



Welcome to

National Design System of Saudi Arabia

Empowering government digital experiences with a unified design language.

[Guidelines](#) [Documentation](#)

Designers

Embarking on your design journey is easy and exciting with the right tools at hand. Below you'll find a curated checklist to swiftly set your creative sails.

Discover More

Design Language

Explore the Aesthetic Foundation



Design Kits

Your Essential Design Toolbox



Guidelines

Blueprints for Best Practices



Developers

Dive into development with platforms code seamlessly. We've compiled a comprehensive checklist to streamline your setup process and kickstart your project.

Discover More

Documentation

Knowledge at Your Fingertips



Installation

Set Up for Success



Resources

Expand Your Toolkit



Features

Building a Unified, Accessible Future

1. Consistency and Unified Identity [Learn More →](#)

The Platforms Code aims to create a consistent digital government experience by standardizing the foundation styles and system components.

2. Accessibility Ease [Learn More →](#)

Designed for community individuals, it ensures a flexible user experience that adheres to accessibility standards.

3. Adherence to Local and International Standards [Learn More →](#)

Committed to developing high-quality experiences and products, we ensure that the Platforms Code meets both local and international standards.

4. Atomic Design

[Learn More →](#)

We adopt an atomic design methodology to ensure organization and sustainability in the development of user interfaces.

5. Contribution and Continuous Development

[Learn More →](#)

The goal is continuous expansion and evolution to keep the Platforms Code comprehensive and adaptable to new challenges, encouraging designers, developers, and stakeholders to contribute.

6. Responsive Design

[Learn More →](#)

Tailored to respond to the varied needs of government services and products.

About Platforms Code

Crafting Excellence in Design

Platforms Code is a specialized open-source design system tailored for government entities, with a focus on accessibility and consistency, it simplifies the creation of digital experiences, empowering public sector teams to deliver services more efficiently.

The system's name reflects its foundational flexibility, allowing for intricate solutions that cater to the unique needs of government operations. Platforms Code is not just a toolkit but a community endeavor, shaped by the collaborative efforts of those dedicated to enhancing civic engagement.

Learn More →



Connect, Collaborate, and Create

Share ideas, get feedback, and stay inspired as you contribute to a dynamic ecosystem of creative collaboration.

Your Feedback Matters

Help us improve

[Take Survey](#)

Discuss, discover, and get support

Find Us On Beem

[Take Survey](#)

Your story can inspire others

Your Adoption

[Take Survey](#)

Insights and Innovations

Dive into a curated collection of articles that provide deep dives into design trends, development techniques, and best practices.

[All Articles](#)

Consistency and Unified Identity

[Design](#)

[Design System](#)

The Platforms Code Design System aims to create a consistent digital government experience by unifying the core styles of the system's components.

8 Jul 2024

[Full Details](#)

Atomic Design

[Design](#)[User Interface](#)

We adopt an atomic design methodology to ensure organization and sustainability in the development of user interfaces.

4 Jun 2024

[Full Details](#)[→ Load More](#)[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)



Home > Start Designing

Start Designing

Starting a project or refining skills? Below is a checklist to set up and proceed confidently.

About Platforms Code

Building Tomorrow's Solutions, Today.

Platforms Code, an open-source design system funded by DGA, streamlines digital projects by combining code, design tools, and guidelines in a community-driven approach. It allows for quick development of consistent experiences, letting designers and developers focus on customizing for specific needs using pre-built assets.

Platforms Code, an open-source initiative, meets government needs and encourages community contributions, evolving to address diverse requirements and expanding its utility. Learn about its development and impact by watching our video on Platforms Code's history.



33+

Component

for Streamlined Design Solutions



4,650

Variants

for Tailored User Experiences



100

Styles

Aligned with Official Guidelines



Platforms Code UI Kit



Family Font

Design Kits

Design. Simplify. Excel.

Start with our comprehensive Figma core design kits. These kits enhance consistency and accelerate the design process, allowing teams to focus on specific user challenges with standardized resources.

Design Kits

Mastering the Language of Innovation

Our design principles guide decision-making and shape our approach to challenges, rooted in understanding user interactions and empowering users, essential for crafting innovative product experiences.



Intuitive Across Devices

Experience consistency and intuitive interaction, no matter the device.



Streamlined for Engagement

Designed to keep you in the flow, our products move you forward effortlessly.



Inclusivity by Design

Our products are crafted for everyone, embracing a wide range of perspectives and abilities.

Distinctly Platforms Code



Recognize and connect with our signature style—one platform, one seamless experience.

Foundations

Core design principles for consistent user experiences



Accessibility

Standards and practices to ensure inclusivity and usability for all users

Components

Modular UI elements for rapid interface building



Design Tokens

Variables for uniformity in visual design



Resources

Resources to Empower Your Creativity

Resources



Get started quickly with our complete suite of tools and guides, designed to create stunning, accessible user experiences and efficiently bring your ideas to life.

Fonts

Curated fonts for perfect style



Test Your Design

Validate design with essential tools

soon

Install the Design system libraries.

Smooth setup and seamless incorporation



Migration Guide

Streamline upgrades with clear steps



Frequently Asked Questions



Check our FAQs for fast answers to common questions, designed to help you efficiently resolve issues and improve your understanding of our platform.

[About Platforms Code](#)[General](#)[Components](#)[Guidelines](#)[Contributing](#)[Telemetry](#)[Assessment Criteria](#)

Community Discussions



Join our community to connect with peers, share insights, and seek advice. This collaborative platform lets members exchange ideas, address challenges, and stay updated on the latest trends and best practices.

Join Us

Assessment Criteria



Define the standards and metrics for evaluating your project quality and effectiveness. Our assessment criteria ensure your work aligns with our goals and meets high expectations for performance and user satisfaction.

Download Checklist

Essential download verification steps



Page Speed Index

Measure site loading performance

soon

Compliance Score

Evaluate regulatory adherence level



Contact Us

Connect With Us

Whether you have questions, need support, or want to share your feedback, we're here to listen and help.

DS-DGA@dga.gov.sa

[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)

[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)



Home > Designing for mobile

Designing for Mobile

Starting a project or refining skills? Below is a checklist to set up and proceed confidently.

Mobile Design Kits

Efficiency. Consistency. Innovation.

Begin your journey with our extensive Mobile UI Kit in Platforms Code, meticulously crafted for mobile interfaces. Similar to its web counterpart, this kit comes packed with additional components optimized for mobile views. It equips designers with essential elements to craft cohesive and user-friendly mobile experiences, ensuring consistency across government platforms while meeting the unique requirements of mobile devices.



Platforms Code Mobile UI Kit
Design assets

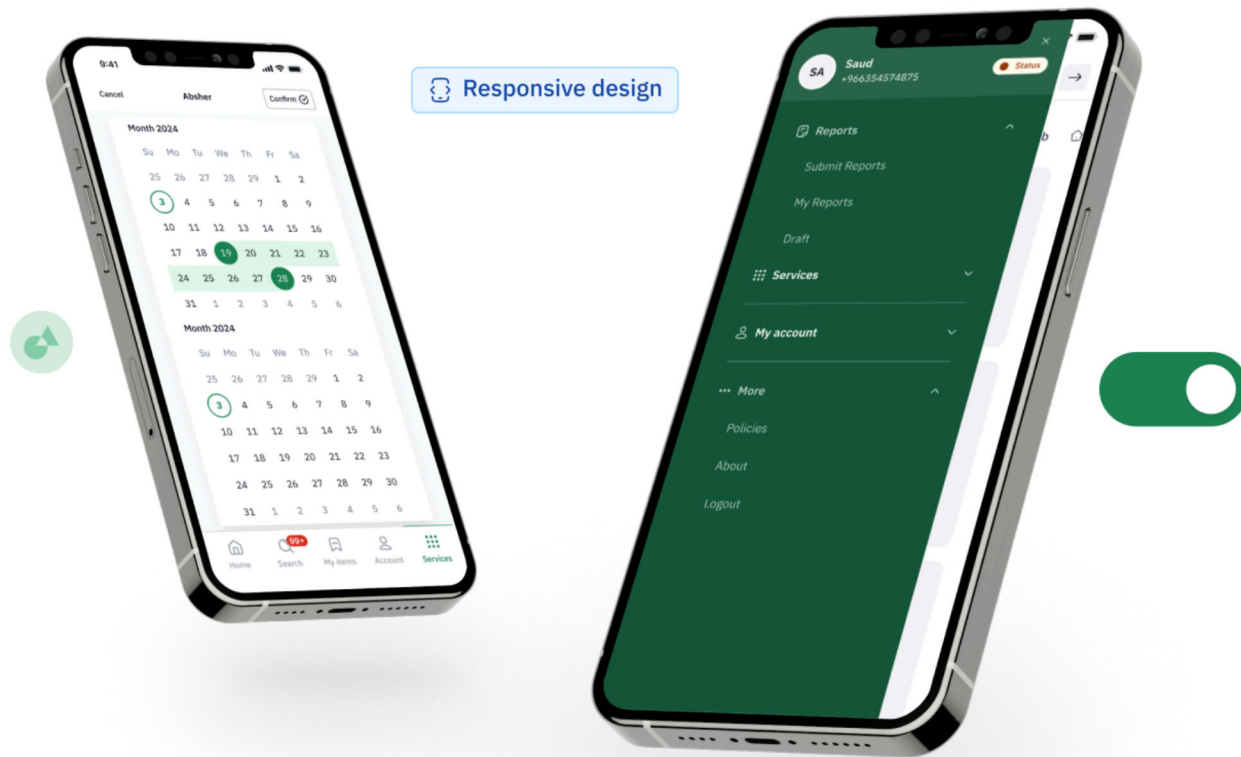
Design Language

Why Platforms Code Mobile?

The Platforms Code mobile UI kit offers a host of additional components, designed to elevate the mobile user experience. From intuitive navigation bars to dynamic mobile-specific icons, these elements empower designers to create captivating mobile applications.

Tailored Experience

- Mobile Navigation Bar
- Tap Bar
- Top Bar
- Splash Screen
- Mobile Modal
- Date Picker



Contact Us

Connect With Us

Whether you have questions, need support, or want to share your feedback, we're here to listen and help.

DS-DGA@dga.gov.sa

[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)

[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)



Home > Install the Design system libraries

Install the Design system libraries.

This guide offers concise instructions for designers to install and integrate the national design system library into their projects, ensuring a smooth setup and seamless incorporation of components and styles.

Comprehensive Design Kits

Efficiency. Consistency. Innovation.

Begin your journey with our extensive and comprehensive design library in Platforms Code, These Figma library files contains the components and design language (foundations) and the icons pack of Platforms code design system, making it easy to access ready-made components and consistent design guidelines for developing innovative and cohesive products.

**PC 1.0 Foundations**

Design assets

**PC 1.0 icon pack**

Design assets

**PC 1.0 Components - Desktop UI Kit**

Design assets

**PC 1.0 Components - Mobile UI Kit**

Design assets



Installing and publishing the foundations and icons library files to design files.

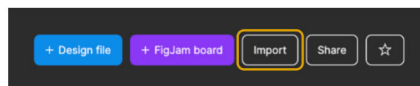
To use the UI kit library files correctly, you must start by installing the foundation library and the icon library files first. These libraries contain atomic-level elements that form the base for the UI component library.

1.Open Figma.

Open Figma and log in to your account.

2.Import the “PC 1.0 Foundations” and “PC 1.0 icons” library files on your project folder.

Import the “PC 1.0 Foundations” and “PC 1.0 Icons” library files into your project folder. Ensure that you have created both a team file and a project folder within the team file.

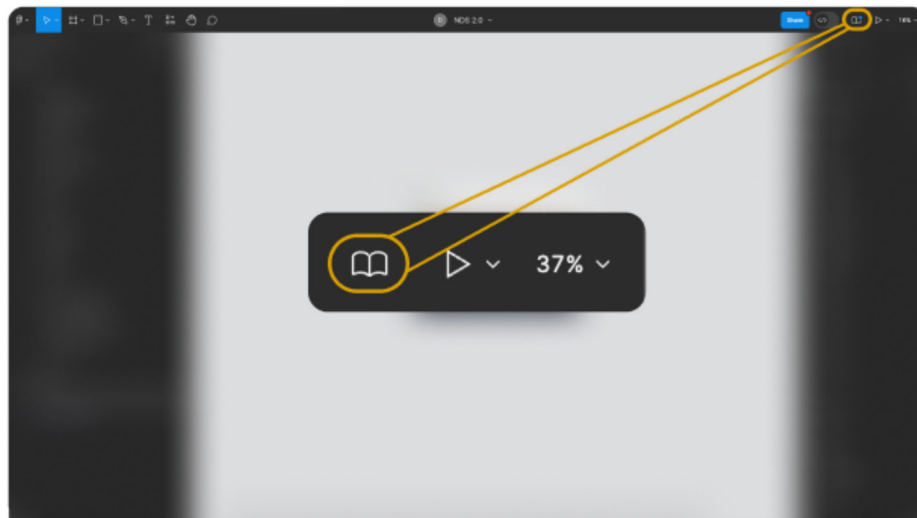


3.Open the “PC 1.0 Foundations” and “PC 1.0 icons” library files on Figma.

Open the “PC 1.0 Foundations” and “PC 1.0 icons” library files, which will be used in another file (your Figma design file).

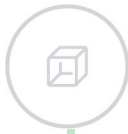
4.Publish the “PC 1.0 Foundations” and “PC 1.0 icons” library files on Figma.

Publish the “PC 1.0 Foundations” and “PC 1.0 icons” library files by clicking on the library icon at the top of the screen, then publish the library to the team that contains the design files.



5. Add the “PC 1.0 Foundations” and “PC 1.0 icons” library files to your design file.

To add the assets library files to the design file, use the library files icon at the top of the screen, then add the “PC 1.0 Components - Desktop” library by clicking on 'Add to file' button.

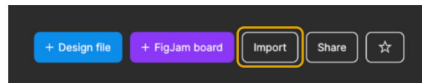


Installing and publishing the components library file to your design files.

Now, let's install our components library. It contains all the UI elements you need to create an exceptional digital experience for your products.

1.Import the “PC 1.0 Components - Desktop” or “PC 1.0 Components - Mobile” on your project folder.

Import the “PC 1.0 Components - Desktop” or “PC 1.0 Components - Mobile” library files into your project folder. Ensure that you have created both a team file and a project folder within the team file.

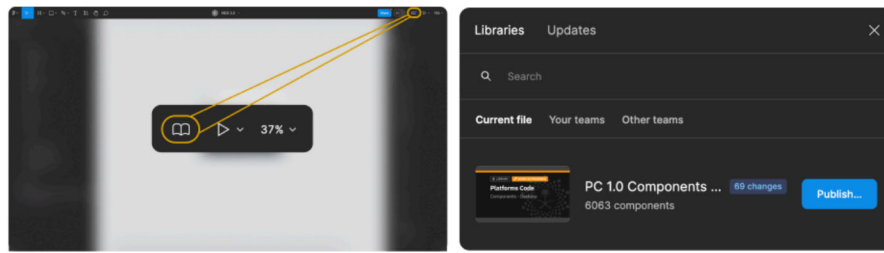


2.Open the “PC 1.0 Components - Desktop” or “PC 1.0 Components - Mobile” library file on Figma.

Open the “PC 1.0 Components - Desktop” or “PC 1.0 Components - Mobile” library file, which will be used in another file (your Figma design file).

3.Publish the “PC 1.0 Components - Desktop” or “PC 1.0 Components - Mobile” library file.

Publish the “PC 1.0 Components - Desktop” or “PC 1.0 Components - Mobile” library file by clicking on the Library icon at the top of the screen, then publish the library to the team that contains your design files.



4. Add the “PC 1.0 Components - Desktop” Library file to your design file.

To add the Component library file to the design file, click on the Library icon at the top of the screen, then add the “PC 1.0 Components - Desktop” library by clicking on “Add to file” button.



Start using design assets and create impactful solutions

Now, finally you can browse all the UI elements, foundation (Design language), starting from text styles to all tokens such as colors tokens and spacing tokens.

You can browse the components through the “Assets” tab on the left side of the screen.

In the design file, you can review and add components from the library to your design file through the "Assets" tab.

You can browse and try with the design language (Foundations) on the right side of the screen.

For example, when selecting any Text element, you can browse all text styles in the "PC 1.0 Components - Desktop" library file on the right side of the screen. Similarly, when selecting any Frame element, you can explore the colors variables to apply to the selected frame.

A dictionary for design decisions, All design tokens in one place

Discover all design tokens, from colors to spacing values for different device sizes, alongside radius and width options, conveniently consolidated in one place.



Linking variables to your components library files

The “PC 1.0 Components - Desktop”, “PC 1.0 Components - Mobile” and “PC 1.0 icon pack” variables collection must be attached to the same variables was defined and published in the “PC 1.0 Foundations “ library file in your project file.

To attach all of the “PC 1.0 Foundations” variables to your components file:

- 1. Make sure you have published the “PC 1.0 Foundations” and “PC 1.0 icons” library files to your team. and add them to the “PC 1.0 Components - Mobile” or “PC 1.0 Components - Mobile”.**
- 2. Select the component set “frame” you want to attach the variables to.**
- 3. Open the Swap Variables plugin to help you swap the local variables with the “PC 1.0 Foundations “ library file on your project file.**

4.Override text style and Swap the variables.



Remember

- If you are a contributor; Feel free to change any style you want! you can change the variables in the “PC 1.0 Foundations” variables collection.

Example: changing the primary color of button to Lavender 500 primary

- 1. Open the variables set in “PC 1.0 Foundations” and change the variable you want**
- 2. Publish the changes you have made in “PC 1.0 Foundations” library and update the “PC 1.0 Components - Mobile” or “PC 1.0 Components - Mobile” files.**
- 3. Select the component set “frame” you want to attach the variables to. in this example select the buttons**
- 4. Open the Swap Variables plugin to help you swap the local variables with the “PC 1.0 Foundations “ library file on your project file.**
- 5. Override text style and Swap the variables.**

After updating the library you will noticed the changed you have made are reflecting in the style sets in “PC 1.0 Components - Mobile” or “PC 1.0 Components - Mobile” files.

Contact Us

Connect With Us

Whether you have questions, need support, or want to share your feedback, we're here to listen and help.

DS-DGA@dga.gov.sa

[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)

[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)



Home > Migration Guide

Library Migration Guide

Clear instructions for designers to update their library with new versions, ensuring a smooth migration.

Crafting the Future of Design

Stay Current with the New Updates

As the national design system evolves, it is crucial for government platforms to stay updated with the latest changes. This allows them to fully utilize the design system's potential and maintain high standards of functionality and user experience. In this guide, you will find instructions for designers on how to update their libraries to new versions, ensuring a smooth migration.

Download updated libraries

**PC 1.0 Foundations**

Design assets

**PC 1.0 icon pack**

Design assets

**PC 1.0 Components - Desktop UI Kit**

Design assets

**PC 1.0 Components - Mobile UI Kit**

Design assets

What's new?

General Enhancement

May 2024

- **Labels:** Enhancements for labels involve improving their clarity, consistency, and accessibility across the user interface. This may include better alignment, color contrast, and font size adjustments to ensure labels are easily readable and effectively guide users

- **Textarea:** General enhancements for textareas focus on optimizing their usability and appearance. This could involve refining the resizing behavior, adjusting padding and spacing for better readability, and ensuring compatibility with various screen sizes and devices.
- **Tabs:** Enhancements for tabs aim to improve their visual presentation, interaction design, and overall user experience. This may include refining the tab layout, adding animations for smoother transitions, and ensuring proper keyboard navigation for accessibility compliance.

Label

Textarea

Tabs



Import and publish the new library file to your design files.

To update the library correctly, you must start by installing and publish the new library, and then replace the old one with new one.

1. Import the new library file on your project folder.

Import the new library file into your project folder. Ensure that you have created both a team file and a project folder within the team file.

+ Design file

+ FigJam board

Import

Share



2. Open the new library file on Figma.

Open the new library file, which will be used in another file (your Figma design file).

3. Publish the the new library file to your team

Publish the new library file by clicking on the library icon at the top of the screen, then publish the library to the team that contains the design files.

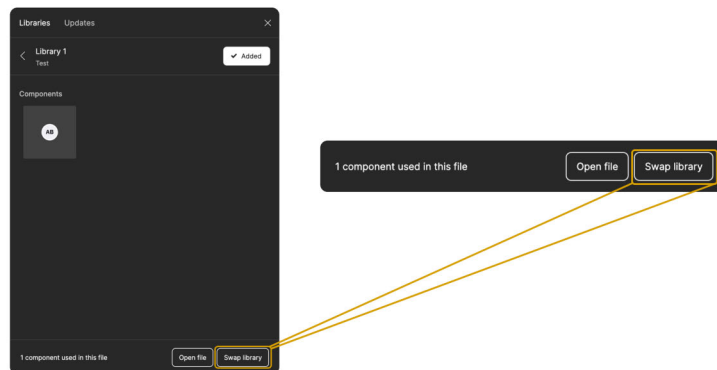
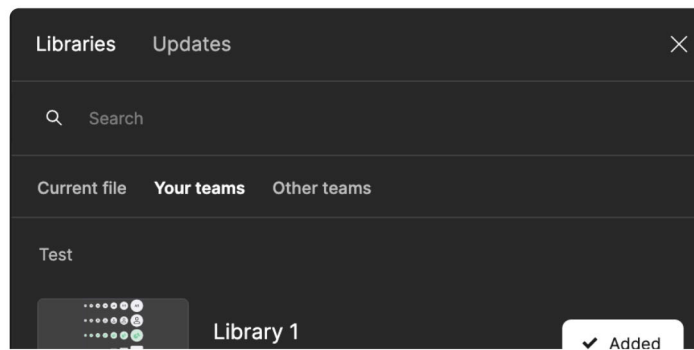


Swapping the old library with the new library

Now, let's swap the old library with the new library to seamlessly update your design assets.

1. Browse the added library

Click on the library icon at the top of the screen to view all the added libraries. click on the library you want to replace.



1. Choose the new library to swap with the old one.

Select the new library to replace the current one for a seamless update. and then the used components in the file will be replaced with the updated one.



Remember

When swapping, Figma updates styles and components with matching names, including sets and variants. Ensure component names haven't changed in the previous library.

Contact Us

Connect With Us

Whether you have questions, need support, or want to share your feedback, we're here to listen and help.

DS-DGA@dga.gov.sa

[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)

[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)



Home > Developing

Start Developing

Ready to start developing on Platforms Code? This page covers essentials from setting up your environment to accessing our libraries, providing all you need to create powerful applications.

Installing

Installing Today's Future Solutions

To integrate Platforms Code UI React into your project, you can use your preferred package manager. Follow the instructions below to add it as a dependency:

Using npm:

```
npm install nds-design-system@^0.1.0
```

Using yarn:

```
yarn install nds-design-system@^0.1.0
```

This will install the latest version of Platforms Code UI React, allowing you to utilize our robust library of React components in your applications.

Troubleshooting

If you run into any issues during the setup of Platforms Code Components React, please visit our [GitHub](#) repository for further guidelines and support. Should you not find an existing issue that addresses your concerns, feel free to [create a new](#) one so we can assist you effectively.



soon

Platforms Code Storybook
Documentation

Documentations

Knowledge at Hand

Access our detailed guides and documentation to fully leverage Platforms Code. From setup to advanced features, get all the essential information to effectively use our tools in your projects.

Design Language

Mastering the Art of Code

Our development principles drive decision-making and problem-solving, rooted in efficient coding practices and empowering developers, essential for crafting high-quality product experiences.

Components

Modular UI elements for rapid interface building



Design Tokens

Variables for uniformity in visual design

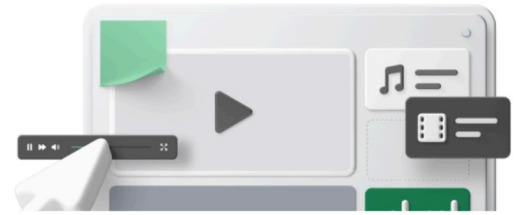
Foundations

Core design principles for consistent user experiences



Accessibility

Standards and practices to ensure inclusivity and usability for all users



Consistent Across Platforms

Experience uniformity and reliable performance, no matter the platform.



Optimized for Efficiency

Designed to keep you productive, our codebase ensures smooth and swift operations.



Inclusivity by Implementation

Our code is developed for everyone, embracing diverse needs and capabilities.



Distinctly Platforms Code

Recognize and engage with our unique coding style, one platform, one coherent experience.

Resources

Resources to Empower Your Code

Resources



Get started quickly with our complete suite of tools and guides, designed to create stunning, accessible user experiences and efficiently bring your ideas to life.

Installation Guide

Step-by-step setup instructions



Usage Guidance

Optimize with essential tips



Migration Guide

Streamline upgrades with clear steps

soon

Frequently Asked Questions



Check our FAQs for fast answers to common questions, designed to help you efficiently resolve issues and improve your understanding of our platform.

[About Platforms Code](#)[General](#)[Components](#)[Guidelines](#)[Contributing](#)[Telemetry](#)[Assessment Criteria](#)

Community Discussions



Join our community to connect with peers, share insights, and seek advice. This collaborative platform lets members exchange ideas, address challenges, and stay updated on the latest trends and best practices.

[Join Us](#)

Assessment Criteria



Define the standards and metrics for evaluating your project quality and effectiveness. Our assessment criteria ensure your work aligns with our goals and meets high expectations for performance and user satisfaction.

Download Checklist

Essential download verification steps



Page Speed Index

Measure site loading performance

soon

Compliance Score

Evaluate regulatory adherence level



Contact Us

Connect With Us

Whether you have questions, need support, or want to share your feedback, we're here to listen and help.

DS-DGA@dga.gov.sa

[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)

[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024



Home > Contributing

Start Contributing

Platforms Code flourishes with contributions from designers and developers like you. Get involved through coding, design, or documentation—here's how to help.

[Contribute Now](#)

Should I contribute?

Contributing to Platforms Code is a rewarding way to give back to the community and improve the system for everyone. Before diving in, it's important to ensure your contribution is aligned with the needs of the platform.

Relevance

Ensure your contribution fixes an issue or enhances the platform meaningfully.

Broad Impact

Aim for contributions that benefit the majority, rather than niche cases.

Minor Enhancements

Small fixes and additions like bug resolutions or new icons are always valuable.

Major Additions

For significant contributions like new components, thorough evaluation is necessary.

Still Unsure?

If you're uncertain about the suitability of your contribution, don't hesitate to reach out to the Platforms Code team. We're here to assist and guide you through the process. Your initiative and contributions are what help Platforms Code grow and evolve.

Contribution

How to Contribute

Interested in contributing to Platforms Code? Here's a simple guide to get started. Follow these steps to ensure your contributions align with our standards and help us grow.

1 Familiarize Yourself

Learn about Platforms Code's core principles and components to ensure your contributions align with our goals.

1 Join the Community

Connect with other contributors through forums and community meetings to collaborate and receive support.

soon

1 Follow Contribution Guidelines

Review the guidelines on our GitHub page for coding standards, design principles, and documentation styles.

soon

1 Submit Your Contributions

Once ready, submit your contributions via GitHub, providing a clear description of your changes and their benefits.

[Submit](#)

Type of Contribute



Design Contributions

Improve or create designs to enhance user interfaces and experiences



Code Contributions

Contribute by fixing bugs, adding features, or enhancing performance

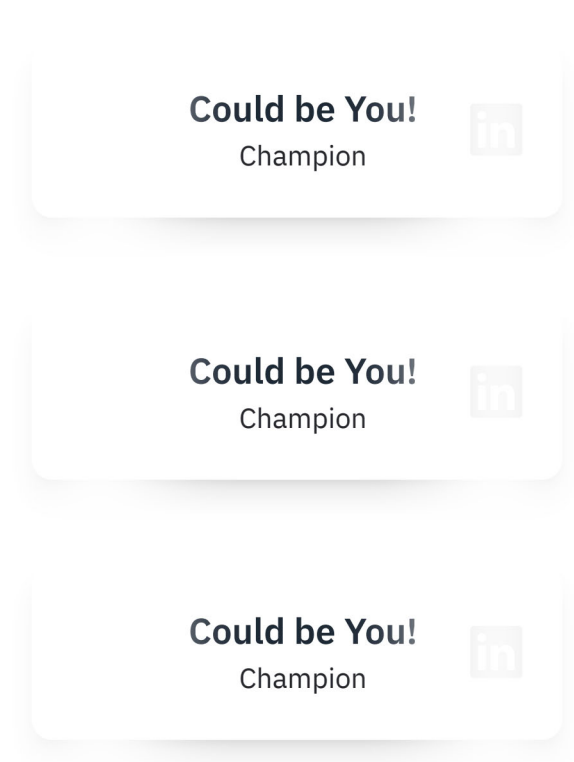


Documentation

Enhance platform accessibility by creating tutorials, guides, or refining documentation

Champions

Build Greatness Together



We invite you to join our efforts and contribute, whether through coding, design, or documentation. Your name and contributions could be highlighted in this very spot. Let's build something great together!

Contribute Now

Contact Us

Connect With Us

Whether you have questions, need support, or want to share your feedback, we're here to listen and help.

DS-DGA@dga.gov.sa

[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)

[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)



Home > Guidelines

Guidelines

Dive into the heart of our design system with these concise standards that guarantee consistency and uphold our user experience quality across all contributions.

Foundations



Core design principles for consistent user experiences

Accessibility



Standards and practices to ensure inclusivity and usability for all users

Components



Modular UI elements for rapid interface building

Design Tokens



Variables for uniformity in visual design

Contact Us

Connect With Us

Whether you have questions, need support, or want to share your feedback, we're here to listen and help.

DS-DGA@dga.gov.sa

[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)

[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024

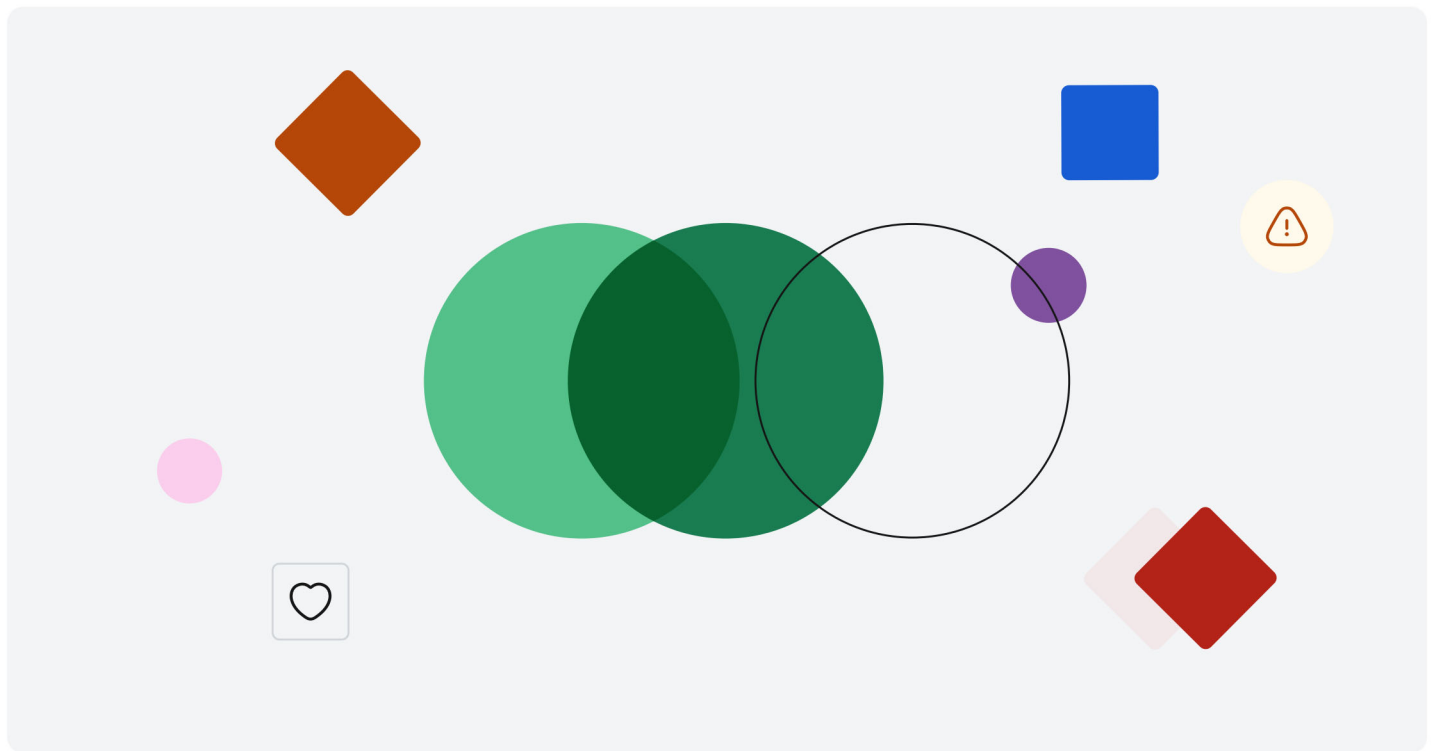
[Terms and Conditions](#) [Privacy Policy](#)



[Home](#) > [Guidelines](#) > [Foundations](#) > [Color System](#)

Colors

Color serves as a medium for expressing style, evoking emotions, and conveying significance. Using a standardized range of colors intentionally ensures a recognizable, pleasant, and uniform experience.

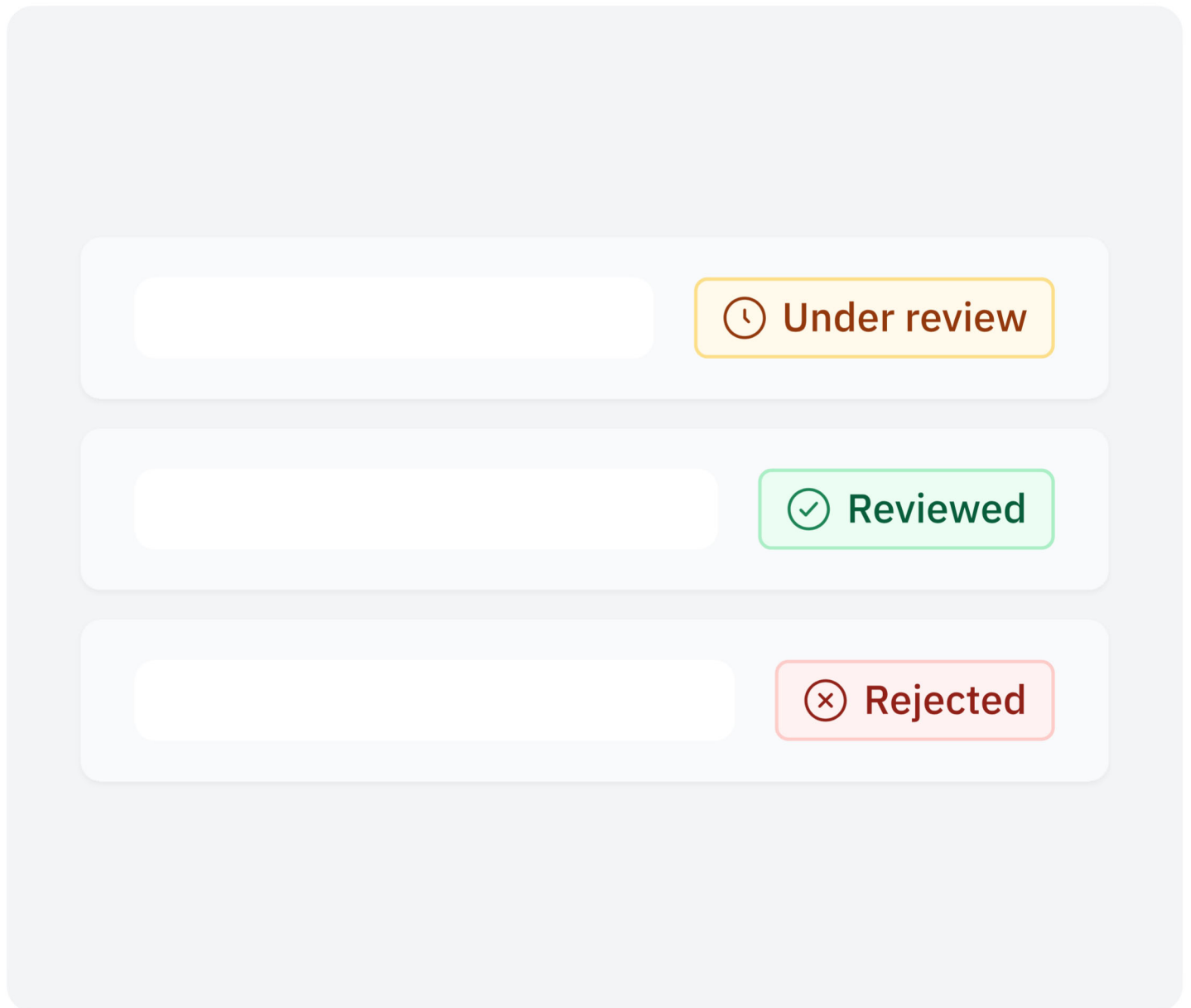


Overview

Consistency and uniformity are the fundamental essence around which the Platforms code design system revolves. Employing colors optimally, as recommended by Platforms code, is considered one of the crucial pillars that ensure the creation of a enjoyable and reliable unified experience.

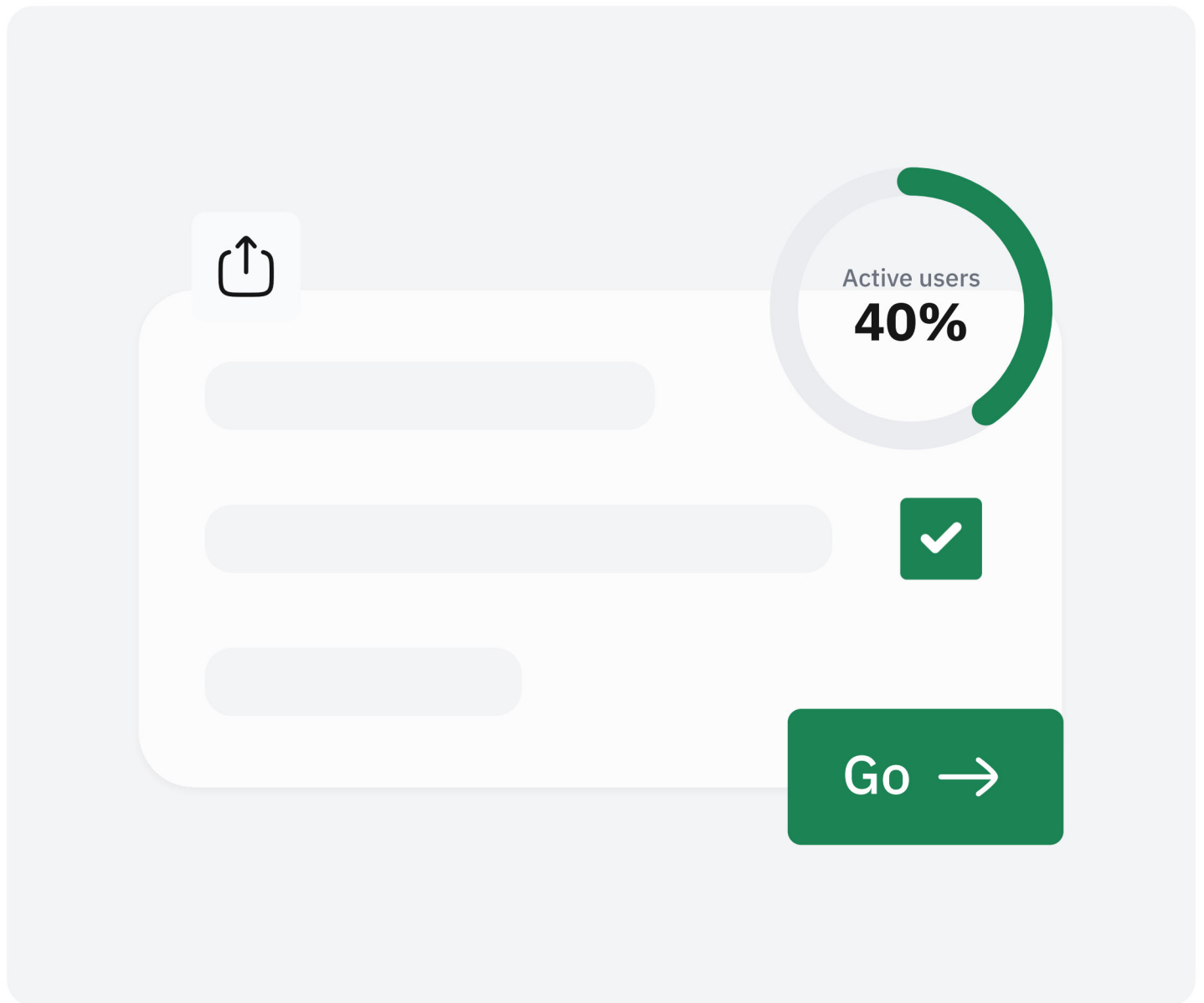
The Primary color “Green” was chosen to symbolize the values of growth, prosperity, unity, cooperation, and national solidarity embodied by the flag of the Kingdom of Saudi Arabia. This color stands as a key symbol in the unified design system aimed at harmonizing the vision and organizing the websites of government entities, with the goal of achieving progress, mutual collaboration, and a cohesive and integrated appearance.





Color serves a purpose.

color use must have a clear purpose, supporting a message or status that user can easily identify.



Colors create an impact.

Choosing a neutral color as a base for surfaces creates a clean canvas. This allows important elements with color to stand out more and captures the user's attention.

Platforms code Palettes

Platforms code organizes colors into three palettes—neutral, primary, and semantic—each serving specific purposes. Delve into each category to grasp their distinct functions and learn effective implementation methods across diverse government products.

Neutral Colors

Gray 25 FCFCFD	Gray 50 F9FAFB	Gray 100 F3F4F6	Gray 200 E5E7EB
Gray 300 D2D6DB	Gray 400 9DA4AE	Gray 500 6C737F	Gray 600 4D5761
Gray 700 384250	Gray 800 1F2A37	Gray 900 111927	Gray 950 0D121C

The neutral palette comprises black, white, and various shades of gray, providing a foundational base for an interface. These colors find application on surfaces, text, and layout elements, often signaling a shift in component states when integrated.

Primary Colors

SA 25 F7FDF9	SA 50 F3FCF6	SA 100 DFF6E7	SA 200 B8EACB
SA 300 88D8AD	SA 400 54C08A	SA 500 25935F	SA 600 1B8354
SA 700 166A45	SA 800 14573A	SA 900 104631	SA 950 092A1E

The primary color palette encapsulates the distinctive hues that define and represent a government platform or product's identity. These specific colors are strategically employed across elements, highlighting The Platforms code brand's essence and fostering recognition and

association within the interface. Utilizing these colors helps establish a visual connection with the brand and reinforces its presence across various components.

Secondary Colors

Gold color palette

Gold 25 FFFEF7	Gold 50 FFFEF2	Gold 100 FFFCE6	Gold 200 FCF3BD
-------------------	-------------------	--------------------	--------------------

The color gold, symbolizes richness, elegance, and prestige. It infuses a sense of luxury and sophistication into a product or visual identity. In design, gold is commonly used to emphasize crucial elements or communicate a feeling of exclusivity. It demands attention and projects warmth and positivity. Including gold in the color palette contributes to the overall design's depth and dimension, enriching user experience and strengthening the platform's identity.

Lavender color palette

Lavender 25

FFFCFF

Lavender 50

F9F5FA

Lavender 100

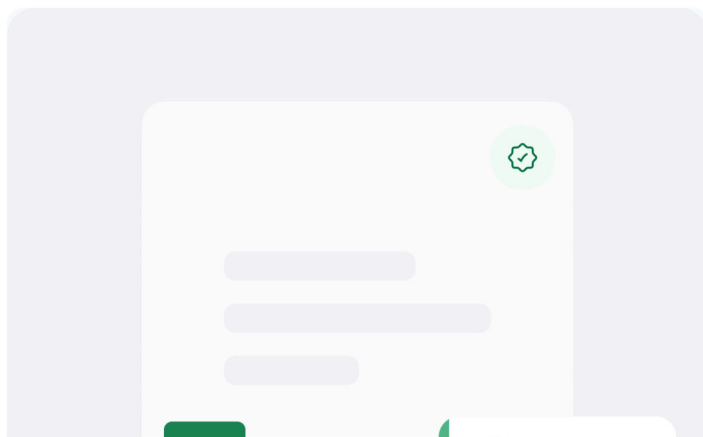
F2F9F5

Lavender 200

F1CCF8

The color lavender imparts a gentle touch of softness and serenity to the overall color scheme. It's frequently used to evoke feelings of calmness, relaxation, and femininity. In design, it can serve as an accent color to create contrast or encapsulate a sense of elegance and refinement. Lavender, when used judiciously, enhances visual appeal and promotes a harmonious and balanced aesthetic.

Semantic Colors



In Platforfom's code palette, certain colors are chosen as semantic indicators to swiftly convey feedback, error, warning, or information. Red signifies danger, yellow indicates caution, and green signals positive feedback. Consistent application and pairing with complementary indicators enhance user understanding, reducing mental effort.

Error color palette

The color red is deliberately selected for its frequent use in capturing immediate attention and conveying critical information, particularly in the form of error alerts. Red, being a highly noticeable and impactful hue, serves as a visual cue to users, signaling the presence of significant issues or validation errors that require prompt acknowledgment and resolution.

Warning color palette



The color yellow is intentionally chosen for its common use in attracting attention and communicating cautionary information, especially in the form of warning alerts. Yellow, as a vibrant and attention-grabbing hue, functions as a visual indicator for users, indicating the existence of important alerts or warnings that demand careful consideration.

Info color palette

The color blue is purposefully chosen for its frequent utilization in conveying informative content and providing a sense of clarity and trust, particularly in the form of informational alerts. Blue, known for its calm and trustworthy associations, acts as a visual cue to users, indicating the presence of informative details or notifications that offer additional insights.

Success color palette

The color green is thoughtfully selected for its prevalent use in symbolizing positive outcomes and successful operations, particularly in the form of success alerts. Green, a color often associated with growth and positivity, functions as a visual indicator to users, signifying the accomplishment of a task or the successful completion of an action.

Gradient Colors

A gradient color palette in Platforms code brings a dynamic and visually engaging dimension, enhancing user experience with versatile and modern design elements. It allows for the creation of stunning visuals, smooth transitions, and captivating backgrounds, fostering creativity and ensuring a contemporary and appealing interface.

Text Colors with Background

A text palette in a Platforms code design system ensures consistency, readability, and brand identity, supporting efficient and scalable design. It adheres to accessibility standards, accommodating users with diverse visual needs. The palette provides flexibility and guidance for designers, promoting harmonious and adaptable user experience.

Dark Text colors on Light backgrounds

Light Text colors on dark backgrounds

Platforms code Text Colors on Light Backgrounds

Semantic Text Colors on Light Backgrounds

Accessibility

This portion focuses on the vital role of contrast in creating user-friendly designs that are accessible to all users, including those with visual impairments. Here we explore the standards, best practices, and specific contrast requirements necessary for effective color utilization in digital interfaces.

Importance of Color Accessibility

Contrast is the difference in brightness between any two elements on a screen. It is a critical factor in enhancing readability and ensuring that all users can clearly perceive content. Effective contrast helps users with visual impairments distinguish text and interactive elements from their backgrounds.

Understanding Color Contrast

Color contrast refers to the difference in light between font (or graphic) color and its background color. Ensuring sufficient contrast helps users see and understand your content.

WCAG Contrast Requirements

The Web Content Accessibility Guidelines (WCAG) specify minimum contrast ratios to ensure that text and graphical content are legible:

- **Small Text:** Text below **24 px** must have a contrast ratio of at least **4.5:1**.
- **Large Text:** Text of **24 px** and larger should have a contrast ratio of no less than **3:1**.
- **Graphical Elements and User Interface Components:** These require a minimum contrast ratio of **3:1** to ensure that they are perceivable by users with visual impairments.

CodePlatform Color Palette

Our design system includes a robust palette consisting of twelve color grades — Black, White, and ten shades for each hue. Each color grade is designed to meet accessibility standards with ease.

Consistent Color Relationships

CodePlatform offers consistent color palettes where each color is assigned a specific role within the overall palette. These predefined relationships help maintain consistent contrast across various applications, ensuring legibility and understanding when combining text, color, and interactive elements with various backgrounds.

Visual Examples and Practical Application

To demonstrate the practical application of these principles, we include a dialog box with various elements highlighted. These elements use color combinations that pass both AA and AAA accessibility standards for contrast, ensuring that all content is accessible to a wide range of users.

Resources and Tools

- **WebAIM Contrast Checker:** <https://webaim.org/resources/contrastchecker>
- **EightShapes Contrast Grid:** <https://contrast-grid.eightshapes.com>

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Guidelines

Elevation

[← Previous](#)

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Foundations](#) > [Elevation](#)

Elevation

Elevation typically refers to the visual representation of the distance between elements or containers of elements and their background surface using shadows and lighting. This representation is used to create a visual hierarchy for the importance of content elements or containers of elements in the user interface.



Overview

What does 'Elevation' refer to?

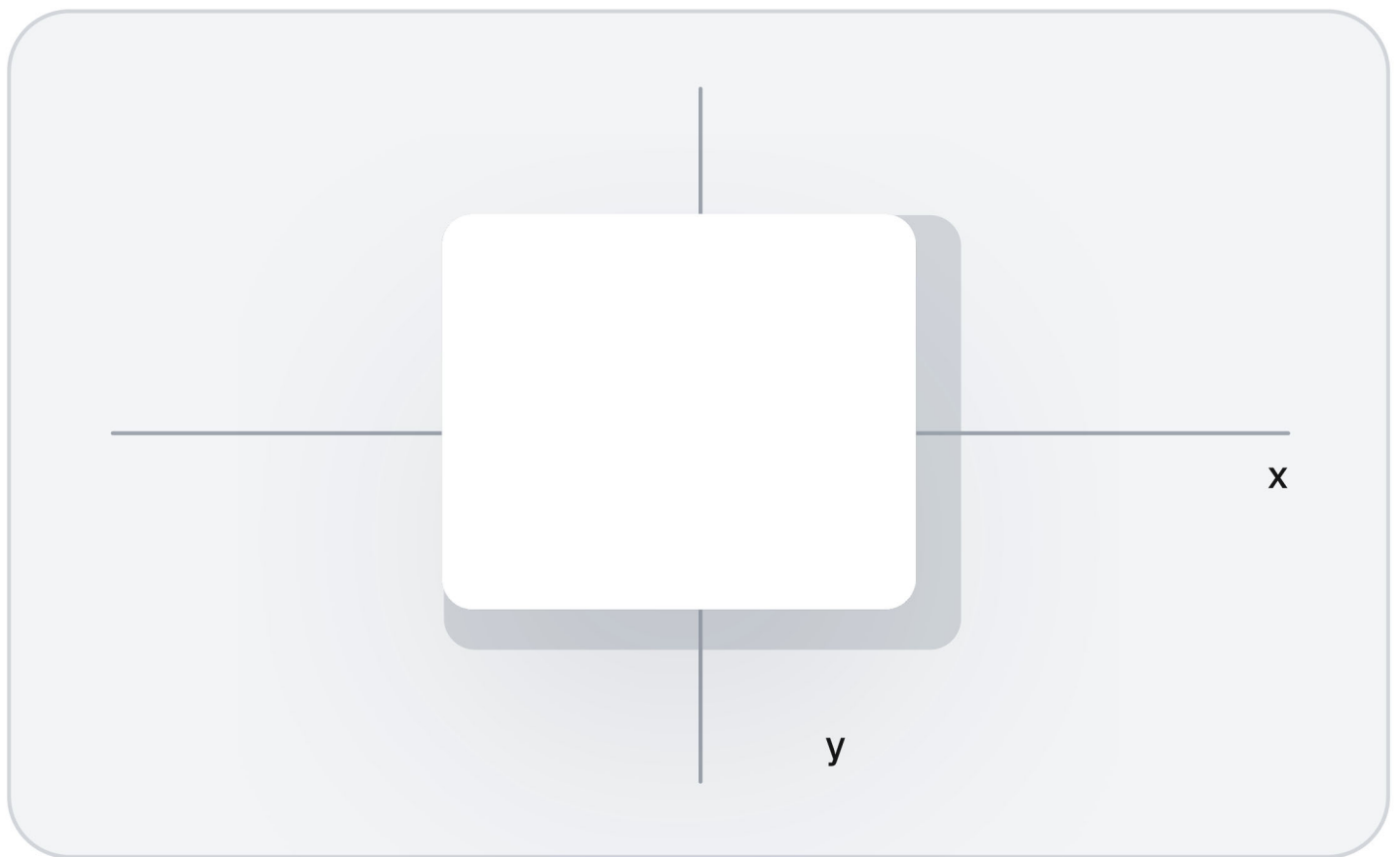
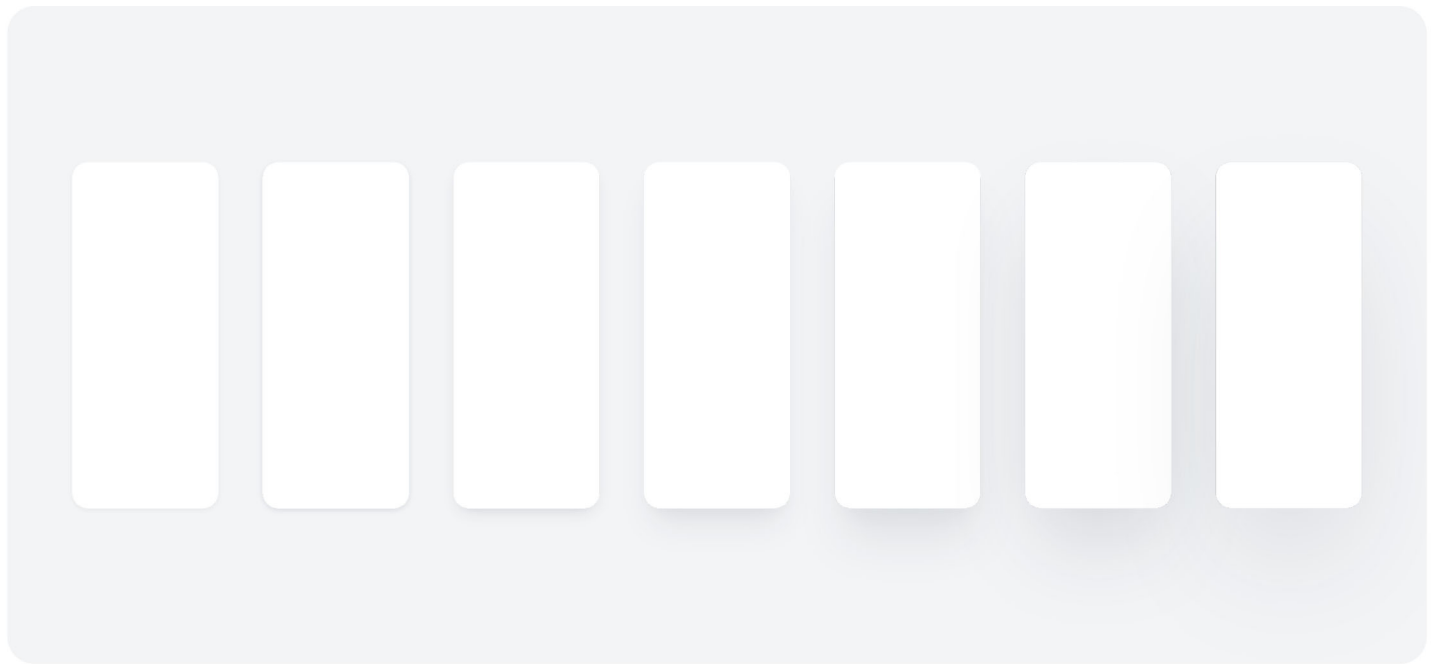
Elevation, visually portraying the distance between elements and their background through shadow and light manipulation, enhances interaction, directs focus to primary content, and improves user experience. Integral to modern UI design, it organizes elements for smooth interaction and understanding.

When is elevation utilized?

Elevation is used in user interface design to convey a sense of depth, hierarchy, and distinction among elements on the screen. It is applied to create a visual hierarchy that indicates the importance of elements, attracts attention, enhances the user experience, and makes the interface more realistic and accessible.

Shadows and creating elevation

Using shadows in design, by directing and shaping them strategically, achieves visual effects that evoke depth and interactivity. This method effectively enhances user experience, imparting a realistic quality to elements that fosters interaction with the interface.



Shadow direction

The angle at which the shadow is cast along the horizontal (x) and vertical (y) axes affects the shape and apparent dimension of the element.



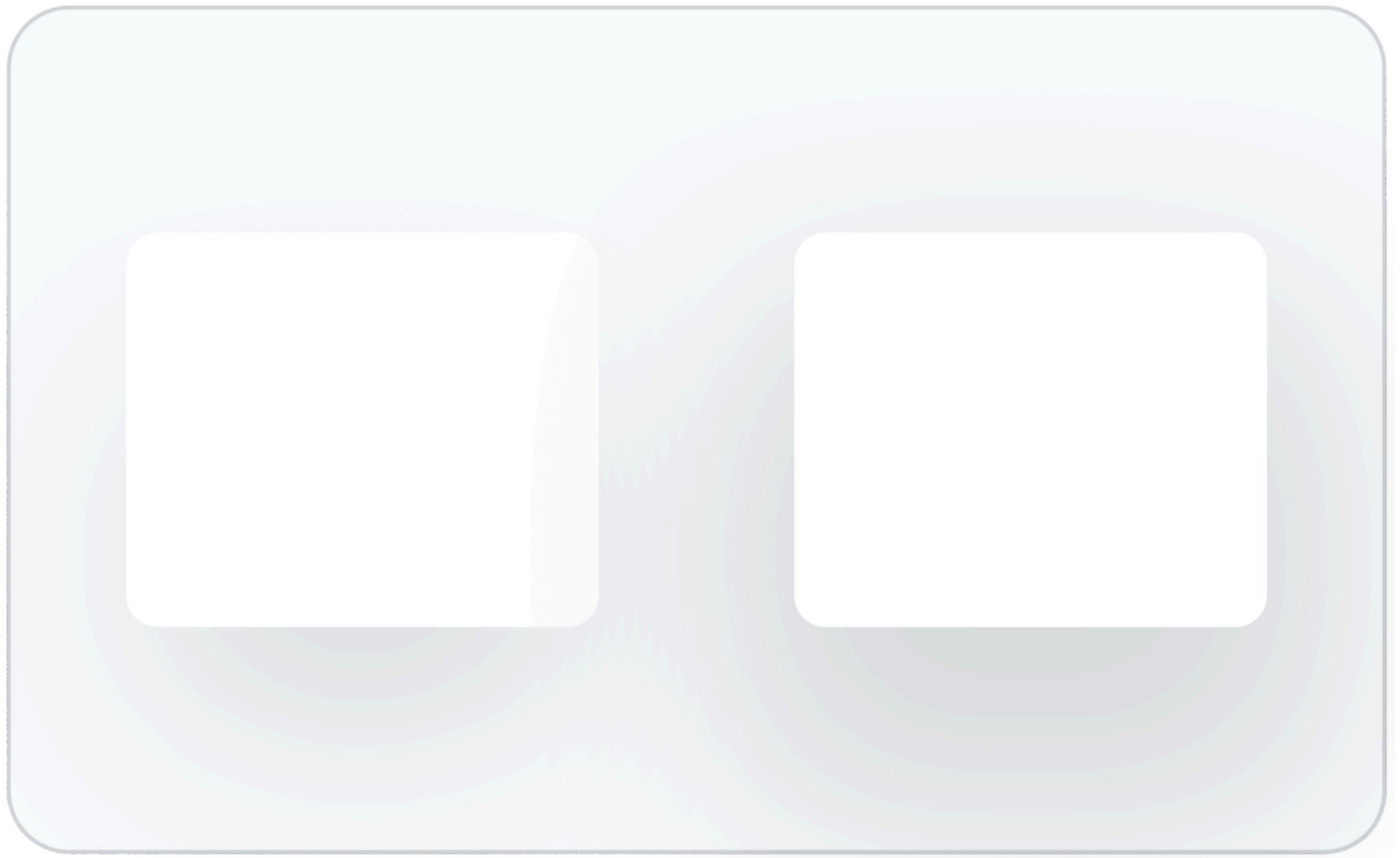
Shadow color

The color intensity and darkness of the shadow behind an element create visual depth and distance, typically being darker and less saturated than the element's primary color.



Shadow Spread

The extent of shadow spread from a certain element affects the size and distribution of the shadow among other elements in the design.

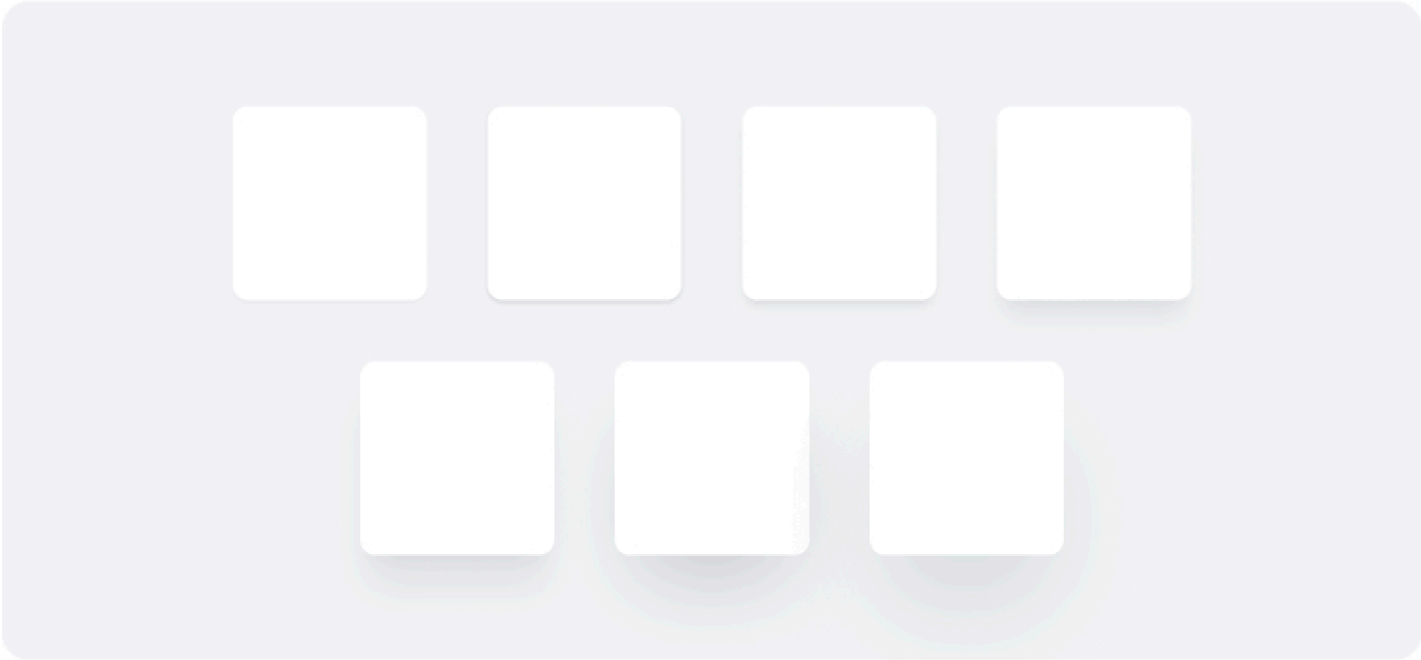


Shadow blur

The degree of blur or softening applied to the shadow edges, adding a smooth or diffused touch.

Shadows in Platforms code

The shadow effect in platforms code encompasses 7 different effects. The subtle shadow effect, which imparts a slight impression of elevation, relies on factors such as shadow direction, spread, blur, and color. This effect is commonly used to establish visual relationships between elements without overly burdening the user interface. Some shadow effects in platforms code consist of two layers of shadows, providing a smoother and more fluid appearance. In the following table, each effect will be mentioned along with its property values, such as shadow direction, and others.



Name	X-axis	Y-axis	Blur	Spread	Color	Color Opacity	Example
Shadow-xs shadow-xs	0	1	2	0	101828	5%	
Shadow-sm shadow-sm	0	1	3	0	101828	10%	
	0	1	2	0	101828	6%	
Shadow-md shadow-md	0	4	8	-2	101828	10%	
	0	2	4	-2	101828	6%	
Shadow-lg shadow-lg	0	12	16	-6	101828	8%	
	0	4	6	-2	101828	3%	
Shadow-xl shadow-xl	0	20	24	-4	101828	8%	
	0	8	8	-4	101828	3%	
Shadow-2xl shadow-2xl	0	240	48	-12	101828	18%	
Shadow-3xl shadow-3xl	0	32	64	-12	101828	14%	

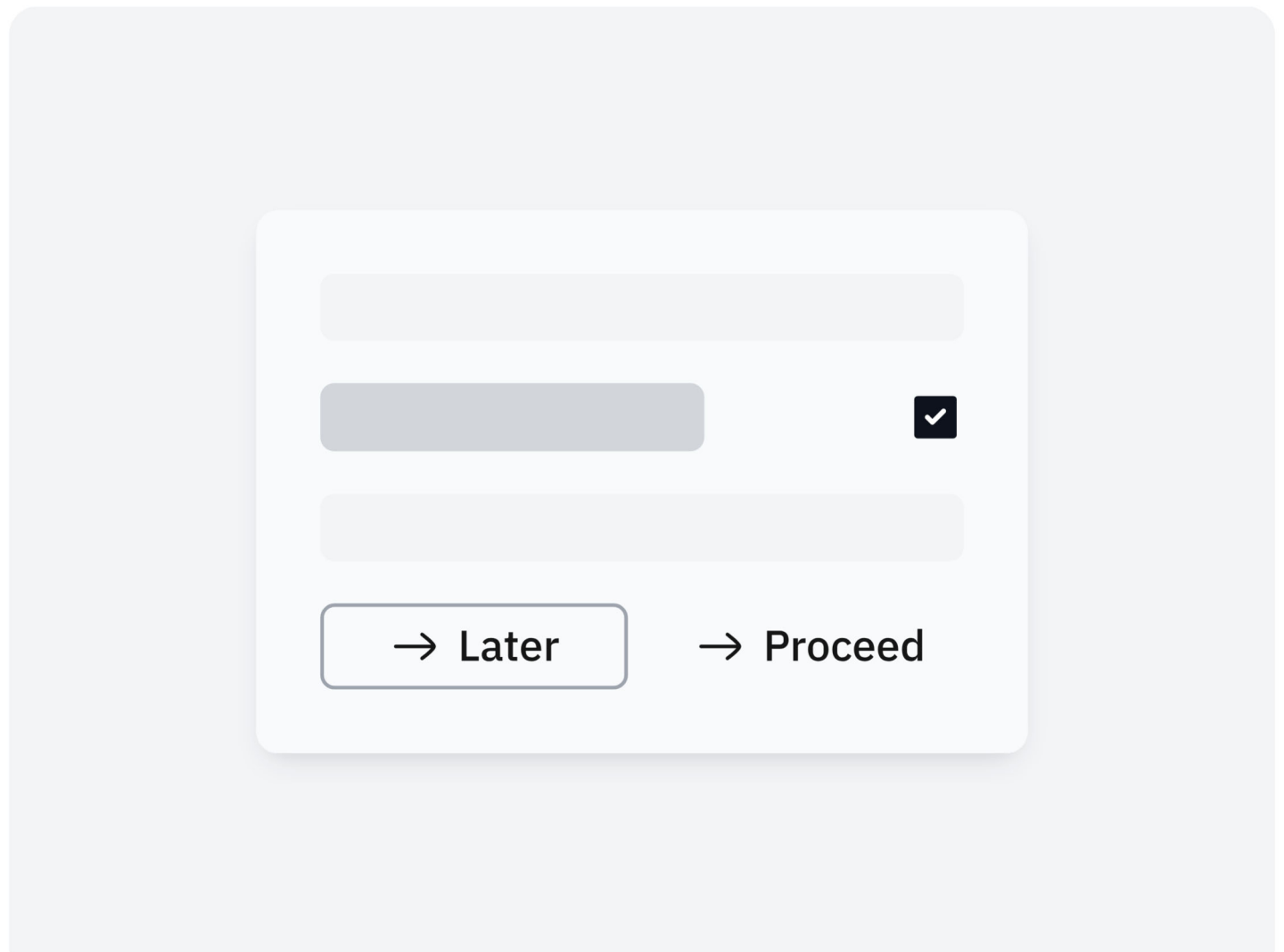
Backdrop Blur effects in Platforms code

The backdrop blur effect in user interface design is used to obscure the background and highlight foreground elements, creating a sense of depth and focus. A gradual blur is applied to the background to achieve this visual effect, which can enhance user experience and design appeal. However, it should be used cautiously to avoid negatively impacting text readability and interface usability.



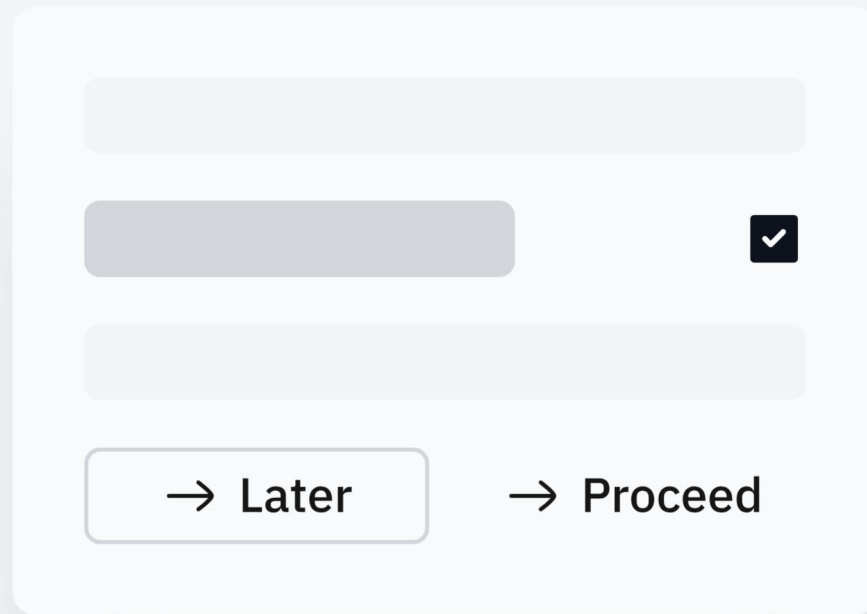
Name	Blur	Spacing
Backdrop-blur-sm	8	
Backdrop-blur-sm	16	
Backdrop-blur-lg	24	
Backdrop-blur-lg	40	

Tips to effectively employ elevation into your designs



✓ Do

Employ subtle elevation by applying light shadows and slight height to create a sense of depth without cluttering the user interface.



✗ Avoid

Using excessive use of elevation: refrain from placing too many shadows or exaggerated heights, as they can visually overwhelm the interface.

Accessibility

Elevation and shadow help indicate hierarchy and differentiate surfaces in UI design by simulating depth. However, these visual cues can be challenging for users with visual impairments. Here's how to ensure your use of elevation and shadow is accessible:

- Make sure that text and important graphical elements maintain a contrast ratio of at least **4.5:1** against their background to comply with **WCAG AA standards**. This is crucial for users with visual impairments.

- Consider using borders or differentiating outlines that can help define element boundaries more clearly for users who might not perceive depth cues effectively.
- For interactive elements like buttons that use elevation on hover or focus, ensure that these states are also distinguished by other means, such as color changes or underlining text, to make state changes clear to those who cannot perceive depth.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Color System

[← Previous](#)

Iconography

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Foundations](#) > [Iconography](#)

Iconography

Iconography (Icons or symbols) are considered a fundamental part of design elements, contributing to enhancing user experience, fostering visual communication, and distinctively reflecting the system's identity.



Overview

Icons play a vital role in enhancing user experience and facilitating interaction with interfaces. They are considered a fundamental element in unifying interfaces and enhancing differentiation

between various elements, thereby contributing to building a strong and distinctive design system identity. By utilizing a diverse set of appropriate and cohesive icons, effective visual communication is achieved, enhancing the user experience comprehensively.

We utilize the Hugeicons set, which offers a wide range of meticulously crafted icons designed to meet diverse needs. This set is continuously updated to stay relevant and useful.

By using the Hugeicons set, developers and designers can overcome the challenge of icon inconsistency and elevate their projects.

For designers, it supports Figma icon library, IconJar, SVGs and a magical Figma plugin

For developers, it supports NPM, SVG, JSZ, CDN, React, and React Native formats, ensuring comprehensive availability.

You can access the Icons here:



Platforms Code Icons Library

System Icons

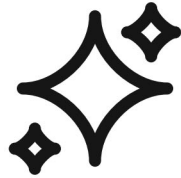


Hugeicons

Explore the Hugeicons set

Types of icons/icon containers in Platforms code

System icons are the icons used in designing interface elements and serve as the foundation for all types of icons, such as distinct icons and element icons.



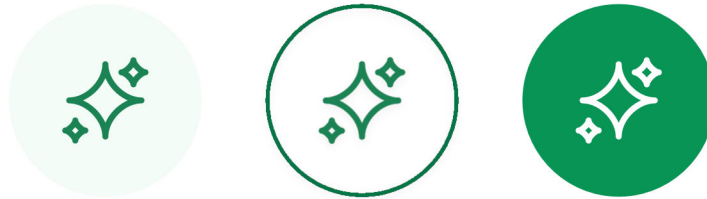
Main Icon

The fundamental icons comprise the icon library in the system, which can be used either independently or within system-designed icon containers or certain elements such as buttons and branching links.



Item icons

Containers containing system icons used in element design.



Featured icons

Distinct containers come in various styles and contain system icons used in designing elements such as alerts or adding visual symbols to enhance page content.



Feedback response icon

Icons representing system responses, including success messages and error alerts, facilitate understanding of the system's status and improve user interaction with it.



Rating star icons

Icons represent the star symbol with multiple instances of yellow color fills, commonly used to express service or product ratings.



Social media logo icons

Icons represent the social media platform logos.



National flags icons

Icons representing national flags, commonly used when entering personal information such as nationality or phone numbers.



Integration tools icons

Icons representing integration tools commonly used in digital products and platforms.



Payment method Icons

Icons representing logos of various payment service providers, commonly used on payment pages.



Help Icon

Icons indicating the presence of help content, commonly used in designing the tooltip element.

Icon Size

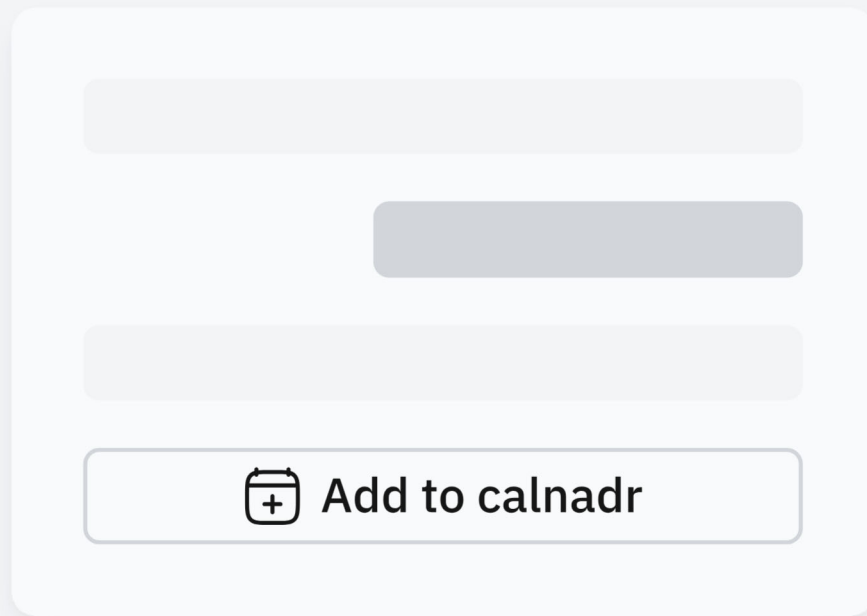
Icons in the Platforms code design system come in various sizes, depending on the container size:

- **10, 14, 16 pixels (Extra Small):** Ideal for extremely narrow spaces, such as in components like buttons and badges (small size).
- **18, 20 pixels (Small):** Suitable for small spaces, such as in components like badge components (small and extra small size).
- **24 pixels (Medium):** The standard size used in most interface components.

- **28, 32 pixels (Large):** Used sparingly to highlight content or in spacious areas. When using large icons, ensure they maintain accuracy and balance with other components in the user interface.

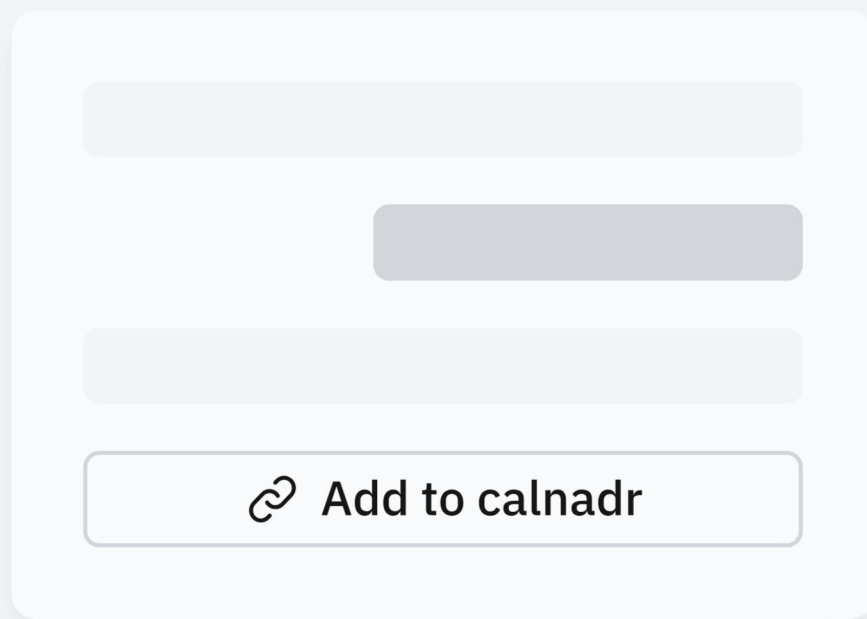
Consider the context of the icon and surrounding elements when selecting the appropriate size to ensure visual harmony and balance.

Tips to effectively employ icons into your designs



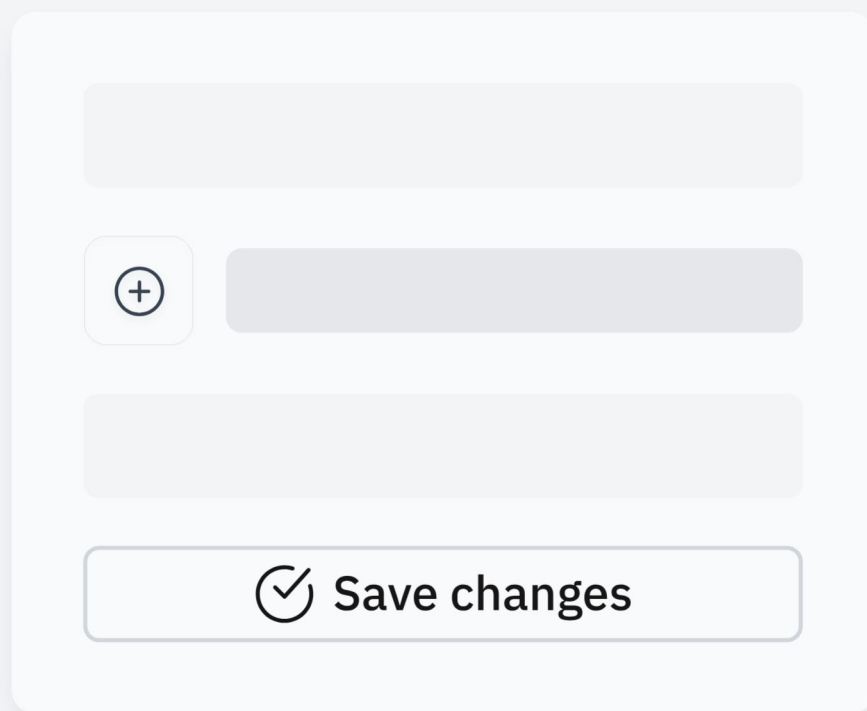
✓ **Do**

Use icons cautiously according to their appropriate function and purpose.



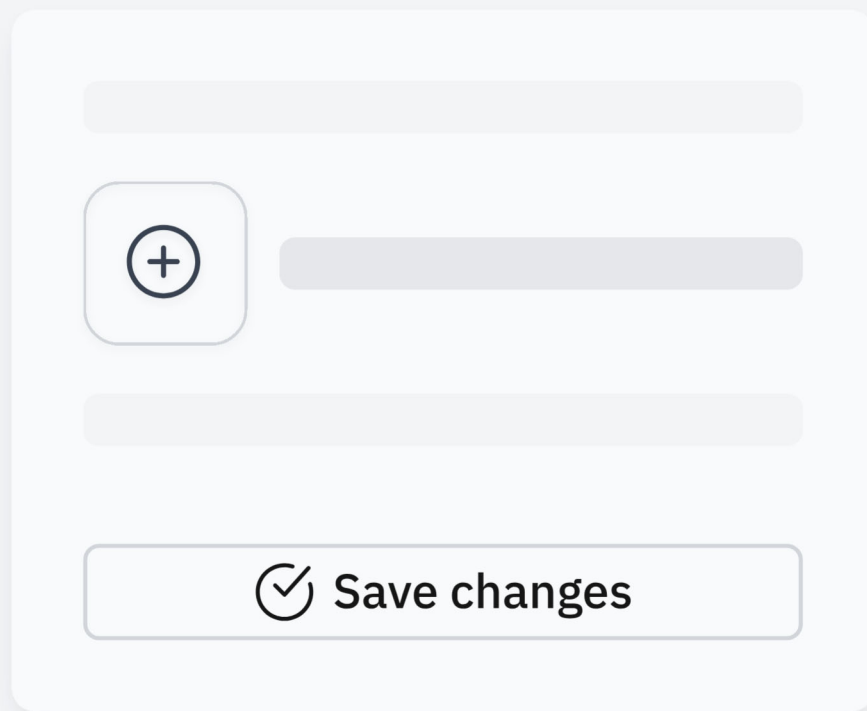
✗ Avoid

Using icons that differ from their meaning and purpose.



✓ **Do**

Consider the context of icons and surrounding elements when determining the appropriate size to ensure visual harmony and balance.



✗ Avoid

Using excessive use of elevation: refrain from placing too many shadows or exaggerated heights, as they can visually overwhelm the interface.

Accessibility

Iconography is crucial in user interfaces, offering visual cues that help with navigation, function communication, and aesthetic enhancement. To make icons accessible to everyone, including those with disabilities, it's important to follow certain accessibility guidelines. Here's a guide to implementing icons that are both usable and accessible:

- Icons that convey important information or functionality should have text alternatives using **alt** attributes (if they are implemented as `img` elements), or appropriate ARIA attributes **aria-**

label or aria-labelledby.

```
<button aria-label="Delete item"></button>
```

- Icons that are purely decorative and do not convey important information should be hidden from assistive technologies, typically by using `< class="text-md-semibold italic">aria-hidden="true"</i>` or ***alt=""***.

```
<span class="icon" aria-hidden="true"></span>
```

- When using SVGs for icons, include ***role="img"*** and provide an accessible name using ***aria-label*** or ***aria-labelledby***.
- For ****** elements used as icons, ensure that the ***alt*** attribute is used to describe the icon's function.

```
<svg role="img" aria-label="Search"><use xlink:href="#search-icon"></use></svg>
```

- If icons are implemented using CSS background images, these icons should have a text alternative nearby, as background images are not accessible to screen readers.
- If using icon fonts, ensure that each icon is accessible by providing appropriate ARIA attributes and that character mappings do not conflict with screen readers' default settings.
- Icons should be of an adequate size for interaction, typically at least ***44x44 pixels*** as per ***WCAG*** touch-target guidelines.
- Provide adequate spacing around icons, especially if they are used as interactive elements (like buttons or links).

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Elevation

Layout and Spacing

[← Previous](#)

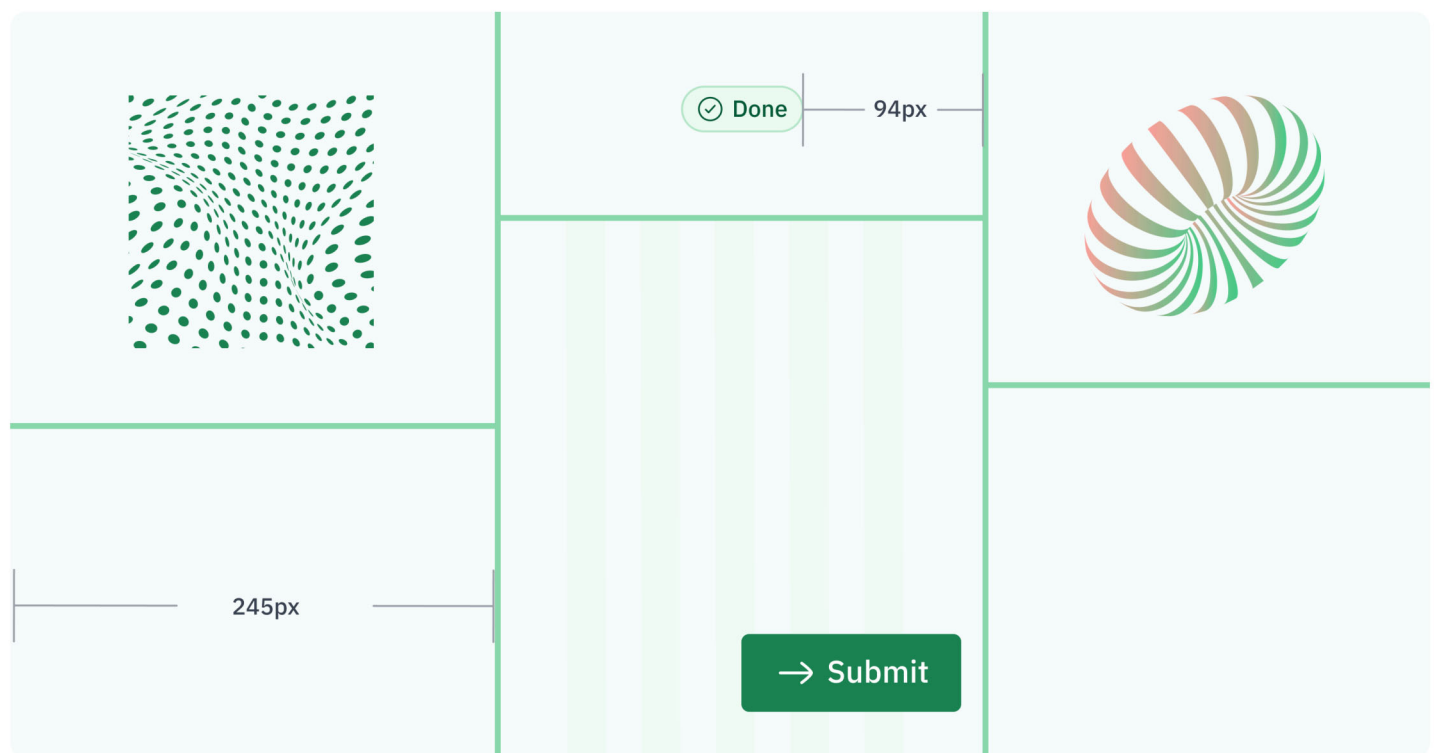
[Up Next →](#)



[Home](#) > [Guidelines](#) > [Foundations](#) > [Layout and Spacing](#)

Layout and Spacing

Platforms code layout system establishes a framework for applications, orchestrating spatial arrangements to establish connections between components, emphasize critical elements, and facilitate user decision-making seamlessly across diverse screen sizes.



Overview

What is Layout:

Layout refers to the arrangement of visual elements on a page or screen. It involves the positioning, grouping, and alignment of various elements such as text, images, and other design components to create a cohesive and aesthetically pleasing composition.

Why should we use Layout:

A well-thought-out layout enhances the readability, usability, and overall visual impact of a design. It helps guide the viewer's eye through the content in a logical and organized manner.

Grid system

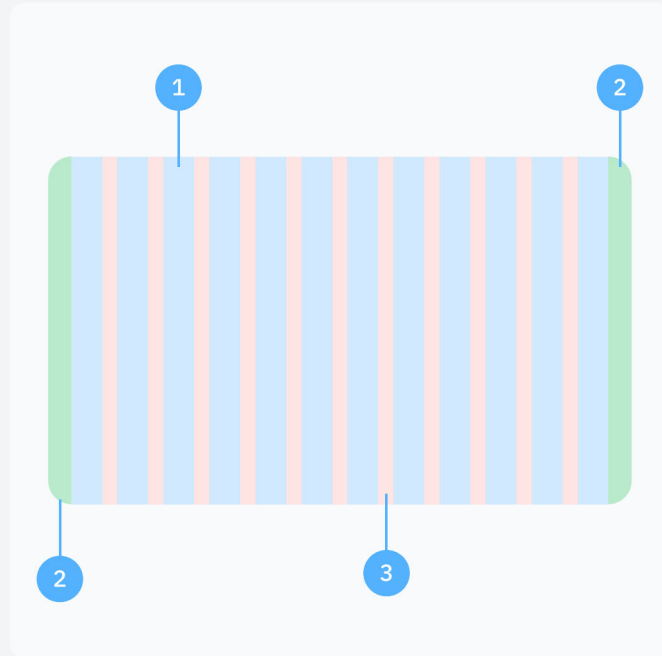
In Platforms Code design system, key components of the grid system include columns, margins and gutters. These elements work together to establish structure, consistency, and adaptability across different screen sizes and interfaces. A grid is a systematic arrangement of horizontal and vertical lines that form a framework for organizing content on a page or screen. It divides the available space into columns and rows, creating a structure that aids in the placement and alignment of design elements.

Columns grid system

In platforms code, column grids are commonly used. They divide the layout into vertical columns, providing a flexible structure for content organization. Designers prefer column grids, especially for complex layouts like web pages or app interfaces. For instance, websites often employ a 12-column grid for desktop views, reducing to 2-4 columns for smaller aspect ratios.

Component Anatomy

- 1 Columns
- 2 Margins
- 3 Gutters



Columns: Columns are vertical divisions within the layout, created by the grid system. The number of columns in a grid is a fundamental characteristic that influences the overall structure of the design.

Margins: Margins are the spaces between the outer edges of the layout and the content. They provide breathing room and define the outer boundaries of the design.

Gutters: Gutters are the spaces between columns. They help define the separation between different columns, preventing content from visually blending together.

Spacing

Name	Size (16px base)	Pixels	Spacing
spacing-none	0rem	0px	I
spacing-xxs	0.125rem	2px	II
spacing-xs	0.25rem	4px	III
spacing-sm	0.375rem	6px	III
spacing-md	0.5rem	8px	III
spacing-lg	0.75rem	12px	III
spacing-xl	1rem	16px	III
spacing-2xl	1.25rem	20px	III
spacing-3xl	1.5rem	24px	III
spacing-4xl	2rem	32px	III
spacing-5xl	2.5rem	40px	III
spacing-6xl	3rem	48px	III
spacing-7xl	5rem	64px	III

spacing-8xl	6rem	80px	
spacing-9xl	7rem	96px	
spacing-10xl	8rem	128px	
spacing-11xl	11rem	160px	

Width

Name	Size (16px base)	Pixels
width-xxs	20rem	320px
width-xs	34rem	384px
width-sm	30rem	480px
width-md	35rem	560px
width-lg	40rem	640px
width-xl	48rem	768px
width-2xl	64rem	1,024px
width-3xl	80rem	1,280px
width-4xl	90rem	1,440px
width-5xl	100rem	1,600px
width-6xl	120rem	1,920px

containers

Name	Size (16px base)	Pixels
<code>container-padding-mobile</code>	1rem	16px
<code>container-padding-desktop</code>	2rem	32px
<code>container-max-width-desktop</code>	80rem	1,280px

Paragraph max-widths

Name	Size (16px base)	Pixels
<code>paragraph-max-width</code>	20rem	720px

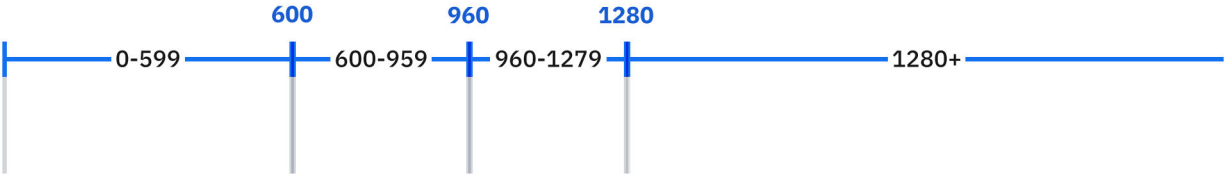
Responsive design

Responsive design is an approach to web design that aims to provide an optimal viewing and interaction experience across a wide range of devices and screen sizes. The primary goal is to ensure that a website or application adapts seamlessly to various devices, from desktop monitors to smartphones and tablets.

Break-Points

Breakpoints in responsive design are specific points at which a website's layout or styling is adjusted to accommodate different screen sizes or resolutions. Common breakpoints are set for various device categories such as desktops, tablets, and smartphones. When the browser window or screen size reaches a defined breakpoint, the layout may be altered, content may be reorganized, or styles may be adjusted to ensure optimal presentation and usability. Breakpoints are essential for creating a seamless and visually pleasing experience across the diverse landscape of devices that users may employ to access a website.

Name	Breakpoint range	Breakpoint
Small; Mobile	0-599	600
Medium; Tablet	600-959	960
Large; Desktop	960-1279	1280
X Large; Desktop	1280+	1280+



Accessibility

Creating an accessible layout means designing a user interface that is easy to interact with and understand for all users, including those with disabilities. This involves proper layout and spacing to minimize cognitive overload, enhance readability, and improve overall user experience. Here are key guidelines for designing accessible layouts and spacing:

Arrange elements in a logical reading order that matches the visual order. This is crucial for screen reader users and those who navigate with a keyboard.

`<header>...</header>`

`<nav>...</nav>`

`<main>`

`<article>...</article>`

`<aside>...</aside>`

`</main>`

`<footer>...</footer>`

Use spacing and borders to group related items. Consider using fieldsets for form elements and ARIA roles or properties to indicate groups and regions.

Provide adequate space between clickable and interactive elements to help users with motor impairments or those using touch screens. WCAG suggests a minimum target size of **44x44 pixels** for interactive elements

Use whitespace effectively to reduce cognitive load, improve readability, and separate sections of content. This helps users with cognitive limitations and those using screen magnifiers.

Use strong visual contrasts and alignments to define the layout structure without relying solely on color. Ensure that text has a contrast ratio of at least **4.5:1** against its background.

Use readable font sizes and adequate line spacing. WCAG suggests using a line height (line spacing) of at least **1.5** times the font size for body text.

Ensure the layout is responsive and adapts to different screen sizes and orientations. This helps users who access content on mobile devices or through different assistive technologies.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Iconography

[← Previous](#)

Typography

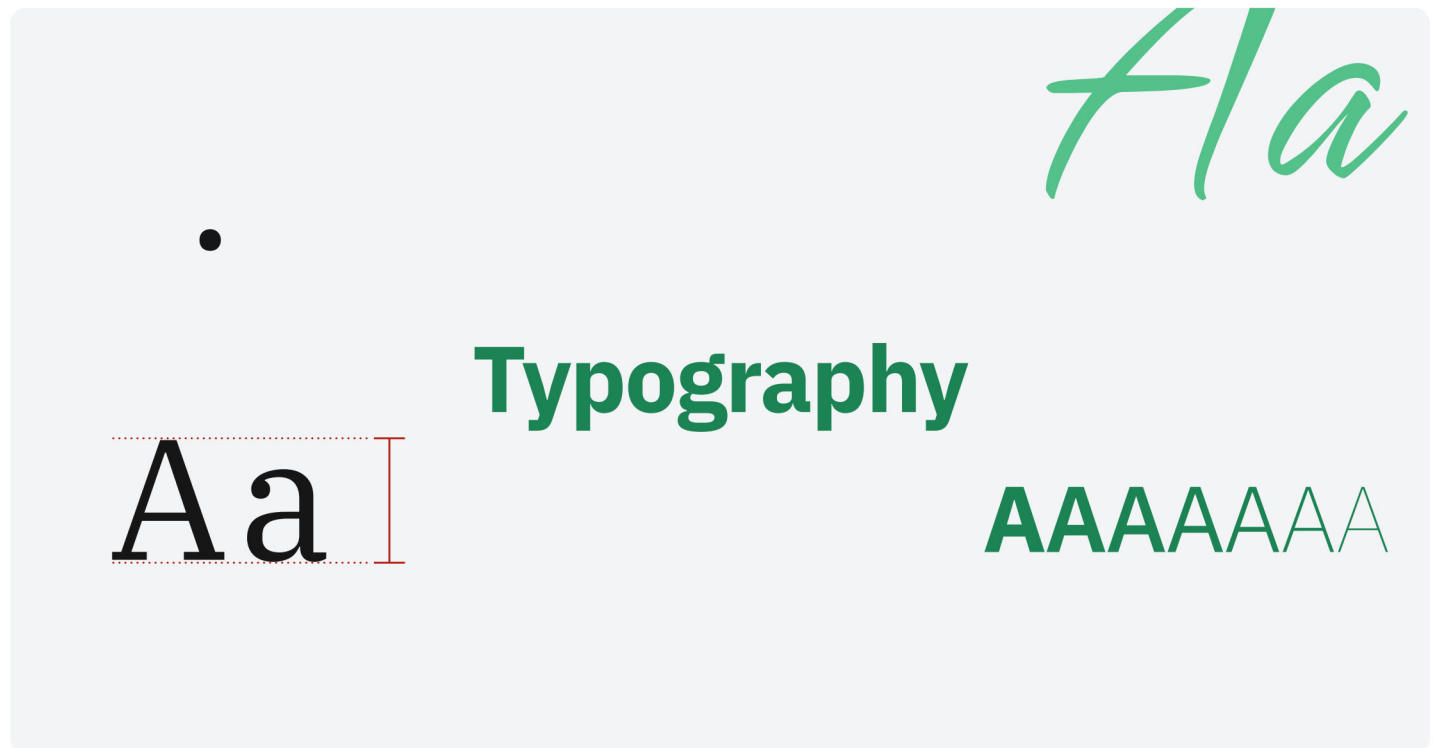
[Up Next →](#)



Home > Guidelines > Foundations > Typography

Typography

Typography in Platforms code design system refers to the selection and arrangement of fonts for consistent and cohesive visual communication.



Overview

The Platforms code categorizes typography into two main styles—Display and Text—each defined for different roles. Explore each style to understand effective implementation methods across a range of government products.

Display and text style

Display
Display
Display
Display

Text
Text
Text
Text

A display text style is intended for use at large sizes, making it well-suited for headings rather than extended passages of body text. This style is designed to grab attention and enhance visual impact in prominent areas of a design. As a general rule, 'text' styles are recommended for the majority of informational copy, including body text, labels, and various UI elements, especially when dealing with smaller sizes in user interfaces. Larger 'display' styles should be reserved specifically for headings to maintain a clean and organized typographic hierarchy.

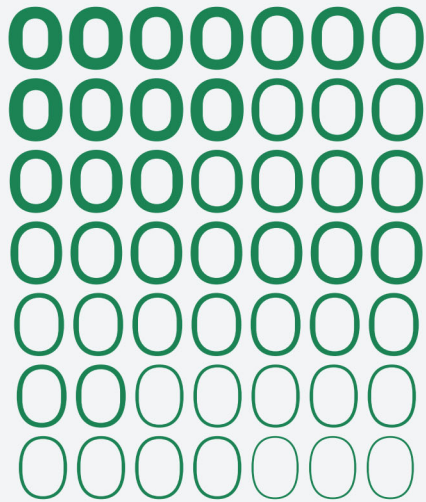
IBM

IBM Plex Sans is a versatile and modern typeface developed by IBM in collaboration with type foundry Bold Monday. Released in 2017 as part of IBM's new corporate design initiative. With a clean and contemporary design, Plex Sans offers excellent legibility across various platforms and devices, making it suitable for both print and digital applications. The font family includes a range of weights and styles, providing flexibility for different design needs.

Aa	Bb	Cc	Dd	Ee	Ff
Gg	Hh	Ii	Jj	Kk	Ll
Mm	Nn	Oo	Pp	Qq	Rr
Ss	Tt	Uu	Vv	Ww	Xx
Yy	Zz				
1	2	3	4	5	6
7	8	9	0		

ح	ج	ث	ت	ب	أ
س	ز	ر	ذ	د	خ
ع	ظ	ط	ض	ص	ش
م	ل	ك	ق	ف	غ
		ي	و	ه	ن
1	2	3	4	5	6
7	8	9	0		

Font weight



Fonts weight in the Platforms codes comes in various font weights, offering a range of options to suit different design requirements. The font family typically includes the following weights:

1. Regular
2. Medium
3. Semibold
4. Bold

These weights provide a spectrum from lighter to bolder options, allowing designers to choose the most appropriate weight for their specific application or visual emphasis.

Line Height

Crafting User Experiences

Line height: 38

Dive into the art and science of UX design.

Line height: 28

Learn how to create seamless and enjoyable interactions for users.

Line height, or leading in traditional typography, is the vertical space between lines of text. It influences readability and aesthetics, ensuring text is appropriately spaced. Designers adjust line height based on factors like font size and typeface to create a balanced reading experience, preventing text from appearing cramped or spread out.

Type scale

Display

Display 2xl

Font size: 72px / 4.5rem | Line height: 90px / 5.625rem | Tracking: -2%

Display 2xl

Display xl

Font size: 60px / 3.75rem | Line height: 72px / 4.5rem | Tracking: -2%

Display xl

Display lg

Font size: 48px / 3rem | Line height: 60px / 3.75rem | Tracking: -2%

Display lg

Display md

Font size: 36px / 2.25rem | Line height: 44px / 2.75rem | Tracking: -2%

Display md

Display sm

Font size: 30px / 1.875rem | Line height: 38px / 2.375rem

Display sm

Display xs

Font size: 24px / 1.5rem | Line height: 32px / 2rem

Display xs

Text

Text xl

Font size: 20px / 1.25rem | Line height: 30px / 1.875rem

Text xl

Text lg

Font size: 18px / 1.125rem | Line height: 28px / 1.75rem

Text lg

Text md

Font size: 16px / 1rem | Line height: 24px / 1.5rem

Text md

Text small

Font size: 14px / 0.875rem | Line height: 20px / 1.25rem

Text small

Text xs

Font size: 12px / 0.75rem | Line height: 18px / 1.125rem

Text xs

Text 2xs

Font size: 10px / 0.625rem | Line height: 14px / 0.875rem

Text 2xs

Tips to effectively use typography into your designs

Learn how to create
seamless and enjoyable
interactions for users.

Appropriate Height line: 28

✓ Do

Set an appropriate line height for improved readability. Ensure that text is spaced out enough to avoid crowding while maintaining a comfortable reading experience.

Learn how to create
seamless and enjoyable
interactions for users.

Small Height line: 16

✗ Avoid

Allow lines of text to become excessively long or short. Aim for an optimal line length to maintain readability and prevent user fatigue.

Crafting User Experiences

Dive into the art and science of UX design. Learn how to create seamless and enjoyable interactions for users.

Appropriate Body font: 14px

✓ Do

Prioritize readable font sizes for body text. Opt for a comfortable size that ensures legibility without causing strain for a positive user experience.

Crafting User Experiences

Dive into the art and science of UX design. Learn how to create seamless and enjoyable interactions for users.

Very Small Body font: 10px

✗ Avoid

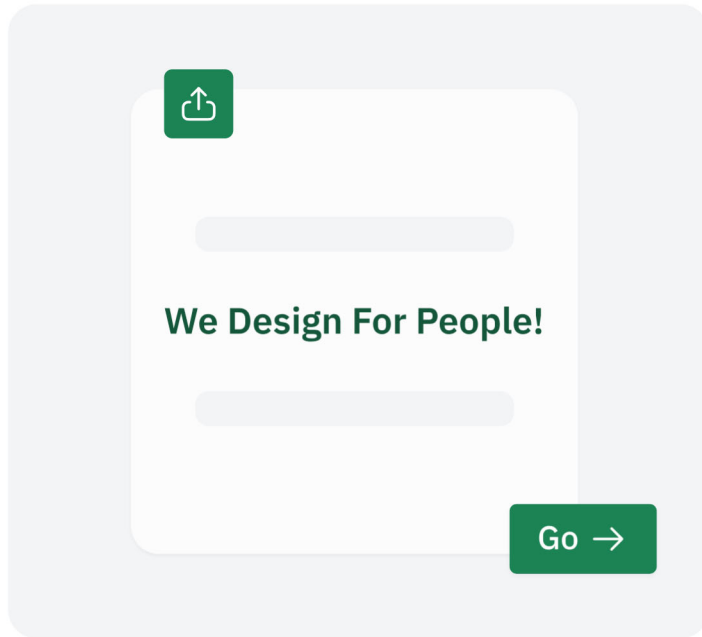
Use excessively small font sizes, especially for body text. Ensure that text is comfortably readable without causing strain.

⁴
Crafting User Experiences

✗ Avoid

Neglect consistent spacing. Inconsistent margins, padding, or line heights can disrupt the visual flow and make the design feel disjointed.

Accessibility and Text Colors



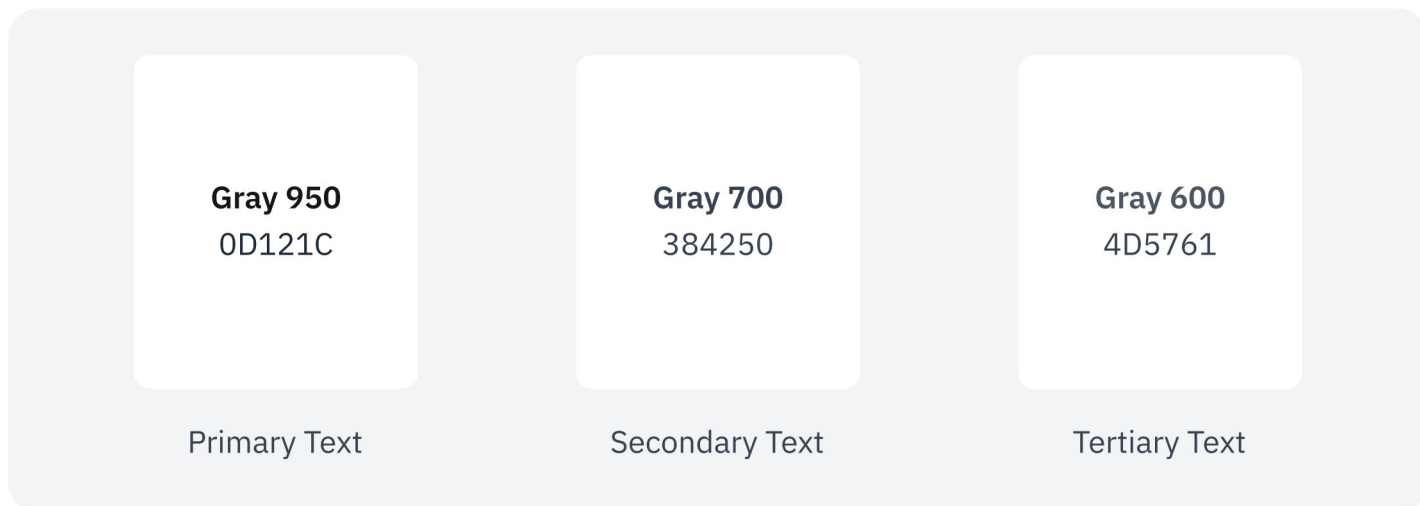
Text palettes in Platforms code design system ensures consistency, readability, and accessibility for users with diverse visual needs. It not only meets accessibility standards but also offers flexibility and guidance to designers, ensuring an efficient, harmonious, and scalable user experience.

Our aim is to comply with WCAG 2.1 AA accessibility standards. To ensure accessible color contrast in text, we suggest using text colors like (Gray 500, Gray 600, Gray 700, and Gray 950) for lighter backgrounds with a number 400 and below. For backgrounds numbered 500 and above, it is advisable to use white text.

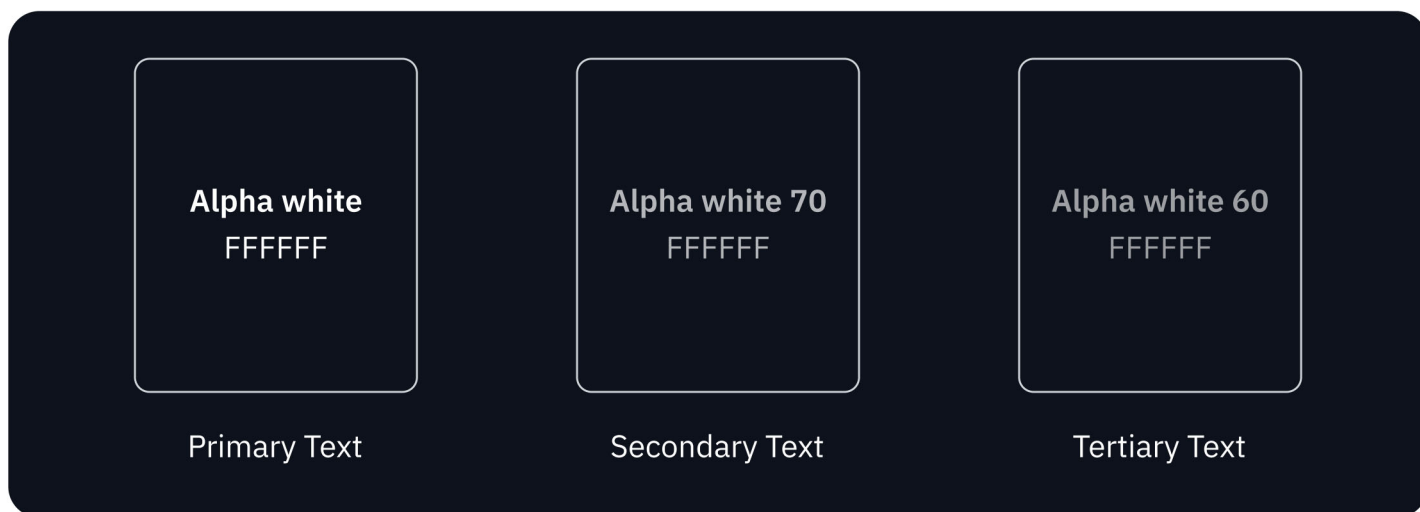
According to WCAG 2.1 standards, the minimum contrast ratios are as follows:

- Level (AA): 4.5:1
- Level (AA) for large text (above 18.5 Bold or 24 Regular): 3:1
- Level (AAA): 7:1
- Level (AAA) for large text (above 18.5 Bold or 24 Regular): 4.5:1

Dark Text colors on Light backgrounds



Light Text colors on dark backgrounds



Platforms code Text Colors on Light Backgrounds

Semantic Text Colors on Light Backgrounds



Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Layout and Spacing

[← Previous](#)

Buttons

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Actions](#) > [Buttons](#)

Buttons

Buttons are interactive components that trigger specific actions. The label on a button describes the action that will occur when it is clicked by the user.

Live Demo



Visual



Code

Button

Test the preview of the buttons comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Appearance

The Platforms Code Design system includes four types of buttons: Standard Buttons, menu buttons, destructive buttons, icon buttons, and close buttons. Standard buttons are used for primary actions, while Destructive buttons are used for significant actions like data deletion, Menu buttons provide multiple options, and close buttons dismiss dialogues or exit functions. This range of buttons caters to a variety of user interactions.

Types

Buttons are available in four variants:

- **Standard Button:** A standard button is a fundamental user interface element that triggers a single action or event. It's typically used to initiate common and routine actions within an application or a website.
- **Destructive Button:** Destructive buttons perform irreversible actions, like deleting data or cancelling important processes. To avoid accidental clicks, they need thoughtful use and user confirmation.
- **Menu Button:** Menu button functions as a toggle to reveal a menu presenting various options.
- **Icon Buttons:** Every type can take the form of an icon button, where the emphasis is primarily on one visual symbols (icon) to communicate their intended purpose.



Standard button



Destructive Button



Menu Button



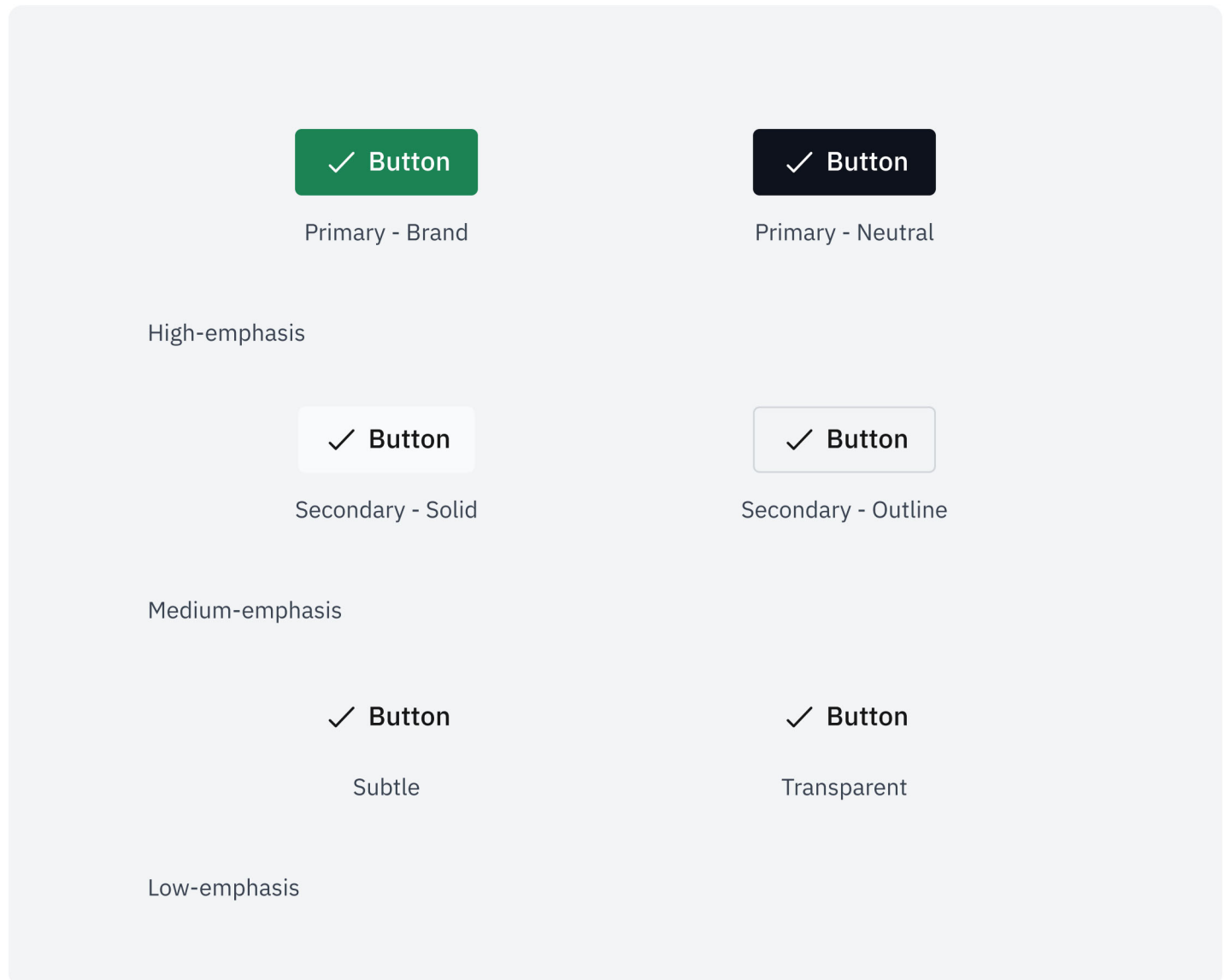
Icon Buttons

Styles

Buttons are available in four styles:

Within the Platforms code, buttons are designed to convey three distinct levels of emphasis, each denoting a specific degree of importance. High-emphasis buttons serve a pivotal role by being strategically employed for actions that demand prominence and need to stand out significantly. These high-emphasis buttons are typically reserved for critical calls-to-action (CTAs) and primary actions, ensuring they capture the user's attention effectively. On the other hand, lower-emphasis buttons are designated for actions of reduced importance, often associated with secondary or less critical functions within the interface. This meticulous differentiation in emphasis levels

contributes to a well-structured and user-centric design, guiding users through the platform with clarity and precision.



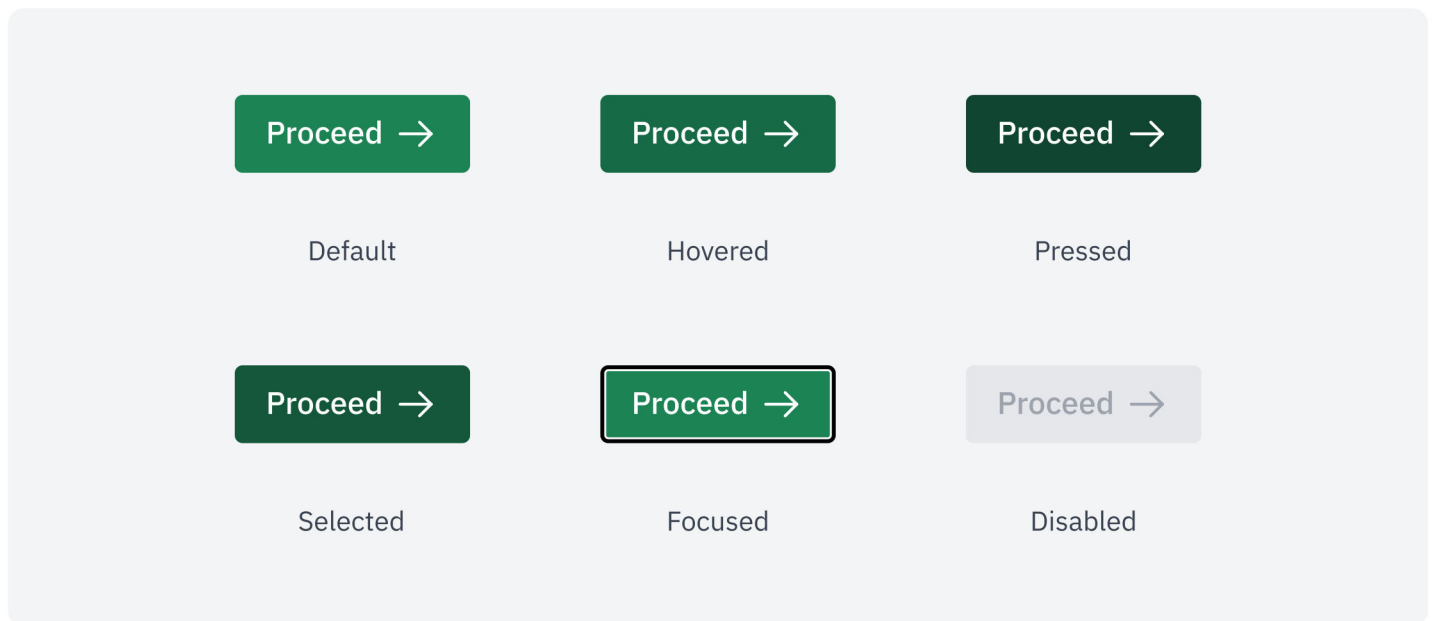
In Platforms Code, buttons convey three levels of emphasis: high, medium, and low. High-emphasis buttons stand out for critical actions like primary calls-to-action (CTAs). Lower-emphasis buttons are for less crucial functions. This differentiation ensures clarity and precision in user interaction.

Behaviors (States)

Buttons in the Platforms Code exhibit various behaviors depending on user interaction. The states of button behaviors are:

- **Default:** The default state of a button when it is not being interacted with. It appears as a standard, clickable button.

- **Hovered:** Occurs when the user moves the cursor over the button without clicking.
- **Pressed:** This state is triggered when the user clicks or taps the button. It provides feedback that the button has been successfully activated.
- **Selected:** Applicable in scenarios where buttons can be toggled on and off. Indicates that the button is currently selected or activated.
- **Disabled:** The disabled state of a button is a condition where the button is not clickable or interactive. This state is often used in user interfaces to communicate that the button cannot be activated under the current circumstances.



Icon Placement in Buttons: Leading Icon and Trailing Icon

Icons in Platforms Code the icon can be a leading icon or trailing icon. A trailing icon follows the button label, while a leading icon precedes it. Both types serve similar purposes, offering visual cues and enhancing the user interface. However, their use varies based on content flow, level of emphasis or interface directionality.

The leading icon often draws attention first, followed by the button text. This can be useful when you want to emphasize a specific action or feature.

Trailing icons work best as a complement to the button label, especially when they serve a supplementary or visual role rather than a primary functional indicator. This approach is suitable when the emphasis is on the text, making it common in situations where the text provides a clear description of the button's purpose.



Leading icon

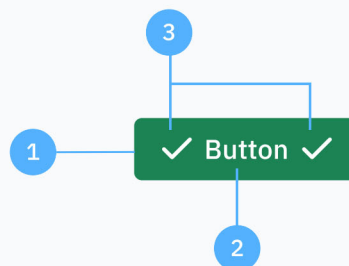


Trailing Icon

Anatomy

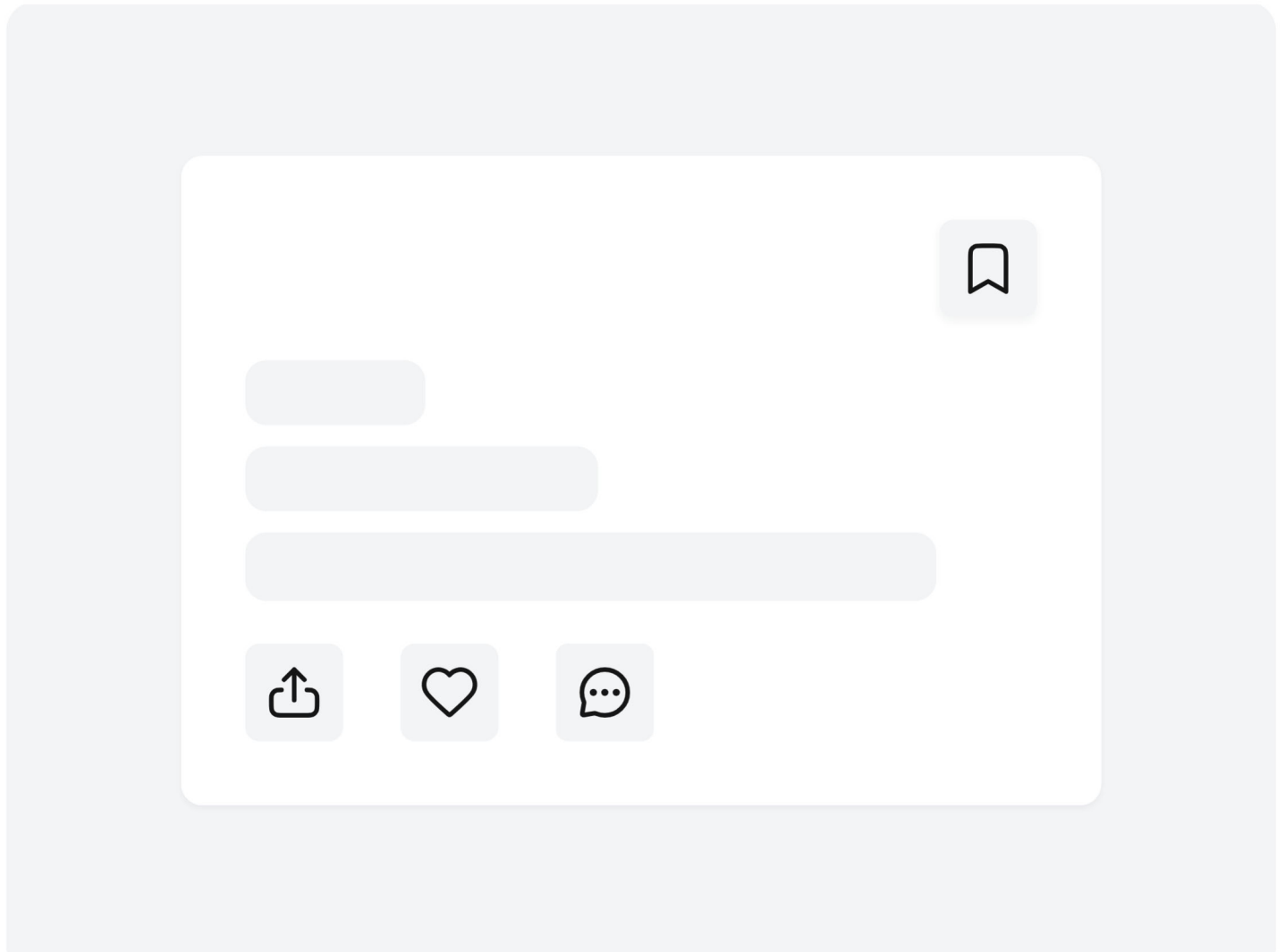
Component Anatomy

- 1 Container
- 2 Button label
- 3 Leading or trailing Icon



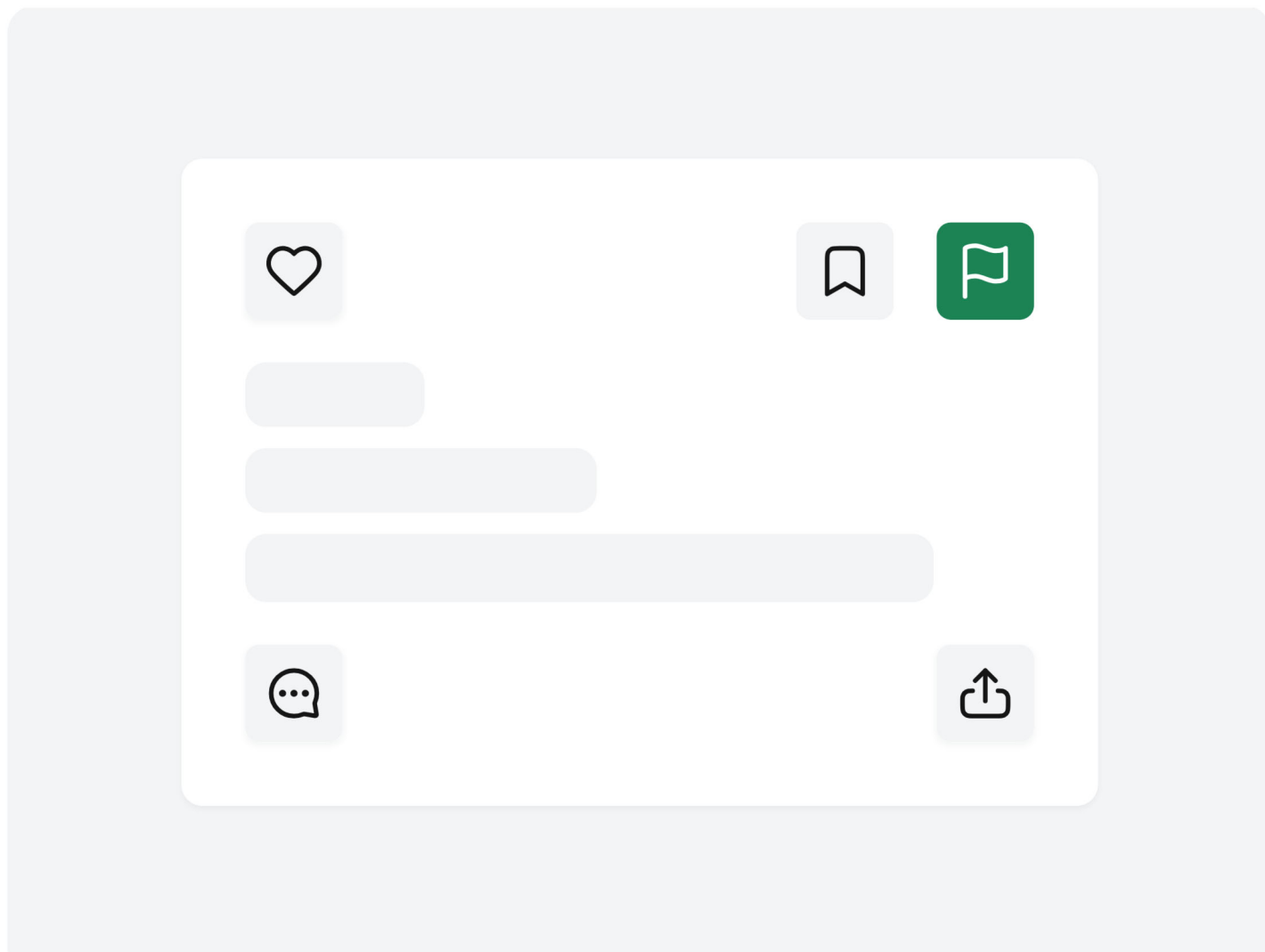
- **Container:** The clickable area that holds all the elements.
- **Button label:** Displays the title of the button.
- **Leading or trailing Icon:** Provides visual representation of the button's purpose or name.

Tips to effectively use Button into your designs



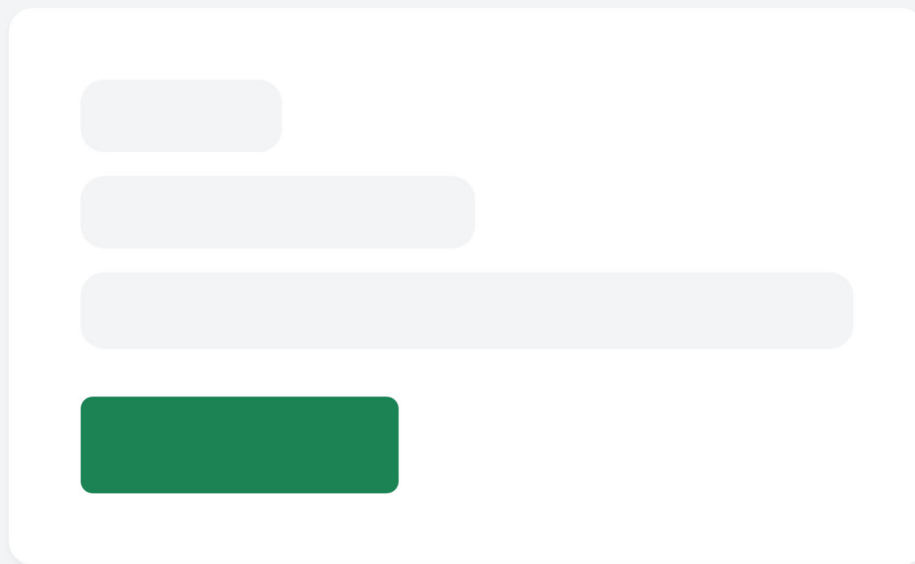
✓ Do

Group related actions together, creating cohesive button clusters that align with specific functionalities. This aids users in quickly identifying and accessing relevant actions.



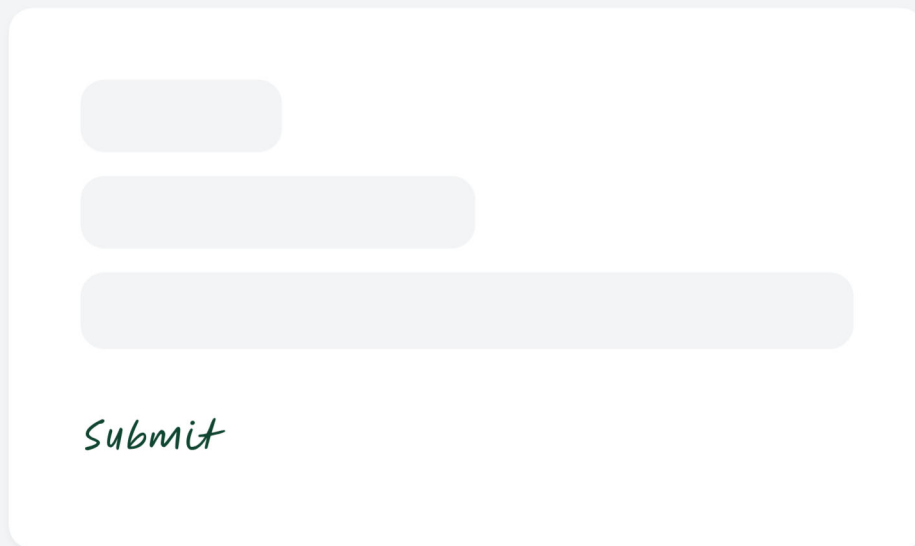
✗ Avoid

Scatter related actions randomly across the interface, as this can confuse users and make it challenging to locate and access relevant actions. Keeping actions grouped together ensures a more intuitive and organized user experience.



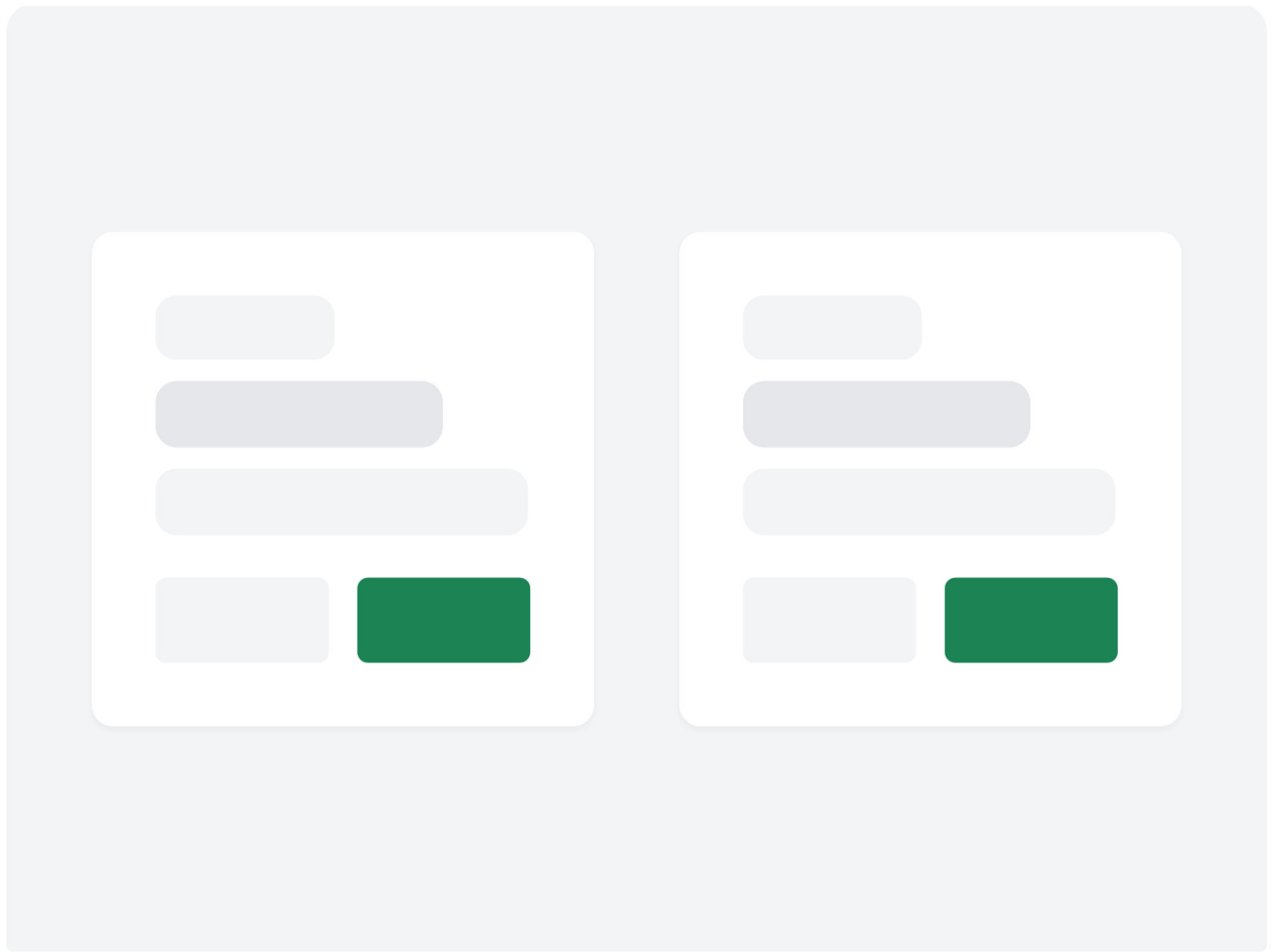
✓ **Do**

Emphasize the placement of primary actions in prominent positions, such as the bottom of forms or in central locations within the interface.



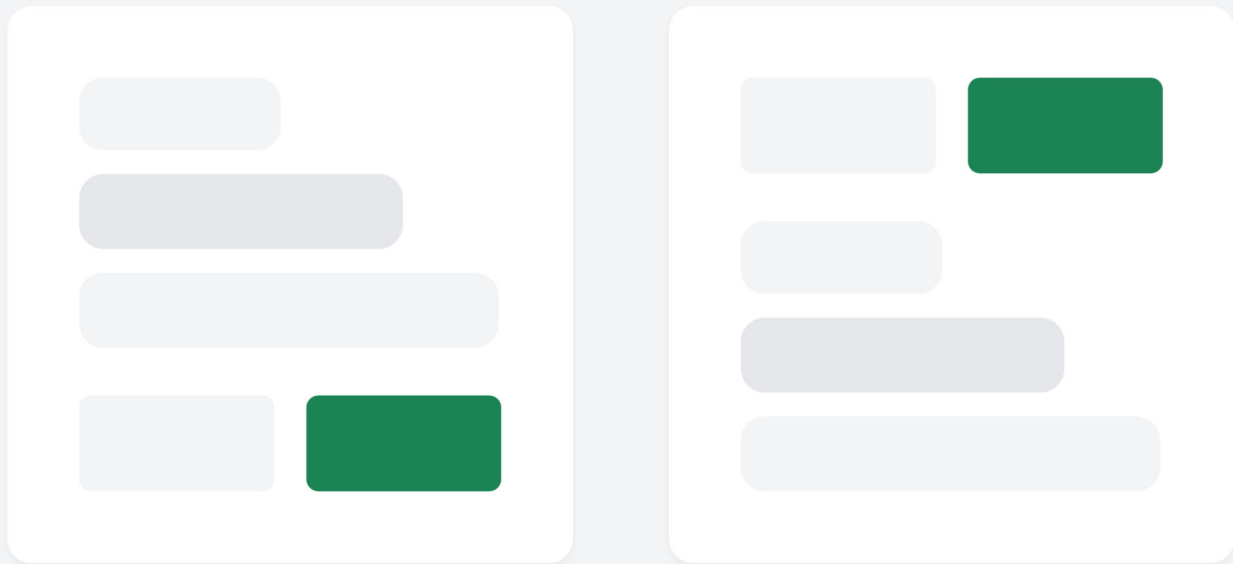
✗ Avoid

Hiding critical buttons or bury them within menus where users might struggle to find them. Prioritize visibility for essential actions.



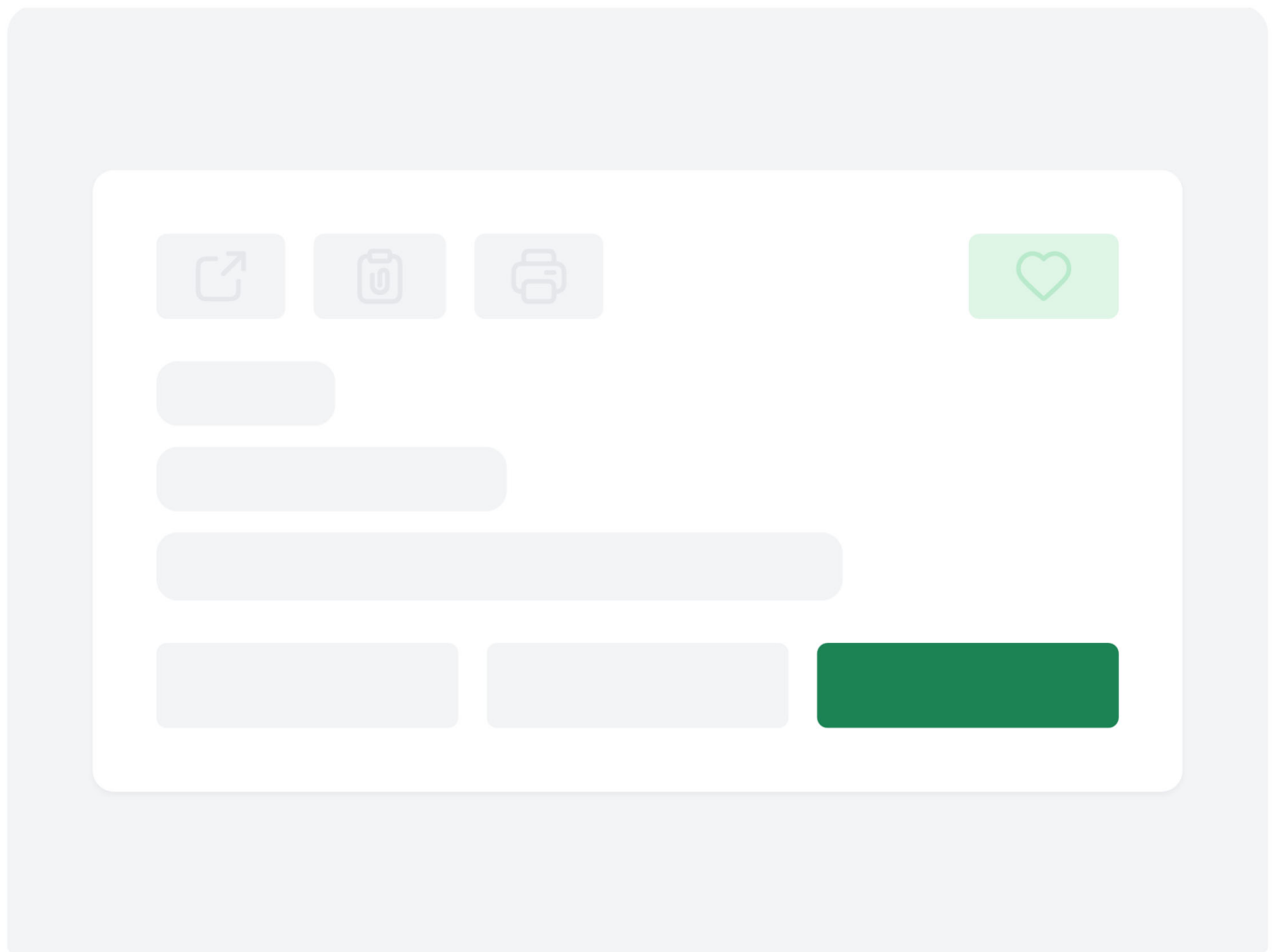
✓ **Do**

Maintain consistent button placement across different screens and components within the design system. This establishes a familiar pattern for users, enhancing overall usability.



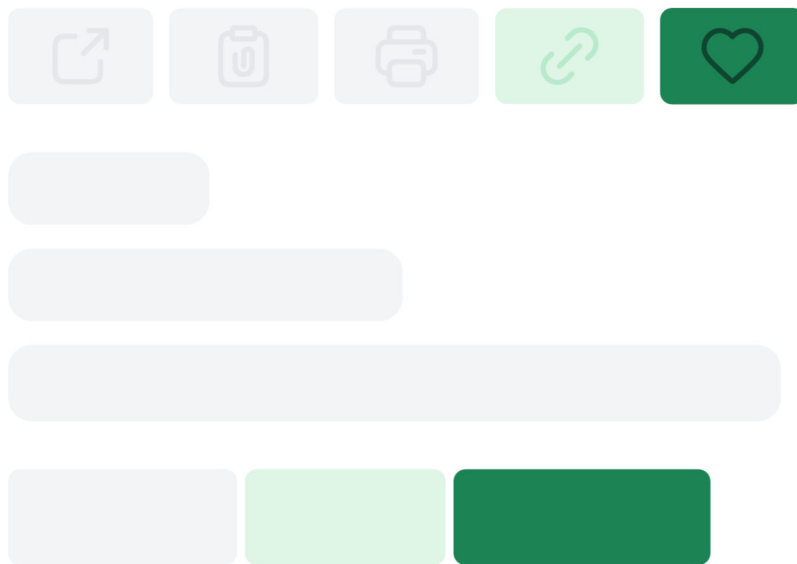
✗ Avoid

Change button placement haphazardly across screens. Inconsistency can lead to confusion and hinder the development of a smooth user flow.



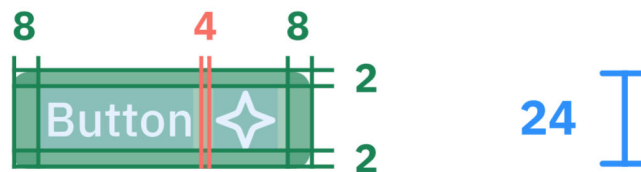
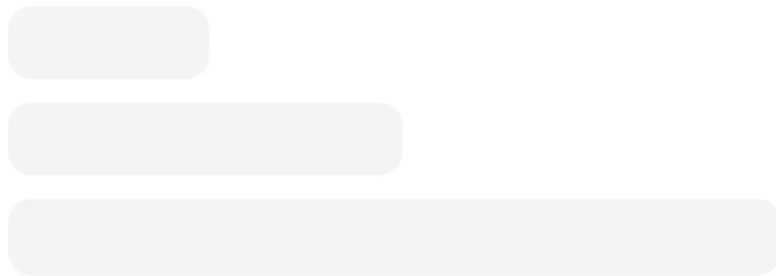
✓ **Do**

Leverage whitespace effectively around buttons to prevent crowding. Clear spacing enhances visual clarity and reduces the likelihood of accidental clicks



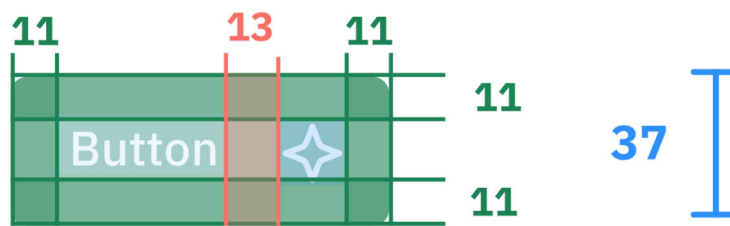
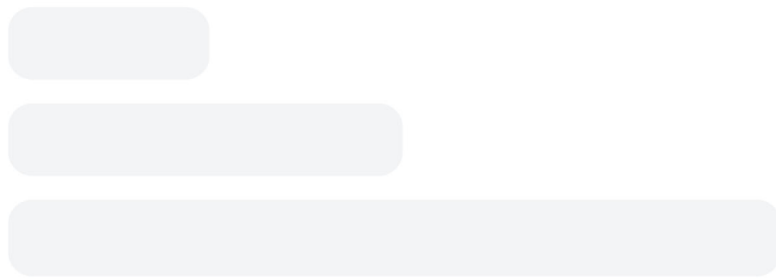
✗ Avoid

Overcrowd the interface with too many buttons or place them in close proximity without clear visual separation. This can lead to user confusion and accidental clicks.



✓ Do

Maintain uniformity by adhering to the established button anatomy outlined in the Platforms Code guidelines. Consistency ensures a cohesive visual language and enhances user familiarity across the interface.



✗ Avoid

Deviate from the established button anatomy as outlined in the Platforms Code guidelines. Inconsistency in button design can lead to confusion and disrupt the overall visual coherence of the interface.

Accessibility

Buttons serve as an essential tool for providing contextual information and aiding workflows, particularly for users with disabilities. It is crucial to ensure that these modal are accessible, allowing all users to perceive, understand, operate, and interact with them effectively.

- Always use the **<button>** HTML element for buttons. This ensures that the button is accessible by default, as it supports keyboard focus and the disabled attribute natively.

- Non-semantic HTML like **<div>** should not be used for actions. If you must use an **<a>** for a button, ensure it performs navigation. Always provide a valid href attribute.
- Critical and Warning Notifications: **Use role="alert"** for immediate announcement by assistive technologies.
- Other Notifications: Apply **role="status"** for announcements that are important but less urgent.
- Equip all notifications with an **aria-live** attribute to announce content updates via assistive technologies. Use **stopAnnouncements** prop to manage these announcements
- Use **aria-describedby** to clarify the purpose of notifications when announced or focused. Utilize the title content or the entire notification content for this attribute, depending on the presence of a title.
- Provide visual feedback for different states such as **hover**, **focus**, and **active**. This helps users understand that the button is interactive and can be clicked
- Ensure text or icons on the button have a contrast ratio of at least **4.5:1** against the button background to comply with **WCAG AA standards**.
- Ensure buttons are large enough to easily interact with, especially for users with motor impairments. A recommended minimum touch target size is **44x44 pixels**.
- Use aria-label when the button does not contain text that describes its function, especially for icon buttons. `<button aria-label="Close"></button>`
- Ensure that buttons can receive focus. The default **<button>** tag is focusable, but if you're using other elements (which should be avoided for actions), set **tabindex="0"**.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Typography

Dropdown

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Actions](#) > [Dropdown](#)

Dropdown

Dropdowns present a list of options from which a user can select one option, or several.

Live Demo



Visual



Code

Label

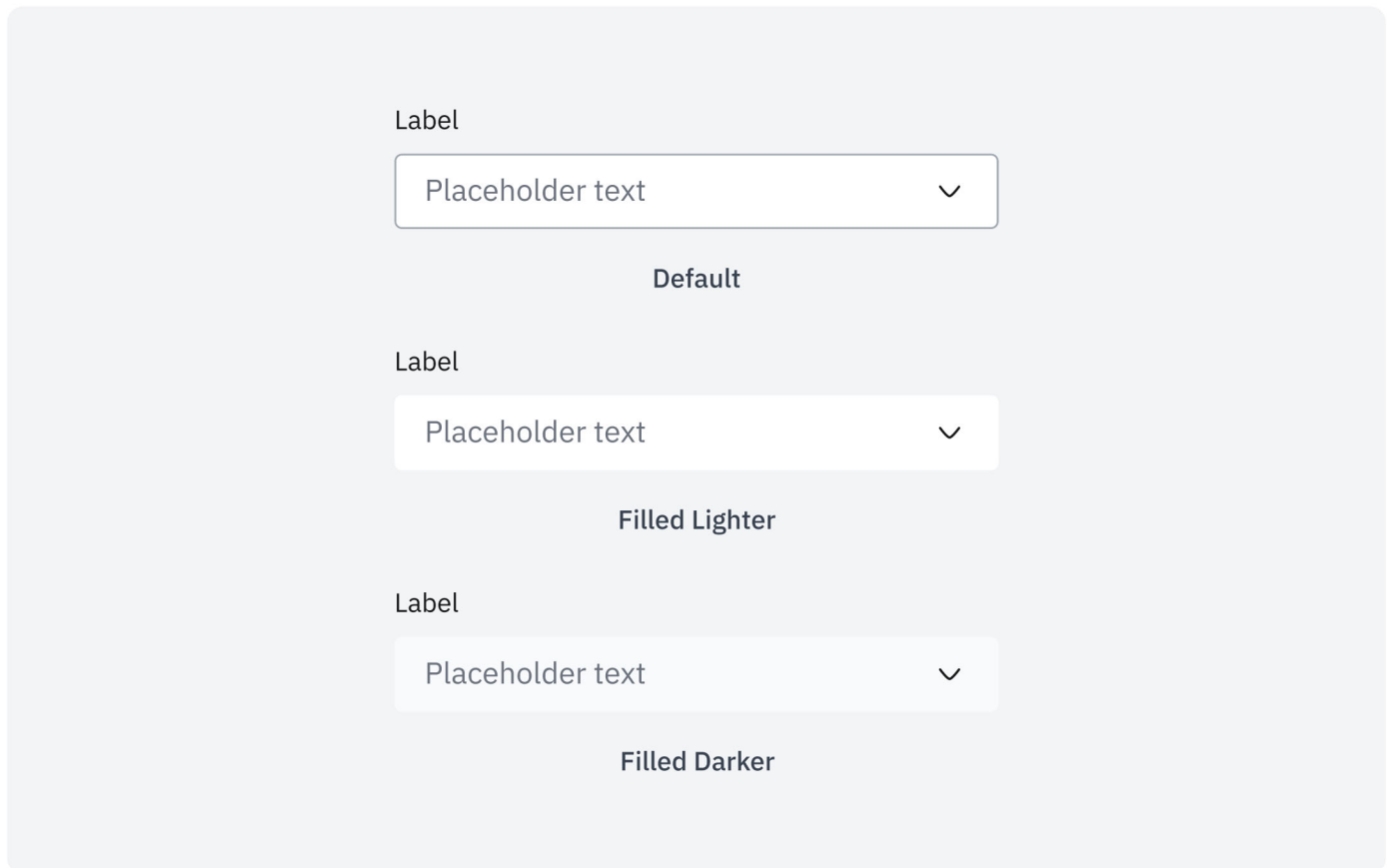
Option

Test the preview of the dropdown comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

Dropdown are available in three common variants:

- **Default:** This is the standard styling of a dropdown component without any special modifications.
- **Filled Lighter:** This variant likely features a lighter background fill for both the dropdown button and the list items.
- **Filled Darker:** The Filled Darker variant uses darker shades for the background of the dropdown button and the list items.



Appearance

The appearance of dropdown components is designed to be visually clear and user-friendly, ensuring that users can quickly identify them as interactive elements and easily navigate the available options. Additionally, an error state is integrated into the design. This state providing clear feedback to correct the issue, which is essential for maintaining a seamless user experience.

The image displays three examples of a dropdown menu component, each with a title 'عنوان' (Title) and a placeholder 'نص تلمیحي' (Placeholder text).

- Example 1:** A standard dropdown menu with a downward arrow icon.
- Example 2:** A dropdown menu with a red border, indicating an error state.
- Example 3:** A dropdown menu with an upward arrow icon, expanded to show a list of options, each labeled 'إختيار' (Select).

Helper Text

- **Helper text:** This is a standard dropdown field accompanied by supporting information. It is designed to guide users to enter accurate data, providing additional details about the field.
- **Error Helper text:** Similar to the standard helper text, this dropdown field is highlighted in red to indicate unsuccessful data entry. It guides users to correct their selection, offering extra information about the error encountered.

Label

Label

Anatomy

Component Anatomy

- 1 Label
- 2 Option
- 3 Dropdown List Item

Label — 1

Option — 2

Option ✓

Option

Option

Option

Option

Option

Option

3

- **Label:** Text that informs the user what to expect in the list of dropdown options.

- **Option:** Each option is a choice available for selection from the dropdown. Options are typically listed vertically within the dropdown menu.
- **Dropdown List Item:** A list of options to choose from, displayed as an open state.

Accessibility

Dropdowns are a fundamental component of many interfaces, facilitating option selection and simplifying workflows. They are especially important in environments where efficiency and ease of use are critical, such as for users with disabilities. Ensuring that dropdowns are accessible is paramount, allowing every user to perceive, understand, operate, and interact with them effectively. Here are key considerations for making dropdowns accessible:

Key Accessibility Attributes and Practices:

ARIA Roles:

role="listbox": Indicates that the element is a listbox that contains selectable options.

role="option": Applied to each option within the dropdown to indicate that they are selectable items.

aria-haspopup="listbox": Used on the trigger to indicate that activating the trigger will open a listbox.

aria-expanded: This attribute on the dropdown trigger indicates whether the dropdown is currently open (true) or closed (false).

aria-selected: Applied to the option that is currently selected.

aria-labelledby or aria-label: Provides an accessible name for the dropdown, which is crucial for users who rely on screen readers.

Keyboard Navigation:

Tab: Should move focus into and out of the dropdown.

Arrow keys: Once the dropdown is focused, pressing the down or up arrow keys should open the dropdown and navigate through the options.

Enter/Space: Selects the currently focused option and closes the dropdown.

Esc: Closes the dropdown if it is open without changing the selection.

Focus Management:

When the dropdown opens, focus should typically be moved to the currently selected option, if any, or the first option otherwise.

Visual Indicators:

Ensure that all focusable elements have a clear visual indicator such as a border highlight, to help users know which element has the focus.

Use both color and text labels to indicate errors (e.g., "Invalid selection" or "Required field").

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Buttons

Link

[← Previous](#)

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Actions](#) > [Link](#)

Link

Links facilitate navigation by allowing users to click (or tap) and navigate users to another location, such as a different site, resource, or section within the same page.

Live Demo



Visual



Code

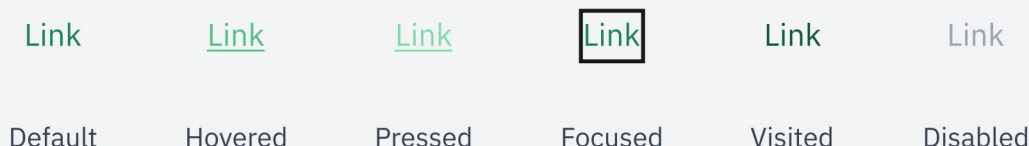
Link

Test the preview of the link comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

Links are available in many fundamental formats:

- **Default:** This is the initial appearance of the link when it is not being interacted with.
- **Hovered:** This state is activated when a user places their cursor over the link.
- **Pressed:** When a link is clicked or tapped but not yet released, it enters the pressed state.
- **Focused:** This state occurs when the link is highlighted through keyboard navigation (using tab keys, for instance).
- **Visited:** After a link has been clicked and leads to a visited page, it may change color to indicate that the destination has already been accessed.
- **Disabled:** A disabled link is one that cannot be interacted with under certain conditions.



Appearance

The design of links in the is crafted to be visually distinct and instantly recognizable, ensuring that users can swiftly identify and interact with the navigational elements presented.

Link Link

With Icons

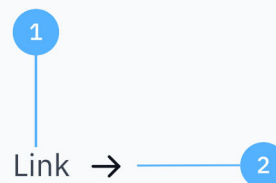
Link Link

Without Icons

Anatomy

Component Anatomy

- 1 Link Text
- 2 Icon (optional)



- **Link Text:** The text is usually descriptive and communicates to the user where the link will lead.
- **Icon (optional):** An icon can be included with the link text to provide additional visual cues or to enhance the aesthetics of the link.

Accessibility

Designing accessible link components is crucial for ensuring that all users, including those with disabilities, can navigate and interact with digital content effectively. Here are key accessibility attributes and practices to consider for link components:

Keyboard Navigation

Tab Navigation: Ensure that all hyperlinks can be focused using the Tab key. This is standard behavior for links but can be disrupted if the tabindex attribute is used incorrectly.

Activation: Users should be able to activate a hyperlink using the Enter key. For links that trigger actions (like buttons), ensure they also respond to the Space key.

Link Text Clarity

Descriptive Text: The text of your hyperlink should clearly describe the purpose or destination of the link. Avoid vague phrases such as "click here" or "go to."

Contextual title Attribute: If the link text is something generic like "read more," use the title attribute to provide additional context.

```
<a href="#" target="_blank" rel="noopener noreferrer" title="Read more about accessibility practices">Read more</a>
```

ARIA Labels: When link text cannot sufficiently describe the destination or action, use aria-label to provide clear guidance.

```
<a href="#" target="_blank" rel="noopener noreferrer" aria-label="Read the accessibility guide PDF">Read more</a>
```

Visual Design

Underlining: Hyperlinks should be underlined by default to ensure they are distinguishable without relying solely on color, which helps users with color blindness.

Color Contrast: Ensure that the color of the hyperlink stands out against the background with sufficient contrast. The WCAG recommends a contrast ratio of at least 4.5:1 for text.

External Links

Security and Performance: For links opening in a new tab, always include rel="noopener" to enhance security and performance.

```
<a href="#" target="_blank" rel="noopener noreferrer">Visit external site</a>
```

Indicating New Tabs: Clearly indicate if a link opens in a new tab. Add visually hidden text for screen readers to inform them of this behavior.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Dropdown

← Previous

Accordion

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Content Display](#) > [Accordion](#)

Accordion

The accordion is used to collapse and expand content to organize it more efficiently. It allows for hiding and displaying content as needed. It reduces clutter and helps maintain the user's focus by showing only the relevant content at a time.

Live Demo



Visual



Code

Accordion



Accordion



Accordion

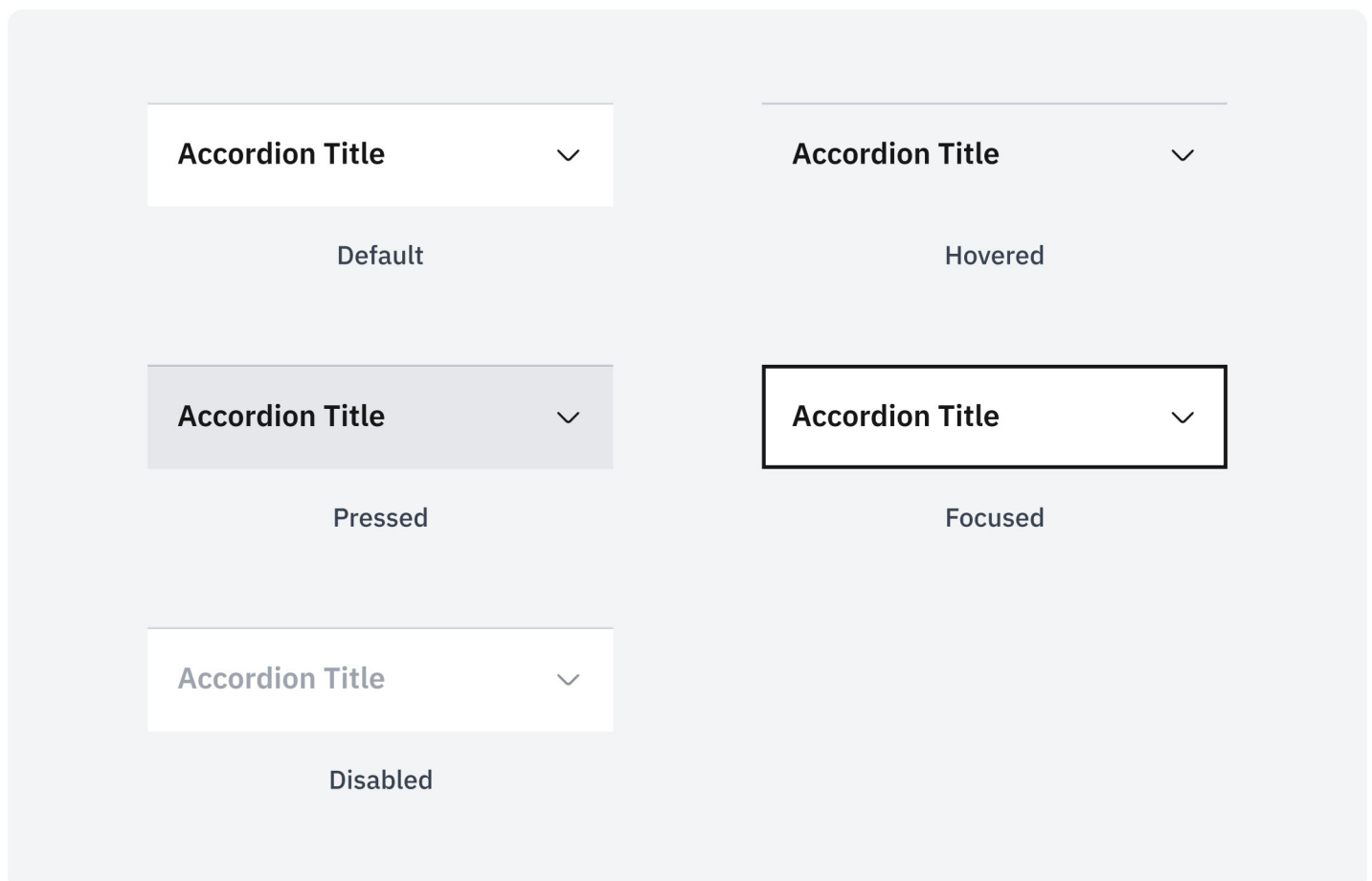


Test the preview of the accordion comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

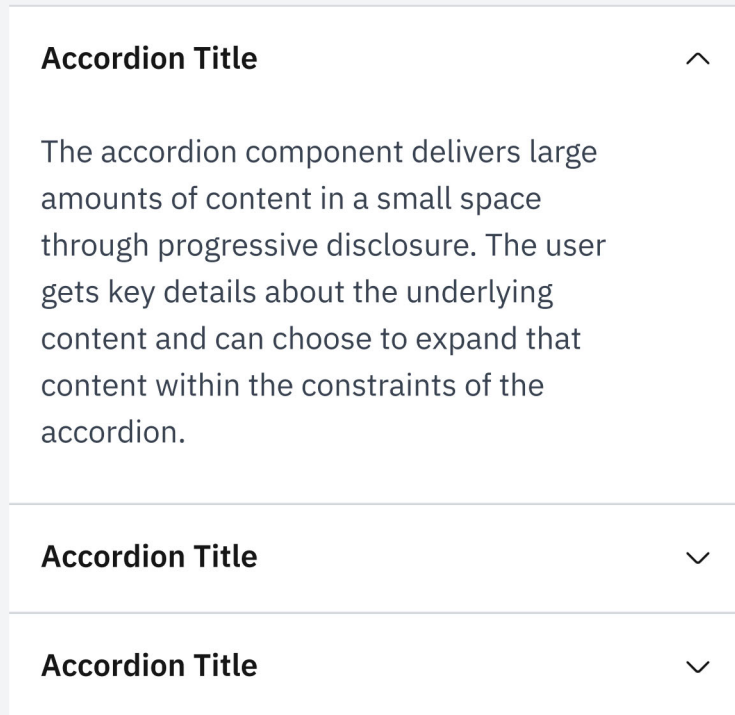
Accordion is available in many fundamental formats:

- **Default:** The default state of the accordion when it is not interacted with. It appears in its initial state without any interaction.
- **Hovered:** It occurs when the user hovers the cursor over the accordion without clicking on it.
- **Pressed:** This state demonstrates how the accordion appears when it is clicked or pressed, indicating interaction.
- **Focused:** This state refers to the accordion when it is selected or active, typically through keyboard navigation, usually via the 'Tab' key.
- **Disabled:** This state shows the accordion when it is not interactive, indicating that the element cannot be interacted with under the current conditions.



Appearance

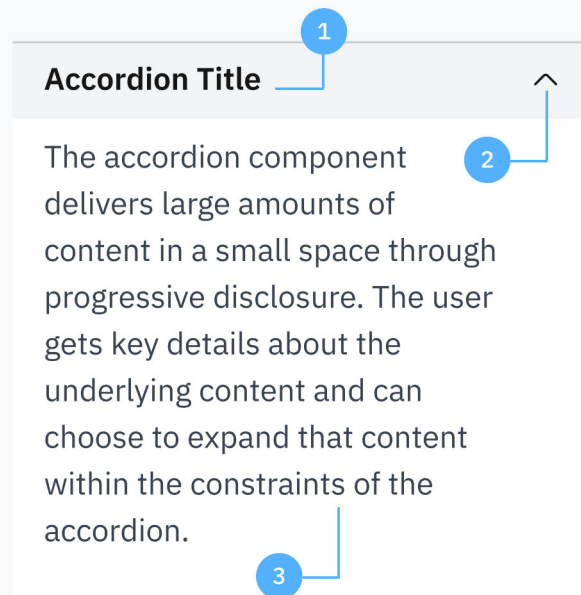
The appearance of the accordion in the is designed to be visually coherent and interactive, ensuring that users can easily navigate and access the information efficiently.



Anatomy

Component Anatomy

- 1 Accordion title
- 2 Expand/Collapse icon
- 3 Panel



- **Accordion title:** Contains the section title and is control for revealing the panel.
- **Expand/Collapse icon:** Indicates if the panel is open or closed.
- **Panel:** The section of content associated with an accordion title.

Accessibility

Accordions, as interactive web components, must be designed with accessibility in mind to ensure they are usable by everyone, including those with disabilities. Here are key accessibility attributes and practices to consider when implementing accordions:

Key Accessibility Attributes and Practices:

ARIA Attributes

Use ARIA (Accessible Rich Internet Applications) attributes to enhance accessibility:

role='button': Applied to each accordion header to indicate that it is interactive.

aria-expanded: This attribute should be set to 'true' or 'false' depending on whether the accordion section is open or closed, respectively.

aria-controls: Connects the accordion header to the content it expands, aiding screen readers in understanding the relationship.

aria-disabled: Used if any accordion header is not interactive.

Keyboard Navigation

Accordions should be fully navigable using keyboard shortcuts. Users should be able to tab to each accordion header and use keys like Enter or Space to expand and collapse sections.

Focus Management

Ensure that when an accordion panel is opened, focus is managed appropriately, often by moving focus to the content inside or keeping it on the header.

Focus indicators (like outlines or borders around the active element) should be clear and distinct to aid those navigating by keyboard.

Screen Reader Support

The content of each accordion header should be clear and informative enough for screen reader users to understand the purpose of the collapsible content.

Visual Design

Ensure that all text within the accordion, including headers and content, has sufficient color contrast ratios as per WCAG (Web Content Accessibility Guidelines) to aid users with visual impairments. And make sure interactive elements like icons or buttons are also easily visible and distinguishable.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Link

Card

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Content Display](#) > [Card](#)

Card

A card serves as a container for information and actions related to a specific concept or object. It typically resembles a rectangular box that combines images, text, and potentially some interactive elements.

Cards enhance the visibility of information and facilitate predictable patterns.

Live Demo



Visual



Code



Card Title

Card content placeholder text goes here

Action

Action

Test the preview of the card comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

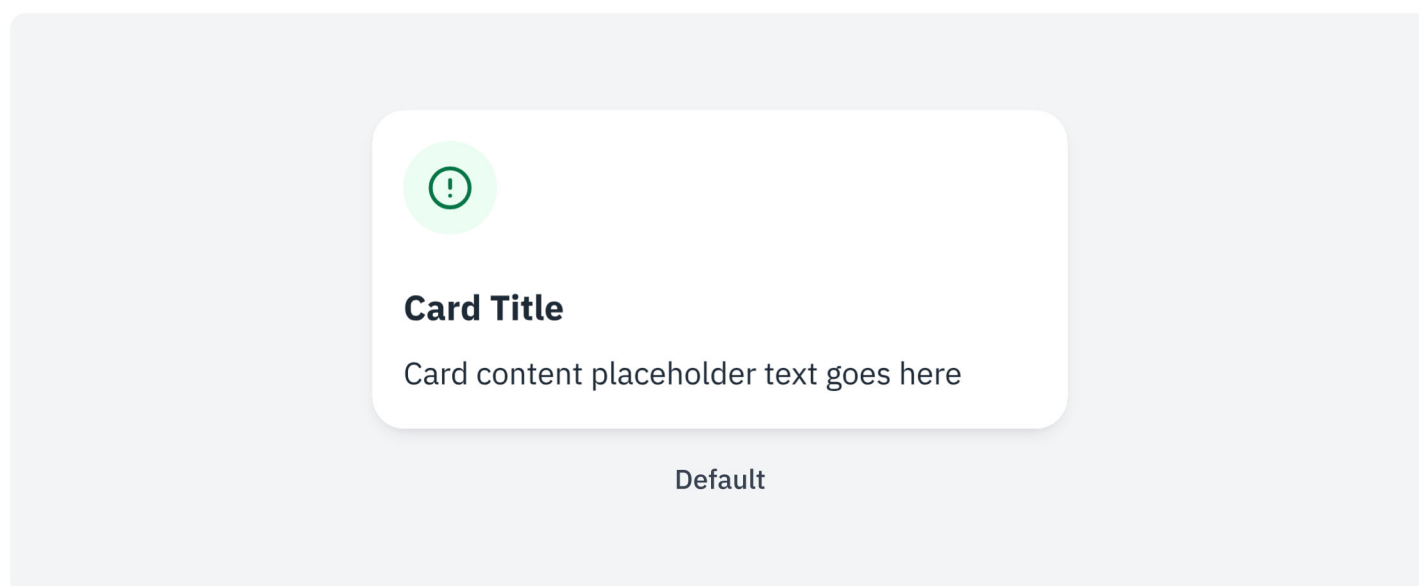
Variants

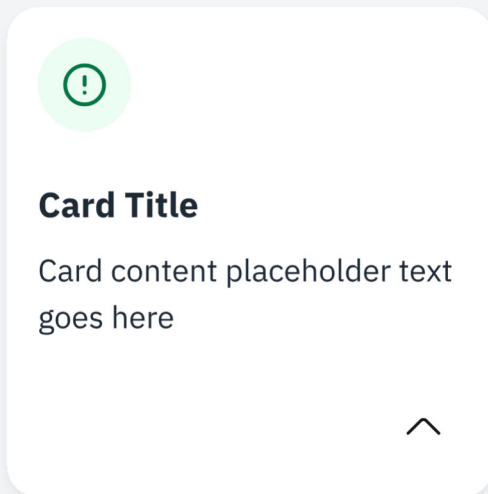
Card is available in three fundamental formats:

- **Default:** Card initial state is the default appearance when no interactions have occurred. It remains unchanged and unaltered until interacted with.
- **Expandable:** Allows users to toggle between a collapsed and expanded view of additional content within the card.
- **Selectable:** Shows that users can interact with or select the content or elements enclosed within the card.

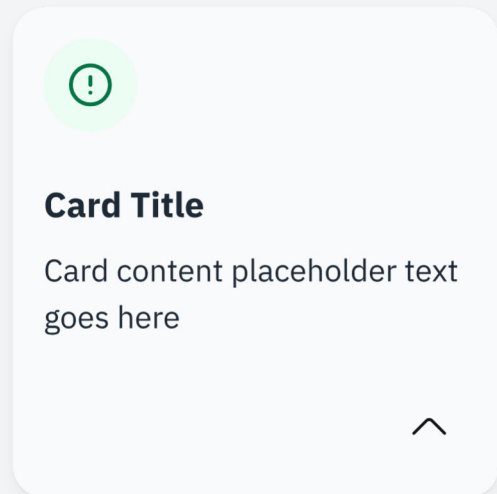
Selectable and **expandable** are available in four states:

- **Default:** Card default state is its initial appearance before any user interaction takes place.
- **Hovered:** It occurs when the user hovers the cursor over the card without clicking on it.
- **Focused:** This state refers to the card when it is selected or active, typically through keyboard navigation, usually via the “Tab” key.
- **Disabled:** This state shows the card when it is not interactive, indicating that the element cannot be interacted with under the current conditions.

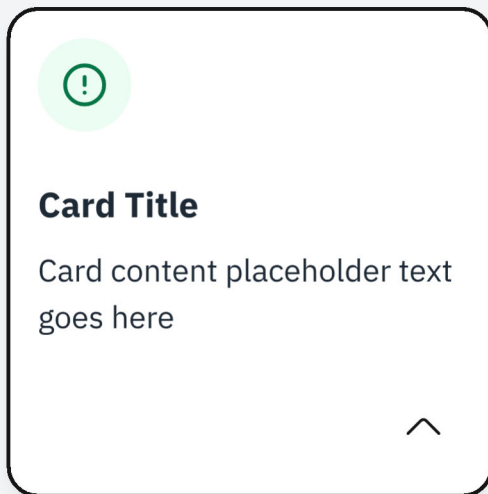




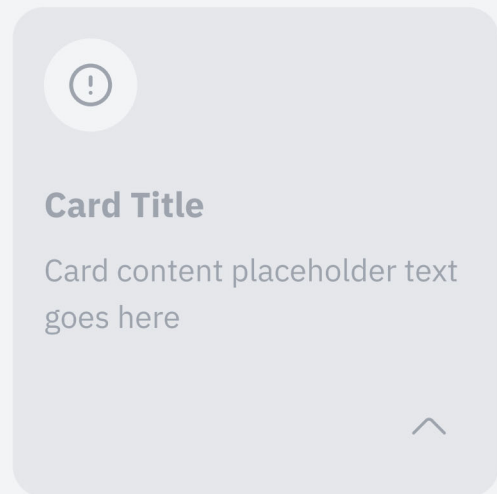
Expanded - Default



Expanded - Hovered



Expanded - Focused



Expanded - Disabled

**Card Title**

Card content placeholder text goes here

Selectable - Default

**Card Title**

Card content placeholder text goes here

Selectable - Hovered

**Card Title**

Card content placeholder text goes here

Selectable - Focused

**Card Title**

Card content placeholder text goes here

Selectable - Disabled

Appearance

The appearance of cards typically includes a rectangular shape with rounded corners, containing a combination of images, text, and possibly interactive elements such as buttons or icons.

Card Title

Card content placeholder text goes here

Without Shadow

Card Title

Card content placeholder text goes here

With Shadow

Card Title

Card content placeholder text goes here

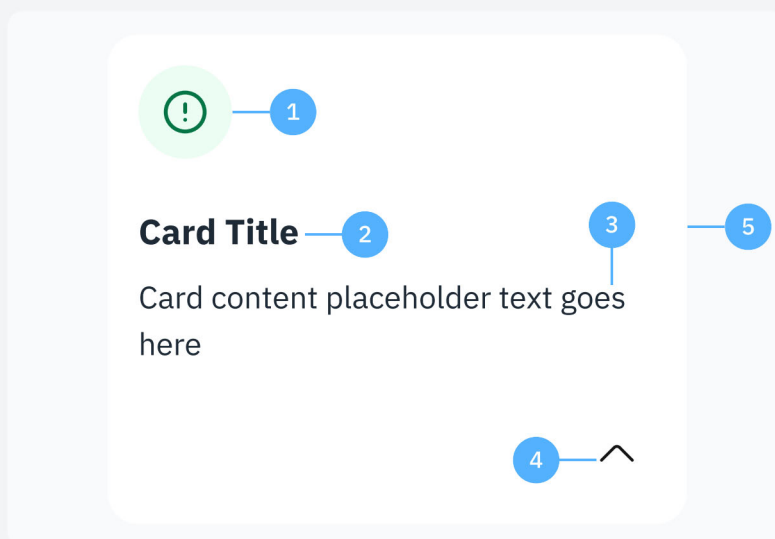
With Stroke

Anatomy

- **Image:** It is a visual element that adds context or enhances the content displayed within the card.
- **Featured Icon:** Visually indicates the type of card.
- **Card Title:** It is the main heading or label that provides a brief description or title for the content within the card.
- **Description:** Provides a text description of the card.
- **Custom Component (e.g., Avatar):** Refers to a specialized UI element integrated within the card layout to provide unique functionality or display tailored content.
- **Tags:** Are small visual elements used to convey specific information or status related to the content.
- **Rating:** Typically represents the user or community's assessment or evaluation of the content displayed within the card.
- **Actions:** Includes interactive elements like ghost or tertiary buttons for user engagement.
- **Container:** The area that holds all the elements.

Expanded Anatomy

- 1 Featured Icon
- 2 Card Title
- 3 Description
- 4 Expand/Collapse icon
- 5 Container

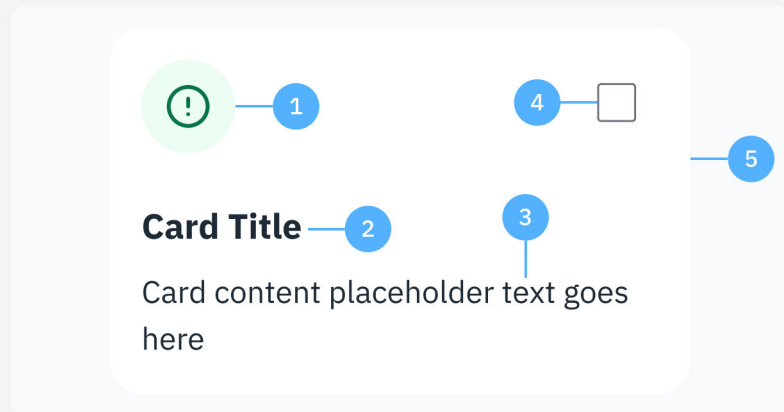


- **Featured Icon:** Visually indicates the type of card.
- **Card Title:** It is the main heading or label that provides a brief description or title for the content within the card.

- **Description:** Provides a text description of the card.
- **Expand/Collapse Icon:** Indicates if the card is open or closed, allowing users to toggle between viewing more or less content within the card.
- **Container:** The area that holds all the elements, encapsulating them within a distinct section or module of the user interface.

Selectable Anatomy

- 1 Featured Icon
- 2 Card Title
- 3 Description
- 4 Checkbox
- 5 Container



- **Featured Icon:** Visually indicates the type of card.
- **Card Title:** It is the main heading or label that provides a brief description or title for the content within the card.
- **Description:** Provides a text description of the card.
- **Checkbox:** Allows users to select or deselect items associated with the card's content.
- **Container:** The area that holds all the elements.

Accessibility

Accordions, as interactive web components, must be designed with accessibility in mind to ensure they are usable by everyone, including those with disabilities. Here are key accessibility attributes and practices to consider when implementing accordions:

Key Accessibility Attributes and Practices:

ARIA Attributes:

Use ARIA (Accessible Rich Internet Applications) attributes to enhance accessibility:

role="button": Applied to each accordion header to indicate that it is interactive.

aria-expanded: This attribute should be set to "true" or "false" depending on whether the accordion section is open or closed, respectively.

aria-controls: Connects the accordion header to the content it expands, aiding screen readers in understanding the relationship.

aria-disabled: Used if any accordion header is not interactive.

Keyboard Navigation:

Accordions should be fully navigable using keyboard shortcuts. Users should be able to tab to each accordion header and use keys like Enter or Space to expand and collapse sections.

Focus Management:

Ensure that when an accordion panel is opened, focus is managed appropriately, often by moving focus to the content inside or keeping it on the header.

Focus indicators (like outlines or borders around the active element) should be clear and distinct to aid those navigating by keyboard.

Screen Reader Support:

The content of each accordion header should be clear and informative enough for screen reader users to understand the purpose of the collapsible content.

Visual Design:

Ensure that all text within the accordion, including headers and content, has sufficient color contrast ratios as per WCAG (Web Content Accessibility Guidelines) to aid users with visual impairments. And make sure interactive elements like icons or buttons are also easily visible and distinguishable.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Accordion

List

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Content Display](#) > [List](#)

List

Lists are used to organize a set of items into a single, cohesive unit, typically displayed as a series of options or links. List items begin with either a number or a bullet.

Live Demo



Visual



Code

- 1. List Item
 - a. List Item
 - b. List Item
 - c. List Item

- List Item
 - List Item
 - List Item
 - List Item

- ☑ List Item
 - ☑ List Item
 - ☑ List Item
 - ☑ List Item

Test the preview of the list comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

Lists are available in two fundamental formats:

- **Ordered List:** An ordered list is used when the sequence of the items is important. It is typically used for items that need to be viewed or processed in a specific order.
- **Unordered List:** An unordered list is used when the order of the items does not matter. It is suitable for listing items without any inherent hierarchy or sequence.
- **List with Icons:** Similar to an unordered list, but each item is preceded by an icon instead of a solid bullet. This style can be visually appealing and can be used to draw attention or to fit with a specific design theme.

1- List item

a- List item

b- List item

c- List item

Ordered List

- List item

• List item

• List item

• List item

Unordered List

☑ List item

☑ List item

☑ List item

☑ List item

List with Icons

Appearance

The design of list in the Platforms code is meticulously crafted to ensure they are visually cohesive and easily navigable, providing a clear and organized way to present multiple items as a unified component.

- 1- List item
 - a- List item
 - b- List item
 - c- List item

- 1- List item
 - a- List item
 - b- List item
 - c- List item

- 1- List item
 - a- List item
 - b- List item
 - c- List item

- List item
 - List item
 - List item
 - List item

- List item
 - List item
 - List item
 - List item

- List item
 - List item
 - List item
 - List item

- ☑ List item
 - ☑ List item
 - ☑ List item
 - ☑ List item

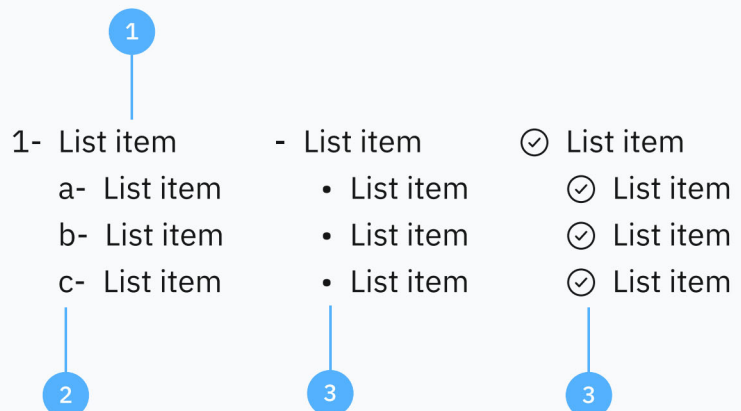
- ☑ List item
 - ☑ List item
 - ☑ List item
 - ☑ List item

- ☑ List item
 - ☑ List item
 - ☑ List item
 - ☑ List item

Anatomy

Component Anatomy

- 1 List Item
- 2 Numbers/Letters in Ordered Lists
- 3 Bullets / Icons in Unordered Lists



- **List Item:** Represent individual selectable options or actions.
- **Numbers in Ordered Lists:** Numbers in ordered lists serve to indicate sequence and hierarchy.
- **Bullets / Icons in Unordered Lists:** Bullets or icons are used in unordered lists where the arrangement of items does not follow any specific sequence.

Accessibility

Designing accessible link components is crucial for ensuring that all users, including those with disabilities, can navigate and interact with digital content effectively. Here are key accessibility attributes and practices to consider for link components:

Key Accessibility Attributes and Practices:

ARIA Roles

ARIA-labelledby: This attribute can be used to associate the list group with a label or header, helping users understand the context of the list items.

ARIA-current: For lists used in navigation, the `aria-current='page'` attribute can be used to indicate which list item corresponds to the current page, helping users orient themselves within the navigation.

Keyboard Accessibility

Ensure all interactive elements in the list group can be accessed and navigated using a keyboard alone. This typically means using Tab and Shift+Tab to move through items, with additional keyboard controls as necessary.

It's important to provide a clear visual indication of which list item has focus when navigated via keyboard to help users with limited mobility or visual impairments.

Clear Labels

All interactive elements should have clear, concise labels. If icons or buttons are used within the list, they should have text alternatives (e.g., using `aria-label` attributes) to describe their action or purpose.

Responsive and Adaptive Design

Ensure list groups are usable on mobile devices, not only in terms of visual layout but also in terms of interaction. Larger touch targets and spacing can help users who have difficulty with precise gestures.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Card

Digital Signature

[← Previous](#)

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Content Display](#) > [Digital Signature](#)

Digital Signature

The Digital Signature Component is an interface element that ensures users are interacting with a verified and secure source, confirming the integrity and authenticity of government sites. It provides an encrypted digital stamp, reinforcing trust and transparency in the website's content.

Live Demo



Visual



Code



An Official website of the Kingdom of Saudi Arabia [How you know?](#) 櫛

Experience a preview of the digital signature component's functionality and styles in our live demo section. For a comprehensive exploration, including all variants and features, please visit our complete demo on [Storybook](#).

Variants

Digital Signature is available in two common variants:

- **Closed:** It's used as the default view, It provides users with a quick assurance of authenticity without taking up much space.
- **Opened:** It's Activated when a user clicks on "How you know?" It provides comprehensive details for users who want to verify the website's authenticity more thoroughly.

Appearance

The Digital Signature Component ensures authenticity and security, assisting users in quickly verifying that a website is an official government or affiliated organization platform. This component is designed with a clean and informative layout, integrating seamlessly into various interfaces to provide clear indications of website authenticity. Key details, such as domain confirmation (e.g., .gov.sa, .edu.sa, .med.sa, .org.sa, .sch.sa), HTTPS usage, and Digital Government Authority registration, reinforce user trust and security. By ensuring a consistent and transparent experience across different contexts, this component reassures users that they are interacting with a secure and verified site, helping them identify the specific type of organization behind the platform.

Digital Signature for different websites extensions:

- **gov.sa:** Ministries, authorities, public institutions, councils, national centers, and equivalent entities.
- **edu.sa:** Educational and training institutions that provide direct services to students and trainees, such as public universities, colleges, and government training institutes.
- **org.sa:** Non-profit entities, including related activities such as conferences, committees, and organizations affiliated with government entities.
- **sch.sa:** Government entities classified as public educational institutions for secondary education and below.

- **med.sa:** Government entities affiliated with the Ministry of Health and hospitals under other government agencies, such as hospitals affiliated with the Ministry of Defense. This includes, for example, facilities classified as hospitals, health centers, and medical cities.
- **.sa:** Designated for the activities, initiatives, and programs of government entities in the Kingdom.

Anatomy

- **Government Website Indicator:** A label that clearly indicates the website belongs to an official government organization.
- **How You Know? Link:** A clickable link, that expands to provide additional information about the authenticity indicators.
- **Domain Verification Icon and Message:** It is used to visually confirm that the website's domain name is officially government-approved, enhancing trust.
- **Security Confirmation Icon and Message:** It is used to reassure users that the website's data transmission is encrypted, protecting their information.
- **Digital Government Authority Registration:** This registration provides an extra layer of verification by referencing a trusted digital government entity, strengthening the website's credibility. The registration number is linked to the government entity's License Number, ensuring authenticity and compliance.

Accessibility

Digital Signature Components are a crucial feature in many government websites, providing users with a clear and trustworthy way to verify the authenticity of the site. They are essential in environments where confirming website authenticity is critical. Ensuring that Digital Signature Components are accessible is important, allowing every user to perceive, understand, and rely on them effectively. Here are key considerations for making Digital Signature Components accessible:

Key Accessibility Attributes and Practices:

ARIA Roles:

Use ***role="status"*** to signal that the Digital Signature Component contains important status information, especially for users with screen readers. This makes the component's information more accessible without being disruptive.

Use ***aria-label*** or ***aria-describedby*** attributes on non-text elements, like icons, to add context for screen reader users. This ensures that icons denoting security or authenticity are meaningful to visually impaired users.

Keyboard Navigation:

Ensure that the component is fully accessible via keyboard navigation. All interactive elements, such as "How you know?" links or expandable sections, should be focusable and operable with the keyboard.

Provide a visible focus indicator on all elements to help users who rely on keyboard navigation know where they are on the page.

Decorative Icons:

Set ***aria-hidden="true"*** for decorative icons that don't add essential information to avoid redundant announcements for screen reader users.

Responsive Design:

The layout and text should adjust appropriately for different screen sizes, including mobile devices, ensuring that all content remains accessible and readable.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

List

Avatar

[← Previous](#)

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Data Display](#) > [Avatar](#)

Avatar

The avatar is a visual representation of a user, commonly displayed in profiles, comments, or messaging apps, enhancing user engagement and recognition.

Live Demo



Visual



Code



Test the preview of the avatar comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

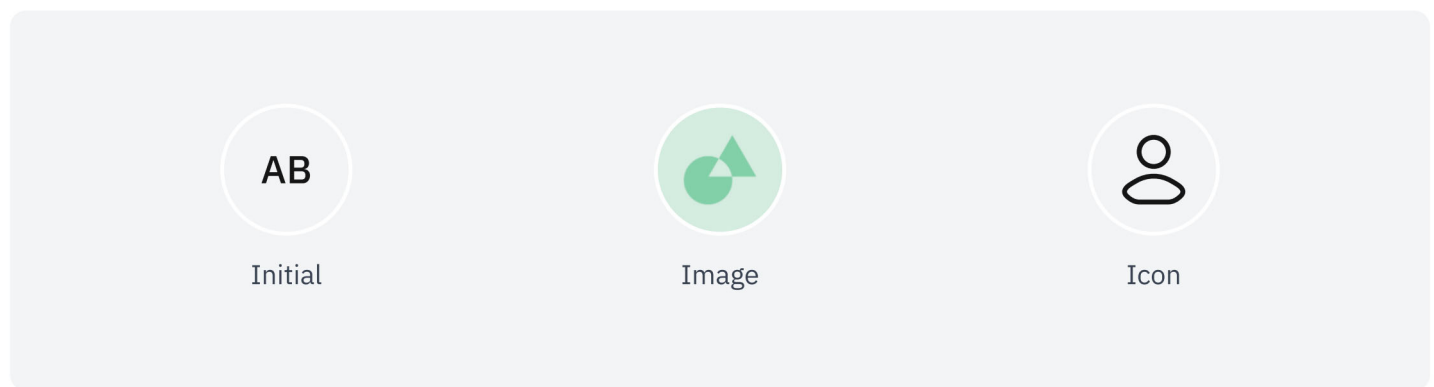
Appearance

The Platforms code Design System presents a diverse range of avatar variations, customized to meet varied design needs.

Types

Avatars are available in three types:

- **Initial:** Displays the initials of the username or user name.
- **Image:** Displays the user's image, whether it's pre-uploaded from the system or uploaded by the user.
- **Icon:** Displays an icon representing the user, used when the system doesn't allow the user to change the avatar or when the user hasn't yet changed their avatar.

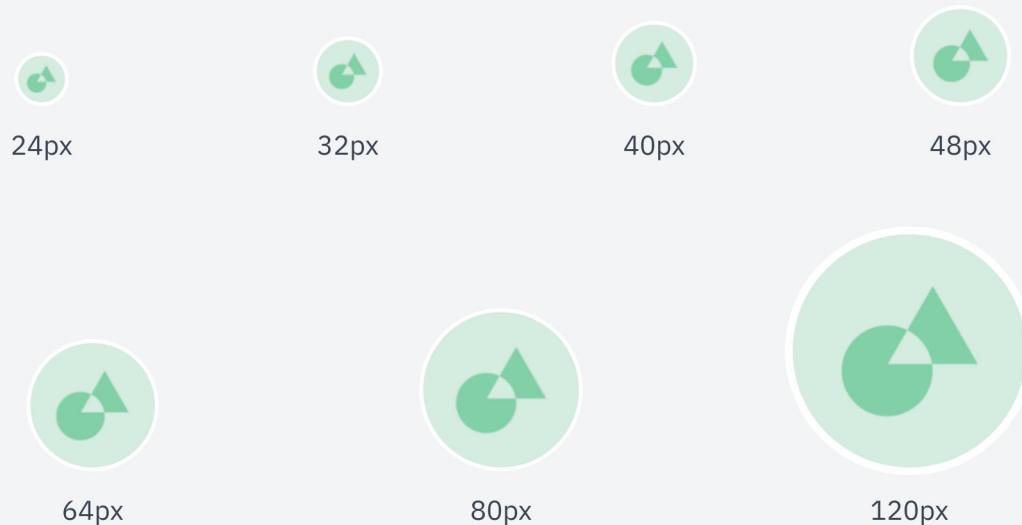


Avatar size

Avatars in the Platforms code design system come in various sizes, depending on the container size:

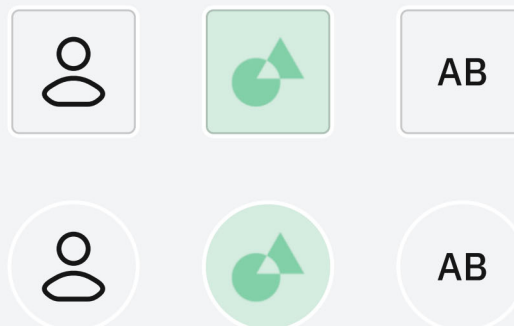
- **24px and 32px (Small):** Suitable for small spaces, such as in components like table cell or small components.
- **40px and 48px (Medium):** The standard size used in most interface components such as profile panels or cards.
- **64px, 80px and 120px (Large):** Used sparingly to highlight user or in spacious areas. When using large avatars, ensure they maintain balance with other components in the user interface.

Consider the size of the icon and surrounding elements when selecting the appropriate size to ensure visual harmony and balance.



Style

Avatars in the Platforms code are available in two distinct forms: the standard circular shape and a rectangular shape, offering increased versatility to meet different design contexts and user interface needs.



Avatar groups

In Platforms code, individual avatars can be combined to symbolize multiple users within a single element. Avatar groups come in two forms: the stacked version, where avatars are positioned one on top of the other without gaps, and the unstacked version, where there is space between each avatar.



Stacked avatar group

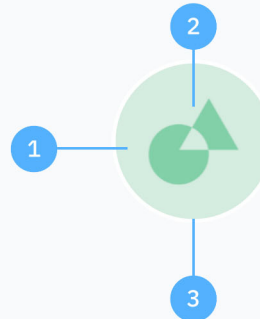


Unstacked avatar group

Anatomy

Component Anatomy

- 1 Container
- 2 Content (Initials, icon or image)
- 3 Boarder



- **Container:** The area that holds all the elements.
- **Content:** Displays avatar's core elements, whether it's user initials, image, or icon.
- **Boarder:** Surrounds the avatar, adding a visual distinction or boundary.

Accessibility

Avatars serve as visual representations of users, thus accessibility is crucial to ensure that everyone can understand and interact with these components. Following accessibility best practices helps create a more inclusive and user-friendly experience.

- Always include ***alt=""*** as an attribute for the avatar's image tag and specify the user's name within it. The alt text should describe the purpose of the avatar, enabling users with visual impairments to grasp its significance.
- While not mandatory, adding ***role="img"*** to the container that holds the avatar can be beneficial. This attribute signals that the container functions as an image, providing clearer semantic meaning to assistive technologies.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Digital Signature

Table

[← Previous](#)

[Up Next →](#)



Table

A table is a structured format for displaying data in rows and columns, which helps users to understand and interact with complex information efficiently. It is a crucial element in many user interfaces, particularly for applications that require the organization, analysis, or manipulation of large amounts of data.

Live Demo



Visual



Code

☐	Header	Header	Header	Header
☐	Link	Cell	Cell	Tag
☐	Link	Cell	Cell	Tag
☐	Link	Cell	Cell	Tag
☐	Link	Cell	Cell	Tag

Test the preview of the table comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

Tables variants are available for row cells and column cells, To enhance the flexibility and usability of the tables:

Table Row Cell

- **Default:** The basic cell container where content or data is placed.
- **Link:** A cell that functions as a hyperlink, presumably taking the user to another page or causing some other action.
- **Entered Text:** A cell variant where the user can input text, indicating an editable field.
- **Tag:** This likely refers to a cell that contains a tag or label, which could be used to represent status, categorization, or other properties.
- **Status:** This appears to show different status indicators that might reflect the state of the content within that cell (e.g., active, inactive, pending).
- **Switch:** A toggle switch, which users can interact with to change a setting directly from the table cell.
- **Feedback Icon:** This is an icon that indicates some type of feedback, status, or action related to the content or interaction within the cell.
- **Actions:** This likely refers to interactive elements like buttons or icons within a cell that allow the user to perform actions directly from the table, such as editing or deleting an entry.
- **Avatar:** An avatar typically represents a user. In the context of a table, this could be a profile picture or a graphical placeholder to represent an individual associated with the row.
- **Checkbox:** A checkbox within a table cell is used for selection purposes, allowing the user to select one or more rows for actions.
- **Custom:** This might indicate a cell designed for a specific, perhaps more complex interaction or display that doesn't fit into the standard types of cell components.

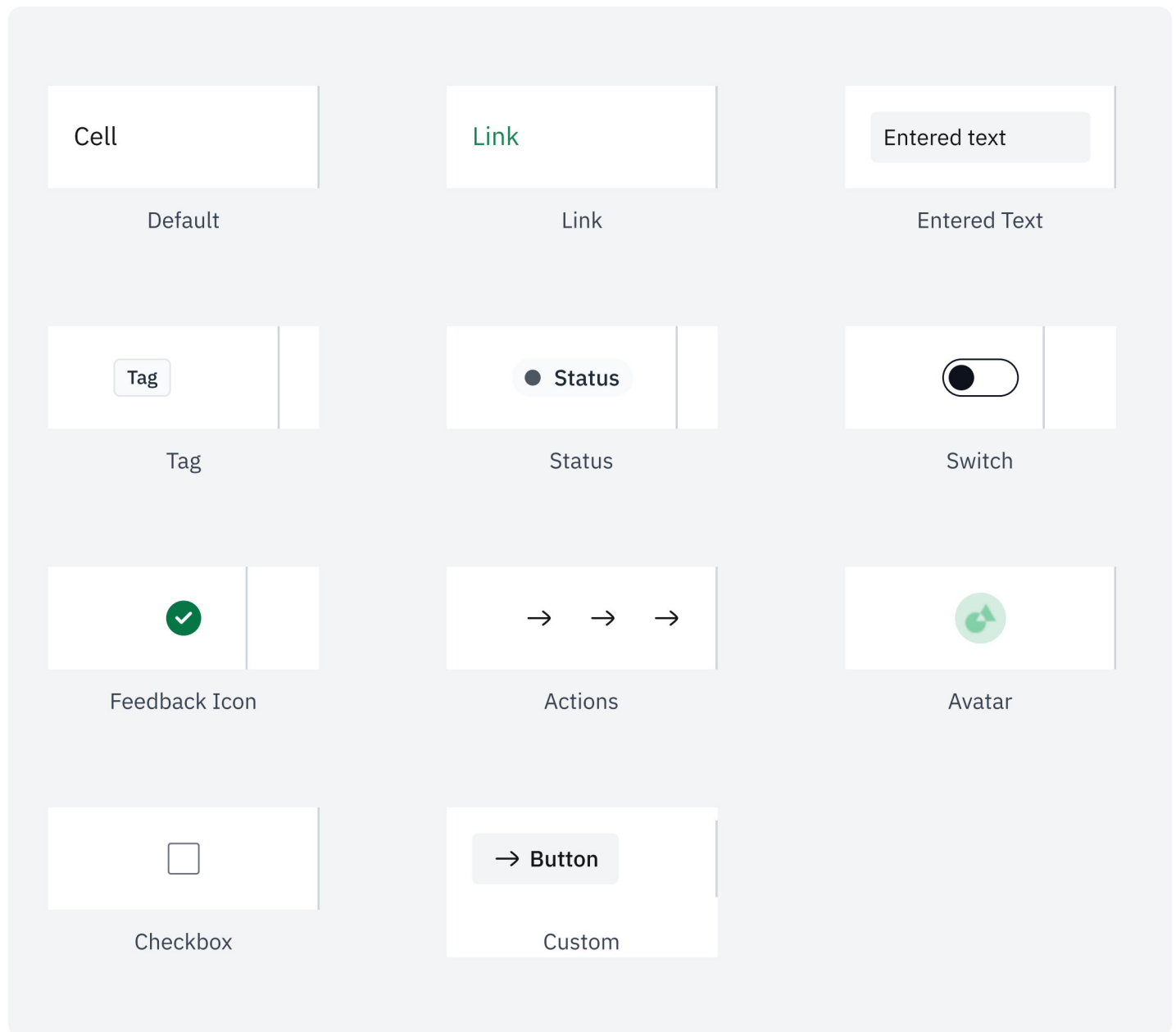
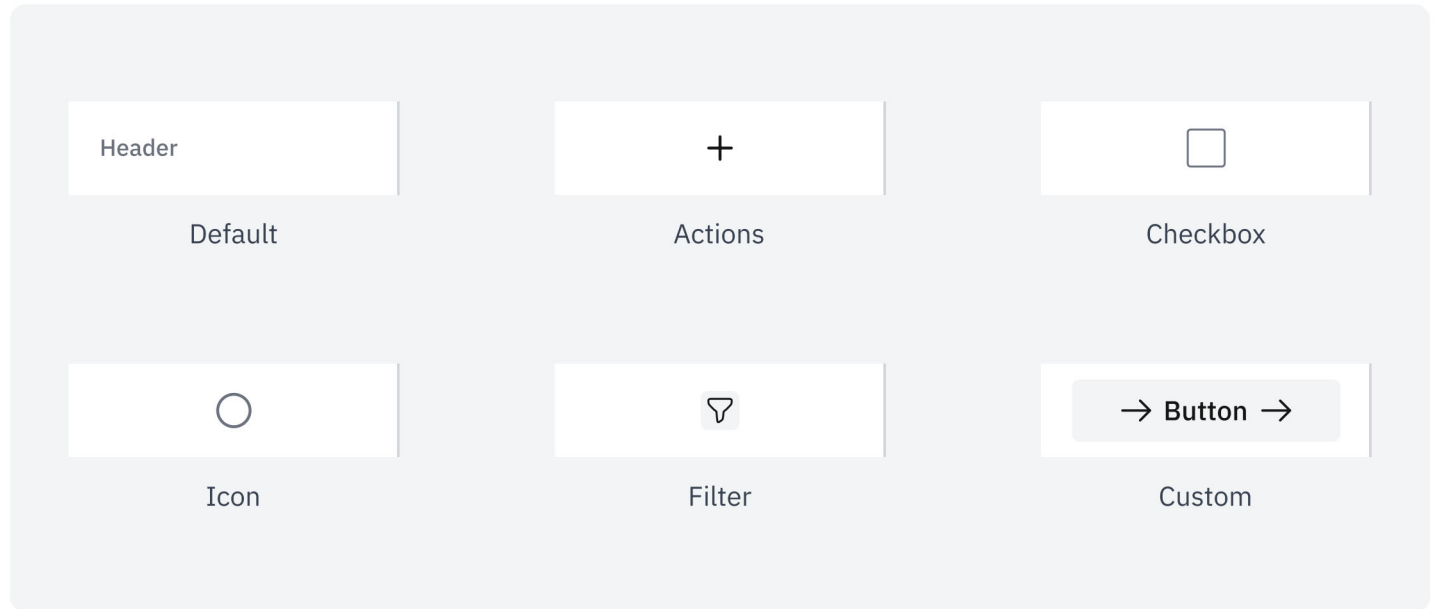


Table Header Cell

- **Default:** This is the standard header cell, which usually contains the name or title of the column below it.
- **Actions:** That might contain buttons or other interactive elements, allowing users to perform actions that affect the entire column or table, such as adding a new row or exporting data.
- **Checkbox:** A header cell with a checkbox typically allows users to select or deselect all rows in the table, often used for bulk actions.
- **Icon:** Including an icon in a header cell can help to visually represent what the column is about or indicate that a column is sortable or contains a certain type of data.
- **Filter:** This variant usually includes a filter icon or mechanism directly in the header cell, indicating that users can filter the rows in the table based on the data in this column.

- **Custom:** The custom header cell allows for a specific design that doesn't conform to the standard header cell types. This could include unique interactive elements or data representations relevant to the table's functionality.



Appearance

The appearance of tables is designed to be visually coherent and easily navigable, ensuring that users can quickly recognize and understand the data being displayed.

☐	Header	Header	Header	Header	
☐	Link	Cell	Tag	☒ Status	→
☐	Link	Cell	Tag	☐ Status	→
☐	Link	Cell	Tag	☒ Status	→
☐	Link	Cell	Tag	☐ Status	→
☐	Link	Cell	Tag	☒ Status	→
☐	Link	Cell	Tag	☐ Status	→

Anatomy

Component Anatomy

- 1 Table header
- 2 Table row
- 3 Table column
- 4 Header Icon

Header	Header	Header
Link	Cell	Cell
Link	Cell	Cell
Link	Cell	Cell

- **Table header:** Contains header cells that label the columns. This row is often visually distinct from the rest of the table to denote its role as a reference point for the data in the table.
- **Table row:** Is a horizontal grouping of cells within a table. Each row represents a single record or unit of information, and it contains multiple cells that can hold various types of data.
- **Table column:** Is a vertical grouping of cells within a table. Each column is usually associated with a specific type of information or data field represented across multiple rows.
- **Header Icon:** These icons enhance the user experience by providing visual cues and interactive elements that help users interact with.

Accessibility

Creating accessible tables is crucial for ensuring that all users, including those who rely on assistive technologies, can access and understand data presented in tabular format.

Keyboard Accessibility

Ensure that all interactive elements in the table (like buttons or links) are accessible via keyboard.

Screen Reader Support

Avoid using complex layouts that are difficult for screen readers to interpret.

Use ARIA roles and properties if necessary to enhance semantic meaning, but only as a last resort when standard HTML elements are insufficient.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Avatar

Content Switcher

[← Previous](#)

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Data Display](#) > [Content Switcher](#)

Content Switcher

Content switchers in the platforms code allow users to navigate between related views efficiently. They display one section at a time, ensuring a focused and user-friendly experience.

Live Demo



Visual



Code

Item

Item

Item

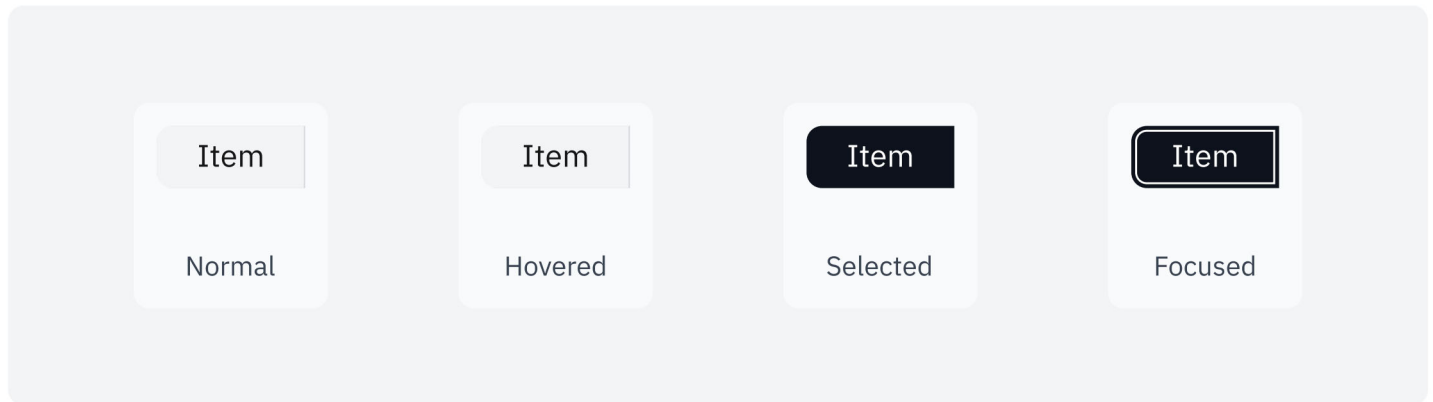
Item

Test the preview of the content switcher comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

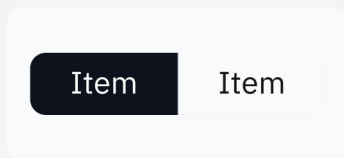
Content switcher is available in four fundamental formats:

- **Normal:** This is the standard appearance of the Content switcher.
- **Hovered:** Occurs when the user moves the cursor over the content switcher without clicking.
- **Selected:** Indicates that the Content switcher is currently selected or activated.
- **Focused:** Occurs when a content switcher receives keyboard focus, usually through the 'Tab' key. Important for accessibility to ensure users can navigate through the interface using the keyboard.

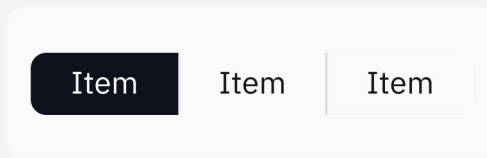


Appearance

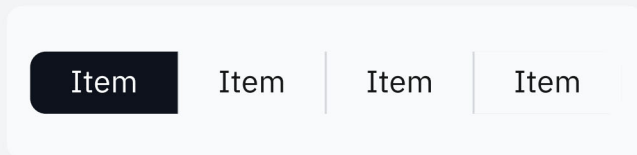
The appearance of a content switcher typically includes a set of options or tabs arranged horizontally. Each option represents a different piece of content or view that users can switch between. The option limits for content switchers range from two to four. Having less than two options would render a content switcher useless, while having more than four could clutter and complicate the interface. Tabs are a better choice for managing larger amounts of options, as they provide a more organized and clear method to display multiple categories or sections



Two options



Three options



Four options

Anatomy

Component Anatomy

- 1 Container
- 2 Text



- **Container:** The clickable area that holds all the elements.
- **Text:** Displays the title of the Content switcher.

Accessibility

Ensuring accessibility in content switchers is crucial so that all users, including those using assistive technologies, can easily navigate and interact with them. Here's a detailed accessibility guide for creating accessible content switcher components:

- Utilize **role="tablist"**, **role="tab"**, and **role="tabpanel"** to define the structure of the content switcher.
- Ensure each tab is associated with its respective content panel using **aria-controls** and indicate the active tab using **aria-selected="true"**

```
<div role="tablist" aria-label="Sample Tabs">
  <button role="tab" aria-selected="true" aria-controls="panel1" id="tab1">Tab 1</button>
  <button role="tab" aria-controls="panel2" id="tab2">Tab 2</button>
</div>
<div id="panel1" role="tabpanel" aria-labelledby="tab1">Content 1</div>
<div id="panel2" role="tabpanel" aria-labelledby="tab2" hidden>Content 2</div>
```
- Use **aria-label** or **aria-labelledby** to provide accessible names for the content switcher and its parts.
- Ensure that tabs can be navigated using arrow keys according to the ARIA Authoring Practices (left/right for horizontal tabs, up/down for vertical tabs).
- The **Tab** key should move focus between the tab list and other controls, while arrow keys change the selection within the tab list.
- Use **aria-hidden="true"** to hide inactive panels from assistive technologies.
- Dynamically toggle **aria-hidden** and **hidden** attributes as tabs are activated or deactivated.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Table

[← Previous](#)

Notification

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Feedback](#) > [Notification](#)

Notification

Notifications are essential tools for conveying information to users about updates or changes within the system. They are categorized into inline, toast, and actionable notifications, each serving distinct purposes within user interactions.

Live Demo



Visual



Code



Notification/Alert message title



When a Notification/Alert needs a further detailed explanation, it goes here.

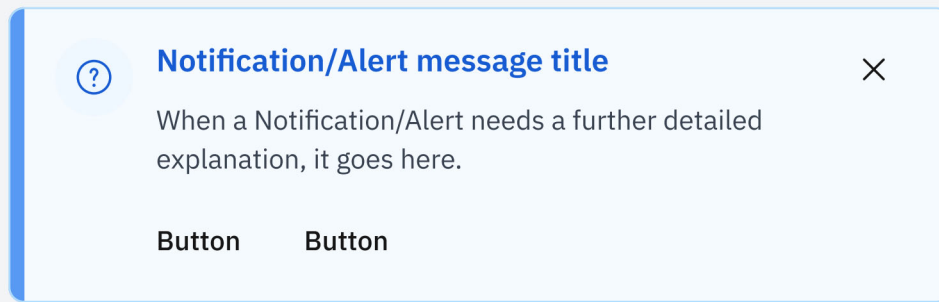
Button Button

Test the preview of the notification comprehensive exploration, including all variants and features, our full demo will soon be available on **Storybook**.

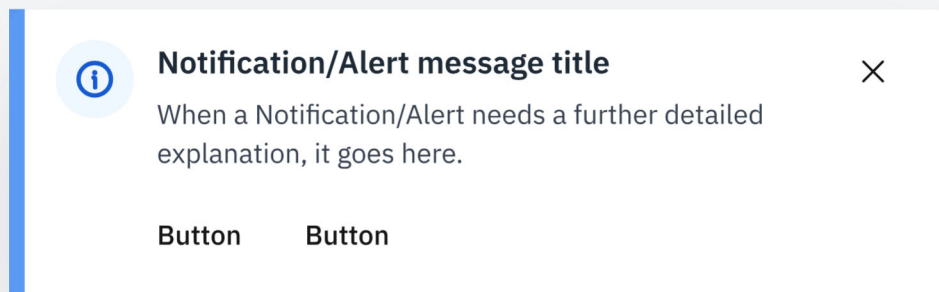
Variants

Notifications are available in three fundamental formats:

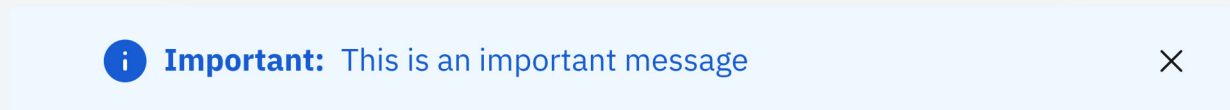
- **Inline Notifications:** Positioned within task flows, these notifications inform users about the status of an ongoing action, typically displayed at the top of the main content area.
- **Toast Notifications:** These are transient, non-modal messages that appear at the top of the screen to provide brief alerts, disappearing after a set duration.
- **Notifications:** Notifications are prominent UI elements used to convey important information or alerts to users. They typically appear at the top of a webpage or application screen, drawing immediate attention to their content.



Inline Notifications



Toast Notifications

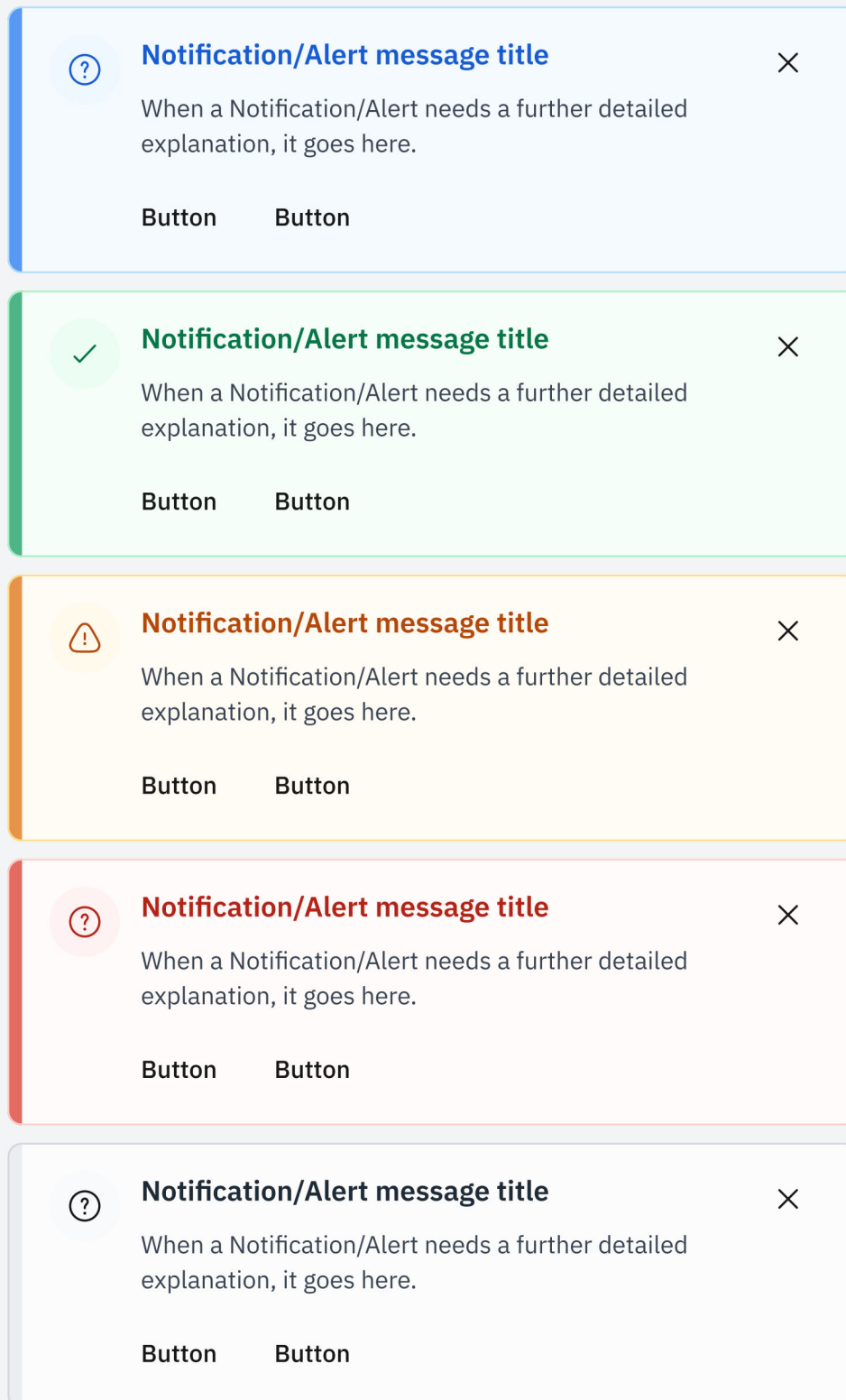


Notifications

Appearance

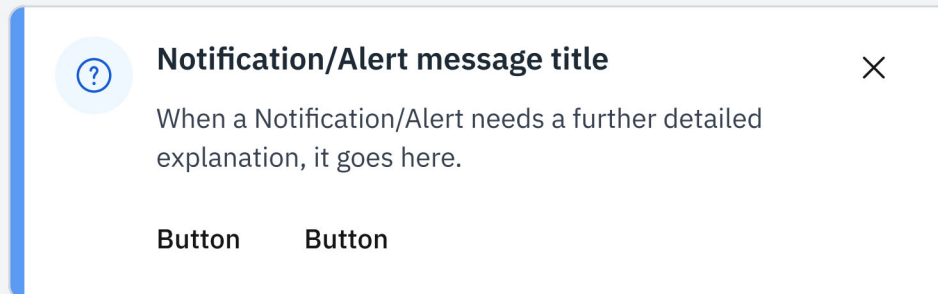
The appearance of notifications is designed to be visually intuitive and easily distinguishable, ensuring that users can quickly recognize and understand the information being presented

Inline Notifications

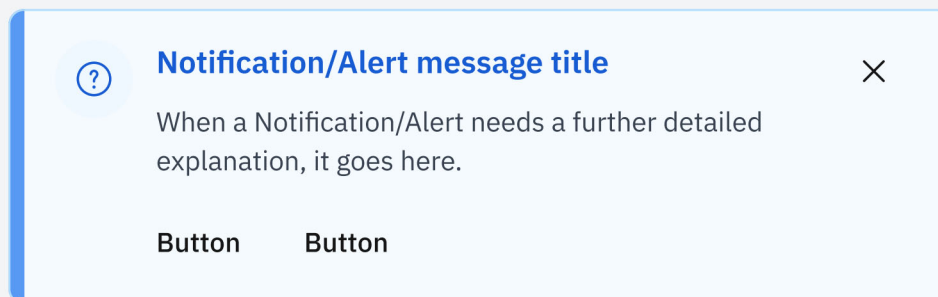


- **Info Inline Notifications:** Offers supplementary information or system updates.
- **Success Inline Notifications:** Indicates successful completion of an action or process.
- **Warning Inline Notifications:** notify users to a problem or challenge requiring action.

- **Critical/Error Inline Notifications:** Indicates errors or failures in the process that need correction.
- **Neutral Inline Notifications:** Conveys general information or acknowledgments without invoking any particular urgency or action.



Without background color



With background color

- **Without background color.**
- **With background color.**

Toast Notifications

**Notification/Alert message title**

When a Notification/Alert needs a further detailed explanation, it goes here.

Button**Button****Notification/Alert message title**

When a Notification/Alert needs a further detailed explanation, it goes here.

Button**Button****Notification/Alert message title**

When a Notification/Alert needs a further detailed explanation, it goes here.

Button**Button****Notification/Alert message title**

When a Notification/Alert needs a further detailed explanation, it goes here.

Button**Button****Notification/Alert message title**

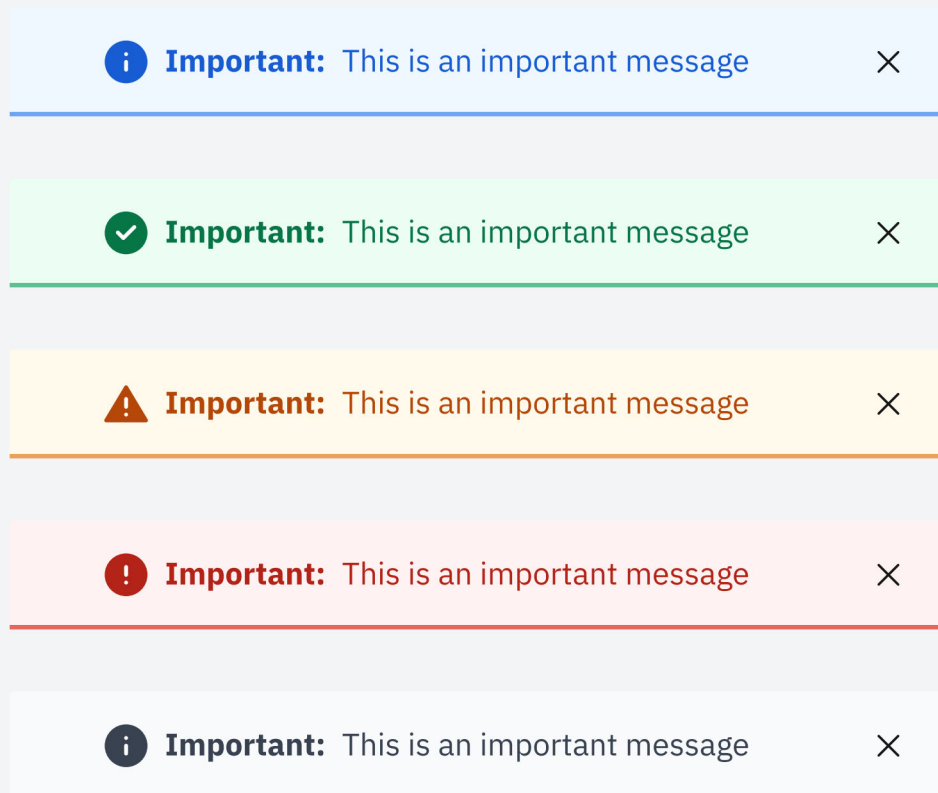
When a Notification/Alert needs a further detailed explanation, it goes here.

Button**Button**

- **Info Toast Notifications:** Offers supplementary information or system updates.

- **Success Toast Notifications:** Indicates successful completion of an action or process.
- **Warning Toast Notifications:** Alerts users to a problem or challenge requiring action.
- **Critical/Error Toast Notifications:** Indicates errors or failures in the process that need correction.
- **Neutral Toast Notifications:** Conveys general information or acknowledgments without invoking any particular urgency or action.

Notification

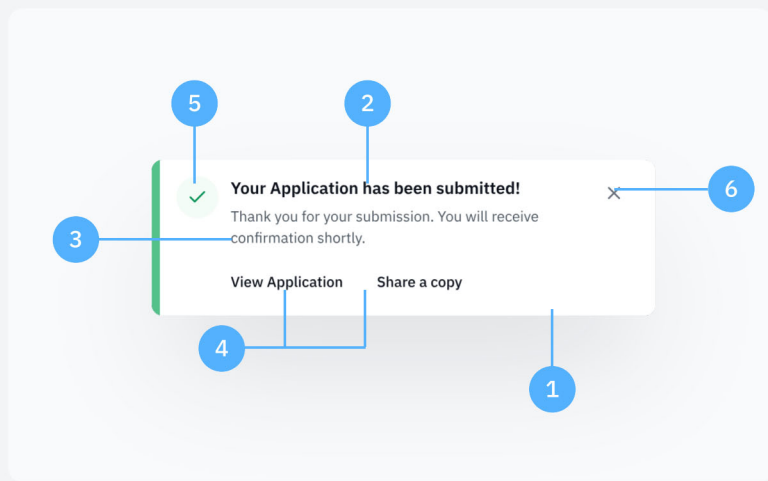


- **Info Notification:** Offers supplementary information or system updates.
- **Success Notification:** Indicates successful completion of an action or process.
- **Warning Notification:** Alerts users to a problem or challenge requiring action.
- **Critical/Error Notification:** Indicates errors or failures in the process that need correction.
- **Neutral Notification:** Conveys general information or acknowledgments without invoking any particular urgency or action.

Anatomy

Component Anatomy

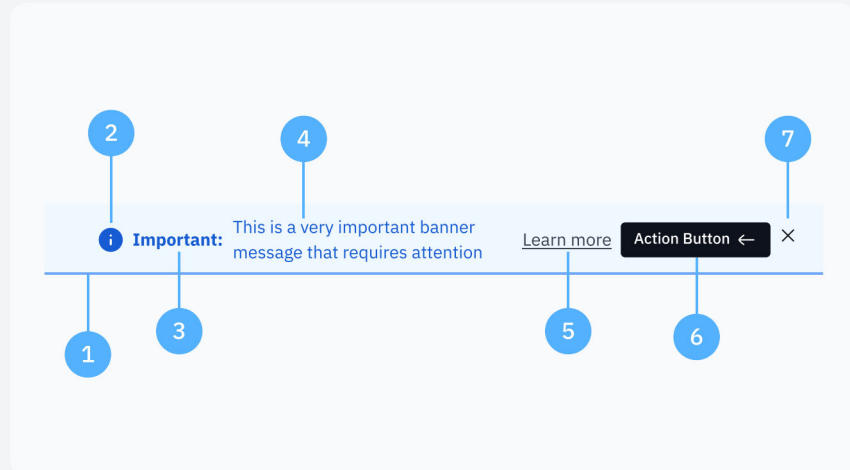
- 1 Container
- 2 Title
- 3 Helper Text
- 4 Actions
- 5 Featured Icon
- 6 Closing Button



- **Container:** The area that holds all the elements.
- **Title:** Provides a concise summary of the banner content.
- **Helper text:** Delivers further details and guidance on any necessary actions.
- **Actions (only for actionable notifications):** Includes interactive elements like ghost or tertiary buttons for user engagement.
- **Featured Icon:** Visually indicates the type of notification.
- **Close Button (optional):** Allows users to manually dismiss the notification.

Component Anatomy

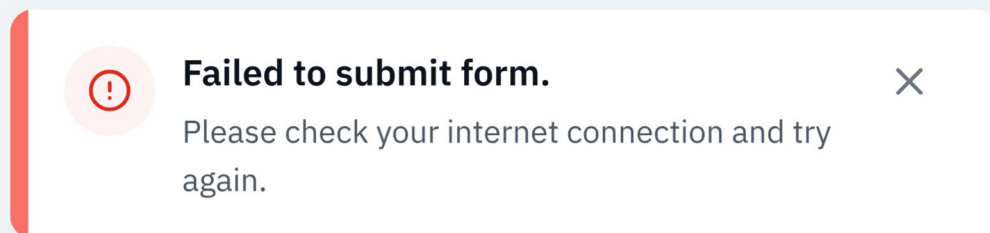
- 1 Container
- 2 Feedback Icon
- 3 Title
- 4 Helper Text
- 5 Link
- 6 Action
- 7 Closing Button



- **Container:** The area that holds all the elements.

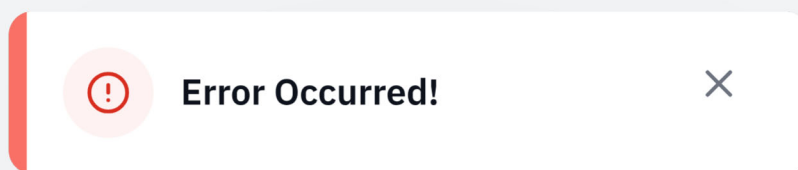
- **Feedback Icon:** This icon serves as a visual cue to quickly convey the type of message or alert, such as information, warning, or error.
- **Title:** Provides a concise summary of the banner content.
- **Helper text:** Delivers further details and guidance on any necessary actions.
- **Link:** A link included within the notification that users can click on to take them directly to a page or a section that is relevant to the message being displayed.
- **Actions (only for actionable notifications):** Includes interactive elements like ghost or tertiary buttons for user engagement.
- **Close Button (optional):** Allows users to manually dismiss the notification.

Tips to effectively use tags into your designs



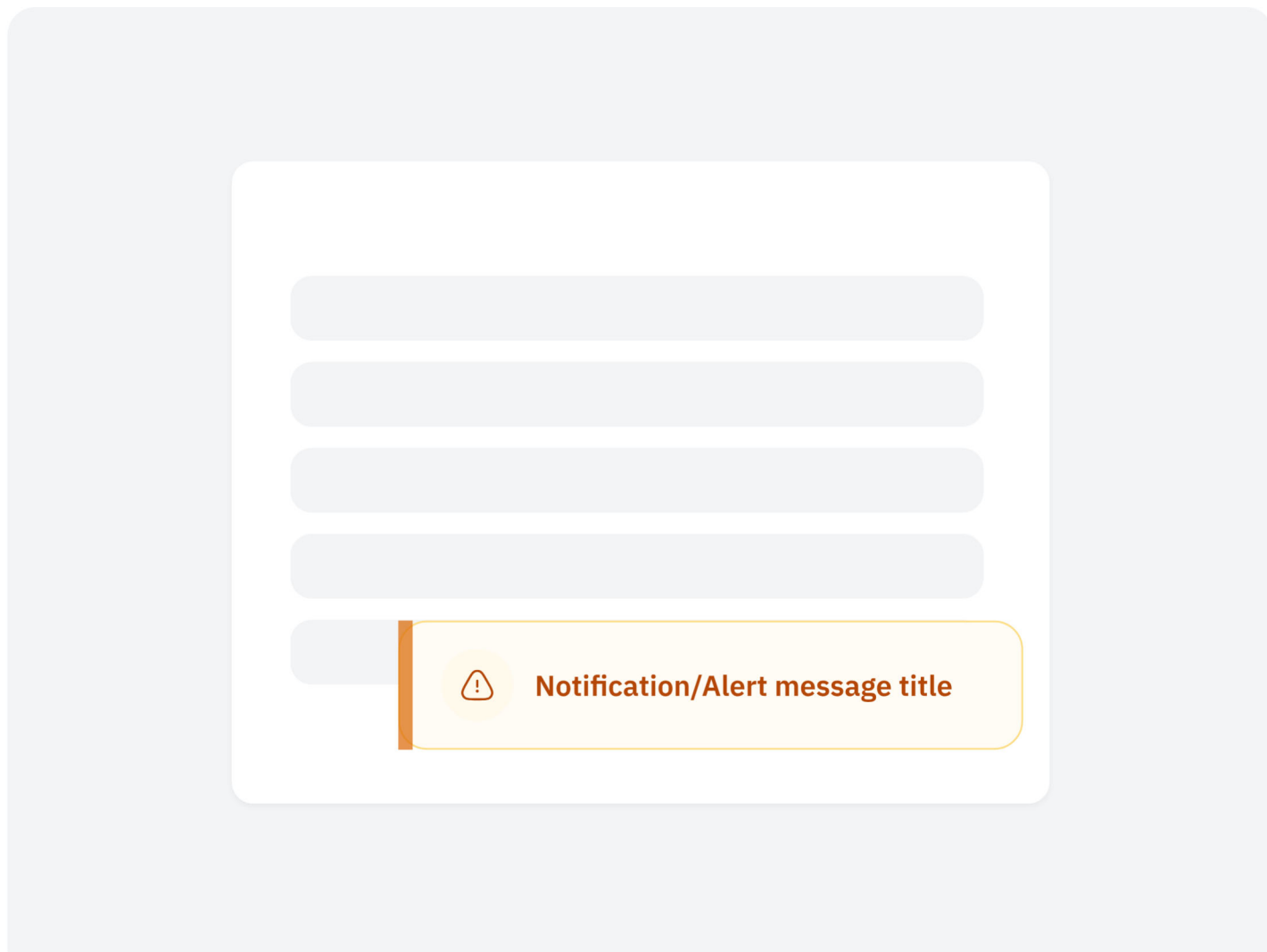
✓ Do

Ensure informative, clear and concise language in notifications to enhance user trust and clarity.



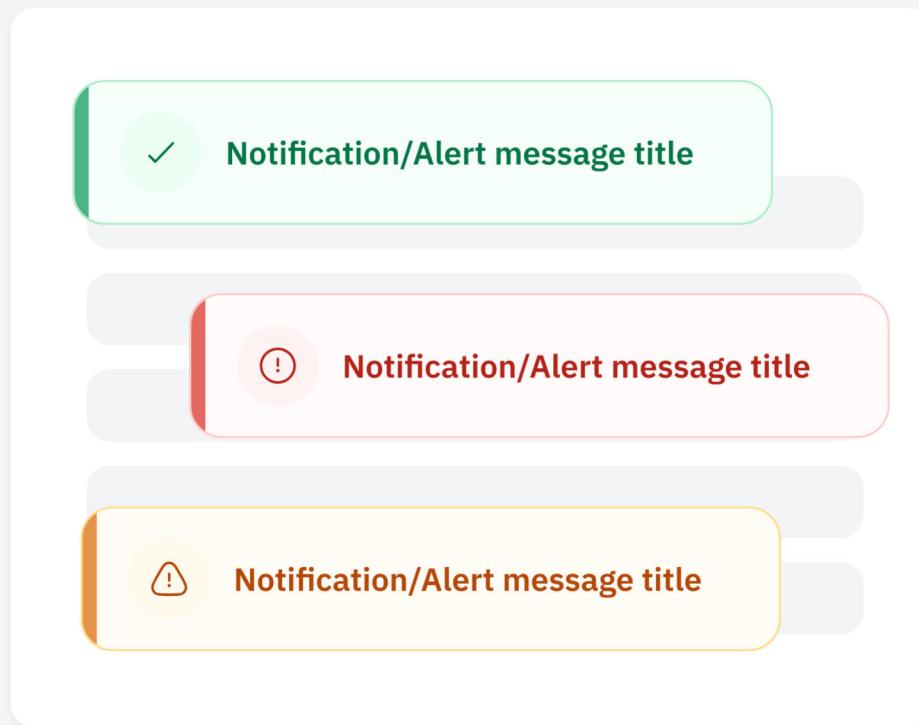
✗ Avoid

Unclear language in notifications to maintain user trust and clarity.



✓ **Do**

For Notification toast encourage users with appropriate and timely notification alerts, as this can lead to engagement and focus.



✗ Avoid

Avoid bombarding users with excessive notification alerts, as this can lead to frustration and distraction.

Accessibility

Notifications serve as an essential tool for providing contextual information and aiding workflows, particularly for users with disabilities. It is crucial to ensure that these notifications are accessible, allowing all users to perceive, understand, operate, and interact with them effectively.

Ensuring accessibility for notification or alert components in web applications is critical because these elements often convey important information that must be immediately recognizable by all users, including those with disabilities. Here's a detailed accessibility guide for creating accessible notification or alert components:

- Use **role="alert"** for important alerts that require immediate attention. This role automatically sends the content to assistive technologies and sets the **aria-live** property to **assertive**
- For less urgent notifications, consider using **aria-live="polite"** which will be announced when the user is not busy with other tasks
- Provide users sufficient time to read and interact with notifications. For persistent or critical information, consider not having a timeout.
- For notifications that do disappear, ensure they remain on the screen for an adequate duration, typically at least **5 seconds**.
- Allow users to easily dismiss or interact with notifications. Provide a clear mechanism to close the alert, such as a **close** button
- Ensure that interactive elements in notifications are keyboard accessible.
- For modal alerts that require immediate interaction, bring focus to the alert when it appears, and trap focus within the alert until it is dismissed.
- Return focus to an appropriate place in the application once the alert is dismissed.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Content Switcher

Modal

[← Previous](#)

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Feedback](#) > [Modal](#)

Modal

Modal is a popup window that appears on top of the main screen to grab user attention for immediate actions or input. It often contains focused content like forms or alerts and blocks interaction with the background until dismissed. Modals are useful for tasks needing user confirmation, feedback, or critical alerts, helping to keep focus and simplify interaction.

Live Demo



Visual



Code



Title goes here

When a Modal needs a further detailed explanation, it goes here.

Button

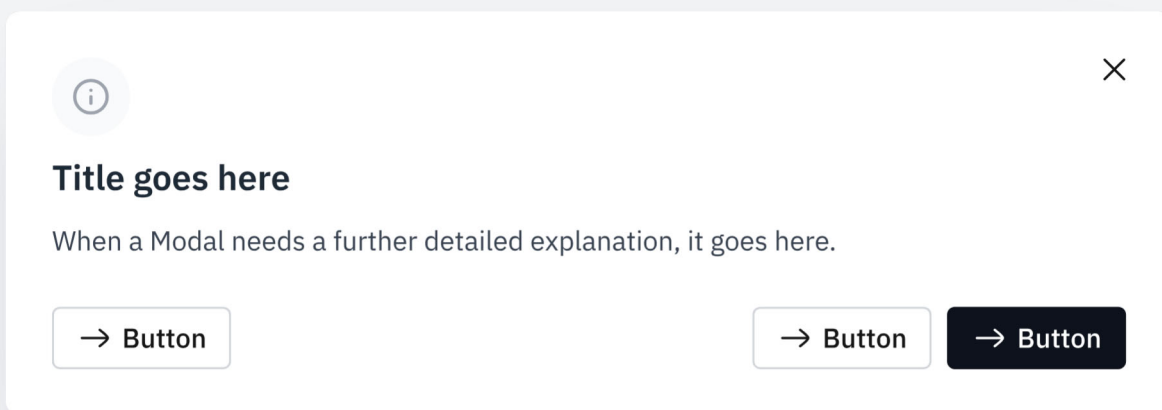
Button

Button

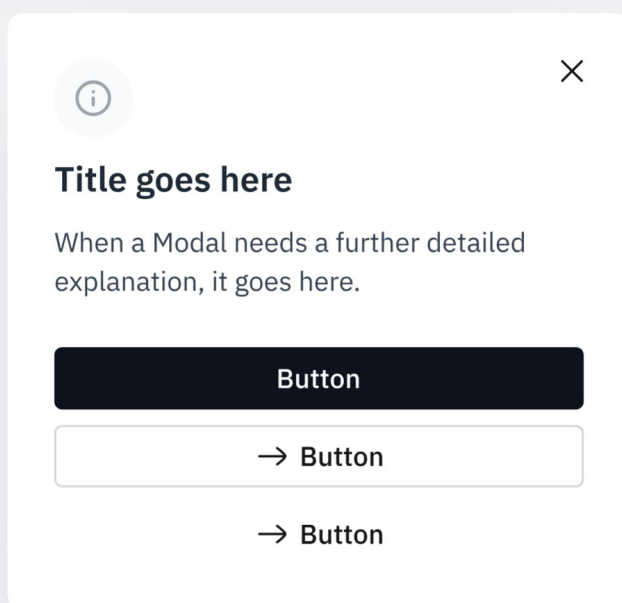
Test the preview of the modal comprehensive exploration, including all variants and features, our full demo will soon be available on **Storybook**.

Appearance

Modals are visually designed to be easily distinguishable, ensuring users quickly understand the information presented. This includes clear visual cues, such as prominent titles and distinct design, enhancing readability and guiding users through tasks.



Desktop



Responsive

Anatomy

- **Container:** The area that holds all the elements.
- **Title:** Provides a concise summary of the modal content.
- **Body Message:** Delivers further details and guidance on any necessary actions.
- **Actions (only for actionable notifications):** Includes interactive elements like ghost or tertiary buttons for user engagement.
- **Featured Icon:** Visually indicates the type of modal.
- **Close Button (optional):** Allows users to manually dismiss the modal.

Accessibility

Designing accessible modals is crucial because they often capture user input and can obstruct access to other page content. Here's a detailed accessibility guide for creating modal components.

- Use **role="dialog"** for the modal container to identify it as a dialog to accessibility APIs.
- Use **aria-modal="true"** to indicate that the modal is modal and that user interaction is limited to the dialog.
- Provide a meaningful label using **aria-labelledby** that points to the modal's title.

```
<div role="dialog" aria-modal="true" aria-labelledby="modalTitle">
  <h2 id="modalTitle">Modal Title</h2> <!-- Modal content --> </div>
```
- When the modal opens, move focus to the first interactive element or the modal container if no interactive elements are present.
- Trap focus within the modal while it is open. Ensure that tabbing cycles through the modal's interactive elements and does not escape to the background page.
- Provide a clearly labeled close button, and allow the modal to be closed using the **ESC** key.

```
<button aria-label="Close" onclick="closeModal()">Close</button>
```
- While the modal is open, ensure that elements outside the modal are not interactive. This can be achieved by hiding them from assistive technologies using **aria-hidden="true"** on background content and disabling tabbing to them.
- When the modal closes, return focus to the element that opened it. This helps maintain a logical flow of interaction for keyboard and screen reader users.
- Ensure that text within the modal has sufficient contrast against the background. Modal content should be easy to read.
- Consider all device sizes and orientations to ensure that modals are usable on mobile devices as well as desktops.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Notification

[← Previous](#)

Tooltip

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Feedback](#) > [Tooltip](#)

Tooltip

A tooltip is to provide additional information about a particular element when the user hovers their cursor over it. It typically appears as a small box or pop-up near the element and may contain text, images, or other media to provide context or guidance to the user. Tooltips are commonly used to explain the purpose of buttons, icons, or other interactive elements in an interface, enhancing user understanding and usability.

Live Demo



Visual



Code

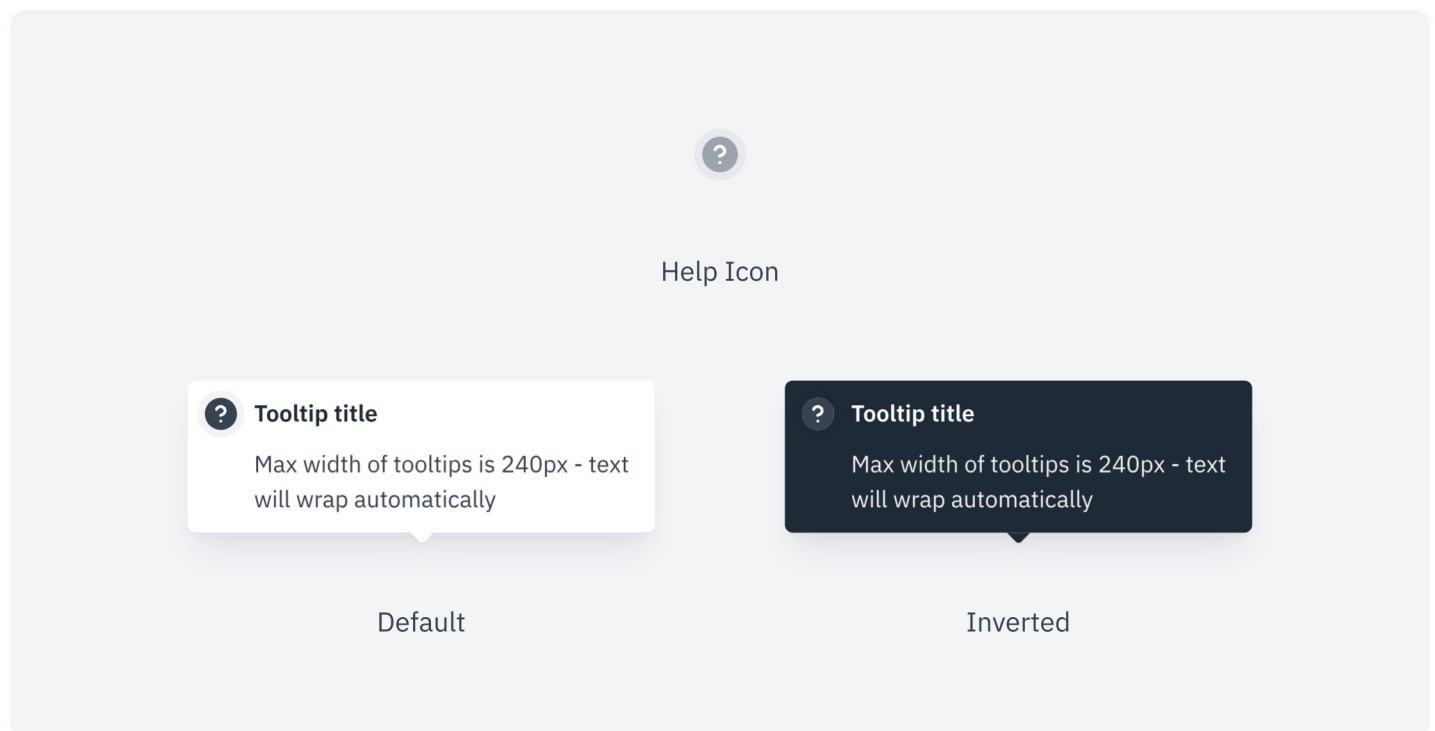
Hover Here

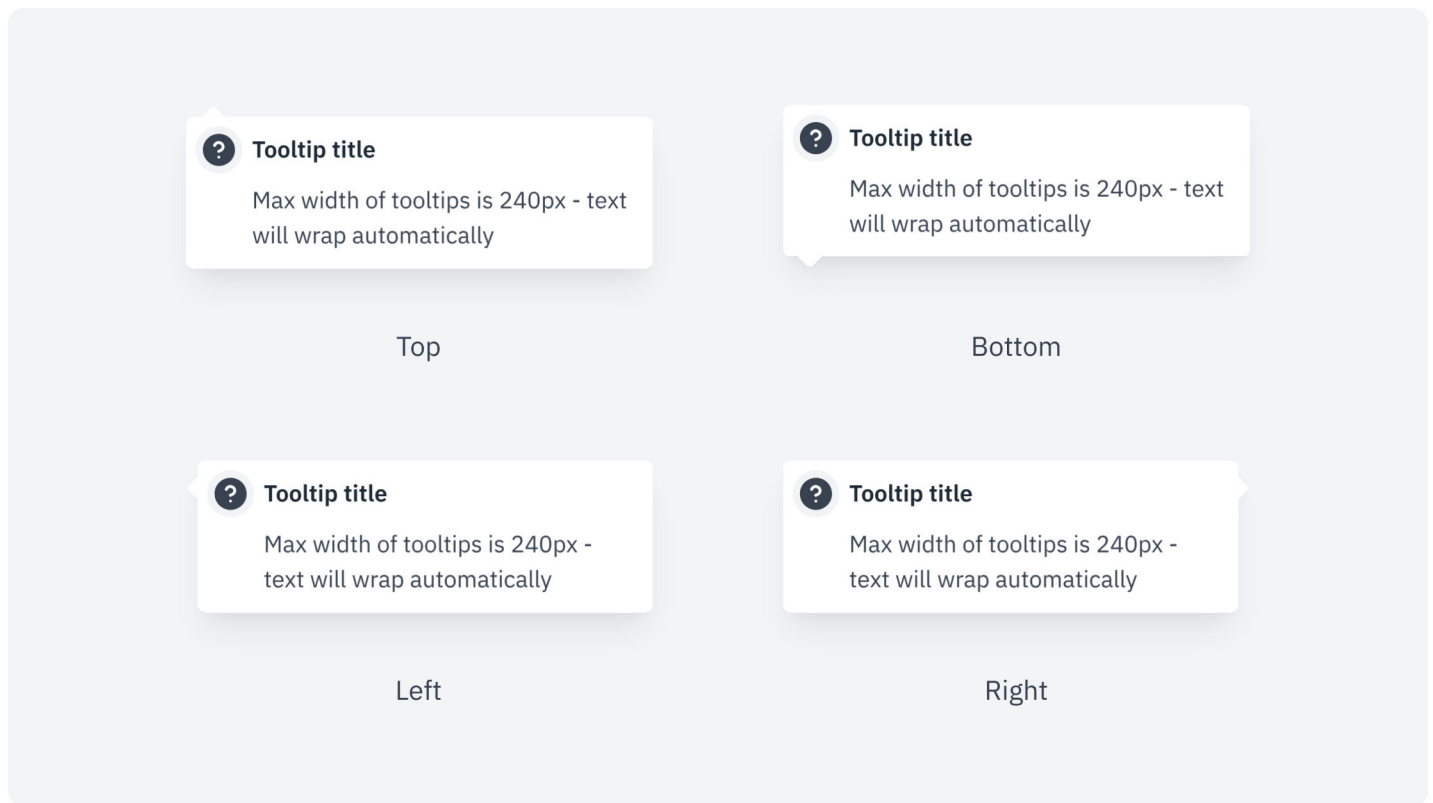
Test the preview of the tooltip comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

Tooltip is available in six fundamental formats:

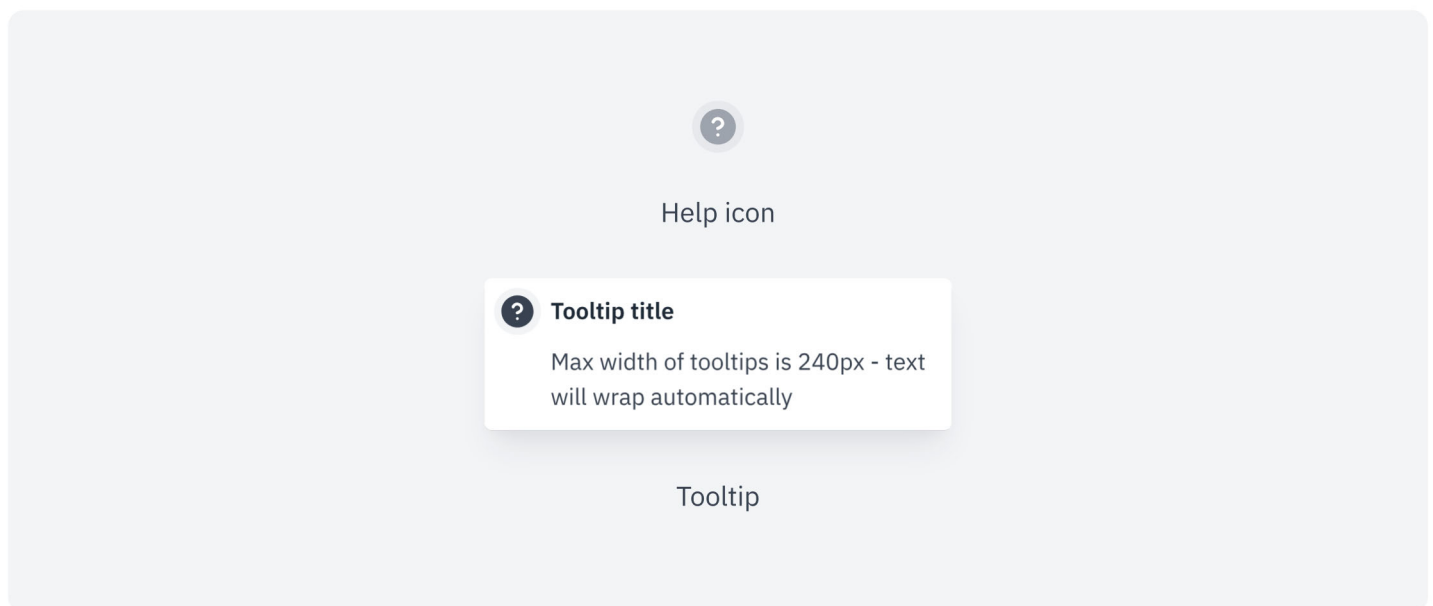
- **Default:** This is the standard appearance of the tooltip.
- **Inverted:** The tooltips are designed to have two different background colors, darker background for dark mood compared to the default light tooltip style.
- **Top:** The small arrow or triangular marker, known as the "beak," is positioned at the top of the tooltip box.
- **Bottom:** The small arrow or triangular marker, known as the "beak," is positioned at the bottom of the tooltip box.
- **Left:** The small arrow or triangular marker, known as the "beak," is positioned at the left of the tooltip box.
- **Right:** The small arrow or triangular marker, known as the "beak," is positioned at the right of the tooltip box.





Appearance

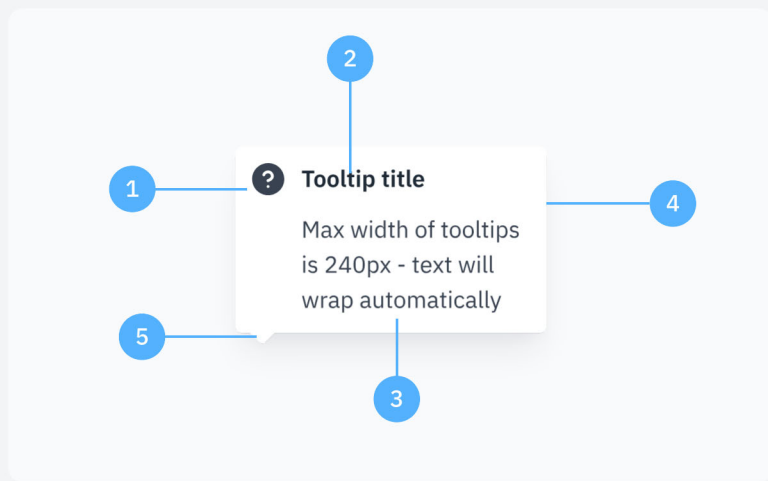
The tooltip typically involves a small box or pop-up window that appears near the UI element when the user hovers their cursor over it. It often contains text, but may also include images or other media.



Anatomy

Component Anatomy

- 1 Help icon
- 2 Title
- 3 Body Message
- 4 Container
- 5 Beak



- **Help icon:** Indicates availability of additional information or guidance.
- **Title:** Provides a concise summary of the tooltip content.
- **Body Message:** Delivers further details and guidance on any necessary actions.
- **Container:** The area that holds all the elements.
- **Beak:** Refers to a small triangular pointer or arrow that extends from the tooltip box, indicating the source or associated element that triggered the tooltip to appear.

Accessibility

Designing accessible tooltips is crucial because they must be usable by everyone, including people who rely on keyboard navigation and screen readers. Here is a comprehensive accessibility guide for creating tooltips:

- Assign ***role="tooltip"*** to the tooltip's container to semantically identify it as a tooltip.
- Ensure that tooltips are shown when the target element receives focus and hidden when it loses focus.
- Make sure that showing the tooltip does not move focus away from the triggering element unless necessary for interaction.
- Use ***aria-describedby*** on the element that triggers the tooltip, pointing to the ID of the tooltip container. This attribute should be dynamically added when the tooltip is visible and removed when it is not.

```
<button aria-describedby="tooltip1">Info</button>
```

```
<div role="tooltip" id="tooltip1" style="display:none;">More information here</div>
```

- Tooltips should only appear on hover or focus and disappear when focus and hover are removed. Ensure they do not cover other interactive elements and are positioned close to the element they describe.
- Consider adding a slight delay before showing and hiding the tooltip to prevent it from appearing too quickly or frequently. However, make sure it remains open long enough to be read by all users.
- Ensure high contrast between the text and background of the tooltip to make it easily readable.
- Make sure the text size within the tooltip is large enough to be read easily. Avoid cramping too much text into a small tooltip.
- Position tooltips so that they do not block other interface elements or prevent interaction with other controls.

Advanced Considerations

- **Dismissal:** Provide a mechanism to dismiss the tooltip without triggering any action on the underlying element. This can be implemented via an Esc key event handler if the tooltip captures focus.
- **Interactive Tooltips:** If tooltip includes links or buttons, ensure that it is treated as a modal dialog with appropriate roles and focus management.
- **Avoid Auto-Hide:** Automatically hiding tooltips can be problematic for users who read slowly. Consider alternatives like click-to-dismiss or persistent visibility while the trigger has focus.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Modal

Rating

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Feedback](#) > [Rating](#)

Rating

Rating is a method used to express user satisfaction or opinion. It involves selecting symbols like stars, and is used in product reviews, and feedback forms, helping users make decisions and provide feedback.

Live Demo



Visual



Code

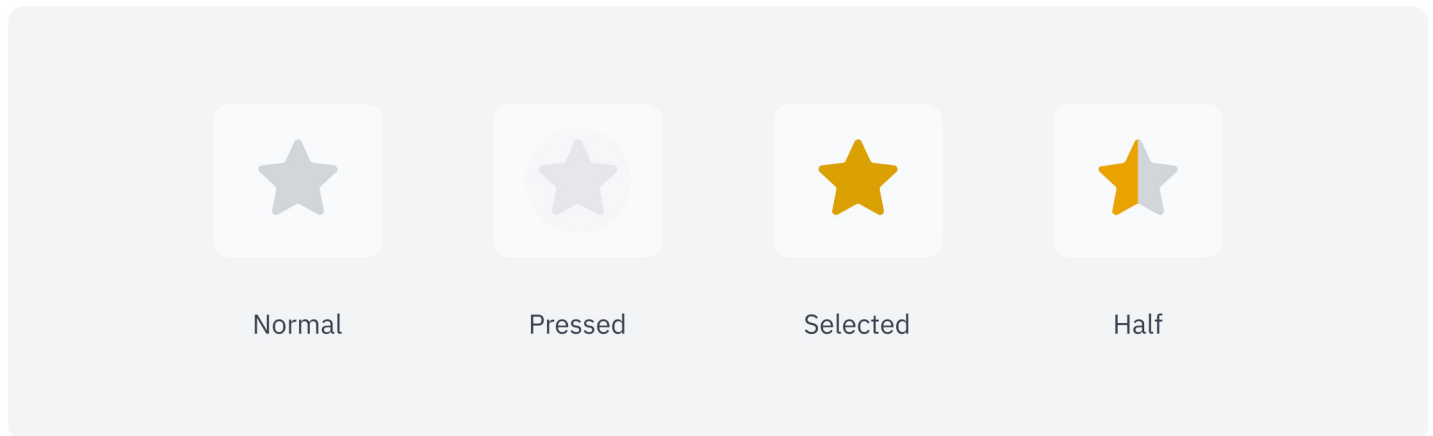


Test the preview of the rating comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

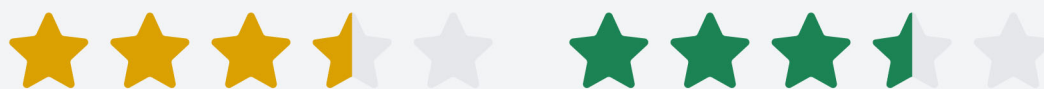
Rating is available in four fundamental formats:

- **Normal:** This is the standard appearance of the star.
- **Pressed:** This state is triggered when the user clicks the star. It provides feedback that the star has been successfully activated.
- **Selected:** Indicates that the star is currently selected or activated.
- **Half:** This symbolizes half of a complete star, used to indicate a rating that falls between two whole numbers.



Appearance

A rating's appearance typically includes a horizontal arrangement of stars, each representing a rating value. These stars come in two different types: default, and brand. With three sizes: large, medium, and small.



Default - Large

Brand - Large



Default - Medium

Brand - Medium



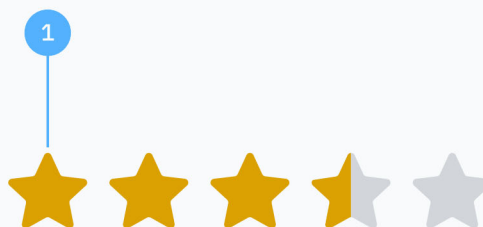
Default - Small

Brand - Small

Anatomy

Component Anatomy

1 Rating Star



- **Rating Star:** Represent a level of satisfaction or preference.

Accessibility

Creating an accessible rating component, such as those used for product reviews or feedback systems, is important to ensure that all users, including those with disabilities, can interact with them effectively. Here is a comprehensive accessibility guide for developing an accessible rating component:

- Use ***role="radiogroup"*** for the container of the rating component to indicate that it contains related interactive elements.
- Each rating option (e.g., each star) should have ***role="radio"***.
- Use ***aria-checked*** to indicate the current rating state, and ***aria-labelledby*** or ***aria-label*** for descriptive labeling.
- Ensure each interactive element (e.g., each star) is labeled so that the purpose and function are clear. ***aria-label*** can be used to provide explicit labels like "1 star", "2 stars", etc.
- Implement keyboard navigation using arrow keys to allow users to select a rating. Typically, left/right arrows can be used for a horizontal rating component.
- Ensure that the ***Tab*** key moves focus into and out of the rating component as a whole, not individual ratings.
- When a user selects a rating, update ***aria-checked*** on all the rating elements to reflect the current state accurately.
- Optionally, use ***aria-live*** to announce changes in selection for screen readers if the update is not immediately adjacent to or part of the focused element.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Tooltip

Checkbox

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Forms and Inputs](#) > [Checkbox](#)

Checkbox

A checkbox allows users to make a single selection or indicate a binary choice.

Checkboxes enable users to interact with options within an interface. They can be toggled individually for specific selections or grouped together to allow multiple choices. Additionally, checkboxes can serve as triggers for related actions or display additional information based on user selection.

Live Demo



Visual



Code



Checkbox label

When a selection needs a further detailed explanation, it goes here.



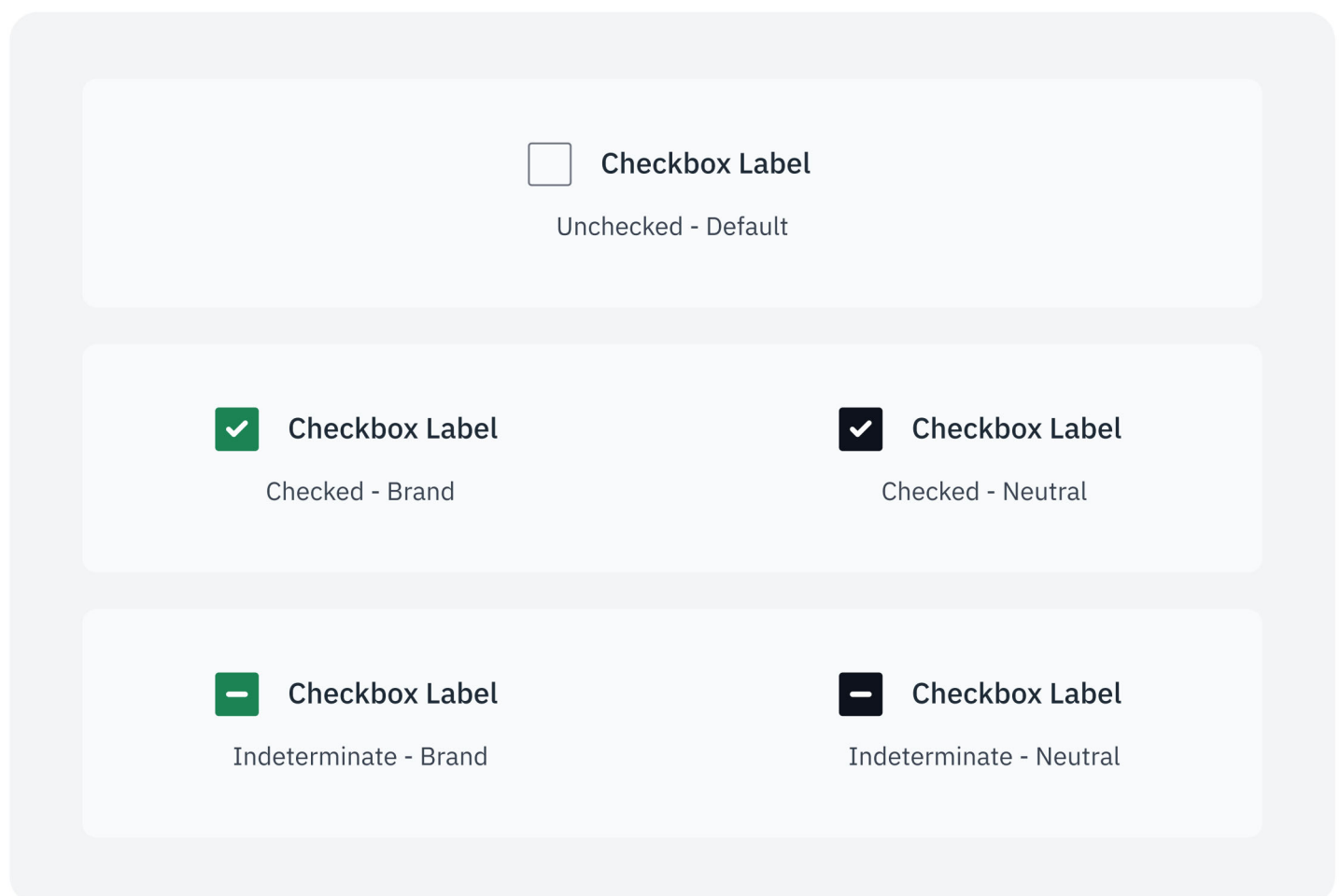
Error/Warning message

Test the preview of the checkbox comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

checkbox are available in three fundamental formats:

- **Unchecked:** In this state, the checkbox is not selected or inactive. It represents a negative or non-affirmative choice or condition.
- **Checked:** This state indicates that the checkbox is selected or active. It typically represents a positive or affirmative choice or condition.
- **Indeterminate:** This state is often used when the checkbox's state is neither fully checked nor unchecked. It's commonly employed when the checkbox is part of a group where some, but not all, options are selected. This state is useful for indicating partial selection or when the exact status is unknown or cannot be determined.



Appearance

The appearance of the checkbox is designed to be visually intuitive and easily distinguishable, ensuring that users can quickly recognize and understand the information being presented.



Checkbox Label

When a selection needs a further detailed explanation, it goes here.



Error/Warning message

Anatomy

Component Anatomy

- 1 Checkbox input
- 2 Checkbox label
- 3 Helper text
- 4 Alert message

1



Checkbox Label

When a selection needs a further detailed explanation, it goes here.



Error/Warning message

4

3

- **Checkbox input:** This is a graphical user interface element that allows users to toggle between two states, typically checked and unchecked. It's represented by a small box that can be clicked or tapped to change its state.
- **Checkbox label:** This is the text associated with a checkbox input that provides context or describes the option being selected or deselected.
- **Helper text:** A checkbox refers to additional information provided to assist users in understanding the purpose or implications of selecting the checkbox.

- **Alert message:** It is a notification that appear when a user interacts with the checkbox in a way that triggers a specific condition or action.

Accessibility

Ensuring that checkbox buttons are accessible is crucial because they are a common method for capturing user preferences and decisions. Here's a detailed accessibility guide for creating accessible radio button components:

- Use the standard `<input type="checkbox">` element for checkboxes. This ensures built-in browser handling of checkbox behavior, including accessibility features.
- Associate each checkbox with a `<label>` element to provide a textual description. This can be done by placing the input inside the label element or by using the `for` attribute linked to the checkbox's id.
`<label for="subscribe">Subscribe to newsletter`
`<input type="checkbox" id="subscribe" name="subscribe">`
OR
`<label><input type="checkbox" name="subscribe"> Subscribe to newsletter`
- While standard checkboxes typically do not require additional ARIA attributes due to their inherent accessibility features, ensure proper use of **aria-labelledby** or **aria-label** when necessary, especially in complex forms or layouts where the association might not be clear.
- Ensure that checkboxes can be focused and toggled using the keyboard. The standard HTML checkbox can be focused using the **Tab** key and toggled with the **Spacebar**.
- Changes in state (checked/unchecked) are automatically announced by screen readers when using standard HTML checkboxes, as long as they are properly labeled.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Rating

Datepicker

[← Previous](#)

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Forms and Inputs](#) > [Datepicker](#)

Date Picker

A Date Picker is a UI component that enables users to select dates via a pop-up or dropdown calendar. This tool is often triggered by clicking a calendar icon or input field. It is widely used in forms and applications for tasks like scheduling events, booking appointments, or entering birthdates.

Live Demo

Visual

Code

December 2024

< >

Su	Mo	Tu	We	Th	Fr	Sa
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	31	1	2		

Test the preview of the datepicker comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Appearance

Date Pickers are visually designed to be easily distinguishable, ensuring users quickly grasp the available dates. This includes clear visual cues, such as intuitive calendar layouts and distinct styling, enhancing usability and guiding users through date selection effortlessly.

In Platforms Code, we offer two main styles of Date Picker to accommodate different user preferences and interface requirements: Single-Month and Dual-Month view. The Single Month view displays one month at a time, providing a compact and focused interface for selecting dates. In contrast, the Dual Month view presents two months side by side, offering a broader range of date options and enhancing visual context for date selection.

Single-Month view

Label

Month DD, YYYY

?

 Help Text

January 2024

←

→

Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	29	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
31	1	2	3	4	5	6

Label

Month DD, YYYY

Label

Month DD, YYYY

?

 Help Text

?

 Help Text

January 2024

←

→

Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	29	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
31	1	2	3	4	5	6

Dual-Month view

Label

Month DD, YYYY

ⓘ Help Text

Month 2024 ▾

Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	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	1	2
3	4	5	6	7	8	9

Label

Month DD, YYYY

ⓘ Help Text

January 2024 ▾

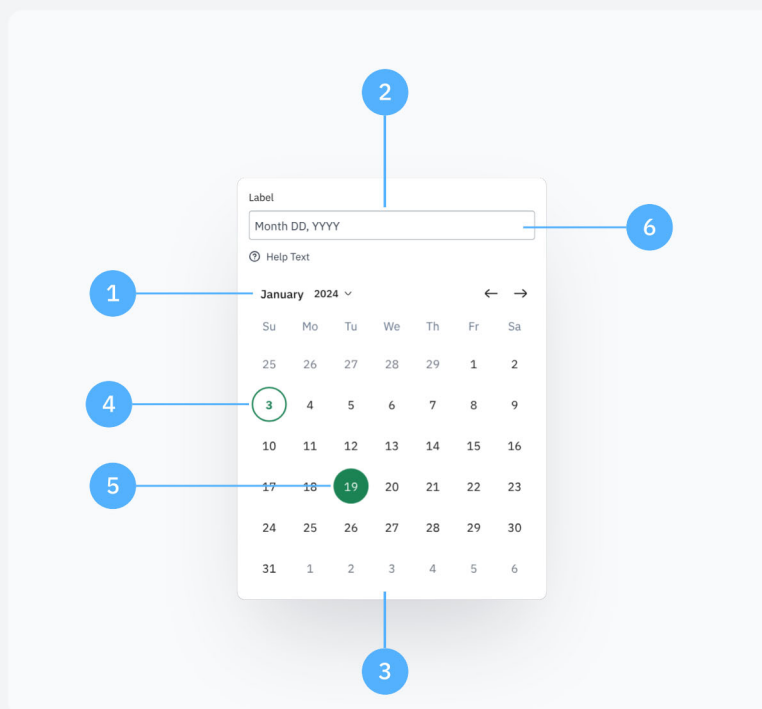
← →

Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	29	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
31	1	2	3	4	5	6

Anatomy

Component Anatomy

- 1 Month and Year
- 2 Date Input
- 3 Container
- 4 Today indicator
- 5 Selected Date
- 6 Navigation buttons



- **Month & Year:** Displays the current month and year being viewed by the user.
- **Date Input:** Allow user to enter date using their keyboard
- **Container:** The space containing all elements.
- **Today:** Circular indicator highlighting the current date on the calendar.
- **Selected Date:** Circular indicator indicating the date selected by the user or by default.
- **Actions (only for actionable notifications):** Includes interactive elements such as ghost or tertiary buttons for user engagement.
- **Navigation Buttons:** Enable users to move between months for efficient date selection.

Accessibility

Ensuring that datepicker is accessible is crucial because they are a common method for capturing user preferences and decisions. Here's a detailed accessibility guide for creating accessible radio button components:

- Use the **role="dialog"** for the date picker popup to indicate that it is a modal dialog.
- Apply **role="application"** to the date grid inside the dialog to allow for more complex keyboard commands typical of applications.
- Use **grid**, **gridcell**, **row**, and **columnheader** roles to structure the date grid.
- Use **aria-labelledby** and **aria-describedby** to provide accessible name and descriptions respectively.
- Ensure each interactive element within the date picker is appropriately labeled with **aria-label** or **aria-labelledby**.
- When the date picker opens, move focus to the currently selected date, or to the current day if no date is selected.
- Ensure that keyboard navigation within the date grid is logical (using arrow keys to navigate dates).
- Implement keyboard shortcuts for navigation (e.g., arrow keys for navigating dates, Enter to select a date, **Esc** to close the picker).
- Provide mechanisms to quickly navigate months and years, which are also keyboard accessible.
- Use **aria-live** regions to announce the currently focused day and any changes in the month or year.
- Update **aria-selected** on the currently selected date.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Checkbox

Input

← Previous

Up Next →



Home > Guidelines > Components > Forms and Inputs > Input

Input

Text input in Platforms Code are interactive elements where users can enter or edit text, numbers, or other data. They provide a means for users to input information, such as usernames, passwords, or search queries, and are commonly found in forms or search bars. Input fields typically feature visual cues like placeholder text or labels to guide users and enhance usability.

Live Demo



Visual



Code

Entered text

Test the preview of the input comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Appearance

The Platforms Code Design System offers a wide range of input field variations tailored to diverse design needs and user interactions, accommodating various elements inside the container. This variety of inputs caters to a wide range of user interactions, ensuring flexibility and adaptability.

Types

Inputs are available in four types:

- **Standard input:** A standard input is a fundamental user interface element that allow user to enter any information using their keyboard.
- **Input with Icon:** A standard input field featuring an icon to visually represent its purpose.
- **Input with prefix and suffix:** An input field with two visually separated extended containers, often interactive, used to provide contextual information. For example, adding the country code before the phone number or the currency symbol after the amount.
- **Read only Input:** A Read-Only Input is a text field within a UI that displays information but doesn't allow user modification

Label

Entered text

Standard input

Label

Q Entered text

Input with icon

Label

+966 Q Entered text

Prefix

Label

Q Entered text SAR

Suffix

Label

Q Entered text

Read only

Filled styles

- **Filled - Default** : The default style of the filled input with a standard background and stroke.
- **Filled - Lighter** : A lighter style of the filled input with no stroke and a light background.
- **Filled - Darker** : A darker style of the filled input with no stroke and a darker background.

Label

Q Entered text

Filled - Default

Label

Q Entered text

Filled - Lighter

Label

Q Entered text

Filled - Darker

Helper Texts

- **Helper Texts:** This is a standard input field accompanied by supporting information. It is designed to guide users to enter accurate data, providing additional details about the field.
- **Error Helper text:** Similar to the standard helper text, this input field is highlighted in red to indicate unsuccessful data entry. It guides users to correct their input, offering extra information about the error encountered.

Label

Entered text

Helper text

Label

Entered text

Error Helper text

Behaviors (States)

Inputs in the Platforms Code exhibit various behaviors depending on user interaction. These behaviors include:

- **Default:** The initial state of an input field when not being interacted with, appearing as a standard, editable field.
- **Hovered:** Triggered when the user hovers over the input field without clicking, providing visual feedback.
- **Pressed:** This state is triggered when the user clicks or taps the input field, providing feedback that the field has been successfully activated.
- **Focused:** Occurs when an input field receives keyboard focus, typically through the "Tab" key, essential for accessibility to navigate through the interface using the keyboard.
- **Disabled:** A state indicating that the input field is not editable or interactive, often used in user interfaces to communicate that the input field cannot be modified under the current circumstances.

Label

Entered text

Default state

Label

Entered text

Hovered state

Label

Entered text

Pressed state

Label

Entered text

Focused state

Label

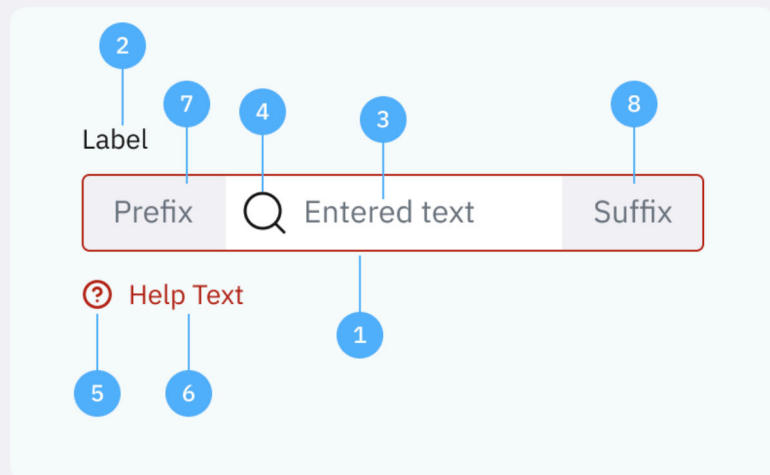
Entered text

Disabled state

Anatomy

Component Anatomy

- 1 Container
- 2 Input Label
- 3 Placeholder
- 4 Icon
- 5 Feedback Icon
- 6 Helper Text
- 7 Prefix
- 8 Suffix



- **Container:** The editable area that holds all the elements
- **Input label:** Displays the title of the input
- **Placeholder:** A hint or example text shown inside the input field when it is empty.
- **Icon:** Provides visual representation of the input purpose/name.
- **Feedback icon:** Provides visual representation of the entry validity
- **Helper Text:** Provides user with extra information to help deli
- **Prefix:** Information added before the main input field to provide contextual information.
- **Suffix:** Information added after the main input field to provide contextual information.

Accessibility

Ensuring that these input fields are accessible is crucial because they often form the backbone of user interactions on websites and applications. Here's a detailed accessibility guide for creating accessible input components.

- Use the correct HTML elements and types for the input to ensure that the browser and assistive technologies can correctly interpret the element's purpose.
- Ensure that all input fields can be accessed and used via the keyboard alone.
- Use specific type attributes for **<input>** elements when appropriate, such as **email**, **tel**, **number**, etc., to invoke the correct keyboard and provide built-in validation in

some browsers.

```
<input type="email" id="email">
```

- Use **<label>** elements to provide a clear, descriptive label for each input. For screen readers, associate the label with the input using the `for` attribute that matches the input's `id`. If a visible label is not desirable (eg., for design reasons), use `aria-label` or **`aria-labelledby`** to provide an accessible name.

```
<label for="email">Email:</label>
```

```
<input type="email" id="email" name="email">`
```

- Use **`aria-describedby`** to link any additional descriptive text or error messages to the input, helping users understand constraints or corrections.

```
<input type="text" id="username" aria-describedby="usernameHelp">
```

```
<small id="usernameHelp">Your username must be 8-20 characters long.</small>
```

- Ensure that users can navigate to and from the input field using standard keyboard controls **Tab** and **Shift + Tab**
- Clearly indicate fields that have errors after validation and provide helpful error messages. Use **`aria-invalid="true"`** on inputs with errors and link to a descriptive error message using **`aria-describedby`**

```
<input type="text" id="username" aria-invalid="true" aria-describedby="errorUsername">
```

```
<span id="errorUsername">Error: Username is already taken.</span>
```
- Provide clear visual indicators for focus states, such as borders or shadows, to help users determine which element is active.
- Ensure high contrast between text and background colors, and use legible font sizes.
- Ensure inputs are usable on all devices, with touch targets that are adequately sized for mobile users.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Datepicker

← Previous

File Uploader

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Forms and Inputs](#) > [File Uploader](#)

File uploader

File uploader facilitates selecting and uploading files from the user's device to a web platform. Users initiate the process through a designated area, review selections, and monitor upload progress. Upon completion, they receive confirmation, ensuring a smooth experience.

Live Demo



Visual



Code



Drag and drop files here to upload

csv, pdf, png, jpeg

Browse Files

Test the preview of the file uploader comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Appearance

File uploader in UI is designed for user-friendly file selection and upload. With intuitive buttons or designated areas, users easily choose and review files. Visual indicators, like progress bars, ensure users track upload status. The aim is a seamless, efficient experience guiding users through the process.

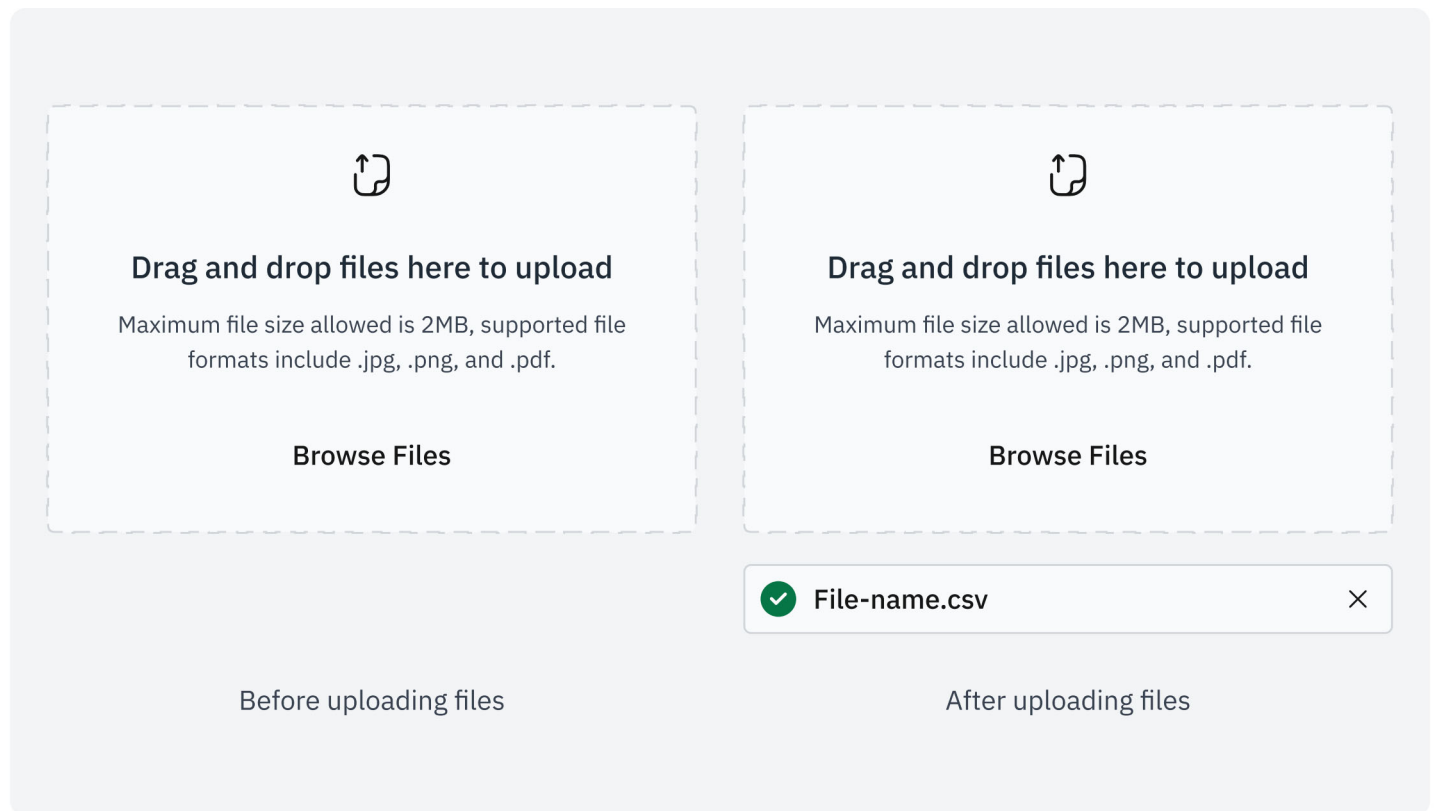
Variants

The file uploader comes two various variants:

Single file Uploader

Upload files Maximum file size allowed is 2MB, supported file formats include .jpg, .png, and .pdf. Browse Files Before uploading a file	Upload files Maximum file size allowed is 2MB, supported file formats include .jpg, .png, and .pdf. ✓ File-name.csv ✕ After uploading a file
---	--

Multi files Uploader



Behaviors (States)

The file uploader exhibits various behaviors (states) based on user interactions.

- **Default:** The initial state of the file uploader component when no interaction has occurred.
- **Drag + Hovered:** Triggered when the user hovers over the file uploader area or drags a file into it.
- **Disabled:** A state indicating that the file uploader component is disabled and not accepting files. This is often used to communicate that the component cannot be activated in the current context, such as when a switch is turned off.



Drag and drop files here to upload
Maximum file size allowed is 2MB, supported file formats include .jpg, .png, and .pdf.

Browse Files

Default



Drag and drop files here to upload
Maximum file size allowed is 2MB, supported file formats include .jpg, .png, and .pdf.

Browse Files

Drag + Hover



Drag and drop files here to upload
Maximum file size allowed is 2MB, supported file formats include .jpg, .png, and .pdf.

Browse Files

Disabled

Upload files
Maximum file size allowed is 2MB, supported file formats include .jpg, .png, and .pdf.

Browse Files

Default

Upload files
Maximum file size allowed is 2MB, supported file formats include .jpg, .png, and .pdf.

Browse Files

Disabled

Anatomy

- **Drag and drop area:** The area where users can drop files.
- **Title:** Provides a brief overview of the component's purpose.
- **helper text:** Offers additional details and guidance, including accepted file types for upload.
- **Brows files button:** Allows users to access their local device file system for file selection.
- **Featured Icon:** Visual indicator of the action type, such as uploading files or images.
- **File uploading status icon:** Enables users to monitor the status of file uploads.
- **File name:** Displays the name of the uploaded or currently uploading files names.
- **File bar helper text:** Provides guidance or error messages related to file upload.
- **Delete file:** Removes uploaded or currently uploading files from the queue.

Component Anatomy

- 1 Label
- 2 Helper Text
- 3 Button

Upload files — 1

Maximum file size allowed is 2MB, supported file formats include .jpg, .png, and .pdf.

Browse Files — 3

2

- **Label:** Provides a brief overview of the component's purpose.
- **helper text:** Offers additional details and guidance, including accepted file types for upload.
- **Button:** The area where users can drop files.

Accessibility

The File Uploader component is designed to be fully accessible, building upon the native HTML `<input type="file">` element. It includes a visual `<button>` and a drag-and-drop area, both of which are optimized for accessibility and can receive keyboard focus.

Keyboard Support

The component supports keyboard navigation and operation, making it accessible for users who rely on keyboard input instead of a mouse. Here are the steps and key operations for using the File Uploader with a keyboard:

Navigating to the Component:

Users can navigate to the File Uploader component using the Tab key to move forward through interactive elements, or Shift + Tab to navigate backwards.

Activating the File Input:

Once the File Uploader drag-and-drop area or the button has focus, users can activate the file input dialog by pressing the Enter (or Return) or Space keys.

This setup ensures that all users, regardless of their device or abilities, can interact with the File Uploader component effectively. By leveraging native HTML capabilities enhanced with ARIA (**`aria-label="File upload area, press Enter or Space to upload files"`**) attributes and proper keyboard event handling, the component remains accessible and user-friendly.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Input

Radio

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Forms and Inputs](#) > [Radio](#)

Radio

A radio button lets users choose one option from a group of options. When they pick one, any previously selected option is deselected. This way, users can only select one option at a time.

Radio buttons are commonly used in forms and surveys when users need to pick just one choice from a list.

Live Demo



Visual



Code



Radio Button label

Test the preview of the radio comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

Radio button are available in two fundamental formats:

- **Unselected buttons:** In this state, the radio button is not selected or inactive. It represents a negative or non-affirmative choice or condition.
- **Selected buttons:** This state indicates that the radio button is selected or active. It typically represents a positive or affirmative choice or condition.



Radio Label

Selected - Brand



Radio Label

Selected - Neutral



Radio Label

When a selection needs a further detailed explanation, it goes here.

Helper text - Neutral



Radio Label



Error/Warning message

Alert message - Neutral



Radio Label

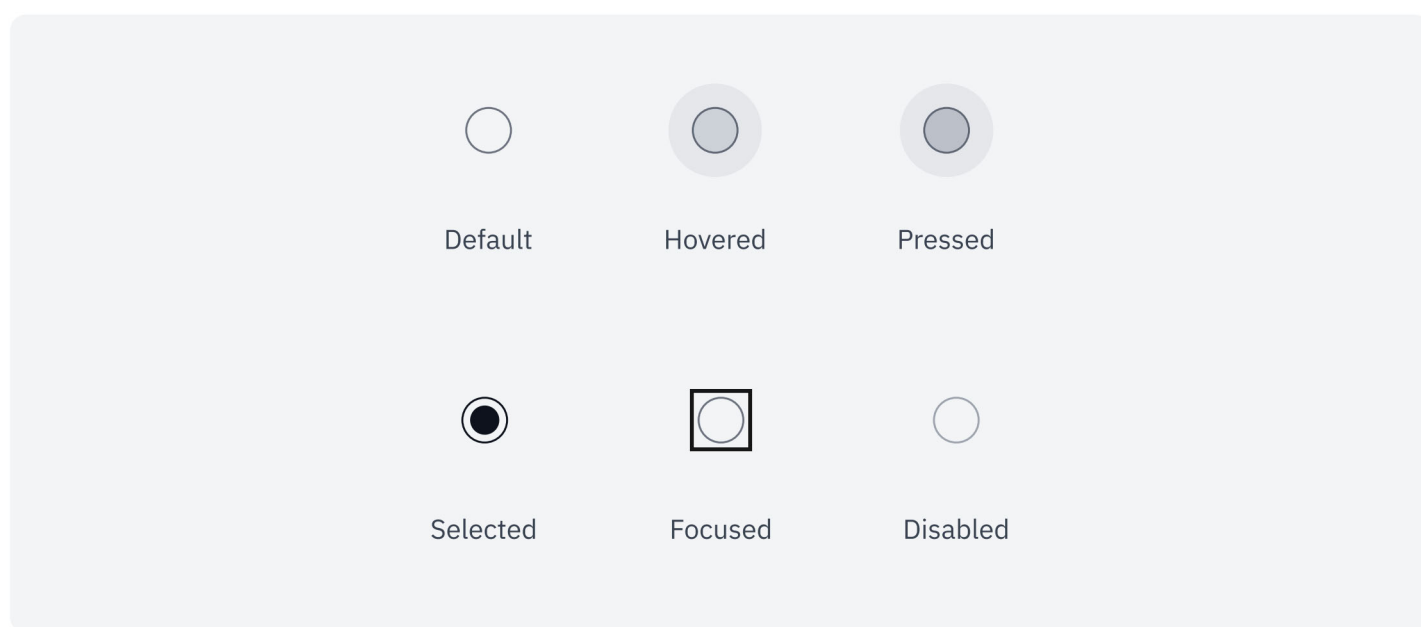
Unselected - Default

Behaviors (States)

Radio buttons in the Platforms Code exhibit various behaviors when they are selected and not selected. Radio buttons behaviors state are:

- **Default:** The default state of a radio button when it is not being interacted with. it appears as a standard, clickable radio button.

- **Hovered:** Occurs when the user moves the cursor over the radio button without clicking.
- **Pressed:** This state is triggered when the user clicks or taps the radio button. It provides feedback that the radio button has been successfully activated
- **Selected:** Applicable in scenarios where radio buttons can be toggled on and off. Indicates that the button is currently selected or activated.
- **Focused:** Occurs when a radio button receives keyboard focus, usually through the "Tab" key. Important for accessibility to ensure users can navigate through the interface using the keyboard.
- **Disabled:** The disabled state of a radio button is a condition where the radio button is not clickable or interactive. This state is often used in user interfaces to communicate that the radio button cannot be activated under the current circumstances



Appearance

The appearance of radio button is designed to be visually intuitive and easily distinguishable, ensuring that users can quickly recognize and understand the information being presented.



Radio Label

When a selection needs a further detailed explanation, it goes here.



Error/Warning message

Anatomy

Component Anatomy

- 1 Radio input
- 2 Radio label
- 3 Helper text
- 4 Alert message

1



Radio Label

When a selection needs a further detailed explanation, it goes here.



Error/Warning message

4

3

- **Radio input:** Indicates the state of a radio button. By default, no option will be preselected.
- **Radio label:** This is the text associated with a radio button input that provides context or describes the option being selected or deselected.
- **Helper text:** A Radio button refers to additional information provided to assist users in understanding the purpose or implications of selecting the checkbox.
- **Alert message:** It is a notification that appear when a user interacts with the radio button in a way that triggers a specific condition or action.

Accessibility

Ensuring that radio buttons are accessible is crucial because they are a common method for capturing user preferences and decisions. Here's a detailed accessibility guide for creating accessible radio button components:

- Utilize the `<input type="radio">` for each option and group them by giving them the same name attribute.
- Always use `<label>` elements to provide a clear, descriptive label for each radio button. Associate the label with its corresponding radio button using the `for` and `id` attributes.
- Use `aria-label` to provide an accessible name.
- Use a fieldset and legend to group radio buttons and provide a group label. This helps users understand the context of the radio buttons.

```
<label for="option1">Option 1</label>
```

```
<input type="radio" id="option1" name="group1" value="1">
```

```
<fieldset>
```

```
<legend>Choose an Option</legend>
```

```
<label for="option1">Option 1</label>
```

```
<input type="radio" id="option1" name="group1" value="1">
```

```
<label for="option2">Option 2</label>
```

```
<input type="radio" id="option2" name="group1" value="2">
```

```
</fieldset>
```

to provide an accessible name.

- Radio buttons should be focusable with the **Tab** key. Only one radio button in the group should be in the tab sequence; navigation within the group should be handled with arrow keys.
- Ensure that left and right arrow keys (and up and down) can be used to change the selection within the group. This should cycle through the options without needing to tab between each one.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

File Uploader

Switch

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Forms and Inputs](#) > [Switch](#)

Switch

A switch is a control used to alternate quickly between two possible states. Toggles are used specifically for binary actions that occur immediately when the user "flips the switch". They are most commonly used for "on/off" settings.

Live Demo

Visual

Code



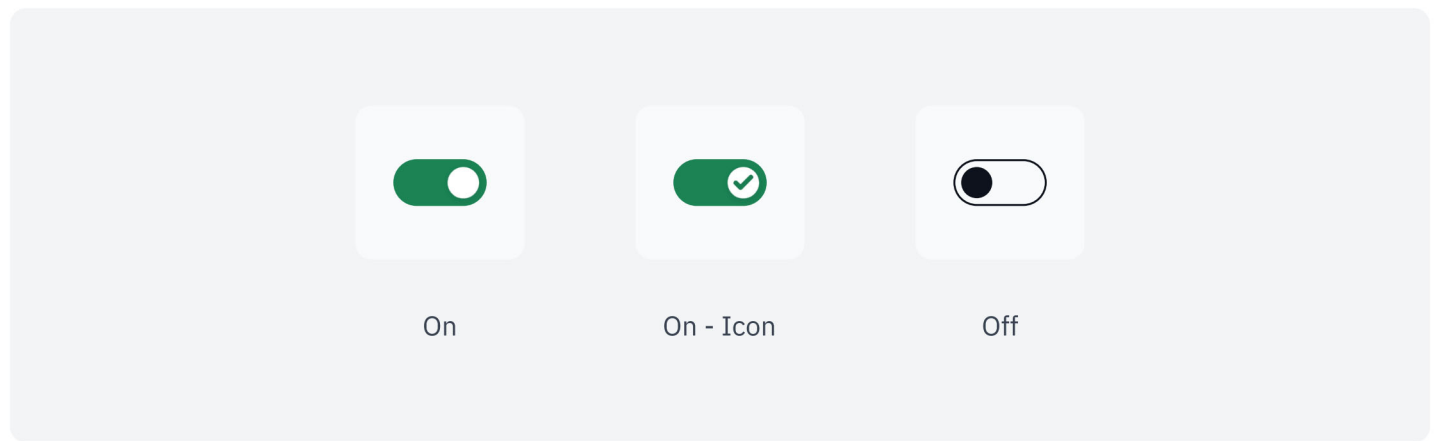
Switch label

Test the preview of the switch comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

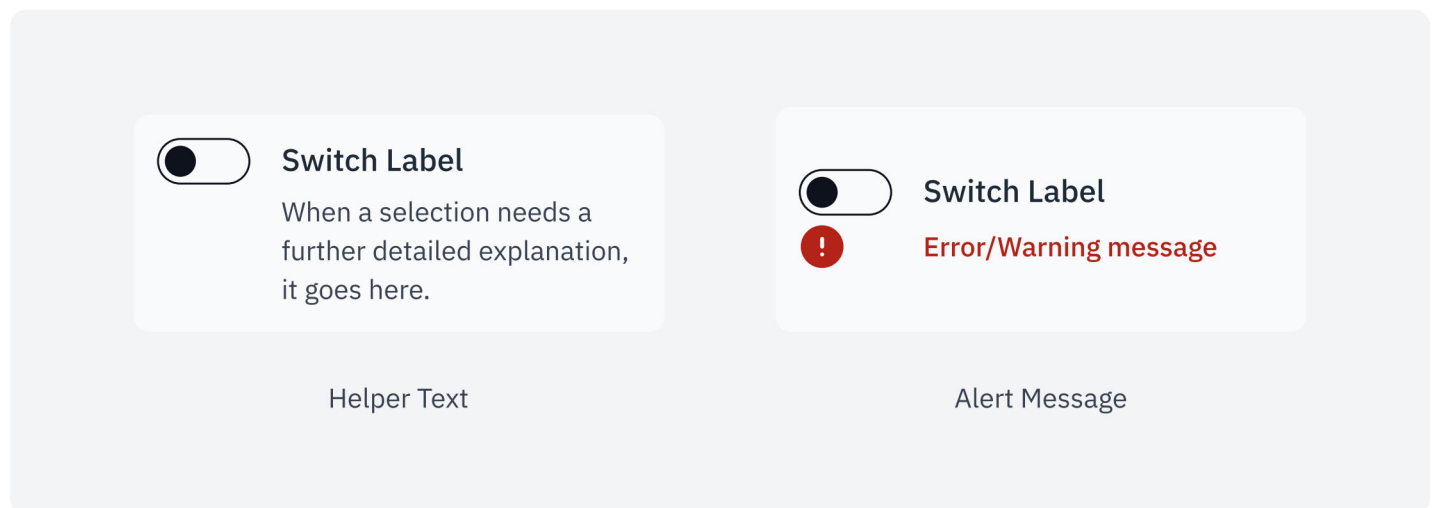
Variants

Switch are available in two fundamental formats:

- **On:** Indicates the active state of the toggleable option.
- **off:** Indicates the state when the toggleable option is deactivated or disabled.



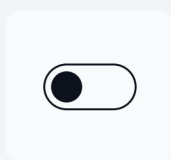
- **Helper Text:** This offers additional context or instructions to users about the function or purpose of the switch.
- **Alert Message:** This refers to a notification or prompt that appears when a specific action is taken with the switch.



Behaviors (States)

Switch in the Platforms Code exhibit various behaviors when they are selected and not selected. Radio buttons behaviors state are:

- **Default:** The default state of a switch when it is not being interacted with. it appears as a standard, clickable switch.
- **Hovered:** Occurs when the user moves the cursor over the switch without clicking.
- **Pressed:** This state is triggered when the user clicks or taps the switch. It provides feedback that the switch has been successfully activated.
- **Focused:** Occurs when a switch receives keyboard focus, usually through the "Tab" key. Important for accessibility to ensure users can navigate through the interface using the keyboard.
- **Disabled:** The disabled state of a switch is a condition where the switch is not clickable or interactive. This state is often used in user interfaces to communicate that the switch cannot be activated under the current circumstances.



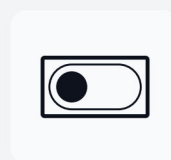
Default



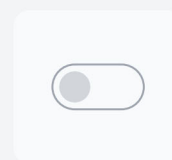
Hovered



Pressed



Focused



Disabled

Appearance

The appearance of switch is designed to be visually intuitive and easily distinguishable, ensuring that users can quickly recognize and understand the information being presented.



Switch Label

When a selection needs a further detailed explanation, it goes here.

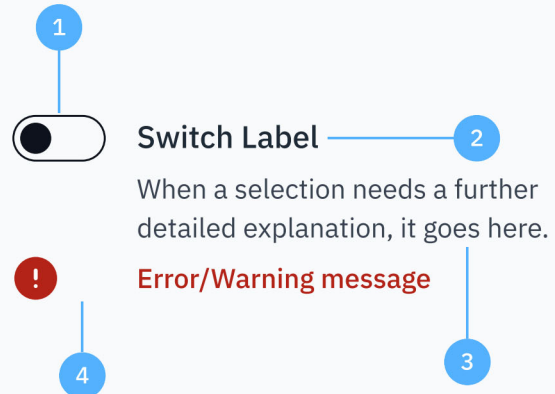


Error/Warning message

Anatomy

Component Anatomy

- 1 Switch
- 2 Switch Label
- 3 Helper text
- 4 Alert message



- **Switch:** Indicates the state of a switch, which is off by default
- **Switch label:** This text is associated with a switch input. It provides context or describes whether the option is on or off.
- **Helper text:** To guide on their purpose or function.
- **Alert message:** It is a notification that appear when a user interacts with the switch in a way that triggers a specific condition or action.

Accessibility

It is important to design the switch to be accessible to ensure that all users, including those with disabilities, can easily recognize and interact with it. Here is a guide to building an accessible toggle switch component:

- Since a switch typically represents a binary option, it can be implemented using a `<input type="checkbox"/>`. This approach provides built-in support for many accessibility features.
- Use ***aria-checked*** to indicate whether the switch is on (***true***) or off (***false***)
- Use ***role="switch"*** to explicitly convey that the element is a switch to assistive technologies, particularly when the visual design strongly suggests a switch `<input`


```
type="checkbox" id="darkMode" name="darkMode" role="switch" aria-checked="false">
```

- Always use a **<label>** element or **aria-label** to provide a clear and concise label for the switch.
- If the switch changes the state dynamically, consider using **aria-labelledby** to reference an element that can have dynamic text indicating the state.
- Ensure the switch can be toggled using the keyboard. Typically, the **Spacebar** toggles the state of a checkbox.
- The switch should be focusable with the **Tab** key.
- The visual state of the switch and any associated text labels must update immediately upon state change.
- If using JavaScript to enhance the switch, ensure that updates to **aria-checked** are managed correctly.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Radio

Textarea

[← Previous](#)

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Forms and Inputs](#) > [Textarea](#)

Textarea

A text area is a UI element that enables users to input and edit multiple lines of text, providing a larger space than a regular text input field, making it suitable for longer text entries, such as comments, descriptions, or messages.

Live Demo



Visual



Code

Enter Text

Test the preview of the textarea comprehensive exploration, including all variants and features, our full demo will soon be available on **Storybook**.

Appearance

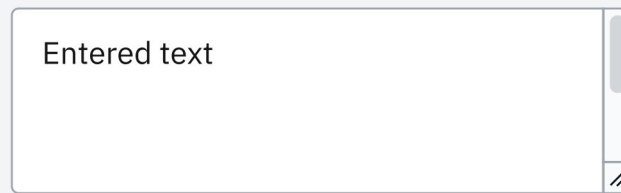
Text area element provides a wide range of variations tailored to diverse design needs and user interactions, This variety caters to a wide range of user interactions, ensuring flexibility and adaptability.

Types

Textarea are available in three types:

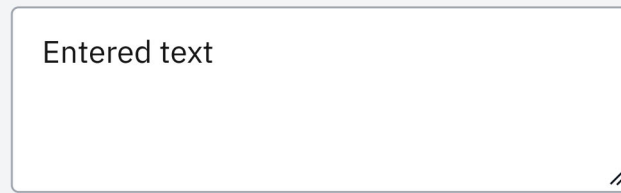
- **Standard Textarea:** This textarea includes a scrollbar on the right side, allowing users to scroll through the content if it exceeds the visible area. The textarea is resizable, which means users can drag the corner to adjust the size.
- **Textarea Without Scrollbar:** This textarea does not have a scrollbar, meaning it will not allow users to scroll through content if it exceeds the visible area. The textarea is resizable, so users can adjust its size by dragging the corner.
- **Non-Resizable Textarea:** This textarea includes a scrollbar on the right side, similar to the standard textarea, allowing users to scroll through content. The textarea is non-resizable, meaning users cannot change its size.

Label

A rectangular text input field with a thin grey border. It contains the text "Entered text". On the right side, there is a vertical scrollbar and a small square handle with two diagonal lines at the bottom right corner, indicating it is resizable.

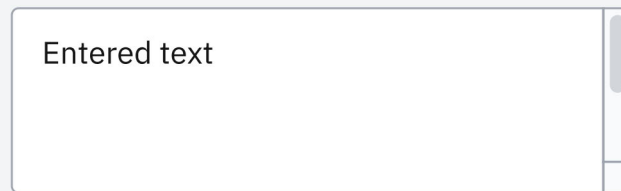
Standard textare

Label

A rectangular text input field with a thin grey border. It contains the text "Entered text". There is no scrollbar on the right side, but a small square handle with two diagonal lines is visible at the bottom right corner, indicating it is still resizable.

Without scrollbar

Label

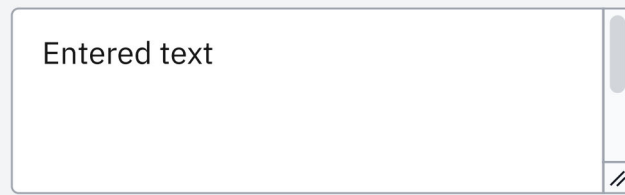
A rectangular text input field with a thin grey border. It contains the text "Entered text". On the right side, there is a vertical scrollbar. The bottom right corner does not have a resize handle, indicating it is non-resizable.

non resizable text area

Filled

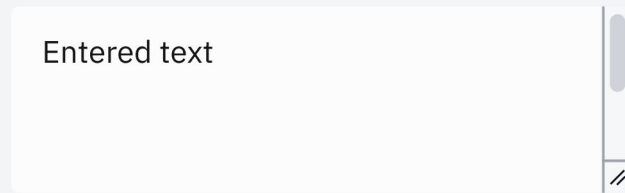
- **Filled - Default:** The default style of the filled textarea with a standard background and stroke.
- **Filled - Lighter:** A lighter style of the filled textarea with no stroke and a light background.
- **Filled - Darker:** A darker style of the filled textarea with no stroke and a darker background.

Label

A rectangular text area with a thin grey border. It contains the text "Entered text" in a dark grey font. On the right side, there is a vertical scrollbar. At the bottom right corner, there is a small icon consisting of two parallel diagonal lines.

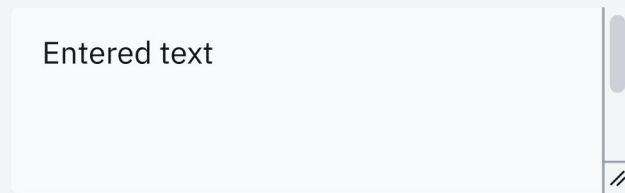
Filled - Default

Label

A rectangular text area with a thin grey border. It contains the text "Entered text" in a dark grey font. On the right side, there is a vertical scrollbar. At the bottom right corner, there is a small icon consisting of two parallel diagonal lines.

Filled - Lighter

Label

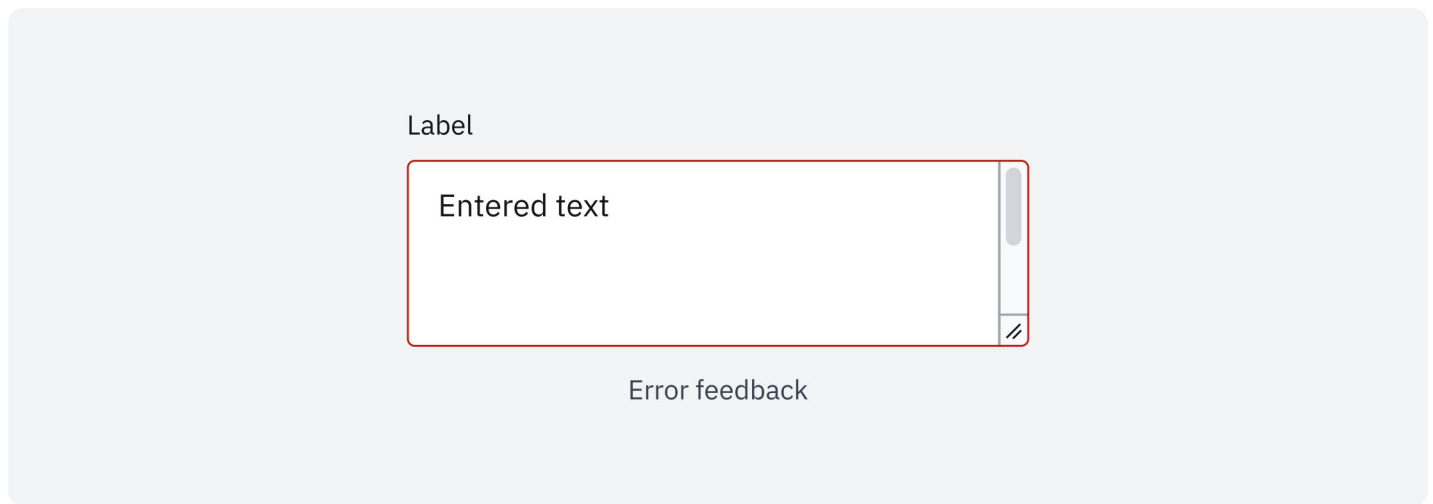
A rectangular text area with a thin grey border. It contains the text "Entered text" in a dark grey font. On the right side, there is a vertical scrollbar. At the bottom right corner, there is a small icon consisting of two parallel diagonal lines.

Filled - Darker

Error Feedback

While the input field may require validation by the system in certain cases, which necessitates a successful icon. However, errors may occur in text areas if forbidden characters are entered.

- **Error feedback:** A standard textarea field with an error icon, indicating unsuccessful input.

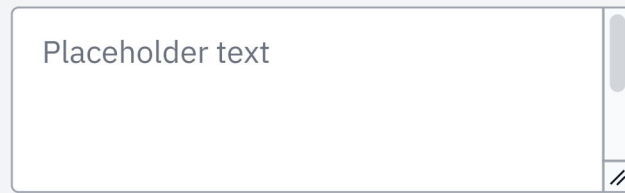


Behaviors (States)

Textarea in the design system exhibit various behaviors depending on user interaction. These behaviors include:

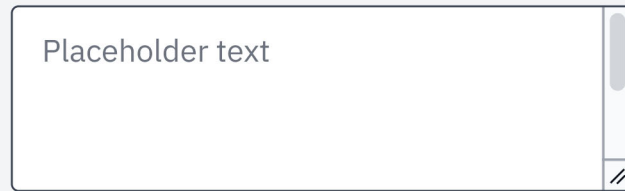
- **Default:** The initial state of a textarea when not being interacted with, appearing as a standard, editable field.
- **Hovered:** Triggered when the user hovers over the textarea without clicking, providing visual feedback.
- **Pressed:** This state is triggered when the user clicks or taps the textarea, providing feedback that the field has been successfully activated.
- **Focused:** Occurs when a textarea receives keyboard focus, typically through the "Tab" key, essential for accessibility to navigate through the interface using the keyboard.
- **Read Only:** The textarea is not editable, meaning the user can view but not modify the content.
- **Disabled:** A state indicating that the textarea is not editable or interactive, often used in user interfaces to communicate that the textarea cannot be modified under the current circumstances

Label

A rectangular text area with a light gray border. Inside, the text "Placeholder text" is displayed in a light gray font. On the right side, there is a vertical scrollbar. In the bottom right corner, there is a small icon consisting of two parallel diagonal lines.

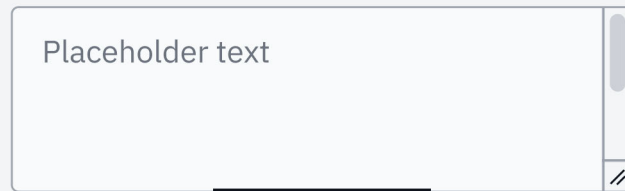
Default state

Label

A rectangular text area with a light gray border. Inside, the text "Placeholder text" is displayed in a light gray font. On the right side, there is a vertical scrollbar. In the bottom right corner, there is a small icon consisting of two parallel diagonal lines.

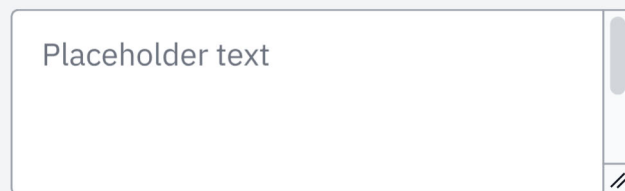
Hovered state

Label

A rectangular text area with a light gray border. Inside, the text "Placeholder text" is displayed in a light gray font. On the right side, there is a vertical scrollbar. In the bottom right corner, there is a small icon consisting of two parallel diagonal lines. A thick black horizontal line is positioned below the text area.

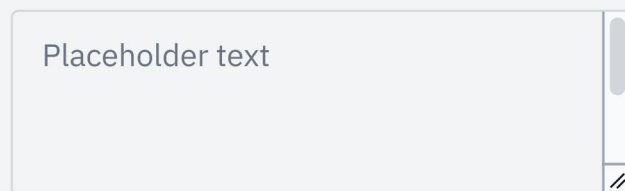
Pressed state

Label

A rectangular text area with a light gray border. Inside, the text "Placeholder text" is displayed in a light gray font. On the right side, there is a vertical scrollbar. In the bottom right corner, there is a small icon consisting of two parallel diagonal lines. The border of the text area is slightly thicker than in the default state.

Focused state

Label

A rectangular text area with a light gray border. Inside, the text "Placeholder text" is displayed in a light gray font. On the right side, there is a vertical scrollbar. In the bottom right corner, there is a small icon consisting of two parallel diagonal lines. The text area has a slightly different background shading compared to the other states.

Read only

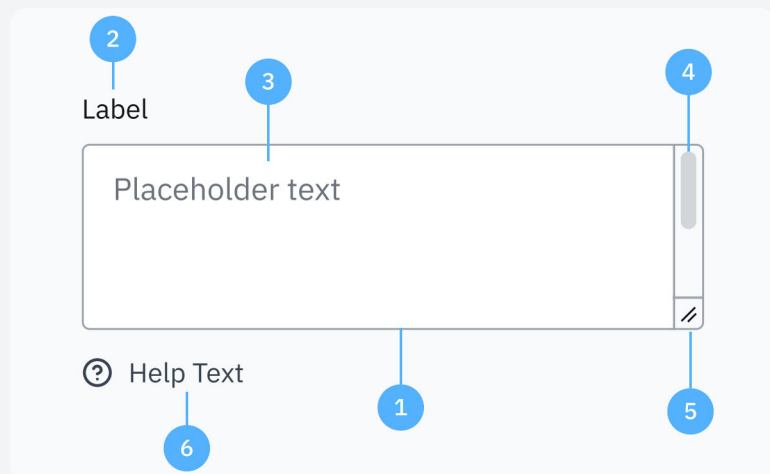
Label

Disabled state

Anatomy

Component Anatomy

- 1 Container
- 2 Textarea label
- 3 Placeholder text
- 4 Scroll bar
- 5 Resize
- 6 Helper Text



- **Container:** The editable area that holds all the elements including the text to be entered.
- **Textarea label:** Displays the title of the input.
- **Placeholder text:** Provide a hint or example of the type of information expected to be entered by the user.
- **Scrollbar:** Allows users to navigate through the text area's content.
- **Resize:** Allows users to adjust the size of the text area.
- **Helper Text:** This text provides extra guidance or context to the user. It can include details about the format of the required input, any specific requirements (like character limits), or why certain information is needed.

Accessibility

Ensuring accessibility for a textarea component is important to make sure that all users, including those with disabilities, can effectively use it to enter information. Here is an accessibility guide specifically tailored for implementing accessible textarea components:

- The **<textarea>** element is inherently accessible when used with proper labeling. Stick with this native HTML element to ensure compatibility and accessibility.
- Always use a **<label>** element to provide an accessible name for the textarea. The `for` attribute of the label must match the `id` of the textarea.
- If a visible label is not desirable for design reasons, you can still label the textarea accessibly using **aria-label** or **aria-labelledby**.
- Use **aria-describedby** to link the textarea to any additional descriptive text that helps the user understand how to fill in the data expected. This could include formatting requirements or limitations.
`<label for="bio">Biography:</label>`
`<textarea id="bio" name="bio" aria-describedby="bioHelp"></textarea>`
`<div id="bioHelp">Describe your background and experience.</div>`
- Ensure that the textarea can receive focus without the user having to click it with a mouse. It should be reachable using **Tab** navigation.
- Consider providing users with the ability to resize the textarea, especially vertically, unless there is a specific reason to restrict this functionality. This can be controlled with **CSS** using the `resize` property.
`textarea {resize: vertical;}`

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Switch

Loading

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Loading and Status](#) > [Loading](#)

Loading

The loading component indicates that a process is ongoing and that the user should wait for it to complete.

Live Demo



Visual



Code

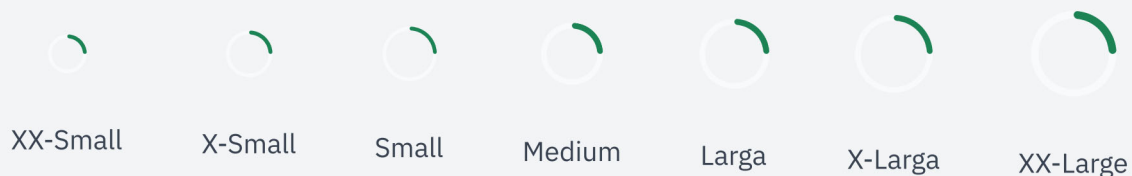


Test the preview of the loading comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

Loading is available in many fundamental formats:

- **XX-Small:** The smallest size available, suitable for areas with limited space or for a subtle indication of loading.
- **X-Small:** A bit larger than Tiny, providing a clearer visual cue.
- **Small:** A common size for loading indicators that balances visibility with discretion.
- **Medium:** A standard size that's easily noticeable without dominating the interface.
- **Large:** A more prominent indicator for situations where the loading process is a significant aspect of the user experience.
- **X-Large:** Even more conspicuous, suitable for main areas of focus where users expect to wait.
- **XX-Large:** The largest option, likely to be used for significant loading processes or as a central visual element during longer waits.



Appearance

The design of the loading in the Platforms Code is crafted to be visually engaging and immediately indicative of an in-progress state, ensuring that users are informed of the system's active processes. This intuitive visual cue allows for a clear communication of the wait time, enhancing user interaction with the application while maintaining the flow of the user experience.



Anatomy

Component Anatomy

1 Circular Progress Bar



- **Circular Progress Bar:** A circular progress visually represents the progress of a task in a circular form. It's designed to inform users about the status of ongoing processes.

Accessibility

Loading indicators, often visualized as spinners or progress bars, are common UI components that signal to users that a background process is taking place and they should wait for a task to complete. Because these components are frequently encountered, especially in web applications, ensuring their accessibility is crucial. Here's an accessibility guide specifically tailored for implementing accessible loading indicators or spinner components:

- Use the ***role="status"*** or ***role="alert"*** for loading indicators, depending on the urgency of the information.
- ***role="status"*** is typically less disruptive as it does not require the screen reader to interrupt what it's currently voicing, whereas ***role="alert"*** will immediately announce the change.
`<div role="status" aria-live="polite">Loading your content...</div>`
- Always include a text description along with the visual spinner. This text helps users who might not see the spinner understand that a loading process is ongoing.
`<div role="status" aria-live="polite">`

```

```

Loading your content...

```
</div>
```

- Avoid automatically moving focus to the loading indicator, as this can be disorienting. Instead, use **aria-live** to announce the message.
- Ensure that the loading indicator is not focusable and does not receive keyboard focus, as it is not an interactive element.
- If possible, provide information about the **duration** of the process or provide users with the ability to cancel the operation if it takes too long.
- Ensure that any animations do not flash more than three times per second to comply with **WCAG** guidelines and avoid causing seizures.
- Provide an option to reduce motion for users who are sensitive to movement. This can be achieved by respecting the **prefers-reduced-motion** media query.

```
@media (prefers-reduced-motion: reduce) {  
  .spinner {animation: none;}  
}
```

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Textarea

Breadcrumbs

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Navigational](#) > [Breadcrumbs](#)

Breadcrumbs

Breadcrumb navigation is a user interface element that enhances user orientation within a website or application. It outlines a pathway, showing how the current page fits into the overall site hierarchy. This feature improves usability by enabling users to backtrack to more general pages seamlessly.

Live Demo

 **Visual**

 Code

[Link](#) > [Link](#)

Test the preview of the breadcrumbs comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

Variants and states of breadcrumb navigation:

- **States:** Breadcrumbs have seven states for items — enabled, hover, active, focus, visited, disabled, and read-only.
- **Wrapping:** If a breadcrumb exceeds the content width, it can wrap to the next line to accommodate the overflow.
- **Trail Truncation:** By default, up to five breadcrumb items are displayed. If there are more, the trail is truncated to show only the first and last items with an ellipsis indicating additional items. Clicking the ellipsis expands the trail to show all items.

Link · Link

Link · Link · Link

Link · Link · Link · Link

Link · Link · Link · Link · Link

Link · ... · Link · Link

Breadcrumb Link

Breadcrumb Link

Breadcrumb Link

Breadcrumb Link

Appearance

- **Large Breakpoints (L to 5XL):** Use the full breadcrumb trail until it exceeds the available space, at which point it truncates. Clicking the ellipsis reveals the full breadcrumb.
- **Medium Breakpoints:** Automatically use a truncated version of the breadcrumb to fit smaller grids.
- **Small and Extra Small (Mobile):** Display only a link to the previous page and a back arrow for ultra-compact navigation.
- **Large to Extra-Large Grids:** Employ the full breadcrumb navigation until spatial constraints require truncation.
- **Medium Grids:** Implement a truncated breadcrumb version, ideal for medium-sized displays.
- **Small to Extra Small Grids:** Optimize for mobility by limiting display to a single back link and arrow, ensuring clarity and usability on the smallest screens.

Link · Link · Link · Link · Link

Link > ... · Link

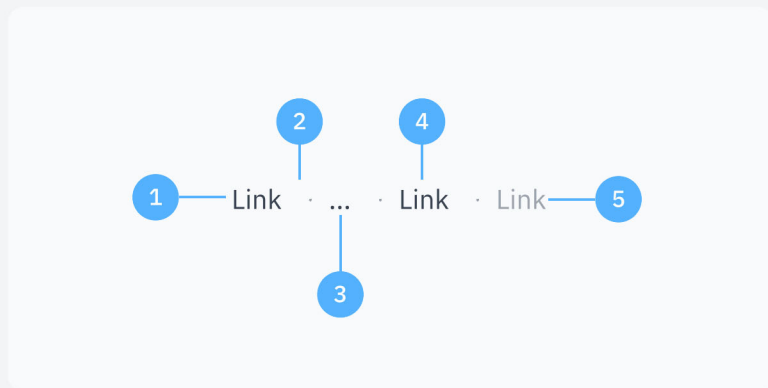
Link · Link

Link · ... · Link · Link

Anatomy

Component Anatomy

- 1 Root
- 2 Separator
- 3 Middle-overflow
- 4 Middle
- 5 Current Page



- **Root:** It is usually represented as the first item in the breadcrumb sequence and often labeled as "Home" or with the name of the website or application.
- **Separator:** Clearly distinguishes between each page.
- **Middle:** Refers to the segments between the first and last elements.
- **Middle-overflow:** Middle-overflow in breadcrumbs occurs when there are too many breadcrumb items to display within the available space, causing them to overflow or extend beyond the visible area of the breadcrumb container.
- **Current Page:** typically appears as the last item in the sequence, representing the page or location the user is currently viewing.

Accessibility

While breadcrumbs inherently support navigational clarity, specific accessibility features are essential to ensure they are fully usable by everyone, including individuals with disabilities. Adhering to accessibility standards when implementing breadcrumbs enhances user interaction and conforms to web accessibility guidelines.

- Implement breadcrumbs as a list of links to reflect their structure semantically. This helps users and assistive technologies understand the layout and function of the breadcrumbs.

```

<nav aria-label="Breadcrumb">
<ul>
  <li><a href="#">Home</a></li>
  <li><a href="#">Category</a></li>
  <li aria-current="page">Current Page</li>
</ul>
</nav>
  
```


</nav>

- Use ***aria-label*** on the navigation element to define the purpose of the navigation block (e.g., Breadcrumb).
- Use ***aria-current="page"*** to indicate the current page in the breadcrumb trail. This attribute helps assistive technology users understand which page is active.
- Ensure all links in the breadcrumbs are accessible, which includes having discernible text and being focusable.
- All links in the breadcrumb should be navigable using the ***Tab*** key .
- Provide visual feedback for focus, hover, and active states to improve interaction feedback for users who rely on the keyboard or mouse.
- Use visual separators between breadcrumb items. If a custom icon is used as a separator, ensure it has the ***aria-hidden="true"*** attribute to prevent it from being read by screen readers, as it is purely decorative.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Loading

Menu

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Navigational](#) > [Menu](#)

Menu

A menu UI element shows a list of options on a temporary surface, triggered by user interactions with buttons or controls. It organizes menu items, often grouped for better structure, to enhance navigation and functionality.

Live Demo



Visual



Code

Test the preview of the menu comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

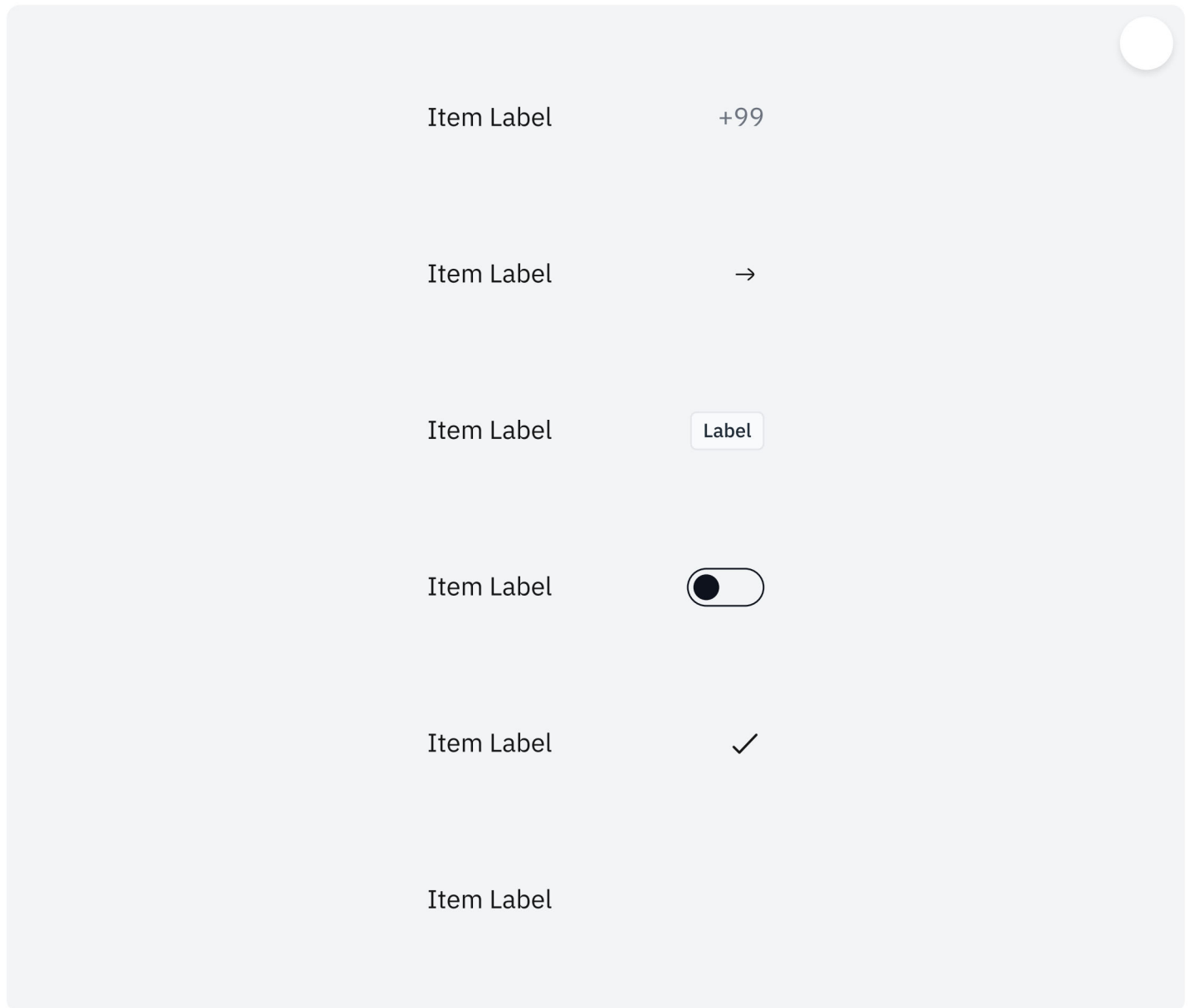
Variants

Menu list item is available in the Platforms Code in many fundamental formats.

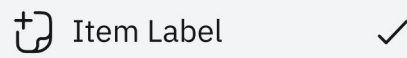
Menu list item

Each menu item has a trailing elements which either present data or allow user to interact with such as tags, controllers or buttons.

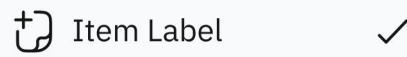
- **Text:** This item displays external text to provide additional information to the user, enhancing their understanding of the option.
- **Button:** This item allows user to make an action when clicking on the option.
- **Tag:** This item enables the user to include supplementary information, often in the form of labels or categories, for better context or classification.
- **Switch:** This item provides a toggle control for the user to enable or disable a feature, such as turning on notifications or activating a dark mode.
- **Icon:** This item displays a visual indicator, helping the user quickly recognize the function or status of the option through graphical representation.
- **None:** This is the standard menu item type, which contains only the menu item text without any additional trailing elements, offering a simple selection option.



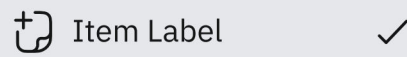
- **Default:** This is the initial state of the menu item when it is not being interacted with by the user.
- **Hovered:** This state is triggered when the user points their cursor over the menu item.
- **Pressed:** This state occurs when the user clicks or taps the menu item.
- **Focused:** Focused state is important for accessibility, indicating that the item has keyboard focus (such as when navigating with tab keys).
- **Disabled:** This state is used when the action represents cannot be performed at the current time.



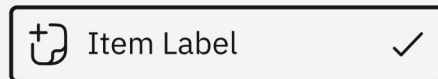
Default



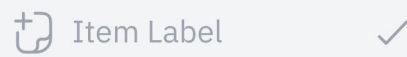
Hovered



Pressed



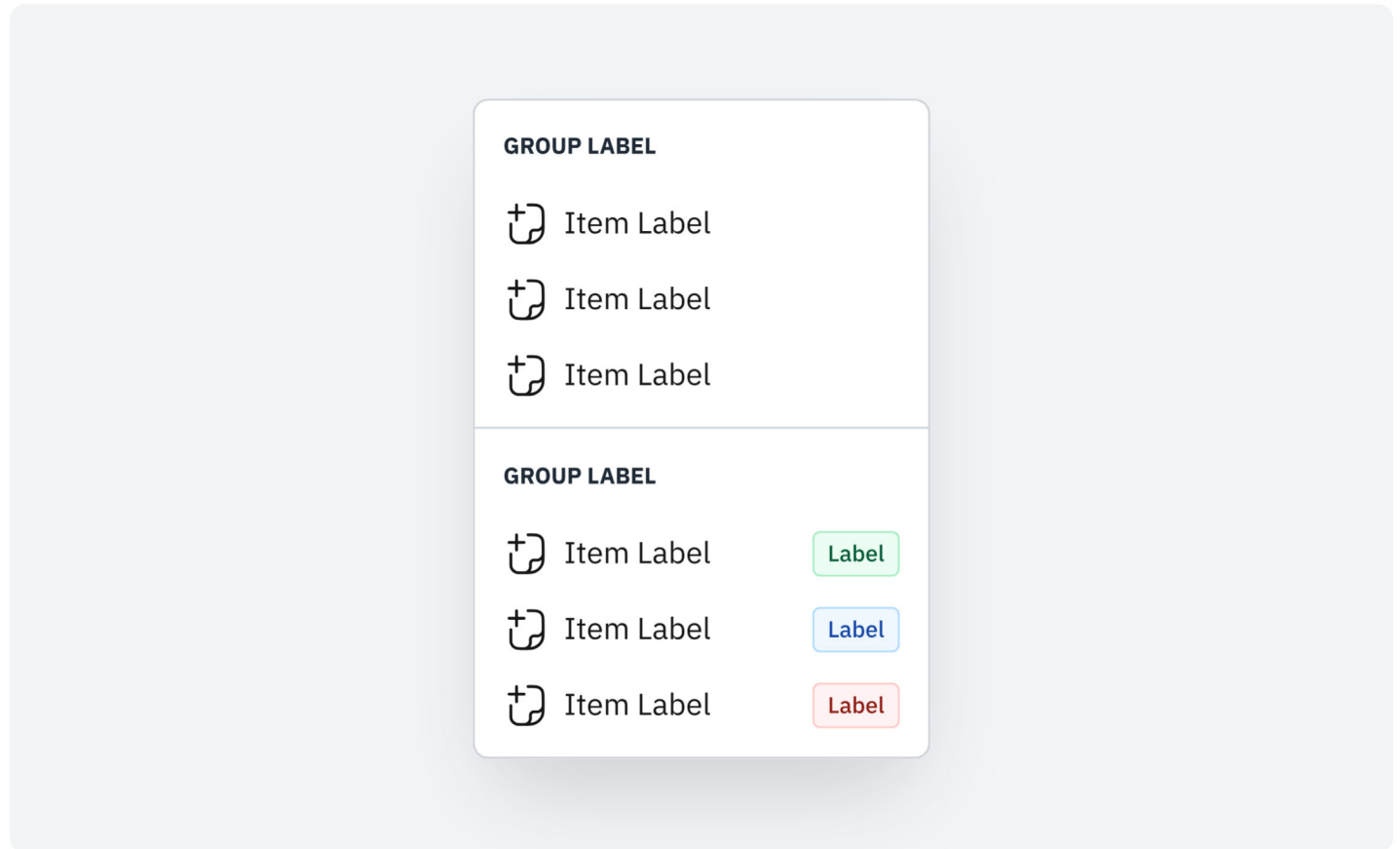
Focused



Disabled

Appearance

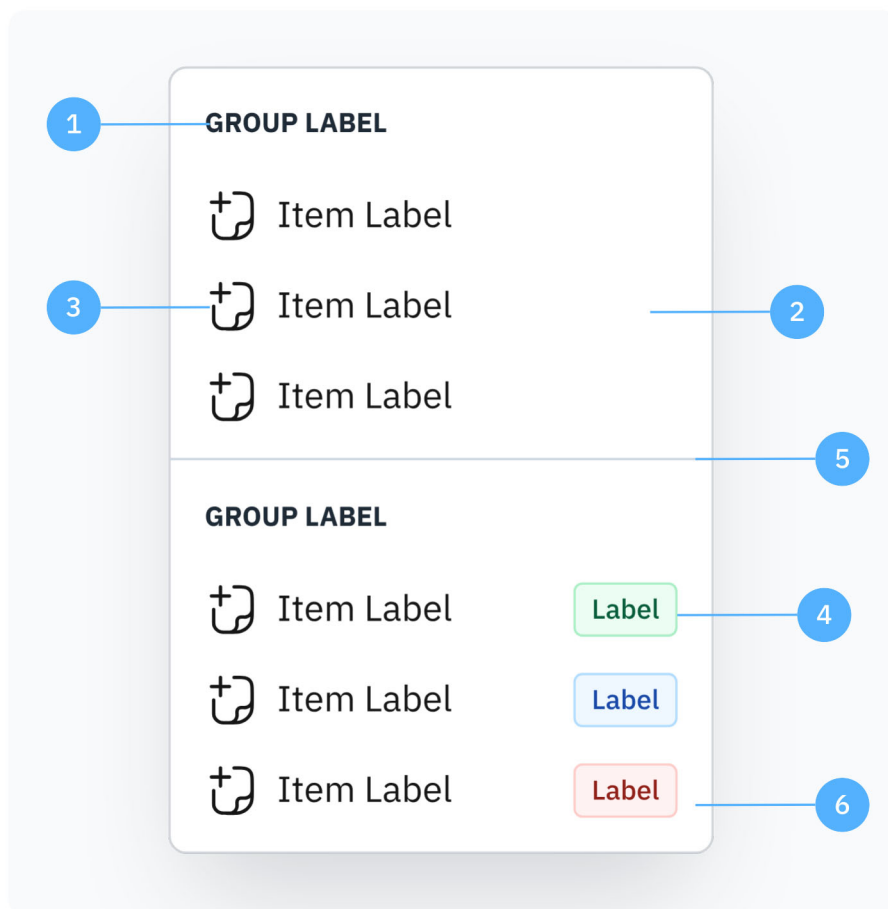
Menu list item is available in the Platforms Code in many fundamental formats.



Anatomy

Component Anatomy

- 1 Group label
- 2 Item
- 3 Item lead icon
- 4 Item trailing element
- 5 Divider
- 6 Menu container



- **Group label:** A heading that categorizes and labels a group of related menu items, helping users quickly identify and navigate through different sections.
- **Item:** An individual item within a grouped menu, usually related in function or context to other items in the same group, providing a structured organization.
- **Item lead icon:** An icon placed at the beginning of a menu item that visually represents the item's function or category, aiding in quick recognition.
- **Item trailing element:** An additional element placed at the end of a menu item, such as a button, switch, or tag, offering extra functionality or information.
- **Divider:** A horizontal line used to separate different groups within a menu, enhancing readability and organization.
- **Container:** The overall framework that holds and displays all menu items, ensuring a cohesive and organized presentation.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Breadcrumbs

[← Previous](#)

Pagination

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Navigational](#) > [Pagination](#)

Pagination

Pagination is used to divide large sets of content into multiple pages, allowing users to navigate through them sequentially.

Live Demo

Visual

Code

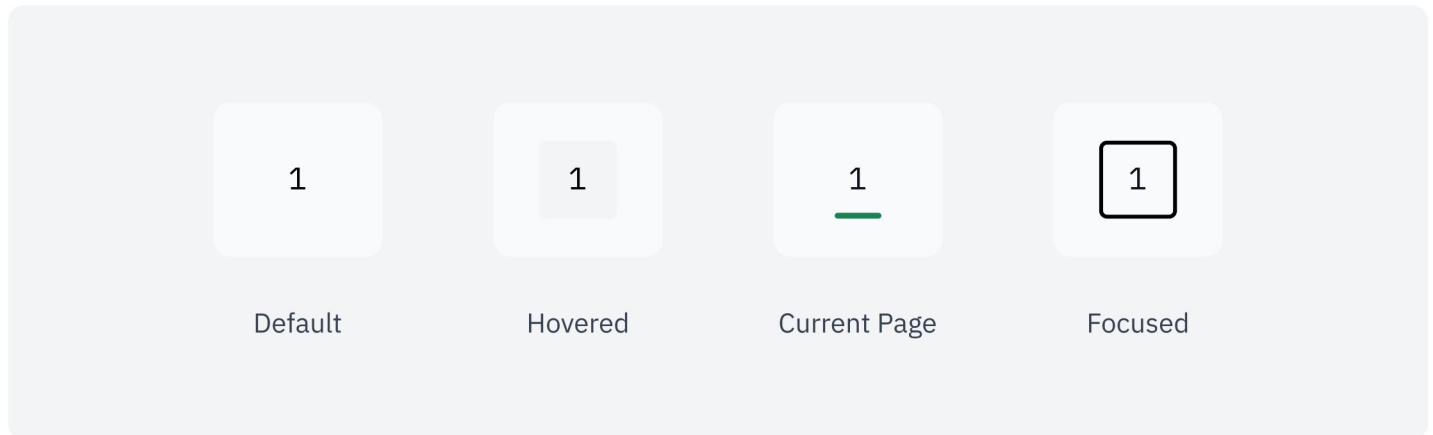
< 1 2 3 4 5 ... 999 >

Test the preview of the pagination comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

Pagination is available in four fundamental formats:

- **Default:** This is the standard appearance of the pagination links when they are not being interacted with.
- **Hovered:** The hover state activates when the user positions the cursor over a pagination link.
- **Current Page:** This state highlights the page number that the user is currently viewing.
- **Focused:** The focused state comes into play when a pagination link receives focus, typically from keyboard navigation (such as using the tab key).



Appearance

The design of the pagination component is visually intuitive and easily distinguishable, ensuring that users can quickly recognize their current location within the pages and navigate effortlessly through the available content.

< 1 2 999 >

< 1 2 3 ... 999 >

< 1 2 3 ... 999 >

< 1 2 999 >

4 ✓

5

6

7

8

9

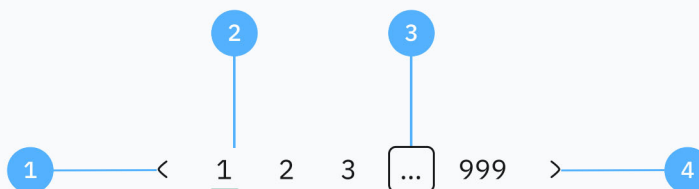
10

11

Anatomy

Component Anatomy

- 1 Previous Navigation
- 2 Active Page
- 3 Page Items
- 4 Next Navigation



- **Previous Navigation:** Allows users to navigate to the previous page.
- **Active Page:** This element highlights the current page that the user is on.
- **Page Items:** These include the numbers or labels for each page that users can click to navigate directly to a specific page.
- **Next Navigation:** Similar to the previous navigation but designed to take users to the next page.

Accessibility

Ensuring that pagination is accessible is important so that all users, including those who use assistive technologies, can navigate through content efficiently. Here's a comprehensive accessibility guide for creating accessible pagination components:

- Encapsulate pagination links within `<ul>` (unordered list) or `<ol>` (ordered list) to indicate the relationship between the items. Each page link should be wrapped in an `<li>` (list item).

`<nav aria-label="Pagination">`

`<ul>`

`<li><a href="#">Prev</a></li>`

`<li><a href="#" aria-current="page">1</a></li>`

`<li><a href="#">2</a></li>`

`<li><a href="#">next</a></li>`

</nav>

- Use ***aria-label="Pagination"*** on the navigation element to describe the type of navigation provided.
- Use ***aria-current="page"*** to indicate which page link is currently active. This helps screen readers announce the current page.
- Use ***<a href>*** for links that lead to other pages.
- If the pagination is performed dynamically (e.g., via AJAX), use ***<button>*** elements and manage the content update accessibly.
- Ensure that focus can be moved between the different pagination links using the ***Tap*** key.
- Allow pagination links to be activated using the ***Enter*** key when they are focused.
- Supporting additional keyboard shortcuts, like using arrow keys to move between pages if the pagination component has many pages.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Menu

← Previous

Tabs

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [Navigational](#) > [Tabs](#)

Tabs

A tablist is a set of tabs that allows for easy navigation between related information categories, removing the need to switch between different pages.

Live Demo



Visual



Code

 First Tab

 Second Tab

 Third Tab

 Forth Tab

 Fifth Tab



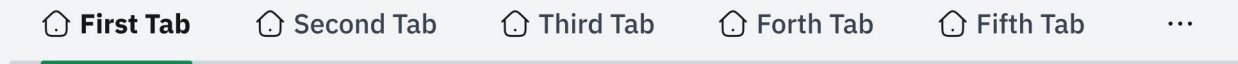
Test the preview of the tabs comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Appearance

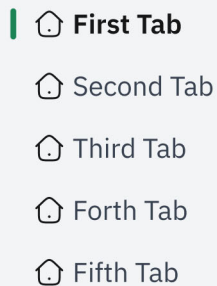
By default, tablists are horizontally aligned, but they offer the flexibility to be configured for vertical display. In a horizontal tablist, tabs are confined by the container width and do not scroll or wrap to the next line. If there's a need to show more tabs, it is recommended to incorporate an overflow menu button for accommodating additional tabs.

Variants

Horizontal Tablist



Vertical Tablist



Behaviors (States)

Tabs in the Platforms code exhibit various behaviors when they are selected and not selected. Tabs behaviors state are:

- **Default:** The default state of a tab when it is not being interacted with. it appears as a standard, clickable tab.
- **Hovered:** Occurs when the user moves the cursor over the tab without clicking.

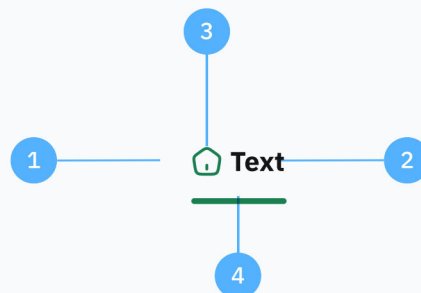
- **Pressed:** This state is triggered when the user clicks or taps the tab. It provides feedback that the tab has been successfully activated.
- **Focused:** Occurs when a tab receives keyboard focus, usually through the "Tab" key. Important for accessibility to ensure users can navigate through the interface using the keyboard.
- **Disabled:** The disabled state of a tab is a condition where the tab is not clickable or interactive. This state is often used in user interfaces to communicate that the tab cannot be activated under the current circumstances.

- | 🏠 **First Tab**
- 🏠 Second Tab
- 🏠 Third Tab
- 🏠 Forth Tab
- 🏠 Fifth Tab

Anatomy

Component Anatomy

- 1 Container
- 2 Tab title
- 3 Icon
- 4 Selection indicator

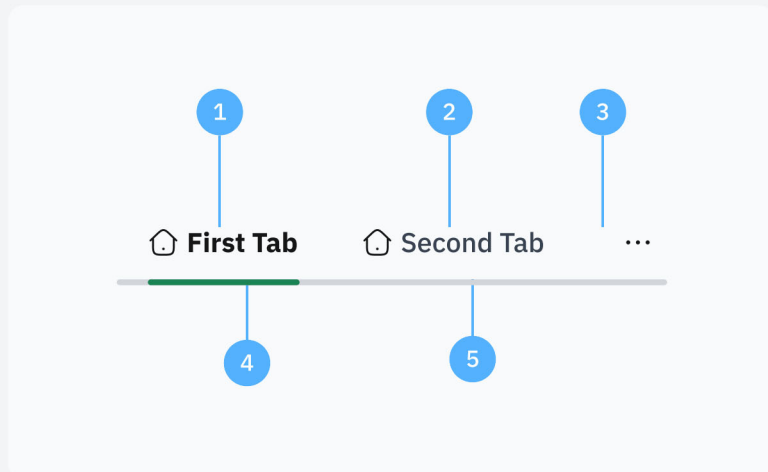


- **Container:** The area that holds all the elements.
- **Tab title:** Displays the title of the tab.

- **Icon:** Provides visual representation of the tab name or content.
- **Selection indicator:** Visual element indicating the currently selected tab.

Component Anatomy

- 1 First tab
- 2 Second tab
- 3 More button
- 4 Selection indicator
- 5 Divider



First Tab: The first tab in the tab bar, often representing the initial or default view.

Second Tab: It represents another view that users can switch to.

More button: A button represented by three dots (ellipsis) that indicates more options or additional tabs that are not visible in the current view. It provides a way to access overflow tabs or additional actions.

Selection indicator: Visual element indicating the currently selected tab.

Divider: A visual separator that distinguishes different sections of the tab.

Accessibility

This guide outlines the best practices for designing and implementing accessible tabs within CodePlatform. Ensuring accessibility in tabs is crucial for providing an inclusive user experience, allowing users of all abilities to navigate and interact with our interfaces effectively.

Keyboard Navigation

Arrow Keys: Navigate between tabs.

home/End keys: Jump to the first or last tab.

Tab Key: Navigate into and out of the tab list.

ARIA Roles and Attributes

Roles: Use role="tablist", role="tab", and role="tabpanel" for the tab structure.

Attributes: Implement aria-controls, aria-selected, aria-labelledby, and manage focus with tabindex.

Visual Design

Focus Indicators: Visible focus indicators are essential.

Contrast and Colors: Ensure high contrast for readability and active tab distinction.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Pagination

[← Previous](#)

Tags

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [Search and Filters](#) > [Tags](#)

Tags

Tags visually convey important information or actions related to components. They use brief text, color, and icons to offer quick insights or alerts to users. Positioned near relevant content, tags grab attention and aid comprehension.

Live Demo



Visual



Code

☆ Label

Test the preview of the tags comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

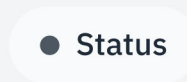
Variants

Tags are available in tow variants:

- **Standard Tag:** Standard tags are visual elements conveying primary actions or features in a user interface, often using concise text or icons for easy recognition.
- **Status Tags:** Used to visually indicate the status or condition of an element, task, or process. These tags provide quick and easily understandable indicators to users, enabling them to grasp important information swiftly.



Standard Tag



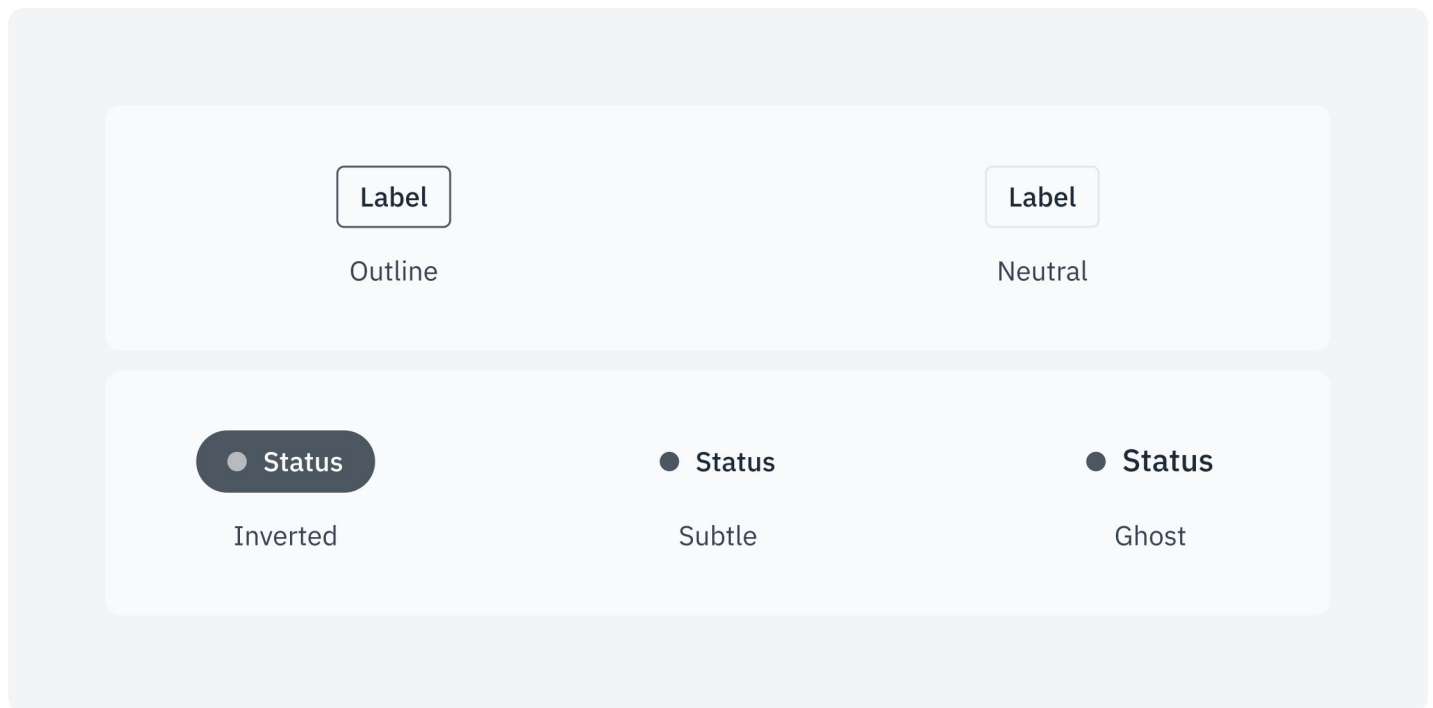
Status Tags

Appearance

The Platforms Code Design system includes two types of tags: standard tags and status tags. Standard tags are used for primary actions, while status tags are used to visually indicate the status or condition of an item, task, or process. This array of tags caters to a variety of user interactions.

Tags are available in five styles:

- **Outline:** The outline offers a clean and modern look for tags while maintaining clarity and readability within the user interface.
- **Neutral:** The neutral helps avoid overwhelming the user with visual noise while still providing important information or status indications.
- **Inverted:** The inverted is often used to indicate important or critical status information, effectively drawing the user's attention.
- **Subtle:** The subtle is characterized by soft hues or subtle color variations, which provide a gentle visual cue without overwhelming the user interface.
- **Ghost:** This style creates a subtle and minimalist appearance, allowing the tag to blend seamlessly with the surrounding content while still providing important status information. Ghost tags are often used for secondary or less prominent status indicators, where a more subdued visual presentation is desired.

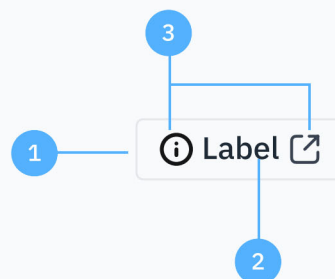


Within the Platforms Code, tags are designed to convey two distinct levels of display styles, each denoting a specific degree of importance. Standard tags are used for important items that need to stand out, While Status tags are used for less critical items. This important distinction in prominence helps design a system that guides users clearly and accurately during system use.

Anatomy

Component Anatomy

- 1 Container
- 2 Text
- 3 Leading or trailing Icon

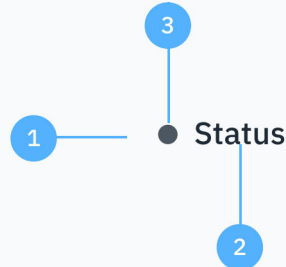


- **Container:** The area that holds all the elements.

- **Text:** Displays the label of the tag.
- **Leading or Trailing Icon:** Refers to the placement of the icon relative to the text within the tag.

Component Anatomy

- 1 Container
- 2 Text
- 3 Status Indicator



- **Container:** The area that holds all the elements.
- **Text:** Displays the status of the tag.
- **Status Indicator:** Refers to the visual element that communicates the current state or condition associated with a particular component or item.

Tips to effectively use tags into your designs

D

Gr
re
sta
to
cre
co
ta
clu
th
ali
wi
sp
ful
Th
aic
us
in
qu
ide
an
ac
re
ac

● Pending

● Done

● On hold

● In progress

✗ Avoid

Mixing unrelated statuses together, by creating heterogeneous tag clusters that align with unspecified functionalities. This can confuse users and create false assumptions about the statuses.

D

Us
we
th
de
a
sta
or
sta
th
wi
aic
th
in
ma
inf
de
or
tal
ap
ac

✓ You have successfully logged

⚠ Warning: please update your...

⚠ Something went wrong...

✗ Avoid

avoid labels that will truncate. Truncated labels can cause confusion for the users.



✓ Success



⚠ Warning



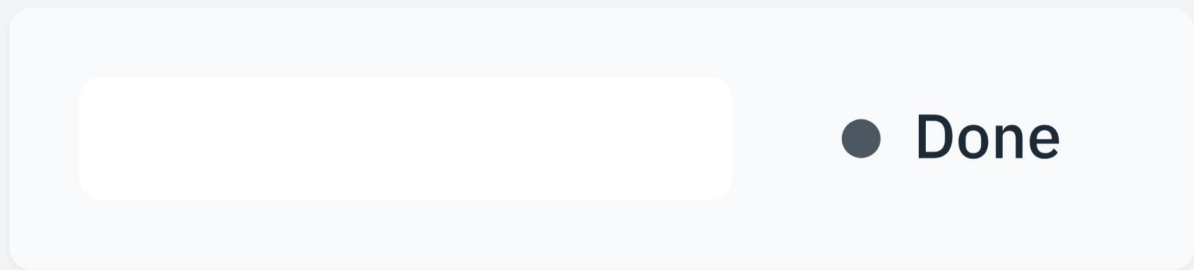
⚠ Error

✓ **Do**

Use words that describe a state or status. this will aiding them in making informed decisions or taking appropriate actions.

D

Do
all
tag
to
hu
th
lal
Th
wi
he
wi
cre
a
me
vis
co
an
or
int



✗ Avoid

Avoid stretch tags containers. Stretching the containers can distort the visual appearance of the tag.

Accessibility

Tags, often called "labels" or "chips" in user interface design, are small elements that categorize content, highlight attributes, or allow users to select or remove items quickly. Designing accessible tags is important to ensure that all users, including those who rely on assistive technologies, can interact with them effectively. Here's a comprehensive accessibility guide for creating tags:

- Use appropriate HTML elements to mark up the tags. If a tag is interactive (e.g., it can be removed or clicked), consider using **<button>** elements.
- Ensure that the role, state, and functionality of the tag are clearly conveyed to screen reader users.

- Non-interactive tags can simply be **** elements if they only display information.
- Interactive tags should be marked up as **<button>** when they perform an action (like removing a tag). This automatically provides keyboard accessibility and the correct semantic role.
- Use **aria-label** to provide accessible names for interactive elements, especially if the visual text is not sufficient to describe the action (e.g., a close icon).
- Ensure that all interactive tags can receive focus, and manage focus order to be logical and intuitive.
- Interactive tags should be operable with the keyboard, typically with the Enter and Space keys to activate the button actions.
- Provide a way to navigate between tags using arrow keys if they are part of a larger collection (like in a multi-select input field).
- Provide clear visual feedback for focus, hover, and active states to help users determine which element is currently selectable or focused
- Ensure tags are large enough to be easily clicked or tapped, and provide adequate spacing between tags for easy interaction.
- Ensure sufficient contrast between the text (or icons) and the background of the tag for readability.
- For tags that can be removed, consider adding a confirmation step or an undo option for accidental removals, especially if the removal action has significant consequences.
- When tags are added or removed, ensure that these changes are announced by screen readers using ARIA live regions.

```
<div aria-live="polite" aria-atomic="true" class="visually-hidden">
```

```
Tag JavaScript removed.
```

```
</div>
```

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Tabs

Navigation Header

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [UI Shell](#) > [Navigation Header](#)

Navigation Header

A Navigation Header is a critical component in design system that guides users through the structure of a website. It typically appears at the top of each page.

Live Demo



Visual



Code



Home ▾

Services ▾



Sign
up

Log
in

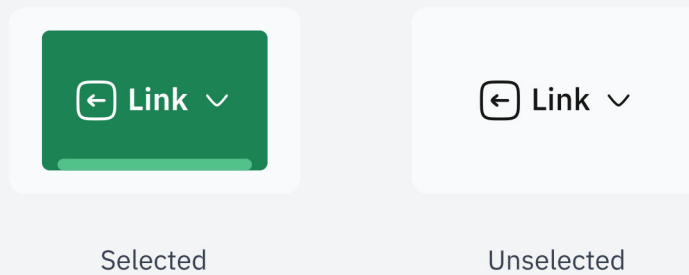
Test the preview of the navigation header comprehensive exploration, including all variants and features, our full demo will soon be available on [Storybook](#).

Variants

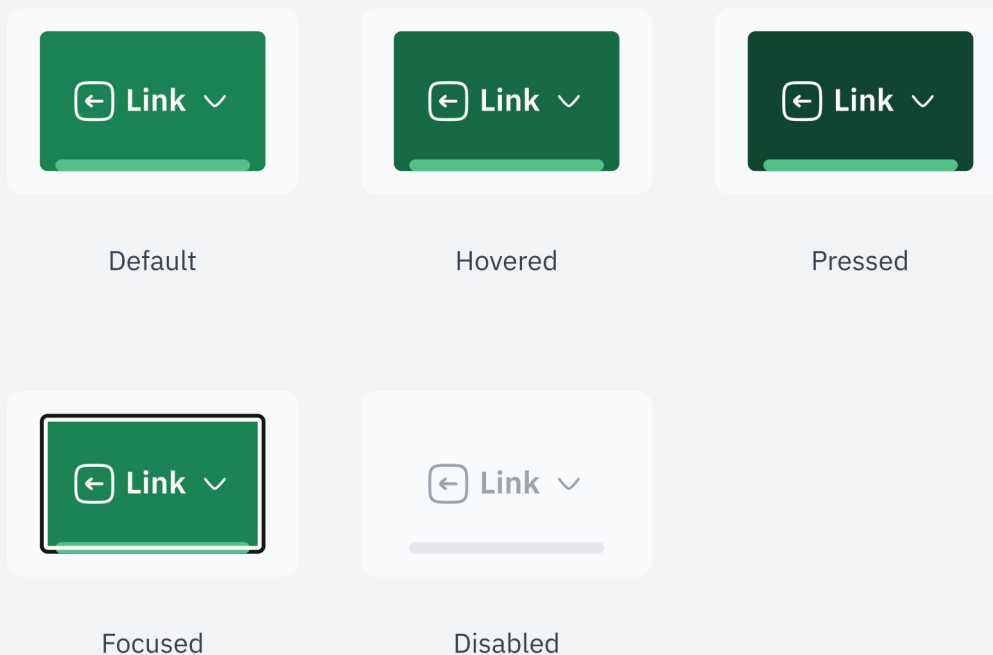
Header is available in the Platforms Code in many fundamental formats:

Header Menu Item

- **Selected:** This variant indicates the page or section that the user is currently viewing. It helps users understand their current location.
- **Unselected:** This variant represents the navigation options that the user has not selected.

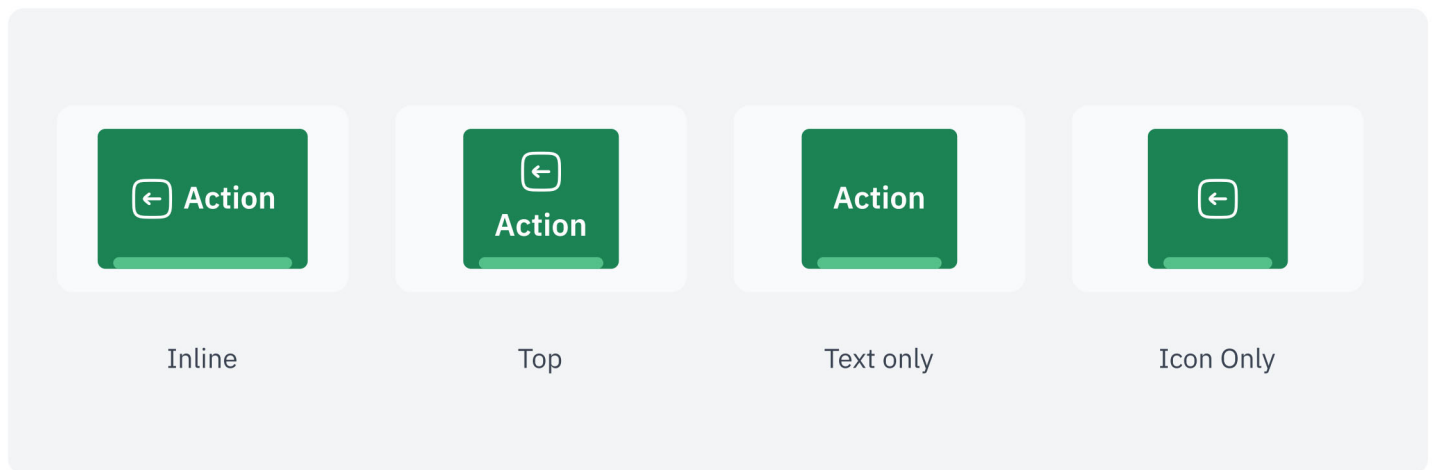


- **Default:** This is the initial state of the menu item when it is not being interacted with by the user.
- **Hovered:** This state is triggered when the user points their cursor over the menu item.
- **Pressed:** This state occurs when the user clicks or taps the menu item.
- **Focused:** Focused state is important for accessibility, indicating that the item has keyboard focus (such as when navigating with tab keys).
- **Disabled:** A disabled state is used when a menu item is not available for interaction.

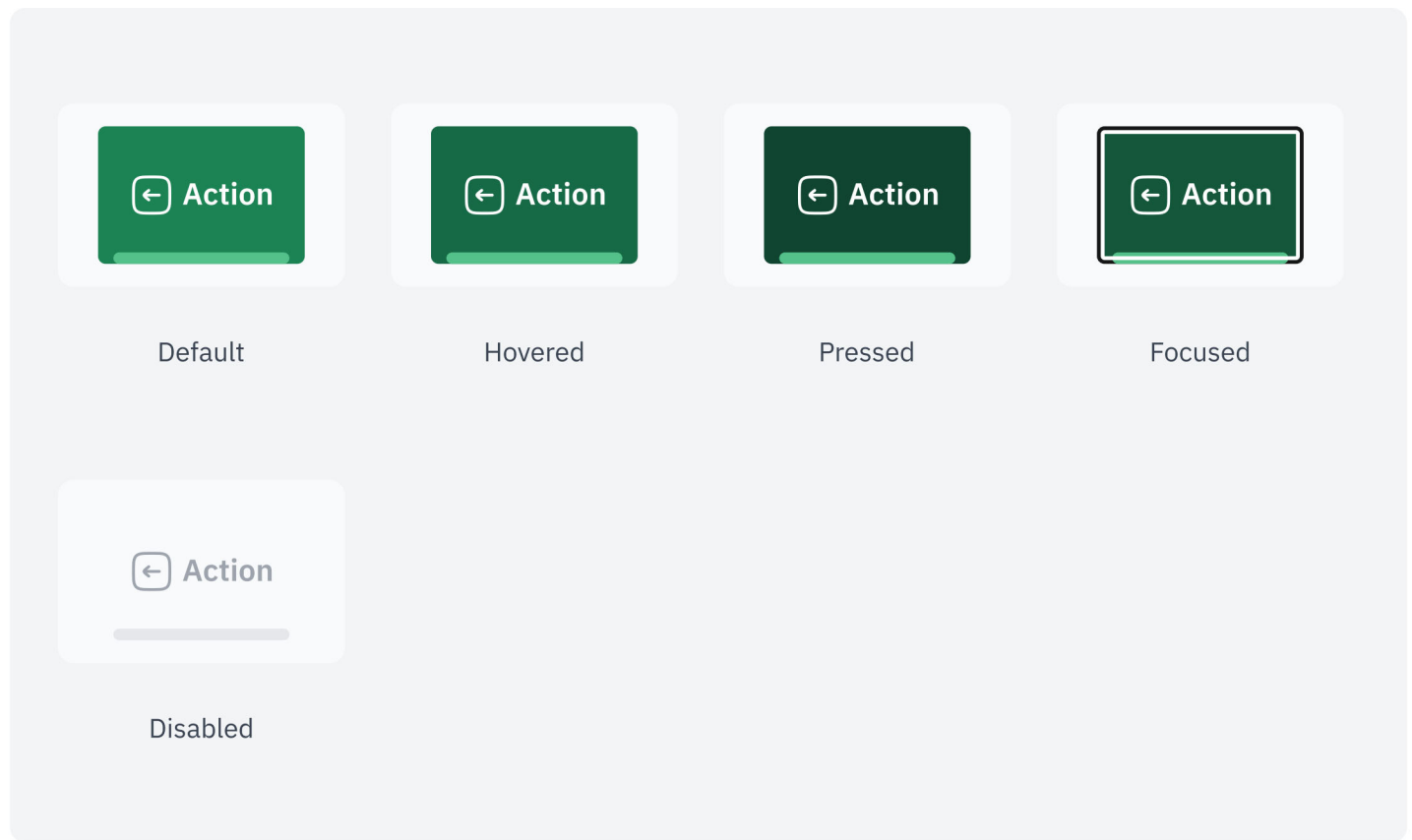


Header Action

- **Inline:** The icon and text are on the same line, suggesting a compact and efficient use of space.
- **Top:** The icon is placed above the text, which creates a more prominent visual cue for the action.
- **Text only:** This variant only shows text, which might be used in situations where the action is already well understood.
- **Icon Only:** Only the icon is displayed without text. This design is common for familiar actions (like a search magnifying glass or settings gear).

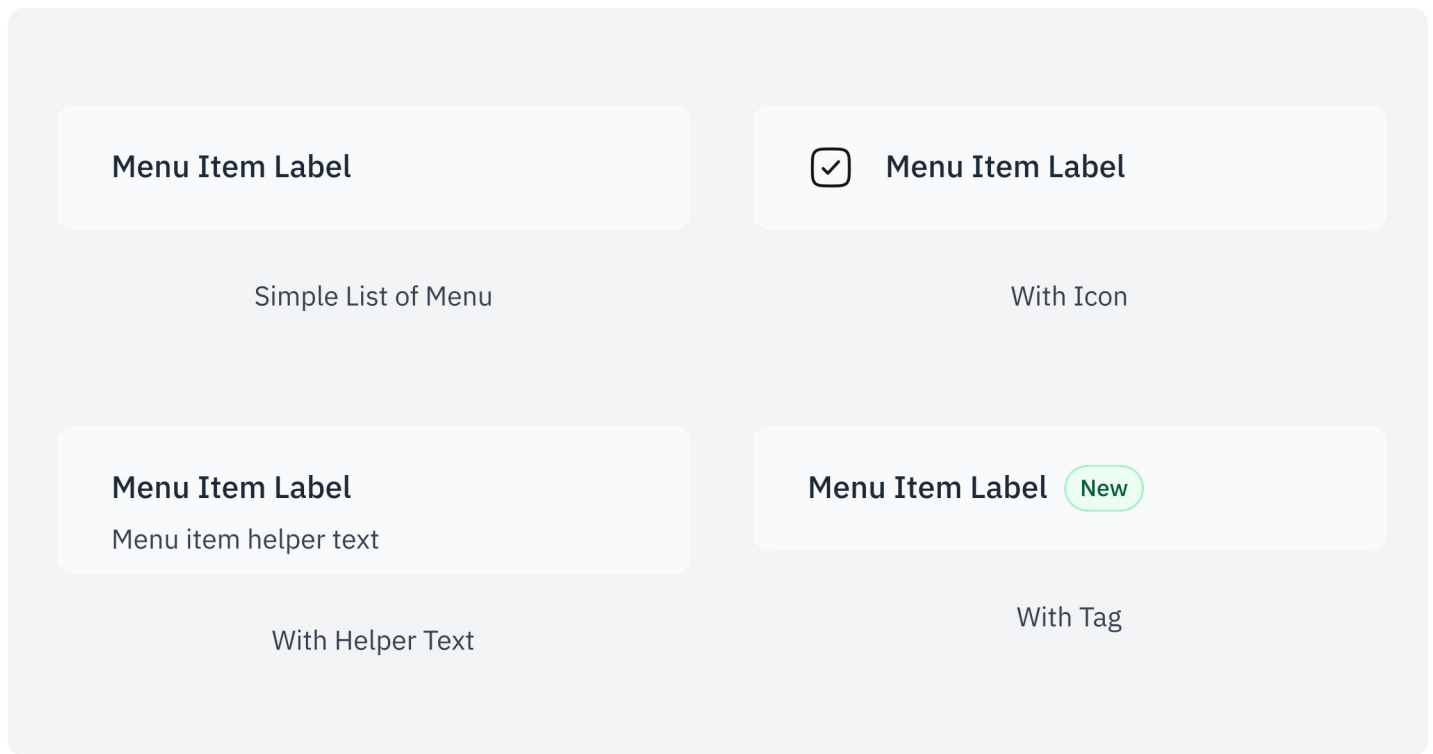


- **Default:** This state is visible when the user has not yet interacted with the header action.
- **Hovered:** This state is activated when a user moves their pointer over the header action, indicating they can click it.
- **Pressed:** This state appears when the user clicks or taps the header action and holds it down.
- **Focused:** This state is crucial for accessibility, as it indicates that the user has tabbed to this header action using a keyboard or other input method.
- **Disabled:** This state is used when the action represents cannot be performed at the current time.

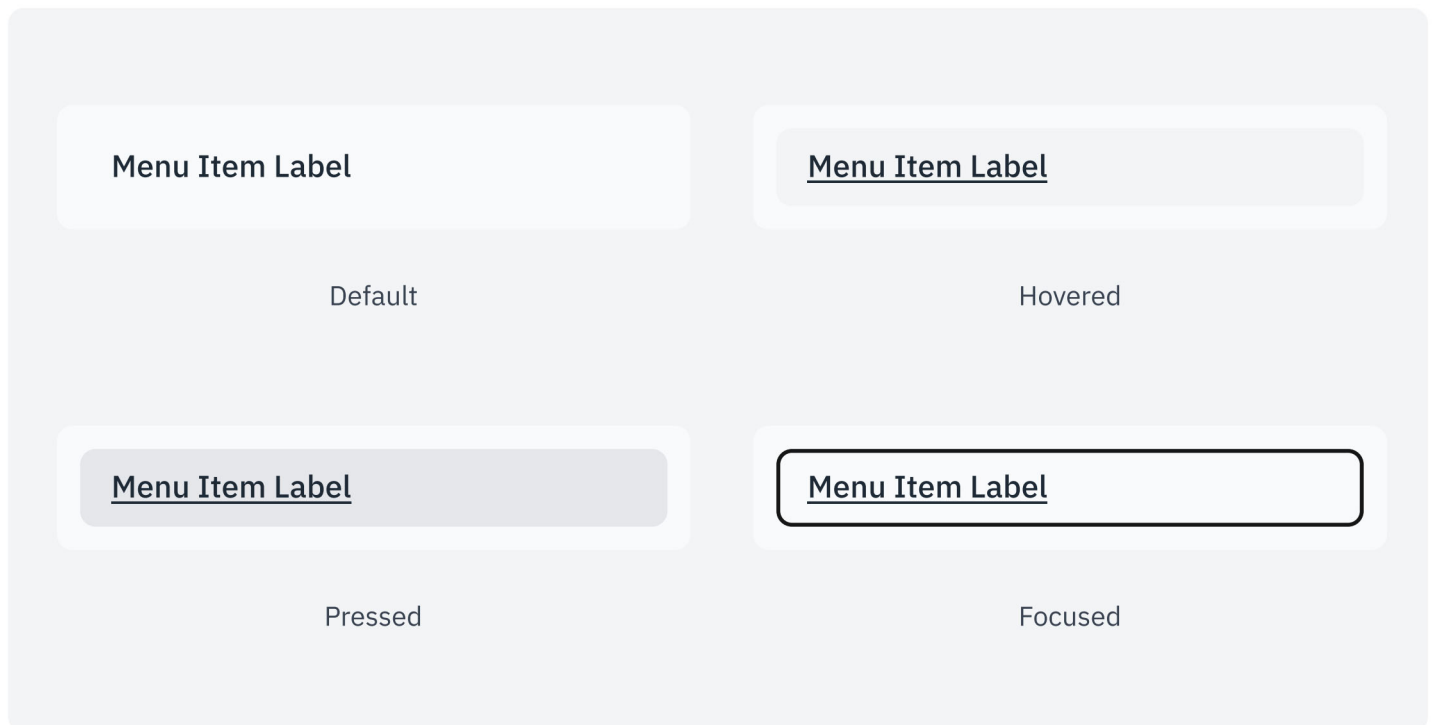


Header Sub-Menu Item

- **Simple List of Menu:** This variant shows a straightforward list item style, with just the text label for the menu item.
- **With Icon:** This variant includes an icon next to the menu item label, which can help users quickly identify the function or category of the menu item.
- **With Helper Text:** This variant provides additional descriptive text beneath the menu item label, offering context or assistance to understand the menu item's purpose.
- **With Tag:** The Tag serves as an attention-catching feature that can signify status, promote engagement, or provide a quick visual cue to the users about something noteworthy regarding the menu item.

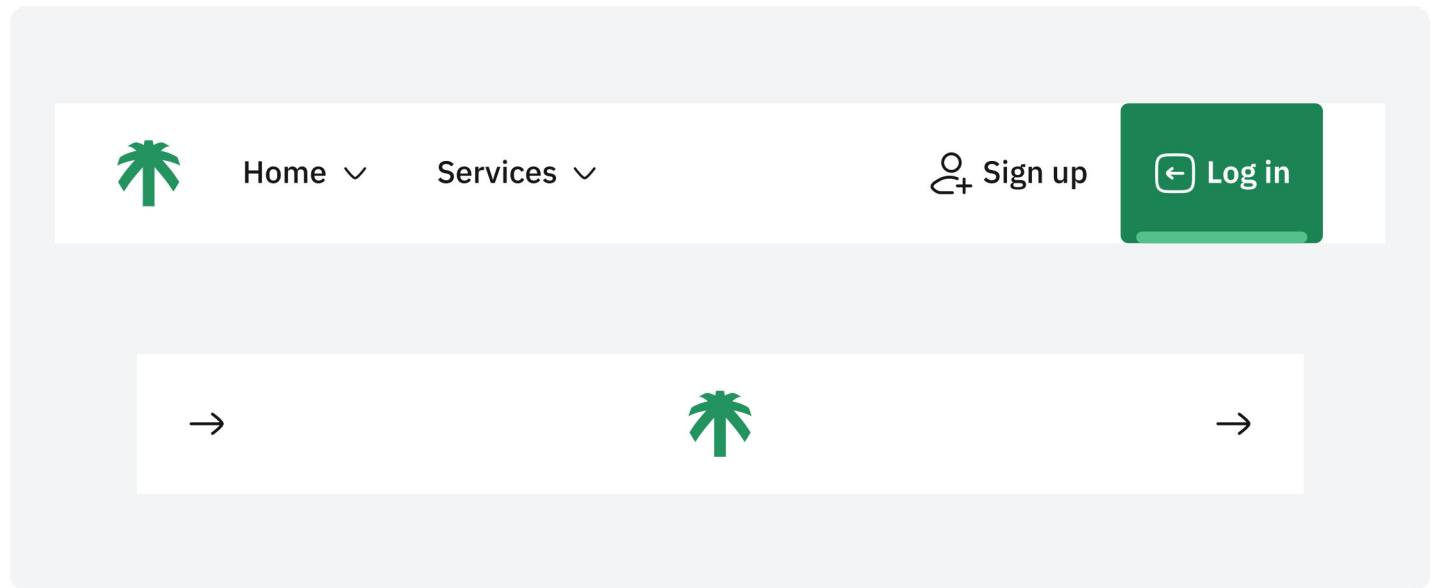


- **Default:** This is the standard state of the menu item when it is not being interacted with.
- **Hovered:** This state is displayed when the user moves their cursor over the menu item.
- **Pressed:** This variant is shown when the user clicks or taps on the menu item.
- **Focused:** The focused state is highlighted when the menu item has keyboard focus, which can happen when tabbing through elements with the keyboard.



Appearance

The header design is crafted to be distinctive and intuitive, enabling users to quickly recognize and access the primary functionalities and sections. This approach ensures a seamless and efficient user experience, promoting effortless navigation and interaction.



Anatomy

- **Brand Logo:** The image on the left likely represents a company or website logo that helps to establish brand identity.
- **Navigation Links:** Navigation links are interactive text elements that allow users to navigate from one page to another within a website.
- **Action:** This could be indicators for user actions that can be taken or buttons for additional functions.

Accessibility

Designing accessibility into the design of a header involves several key attributes and practices that ensure the header is usable by all individuals, including those with disabilities. Here are some essential considerations:

Key Accessibility Attributes and Practices:

ARIA Roles:

Apply ARIA (Accessible Rich Internet Applications) roles, states, and properties where appropriate to enhance the semantic information of elements, especially for components that do not have native HTML equivalents. For instance, use **role="banner"** for the main header or **role="navigation"** for the navigation region.

Keyboard Navigation:

Ensure that all interactive elements in the header can be accessed using a keyboard. This typically involves using the **Tab** key to move through elements and **enter or space to activate links or buttons**.

Focus Indicators:

Provide clear visual focus indicators for interactive elements when they are focused, such as when using **:focus** CSS pseudo-class. This helps users with visual and cognitive disabilities track where they are on the page.

Skip Navigation Links:

Offer a "Skip to Content" link at the beginning of the header so that keyboard and screen reader users can bypass the navigation menu and go straight to the main content.

Contrast and Colors:

Ensure sufficient contrast between text and background colors to aid users with low vision. Adhere to WCAG (Web Content Accessibility Guidelines) standards for contrast ratios.

Labeling:

Interactive elements like search boxes and menu toggles should have descriptive labels or **aria-label** attributes to inform users of their function.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Tags

[← Previous](#)

Navigation Drawer

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [UI Shell](#) > [Navigation Drawer](#)

Navigation Drawer

A navigation drawer typically appears as a panel from the side of the screen, revealing a set of navigation options such as links to different sections.

Live Demo



Visual



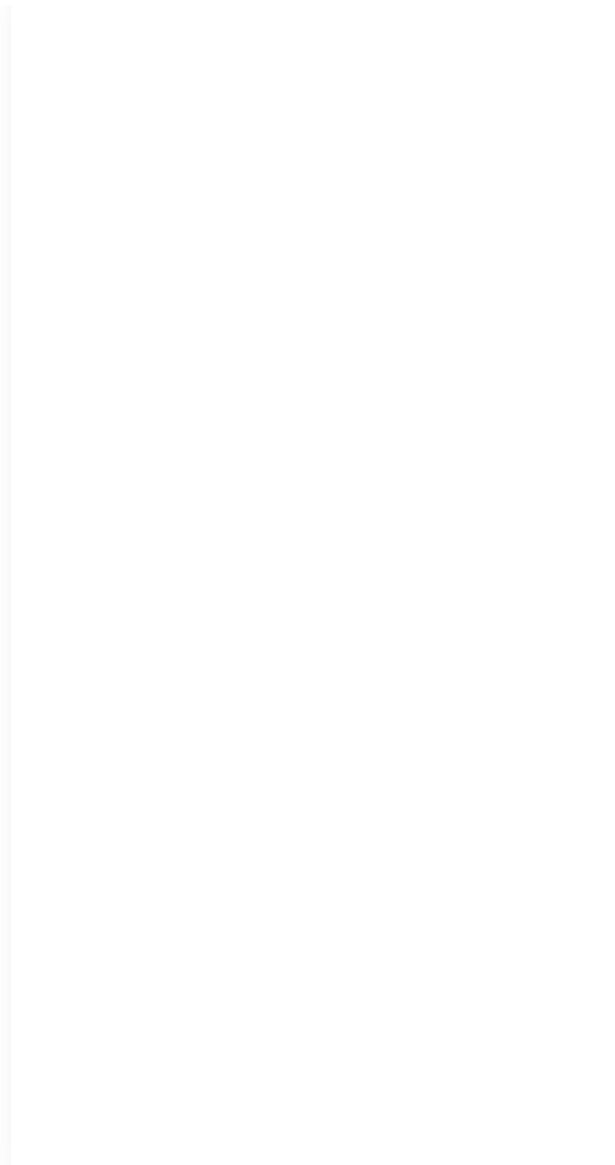
Code

☐ Home



☒ Components



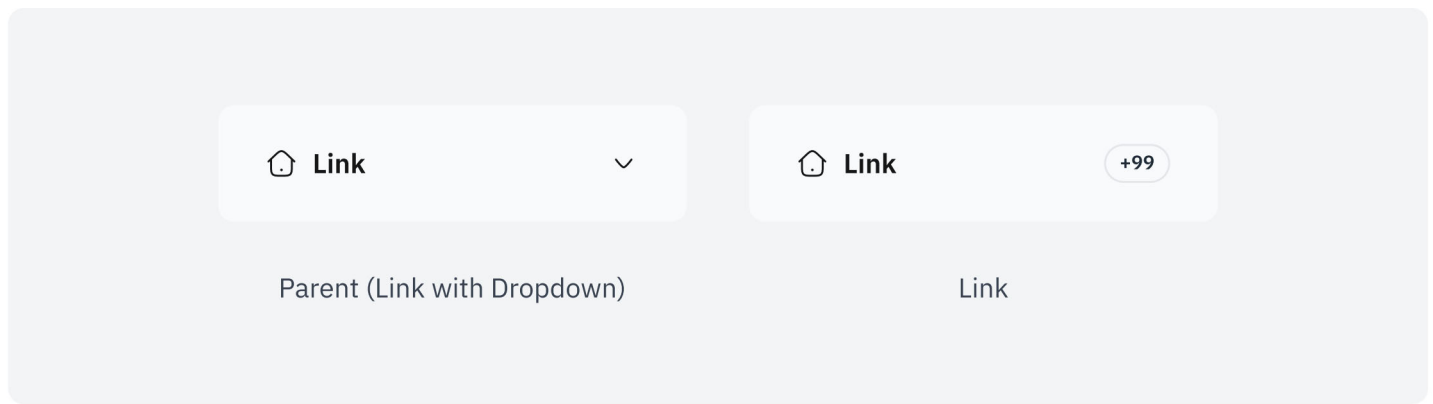


Variants

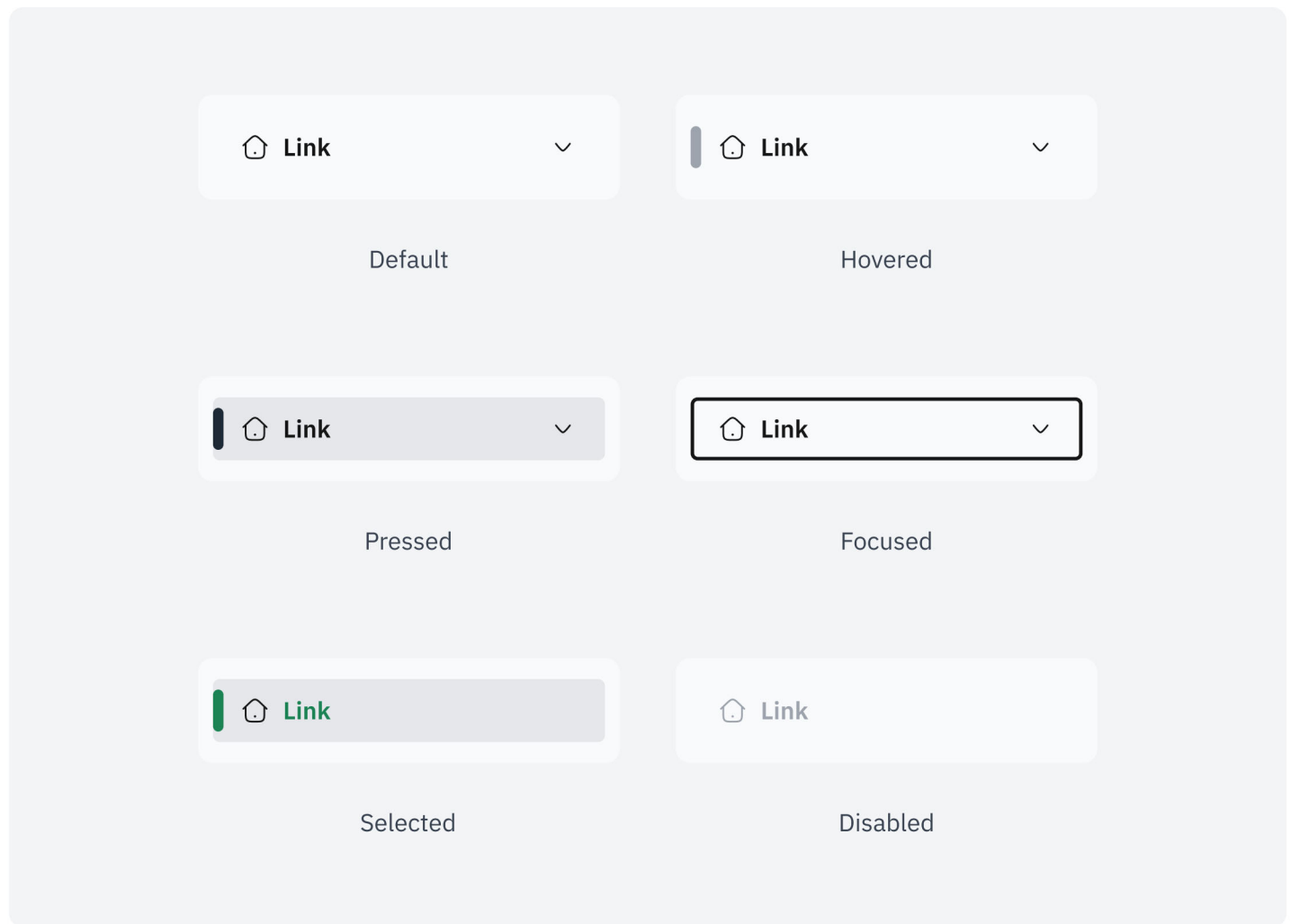
Navigation drawer is available in the Platforms Code in many fundamental formats:

Navigation Drawer Item

- **Parent (Link with Dropdown):** This is a parent link that has a dropdown indicator. When clicked or tapped, this would likely expand to show additional links, known as child links.
- **Link:** This is a simple link without any dropdown functionality. It is usually used for single-level navigation menus where each link takes you directly to a page or section.

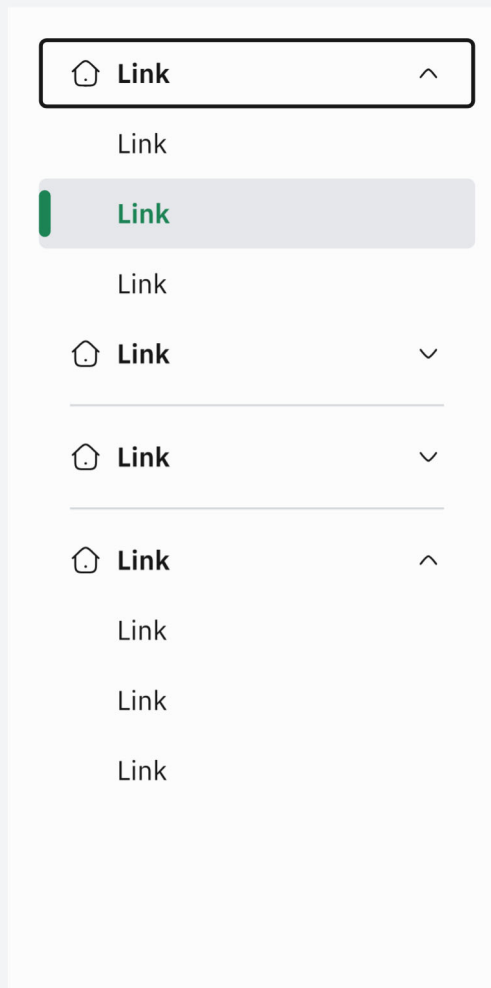


- **Default:** This is the normal state of the navigation link when it is not being interacted with.
- **Hovered:** This state occurs when the user moves the cursor over the navigation link. It often changes visually to indicate that the link is interactive.
- **Pressed:** This is the state when the user clicks or taps on the navigation link but has not yet released it. It's a feedback mechanism to indicate that the link is being activated.
- **Focused:** This state is typically seen when the link is navigated to via keyboard interactions.
- **Selected:** This state shows that the navigation link has been selected or is currently active. It may indicate the current page or section the user is on.
- **Disabled:** This indicates that the link is currently not interactive. It might be used to show that a certain navigation option is not available at the moment.



Appearance

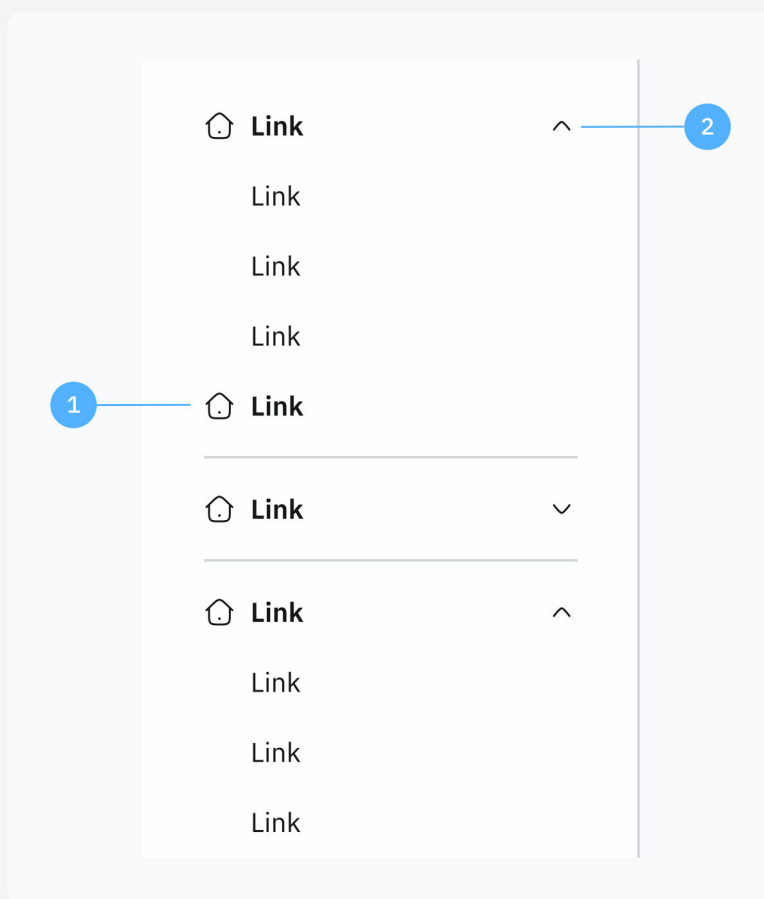
The navigation drawer is designed to be both aesthetically pleasing and functionally streamlined, ensuring users can effortlessly identify and utilize the core features and areas. It facilitates swift and intuitive user interaction, fostering an engaging and straightforward navigational experience.



Anatomy

Component Anatomy

- 1 List Items
- 2 Expand/Collapse Arrows



- **List Items:** Individual items or selections are available within the navigation drawer. Each represents a clickable area that leads to different pages or sections.
- **Expand/Collapse Arrows:** The chevrons (upward '^' and downward 'v') are interactive elements indicating that there are submenus within a list item. An upward chevron ('^') suggests that the submenu is currently collapsed and can be expanded, while a downward chevron ('v') indicates that the submenu is expanded and can be collapsed.

Accessibility

Ensuring the accessibility of a navigation drawer is crucial to accommodate users with disabilities and provide a seamless experience for all users. Here are some key accessibility attributes and practices to consider when implementing a navigation drawer component:

Key Accessibility Attributes and Practices:

ARIA Roles:

Apply appropriate ARIA (Accessible Rich Internet Applications) roles and properties, such as `role="navigation"` for the navigation section and `aria-expanded` to indicate whether a collapsible section of content is currently open or closed.

Keyboard Navigation:

Ensure that all navigation drawer elements can be accessed and activated using a keyboard alone, typically with the Tab, Enter, and Arrow keys, to accommodate users who cannot use a mouse.

Focus Management:

Manage focus carefully, especially for dynamic content. When a submenu is expanded, focus should be moved to the first item in that submenu. Similarly, when a menu is collapsed, focus should return to the parent item.

Visible Focus Indicator:

Provide a visible indicator, such as a border or background change, when an element receives focus. This is critical for users who rely on visual cues to determine which element is focused.

Skip Links:

Include "skip to content" links that allow users to bypass the navigation drawer if it is very long, so they don't have to tab through all the items to get to the main content.

Contrast and Colors:

Use sufficient color contrast ratios for text and interactive elements to aid users with visual impairments.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Navigation Header

Table of Content

← Previous

Up Next →



[Home](#) > [Guidelines](#) > [Components](#) > [UI Shell](#) > [Table of Content](#)

Table Of Content

The Table of Contents is used to provide a navigable outline or summary of content, allowing users to quickly understand and jump to specific sections

Live Demo



Visual



Code

On This Page

[Page Name]

Page Section

Nested Page Section

Nested Page Section

Nested Page Section

Nested Page Section

Nested Page Section

Page Section

Page Section

Nested Page Section

Nested Page Section

Nested Page Section

Page Section

Page Section

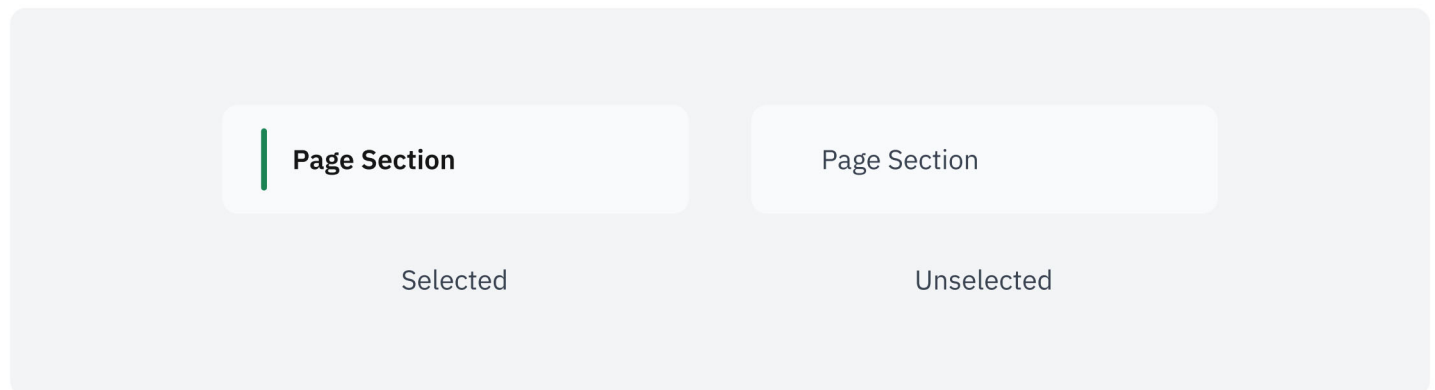
Page Section

Test the preview of the table of content comprehensive exploration, including all variants and features, our full demo will soon be available on **Storybook**.

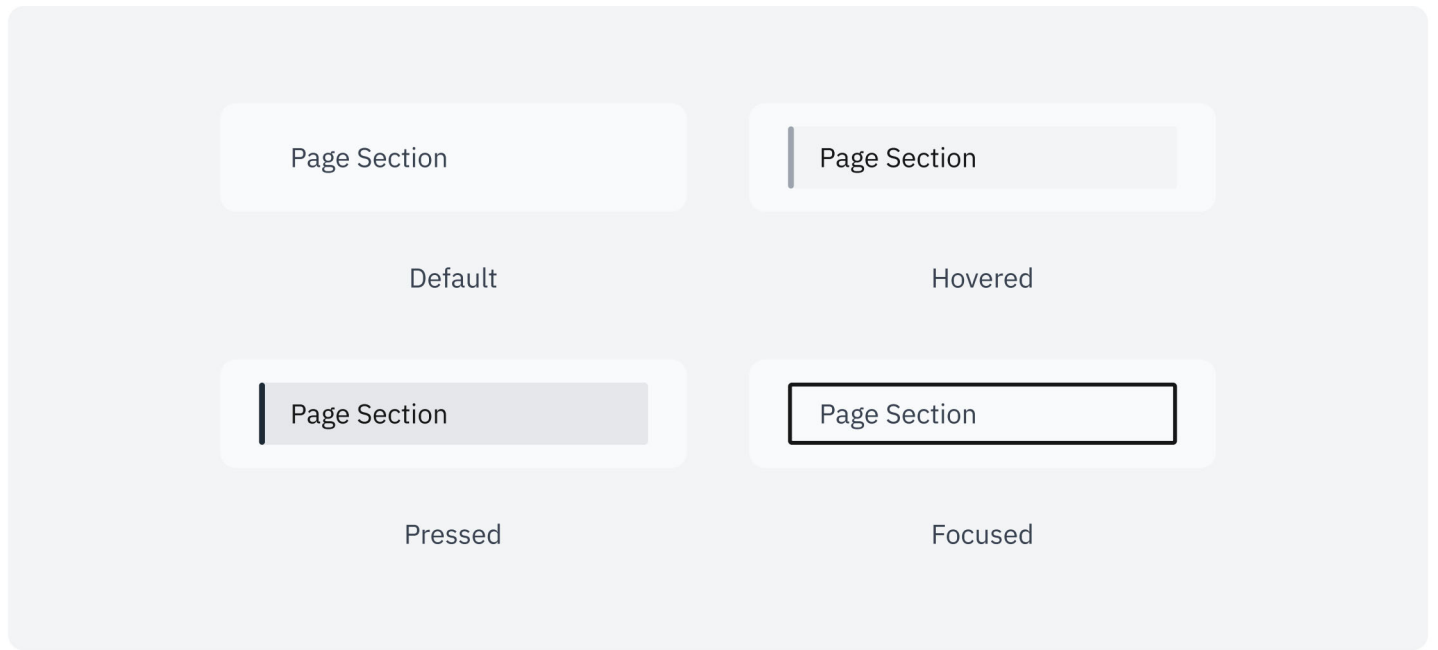
Variants

Table of content (TOC) is available in the Platforms Code in many fundamental formats:

- **Selected:** Once a user has clicked on a TOC item and navigated to a particular section, the TOC item may remain visually highlighted to indicate that it is the currently active section.
- **Unselected:** TOC items that are not currently selected will remain in this state.



- **Default:** This is the initial appearance of the TOC items when the page loads.
- **Hovered:** This state is triggered when a user moves their cursor over a TOC item.
- **Pressed:** When a TOC item is clicked or tapped, it enters the pressed state.
- **Focused:** This state appears when a TOC item is selected through keyboard navigation, like using the Tab key.



Appearance

The design of the Table of Contents is crafted to be visually distinct and instantly recognizable, ensuring that users can swiftly identify and navigate through the sections of content presented.

Table Of Content

On this page

[Page Name]

Page Section

Nested Page Section

Nested Page Section

Nested Page Section

Nested Page Section

Nested Page Section

Page Section

Page Section

Nested Page Section

Nested Page Section

Nested Page Section

Page Section

Page Section

Page Section

Page Section

Page Section

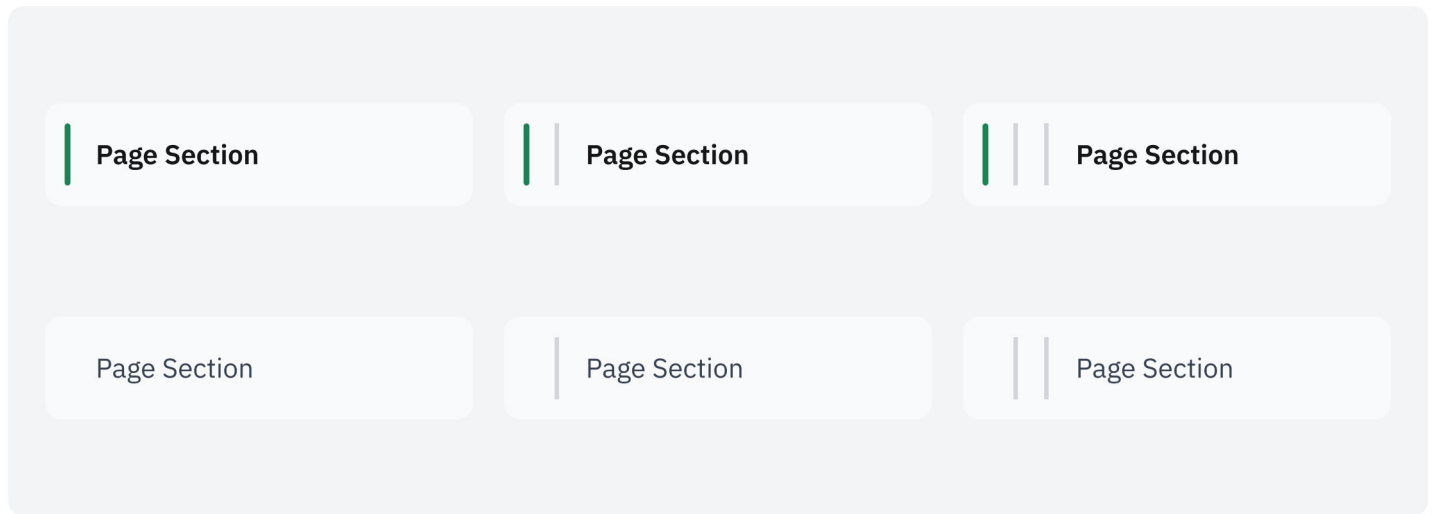
Page Section

Page Section

Page Section

Page Section

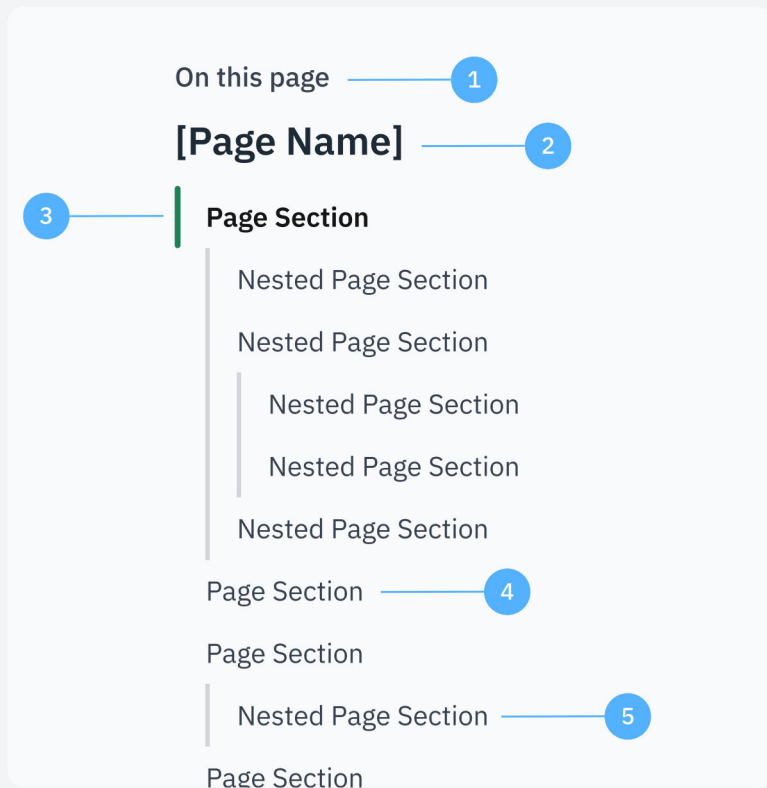
Table Of Content



Anatomy

Component Anatomy

- 1 Title
- 2 Current Page Indicator
- 3 Hierarchy Indicators
- 4 Section Titles
- 5 Nested Section Titles



- **Title:** The label acts as a title for the TOC, indicating its purpose.
- **Current Page Indicator:** The Current Page Indicator serves as an indicator of the current page or section the user is on.

- **Hierarchy Indicators:** The lines and indentation before the section and subsection titles represent the hierarchy. They visually indicate the structure of the content, showing which sections are main topics and which are nested within them.
- **Section Titles:** These are the clickable elements that allow users to navigate to different parts of the content.
- **Nested Section Titles:** These are subsections within the main section titles.

Accessibility

Designing an accessible table of content components it's important to focus on several key attributes and practices. These will ensure that your TOC is usable by everyone, including people with disabilities. Here are some of the main accessibility considerations:

Key Accessibility Attributes and Practices:

ARIA Roles:

Apply ARIA roles such as ***role="navigation"*** to the TOC and use properties like ***aria-label*** to provide a clear description (e.g., ***aria-label="Table of contents"***). For the currently active section, you can use ***aria-current="page"*** to indicate which page or section is active.

Keyboard Navigation:

Ensure that all TOC links can be navigated using a keyboard. This means they should be focusable with the Tab key, and there should be a clear focus state.

Visible Focus Indicator:

Provide a distinct style for ***:focus*** to ensure that keyboard users have a clear indication of which TOC item is selected.

Skip Links:

Implement skip links that allow users to bypass the TOC and go directly to the main content if they wish.

Contrast and Colors:

Text and background colors should have sufficient contrast to be easily distinguishable by users with visual impairments. This is important for both the default and interactive states.

Update Upon Navigation:

When a user activates a TOC link, the page should update (including focus) to reflect the new content, and the TOC should be updated to indicate the current active section.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Navigation Drawer

Footer

[← Previous](#)

[Up Next →](#)



[Home](#) > [Guidelines](#) > [Components](#) > [UI Shell](#) > [Footer](#)

Footer

A footer is serve several functions, including providing navigation, housing links to important information, offering contact information, and sometimes integrating social media icons and subscription forms.

Live Demo



Visual



Code

Group Label

[Footer Link](#)

[Footer Link](#)

[Footer Link](#)

[Footer Link](#)

[Footer Link](#)

Group Label

[Footer Link](#)

[Footer Link](#)

[Footer Link](#)

Group Label

[Footer Link](#)

[Footer Link](#)

[Footer Link](#)

[Footer Link](#)

[Footer Link](#)

Social Media



Accessibility Tools



[Footer Link](#)[Footer Link](#)[Footer Link](#) [Footer Link](#) [Footer Link](#)[Footer Link](#) [Footer Link](#) [Footer Link](#)[Footer Link](#) [Footer Link](#)

Platform Logo



Platform Logo

**All Right Reserved For Digital Government
Authority © 2024**

[Terms and Conditions](#) [Privacy Policy](#)

Test the preview of the footer comprehensive exploration, including all variants and features, our full demo will soon be available on **Storybook**.

Variants

Footer is available in the Platforms Code in two formats:

- **Width greater than 600 pixels:** For screens wider than 600 pixels with a multi-column layout for elements like links and social media icons.
- **Width less than 600 pixels:** For screens narrower than 600 pixels, stacking elements vertically for improved readability on smaller devices.

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

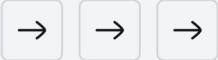
Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Social Media



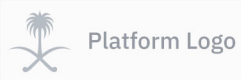
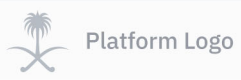
Accessibility Tools



[Footer Link](#) [Footer Link](#) [Footer Link](#)

[Footer Link](#) [Footer Link](#) [Footer Link](#)

[Footer Link](#) [Footer Link](#)



All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)

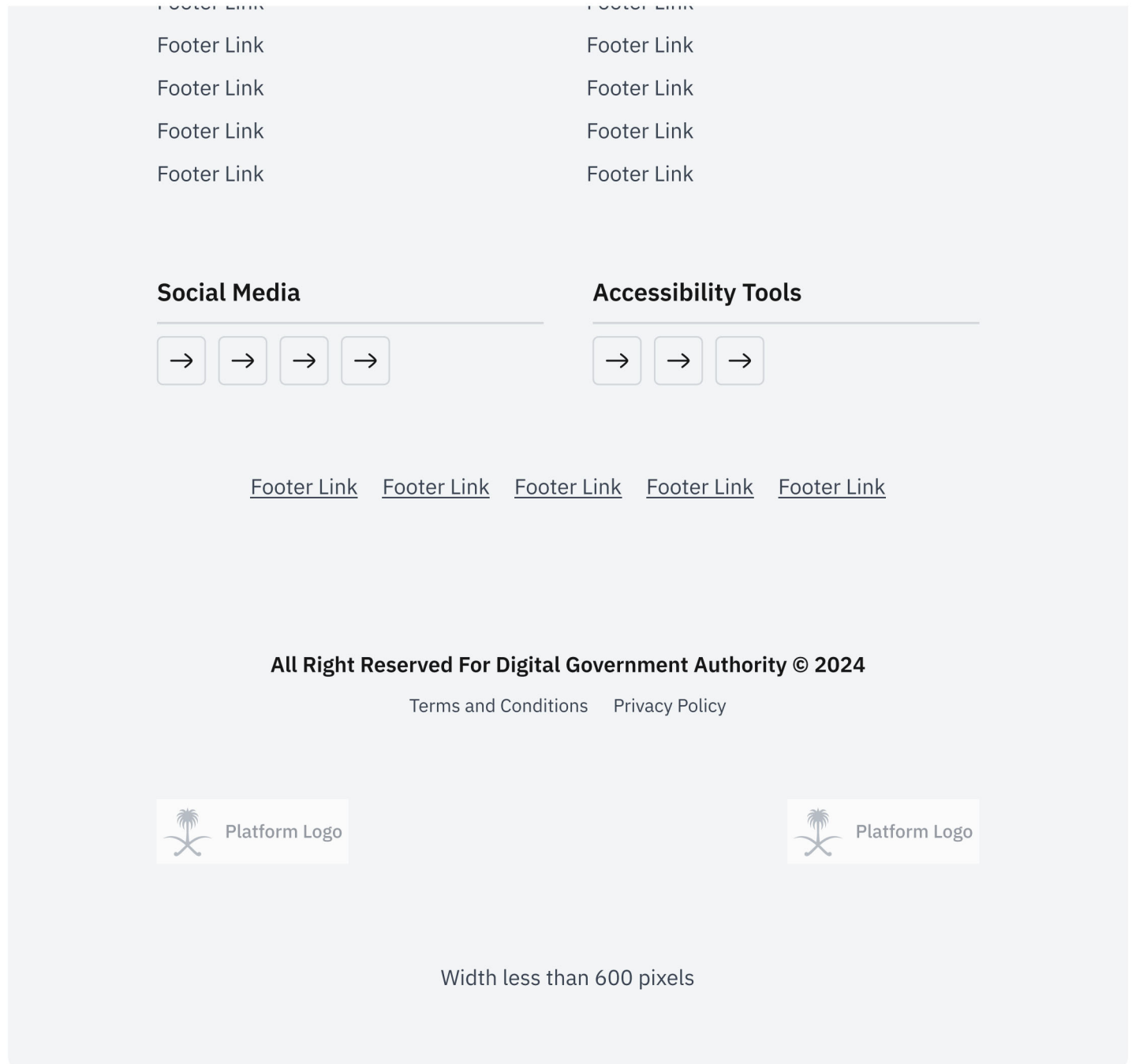
Width greater than 600 pixels

Group Label

Footer Link

Group Label

Footer Link



Appearance

The footer is crafted to be both visually harmonious and practically efficient, guaranteeing that users can easily access and utilize essential information and functionalities. It supports a seamless and direct user journey.

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

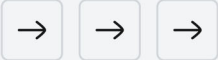
Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Social Media



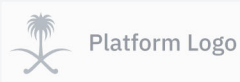
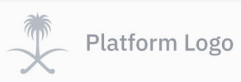
Accessibility Tools



[Footer Link](#) [Footer Link](#) [Footer Link](#)

[Footer Link](#) [Footer Link](#) [Footer Link](#)

[Footer Link](#) [Footer Link](#)



All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)

[Footer Link](#) [Footer Link](#) [Footer Link](#)

[Footer Link](#) [Footer Link](#) [Footer Link](#)

[Footer Link](#) [Footer Link](#)



All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Group Label

Footer Link
Footer Link
Footer Link
Footer Link
Footer Link

Social Media



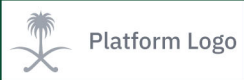
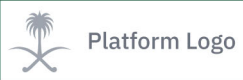
Accessibility Tools



[Footer Link](#) [Footer Link](#) [Footer Link](#)

[Footer Link](#) [Footer Link](#) [Footer Link](#)

[Footer Link](#) [Footer Link](#)



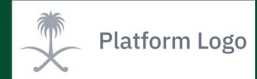
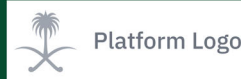
All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)

[Footer Link](#) [Footer Link](#) [Footer Link](#)

[Footer Link](#) [Footer Link](#) [Footer Link](#)

[Footer Link](#) [Footer Link](#)



All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)

Anatomy



Component Anatomy

- 1 Group Labels
- 2 Footer Links
- 3 Social Media Icons
- 4 Accessibility Tools

- **Group Labels:** These act as headers to categorize the footer content and improve scanability, typically including titles like "Contact," "Company," or "Resources".
- **Footer Links:** These are the clickable texts under each group label, guiding users to various pages of the website, such as "About Us," "Careers," or "Support."
- **Social Media Icons:** Visual representations of various social media platforms that offer a direct link to the company's social media pages, facilitating community engagement.
- **Accessibility Tools:** Buttons or links designed to improve the website's usability for people with disabilities, which may include features like font size adjustment or contrast settings.
- **Link:** helps with providing additional information or navigation options to the user.
- **Legal Text:** This usually includes copyright notices and possibly links to privacy policies or terms of service.
- **Company Logo:** A visual identifier for the company logo.

Accessibility

Designing accessible footer components it's crucial to ensure all users, including those with disabilities, can navigate and interact effectively. Here are some key accessibility attributes and practices to consider for a footer component:

Key Accessibility Attributes and Practices:

ARIA Roles:

Implement ARIA roles to identify the footer region ***role='contentinfo'*** so screen readers can easily navigate to it.

Keyboard Navigation:

Ensure that all footer links can be accessed and activated using the keyboard alone, typically by tabbing through the links.

Focus Indicators

Provide visible focus indicators for interactive elements to assist users who rely on keyboard navigation.

Link Purpose (In Context)

Use clear, descriptive text for links to help users understand where each link will take them without needing additional context.

Skip Links

Include a "skip to content" link at the top of the page to allow users to bypass the navigation links in the footer if they are repetitive across pages.

Contrast and Colors:

Maintain high contrast between text and background colors to meet or exceed WCAG (Web Content Accessibility Guidelines) standards.

Collaborate & elevate this component!

If you have ideas to improve our components or want to help make our platform more inclusive and powerful, we'd love to hear from you.

[Contribute Now](#) [Learn More](#)

Table of Content

[← Previous](#)

Thoughts

[Up Next →](#)



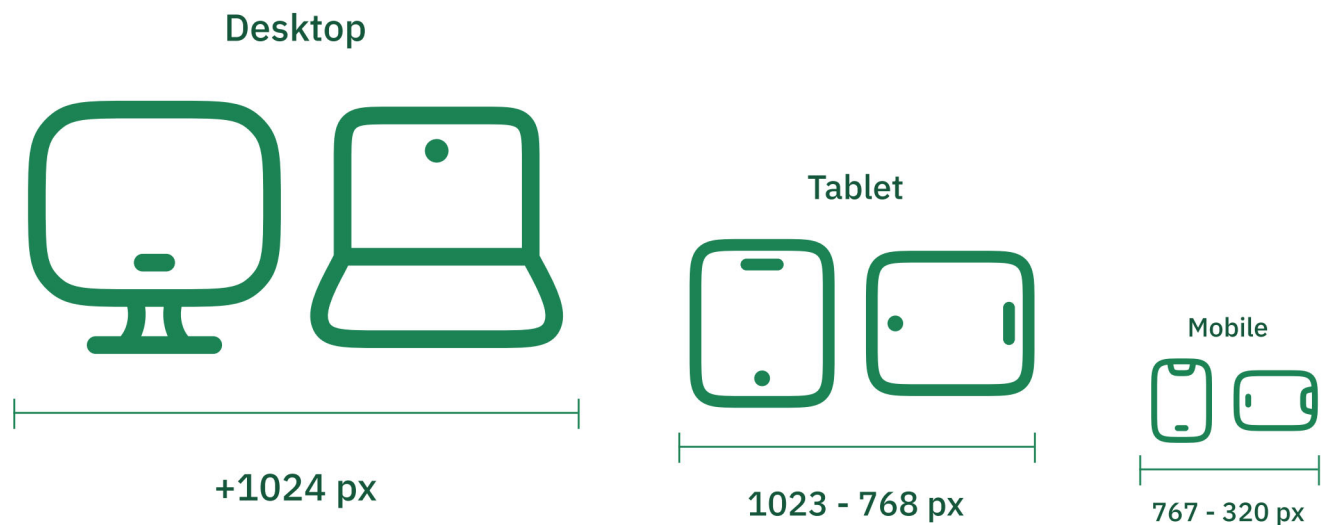
[Home](#) > [Thoughts](#) > [Responsive Design](#)

Responsive Design

Responsive design is a fundamental approach to web design aimed at creating websites that adapt and respond to various screen sizes, devices, and orientations. This ensures an optimal viewing and interaction experience for users across desktops, laptops, tablets, and smartphones. Responsive design is particularly crucial for government websites, which must serve diverse audiences with different devices and accessibility needs.

Purpose of Responsive Design

The purpose of responsive design is to create adaptable, efficient, and user-friendly websites that perform well across all devices, ensuring a consistent and optimal user experience while streamlining the development and maintenance processes.



For Designers

- **User Experience Consistency:** Designers aim to provide a seamless experience for users, whether they access the site on a desktop, tablet, or smartphone. Responsive design ensures that the visual and interactive elements adapt fluidly to different screen sizes, maintaining usability and aesthetic appeal.
- **Brand Cohesion:** Maintaining a consistent brand identity across all devices is essential. Responsive design allows designers to ensure that logos, color schemes, and other brand elements look uniform across different platforms, reinforcing brand recognition and trust.

For Developers

- **Efficient Development:** Developers use responsive design to streamline the development process. By creating a single codebase that adapts to various devices, they save time and resources compared to developing separate versions for each device.
- **Future-Proofing:** With the rapid evolution of technology, new devices with varying screen sizes are constantly emerging. Responsive design helps developers future-proof their projects by ensuring adaptability to new devices without significant overhauls.

Importance of the Responsive Design

Enhanced User Experience

Responsive design ensures that websites adapt seamlessly to various devices and screen sizes, providing a consistent and user-friendly experience for all users.

Expanded Reach

With the proliferation of mobile devices, responsive design enables websites to reach a broader audience by catering to users accessing the internet on smartphones and tablets.

Accessibility Compliance

Responsive design facilitates adherence to accessibility standards, ensuring that websites are usable by individuals with disabilities across all devices.

Important Factors to Consider While Creating Responsive Design Interfaces

Creating responsive design interfaces requires attention to various factors to ensure a seamless and effective user experience across all devices. Here are the key considerations for designers and developers:

For Designers

1. User-Centered Design:

- **User Needs and Behaviors:** Understand the target audience and their behaviors on different devices. Design with user needs in mind, ensuring that key information and functionality are easily accessible on all screen sizes.
- **Touchscreen Navigation:** Consider the use of touch gestures and ensure that interactive elements are appropriately sized and spaced for touch inputs on mobile devices.

2. Visual Hierarchy and Content Prioritization:

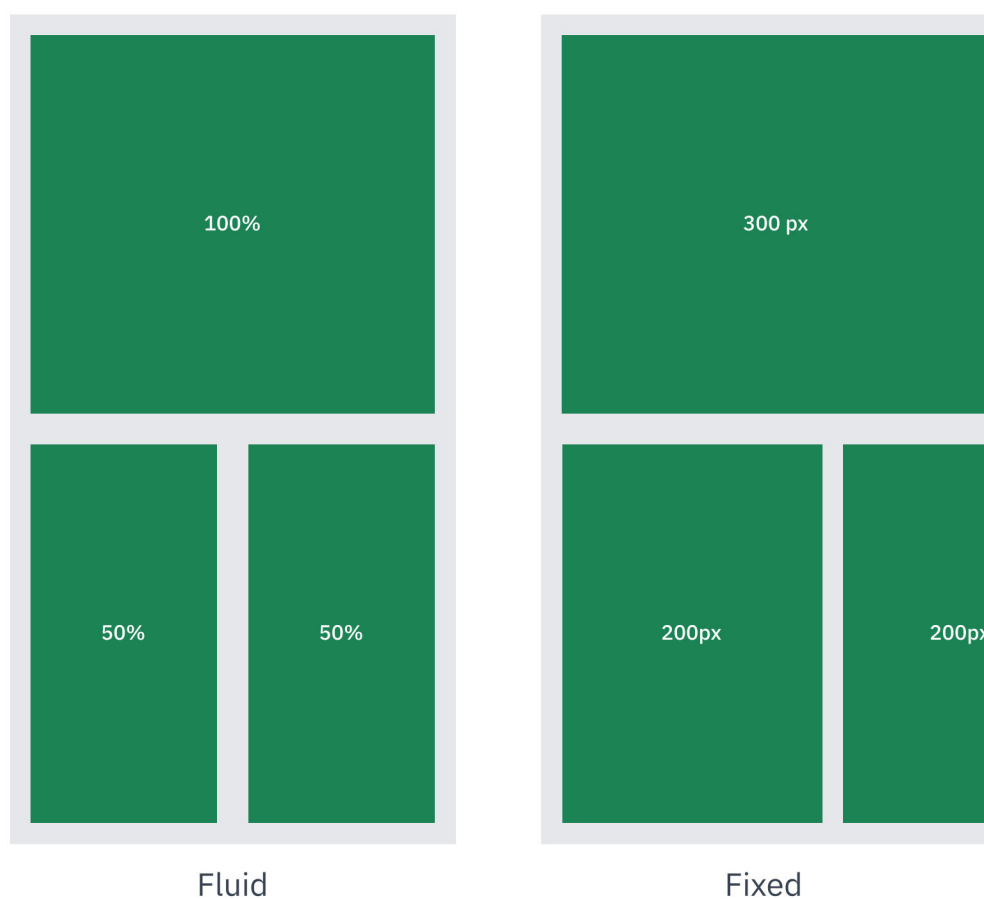
- **Content Hierarchy:** Prioritize content based on its importance. Ensure that essential information is prominently displayed and easily accessible on smaller screens.
- **Scalable Typography:** Use relative units, like 16px (1em, 1rem) for font sizes to maintain readability across different devices. Adjust font sizes and line heights to ensure legibility on both small and large screens.

3. Flexible Layouts:

- **Fluid Grids:** Use fluid grid layouts that adjust based on the screen size. This ensures that content adapts dynamically to various devices.
- **Responsive Breakpoints:** Set up grid systems for different breakpoints (e.g., 12-column grid for desktop, 8-column grid for tablet) to guide the layout adjustments for each screen size.

4. Consistent User Experience:

- **Brand Consistency:** Maintain a consistent visual identity across all devices. Ensure that colors, fonts, and other design elements are cohesive.
- **Cross-Platform Usability:** Design interfaces that provide a similar experience regardless of the device, ensuring users can navigate and interact with the site intuitively.



For Developers

1. Efficient Media Queries:

- **Breakpoints:** Define appropriate breakpoints using CSS media queries to apply specific styles for different screen sizes. Use common breakpoints but also tailor them to the design's needs.

- **Mobile-First Approach:** Start by designing for the smallest screens and progressively enhance the design for larger screens. This ensures a solid base design and better performance on mobile devices.

2. Adaptive Layouts:

- **CSS Flexbox and Grid:** Utilize CSS Flexbox and Grid layouts to create flexible and adaptive designs. These layout systems provide powerful tools for building responsive interfaces that adjust seamlessly to different screen sizes.
- **Container Queries:** Consider using container queries (when they become widely supported) to style elements based on the size of their parent container, rather than the viewport size.

Contact Us

Connect With Us

Whether you have questions, need support, or want to share your feedback, we're here to listen and help.

DS-DGA@dga.gov.sa

[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)

[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)



[Home](#) > [Thoughts](#) > [Accessibility](#)

Accessibility

Accessibility Ease ensures that all people (regardless of ability) can interact with the information or services you provide.

In an era where technology permeates every facet of our lives, the importance of inclusive design cannot be overstated. Accessibility Ease is a pioneering initiative designed to ensure that digital environments are not only user-friendly but truly accessible to all, including those with disabilities. Here, we explore the core aspects of Accessibility Ease and its impact on fostering an inclusive digital community.

Foundations of Accessibility Ease

Accessibility Ease is built on the premise that technology should be accessible to everyone, regardless of their physical or cognitive abilities. This initiative incorporates the latest standards in accessibility, such as the Web Content Accessibility Guidelines (WCAG), which are designed to make web content more accessible to people with a wide range of disabilities, including visual, auditory, physical, speech, cognitive, language, learning, and neurological disabilities.

Accessibility Ease focuses on several key areas.

1. Visual Design

Accessibility Ease ensures that visual design is perceivable by all users. This includes providing sufficient contrast between text and background, using color schemes that are friendly to colorblind users, and offering alternatives to visual information

2. Interactive Elements

Accessibility Ease emphasizes keyboard-friendly navigation, ensuring that all interactive elements are accessible via keyboard commands for those who cannot use a mouse.

3. Screen Reader Compatibility

Ensuring compatibility with screen readers is essential. Text is made screen reader-friendly, with appropriate tagging and ARIA (Accessible Rich Internet Applications) labels that help describe images, icons, and other non-textual elements.

4. Adaptable Content

Content under Accessibility Ease is designed to be fully adaptable, allowing users to modify content presentation without losing information or structure. This includes resizable text, adaptable layouts, and multiple viewing options to accommodate various disabilities.

5. Cognitive Considerations

Recognizing that cognitive disabilities can affect how individuals perceive and interact with digital content, Accessibility Ease incorporates clear, intuitive navigation and consistent layouts to reduce cognitive load.

Impact on the Community

The introduction of Accessibility Ease has had a profound impact on the community. By removing barriers to digital content, this initiative has enabled individuals with disabilities to participate more fully in digital spaces. This not only enhances their personal and professional lives but also enriches the online community with diverse perspectives.

In conclusion, digital accessibility is a vital component in fostering inclusion for people with disabilities in the online world. Moreover, brands that commit to enhancing accessibility are not just acting ethically, they are positioning themselves to benefit and expand significantly!

Contact Us

Connect With Us

Whether you have questions, need support, or want to share your feedback, we're here to listen and help.

DS-DGA@dga.gov.sa

[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)

[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)



Home > Thoughts > Design Tokens

Design Tokens

Design tokens are a way to store and manage design decisions in a centralized, systematic, and scalable manner.

Design tokens represent the smallest pieces of a design system, such as color codes, typography styles, spacing values, border radius, shadows, and more. By abstracting these design properties into variables or tokens, designers and developers can maintain consistency and efficiency across an entire application or suite of products.

Key Features of Design Tokens

1. Centralized Management

Design tokens provide a single source of truth for design values. This means that any changes to the design system can be made in one place and automatically applied throughout the entire project.

2. Consistency

By using tokens, you ensure that the same values are applied consistently across all elements and components. This helps maintain a unified look and feel.

3. Scalability

Design tokens make it easier to scale design systems across multiple platforms and devices. Tokens can be used in web, mobile, and other environments, ensuring a consistent experience.

4. Ease of Updates

By using tokens, you ensure that the same values are applied consistently across all elements and components. This helps maintain a unified look and feel.

5. Improved Collaboration

With design tokens, designers and developers can work more effectively together. Tokens bridge the gap between design and code, making it easier to implement and maintain design systems.

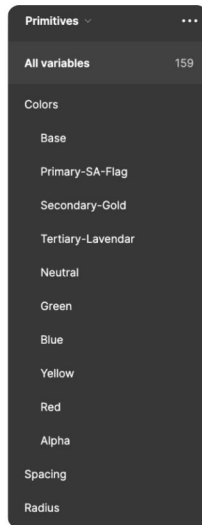
This transition to using variables or design tokens in Figma and other tools has revolutionized the way design systems are managed, making them more efficient, consistent, and scalable.

Design Tokens in Platforms code

Design tokens in Platforms code are organized into two main collections, each containing values for fundamental design elements such as spacing, radius, colors, and widths. Within these collections, there are sub-collections for specific cases, storing values for component states and color values for each theme. These sub-collections derive their tokens from the primitives. In Platforms code, design tokens collections are categorized into two types: the primitive collections and the semantic collections.

Primitive collections (Foundational Tokens)

Primitive tokens serve as the essential elements of a design system, carrying core values like colors, spacing, and radius dimensions. These tokens are raw values and lack contextual meaning regarding their application within an interface. They can be repurposed in other collections for more specific uses, where they are renamed according to their intended function.



		
Name	Values	+
Colors / Base		
 white	 FFFFFFFF	
 black	 161616	
Colors / Primary-SA-Flag		
 25	 F7DFD9	
 50	 F3FCF6	
 100	 DFF6E7	
 200	 B8EACB	
 300	 88D8AD	
 400	 54C08A	
 500	 25935E	

Semantic collections (Context-specific)

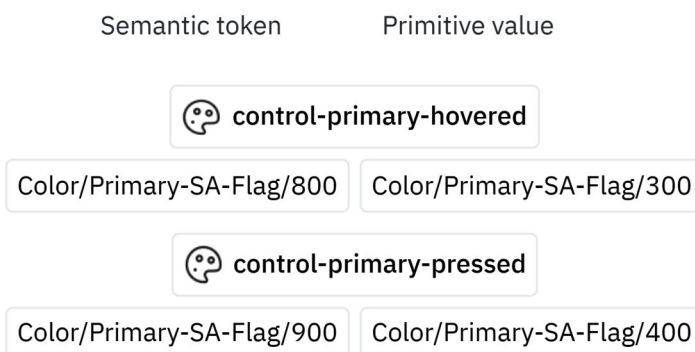
Context-specific tokens that abstract the primitive tokens into meaningful components related to the actual use in the UI design. They add more clarity and intent by describing the purpose of the token in the context of the design.

In Platforms code semantic tokens are divided into a number of collection

1. Theme collection
2. Spacing collection
3. Radius collection

Theme collection

The theme collection carry all the color design choices for each component, drawn from the primitive color collection. Each sub-collection in themes features two different themes (Light and dark) , giving every component its unique values for each theme and allowing for easy scaling or the addition of new themes. The sub-themes of the components cover all possible scenarios where components can appear. For instance, a button may have different values when hovered over or selected, reflecting distinct characteristics in each case.



Light Theme

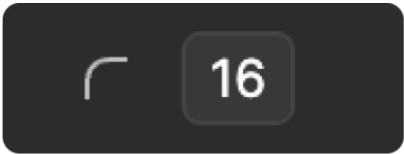


Dark Theme

Themes ▾	...			
All variables	316	Name	Light	Dark
Background		Background		
Text				
Global		background-white	Colors/Base/white	Colors/Neutral/950
Border		background-body	Colors/Neutral/50	Colors/Neutral/900
Button		background-menu	Colors/Base/white	Colors/Neutral/800
Link		background-card	Colors/Base/white	Colors/Neutral/700
Icon		surface-oncolor	Colors/Base/white	Colors/Base/white
Tag				
Form		background-black	Colors/Base/black	Colors/Base/white

Radius collection

The Radius collection includes all the radius values for each component, sourced from the primitive radius collection. Each sub-collection within the Radius collection offers three different frame sizes: Desktop, Tablet, and Mobile.



Radius ▾	...			
All variables	8	Name	Desktop	Tablet
Components		# radius-none	# Radius/radius-0	# Rad
Radius		# radius-xs	# Radius/radius-2	# Rad
		# radius-sm	# Radius/radius-4	# Rad
		# radius-md	# Radius/radius-8	# Rad
		# radius-lg	# Radius/radius-16	# Rad
		# radius-xl	# Radius/radius-24	# Rad

Spacing and width collection

The Spacing and width collections contains all the spacing and width values for each component, derived from the primitive spacing collection. Within both the Spacing and the width collections, each sub-collection is tailored for three different frame sizes: Desktop, Tablet, and Mobile.

# width-xxs	# Spacing/80 (320px)
# width-xs	# Spacing/96 (384px)
# width-sm	# Spacing/120 (480px)
# width-md	# Spacing/140 (560px)
# width-lg	# Spacing/160 (640px)
# width-xl	# Spacing/192 (768px)
# width-2xl	# Spacing/256 (1,024px)
# width-3xl	# Spacing/320 (1,280px)
# width-4xl	# Spacing/360 (1,440px)
# width-5xl	# Spacing/400 (1,600px)
# width-6xl	# Spacing/480 (1,920px)

Spacing ▾	...	☐			
All variables	104	Name	Desktop	Tablet	Mobile
Tooltip		Tooltip			
Tab					
Notification		# tooltip-padding	# Spacing/2 (8px)	# Spacing/2 (8px)	# Spacing/2 (8px)
Table		# tooltip-gap	# Spacing/2 (8px)	# Spacing/2 (8px)	# Spacing/2 (8px)
Control					
Form		Tab			
Accordion		# tab-button-gap	# Spacing/1 (4px)	# Spacing/1 (4px)	# Spacing/1 (4px)
Card		# horizontal-tab-... -padding	# Spacing/3 (12px)	# Spacing/3 (12px)	# Spacing/2 (8px)
Section		# horizontal-tab-... -padding	# Spacing/4 (16px)	# Spacing/4 (16px)	# Spacing/3 (12px)
Text		# horizontal-tab-s... -padding	# Spacing/2 (8px)	# Spacing/2 (8px)	# Spacing/1 (4px)
Button		# horizontal-tab-s... -padding	# Spacing/3 (12px)	# Spacing/3 (12px)	# Spacing/2 (8px)
Icon		# horizontal-tab-s... -padding	# Spacing/3 (12px)	# Spacing/3 (12px)	# Spacing/2 (8px)
Model		# vertical-tab-md-... -padding	# Spacing/1.5 (6px)	# Spacing/1.5 (6px)	# Spacing/1 (4px)
Link		# vertical-tab-md-... -padding	# Spacing/3 (12px)	# Spacing/3 (12px)	# Spacing/2 (8px)
Progress Indicator					

Key Differences

Level of Abstraction:

Primitive tokens are low-level and directly define basic design properties.

Semantic tokens are high-level and provide meaningful context.

Usage Context:

Primitive tokens are used as fundamental building blocks.

Semantic tokens are used to ensure consistency and clarity in design by mapping to their corresponding primitive tokens

Flexibility and Maintenance:

Using semantic tokens makes it easier to maintain and update a design system. Changing a semantic token in one place can update multiple components consistently.

Primitive tokens might require more manual updates across the design system if changes are needed.

By using both primitive and semantic tokens, design systems can be both flexible and consistent, making it easier to maintain and scale UI designs.

Contact Us

Connect With Us

Whether you have questions, need support, or want to share your feedback, we're here to listen and help.

DS-DGA@dga.gov.sa

[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)

[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)



Home > Thoughts > Local & Global Standers

Local & Global Standers

Saudi Vision 2030 aims for digital transformation via the Unified Design System (Code Platforms), adhering to local and international standards. Government compliance ensures alignment and service improvements. This article reviews these indicators and the system's adherence.

As part of Saudi Vision 2030, the Kingdom aims to achieve comprehensive digital transformation through the implementation of the Unified Design System (Code Platforms). This system is characterized by its adherence to the standards of both local and international indicators related to digital transformation. Compliance by any governmental entity with the Unified Design System ensures its alignment with these indicators, thereby enhancing its digital transformation commitment and improving its services to meet international standards.

In this article, we will review the local and international indicators related to this system and how it adheres to them.

Digital Transformation Measurement Indicator

What is Digital Transformation Measurement?

Digital Transformation Measurement is an evaluation process based on a specific methodology aimed at governmental entities to diagnose their current status and track their progress in digital transformation. This is done according to best practices and standards, contributing to the achievement of Saudi Vision 2030 goals.

Objectives of Digital Transformation Measurement:

- 1. Enhancing Governmental Compliance:** Ensures governmental entities comply with orders and decisions related to digital transformation.
- 2. Achieving Digital Transformation Requirements:** Through the optimal application of basic digital transformation standards.
- 3. Developing Digital Government:** Contributes to enhancing the capabilities of the digital government.
- 4. Improving Governmental Performance and Effectiveness:** Aids in increasing the efficiency of governmental performance.
- 5. Enhancing Service Quality:** Improves the quality of services provided to beneficiaries.
- 6. Contributing to International Rankings:** Enhances the Kingdom's position in international reports related to digital transformation.

Objectives of the Digital Experience Maturity Indicator:

- 1. Aligning with Digital Government Strategies:** Aims to improve the alignment with digital strategies.
- 2. Enhancing Digital Experience Maturity:** Focuses on raising the maturity level of digital platforms and government services.

3. Aligning with Digital Government Strategies: Aims to increase beneficiary satisfaction with digital services provided.

4. Establishing Digital Experience Maturity Practices: Based on the best international standards.

5. Accelerating Digital Transformation: Promotes faster digital transformation of government services and platforms

6. Leveraging High-Maturity Platforms and Services: Uses them as national models to emulate.

National Targets of the Digital Experience Maturity Indicator:

Quality of Life: Improves the quality of life for beneficiaries by providing effective digital government services.

Beneficiary Satisfaction: Increases beneficiary satisfaction with digital government services.

Business Practice: Enhances competitiveness and facilitates business practices.

Community Participation: Improves beneficiary participation in public decision-making and the design of digital government services.

International Indicators and Digital Competitiveness: Achieves advanced positions in major international indicators related to digital government development.

Government Digital Transformation: Enhances governmental performance and increases the rate of digital transformation in government services, improving decision-making processes.

International Indicators

The Unified Design System aligns with the requirements of the United Nations, thereby enhancing the Kingdom's position in international digital development indicators. Local standards are aligned with international ones to ensure significant progress in the fields of digital government and digital transformation.

Conclusion

The Unified Design System is a strategic step towards achieving Saudi Vision 2030, contributing to improved governmental performance and enhancing the quality of digital services provided to beneficiaries. Compliance by governmental entities with local and international indicators strengthens the Kingdom's position on the international stage and achieves the goals of comprehensive digital transformation.

Contact Us

Connect With Us

Whether you have questions, need support, or want to share your feedback, we're here to listen and help.

DS-DGA@dga.gov.sa

[Design Language](#) [Guidelines](#) [Documentations](#) [Contribution](#) [Assessment Criteria](#) [FAQ](#)

[About Platforms Code](#) [Sitemap](#)

All Right Reserved For Digital Government Authority © 2024

[Terms and Conditions](#) [Privacy Policy](#)