

GOTO **AMSTERDAM 2023**

#GOTOams



GOTO
Guide

LET US HELP YOU

Ask questions
through **the app**



also remember to rate session



Download on the
App Store



GET IT ON
Google Play

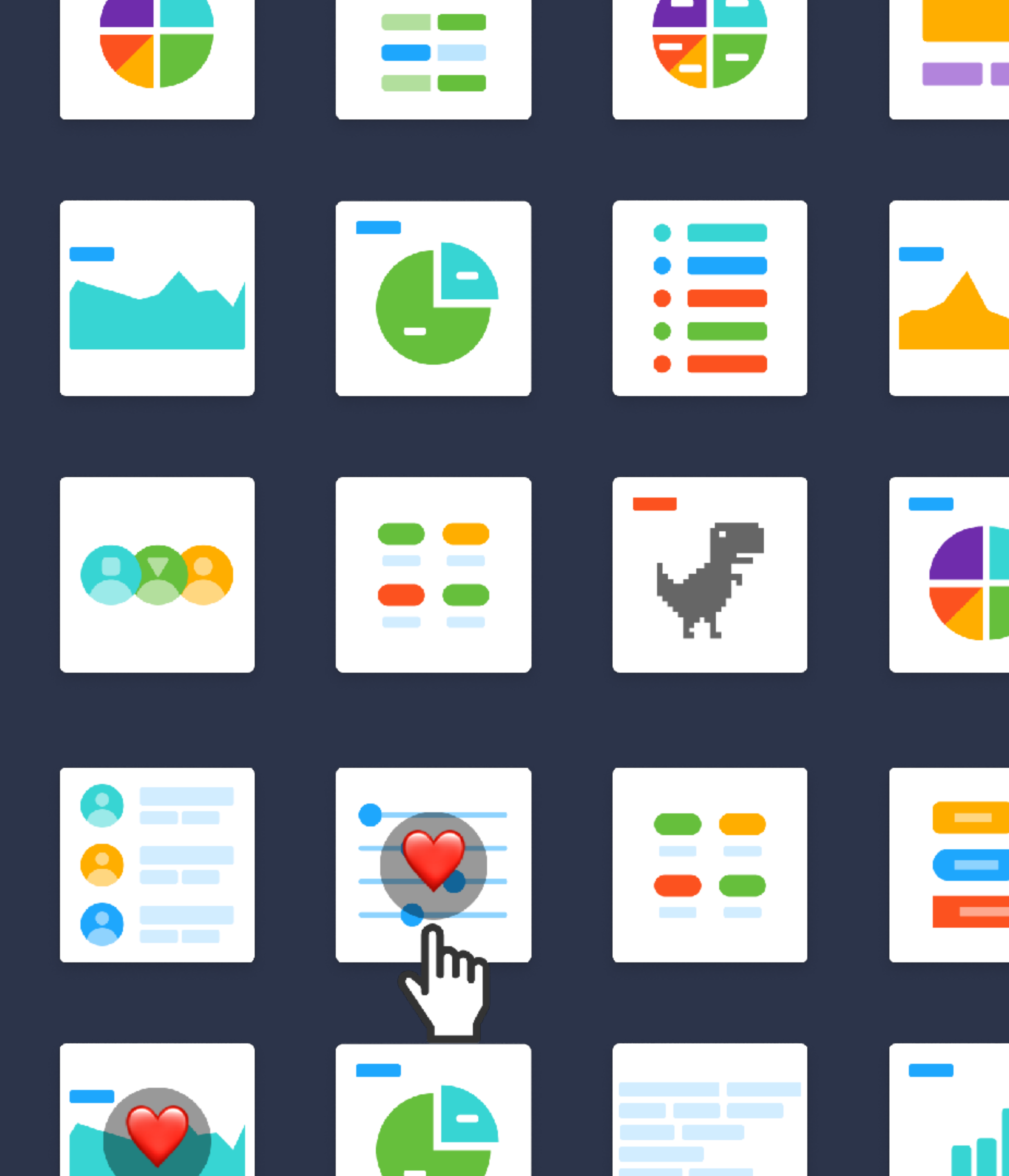
THANK YOU!

#GOTOams

State of Storybook

 Storybook |  chromatic

GOTO CONFERENCE • JUNE 2023



Do you remember working like this?



Spin up the whole platform



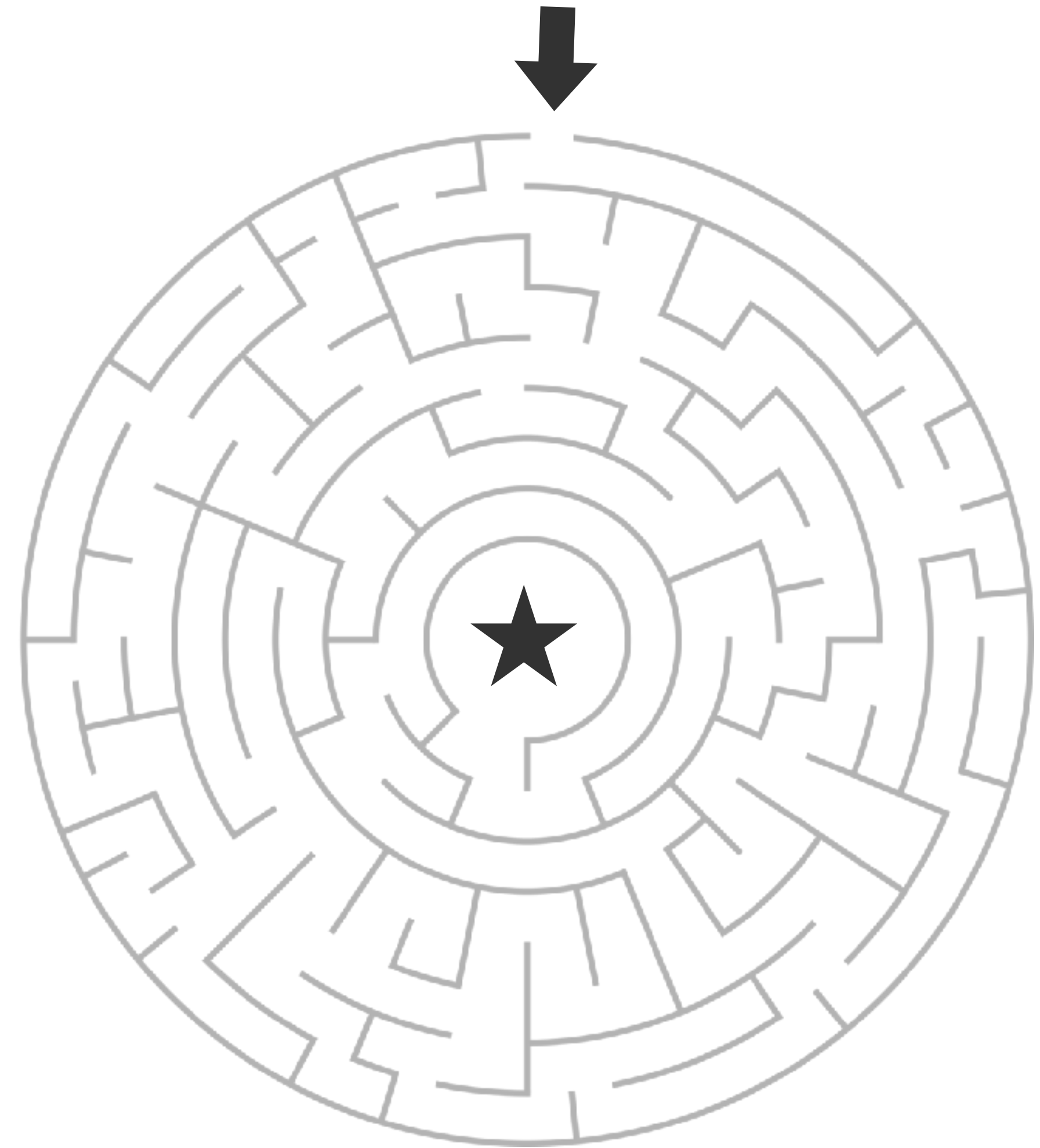
Recompile with each change



Design review two weeks later

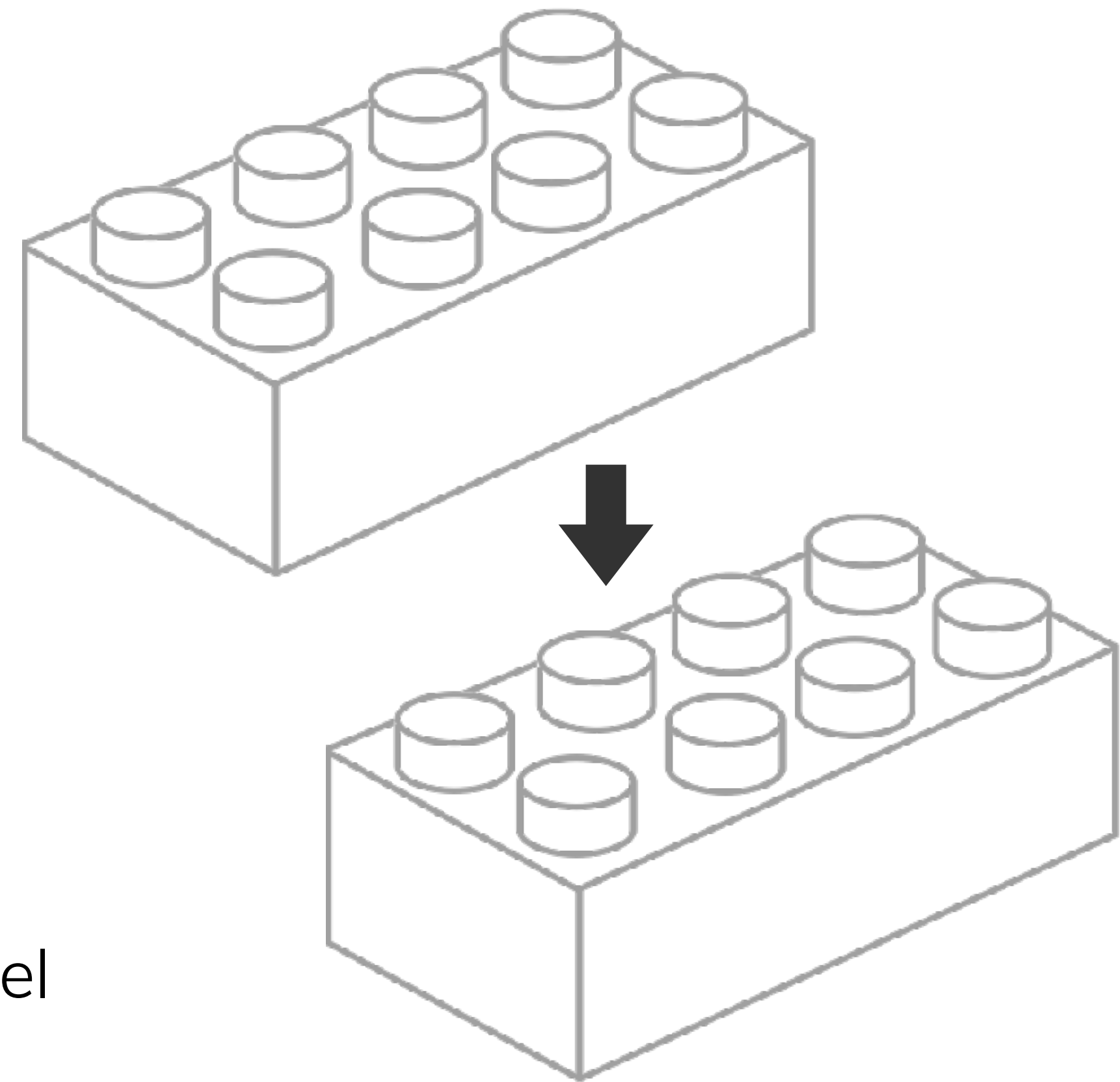


Rework when it's already shipped



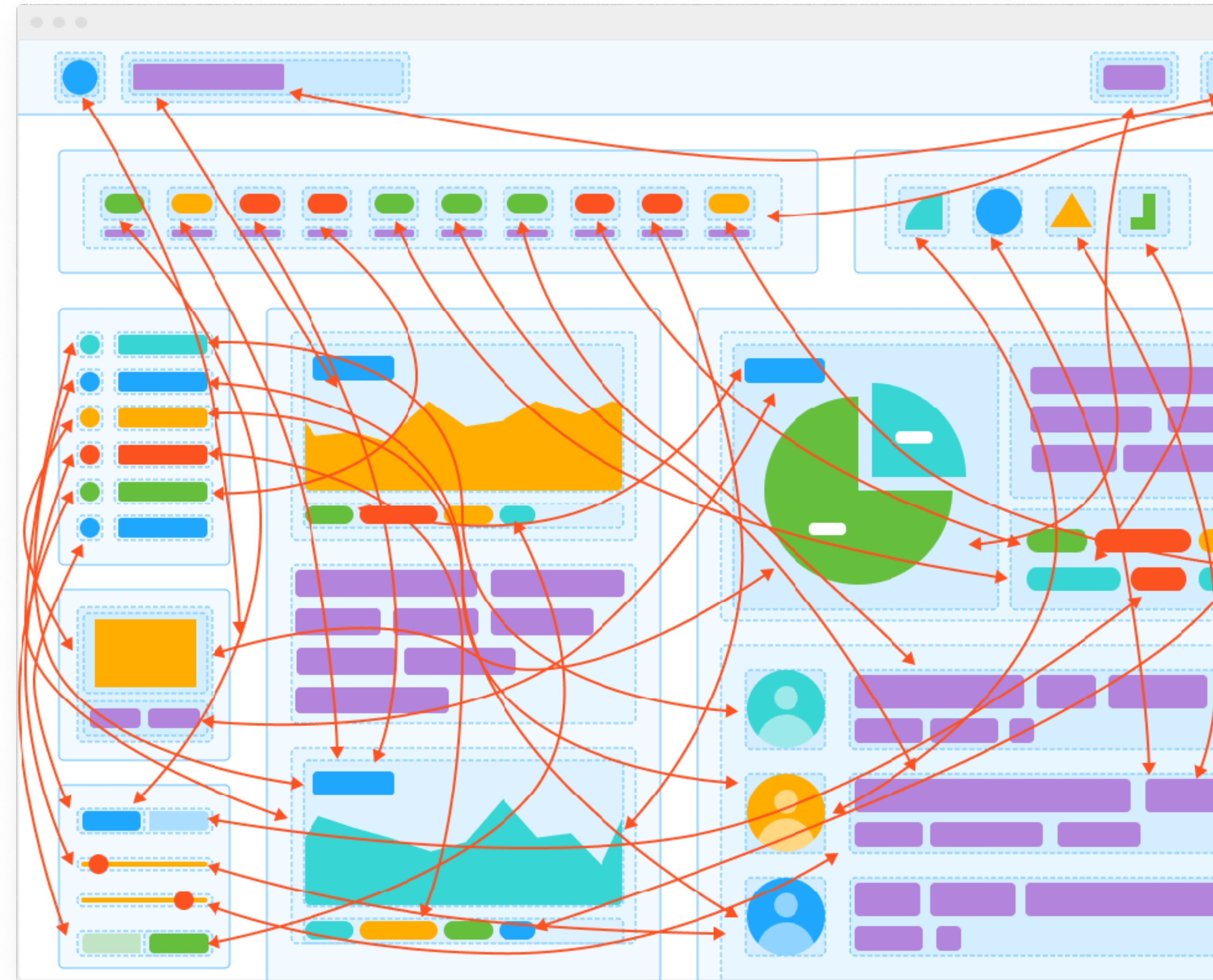
Modern UIs are ~~built~~ *assembled* from components

- ✓ **Efficiency:** Reuse existing components
- ✓ **Speed:** Parallelize development across teams
- ✓ **Quality:** Verify that UIs work in different scenarios
- ✓ **Maintenance:** Pinpoint bugs at the component level

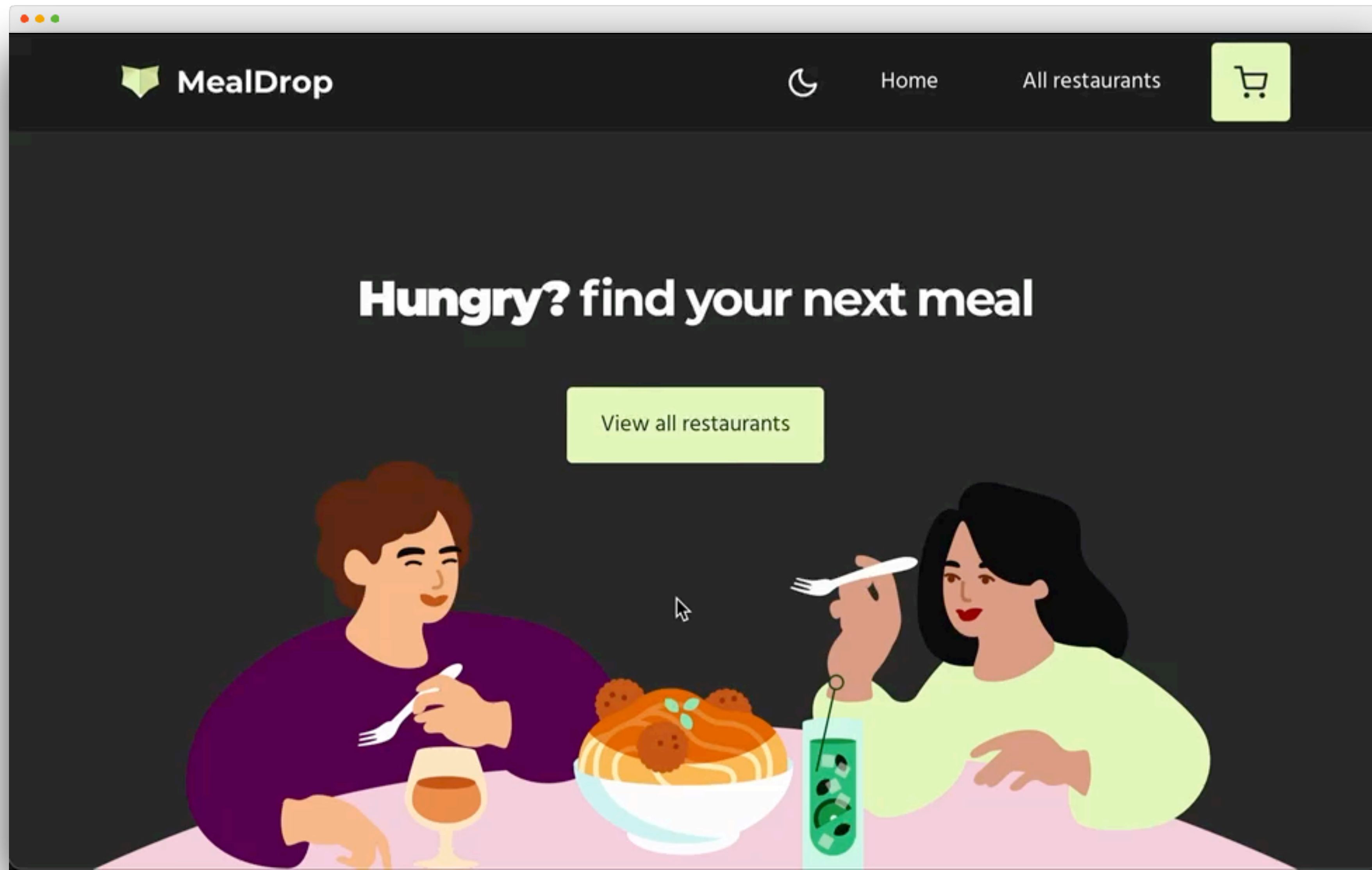


UI development has many challenges

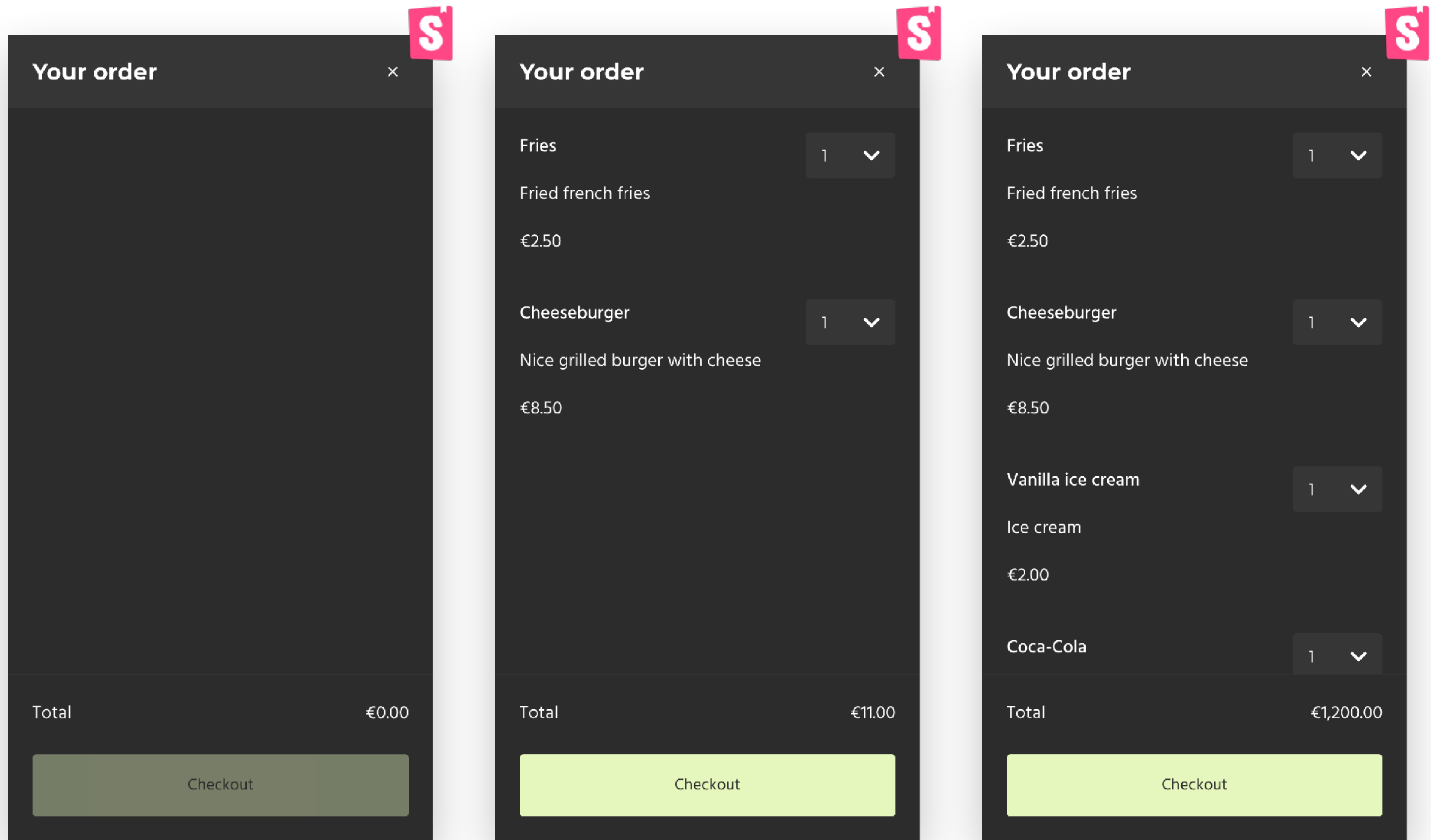
- | | |
|------------------|-----------------|
| 🌐 Cross-browser | ⚡ Render speed |
| 📱 Device compat. | ⚖️ Bundle size |
| 📏 Responsiveness | 📦 Dependencies |
| 🔍 SEO | 🎨 Styling |
| 🔗 Open graph | ♿ Accessibility |
| 🔒 User privacy | 💠 State mgmt. |



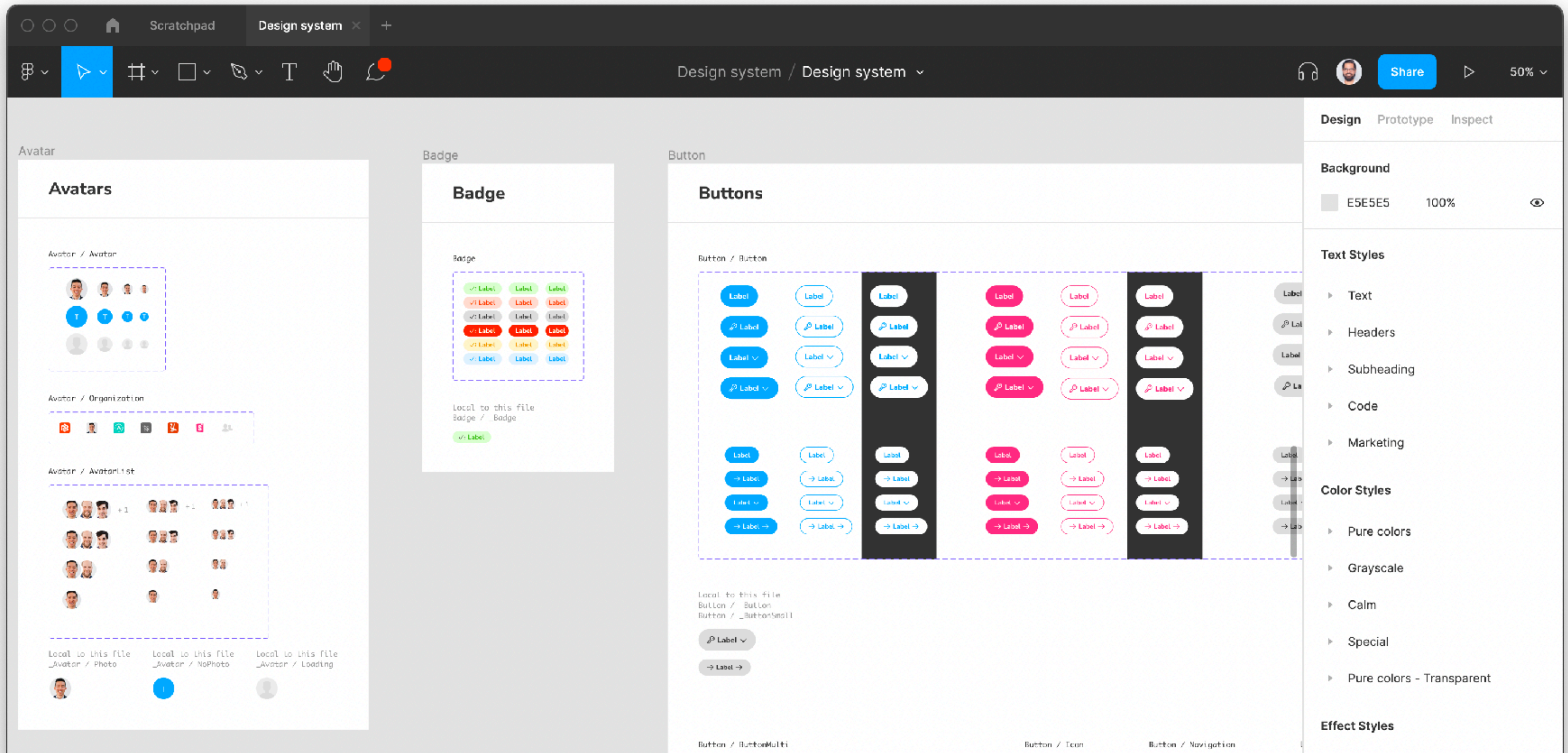
Existing UI development workflows are **clunky**



Isolate **all use cases** of a component

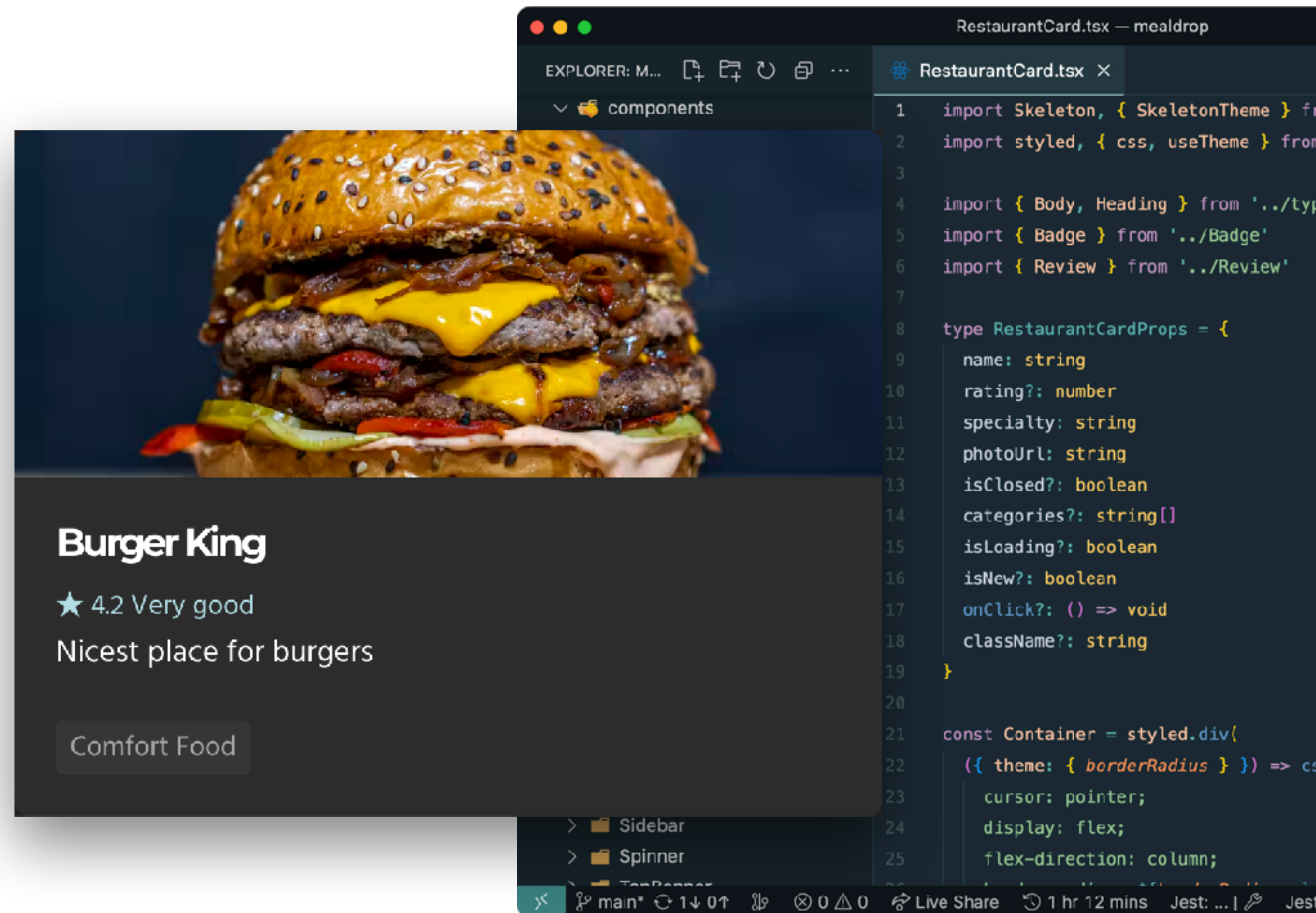


Designers use **sticker sheets**

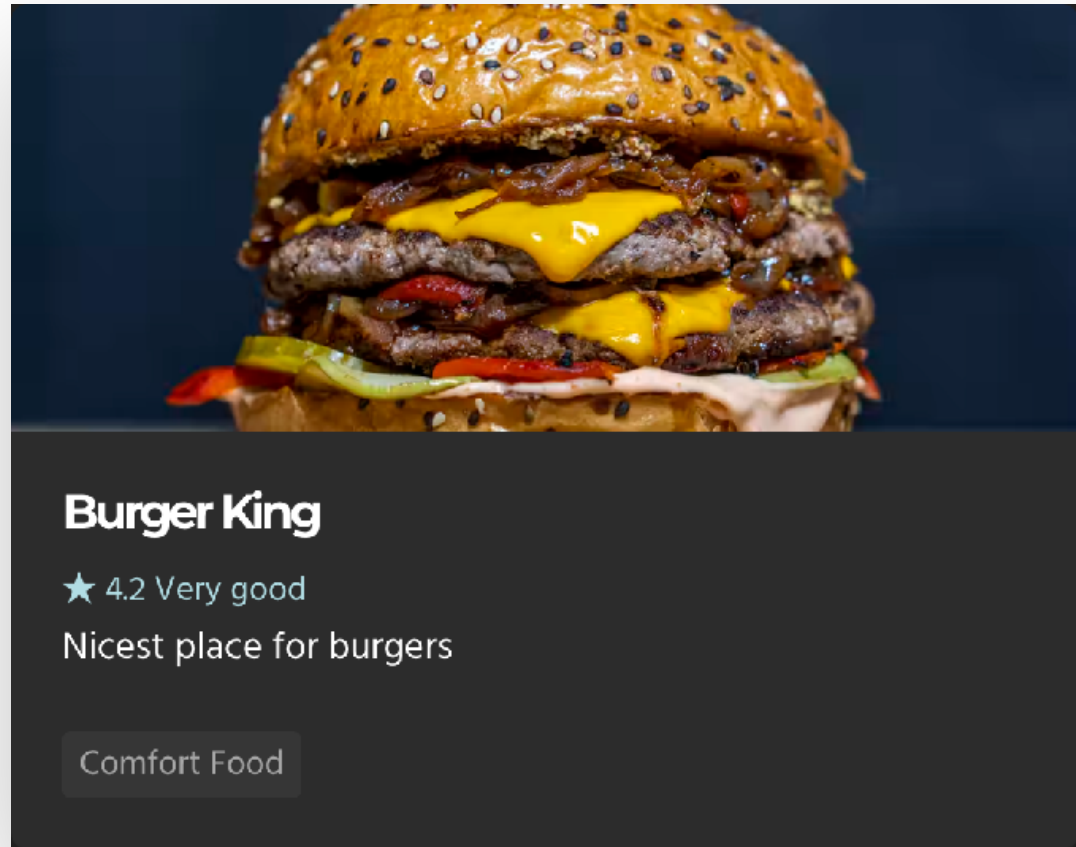


Storybook is a sticker sheet for components

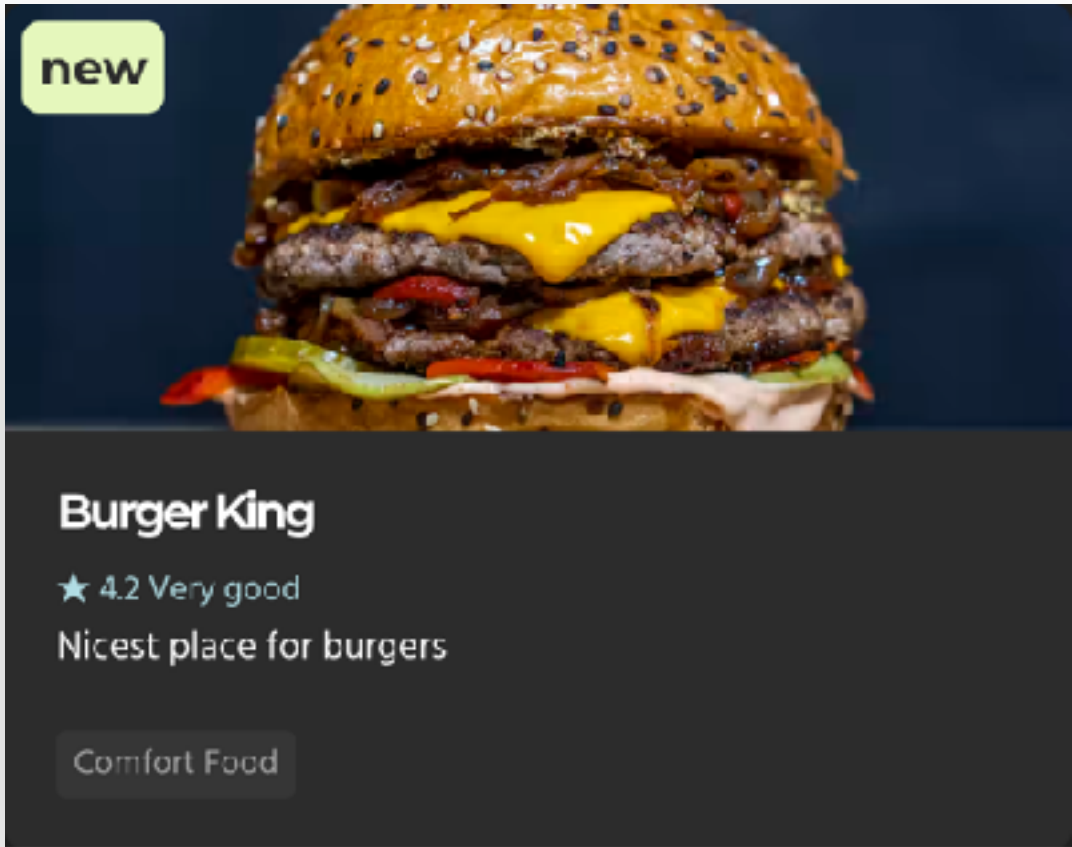
- 🖼️ **Build** components in isolation
- 📖 **Catalog** all components and states
- ✍️ **Document** your component library
- ✅ **Test** complex UI and interactions
- 🔗 **Share** with stakeholders



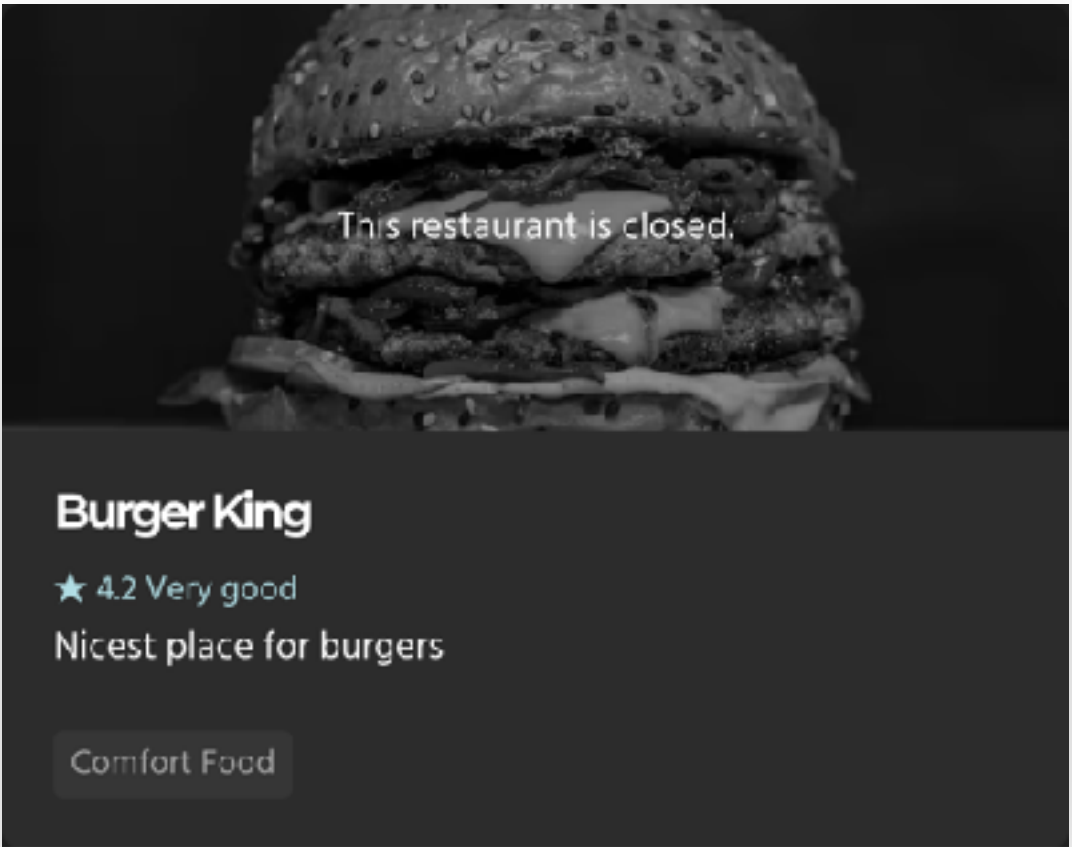
Stress test component **states**



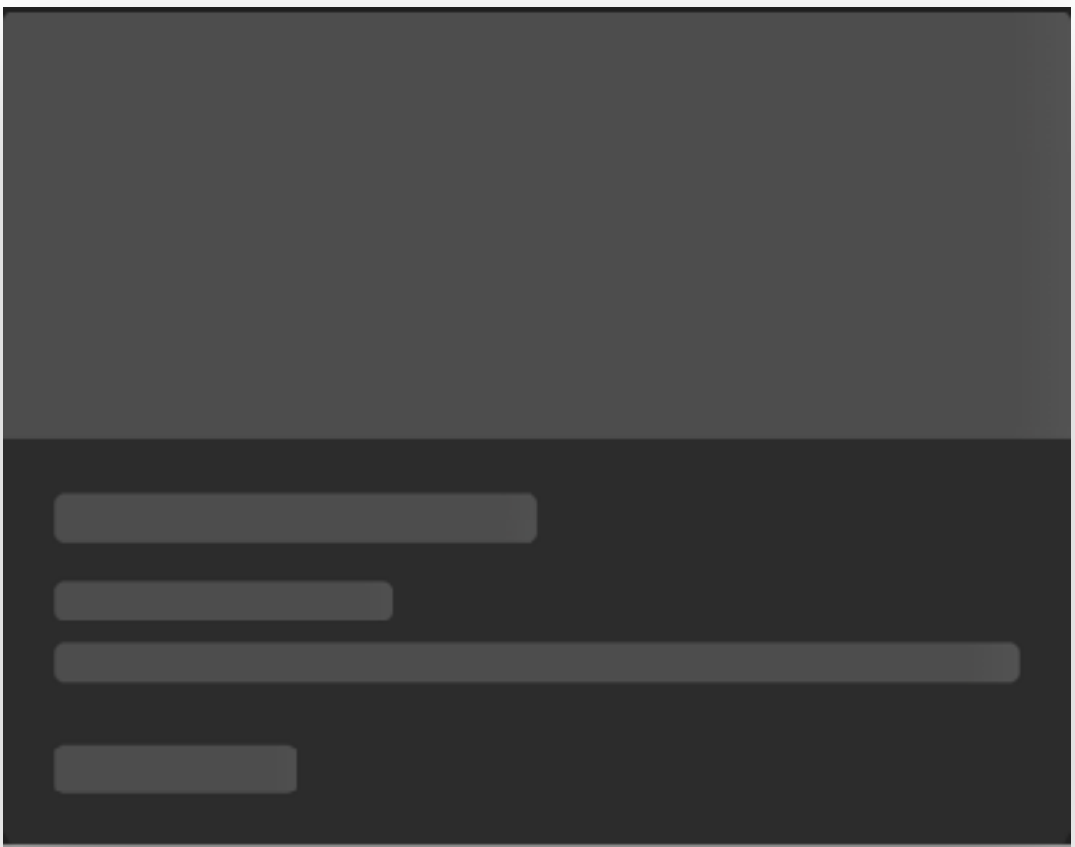
Default



New

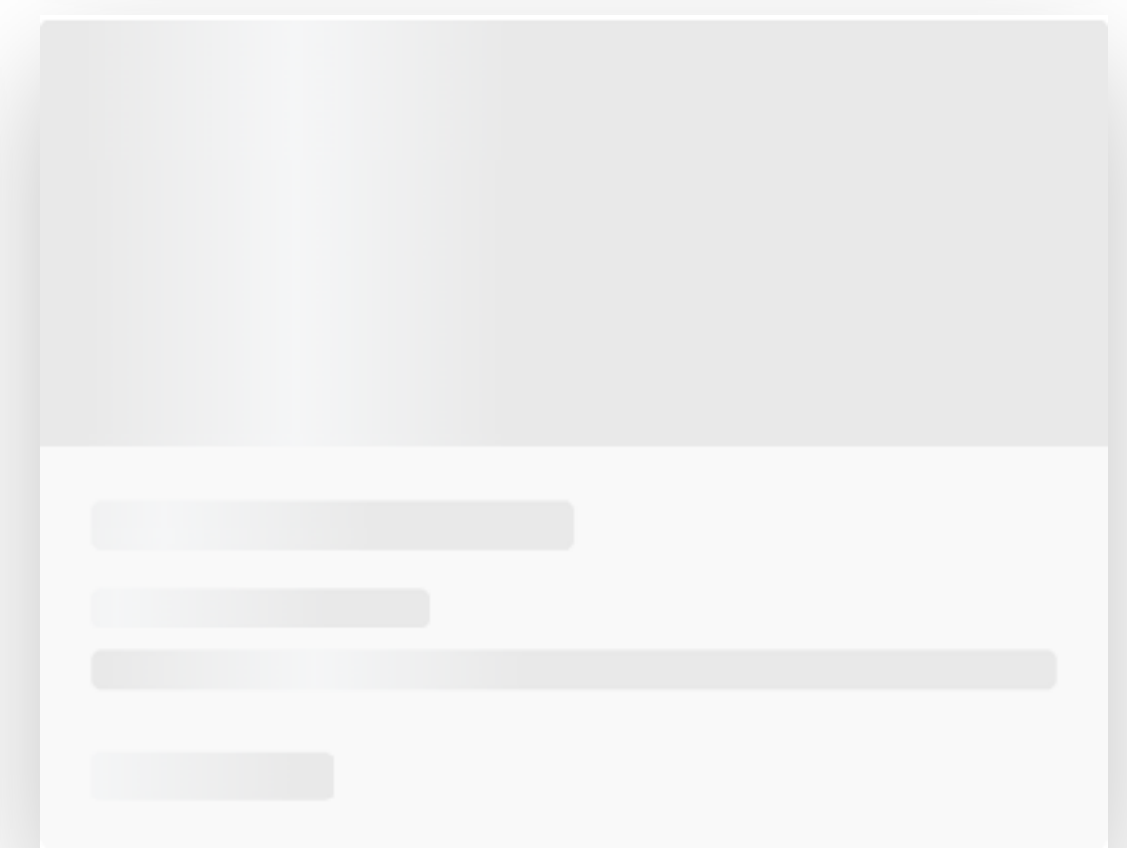
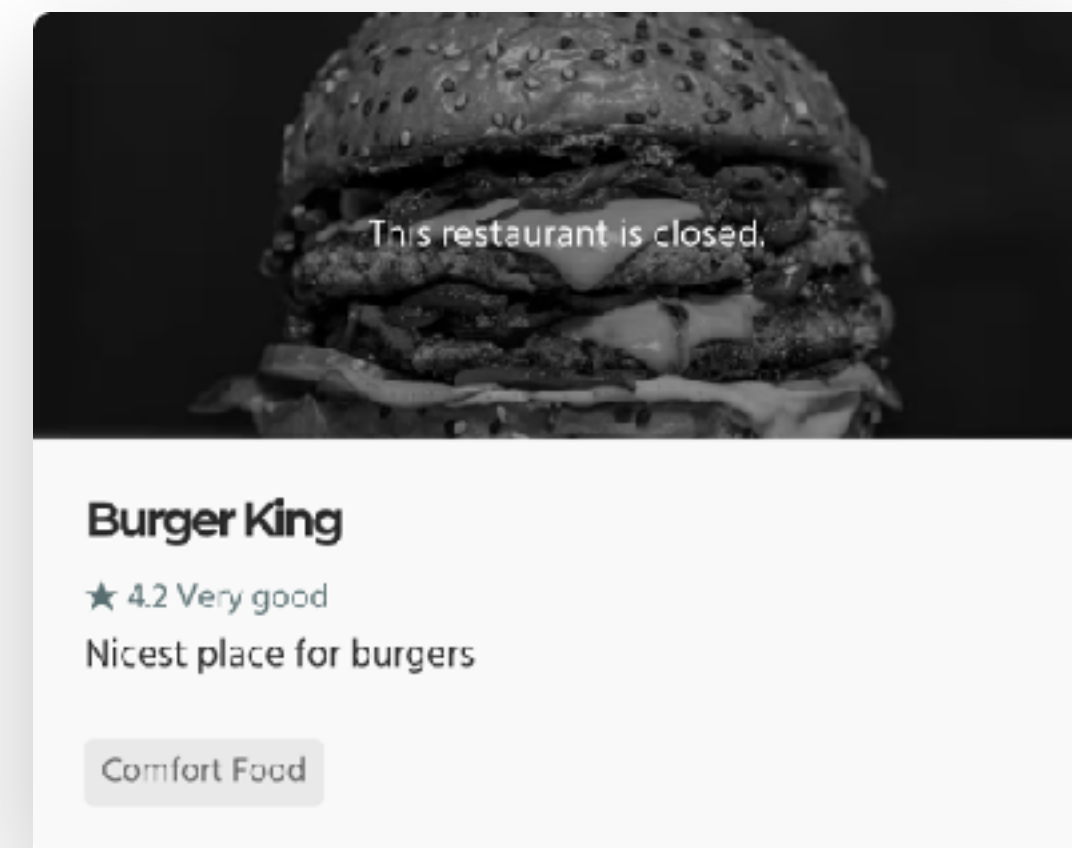
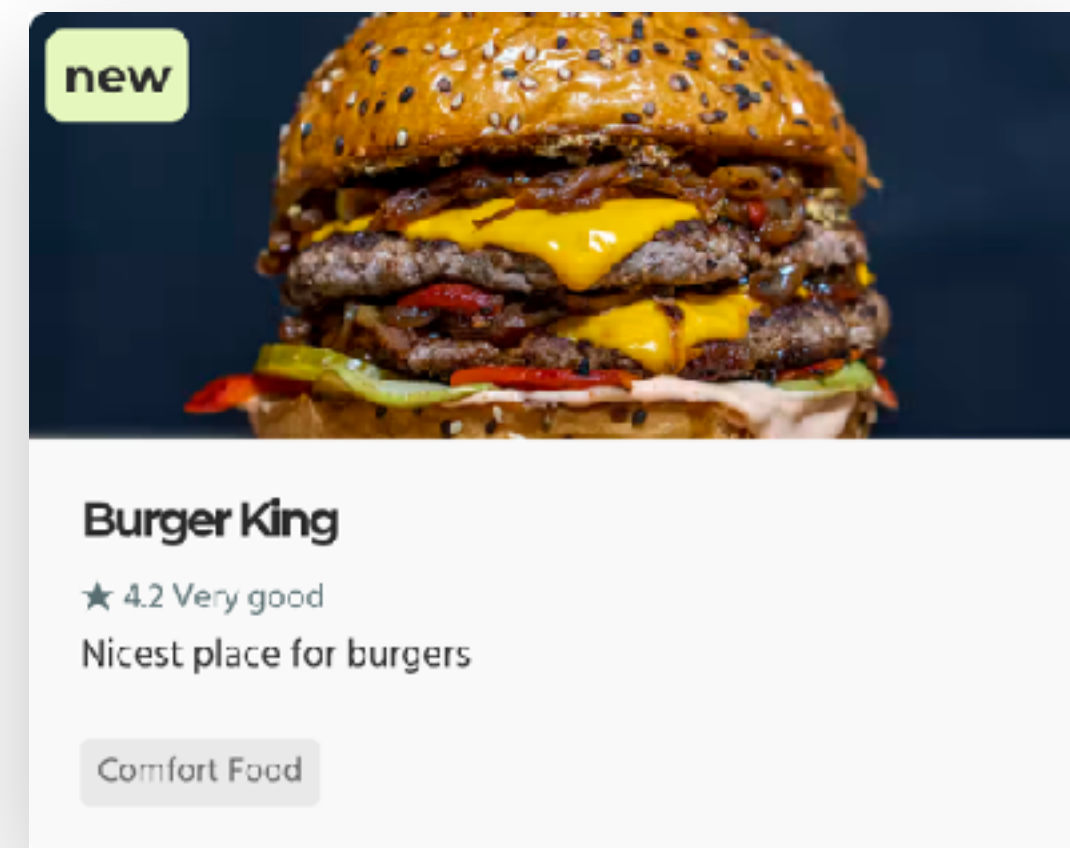
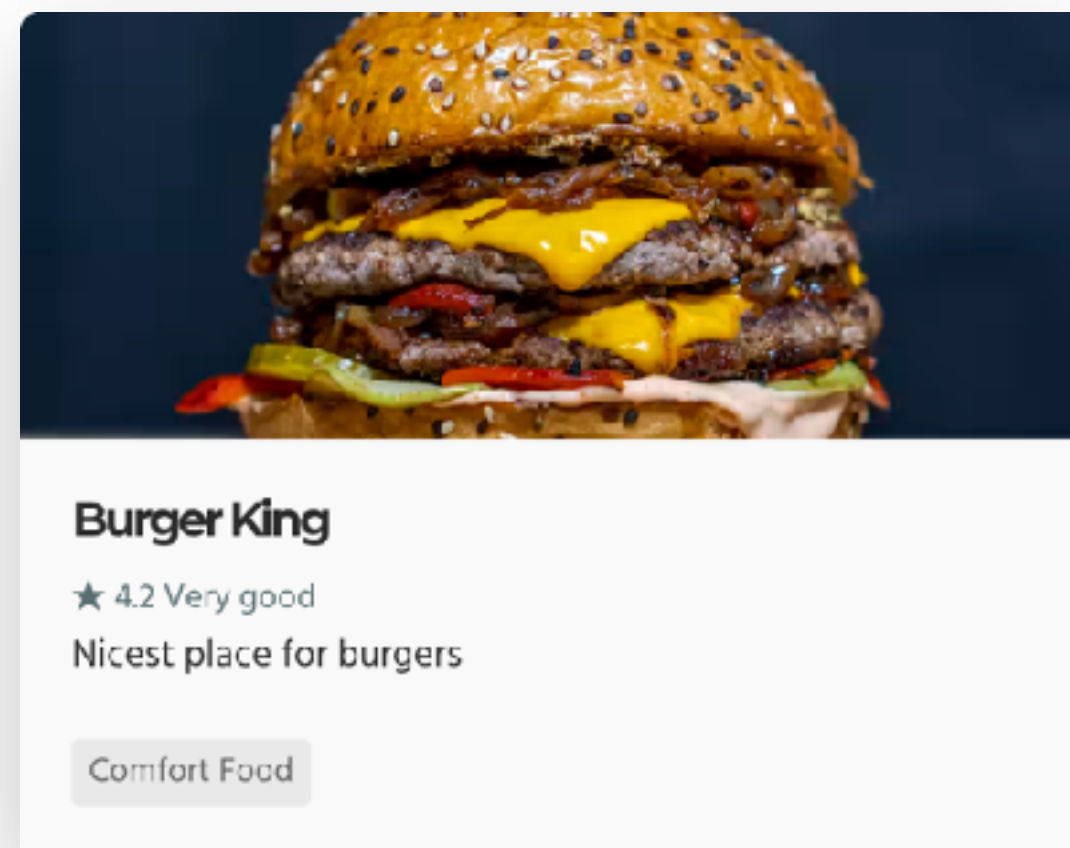
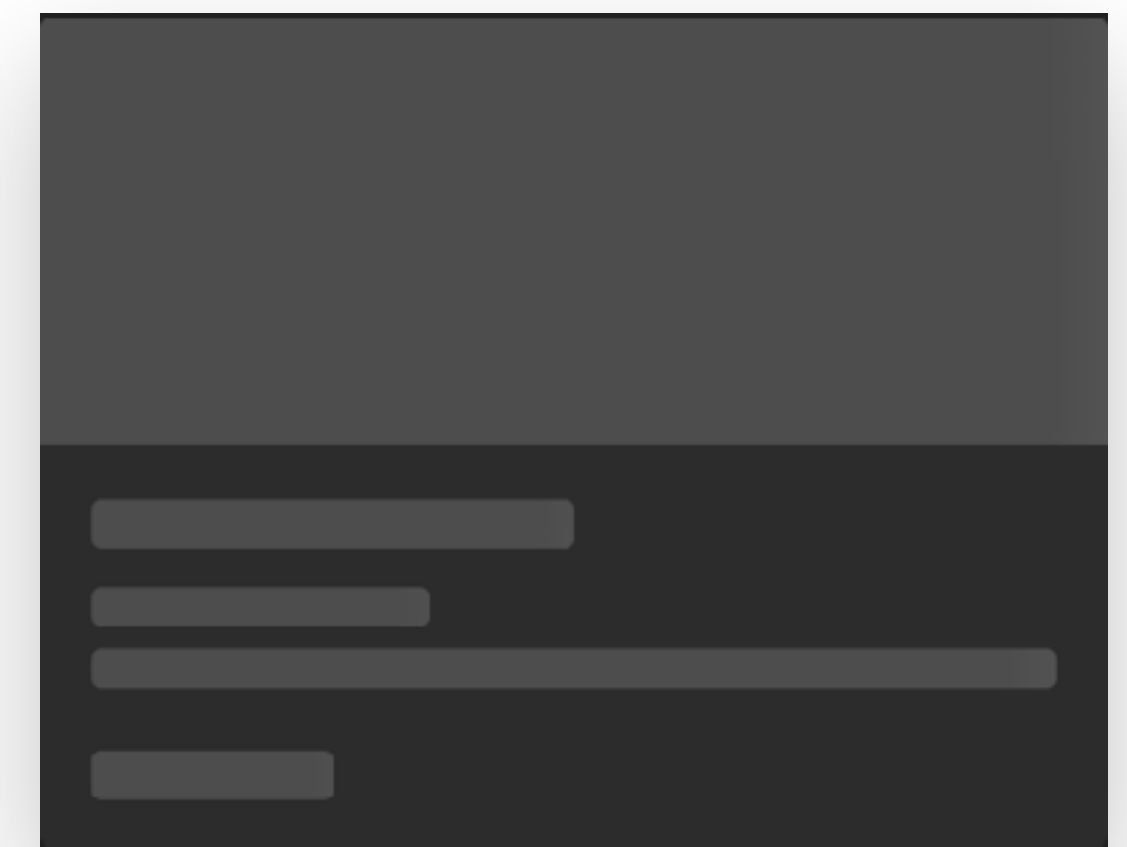
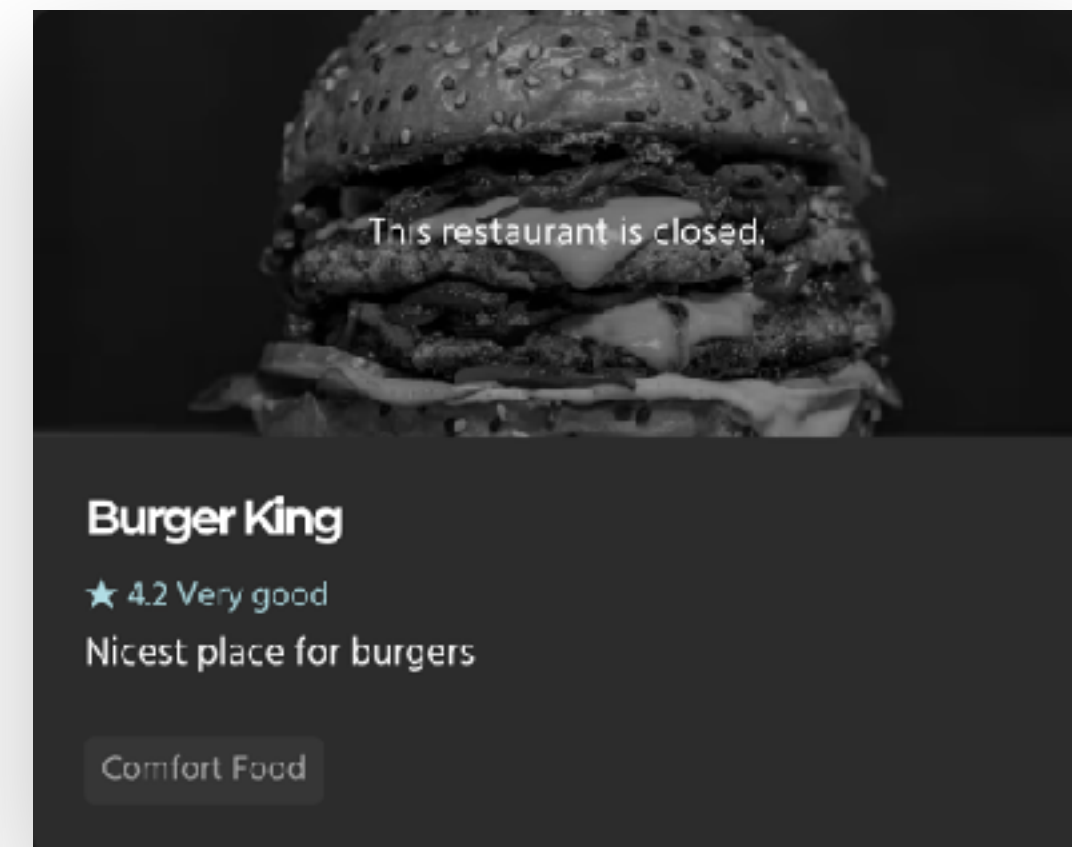
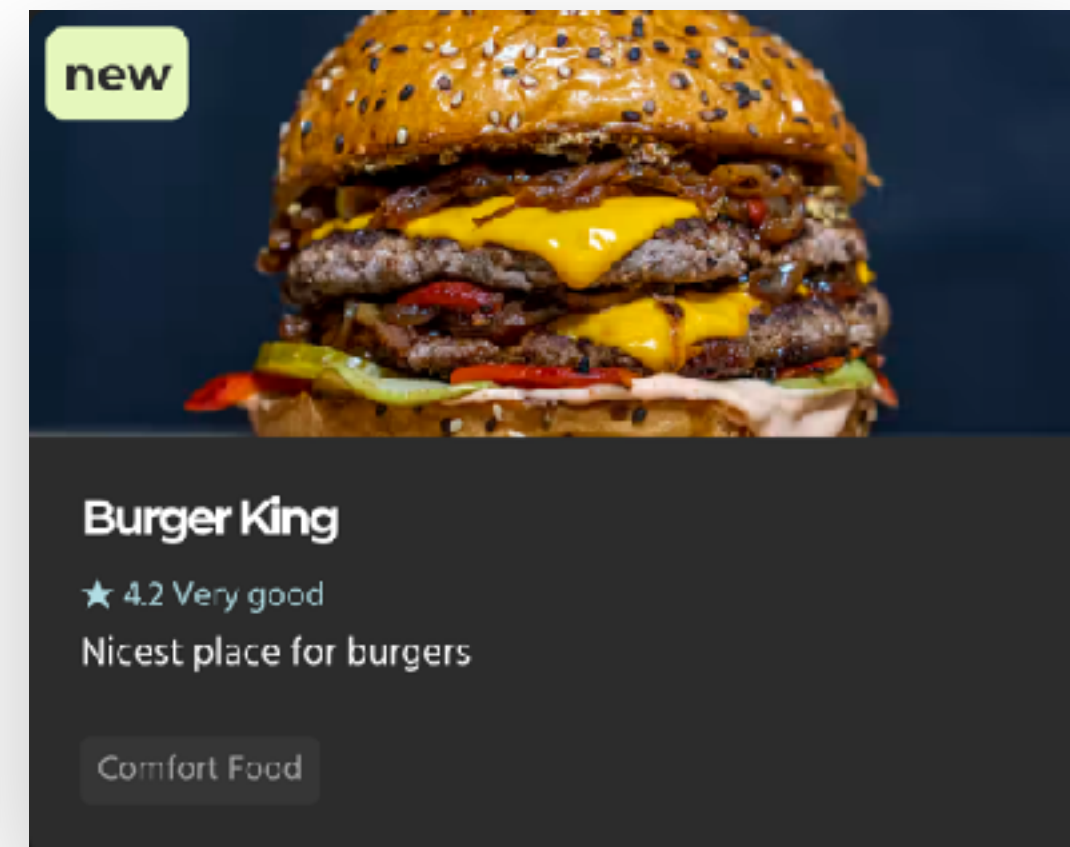
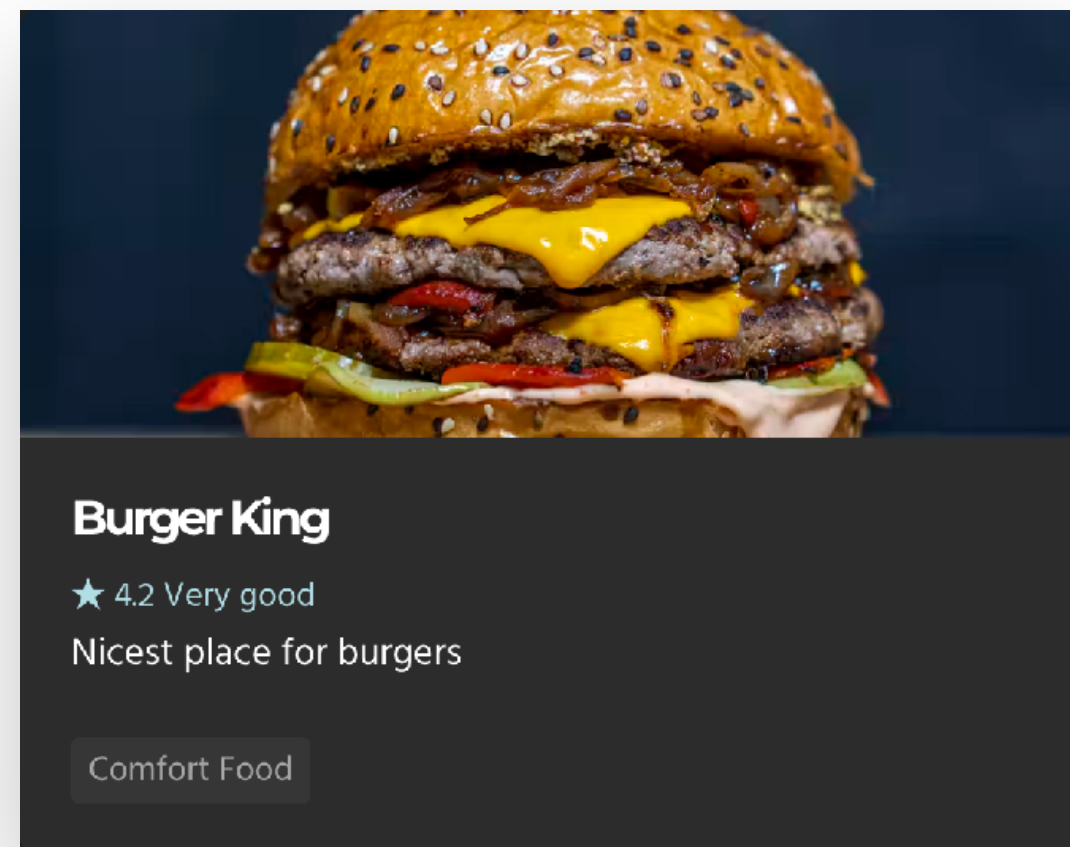


Closed



Loading

Stress test component **variants**



Define use cases with **stories**

States

- Loading
- Disabled
- In progress
- Error
- Accepted / denied
- Expanded / collapsed

Edge cases

- Long or short text
- Big or small images
- Extreme numbers
- Missing data
- Special characters

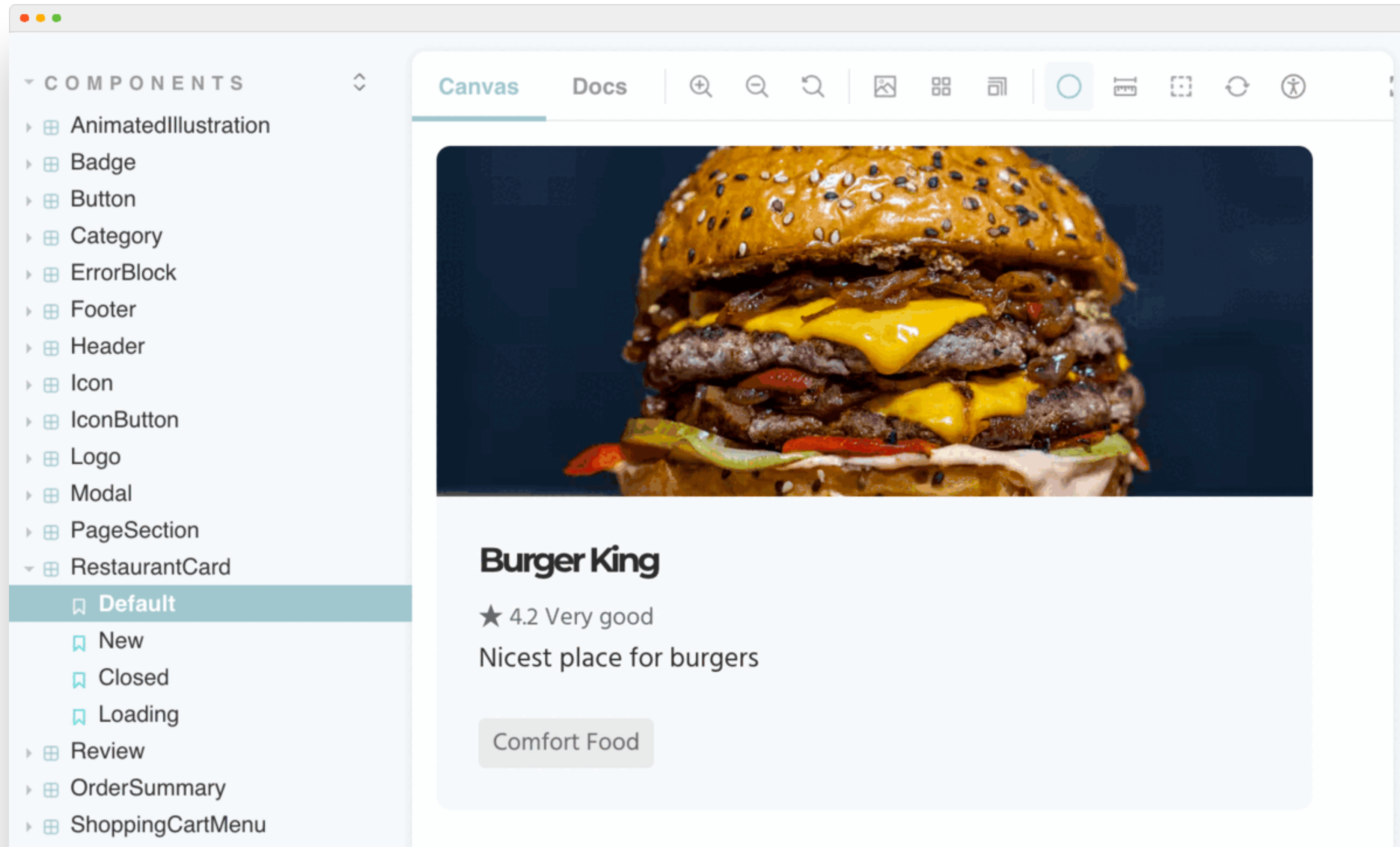
Context

- Signed in / signed out
- Language / location
- Color preference
- A/B test
- Offline

Countless combinations



Keep component development **organized**



The **industry standard** for UI development

 **ATLASSIAN**

Design System

 **shopify**

Polaris

 **Adobe**

Spectrum

 **BBC**

Psammead



Design System

 **WORDPRESS**

Gutenberg

 **GOV.UK**

Design System

priceline

Design System

The Guardian

Web

 **workday**

Canvas

GitHub

Primer

 **chakra**

UI components

 **lonely planet**

Backpack UI

 **airbnb**

Dates

IBM

Carbon

WIX

Style

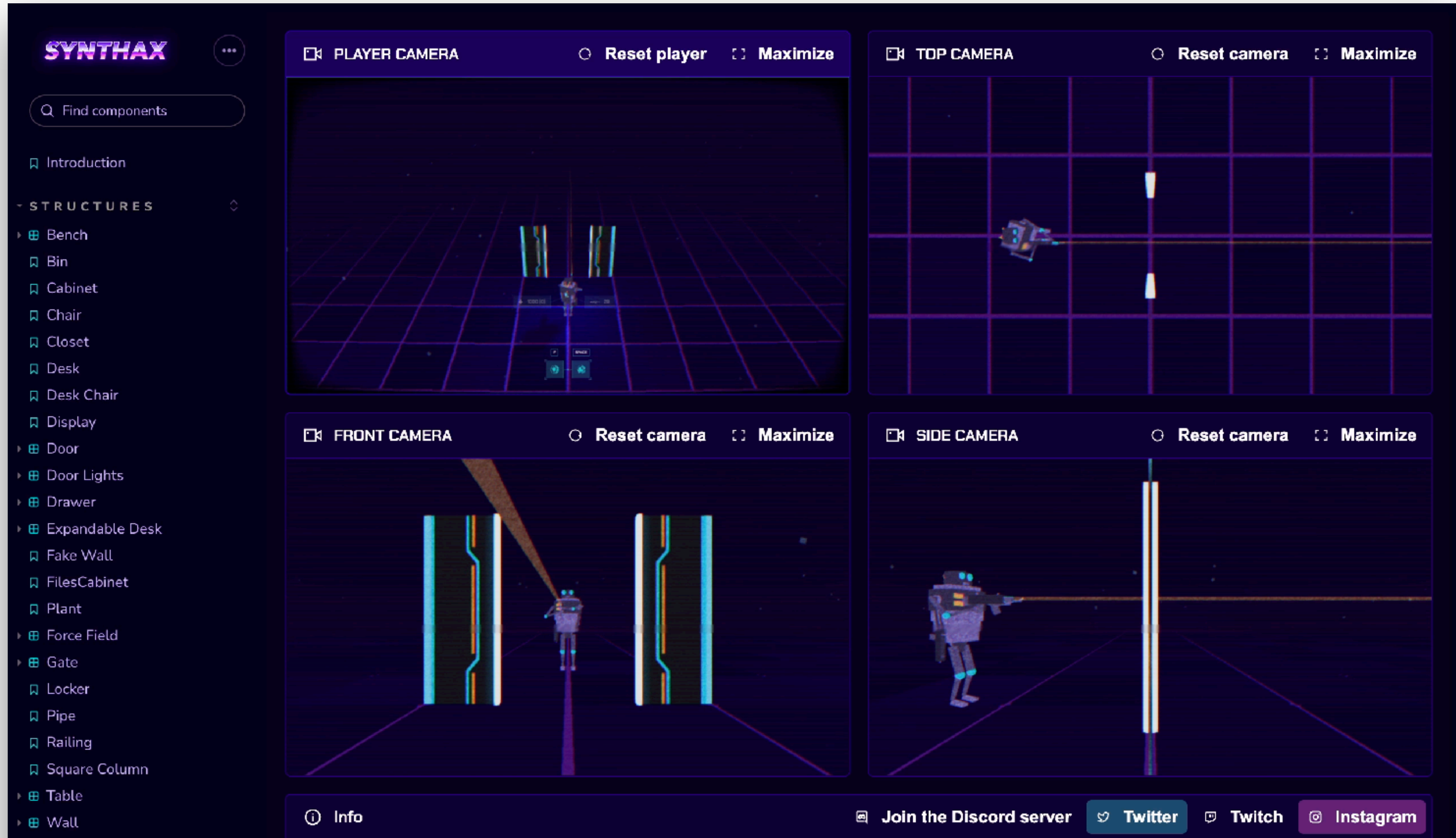
 **GitLab**

UI

 **Grafana Labs**

Design System

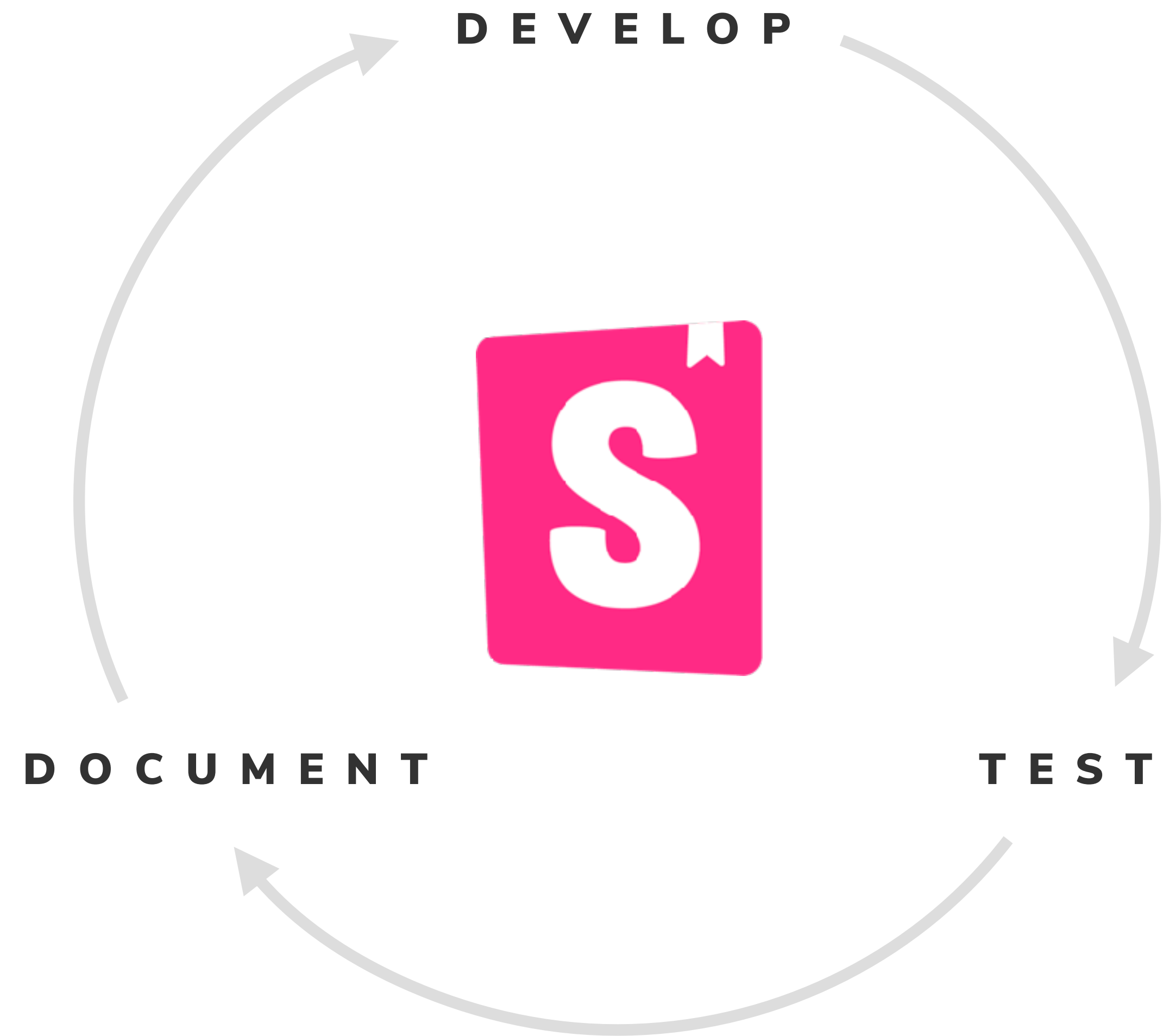
Not just for boring apps



Storybook 7.0 released in March 🎉



Storybook 7.1 is coming...

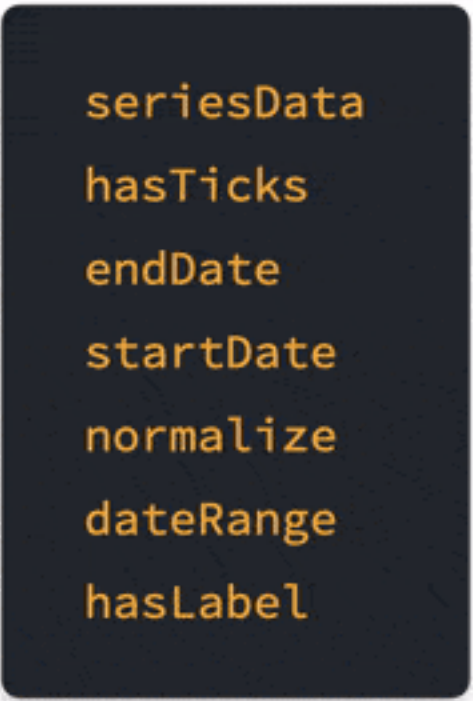


Develop

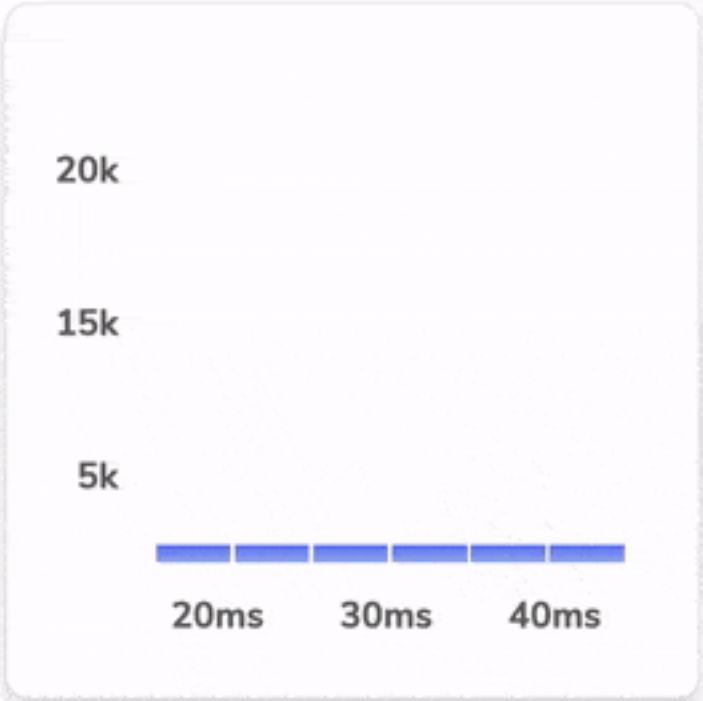
Work fast and with confidence by developing your components in isolation



Capture component use cases as stories



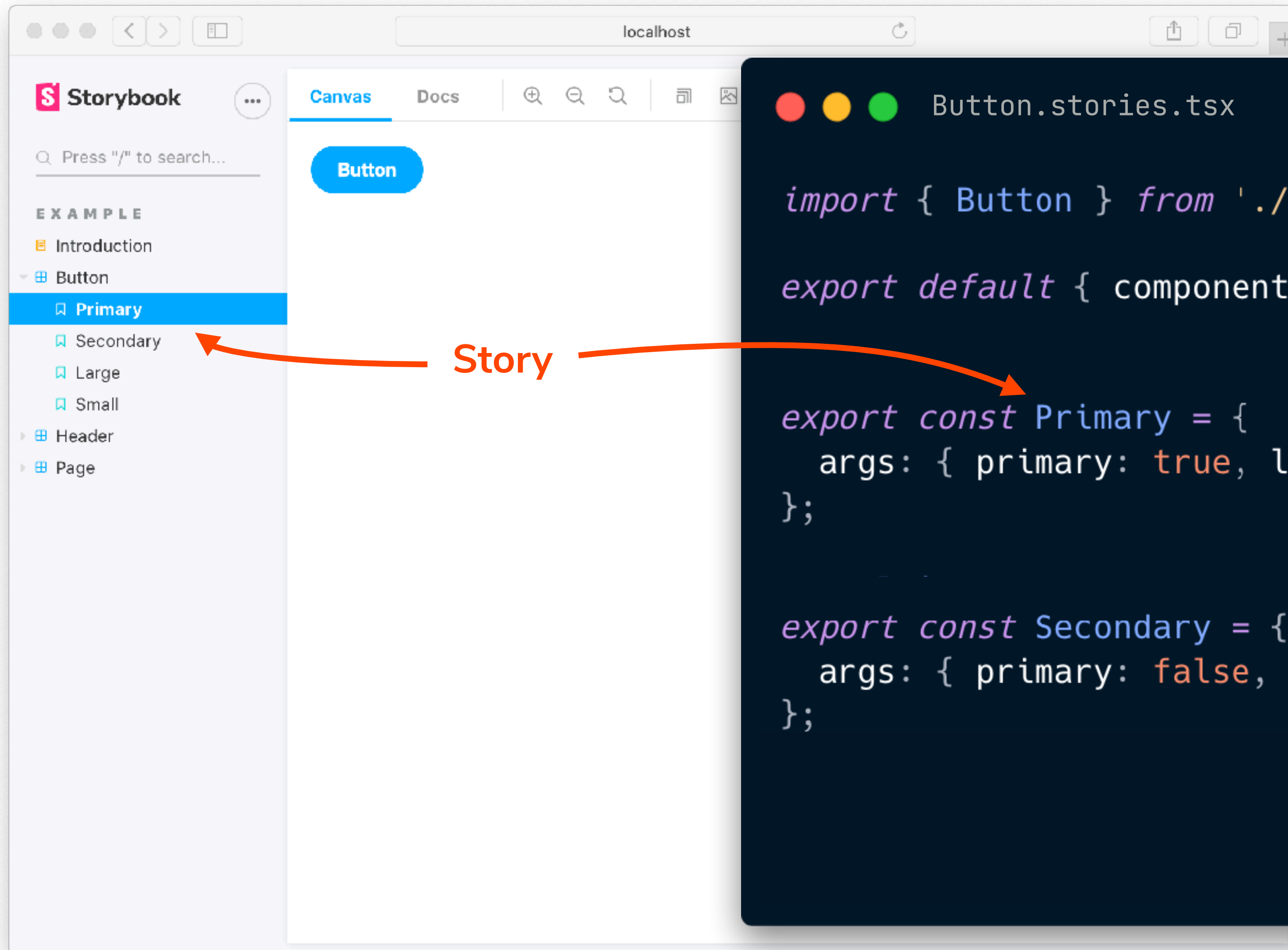
Inputs
(props, slots, input)



Component

Stories are unit tests, but for UIs

Writing a story



Button.stories.tsx

```
import { Button } from './Button';
```

```
export default { component: Button };
```

```
export const Primary = {  
  args: { primary: true, label: 'Button' }  
};
```

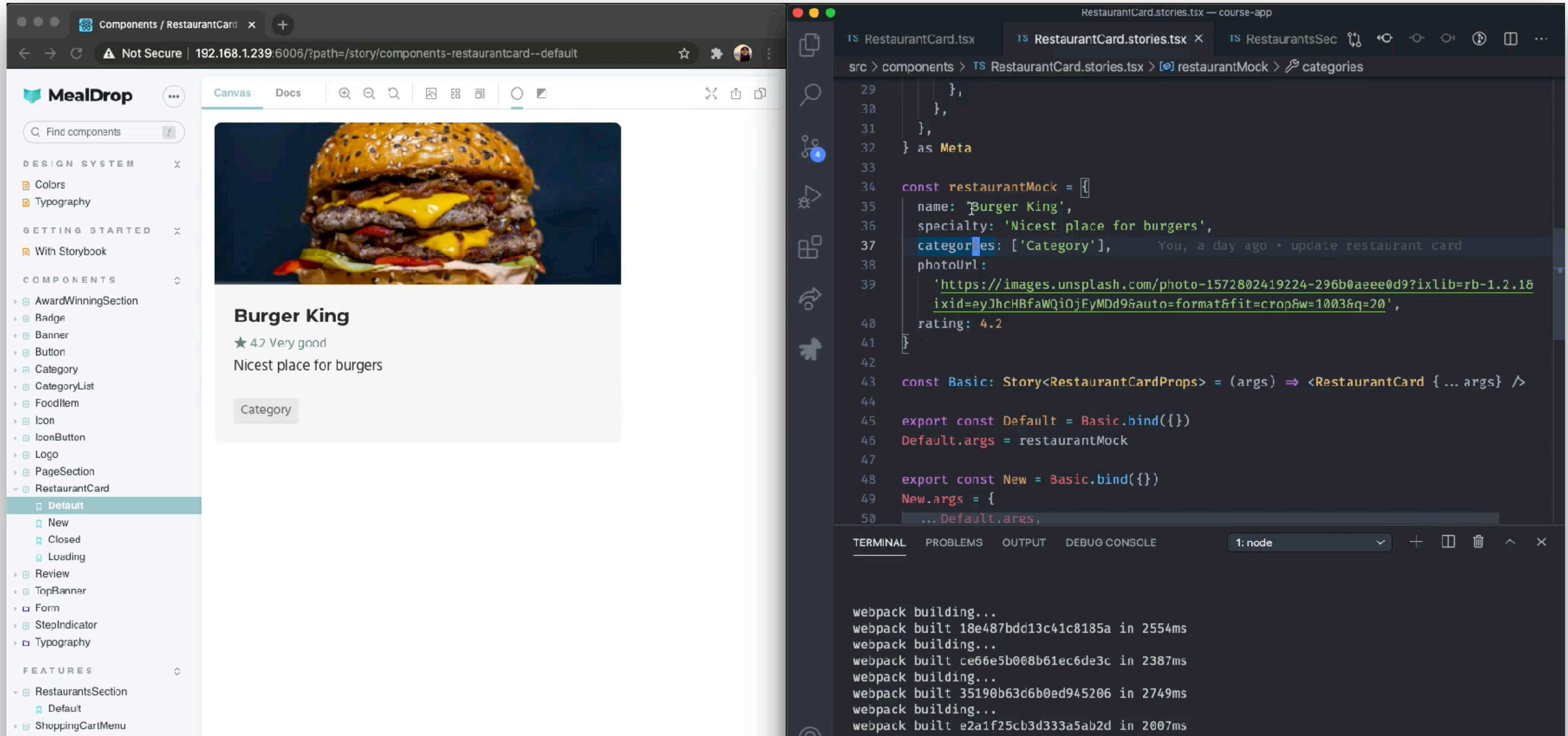
```
export const Secondary = {  
  args: { primary: false, label: 'Button' }  
};
```

Component reference

Meta

Story

Quick feedback loop



The image illustrates a "Quick feedback loop" in a development environment. It is split into two main sections: a web browser on the left and a code editor on the right.

Web Browser (Left): The browser shows a web application named "MealDrop". The left sidebar contains a "Components" list with items like "AwardWinningSection", "Badge", "Banner", "Button", "Category", "CategoryList", "FoodItem", "Icon", "IconButton", "Logo", "PageSection", "RestaurantCard", "Review", "TopBanner", "Form", "StepIndicator", and "Typography". The "RestaurantCard" component is selected. The main canvas displays a preview of the "Burger King" restaurant card, featuring a burger image, the name "Burger King", a star rating of 4.2, the text "Very good", and the description "Nicest place for burgers". Below the card is a "Category" button.

Code Editor (Right): The code editor shows the source code for the "RestaurantCard" component. The file is named "RestaurantCard.stories.tsx" and is part of the "course-app" project. The code defines a mock object for the restaurant and a story function. The mock object is defined as follows:

```
const restaurantMock = {
  name: 'Burger King',
  specialty: 'Nicest place for burgers',
  categories: ['Category'],
  photoUrl: 'https://images.unsplash.com/photo-1572802419224-296b0aeee0d9?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=1003&q=20',
  rating: 4.2
}
```

The story function is defined as:

```
const Basic: Story<RestaurantCardProps> = (args) => <RestaurantCard {... args} />
```

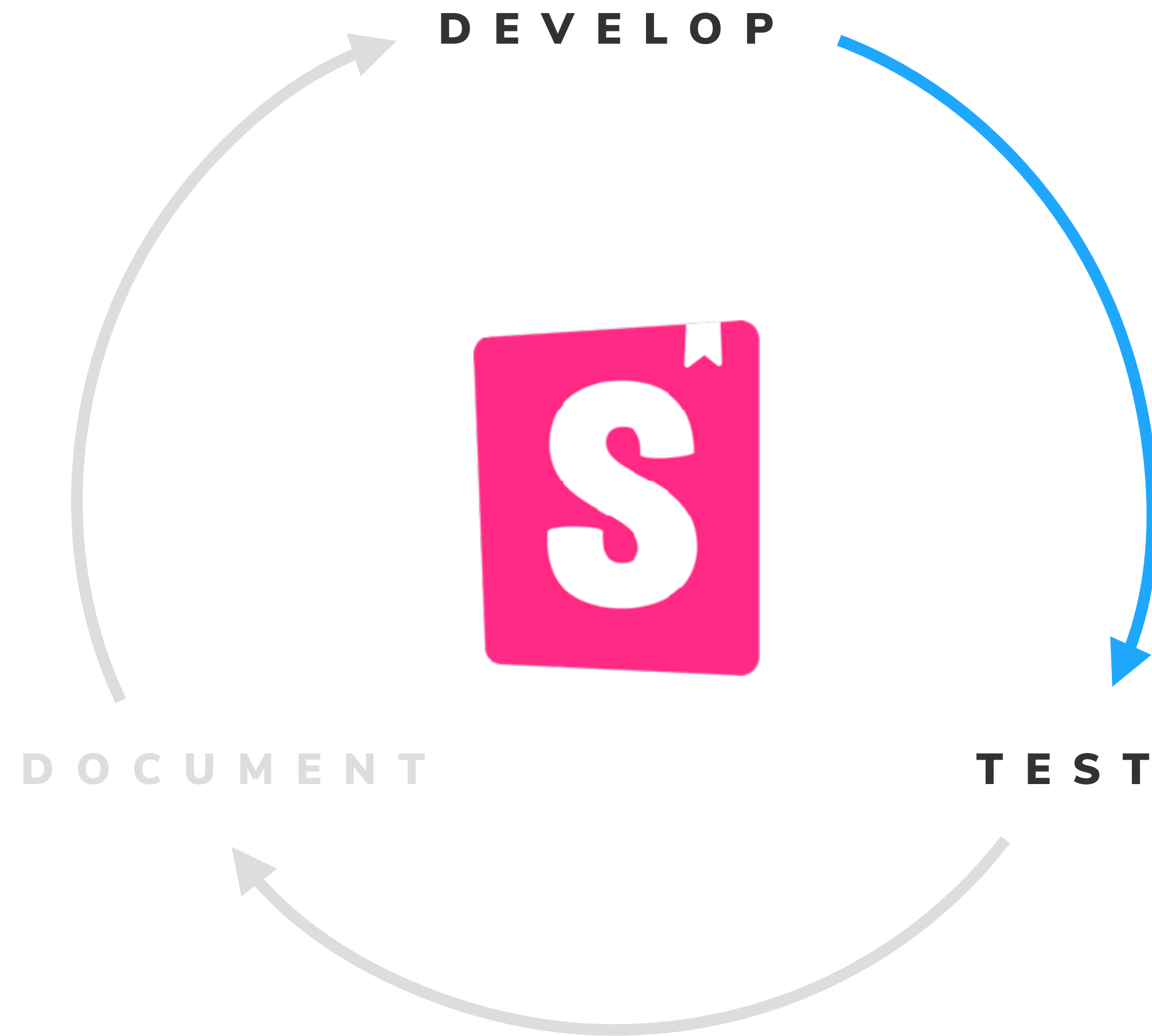
The code also includes the following exports:

```
export const Default = Basic.bind({})
Default.args = restaurantMock

export const New = Basic.bind({})
New.args = {
  ... Default.args,
}
```

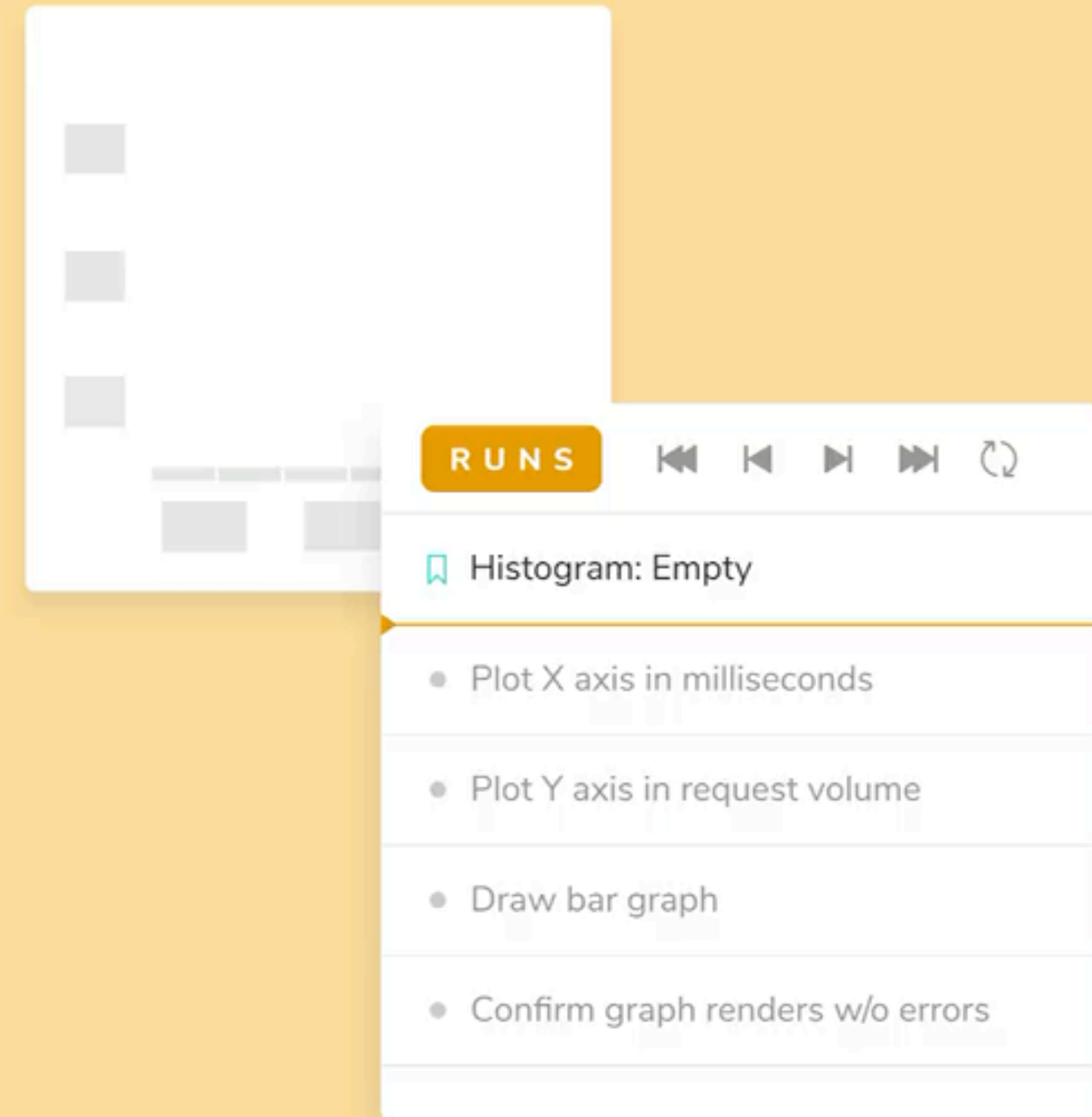
The bottom of the code editor shows a terminal window with the following output:

```
webpack building...
webpack built 18e487bdd13c41c8185a in 2554ms
webpack building...
webpack built ce66e5b008b61ec6de3c in 2387ms
webpack building...
webpack built 35190b63d6b0ed945206 in 2749ms
webpack building...
webpack built e2a1f25cb3d333a5ab2d in 2007ms
```

Test

Simulate user behavior like click, hover, and type, from within your story file.



THE OLD-FASHIONED WAY

Component unit tests



Setup component test case



Render the component



Simulate events with Testing Library



Run assertions with Jest

```
import { render, screen, fireEvent } from 'testing-library/angular'
import { DeleteCustomerDialog } from '../delete-customer-dialog'

describe('Delete Customer dialog', () => {
  it('should open a dialog', async () => {
    await render(DeleteCustomerDialog);
    await fireEvent.click(screen.getByRole('button'));
    await expect(screen.getByText('Are you sure?'))
      .toBeInTheDocument();
  });
});
```

Jest relies on Node and JSDOM 🤖

```
PASS src/InboxScreen.test.js
InboxScreen
  ✓ should pin a task (907 ms)
  ✓ should archive a task (166 ms)
  ✓ should edit a task (135 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        4.858 s, estimated 5 s
Ran all test suites.

Watch Usage: Press w to show more.
```

```
<button
  aria-label="pin"
  class="chakra-button css-xkku7u"
  type="button"
>
  <svg
    aria-hidden="true"
    class="chakra-icon css-onkibi"
  ...

65 |   });
66 |
> 67 |   const task = getByRole('listitem', {
    |               ^
68 |     name: 'Fix bug in input error',
69 |   });
70 |   const taskInput = within(task).getByRole('textbox');

at Object.getElementError (node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/config.js:10:11)
at node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/query-helpers.js:90:38
at node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/query-helpers.js:62:17
at getByRole (node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/query-helpers.js:100:17)
at Object.<anonymous> (src/InboxScreen.test.js:67:18)
```

```
Test Suites: 1 failed, 1 total
```



```
DeleteCustomerDialog.stories.js

import { within, fireEvent } from '@storybook/testing-library';
import { expect } from '@storybook/jest';
import { DeleteCustomerDialog } from './DeleteCustomerDialog';

export default { component: DeleteCustomerDialog };

export const Default = {
  play: async ({ canvasElement }) => {
    const canvas = within(canvasElement);
    await fireEvent.click(
      canvas.getByRole('button', { name: 'Delete Customer' })
    );
    await expect(
      canvas.getByText('Are you sure? You can\'t undo this action afterwards.')
    ).toBeInTheDocument();
  },
};
```

localhost

Canvas Docs

+

+

+

Delete Customer

Delete Customer

Are you sure? You can't undo this action afterwards.

Cancel

Delete

Controls Actions Accessibility Interactions

PASS

Scroll to end

⏮

⏪

⏩

⏭

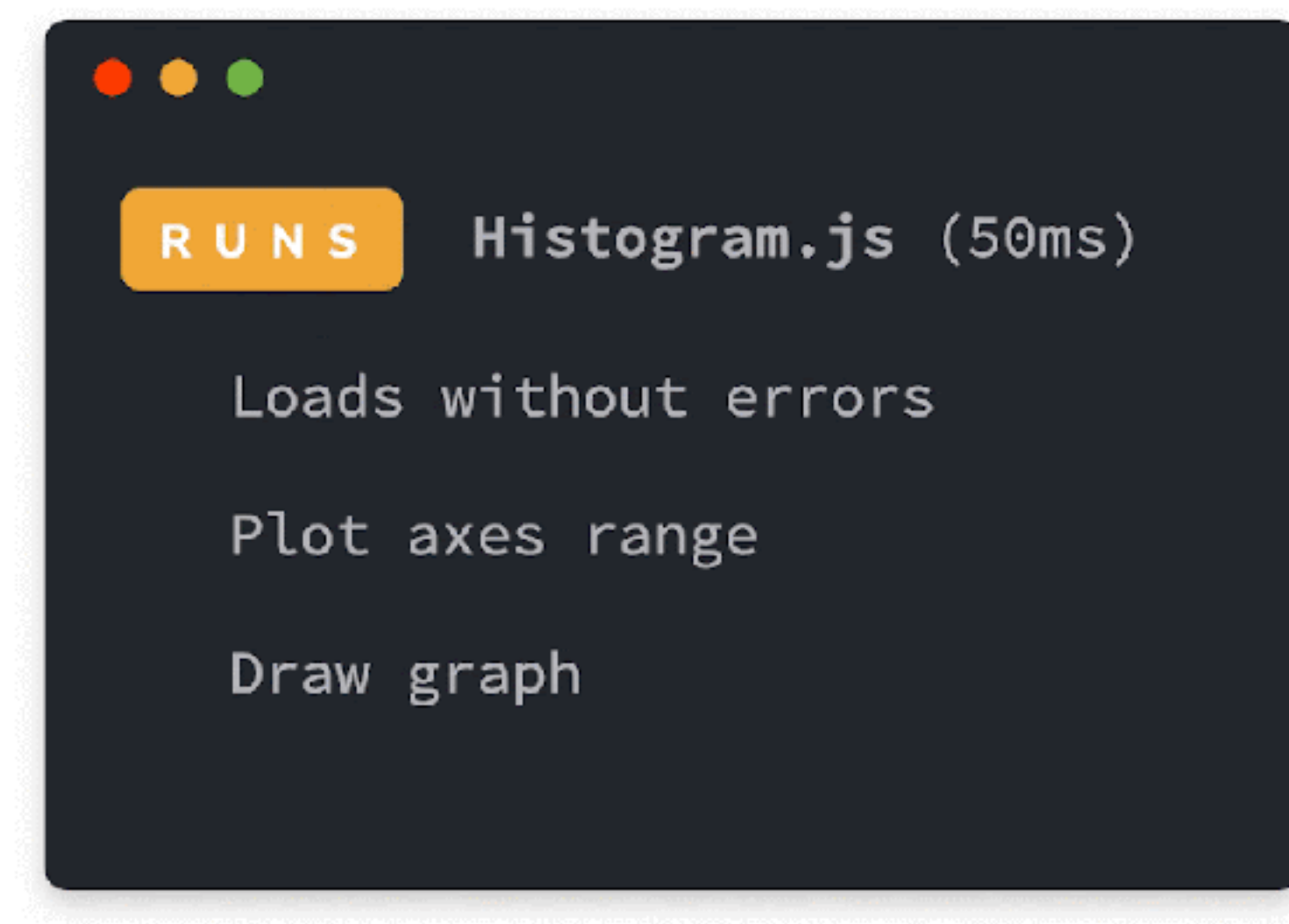
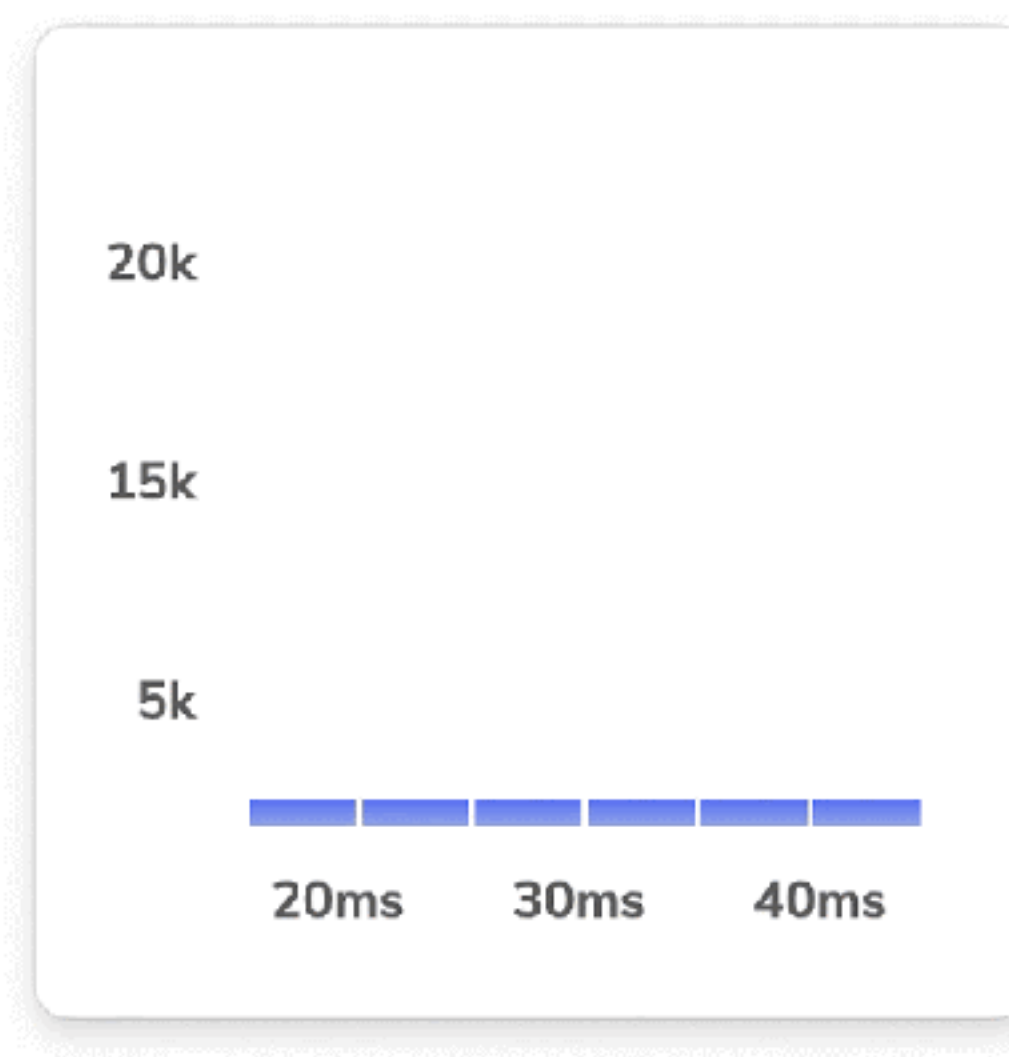
✓ fireEvent.click(within(<div#root>).getByRole("button", { name: "Delete Customer" })))

DialogTest.stories.js

THE STORYBOOK WAY

Interaction tests

- ✍️ Write tests in your stories file
- 🐙 Powered by Testing Library & Jest
- ⚙️ Interactions panel visualizes all steps
- ▶️ Step through the interactions
- 🐛 Debug tests in the browser



Storybook wrapper for Testing Library

```
import { within, userEvent } from '@storybook/testing-library'
import { SearchForm } from './SearchForm'

export default { component: SearchForm }

export const Submitted = {
  play: async ({ canvasElement }) => {
    const canvas = within(canvasElement);

    await userEvent.type(canvas.getByRole('searchbox'), 'query');
    await userEvent.click(canvas.getByRole('button', { name: 'Search' }));
  }
}
```

Play function

Simulated events

Storybook

Find components

ADDONS

Interactions

AccountForm

Demo

Wait For

Standard

Standard Email Filled

Standard Email Failed

Standard Email Success

Standard Password Failed

Standard Fail Hover

Verification

Verification Password 1

Verification Password Mismatch

Verification Success

Interaction

MatcherResult

Mdx

MethodCall

Panel

StatusBadge

StatusIcon

Subnav

Storybook Design System

current

Intro

Colors

Typography

AvatarList

Avatar

Badge

Button

Canvas

Docs

Theme

Create an account to join the Storybook community

Email

gert@chromatic

Please enter a correctly formatted email address

Password

CREATE ACCOUNT

RESET

Controls (1)

Actions (1)

Interactions

Story

Tests

Accessibility

✓

userEvent.type

within

getByTestId

email

gert@chromatic

✓

userEvent.type

within

getByTestId

password1

supersecret

✓

userEvent.click

within

getByRole

button

{ name: /create account/i }

Runs in any browser

Interactions log

Storybook wrapper for Jest

```
import { expect } from '@storybook/jest'
import { within, userEvent } from '@storybook/testing-library'
import { SearchForm } from './SearchForm'

export default { component: SearchForm }

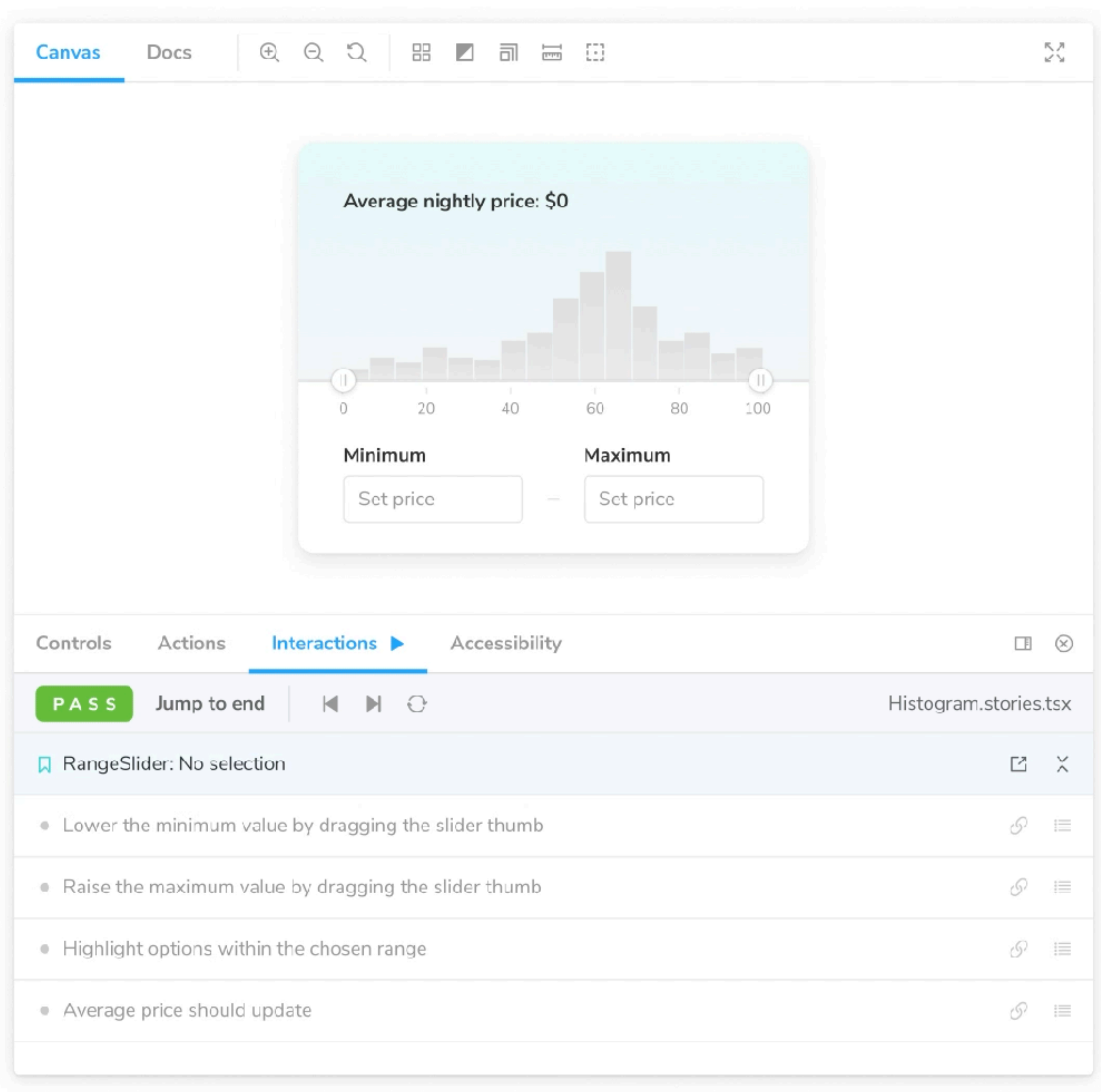
export const Submitted = {
  play: async ({ args, canvasElement }) => {
    const canvas = within(canvasElement);

    await userEvent.type(canvas.getByRole('searchbox'), 'query');
    await userEvent.click(canvas.getByRole('button', { name: 'Search' }));

    await expect(args.onSubmit).toHaveBeenCalled('query');
  }
}
```

Assertion

Automatic spy on actions



Debug with playback controls

- Interactions panel visualizes steps
- Step through the interactions
- Debug tests in the browser

Storybook

Find components

ADDONS

Interactions

AccountForm

Demo

Wait For

Standard

Standard Email Filled

Standard Email Failed

Standard Email Success

Standard Password Failed

Standard Fail Hover

Verification

Verification Password 1

Verification Password Mismatch

Verification Success

Interaction

MatcherResult

Mdx

MethodCall

Panel

StatusBadge

StatusIcon

Subnav

Storybook Design System

Intro

Colors

Typography

AvatarList

Avatar

Badge

Button

Canvas

Docs

Theme

Storybook

Create an account to join the Storybook community

Email

gert@chromatic

Please enter a correctly formatted email address

Password

button.css-yhywg7 167 x 38

CREATE ACCOUNT

RESET

Controls (1)

Actions (1)

Interactions

Story

Test

Accessibility

PASS

Scroll to end

addon-interactions.stories.tsx

✓ userEvent.type(within(<div#root>).getByTestId("email"), "gert@chromatic")

✓ userEvent.type(within(<div#root>).getByTestId("password1"), "supersecret")

✓ userEvent.click(within(<div#root>).getByRole("button", { name: /create account/i })))

✓ within(<div#root>).findByText("Please enter a correctly formatted email address")

✓ expect(action(onSubmit)).not.toHaveBeenCalled()

Elements

Console

<div class="css-1cac0bh"> flex

<button data-testid="submit" aria-disabled="false" type="submit" class="css-yhywg7">Create Account</button>

button.css-yhywg7

Styles

Computed

Layout

Event Listeners

DOM Breakpoints

Filter

element.style {

.css-yhywg7 {

background-color: transparent;

border: 0 none;

outline: none;

appearance: none;

font-weight: 500;

font-size: 12px;

flex-basis: 50%;

cursor: pointer;

padding: 11px 16px;

border-radius: 4px;

text-transform: uppercase;

margin-right: 8px;

background-color: #1EA7FD;

color: #FFFFFF;

opacity: 1;

box-shadow: rgb(30 167 253 / 10%) 0 0 1px inset;

button, input, textarea, select {

font-family: inherit;

font-size: inherit;

box-sizing: border-box;

Browser DevTools

Inspect element

Time travel debugging

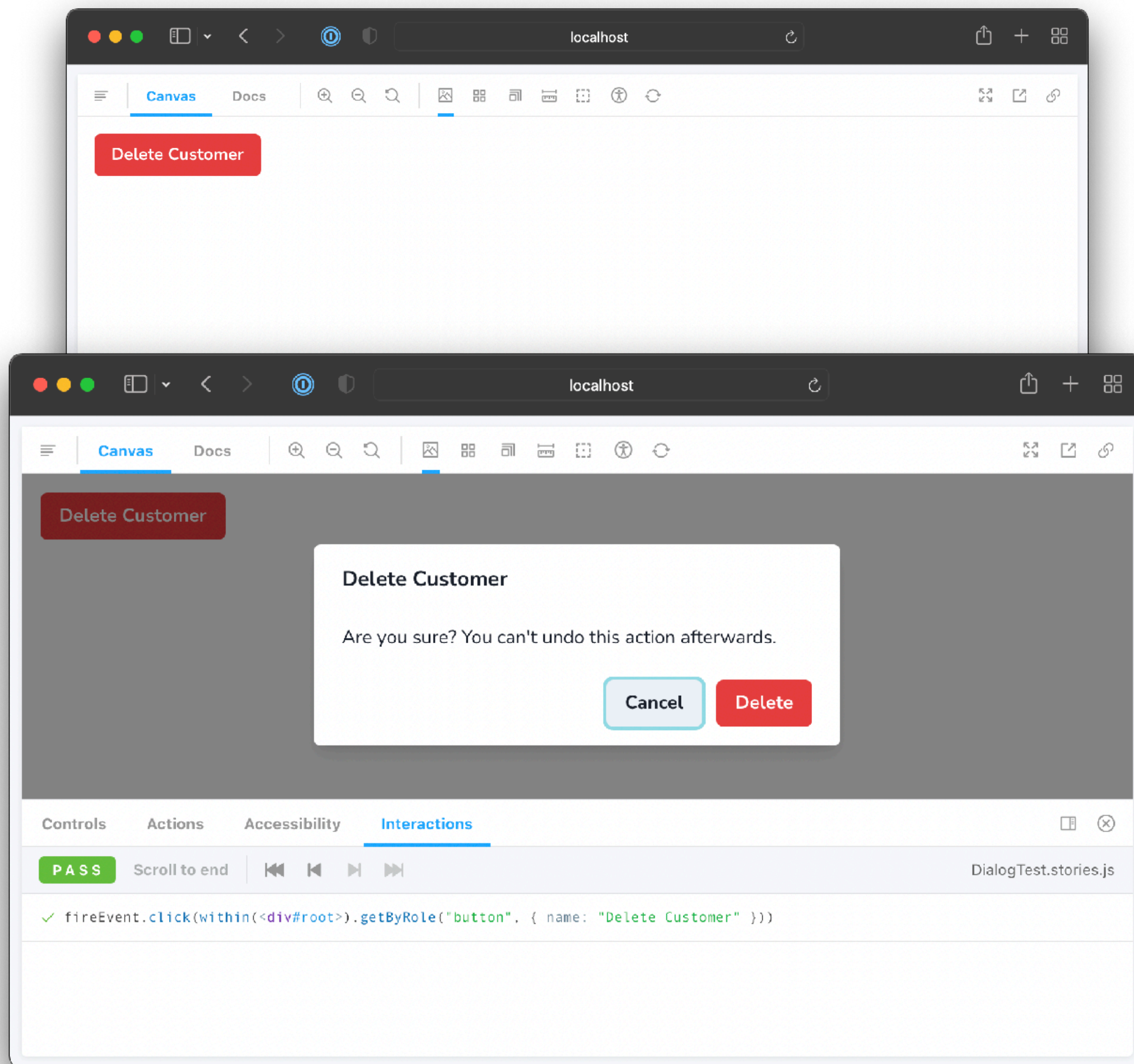
```
$ test-storybook
page loaded in 640ms.
page loaded in 636ms.
page loaded in 655ms.
page loaded in 951ms.
PASS browser: chromium src/LoginScreen.stories.js
PASS browser: chromium src/components/Task.stories.js
PASS browser: chromium src/components/TaskList.stories.js
PASS browser: chromium src/InboxScreen.stories.js

Test Suites: 4 passed, 4 total
Tests: 15 passed, 15 total
Snapshots: 0 total
Time: 4.737 s, estimated 6 s
Ran all test suites.
🌟 Done in 7.79s.

~/Documents/chromatic/ui-testing-guide-code test-runner* 8s
> █
```

Execute tests with test runner

- 🎭 Powered by Jest and Playwright
- ⚡ Run tests in parallel via command line
- 👁 Watch mode, filters, etc.



Test runner converts stories into tests

👁 Checks for rendering errors

🎮 Verifies interactions and assertions



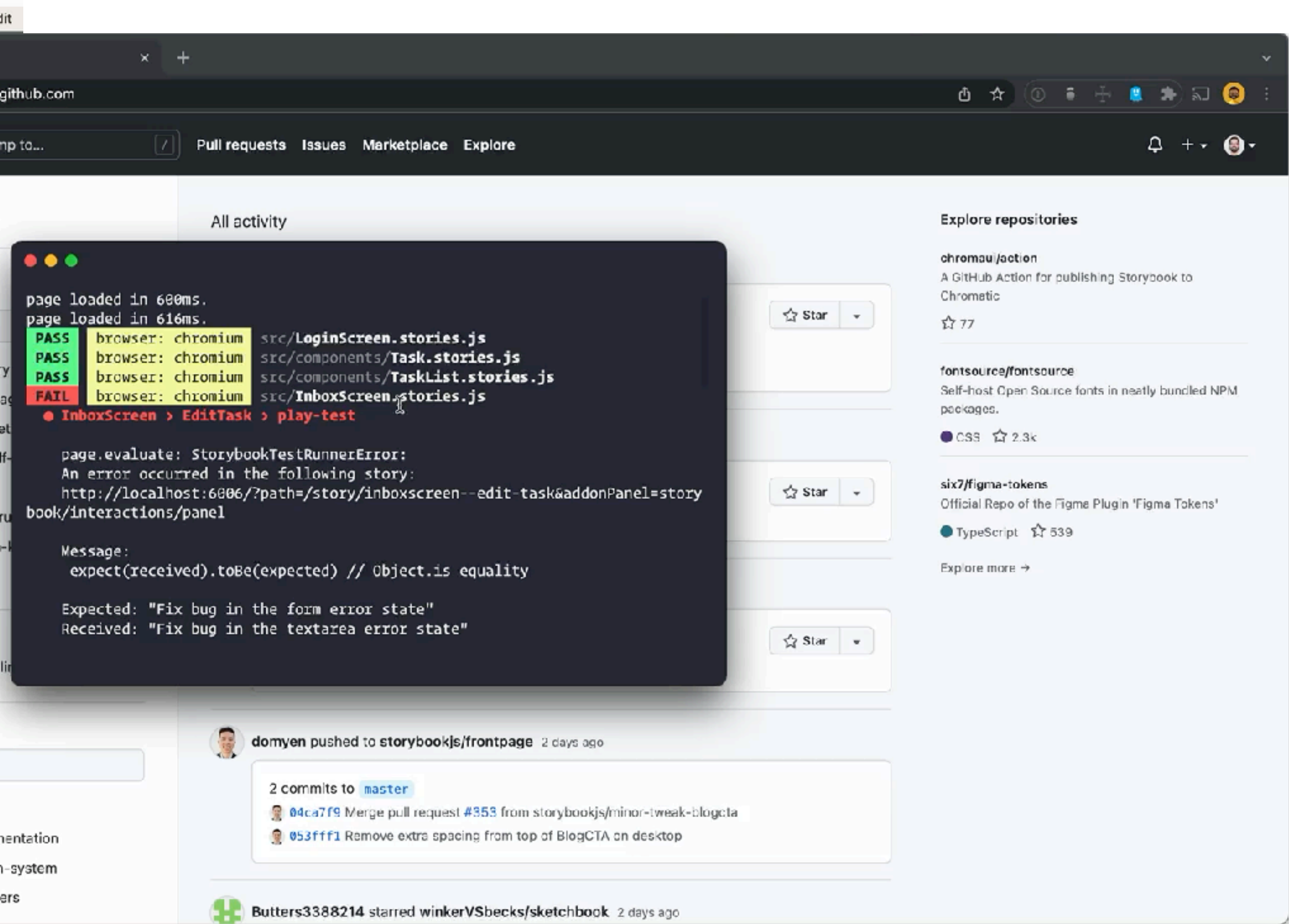
```
~/Documents/chromatic/ui-testing-guide-code test-runner
> yarn test-storybook --browsers chromium firefox webkit
yarn run v1.22.17
$ test-storybook --browsers chromium firefox webkit
page loaded in 704ms.
page loaded in 685ms.
page loaded in 1054ms.
page loaded in 904ms.
page loaded in 1067ms.
page loaded in 1984ms.
page loaded in 1979ms.
```

```
RUNS browser: chromium src/InboxScreen.stories.js
RUNS browser: firefox src/InboxScreen.stories.js
RUNS browser: webkit src/InboxScreen.stories.js
RUNS browser: chromium src/components/TaskList.stories.js
RUNS browser: firefox src/components/TaskList.stories.js
RUNS browser: webkit src/components/TaskList.stories.js
RUNS browser: chromium src/components/Task.stories.js
```

```
Test Suites: 0 of 4 total
Tests:      0 total
Snapshots:  0 total
Time:       5 s, estimated 9 s
```

Cross-browser testing

Powered by Playwright so you can run cross-browser tests in parallel.



Click to debug

Launch and inspect a failing test
in the browser

Works both locally and in CI

```
.storybook/test-runner.js

module.exports = {
  setup() {
    // ...
  },
  async preRender(page, context) {
    // ...
  },
  async postRender(page, context) {
    // ...
  },
};
```

EXPERIMENTAL

Testing hooks

Customize or extend the test runner.

For example, capture snapshots or run accessibility checks.

.storybook/test-runner.js

```
const {
  injectAxe,
  checkA11y,
} = require('axe-playwright');

const config = {
  async preRender(page, context) {
    await injectAxe(page);
  },
  async postRender(page, context) {
    await checkA11y(page);
  },
};

module.exports = config;
```

EXPERIMENTAL

Accessibility testing with hooks

Can inject **axe-playwright** into each story and run accessibility checks.

\$ node bin/tes

PASS

browse

PASS

browse

PASS

browse

Test Suites: 3

Tests: 6

Snapshots: 1

Time: 3

Ran all test s

Coverage file

File

All files

atoms

Button.js

molecules

Header.js

pages

Page.js

🌟 Done in 5.

All files / atoms Button.js

85.71% Statements 6/7 66.66% Branches 4/6 100% Functions 1/1 85.71% Lines 6/7

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

import React from "react";

import PropTypes from "prop-types";

import "../button.css";

/**

* Primary UI component for user interaction

*/

13x export const Button = ({ primary, backgroundColor, size, label, ...props }) => {

21x const mode = primary

? "storybook-button--primary"

: "storybook-button--secondary";

21x if (props.loading) {

return <div>Loading...</div>

}

21x return (

<button

type="button"

className={["storybook-button", `storybook-button--\${size}`, mode].join(

" "

)}

style={backgroundColor && { backgroundColor }}

{...props}

>

{label}

</button>

);

};

EXPERIMENTAL

Code coverage reports

Displays what part of the code is executed (covered) after the tests run

Code coverage

Generated report can be used by external tools which can provide automated checks

https://github.com/codecentric/cxf-spring-boot-starter/pull/100#partial-pull-merging


Open

Update dependency de.codecentric:cxf-spring-boot-starter-reactor to v2.4.1 #100
renovate wants to merge 1 commit into master from renovate/de.codecentric-cxf-s...

codecov-io commented 8 days ago

Codecov Report

Merging #100 (0518a7c) into master (d8562f3) will decrease coverage by 0.20% .
The diff coverage is n/a .



	master	#100	+/-
- Coverage	57.57%	57.37%	-0.21%
+ Complexity	122	121	-1
Files	29	29	
Lines	495	495	
Branches	30	30	

Codecov App 12:54PM

Coverage for your project name increased

✓ All checks have passed

✓

codecov/project/frontend

85% ...

✓

codecov/project/backend

1.4% ...

✓






Test (push)

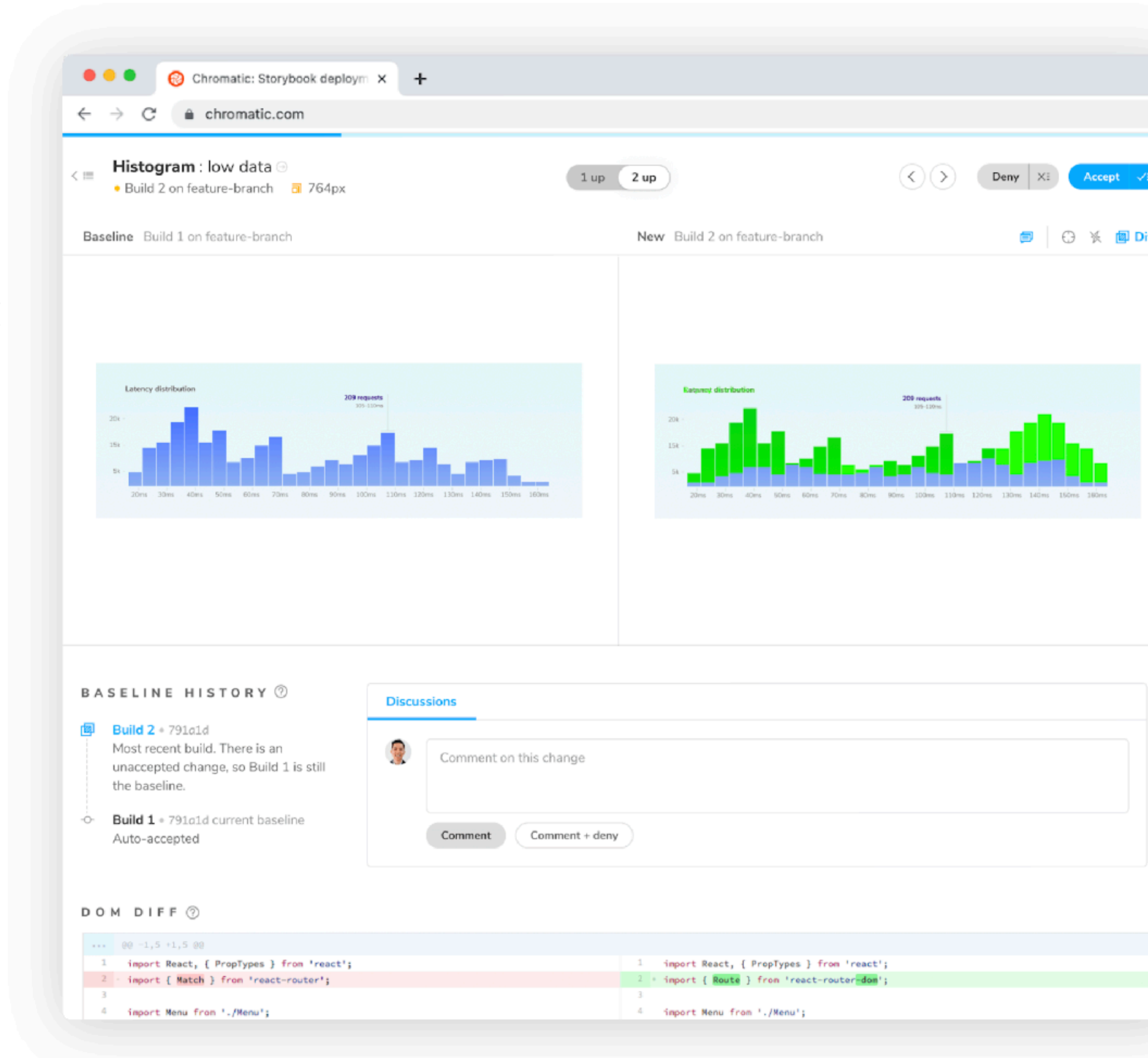
Successful in 32s

Merge Pull request



Automated workflows for Storybook to help teams ship UI faster

-  **Publish** and share your Storybook
-  **Approve** and track changes over time
-  **Visually review** pull requests
-  **Gather feedback** from stakeholders
-  **Sync status** to GitHub, GitLab, Bitbucket



The screenshot displays the Chromatic web interface for a Storybook deployment. The browser tab is titled "Chromatic: Storybook deploym x" and the address bar shows "chromatic.com". The main heading is "Histogram : low data" with a sub-header "Build 2 on feature-branch" and a resolution indicator "764px". Navigation controls include "1 up", "2 up", and buttons for "Deny" and "Accept". Below the heading, there are tabs for "Baseline" and "Build 1 on feature-branch", and a "New" tab for "Build 2 on feature-branch". The interface features two side-by-side histograms labeled "Latency distribution". The left histogram shows a distribution of latency values (20ms to 160ms) with a peak around 40ms. The right histogram shows a similar distribution but with a higher peak around 40ms. Below the histograms, there is a "BASELINE HISTORY" section showing two builds: "Build 2 * 791d1d" (Most recent build. There is an unaccepted change, so Build 1 is still the baseline.) and "Build 1 * 791d1d current baseline" (Auto-accepted). To the right of the history is a "Discussions" section with a comment box and buttons for "Comment" and "Comment + deny". At the bottom, there is a "DOM DIFF" section showing a comparison of code changes between the two builds. The code is displayed in a diff format, with changes highlighted in red and green.

Chromatic: Storybook deploym x

chromatic.com

Histogram : low data

Build 2 on feature-branch 764px

1 up 2 up

Deny Accept

Baseline Build 1 on feature-branch New Build 2 on feature-branch

Latency distribution

209 requests 200-120ms

Latency distribution

209 requests 200-120ms

BASELINE HISTORY

Build 2 * 791d1d
Most recent build. There is an unaccepted change, so Build 1 is still the baseline.

Build 1 * 791d1d current baseline
Auto-accepted

Discussions

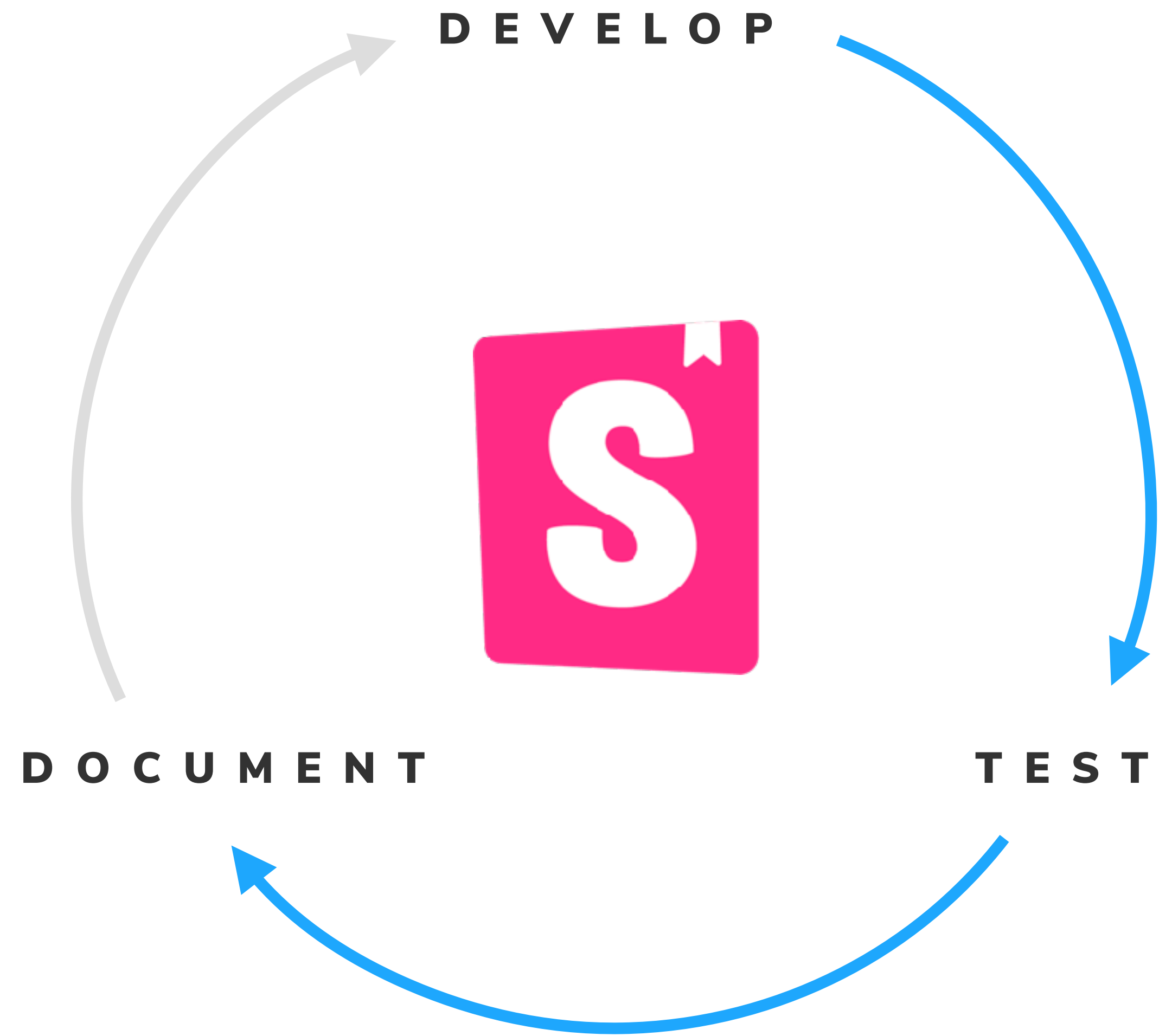
Comment on this change

Comment Comment + deny

DOM DIFF

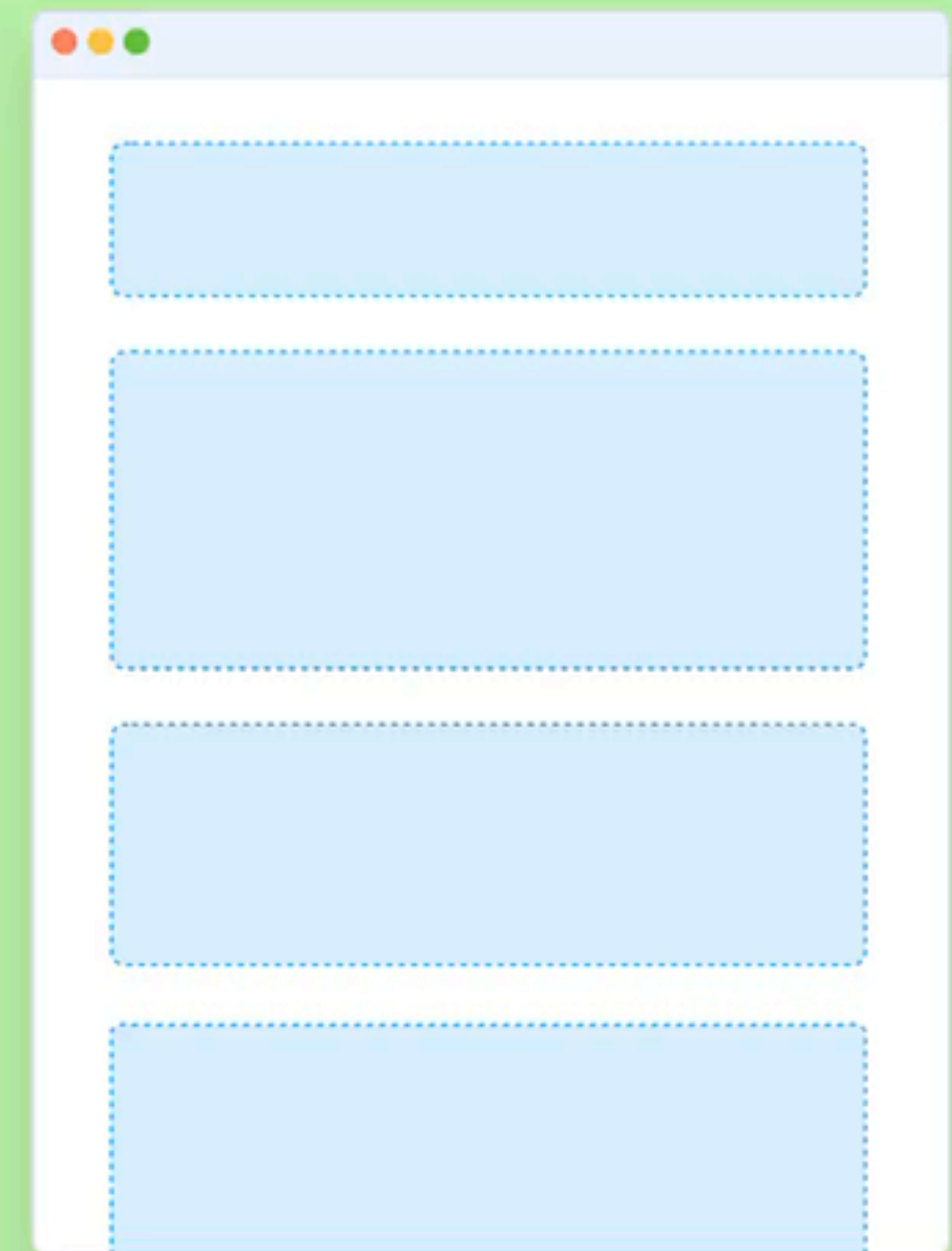
```
... @@ -1,5 +1,5 @@
1 import React, { PropTypes } from 'react';
2 - import { Match } from 'react-router';
3
4 import Menu from './Menu';
```

```
1 import React, { PropTypes } from 'react';
2 + import { Route } from 'react-router-dom';
3
4 import Menu from './Menu';
```

Document

Kickstart your project's UI documentation with MDX 2 support, new architecture, streamlined UX, and readymade doc blocks.



Pages

Document design tokens,
onboard users, and more

```
import { Meta, ColorPalette, ColorItem } from '@storybook/blocks'
import { TabsState } from '@storybook/components'
import { lightTheme, darkTheme } from '../styles/theme'
import { getColors } from './utils'

<Meta title="Design System/Colors" />

# Colors

<TabsState initial="light">
  <div id="light" title="Light mode">
    <ColorPalette>
      {getColors(lightTheme).map(({ name, value }) => (
        <ColorItem key={name} title={`theme.colors.${name}`} subtitle={value} />
      ))}
    </ColorPalette>
  </div>
  <div id="dark" title="Dark mode">
    <ColorPalette>
      {getColors(darkTheme).map(({ name, value }) => (
        <ColorItem key={name} title={`theme.colors.${name}`} subtitle={value} />
      ))}
    </ColorPalette>
  </div>
</TabsState>
```

MealDrop

Find components

GETTING STARTED

With Mealdrop

DESIGN SYSTEM

Colors

Typography

COMPONENTS

AnimatedIllustration

Badge

Button

Category

ErrorBlock

Fcoter

Header

Icon

IconButton

Logo

Modal

PageSection

RestaurantCard

Review

OrderSummary

ShoppingCartMenu

Sidebar

Spinner

TopBanner

Fcrm

Typography

PAGES

CategoryDetailPage

CategoryListPage

CheckoutPage

HomePage

RestaurantDetailPage

SuccessPage

Colors

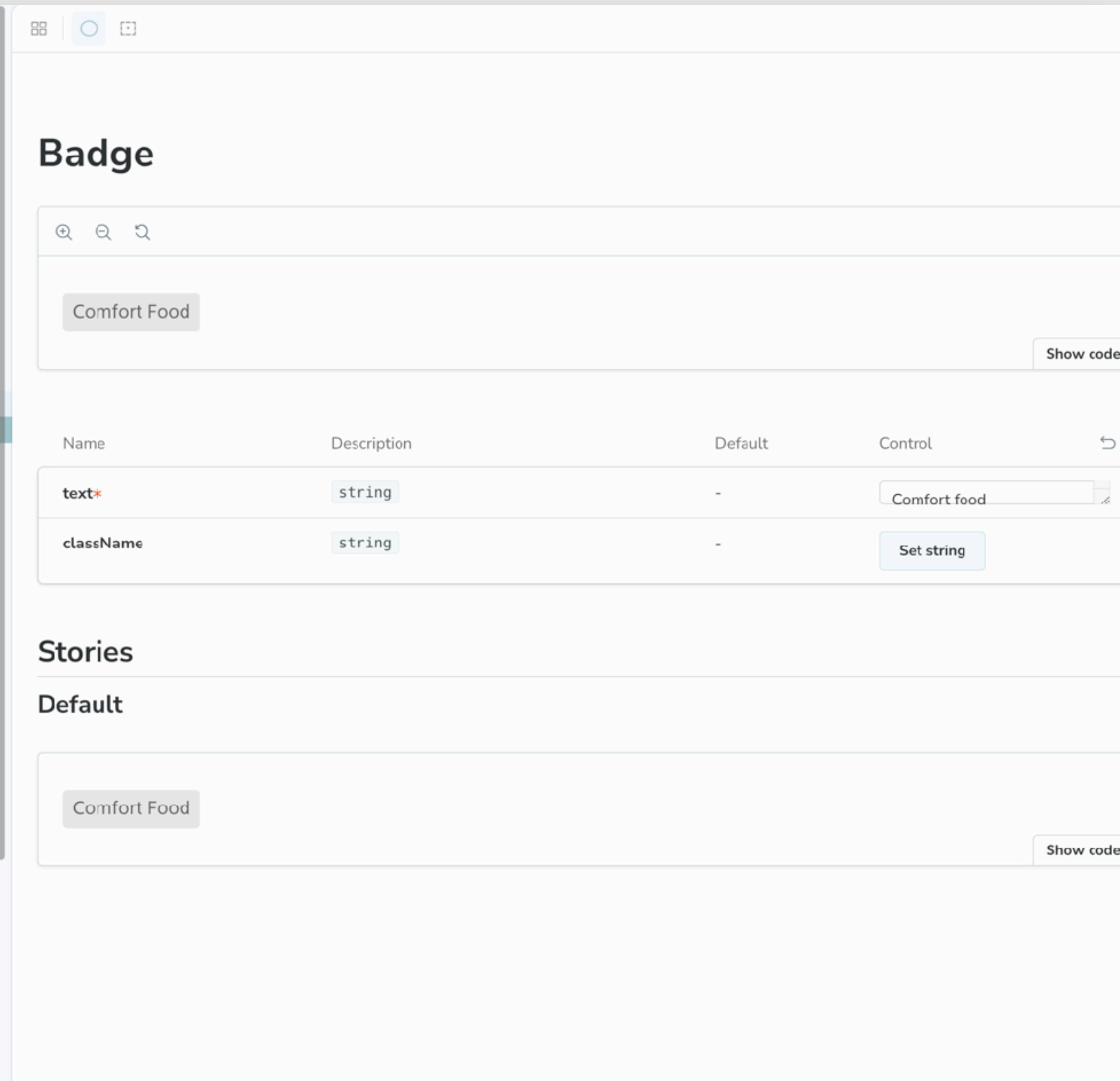
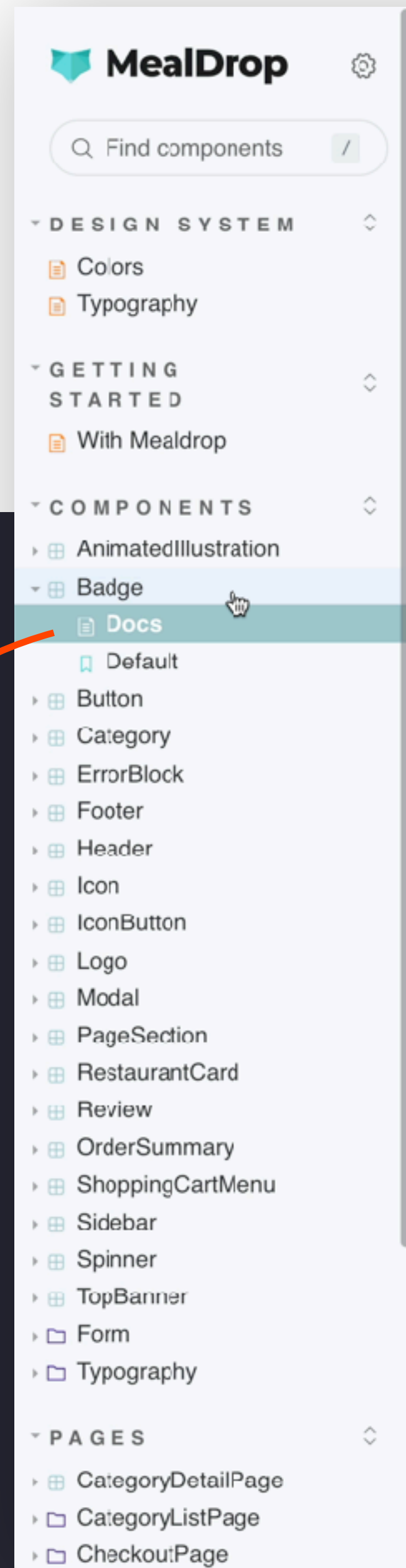
Light modeDark mode

Name	Swatches
theme.colors.accentText accentText	<div>#202020</div>
theme.colors.badgeBackground badgeBackground	<div>#E9E9E9</div>
theme.colors.badgeText badgeText	<div>#636363</div>
theme.colors.bannerBackground bannerBackground	<div>#B1DDE4</div>
theme.colors.black black	<div>#202020</div>
theme.colors.buttonClear buttonClear	<div>transparent</div>
theme.colors.buttonClearHover buttonClearHover	

Autodocs

Automatically generate
documentation pages

```
const config: StorybookConfig = {
  stories: [
    '../src/docs/Introduction.stories.mdx',
    '../src/docs/*.stories.mdx',
    '../src/**/*.stories.@(js|jsx|ts|tsx)',
    '../src/**/*.mdx',
  ],
  addons: [
    '@storybook/addon-essentials',
    '@storybook/addon-interactions',
  ],
  staticDirs: ['../public'],
  framework: {
    name: '@storybook/react-vite',
    options: {},
  },
  docs: {
    autodocs: true,
  },
}
```



Fully custom MDX story

The image shows a web browser on the left and a VS Code editor on the right, illustrating a fully custom MDX story for a Button component.

Browser View (Left):

- URL: `localhost:6006/?path=/docs/components-button--docs`
- Component List (Left Sidebar):
 - Iconography
 - COMPONENTS
 - Button
 - Docs (selected)
 - Default
 - Disabled
 - Clear
 - Icon
 - Icon And Text
 - AnimatedIllustration
 - Badge
 - Category
 - ErrorBlock
 - Footer
 - Header
 - Icon
 - IconButton
 - Logo
 - Modal
 - PageSection
 - RestaurantCard
 - Review
 - OrderSummary
 - ShoppingCartMenu
 - Sidebar
 - Spinner
 - TopBanner
 - Form
 - Typography
 - PAGES
 - CategoryDetailPage
 - CategoryListPage
 - CheckoutPage
 - HomePage
 - RestaurantDetailPage
 - SuccessPage
 - USER FLOWS
 - App
 - TEMPLATES
 - PageTemplate

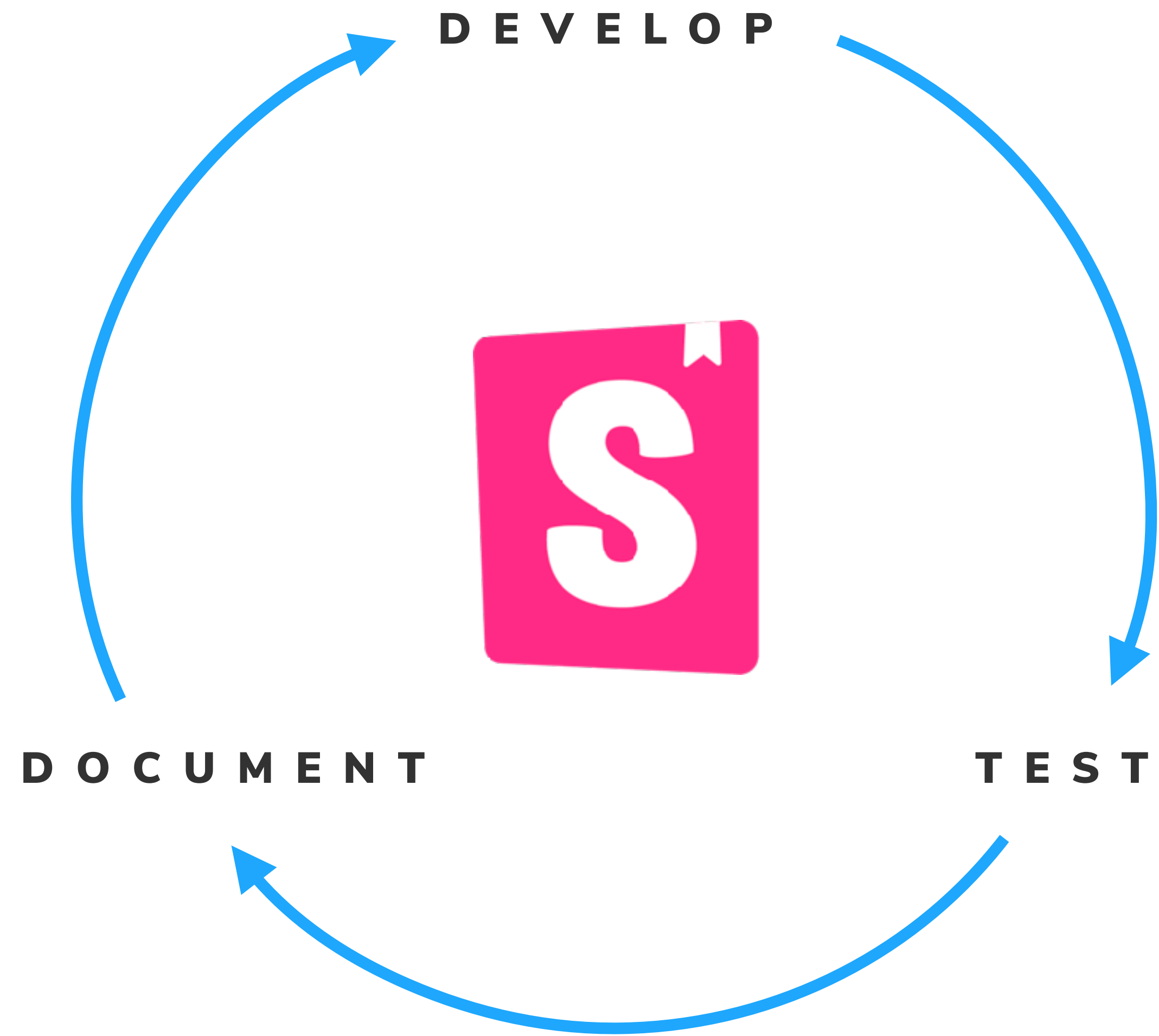
- Component Title: **Button**
- Description: Primary UI component for user interaction
- Text: The button component is used to trigger an action or event, such as submitting a form, opening a dialog, canceling an action, or performing a delete operation.
- Section: **Button variants**
- Visuals: A row of five button variants: a plain white button, a solid black button, a solid grey button, a black button with a white shopping cart icon, and a black button with a white shopping cart icon and text "Order € 8".

VS Code Editor View (Right):

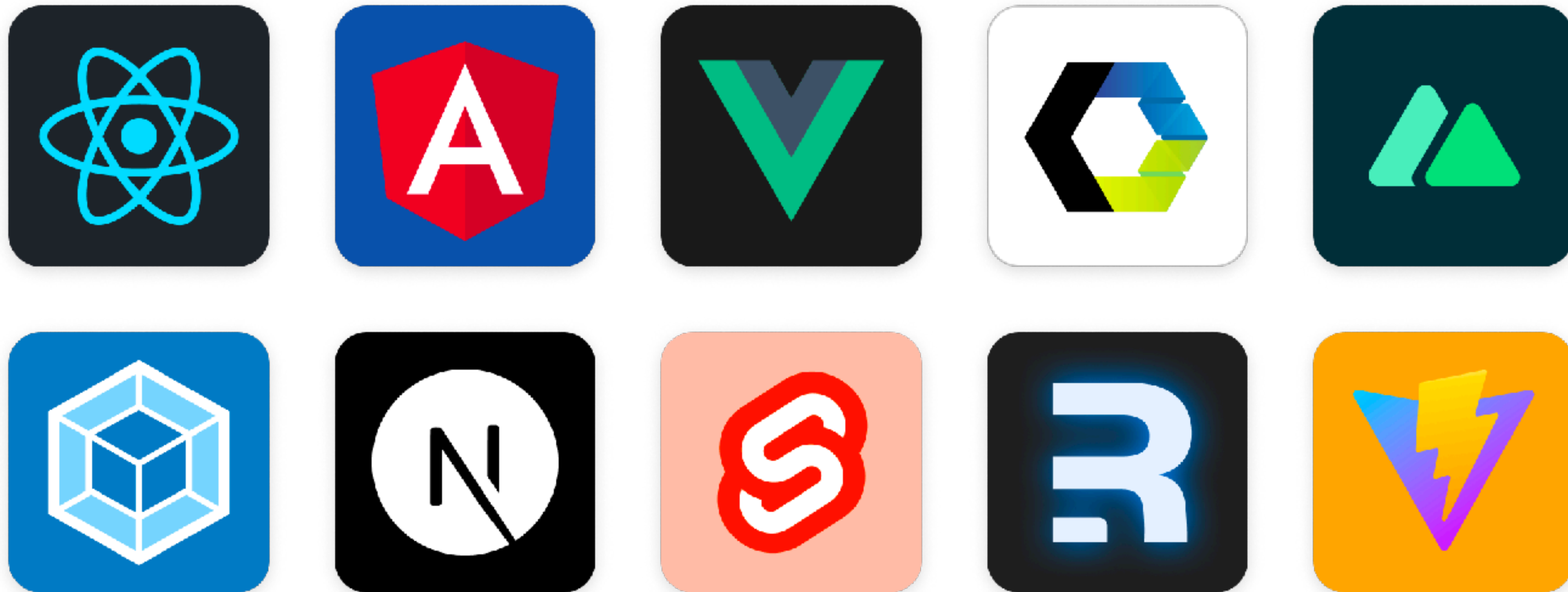
- File Explorer (Left Sidebar):
 - MEALDROP
 - components
 - AnimatedIllustration
 - Badge
 - Button
 - Button.mdx (selected)
 - Button.stories.tsx
 - Button.test.tsx
 - Button.tsx
 - index.tsx
 - utils.tsx
 - Category
 - ErrorBlock
 - Footer
 - FooterCard
 - forms
 - Header
 - Icon
 - IconButton
 - Logo
 - Modal
 - PageSection
 - RestaurantCard
 - Review
 - ShoppingCart
 - ShoppingCartMenu
 - Sidebar
 - Spinner
 - TopBanner
 - typography
 - Portal.tsx
 - docs
 - helpers
 - hooks
 - pages
 - stub
 - styles
 - templates
 - types
 - App.tsx
 - index.tsx
 - logo.svg
 - OUTLINE
 - TIMELINE

- Editor (Right):
- File: `Button.mdx`
- Code:

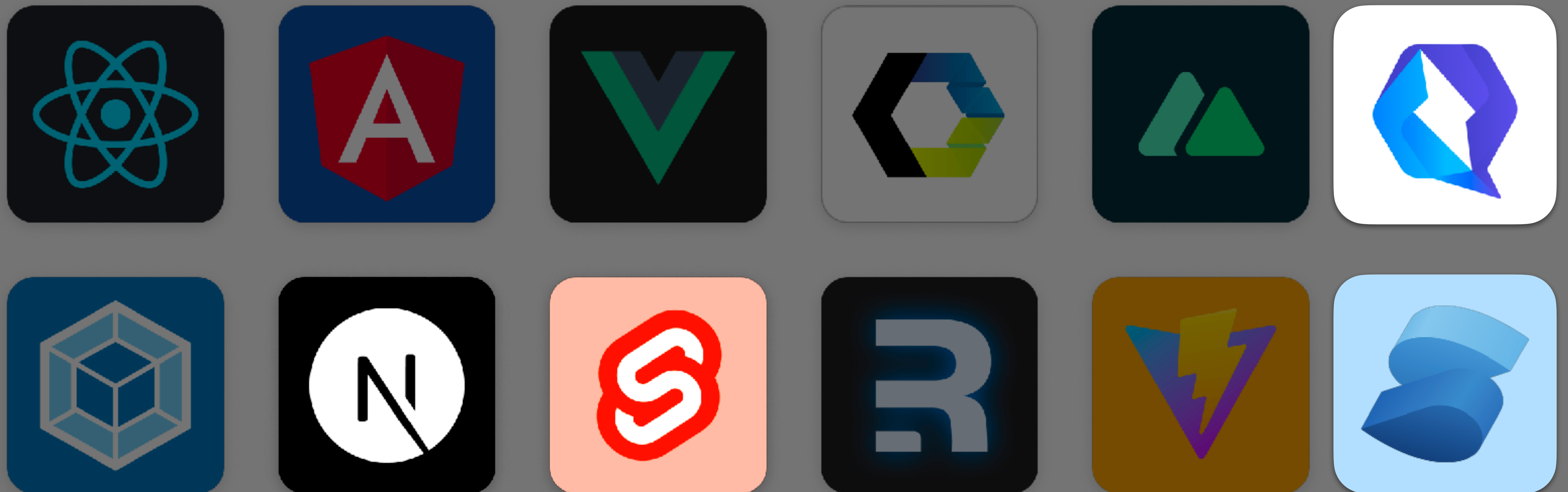
```
1 import { Meta, Description } from '@storybook/blocks'
2 import { Figma } from "storybook-addon-designs/blocks";
3 import { Grid } from './utils'
4 import * as ButtonStories from './Button.stories'
5
6 # Button
7
8 <Meta of={ButtonStories} />
9 <Description of={ButtonStories} />
10
11 > The button component is used to trigger an action or event,
12   such as submitting a form, opening a dialog, canceling an action,
13   or performing a delete operation.
14
15 ### Button variants
16
17 <Grid of={ButtonStories} />
18
```



Toolset for the community to build their own Storybook integrations

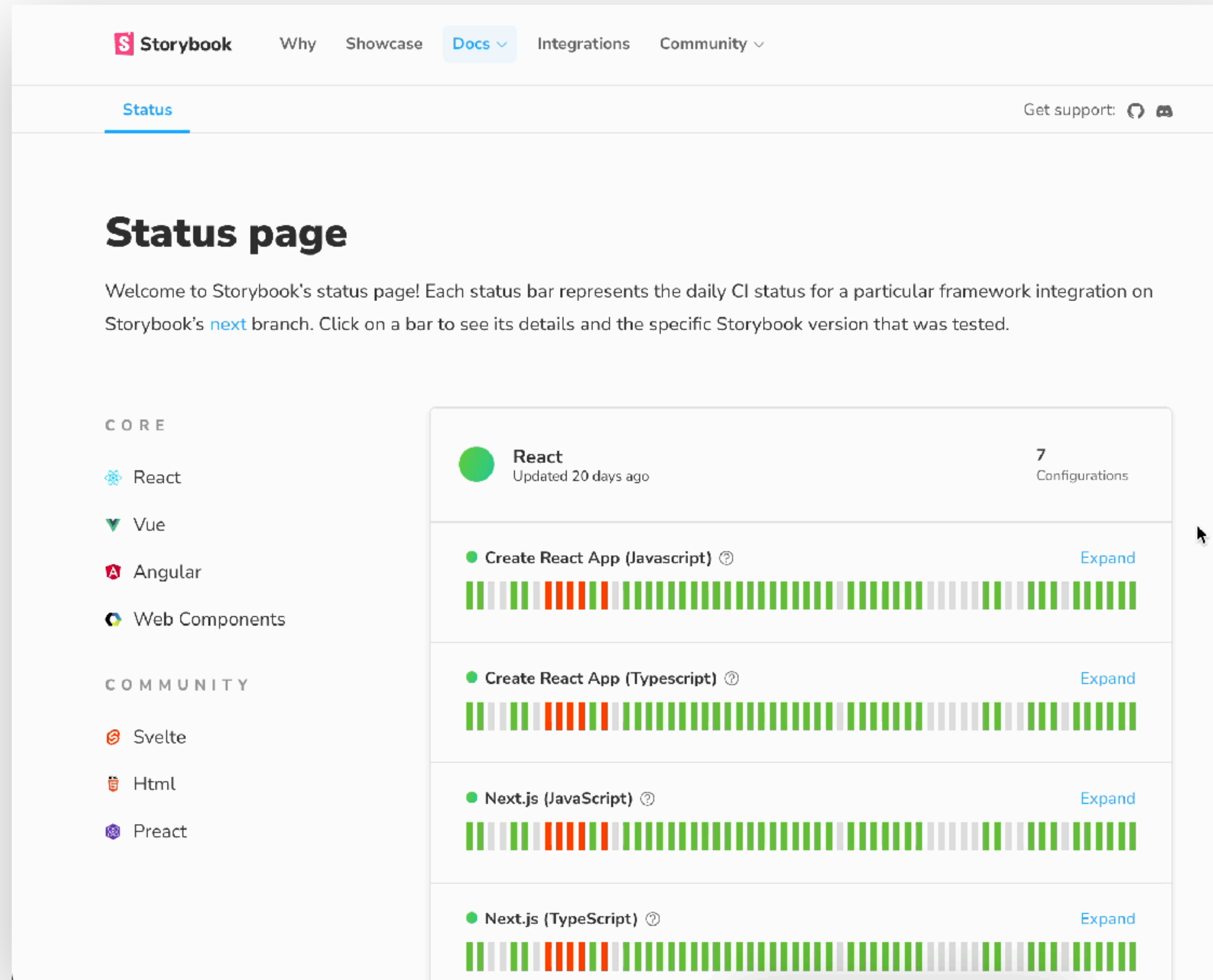


Toolset for the community to build their own Storybook integrations



Ecosystem CI

Dozens of framework configs
heavily tested daily



Storybook 7 brings
hundreds of improvements

You can try it **today**

Initialize a new Storybook in your project:

```
npx storybook@latest init
```

Upgrade + migrate an existing Storybook:

```
npx storybook@latest upgrade
```



GOTO
Guide



Remember to
rate this session

THANK YOU!



#GOTOams