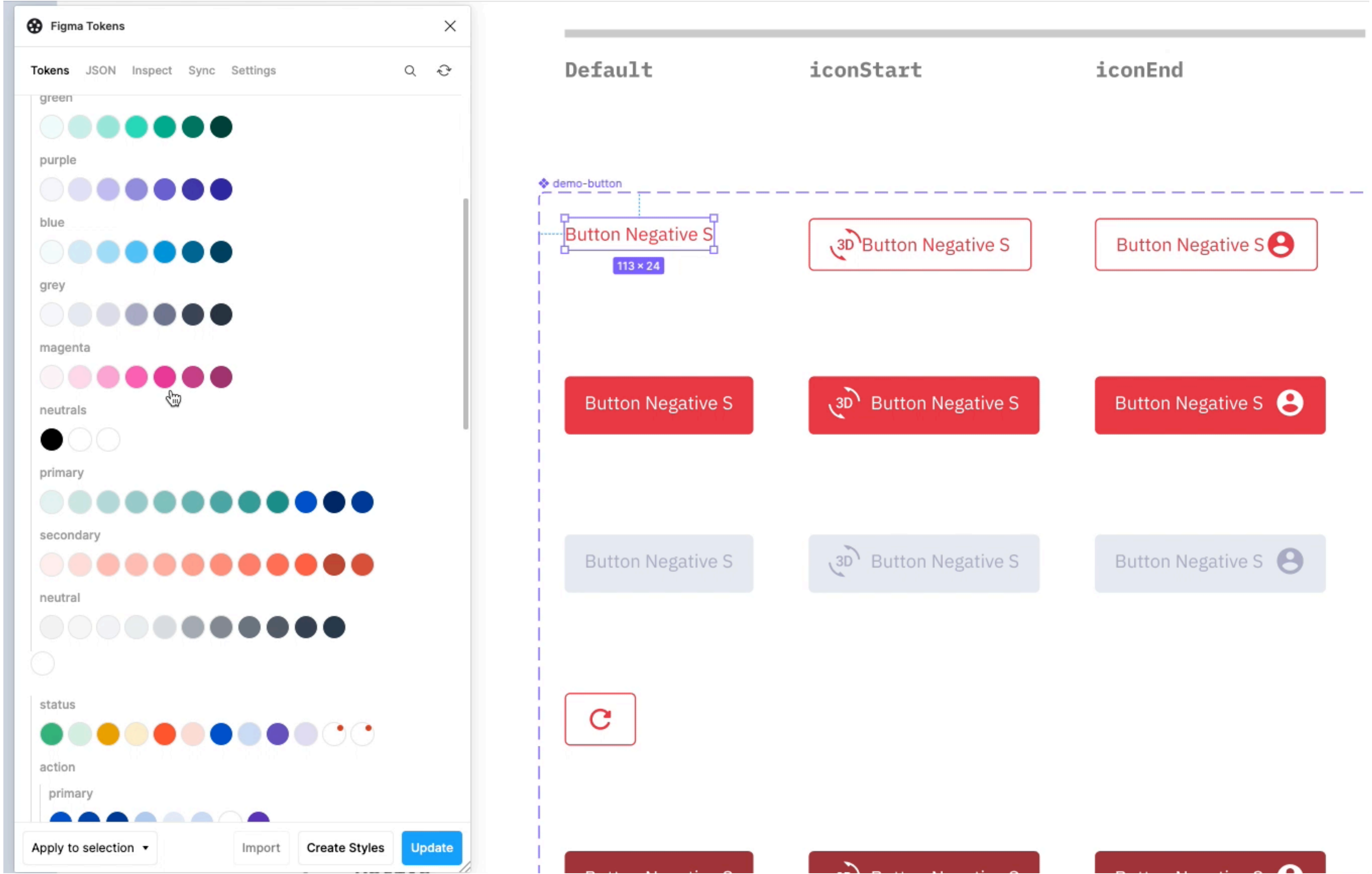
Good example:
Adobe Spectrum

The Good?

Changes become a breeze

The Bad?

Setting up granular tokens takes a lot of time


Figma Tokens
Tokens
JSON
Inspect
Sync
Settings
green
purple
blue
grey
magenta
neutrals
primary
secondary
neutral
status
action
primary
Apply to selection
Import
Create Styles
Update
Default
iconStart
iconEnd
demo-button
Button Negative S
113 × 24
3D

Code File Edit Selection View Go Run Terminal Window Help
Fri 7 Jan 13:58
sux-button.scss — substrate-all (Workspace)
sux-button.tsx
sux-button.scss
components > src > components > sux-button > sux-button.tsx > SuxButton
components > src > components > sux-button > sux-button.scss > .sux-button

```
//
//  Properties
//
//----------------------------------------------------------------------------
/** Applies to the aria-label attribute on the button or hyperlink */
@Prop({ reflect: true }) label: string = 'Button label';

/** The name attribute to apply to the button  */
@Prop({ reflect: true }) name?: string = 'button-';

/** The type attribute to apply to the button  */
@Prop({ mutable: true }) type?: string;

/** specify the appearance style of the button, defaults to solid.  */
@Prop({ reflect: true }) variant: ButtonVariant = "cta";

/** specify the size of the button, defaults to m */
@Prop({ reflect: true }) size: ButtonSize = "m";

/** Is quiet makes the button appear with least prominence.  */
@Prop({ reflect: true }) quiet: boolean = false;

/** is the button disabled  */
@Prop({ reflect: true }) disabled?: boolean = false;

/** optionally pass an icon to display at the start of a button - accepts ui icon r
@Prop({ reflect: true }) iconStart?: boolean = false;

/** optionally pass an icon to display at the end of a button - accepts ui icon nam
@Prop({ reflect: false }) iconEnd?: boolean = false;

/** optionally add a sux-loader component to the button, disabling interaction.  */
@Prop({ reflect: true }) isLoading?: boolean = false;
```

```
    font-size: var(--button-large-text-size);
    padding-left: var(--button-large-padding-left);
    padding-right: var(--button-large-padding-right);
    padding-top: var(--button-large-padding-top);
    padding-bottom: var(--button-large-padding-bottom);
  }
  &--sizeXL {
    font-size: var(--button-extra-large-text-size);
    padding-left: var(--button-extra-large-padding-left);
    padding-right: var(--button-extra-large-padding-right);
    padding-top: var(--button-extra-large-padding-top);
    padding-bottom: var(--button-extra-large-padding-bottom);
  Next (⌥]) Previous (⌥[) Accept (Tab) Open GitHub Copilot (^Enter)
  &--sizeXXL {     You, seconds ago • Uncommitted changes
    font-size: var(--button-extra-extra-large-text-size);
    padding-left: var(--button-extra-extra-large-padding-left);
    padding-right: var(--button-extra-extra-large-padding-right);
    padding-top: var(--button-extra-extra-large-padding-top);
    padding-bottom: var(--button-extra-extra-large-padding-bottom);
  }
  &--cta {
    background-color: var(--button-cta-default-background-color);
    color: var(--button-cta-default-text-color);
    border-color: var(--button-cta-default-border-color);
    &:hover {
      background-color: var(--button-cta-hover-background-color);
      color: var(--button-cta-hover-text-color);
    }
  }
  &--primary {
    color: var(--button-primary-default-text-color);
    border-color: var(--button-primary-default-border-color);
    &:hover {
      background-color: var(--button-primary-hover-background-color);
      color: var(--button-primary-hover-text-color);
```

wip* 2↓ 0↑ 0 0
You, seconds ago Ln 70, Col 3 Spaces: 2 UTF-8 LF SCSS Prettier

In short

1. Define properties (component API)
2. Define visual changes for (some) props (css classes)
3. Provide some rules (what happens when a prop changes)

What if...
we turn the process around

Inputs:

- Design component API first
- Provide rules for each prop
- Suggest a visual hierarchy

Automation:

- Generate a headless component
- Automatically generate and apply tokens

NO ONE READS THIS

How could this work?

Plugin

Takes the component API inputs, as well as the desired visual structure we prefer

Automation

Generate all variants in Figma including component based design tokens for each variant

Headless component

Now we can apply tokens (styles) easy, similar to how a developer would do this

AI aided

There are still repetitive tasks, this is where AI could help, like GitHub copilot.

A possible workflow

1. Design a template component, run the plugin

2. Plugin analyses the component and suggests all relevant tokens based on the component structure

3. Define rules: what tokens or layers should change when an API (variant) property changes

4. The plugin generates a fully tokenised component grid with all possible variants based on the defined props.

Next steps: AI

With this in place, I imagine AI can now add support.

For example while:

- Adding new variants
- style changes

Let's make it happen

Time to hack away!

Thank you!

Any questions? Reach out on slack!

Mike Kamminga | slack: intodesignsystems | Twitter: @mikekamminga