DESIGN SYSTEMS & ACCESSIBILITY: THE GOOD, THE BAD AND THE FRUSTRATED UNICORN.

role=drinks · Amsterdam, NL · June 15, 2019

Design, develop, and deliver

Use Atlassian's end-to-end design language to create straightforward and beautiful experiences.

B Brand

Foundations, including personality, writing style, color, and type.

M Marketing

How color, type, and illustration are used in marketing.

P Product

Foundations, components, and patterns for building applications.

P Design principles

The philosophies behind our approach towards design at Atlassian.

M Atlassian logos

Why prototyping is a must

M Our illustrations

A Atlaskit

When discussion about design system, this is one of the regular exemple mentioned by people: Atlassian Design.

atlassian.design

Packages
Overview
CORE
Analytics next
Analytics
Avatar group
Avatar
Badge
Banner
Blanket
Breadcrumbs
Button
Upgrade guide
Theming guide
Calendar
Checkbox
Examples
Design docs
Changelog

Button

A React component that is a base button.

Install	`yarn add @atlaskit/button`
npm	@atlaskit/button
Source	Bitbucket
Bundle	unpkg.com

LATEST

13.0.8

- Updated dependencies cfc3c8adb3:
  - @atlaskit/docs@8.1.2
  - @atlaskit/checkbox@8.0.2
  - @atlaskit/select@9.1.5
  - @atlaskit/icon@18.0.0

Buttons are used as triggers for actions. They are used in forms, toolbars, dialog footers and as stand-alone action triggers.

Button also exports a button-group component to make it easy to display multiple buttons together.

A really nice design system with an extensive component technical documentation.

ATLASSIAN Design

Overview

FOUNDATIONS

Colors

Iconography

Typography

Writing style

COMPONENTS

Avatars

Badges

Buttons

Date picker

Dropdowns

Forms

Inline dialog

Colors

Where appropriate, we enable people to introduce their own color palettes. Our products adapt intelligently and flexibly to cater for the user's preference. We are also committed to complying with AA standard contrast ratios . To accomplish this, you should choose primary, secondary and extended colors that ensure sufficient color contrast between elements. This allows users with low vision to see and use the interface.

Primary color palette

Our primary palette is comprised of neutrals, white, and blue. These colors are present across most visual elements all the way from marketing to product. If you'd like to use these colors you can download our palette here.

We use N800 primarily for body text and headings. B400, otherwise known as "Pacific Bridge" is used for primary actions and buttons, links, global navigation, indicates progress, and represents authentication. N0 is used for page backgrounds and things that are white.

N800	B400	N0
#172B4D	#0052CC	#FFFFFF

And an even more complete design documentation.

Storybook
Press "/" to search...
0. GENERIC
Welcome
1. QUARKS
Helpers
Typography
Colors
Color palettes
Responsive
Icon Library
2. ATOMS
Checkbox
Label
Headings
Link
Radio Option
Buttons
Icon
Text Input
Validation Messages
3. MOLECULES
Radio Group
Input Group
Canvas
Notes

Color Palettes

Blue Colors Green Colors Generic Colors Neutral Colors Orange Colors Purple Colors Red Colors Skyblue Colors Yellow Colors Brand Colors Semantic Colors

Blue Colors

Name	blue.b30	
Hex	#f4faff	
RGB	rgb(244, 250, 255)	
Usable with white	(1.1:1)	✗
Usable with n800	(20:1)	✓
Usable with EDC color	(3.9:1)	!
Usable with SMS color	(6.7:1)	✓

Name	blue.b40	
Hex	#f0f8ff	
RGB	rgb(240, 248, 255)	
Usable with white	(1.1:1)	✗
Usable with n800	(19.6:1)	✓
Usable with EDC color	(3.8:1)	!
Usable with SMS color	(6.5:1)	✓

Name	blue.b50	
Hex	#e7f3ff	
RGB	rgb(231, 243, 255)	
Usable with white	(1.1:1)	✗
Usable with n800	(18.7:1)	✓
Usable with EDC color	(3.6:1)	!
Usable with SMS color	(6.2:1)	✓

Name	blue.b100	
Hex	#5fafff	
RGB	rgb(95, 175, 255)	
Usable with white	(2.3:1)	✗
Usable with n800	(9.1:1)	✓
Usable with EDC color	(1.8:1)	✗
Usable with SMS color	(3:1)	✗

At the same time, what I am seeing most of the time, or what even my projects are having component libraries more looking like this.

Hi! 👋

I’m Damien.

Hi! 👋

I’m **Damien**.

I am a queer **digital designer**,
specialised in **accessibility**. 🌈

I work for **Castor EDC** in Amsterdam
as a **Design systems & Accessibility Lead**.

Oh, and my pronouns are **they/them/their**.

I am a designer,
and I **write** & show **code** 🙀

Let’s talk about crushing dreams.

Let’s talk about frustrations.

So basically, let’s talk about **design systems** & **accessibility**.

1.

Design Systems & Accessibility: a reality check.

Photo by Balázs Kétyi on Unsplash

Design systems and accessibility improvements have a common point:
it’s a never ending work, and you should not too much time to make it pretty.

Design systems will not make accessibility less or more complex.

Even with amazing component libraries, accessibility can not rely only on it.

On the design side:
your designs should be accessible.

On the component library side:
your codebase should be as accessible as possible.

React is offering a lot
of accessibility features
and support...
but **React will not magically
solve every issues.**

Even with this principle,

so many things can go wrong.

Let’s do a test!

✍️ Is there any user generated content?

♿ Are your teammates trained on accessibility?

🎨 How accessible is the brand colour palette?

🕹️ Do you have a device lab with screen readers?

🎪 Is there a corporate website not using components?

You’re doomed.

GOOD DOOMED OR BAD DOOMED?

Relax, breath, :smile: there is solutions. 🧘‍♀️

2.

How to maintain accessibility in the long run?

I like to consider design systems project as *open-source-within-a-company* projects.

The main flaw with this idea:
You can quickly start working in isolation.

This button is a completely legit one.
This door could be hard to open for a lot of people,
making this button useful.

But is the icon the good one?
Is it really fulfilling its accessible purpose
with this misleading label?

Accessibility is about the global experience.

Accessibility is about details.

Accessibility is about everything.

Within the component library,
you can care about a lot about details.

Outside of the component library,
accessibility is mainly about the bigger picture:
accessibility is about moving users’ focus.

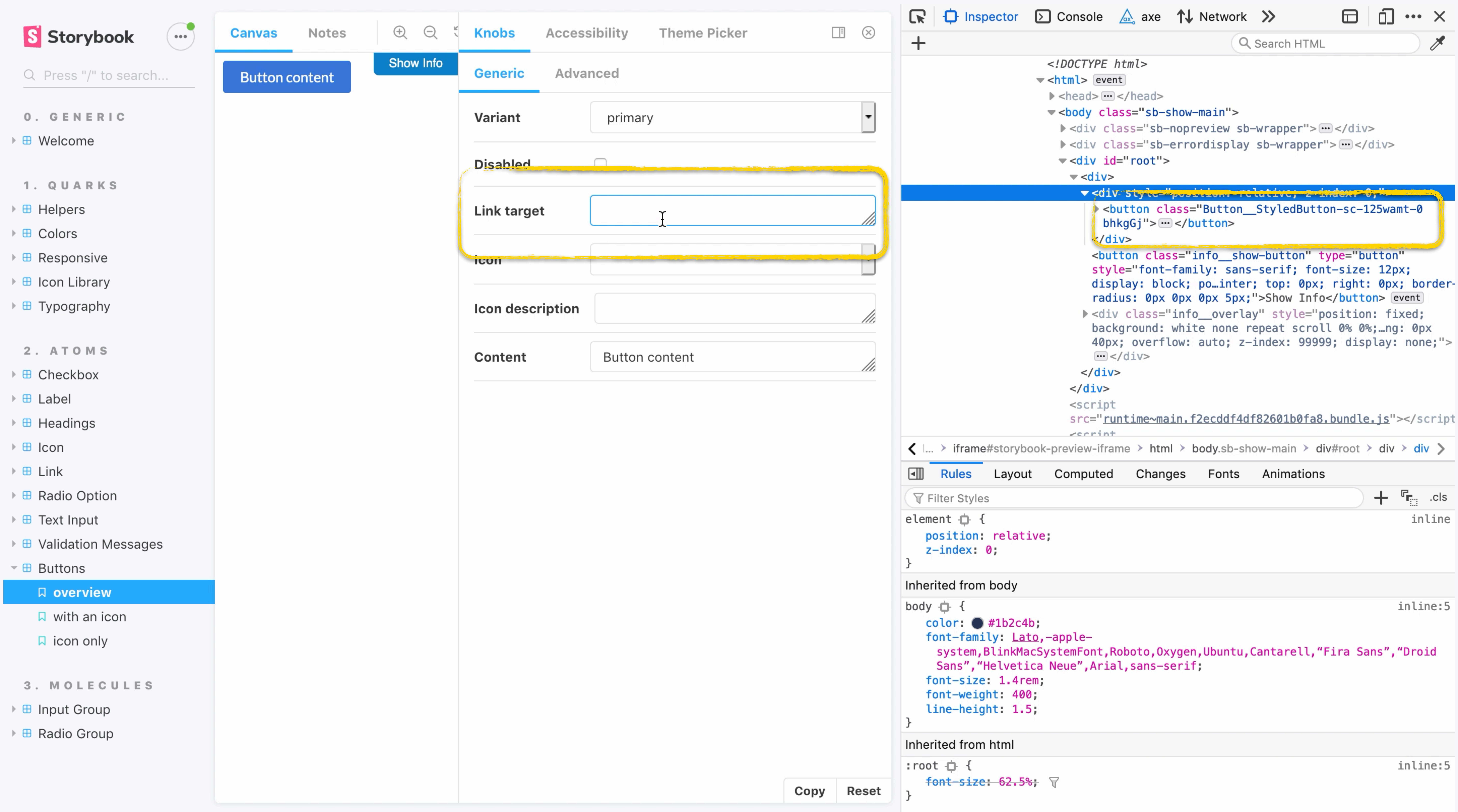
Between components, mainly two challenges: moving focus logically & sharing current state.

What can we do?

First things first:

automise DOM variations as much as possible.

A good component should adapt the markup depending on the content provided, magically ✨



Variant: primary

Disabled: □

Link target:

Icon:

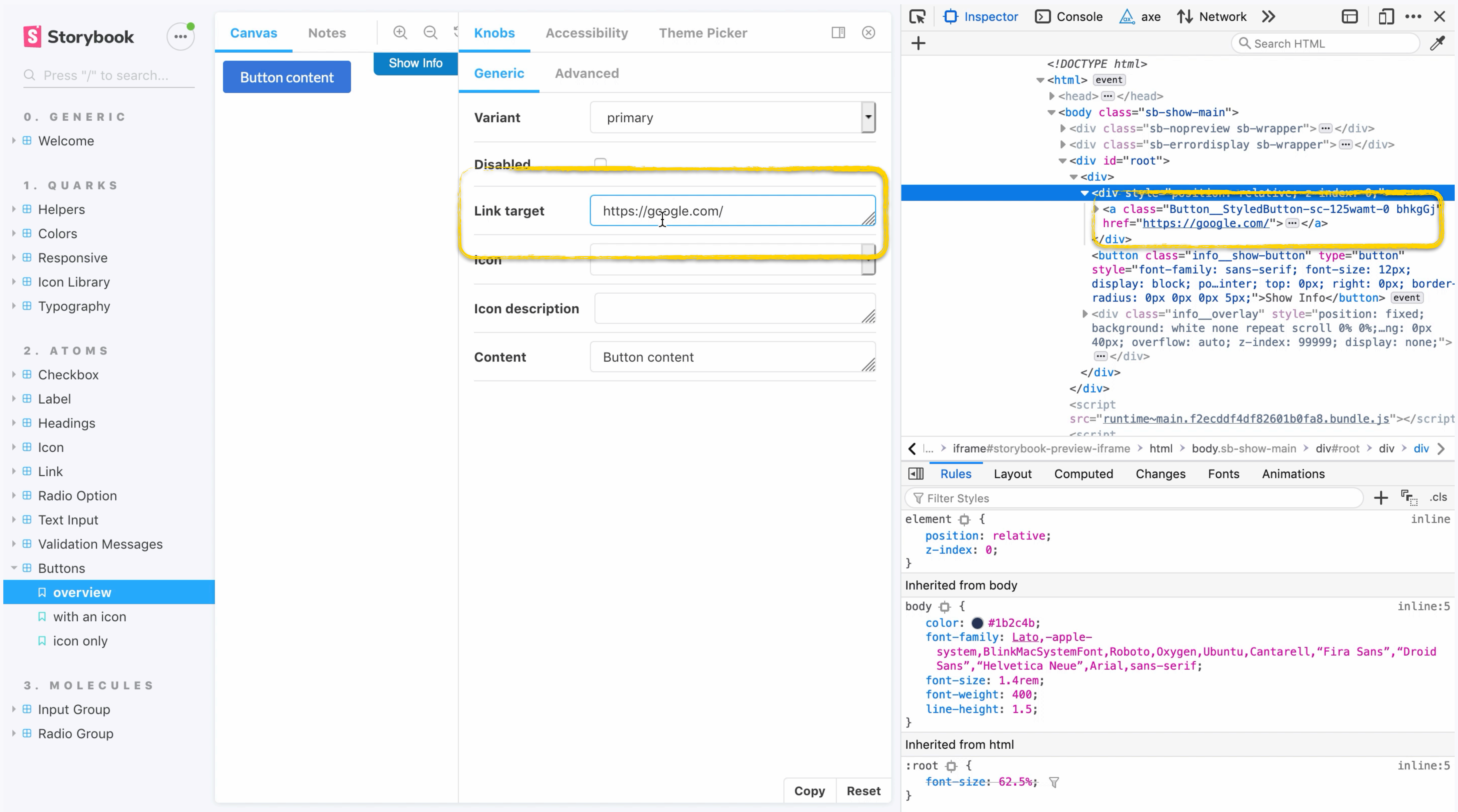
Icon description:

Content: Button content

```
<!DOCTYPE html>
<html> event
  <head>…</head>
  <body class="sb-show-main">
    <div class="sb-nopreview sb-wrapper">…</div>
    <div class="sb-errordisplay sb-wrapper">…</div>
    <div id="root">
      <div>
        <div style="position: relative; z-index: 0;">
          <button class="Button__StyledButton-sc-125wamt-0 bhkgGj">…</button>
        </div>
        <button class="info__show-button" type="button"
        style="font-family: sans-serif; font-size: 12px;
        display: block; po…inter; top: 0px; right: 0px; border-
        radius: 0px 0px 0px 5px;">Show Info</button> event
        <div class="info__overlay" style="position: fixed;
        background: white none repeat scroll 0% 0%;…ng: 0px
        40px; overflow: auto; z-index: 99999; display: none;">
        …</div>
      </div>
    </div>
    <script
    src="runtime~main.f2ecddf4df82601b0fa8.bundle.js"></script
```

I... > iframe#storybook-preview-iframe > html > body.sb-show-main > div#root > div > div

```
element {                                              inline
   position: relative;
   z-index: 0;
}
Inherited from body
body {                                                 inline:5
   color: #1b2c4b;
   font-family: Lato,-apple-
     system,BlinkMacSystemFont,Roboto,Oxygen,Ubuntu,Cantarell,“Fira Sans”,“Droid
     Sans”,“Helvetica Neue”,Arial,sans-serif;
   font-size: 1.4rem;
   font-weight: 400;
   line-height: 1.5;
}
Inherited from html
:root {                                                inline:5
   ~~font-size: 62.5%;~~
}
```



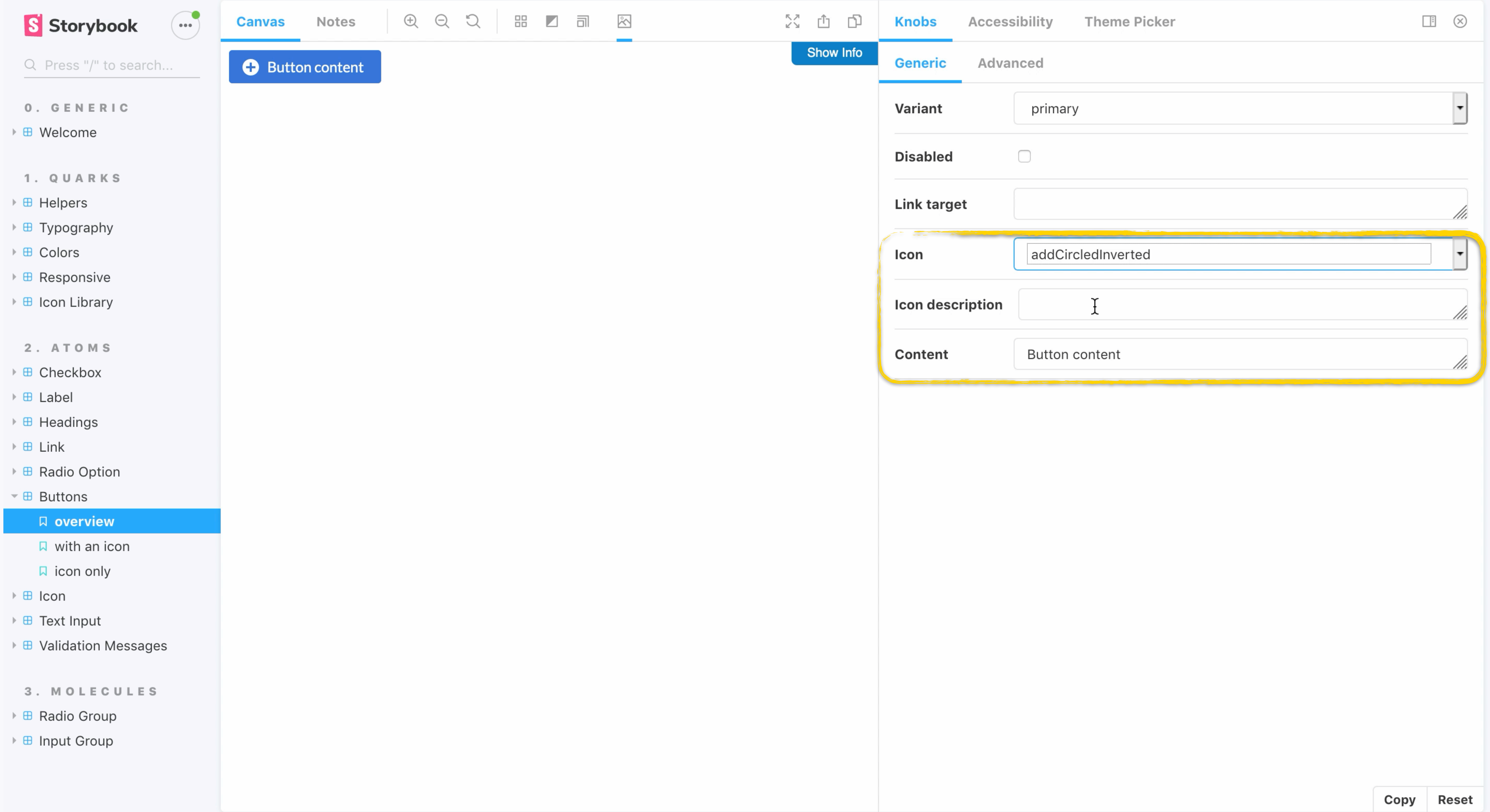
Variant	primary
Disabled	□
Link target	https://google.com/
Icon	
Icon description	
Content	Button content

```
<!DOCTYPE html>
<html> event
  <head>…</head>
  <body class="sb-show-main">
    <div class="sb-nopreview sb-wrapper">…</div>
    <div class="sb-errordisplay sb-wrapper">…</div>
    <div id="root">
      <div>
        <div style="position: relative; z-index: 0;">
          <a class="Button__StyledButton-sc-125wamt-0 bhkgGj"
          href="https://google.com/">…</a>
        </div>
        <button class="info__show-button" type="button"
        style="font-family: sans-serif; font-size: 12px;
        display: block; po…inter; top: 0px; right: 0px; border-
        radius: 0px 0px 0px 5px;">Show Info</button> event
        <div class="info__overlay" style="position: fixed;
        background: white none repeat scroll 0% 0%;…ng: 0px
        40px; overflow: auto; z-index: 99999; display: none;">
        …</div>
      </div>
    </div>
    <script
    src="runtime~main.f2ecddf4df82601b0fa8.bundle.js"></script
```

… › iframe#storybook-preview-iframe › html › body.sb-show-main › div#root › div › div

```
element {                                              inline
  position: relative;
  z-index: 0;
}
Inherited from body
body {                                               inline:5
  color: #1b2c4b;
  font-family: Lato,-apple-
   system,BlinkMacSystemFont,Roboto,Oxygen,Ubuntu,Cantarell,"Fira Sans","Droid
   Sans","Helvetica Neue",Arial,sans-serif;
  font-size: 1.4rem;
  font-weight: 400;
  line-height: 1.5;
}
Inherited from html
:root {                                              inline:5
  ~~font-size: 62.5%;~~
}
```

Build your components in a way that it avoids not accessible usages of it.



Variant: primary

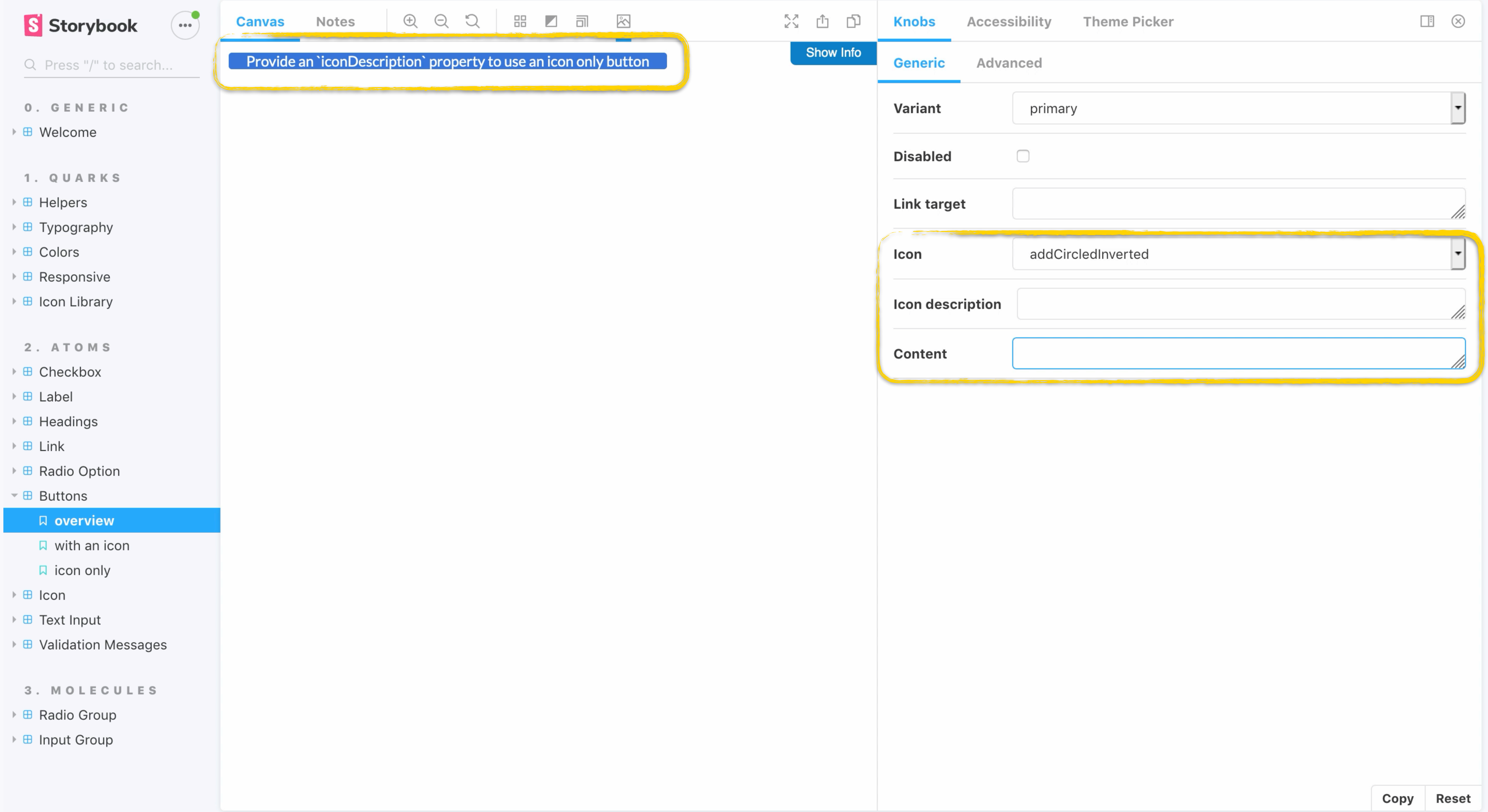
Disabled: □

Link target:

Icon: addCircledInverted

Icon description:

Content: Button content



Provide an `iconDescription` property to use an icon only button

Variant: primary

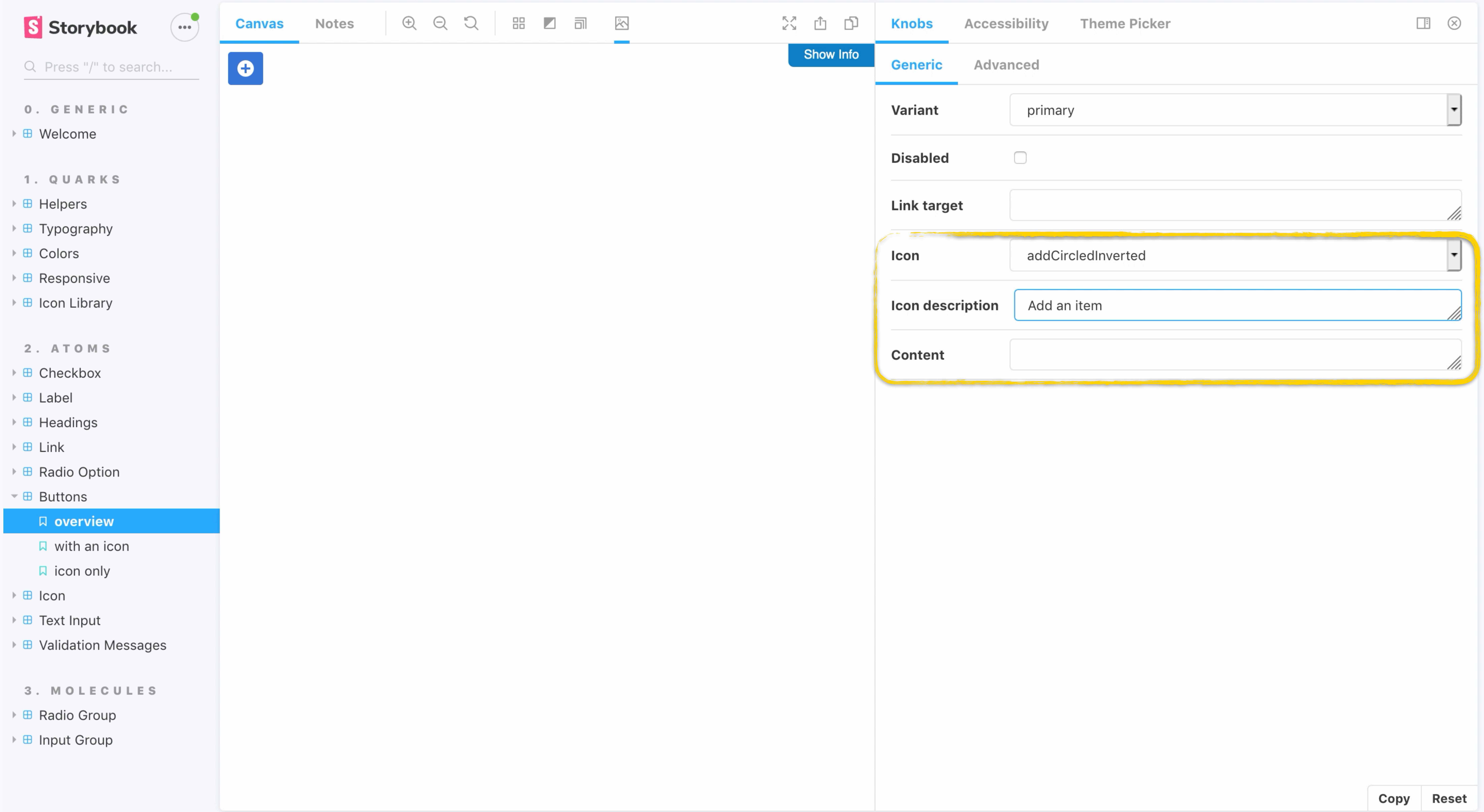
Disabled: □

Link target:

Icon: addCircledInverted

Icon description:

Content:



Variant	primary
Disabled	□
Link target	
Icon	addCircledInverted
Icon description	Add an item
Content	

Create an environment
where HTML & CSS are valued.

You are mainly delivering HTML and CSS to users. Please **care**.

Create opportunities to learn.

Develop a team of accessibility champions with members in every teams.

This team is here to help **finding solutions** or **mentor colleagues** around **accessibility**.

#shareTheLove

Develop and document a series of manual tests everybody can and should do on their work.

#contributing.md

⌨️ Did you test your work with keyboard navigation?

🕹️ Did you test it with an assistive technology?

🤖 Did you run Accessibility Insights for Web?

🗺️ Did you check your landmarks in the Web Rotor?

✅ Are all tests successful? Any limitation to mention?

🏃🏾‍♀️ Do you know where the device lab is?

3.

How can we document our systems for a11y?

WCAG is... not the most readable document.

Your documentation should give context-aware guidance on how to deliver an accessible product.

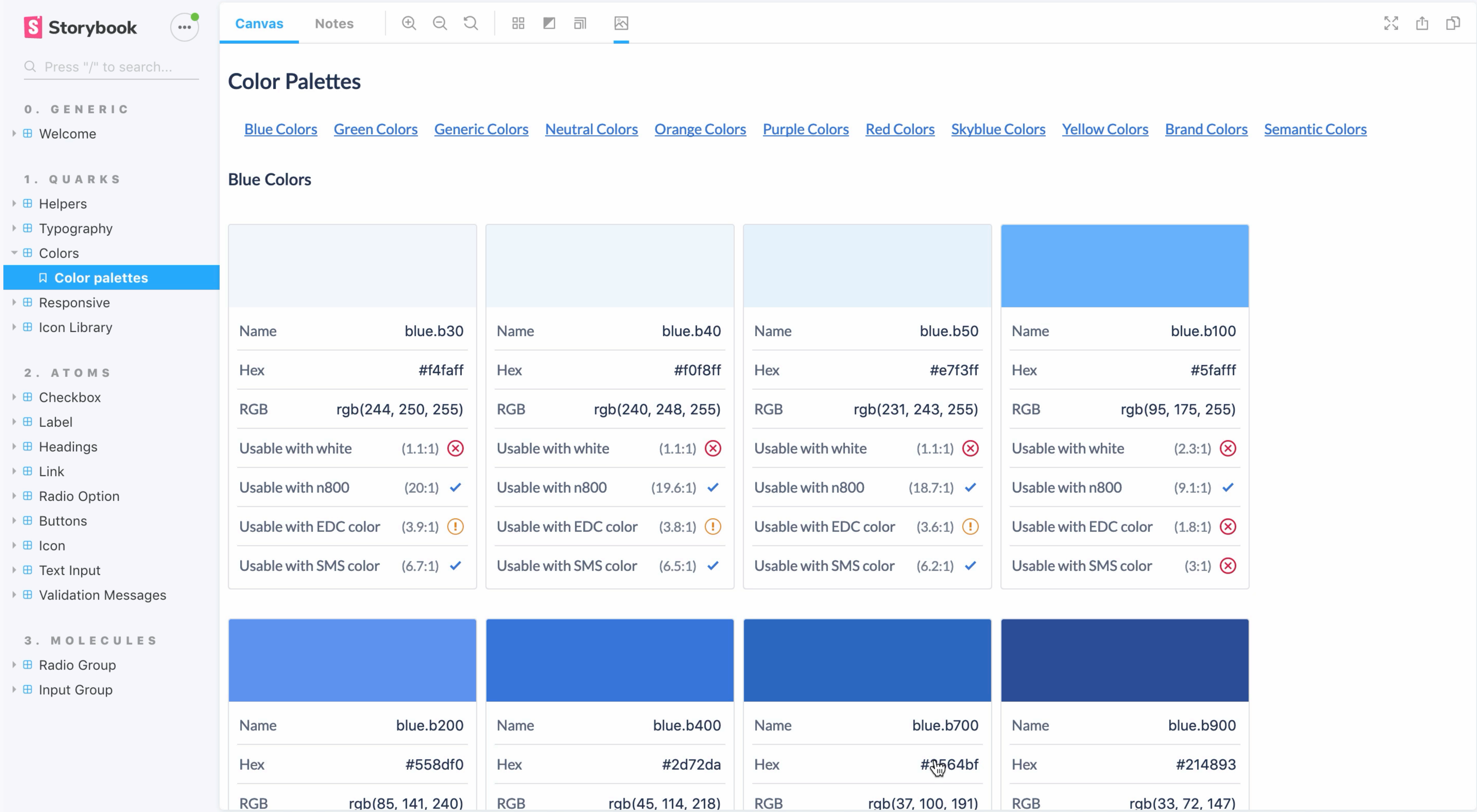
And you should start with an accessibility policy.

An accessibility policy is an important document about the goals, what's supported and what's not.

It will make clear what, when and how to test accessibility.

And it is a good starting point for the documentation.

Let’s talk about component documentation



Color Palettes

Blue Colors · Green Colors · Generic Colors · Neutral Colors · Orange Colors · Purple Colors · Red Colors · Skyblue Colors · Yellow Colors · Brand Colors · Semantic Colors

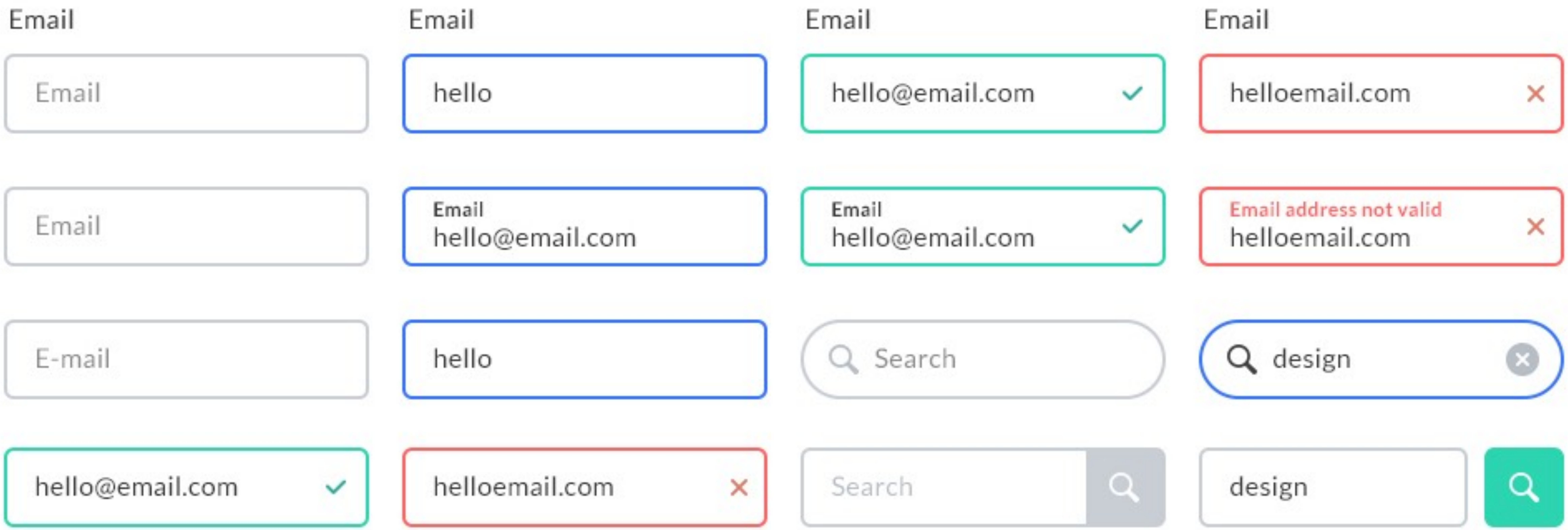
Blue Colors

Name	blue.b30	blue.b40	blue.b50	blue.b100
Hex	#f4faff	#f0f8ff	#e7f3ff	#5fafff
RGB	rgb(244, 250, 255)	rgb(240, 248, 255)	rgb(231, 243, 255)	rgb(95, 175, 255)
Usable with white	(1.1:1) ⊗	(1.1:1) ⊗	(1.1:1) ⊗	(2.3:1) ⊗
Usable with n800	(20:1) ✓	(19.6:1) ✓	(18.7:1) ✓	(9.1:1) ✓
Usable with EDC color	(3.9:1) ⚠	(3.8:1) ⚠	(3.6:1) ⚠	(1.8:1) ⊗
Usable with SMS color	(6.7:1) ✓	(6.5:1) ✓	(6.2:1) ✓	(3:1) ⊗

Name	blue.b200	blue.b400	blue.b700	blue.b900
Hex	#558df0	#2d72da	#2564bf	#214893
RGB	rgb(85, 141, 240)	rgb(45, 114, 218)	rgb(37, 100, 191)	rgb(33, 72, 147)

Each component should support and showcase all possible state.

Input elements



Email

Email

Email

Email

Email

hello

hello@email.com ✓

helloemail.com ✕

Email

Email
hello@email.com

Email
hello@email.com ✓

Email address not valid
helloemail.com ✕

E-mail

hello

Search

design

hello@email.com ✓

helloemail.com ✕

Search

design

A library of form input elements within the design system.

butterfly.com.au

You should provide product-specific guidelines.

P
Product
Overview
FOUNDATIONS
Accessibility
Color
Grid
Iconography
Typography
Writing style
COMPONENTS
Avatars
Badges
Banners
Breadcrumbs
Buttons
Checkboxes
Code
Date/time picker
Dropdown menu
Flag messages
Inline dialog
1

Accessibility

Products and web properties should be accessible to everyone, including those with vision, hearing, cognitive, or motor impairments. Accessible design lets people of all abilities interact with, understand, and navigate our products.

Keep in mind that:

- Accessibility is about having a good understanding of our user's journeys and proactively anticipating their needs.
- Accessible design is everyone's responsibility, from information and user experience design, through development, and on into help and support.

Structure & hierarchy

Consistent, clear hierarchy helps users who navigate the page using links or headers. Use headings and titles to outline the page so that users can see the structure and how the sections relate. Give users feedback so they know where they are in the application.

List by level of importance

Place items on the page in order of their level of importance so that users don't have to search for them.

This Atlassian page on Accessibility is interesting: a lot of information about all accessibility good practices. ALL of them. Honestly, would you often re-read it?

P
Product
Overview
FOUNDATIONS
Accessibility
Color
Grid
Iconography
Typography
Writing style
COMPONENTS
Avatars
Badges
Banners
Breadcrumbs
Buttons
Checkboxes
Code
Date/time picker
Dropdown menu
Flag messages
Inline dialog

Validate forms in-line

Validate forms in-line so keyboard users don't have to navigate far to get feedback.

Form

Text field

@#$%^&

You must enter a valid text

Selects

You must select something

Text field

@#$%^&

This will not be shared with others

Sometimes these help texts can be super long and verbose. The general rule of thumb is to keep these as short as possible while also being explicit. Otherwise people will end up with errors that jump around and are impossible to rea...

Meaningful text

Consistent and helpful text makes the user interface accessible to users who use a screen reader. Screen readers help users with visual impairments by reading both visible and non-visible alternative text aloud.

All text should support accessibility, whether it's visible (UI labels, headings, buttons, forms

Having a page with all information can quickly be over-whelming and difficult to maintain.

Prefer accessibility requirements per components: it is context-aware and actionable.

salesforce
Lightning Design System
Search
What's New
Getting Started
Platforms
Design Guidelines
Accessibility
Component Blueprints
Overview
Accordion
Activity Timeline
Alert
App Launcher
Avatar
Avatar Group
Badges
Brand Band
Breadcrumbs
Sections
About Form Elements
Structure
Base
Form Label
Form Control
States
View Mode/Static
Inline Edit
Help Text Icon
Showing tooltip
Feedback
Required
Error
Layout
Stacked
Horizontal
Single Column Support
Standalone
With 50/50 split
Compound
Rows
Fields
Address
Usage Examples
Record Form
View Mode
Stacked Alignment
Horizontal Alignment
Edit Mode
Stacked Alignment
Show Code

```
<div class="slds-form-element">
  <label class="slds-form-element__label" for="form-element-03">
    <abbr class="slds-required" title="required">* </abbr>Form Label</label>
```

Error

If an error has occurred while submitting a form, the form element with an error should provide feedback. The `slds-has-error` class is placed on the `<div class="slds-form-element">` element. Then, the error message for the user is placed in a `<div>` with the `slds-form-element__help` class.

Accessibility requirement

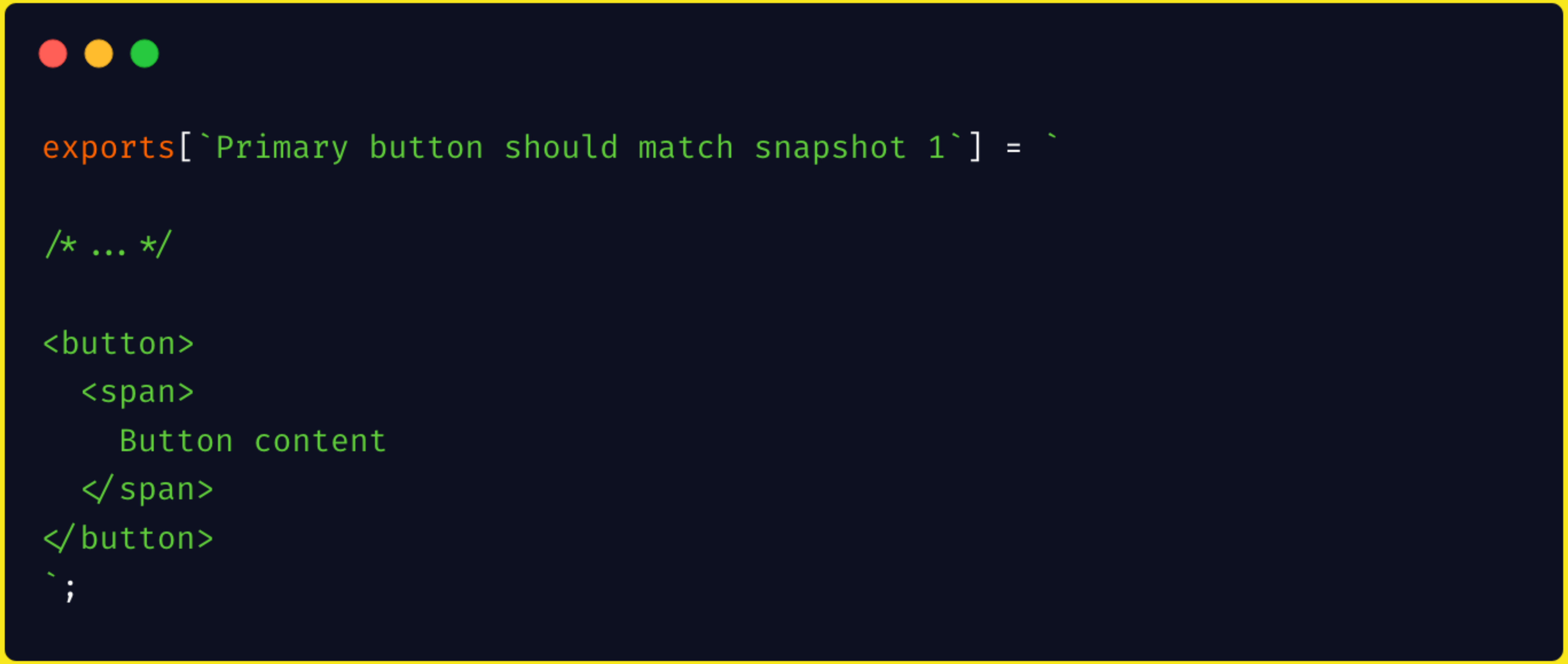
When a form element displays feedback notifying the user of an error, the error string should be linked to the element by adding the **`aria-describedby`** attribute to the **`<input>`**. The **`aria-describedby`** attribute must reference the id of the error message. This configuration allows screen readers to read the associated error message when the invalid field is focused.

```
<div class="slds-form-element slds-has-error">
  <label class="slds-form-element__label" for="form-element-05">
    <abbr class="slds-required" title="required">* </abbr>Form Label</label>
```

4.

How can we test our systems?

1️⃣ Snapshot tests are a must have

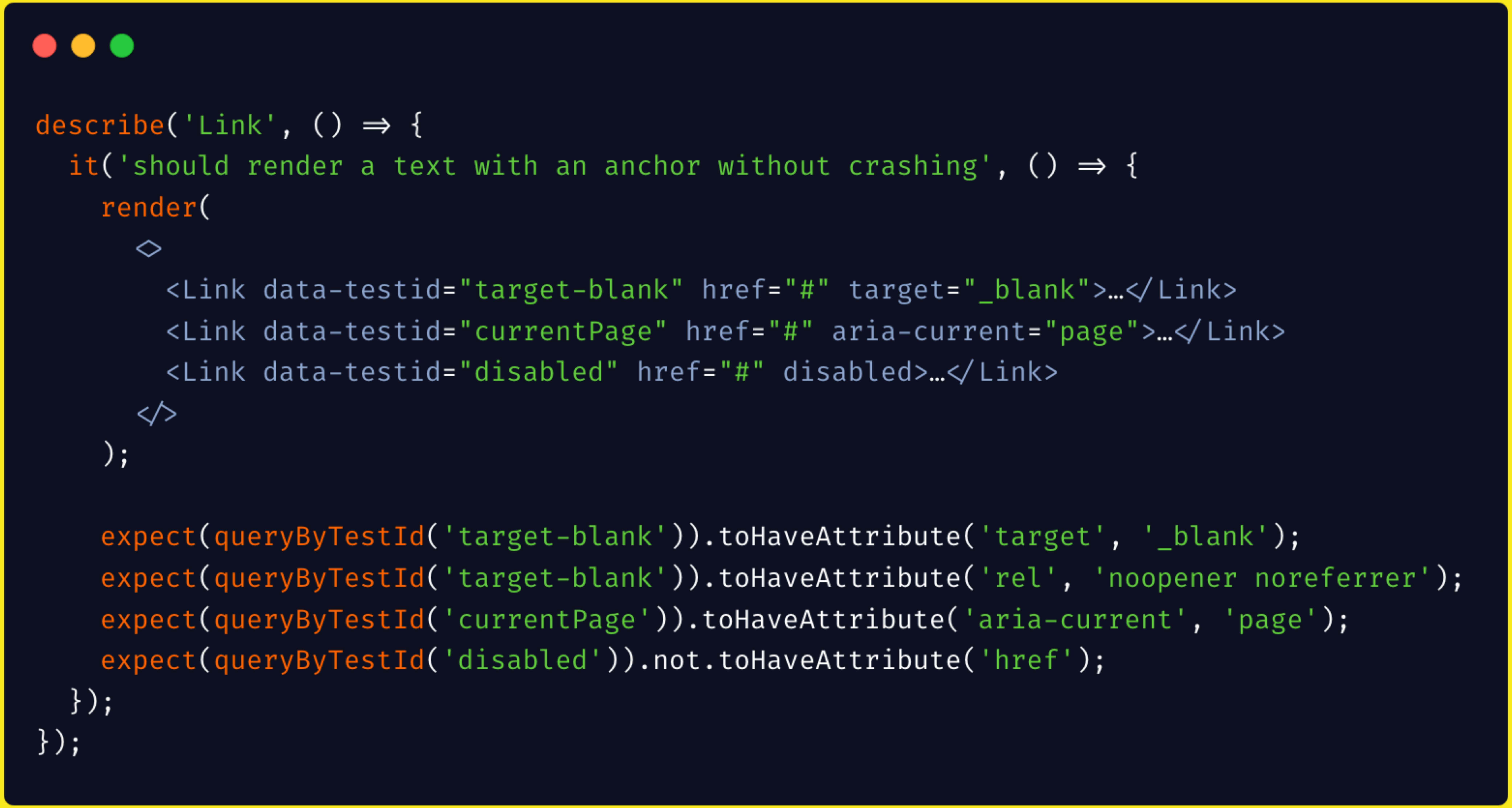


```
exports[`Primary button should match snapshot 1`] = `

/* ... */

<button>
  <span>
    Button content
  </span>
</button>
`;
```

1️⃣ Snapshot tests are a must have

2️⃣ Each default properties should be tested

3️⃣ Each custom properties/states should be tested



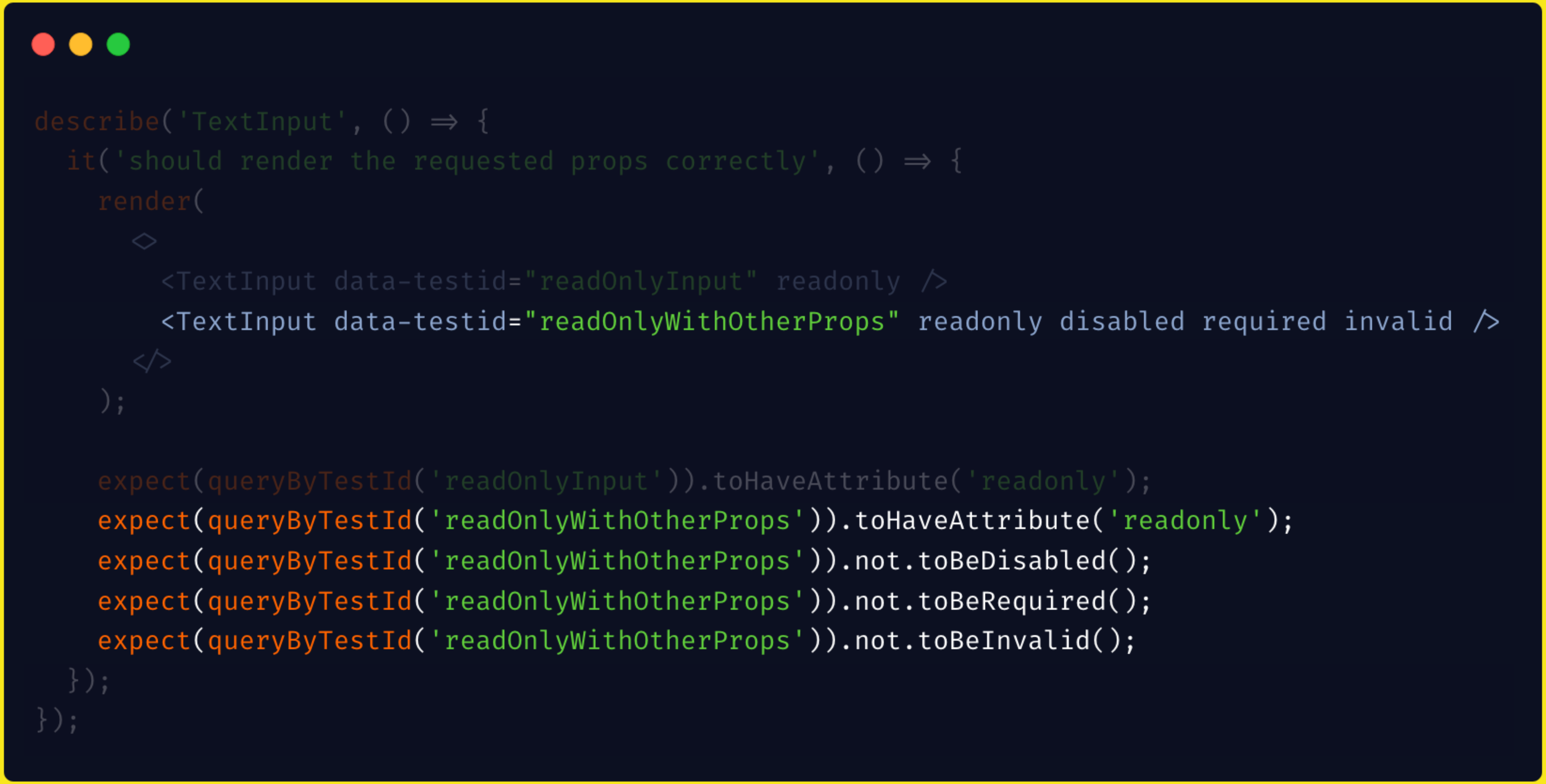
```
describe('Link', () => {
  it('should render a text with an anchor without crashing', () => {
    render(
      <>
        <Link data-testid="target-blank" href="#" target="_blank">...</Link>
        <Link data-testid="currentPage" href="#" aria-current="page">...</Link>
        <Link data-testid="disabled" href="#" disabled>...</Link>
      </>
    );

    expect(queryByTestId('target-blank')).toHaveAttribute('target', '_blank');
    expect(queryByTestId('target-blank')).toHaveAttribute('rel', 'noopener noreferrer');
    expect(queryByTestId('currentPage')).toHaveAttribute('aria-current', 'page');
    expect(queryByTestId('disabled')).not.toHaveAttribute('href');
  });
});
```

1️⃣ Snapshot tests are a must have

2️⃣ Each default properties should be tested

3️⃣ Each custom properties/states should be tested

4️⃣ Magically resolve conflicting properties



```
describe('TextInput', () => {
  it('should render the requested props correctly', () => {
    render(
      <>
        <TextInput data-testid="readOnlyInput" readonly />
        <TextInput data-testid="readOnlyWithOtherProps" readonly disabled required invalid />
      </>
    );

    expect(queryByTestId('readOnlyInput')).toHaveAttribute('readonly');
    expect(queryByTestId('readOnlyWithOtherProps')).toHaveAttribute('readonly');
    expect(queryByTestId('readOnlyWithOtherProps')).not.toBeDisabled();
    expect(queryByTestId('readOnlyWithOtherProps')).not.toBeRequired();
    expect(queryByTestId('readOnlyWithOtherProps')).not.toBeInvalid();
  });
});
```

1️⃣ Snapshot tests are a must have

2️⃣ Each default properties should be tested

3️⃣ Each custom properties/states should be tested

4️⃣ Magically resolve conflicting properties

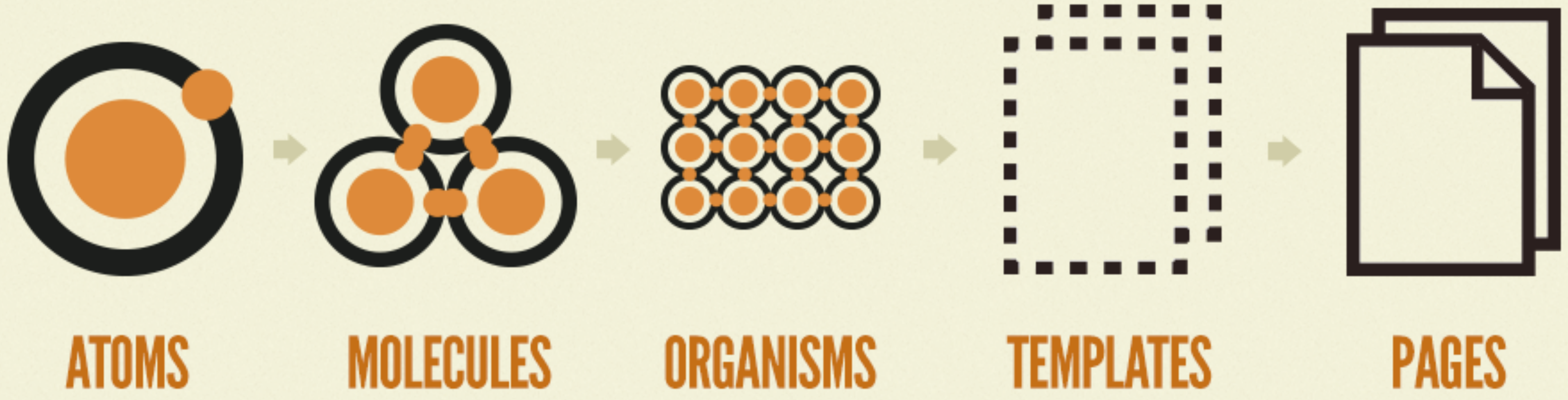
5️⃣ Check that your component handle events correctly

6️⃣ Run your component through tools like aXe

Automatic testing
catch only 15–20%
of accessibility issues.

It is important to regularly run accessibility auditing tools like Accessibility Insights for Web

Working with the atomic design principles allows you to split tests to be more readable



Atomic design by Brad Frost

Atoms are perfect for DOM-related tests

Molecules are working nicely with accessibility tests like aXe

Organisms are the place to be for focus and event handling tests.

Templates can be the higher-level of your tests with a focus on DOM order & sections' interactions.

Ensuring accessibility within a design system is pushing you to create an extensive test culture.

(a test culture a bit different than the usual React one)

🎉 In conclusion...

1.

Accessibility is as fun as frustrating.

2.

Setup an accessibility policy.

3.

Offer ways to learn more about a11y.

4.

Build a team of evangelists.

5.

Propose a documentation adapted to the product

6.

Develop a series of manual and automated tests.

As a last though:
more I work as an accessibility specialist,
more I think our job is not about the code,
it's about making accessibility accessible.

Merci beaucoup ! 🇫🇷

Thank you! 🇬🇧

Tack! 🇸🇪

Bedankt! 🇳🇱

@iamhiwelo

Damien Senger

Digital designer, specialised in accessibility.

raccoon.studio · noti.st/hiwelo

@iamhiwelo