# Dive into UI Design Systems: theory and practice

Level up your design skills with expert insights and practical tips on Design Systems.

Dima Groshev

# Table of content

Version 0.5 (Oct 24, 2024)

THEORY

PRACTICE

# Introduction

Hey there! I’m Dima, a product designer and the founder of 123done.

With over 15 years of experience, I’ve explored various fields such as printed materials, video creation, web design, and mobile interfaces. However, my main passion has always been interface design, which has been the focus of most of my career. From small landing pages to large online stores and portals, I’ve worked on a wide range of projects.

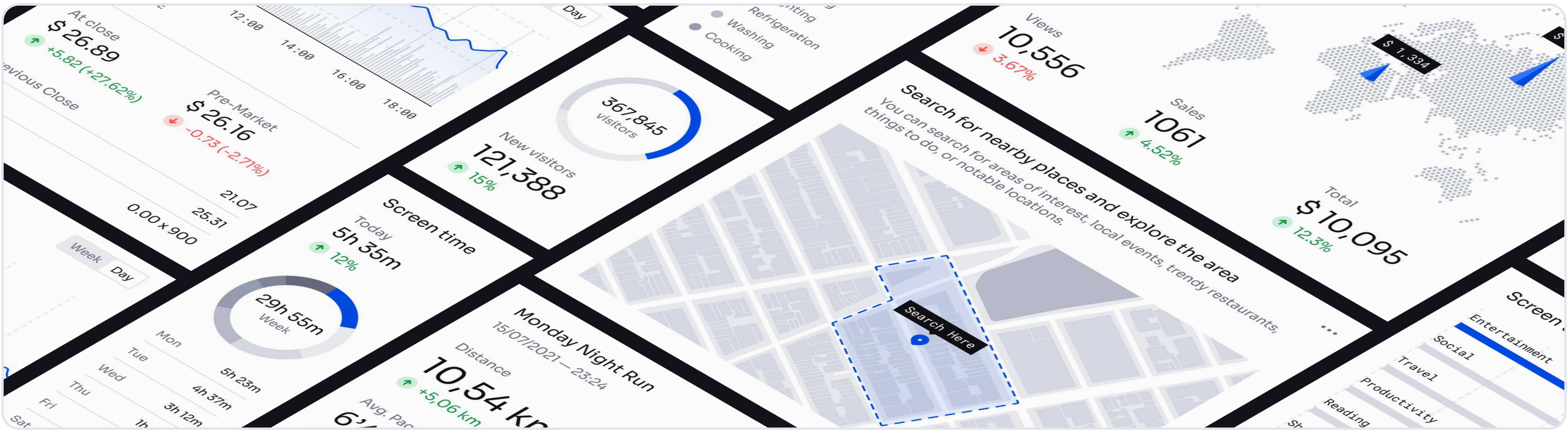
I’ve had the privilege of working with Amway, Bacardi, Danone & many many other great brands.

During my work on commercial projects, I often found myself in need of additional tools to streamline interface design. That’s why four years ago I founded 123done.

The first thing I wanted was a set of icons in different styles, so I created the Universal Icon Set.

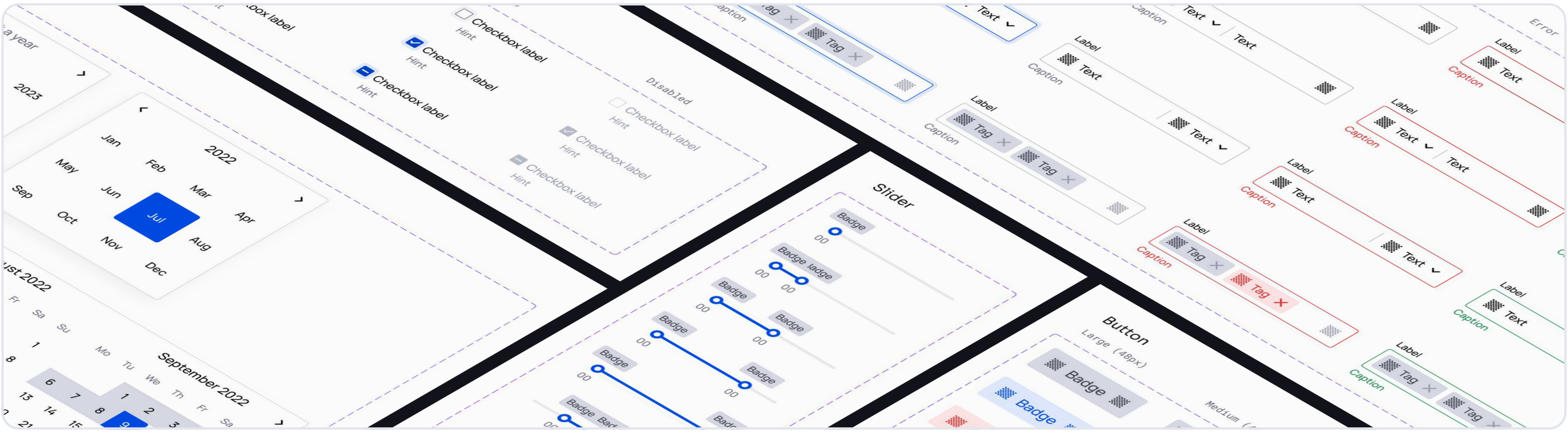
Universal Icon Set

Later, I worked on a project that required drawing numerous graphs, which inspired the creation of the Universal Data Visualization.



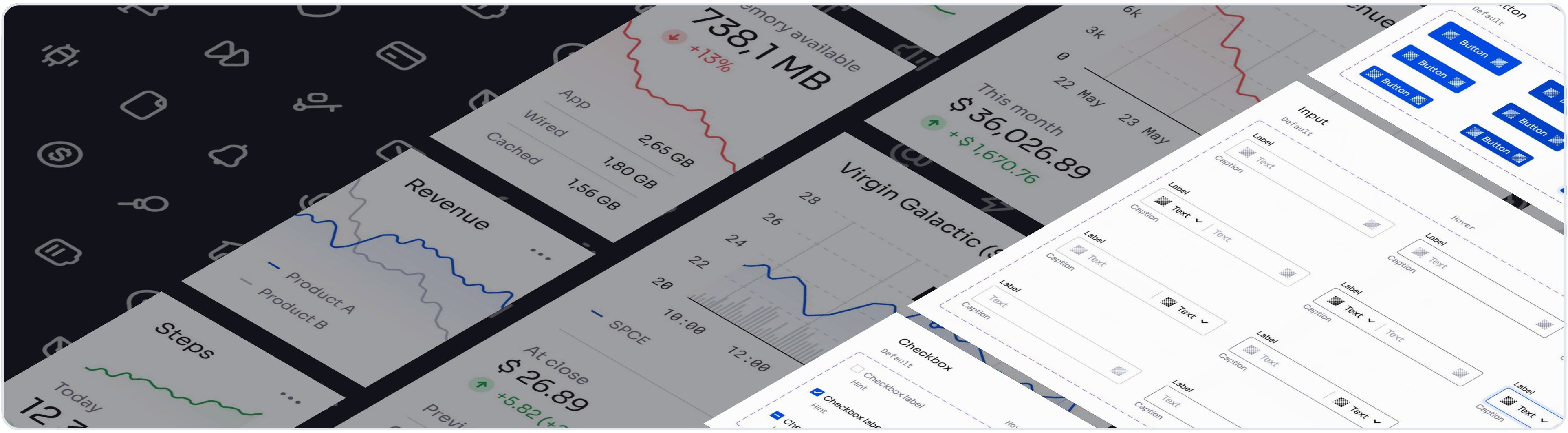
Universal Data Visualization

At the same time, I began working on the Universal UI Kit.



Universal UI Kit

These products are combined together into the Universal Design System.



Universal Design System

At the same time, I started learning more about various design systems by studying best practices from leading companies and reading articles on the subject.

As a result, I acquired both theoretical knowledge and practical skills about design systems, which I'd like to share with you.

## Why did I write this book?

For the last three years, I’ve been working on the Universal Design System and had to redo it several times to get it right.

In this book, I’ve collected the knowledge I’ve gained as a UI designer, especially from my recent work on the design system.

I hope this book helps you avoid the mistakes I made and reach your goals faster.

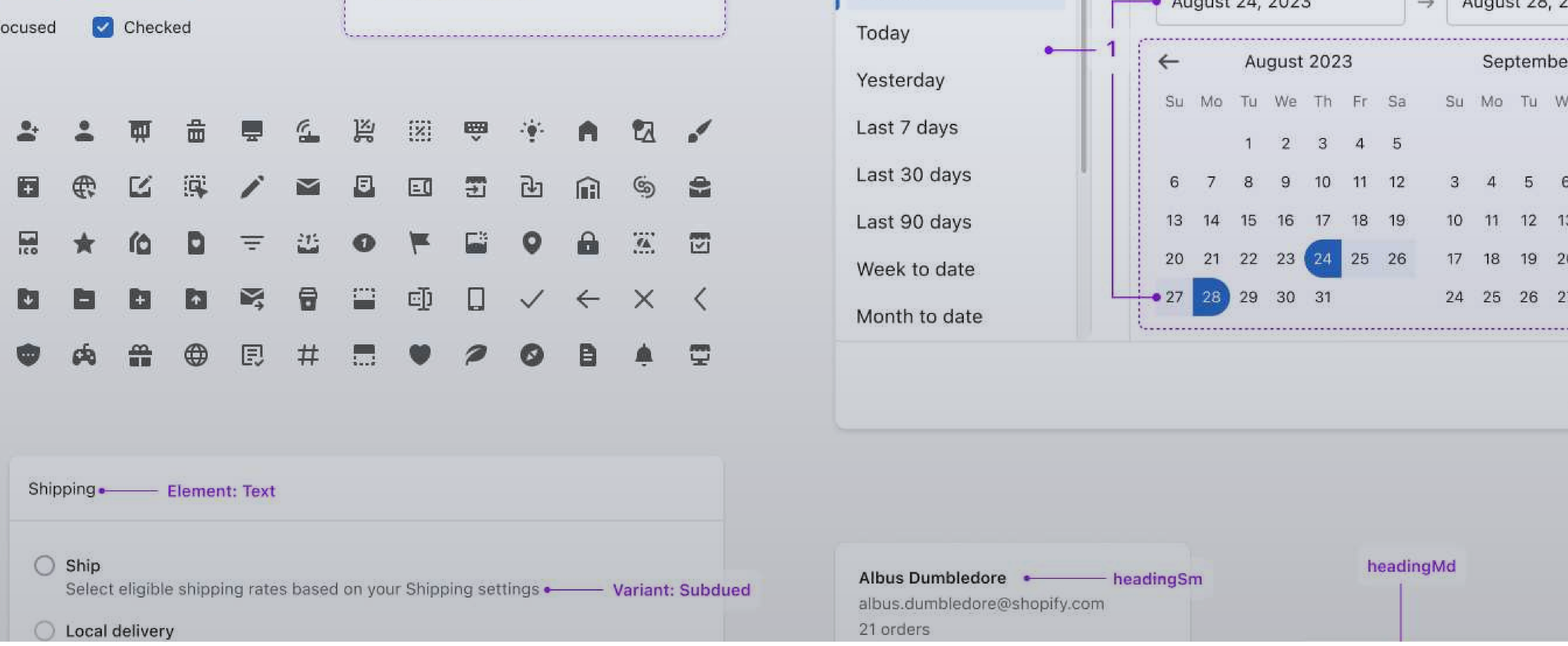
## Book structure

To make it easier to understand, I split the book into two parts: Theory and Practice.

The theory section covers detailed explanations, tips, and best practices.

In the practice section, you’ll find hands-on guidance for using colors, fonts, and components in Figma, which I personally use in my work.

# Theory



# Introduction to Design System

Before we start, let’s first get a grasp of what a design system is and what it does.

# What is a design system?

In my opinion, a design system is like a rulebook for design. It includes guidelines and reusable pieces that help keep designs consistent and easy to create across different projects. It's basically a toolkit that makes designing things easier and more organized.

# Why do you need a design system?

Let's explore the benefits that implementing a design system into your project can offer.

### Consistency

Design systems help keep everything looking and feeling the same across different parts of a product. This makes it easier for users because they know what to expect and how things will work wherever they go.

### Save time

Design systems allow you to save time and money during product development. Usually, you can follow the rules that are already set and use things that are already made instead of making them all over again.

### Expand with ease

Design systems can grow with your project, making it easy for teams to adjust and add more if needed. You can smoothly add new features and updates while keeping the design consistent.

### Teamwork

Design systems speed up the design and development process by offering a unified set of design assets. This helps designers and developers work together more effectively, saving time and resources.

It acts as a shared language for everyone involved, including designers, developers, and other stakeholders.

# Is a design system for everyone?

I’m sure that design systems are really important for projects that will keep growing and changing, using lots of different components and getting regular updates.

If you’re making a small landing page or just a few pages that won’t need many changes or updates, creating a whole design system might be more effort than it’s worth. Instead, a simple UI Kit with the basics could be too effort consuming.

# Structure of a design system

Most design systems consist of four key parts:

### Foundation

Foundation includes guidelines for typography, colors, dimensions, grids, and more.

### Components

Components are the building blocks of our design system. Each one fulfills a specific interaction or UI need, designed to work together seamlessly, creating patterns and intuitive user experiences.

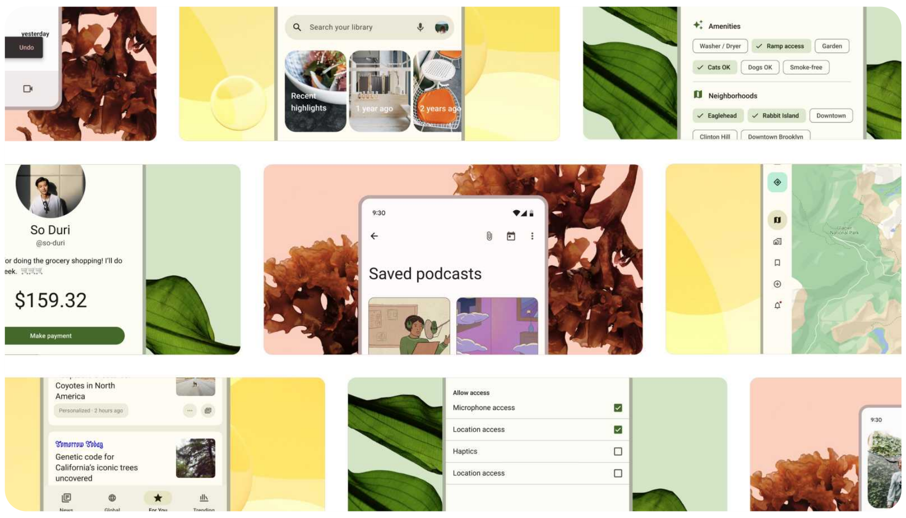
### Patterns

Patterns are best practice solutions guiding users to achieve their goals effectively. They present reusable combinations of components and templates designed to fulfill common user objectives through defined sequences and flows.

### Resources

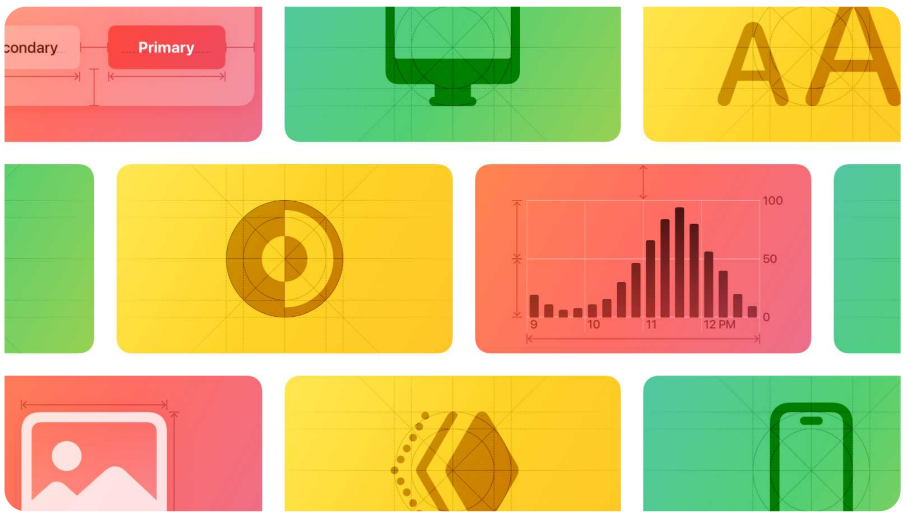
Resources are a collection of articles, tutorials, tools, and plugins designed to simplify working with the design system.

# Best examples of design systems



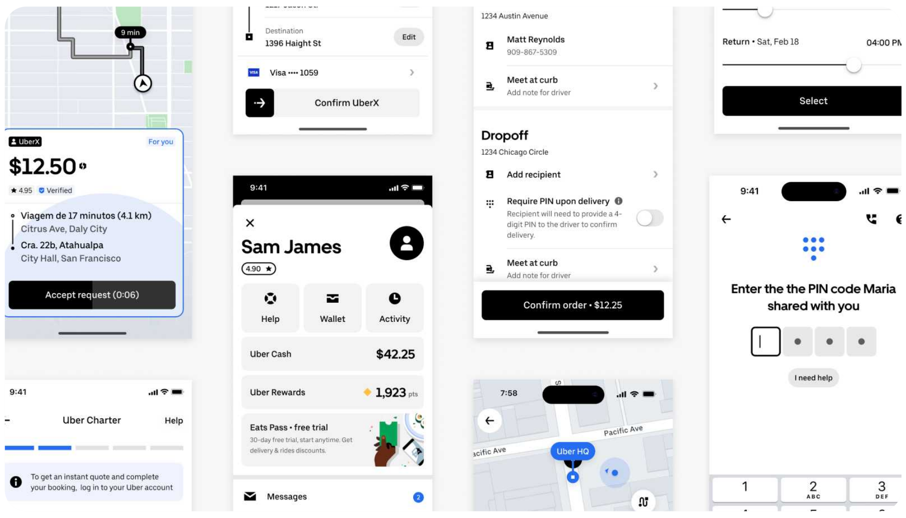
## Material (Google)

Google's open-source design system. Design and build beautiful, usable products.



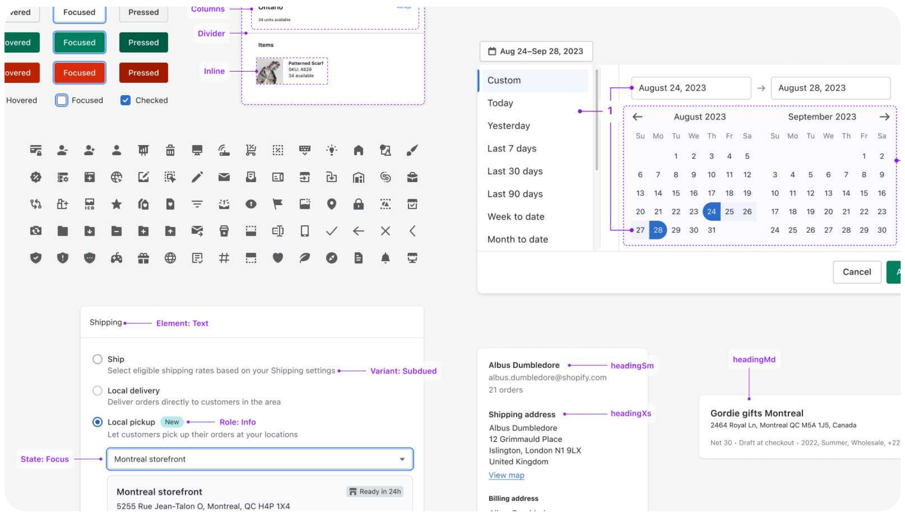
## HIG (Apple)

The Human Interface Guidelines offers guidance and best practices for designing exceptional user experiences across all Apple platforms.



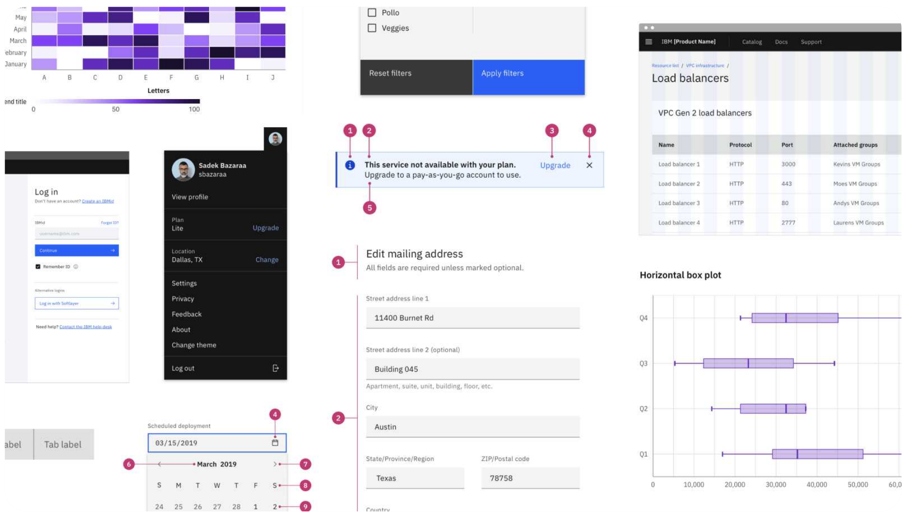
## Base (Uber)

The Base design system defines the foundations of user interfaces across Uber's ecosystem of products & services.



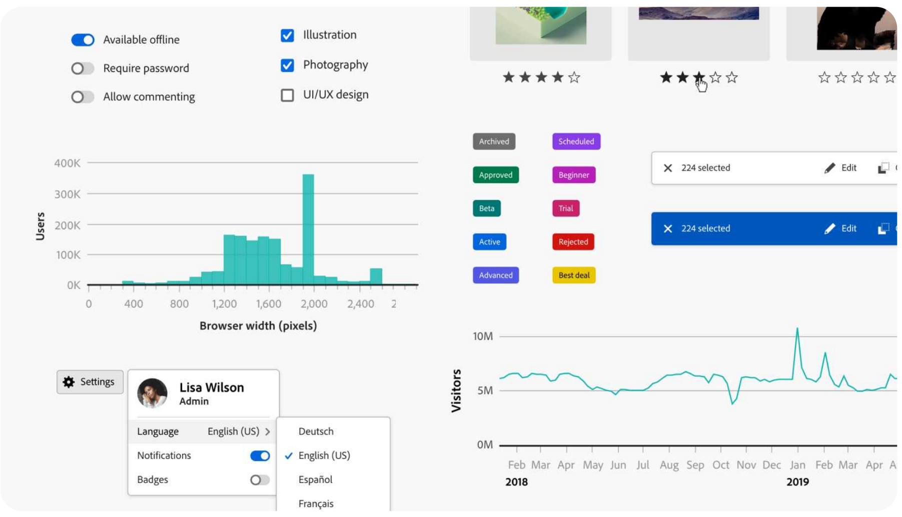
## Polaris (Shopify)

Shape the merchant experience for Shopify's core product, the admin.



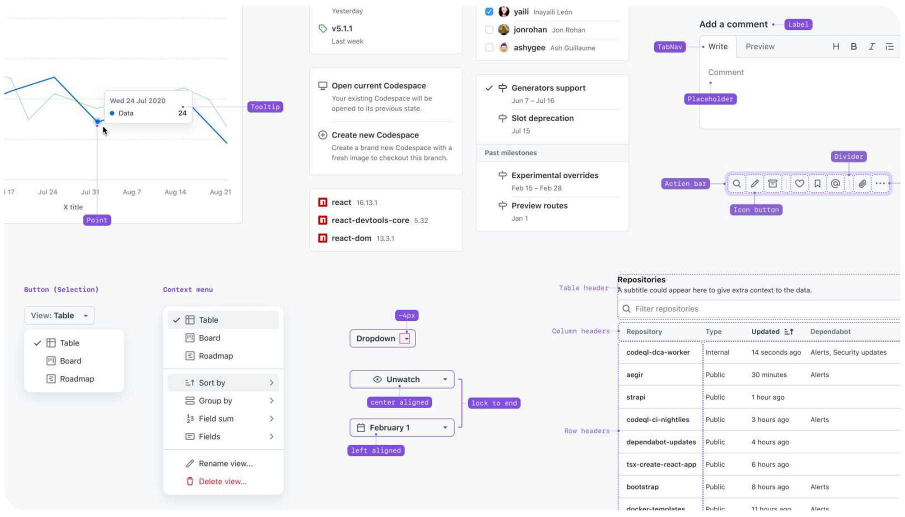
## Carbon Design System (IBM)

Carbon is IBM's open source design system for products and digital experiences.



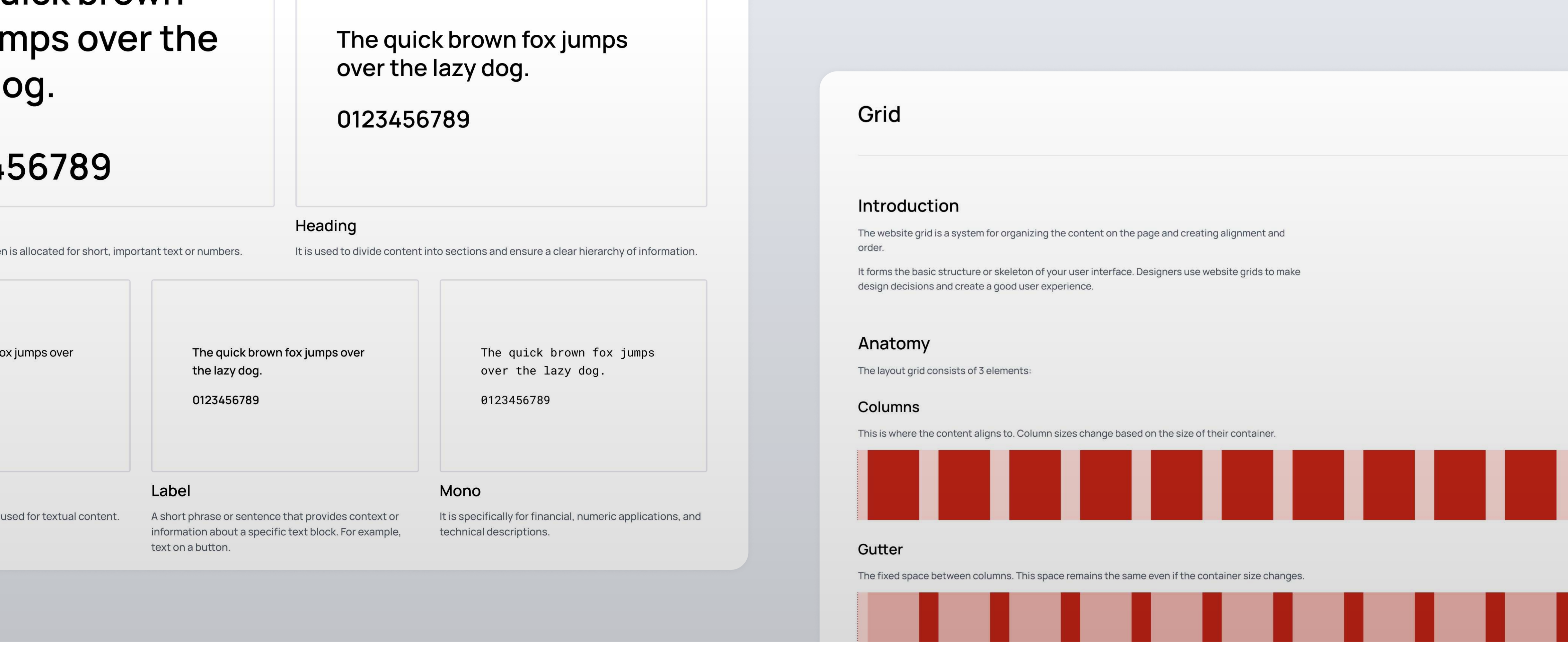
## Spectrum (Adobe)

Spectrum provides components and tools to help product teams work more efficiently, and to make Adobe's applications more cohesive.



## Primer (GitHub)

Primer is a set of guidelines, principles, and patterns for designing and building UI at GitHub. It provides a shared language and standardized approach to delivering cohesive experiences.



mps over the
og.

456789

n is allocated for short, important text or numbers.

The quick brown fox jumps
over the lazy dog.

0123456789

Heading

It is used to divide content into sections and ensure a clear hierarchy of information.

ox jumps over

used for textual content.

The quick brown fox jumps over
the lazy dog.

0123456789

Label

A short phrase or sentence that provides context or information about a specific text block. For example, text on a button.

```
The quick brown fox jumps
over the lazy dog.

0123456789
```

Mono

It is specifically for financial, numeric applications, and technical descriptions.

Grid

Introduction

The website grid is a system for organizing the content on the page and creating alignment and order.

It forms the basic structure or skeleton of your user interface. Designers use website grids to make design decisions and create a good user experience.

Anatomy

The layout grid consists of 3 elements:

Columns

This is where the content aligns to. Column sizes change based on the size of their container.

Gutter

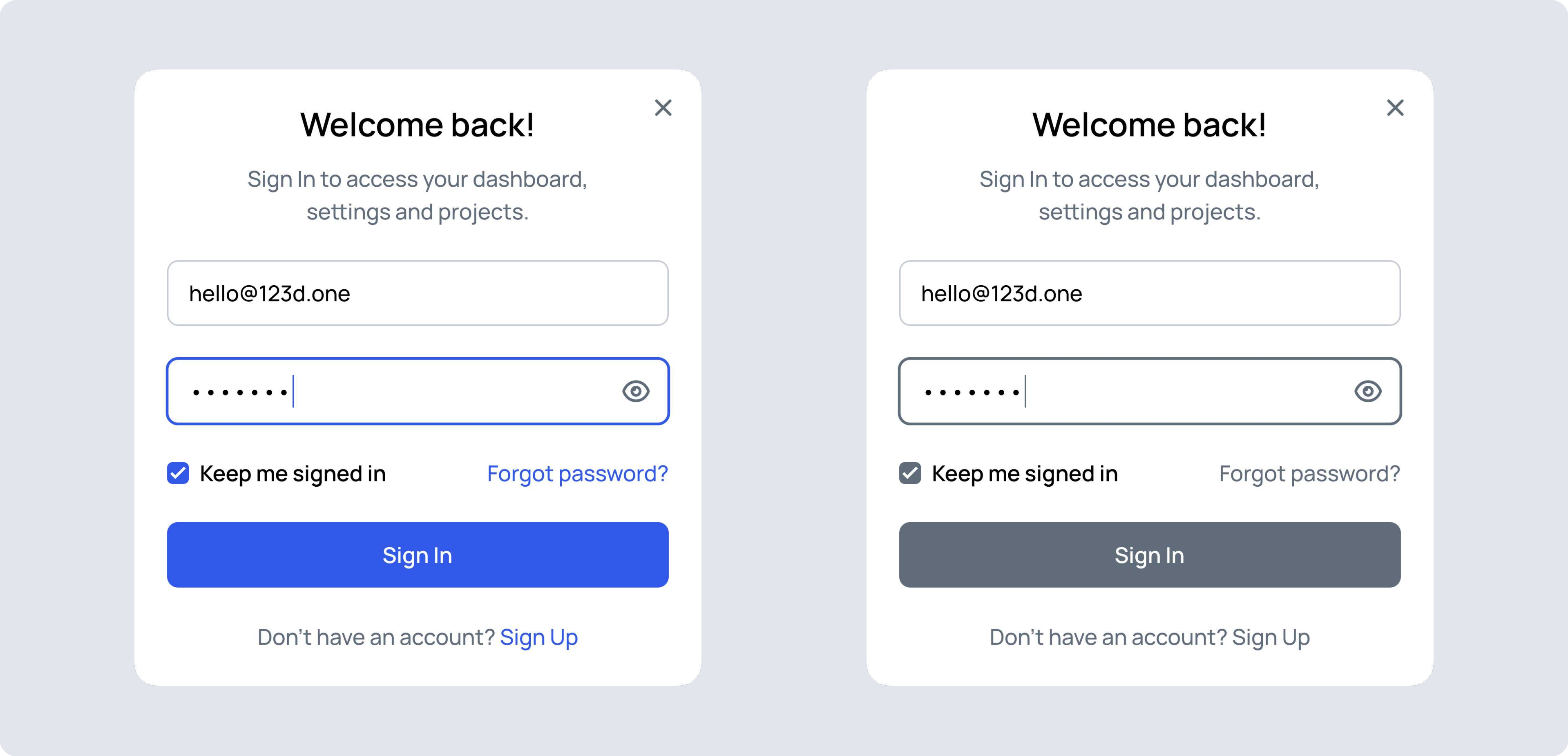
The fixed space between columns. This space remains the same even if the container size changes.

# Foundation

The foundation of a design system includes typography, colors, icons, spacing, grids, and design principles. These elements ensure consistency and user-friendly experiences, helping designers create cohesive interfaces.

# Colors

Colors in design systems affect the way our interface looks. They are not just random colors but a set of rules of how colors are used, how they work together, and how they change depending on different situations.



Welcome back!

Sign In to access your dashboard, settings and projects.

hello@123d.one

•••••••

☑ Keep me signed in

Forgot password?
Sign In
Don't have an account? Sign Up

Welcome back!

Sign In to access your dashboard, settings and projects.

hello@123d.one

•••••••

☑ Keep me signed in

Forgot password?
Sign In
Don't have an account? Sign Up

Comparison of a modal window with various color schemes.

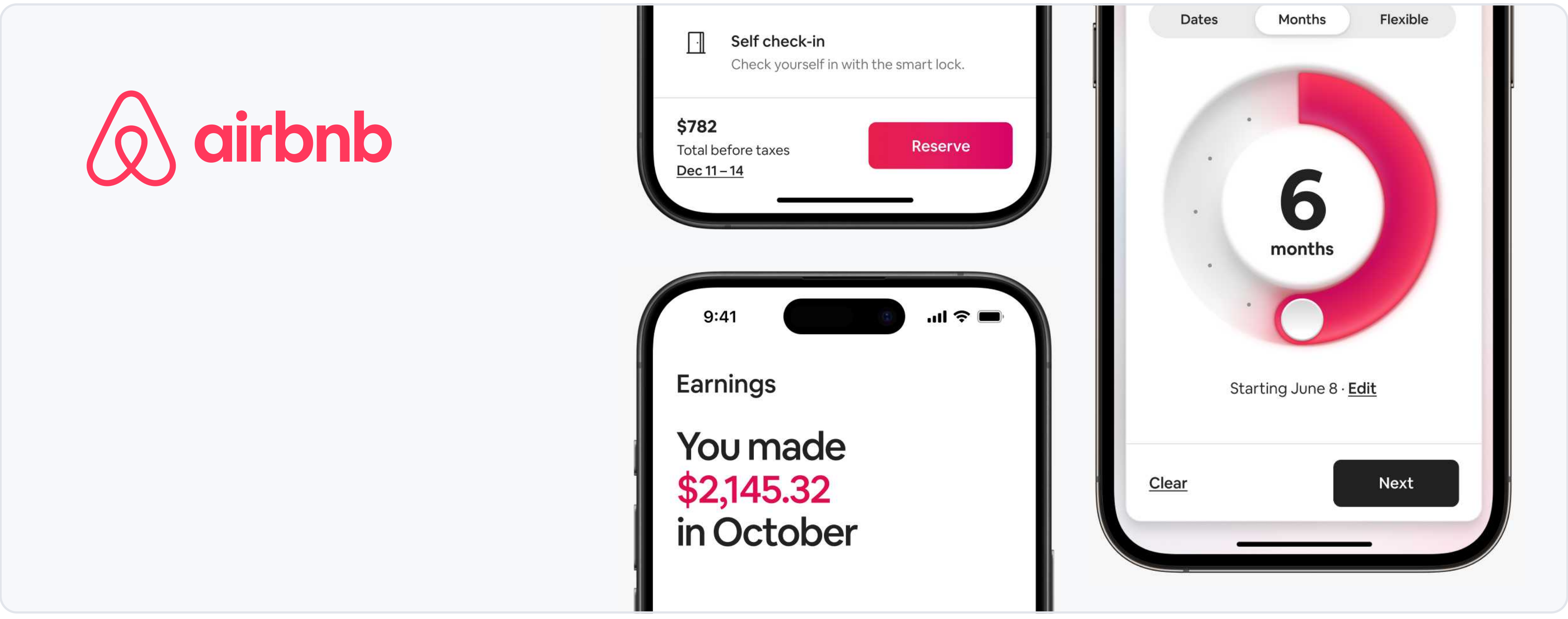
## Base colors

In any design system, it's logical to highlight five main groups of colors that need to be decided first:

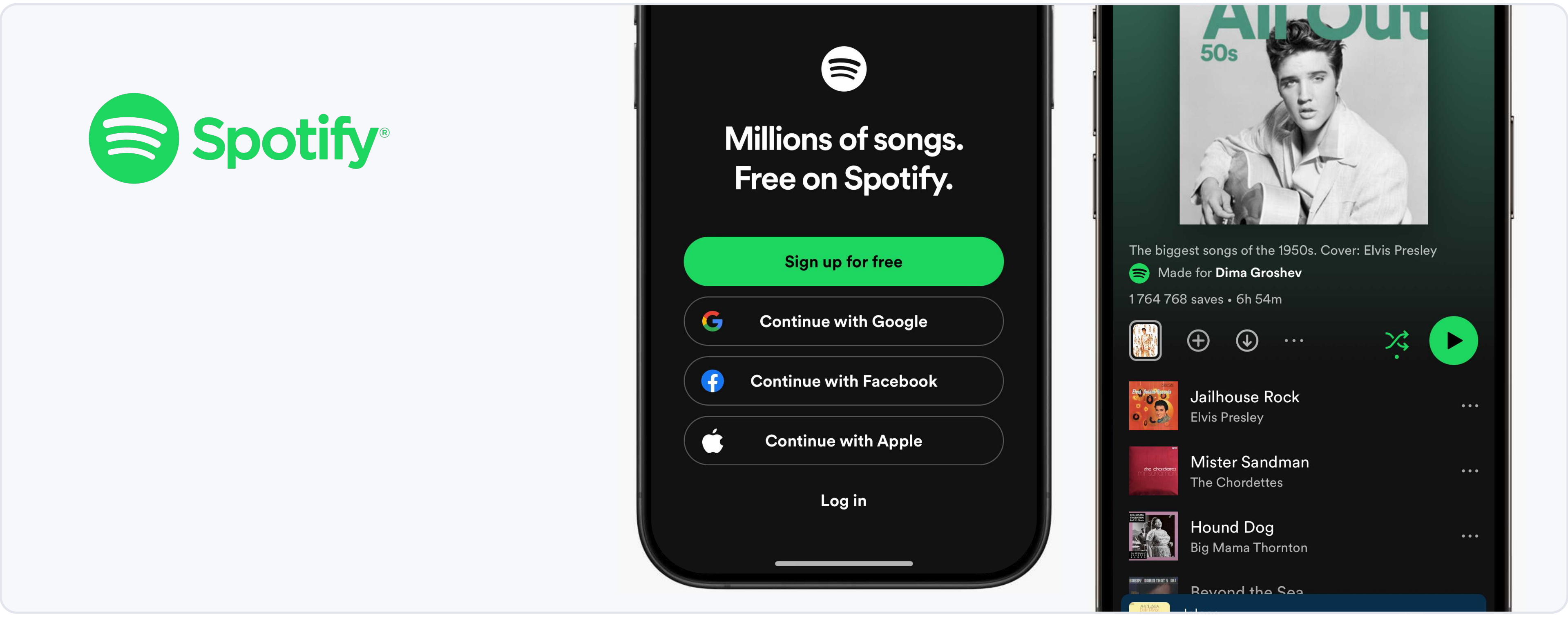
- Primary color
- Black and white colors
- Secondary colors
- Neutral colors
- Semantic or state colors

## Primary color

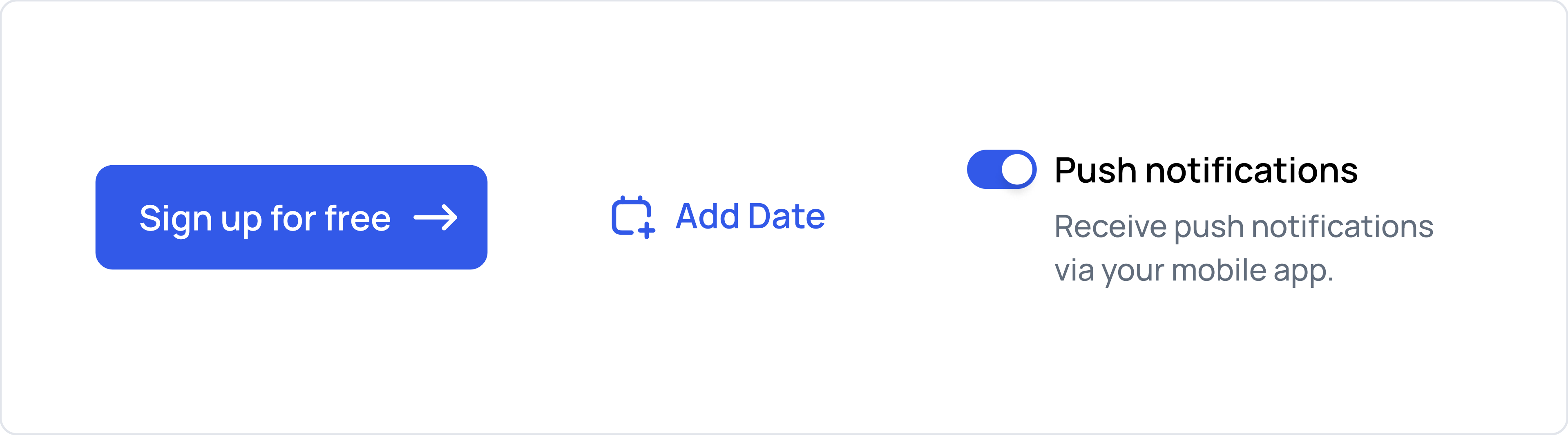
This color gives your interface personality and is its main focus. It typically defines a brand's identity. By using this corporate color as the primary color, people can recognize the brand more easily.



Airbnb's primary color.



Spotify's primary color.

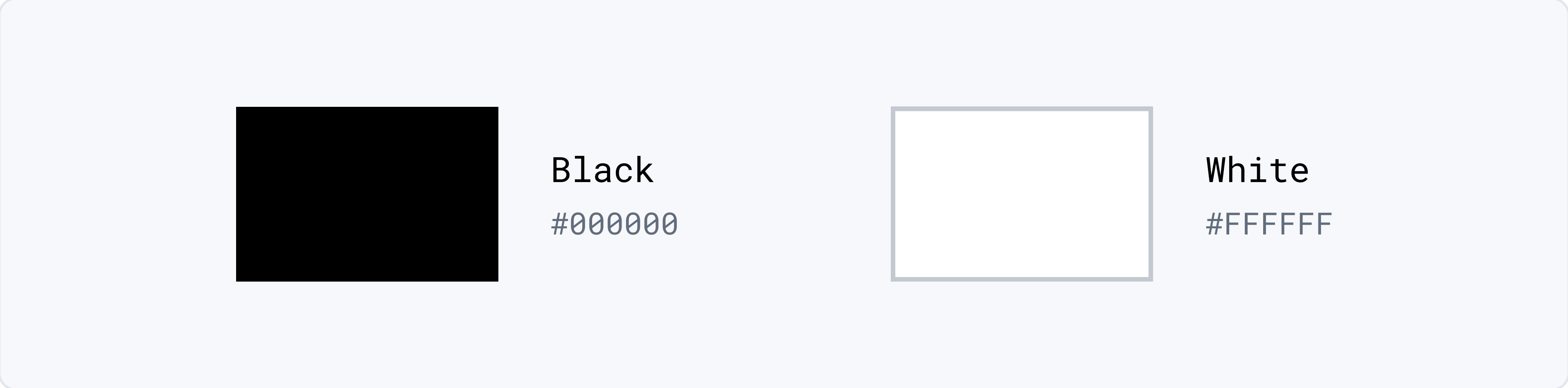


Primary colors are used for buttons, links, and other components.

### Black and white colors

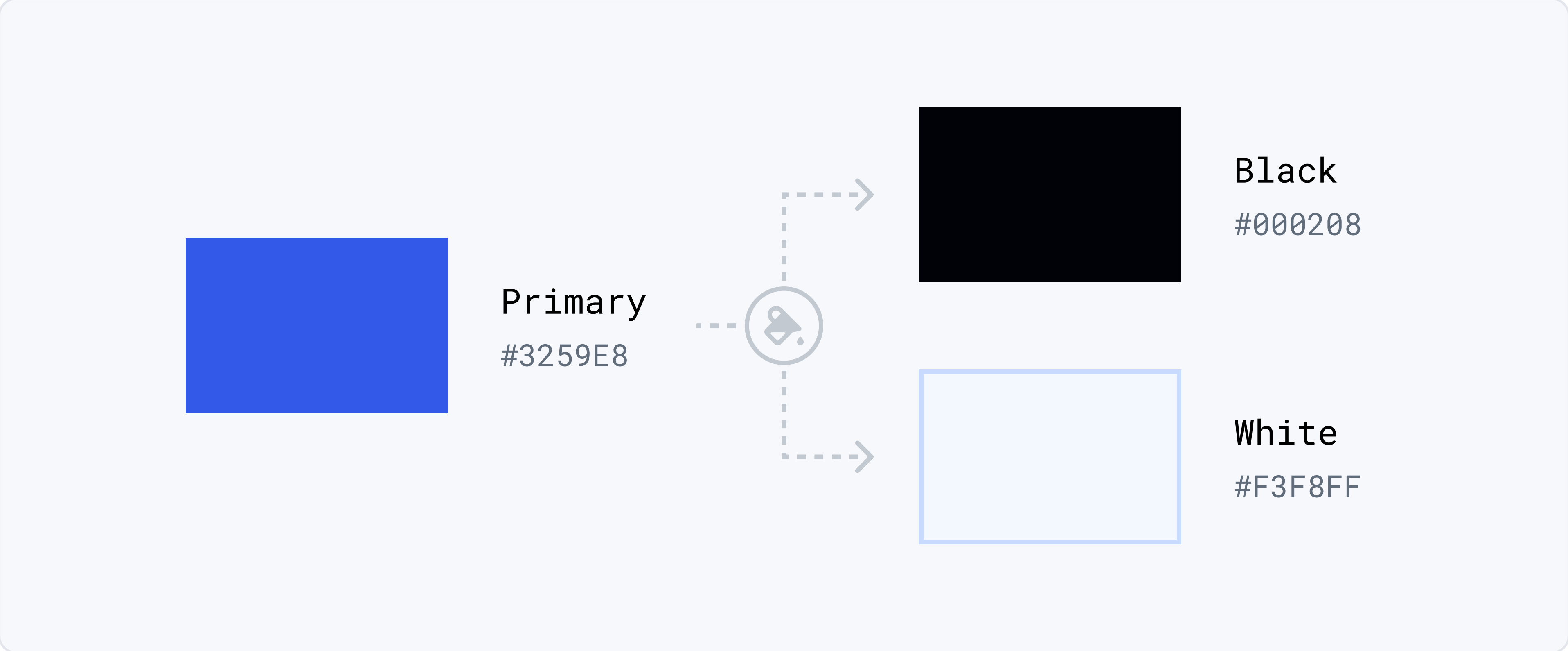
When working on a website or application, it is important to decide on the basic colors. These colors primarily determine the color of the text and background.

In most cases, black will be ■ #000000 and white will be □ #FFFFFF

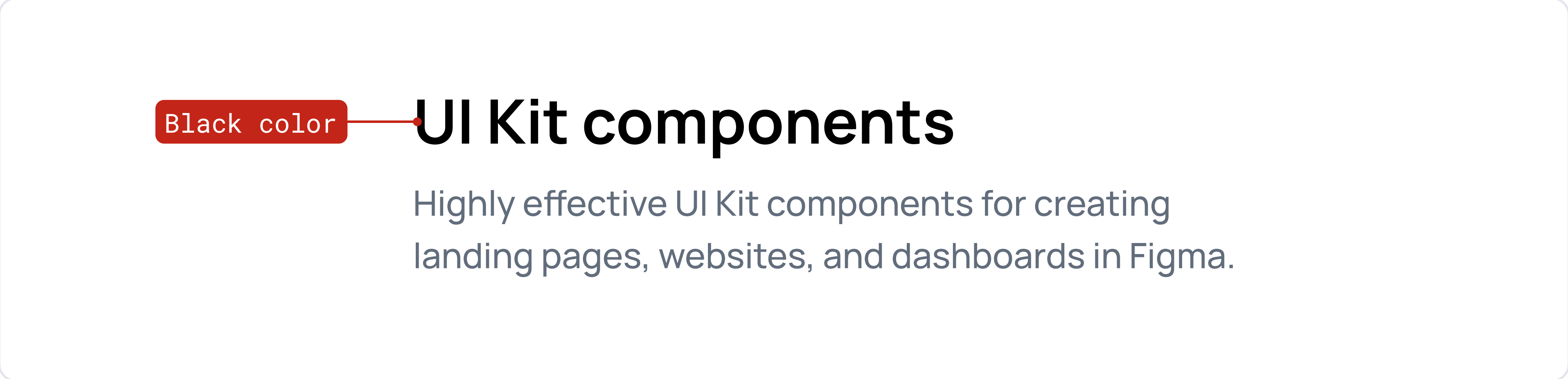


Black and white colors.

However, you can also align your colors with your primary color to make them more complementary.



Enhance design harmony by blending black and white with the primary color.



UI Kit components

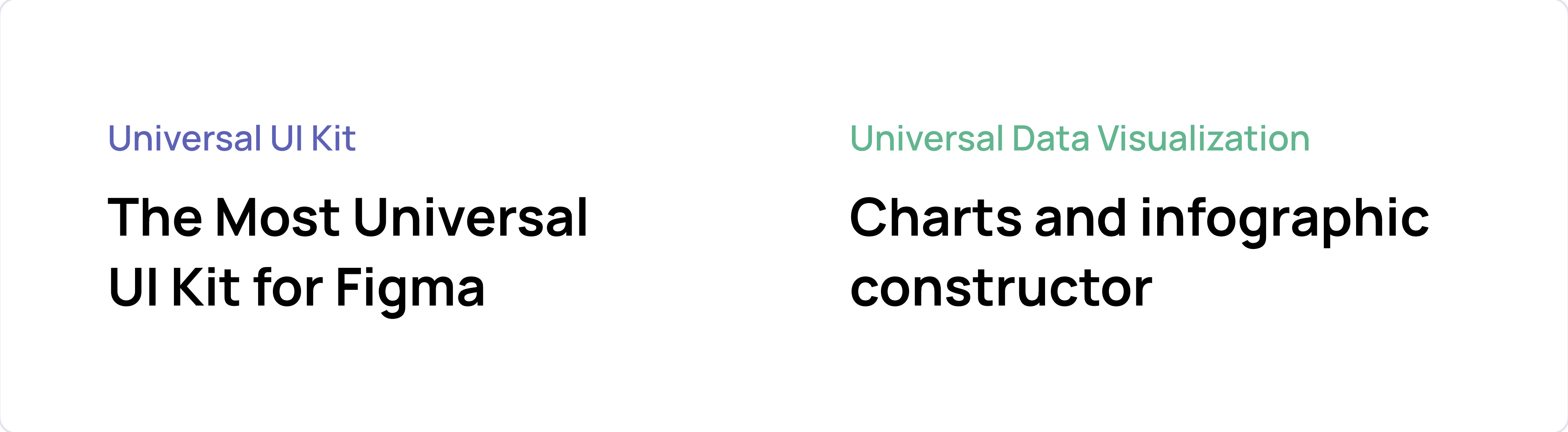
Highly effective UI Kit components for creating landing pages, websites, and dashboards in Figma.

Using black color for the title.

## Secondary colors

These colors complement the primary color. I recommend using them carefully to accentuate rather than overpower the primary color.

Feel free to apply these colors to icons, subheadings, and backgrounds.



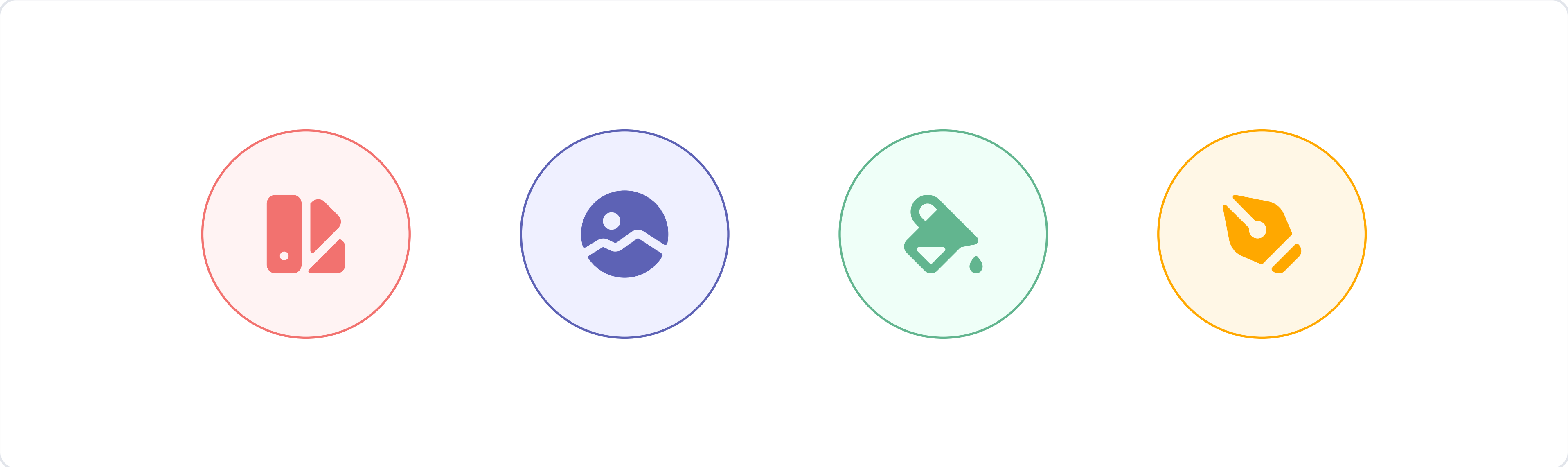
Universal UI Kit

**The Most Universal UI Kit for Figma**

Universal Data Visualization

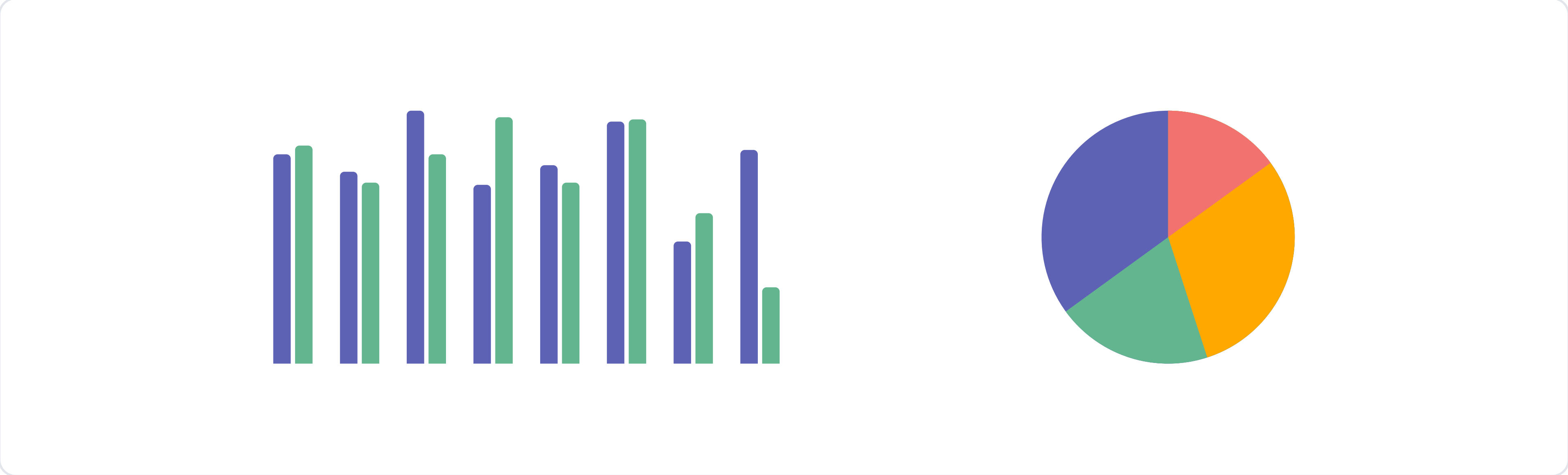
**Charts and infographic constructor**

Using a secondary color palette for subheadings.



Using a secondary color palette for icons.

These colors work well for graphs and illustrations too.

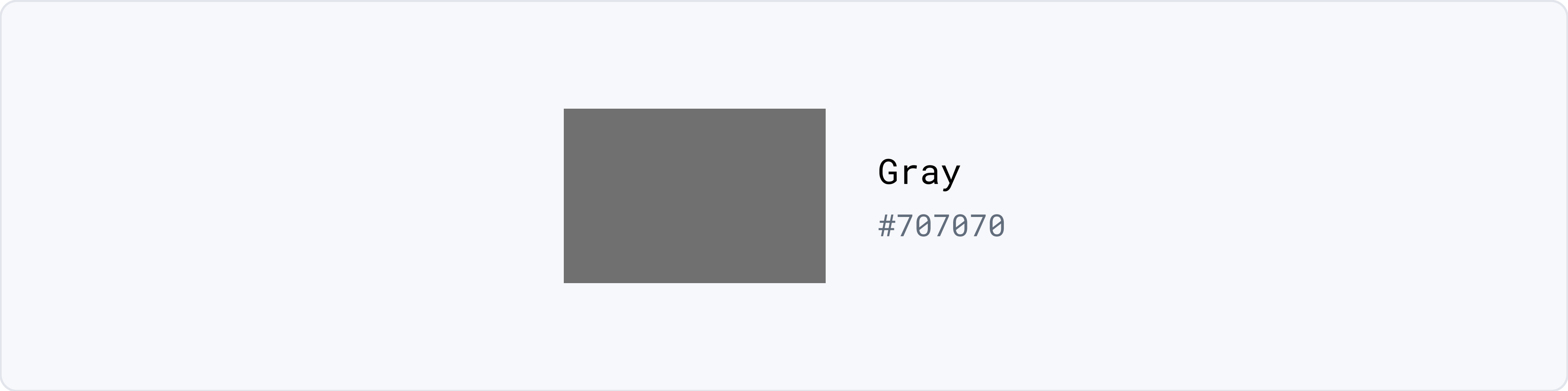


Using a secondary color palette for charts.

## Neutral color

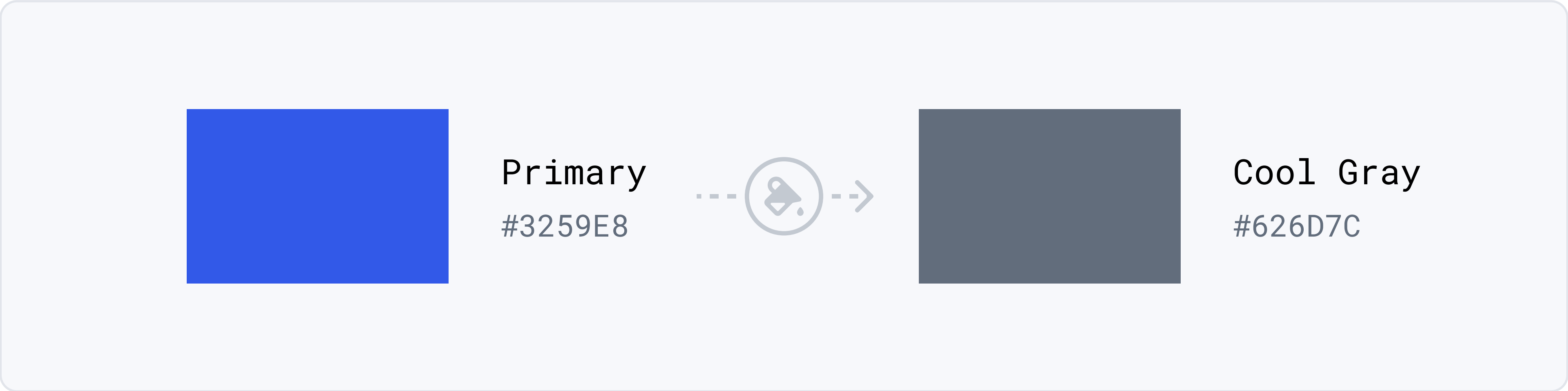
In most cases, this color is a shade of gray. It's used for secondary texts, borders, backgrounds, etc., making up the bulk of your interface.

Neutral colors are the ones that are most commonly used in the interface and make up most of it.

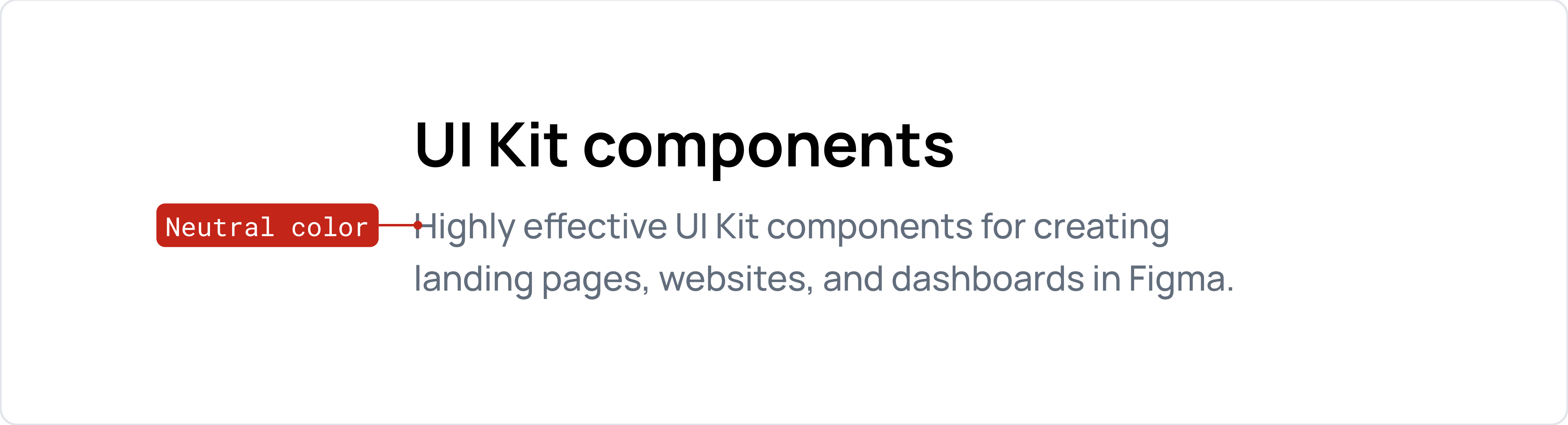


Neutral gray color.

Neutral colors, along with white and black, can be matched with the primary color for better harmony.



Enhance design harmony by blending gray with the primary color.



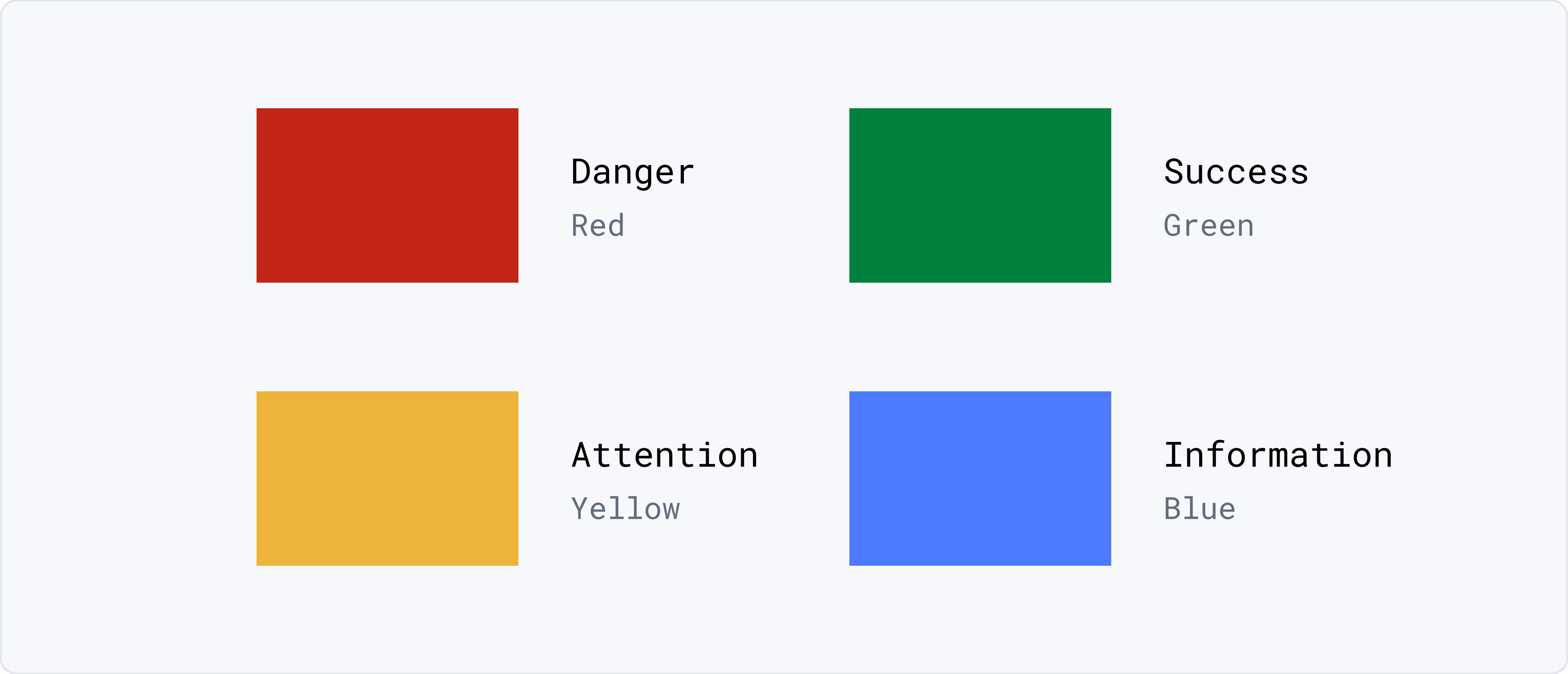
UI Kit components

Highly effective UI Kit components for creating landing pages, websites, and dashboards in Figma.

Using a neutral color for the secondary text.

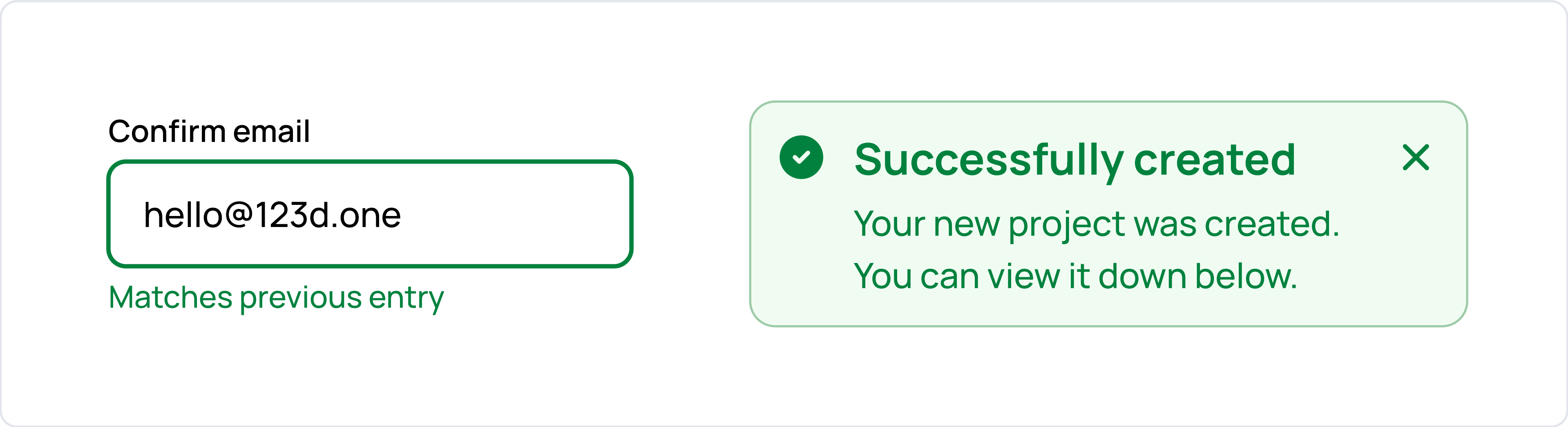
## Semantic or state colors

These colors communicate purpose and help users understand messages.



Typical state colors.

For example, green signifies success or confirmation.



Confirm email

hello@123d.one

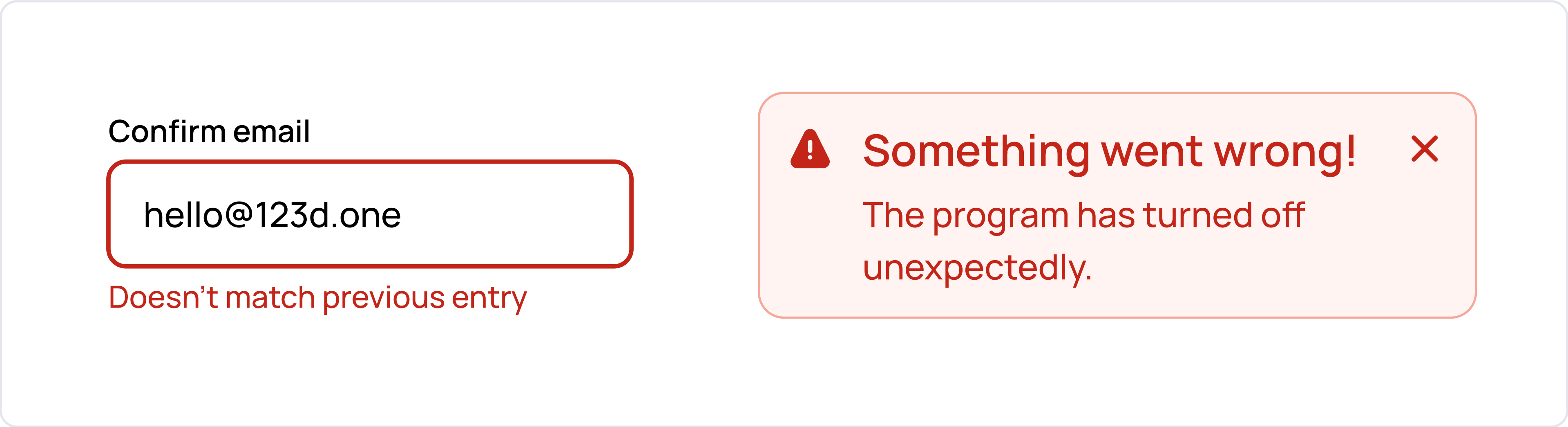
Matches previous entry

Successfully created

Your new project was created. You can view it down below.

Using green color in the components.

Red, on the other hand, indicates an error and signals that something has gone wrong.



Confirm email

hello@123d.one

Doesn’t match previous entry

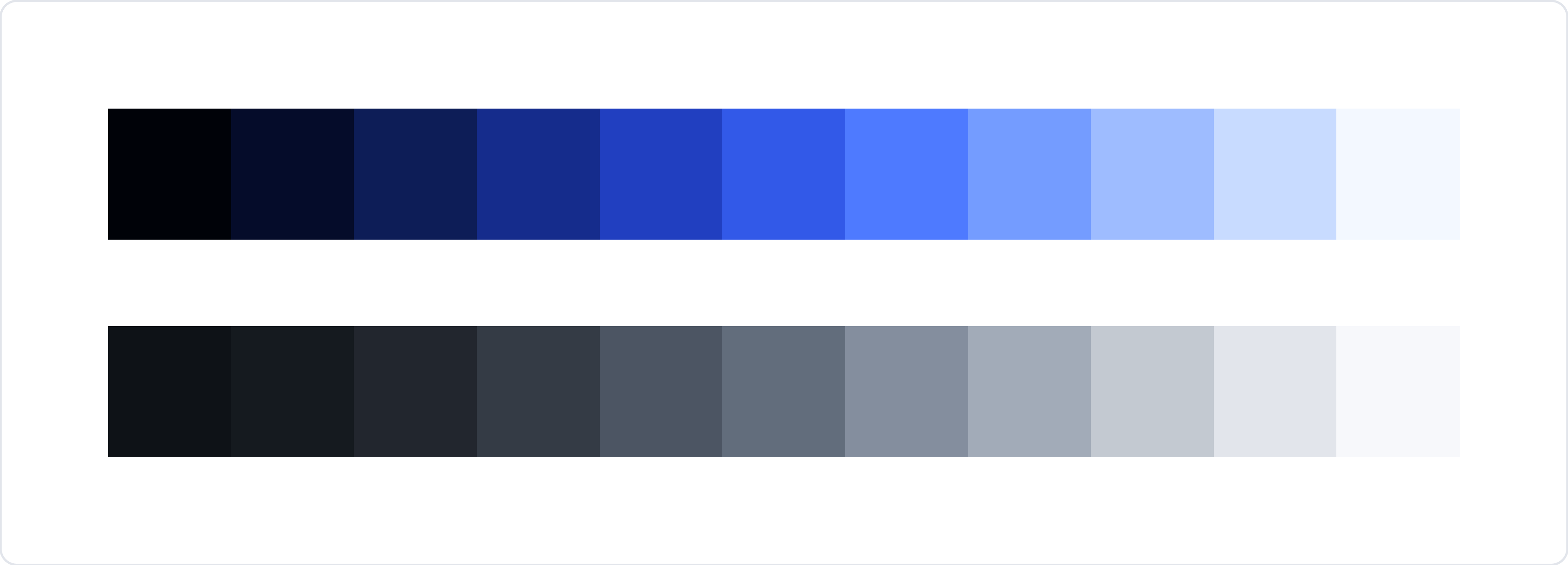
Something went wrong!

The program has turned off unexpectedly.

Using red color in the components.

# Palettes

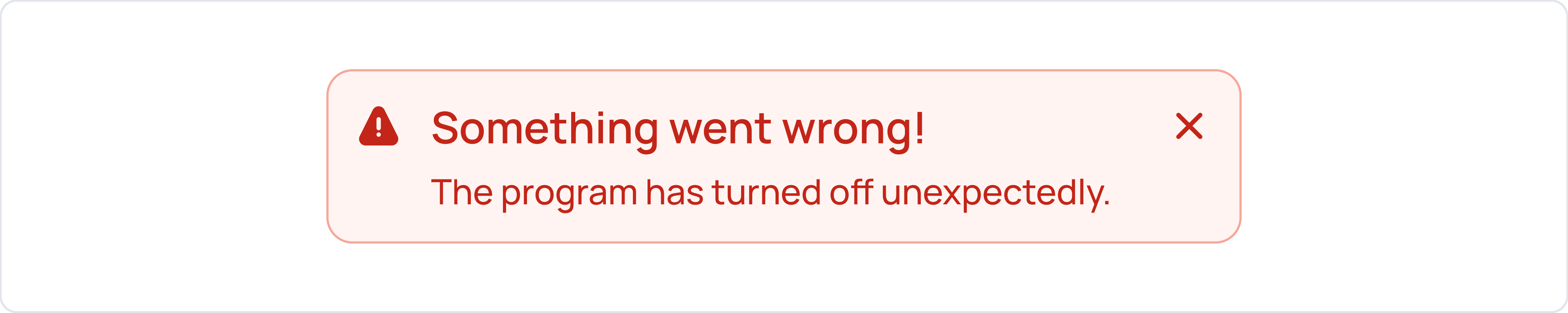
After deciding on the main colors for our design, we need to make complete palettes. Base colors won't be enough; we require a variety of light and dark shades.



Color palettes.

Having different shades allows more design options, like showing different states of elements.

For example, to highlight an error message, we might use a lighter shade of red for the background.

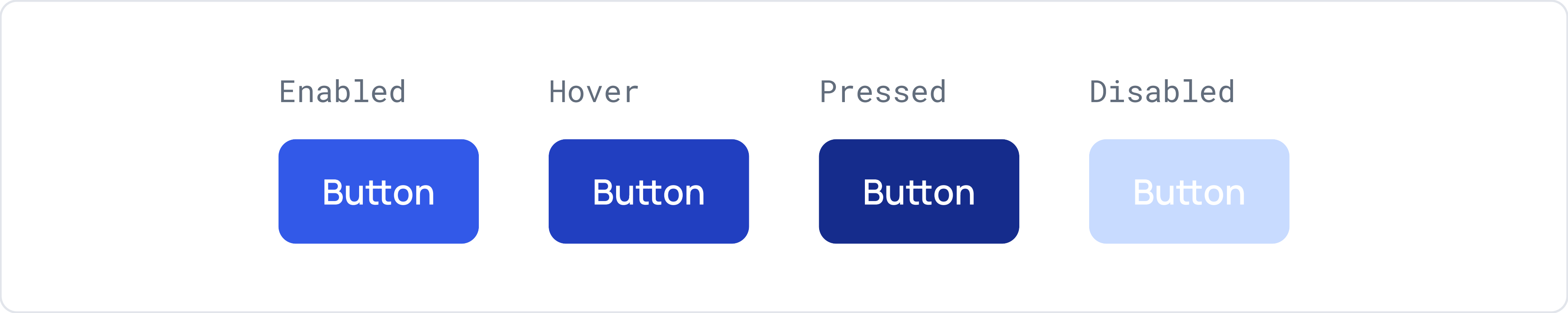


Something went wrong!

The program has turned off unexpectedly.

Error message.

To create different button states, you'll need a primary color palette.



Button states.

You can choose your palette manually or use special websites and software. There's no universal approach. It depends on your needs. Your palette might include 6, 10, or 12 colors — it's up to you.

Here are some services that can help:

### Coolors

Generate or browse beautiful color combinations for your designs.

### OKlch palette generator

OKLCH is a new way to encode colors (like hex, RGBA, or HSL).

### Atmos

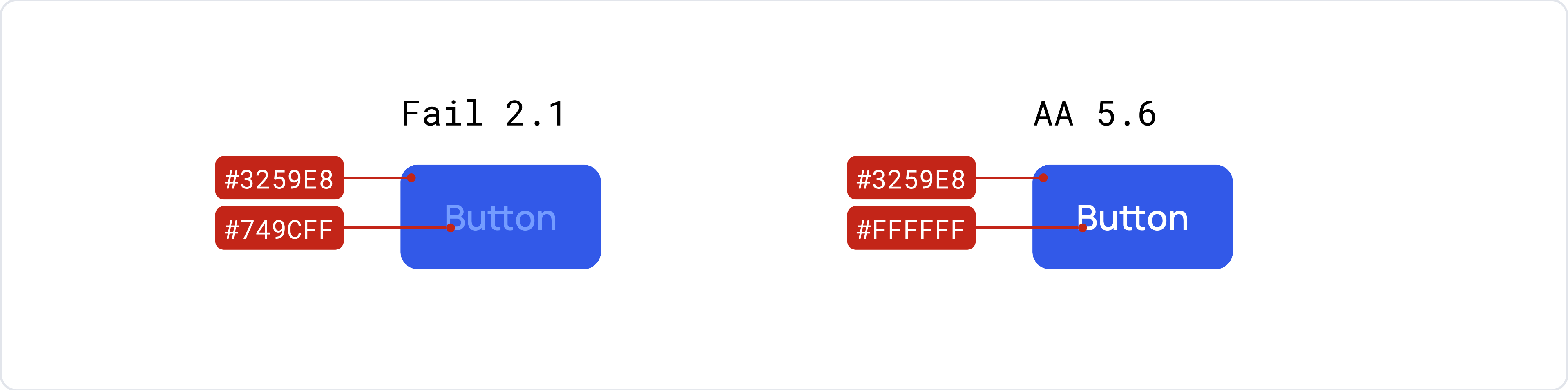
Atmos is a toolbox which specializes in creating UI palettes.

### OKLCH Color Variations

Use this Plugin to generate Color Palettes with Variations in the OKLCH Color Space.

## Testing your color palettes

Testing and refining your color palettes is crucial for a design system. Make sure to try out all shades on components and adjust them as needed for a harmonious look.



Check the contrast between different color combinations.

Follow the Web Content Accessibility Guidelines (WCAG) to make your web content accessible to everyone, including people with disabilities and users on mobile devices.

|  |  |
|---|---|
| **AAA (7.0)** | Best for long texts. |
| **AA (4.5)** | Safe for all text sizes in your designs. |
| **AA Large (3.0)** | Good for big and bold texts, primary buttons, and icons. |
| **Fail (1.5)** | Don’t use it for reading; it’s okay for logos, graphics, and some design elements with no strict contrast rules. |

There are many websites and plugins available that allow you to test your colors based on these guidelines.

Use Contrast

Figma plugin for quick access to WCAG color contrast ratios

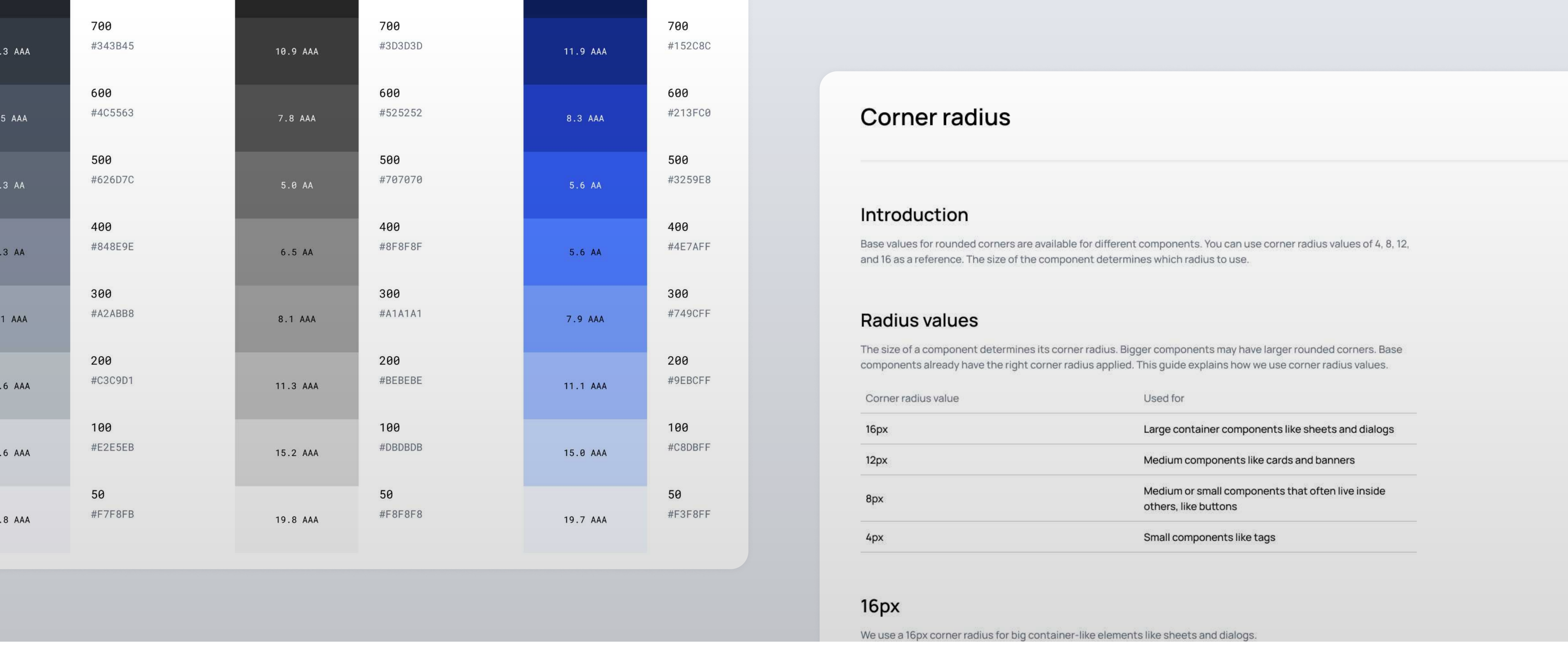
Colour Contrast Checker

Check the contrast between different colour combinations against WCAG standards.

# Practice

# About this part

For me, learning from other people's experience by observing their work has always been interesting. When I started working as a designer, I searched for tutorials where designers shared their workflow. In this part of the book, I've described in detail how we work with texts, colors, components, and other elements of the Universal Design System. I hope my experience proves useful for you in your work.



| | | | | | |
|---|---|---|---|---|---|
| .3 AAA | 700<br>#343B45 | 10.9 AAA | 700<br>#3D3D3D | 11.9 AAA | 700<br>#152C8C |
| 5 AAA | 600<br>#4C5563 | 7.8 AAA | 600<br>#525252 | 8.3 AAA | 600<br>#213FC0 |
| .3 AA | 500<br>#626D7C | 5.0 AA | 500<br>#707070 | 5.6 AA | 500<br>#3259E8 |
| .3 AA | 400<br>#848E9E | 6.5 AA | 400<br>#8F8F8F | 5.6 AA | 400<br>#4E7AFF |
| 1 AAA | 300<br>#A2ABB8 | 8.1 AAA | 300<br>#A1A1A1 | 7.9 AAA | 300<br>#749CFF |
| .6 AAA | 200<br>#C3C9D1 | 11.3 AAA | 200<br>#BEBEBE | 11.1 AAA | 200<br>#9EBCFF |
| .6 AAA | 100<br>#E2E5EB | 15.2 AAA | 100<br>#DBDBDB | 15.0 AAA | 100<br>#C8DBFF |
| .8 AAA | 50<br>#F7F8FB | 19.8 AAA | 50<br>#F8F8F8 | 19.7 AAA | 50<br>#F3F8FF |

Corner radius

Introduction

Base values for rounded corners are available for different components. You can use corner radius values of 4, 8, 12, and 16 as a reference. The size of the component determines which radius to use.

Radius values

The size of a component determines its corner radius. Bigger components may have larger rounded corners. Base components already have the right corner radius applied. This guide explains how we use corner radius values.

| Corner radius value | Used for |
|---|---|
| 16px | Large container components like sheets and dialogs |
| 12px | Medium components like cards and banners |
| 8px | Medium or small components that often live inside others, like buttons |
| 4px | Small components like tags |

16px

We use a 16px corner radius for big container-like elements like sheets and dialogs.

# Foundation

In this section, we will explore how the key elements of the foundation were created in the Universal Design System.

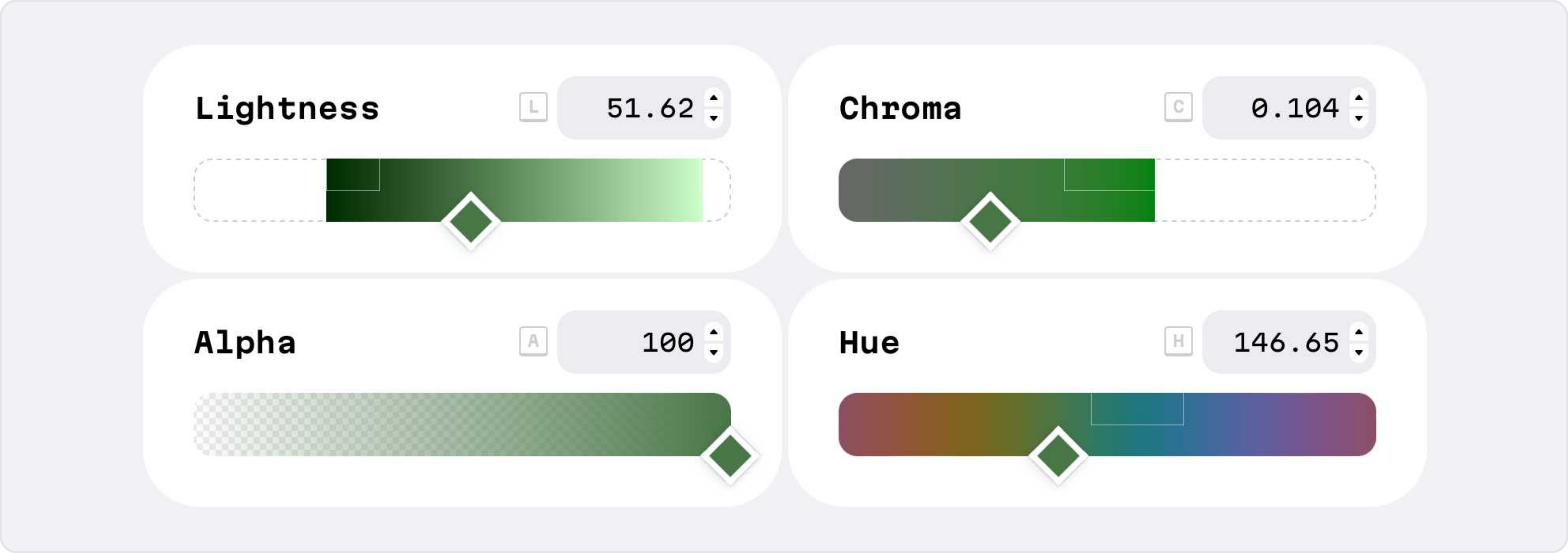
# Colors

## OKLCH color space

In the latest version of the Universal Design System we used the OKLCH color space to create the palette.

### What is OKLCH?

The Oklab and OKLCH color spaces were created by Björn Ottosson in 2020.



Lightness L 51.62

Chroma C 0.104

Alpha A 100

Hue H 146.65

OKLCH is a new way to define CSS colors. In OKLCH (L C H / a) each item corresponds as follows:

**L** is perceived lightness (0%-100%). “Perceived” means that it has consistent lightness for our eyes, unlike L in HSL.

**C** is chroma, from gray to the most saturated color.

**H** is the hue angle (0-360).

**a** is opacity (0-1 or 0-100%).

The benefits of OKLCH:

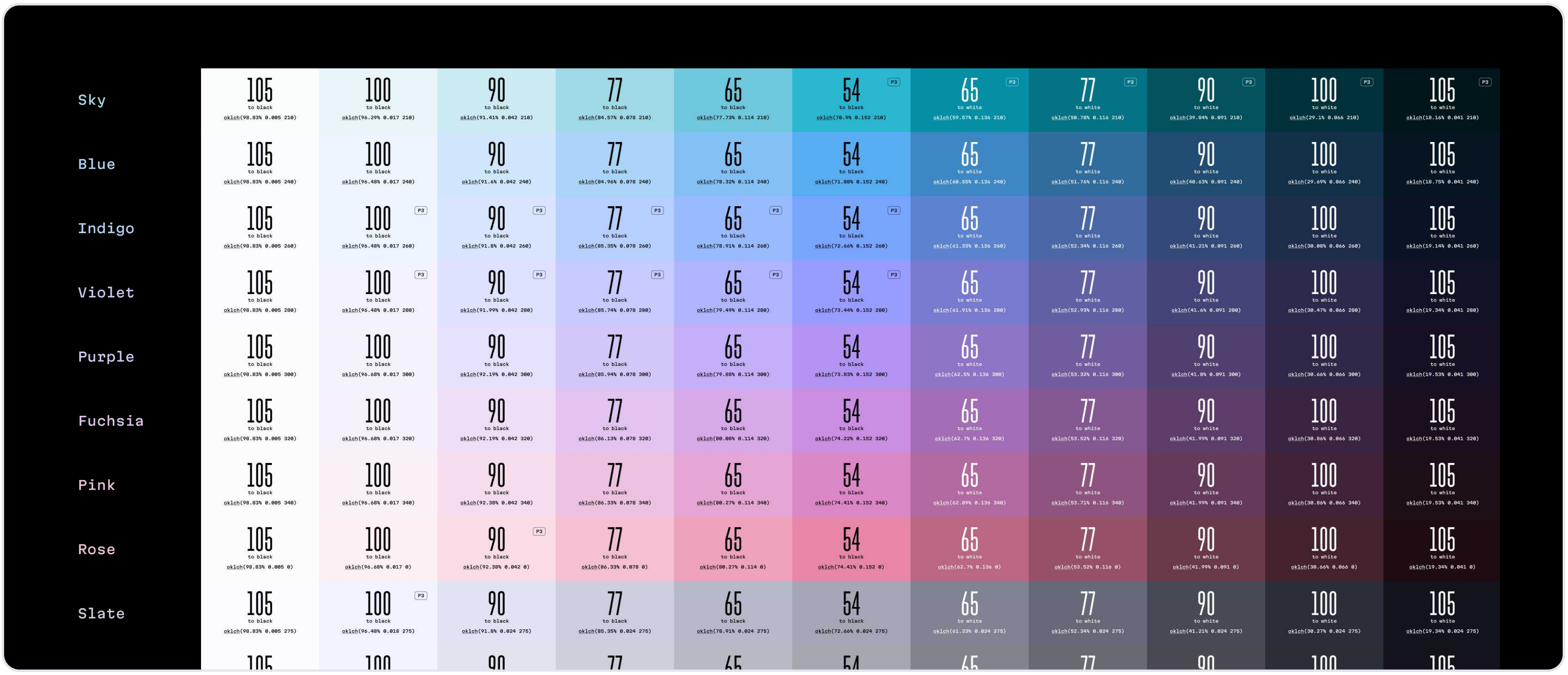
- OKLCH frees designers from the need to manually choose every color. They can set a formula, choose a few colors, and an entire design system palette is automatically generated.
- OKLCH can be used for wide-gamut P3 colors. For instance, new devices (like those from Apple) can display more colors than old sRGB monitors, and we can use OKLCH to specify these new colors.
- Unlike HSL, OKLCH is better for color modifications and palette generation. It uses perceptual lightness, which means there aren't any unexpected results.

If you want to learn more about OKLCH, here's an excellent article from Evil Martians.

## Equal contrast

Traditional color palettes offer a range of lightness levels for each color, such as Blue 100 and Yellow 400. However, switching the primary accent color from blue to yellow often results in an unreadable text.

With the OKLCH color space, you can adjust the colors in the palette so that when you switch from one color to another (Green 500, Blue 500, Yellow 500, etc.), there will be an identical level of contrast. This allows you to freely swap any '500' color for another, ensuring consistent readability for texts and buttons.



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Sky | 105<br>to black<br>oklch(98.83% 0.005 210) | 100<br>to black<br>oklch(96.29% 0.017 210) | 90<br>to black<br>oklch(91.41% 0.042 210) | 77<br>to black<br>oklch(84.57% 0.078 210) | 65<br>to black<br>oklch(77.73% 0.114 210) | 54 P3<br>to black<br>oklch(70.9% 0.152 210) | 65 P3<br>to white<br>oklch(59.57% 0.136 210) | 77 P3<br>to white<br>oklch(50.78% 0.116 210) | 90 P3<br>to white<br>oklch(39.84% 0.091 210) | 100 P3<br>to white<br>oklch(29.1% 0.066 210) | 105 P3<br>to white<br>oklch(18.16% 0.041 210) |
| Blue | 105<br>to black<br>oklch(98.83% 0.005 240) | 100<br>to black<br>oklch(96.48% 0.017 240) | 90<br>to black<br>oklch(91.6% 0.042 240) | 77<br>to black<br>oklch(84.96% 0.078 240) | 65<br>to black<br>oklch(78.32% 0.114 240) | 54<br>to black<br>oklch(71.88% 0.152 240) | 65<br>to white<br>oklch(60.55% 0.136 240) | 77<br>to white<br>oklch(51.76% 0.116 240) | 90<br>to white<br>oklch(40.63% 0.091 240) | 100<br>to white<br>oklch(29.69% 0.066 240) | 105<br>to white<br>oklch(18.75% 0.041 240) |
| Indigo | 105<br>to black<br>oklch(98.83% 0.005 260) | 100 P3<br>to black<br>oklch(96.48% 0.017 260) | 90 P3<br>to black<br>oklch(91.8% 0.042 260) | 77 P3<br>to black<br>oklch(85.35% 0.078 260) | 65 P3<br>to black<br>oklch(78.91% 0.114 260) | 54 P3<br>to black<br>oklch(72.66% 0.152 260) | 65<br>to white<br>oklch(61.33% 0.136 260) | 77<br>to white<br>oklch(52.34% 0.116 260) | 90<br>to white<br>oklch(41.21% 0.091 260) | 100<br>to white<br>oklch(30.08% 0.066 260) | 105<br>to white<br>oklch(19.14% 0.041 260) |
| Violet | 105<br>to black<br>oklch(98.83% 0.005 280) | 100 P3<br>to black<br>oklch(96.48% 0.017 280) | 90 P3<br>to black<br>oklch(91.99% 0.042 280) | 77 P3<br>to black<br>oklch(85.74% 0.078 280) | 65 P3<br>to black<br>oklch(79.49% 0.114 280) | 54 P3<br>to black<br>oklch(73.44% 0.152 280) | 65<br>to white<br>oklch(61.91% 0.136 280) | 77<br>to white<br>oklch(52.93% 0.116 280) | 90<br>to white<br>oklch(41.6% 0.091 280) | 100<br>to white<br>oklch(30.47% 0.066 280) | 105<br>to white<br>oklch(19.34% 0.041 280) |
| Purple | 105<br>to black<br>oklch(98.83% 0.005 300) | 100<br>to black<br>oklch(96.68% 0.017 300) | 90<br>to black<br>oklch(92.19% 0.042 300) | 77<br>to black<br>oklch(85.94% 0.078 300) | 65<br>to black<br>oklch(79.88% 0.114 300) | 54<br>to black<br>oklch(73.83% 0.152 300) | 65<br>to white<br>oklch(62.5% 0.136 300) | 77<br>to white<br>oklch(53.32% 0.116 300) | 90<br>to white<br>oklch(41.8% 0.091 300) | 100<br>to white<br>oklch(30.66% 0.066 300) | 105<br>to white<br>oklch(19.53% 0.041 300) |
| Fuchsia | 105<br>to black<br>oklch(98.83% 0.005 320) | 100<br>to black<br>oklch(96.68% 0.017 320) | 90<br>to black<br>oklch(92.19% 0.042 320) | 77<br>to black<br>oklch(86.13% 0.078 320) | 65<br>to black<br>oklch(80.08% 0.114 320) | 54<br>to black<br>oklch(74.22% 0.152 320) | 65<br>to white<br>oklch(62.7% 0.136 320) | 77<br>to white<br>oklch(53.52% 0.116 320) | 90<br>to white<br>oklch(41.99% 0.091 320) | 100<br>to white<br>oklch(30.86% 0.066 320) | 105<br>to white<br>oklch(19.53% 0.041 320) |
| Pink | 105<br>to black<br>oklch(98.83% 0.005 340) | 100<br>to black<br>oklch(96.68% 0.017 340) | 90<br>to black<br>oklch(92.38% 0.042 340) | 77<br>to black<br>oklch(86.33% 0.078 340) | 65<br>to black<br>oklch(80.27% 0.114 340) | 54<br>to black<br>oklch(74.41% 0.152 340) | 65<br>to white<br>oklch(62.89% 0.136 340) | 77<br>to white<br>oklch(53.71% 0.116 340) | 90<br>to white<br>oklch(41.99% 0.091 340) | 100<br>to white<br>oklch(30.86% 0.066 340) | 105<br>to white<br>oklch(19.53% 0.041 340) |
| Rose | 105<br>to black<br>oklch(98.83% 0.005 0) | 100<br>to black<br>oklch(96.68% 0.017 0) | 90 P3<br>to black<br>oklch(92.38% 0.042 0) | 77<br>to black<br>oklch(86.33% 0.078 0) | 65<br>to black<br>oklch(80.27% 0.114 0) | 54<br>to black<br>oklch(74.41% 0.152 0) | 65<br>to white<br>oklch(62.7% 0.136 0) | 77<br>to white<br>oklch(53.52% 0.116 0) | 90<br>to white<br>oklch(41.99% 0.091 0) | 100<br>to white<br>oklch(30.66% 0.066 0) | 105<br>to white<br>oklch(19.34% 0.041 0) |
| Slate | 105<br>to black<br>oklch(98.83% 0.005 275) | 100 P3<br>to black<br>oklch(96.48% 0.018 275) | 90<br>to black<br>oklch(91.8% 0.024 275) | 77<br>to black<br>oklch(85.35% 0.024 275) | 65<br>to black<br>oklch(78.91% 0.024 275) | 54<br>to black<br>oklch(72.66% 0.024 275) | 65<br>to white<br>oklch(61.33% 0.024 275) | 77<br>to white<br>oklch(52.34% 0.024 275) | 90<br>to white<br>oklch(41.21% 0.024 275) | 100<br>to white<br>oklch(30.27% 0.024 275) | 105<br>to white<br>oklch(19.34% 0.024 275) |

Harmony: Accessible UI Color Palette

# Typography

## Choosing typefaces

It’s more than just picking typefaces – it’s about creating a consistent, user-friendly experience across platforms.

### Purpose

First of all, you need to understand the purpose for which you are choosing a typeface. The typeface for a kid’s app will be completely different from that of a dashboard with analytics.

Typeface for a kid's app

Titan One

Typeface for a dashboard

Inter

### Readability

Make sure your typeface is easy to read on all devices and at all sizes. Don’t let users struggle with your text, especially in large blocks.

The readability of this typeface is low

Jacquard 12 Charted

The readability of this typeface is high

Manrope

### Requirements

Also, consider special requirements like supporting symbols or multiple languages. Not all typefaces can handle these variations.

### Budget

When it comes to choosing a typeface, one of the most crucial aspects is your budget. Here are three options:

- Creating a custom typeface — usually for big companies.
- Buying a paid typeface — great for smaller companies with specific needs.
- Using free typefaces, especially if you don't have specific requirements.

### Manrope

For the Universal Design System, we chose the Manrope typeface, which met all our requirements.

## Choosing font sizes

First, we need to decide on the base size, which in the Universal Design System is 16px.

Considering that most of the text in the interface will be of this size, it's logical to classify it as Body or Paragraph text. In the Universal Design System, this text style is called 'Paragraph / Medium'.

Next, we create a size scale, as described in the theoretical part. The number of sizes will depend on your needs.

For a paragraph, I use a step of 2.

| Size | Name |
|---|---|
| 12 | Paragraph I X Small |
| 14 | Paragraph I Small |
| 16 | Paragraph I Medium |
| 18 | Paragraph I Large |

Next, we move on to creating headings with a step of 4.

| Size | Name |
|---|---|
| 20 | Heading I X Small |
| 24 | Heading I Small |
| 28 | Heading I Medium |
| 32 | Heading I Large |
| 36 | Heading I X Large |
| 40 | Heading I XX Large |

For the Display role, we chose values so that they are visually different from each other. Therefore, in the first case, the difference is 16, and in the second, it is 20.

| Size | Name |
|---|---|
| 56 | Display I Small |
| 72 | Display I Medium |
| 92 | Display I Large |

## Choosing line heights

After we have selected the sizes for the text styles, we need to choose the line heights. Let's use the following formula for the paragraph:

**Font Size x Ratio = Line Height**

For the role paragraph, we take a ratio of 1.4. We suggest rounding the result to a number divisible by the base size(4 in our case) for a harmonious design where text fits the grid.

| 14 | x 1.4 | = 19.6 |
|---|---|---|
| Font Size | Ratio | Line Height |

In this case, the closest value to 19.6 is 20.

| Size | Line height | Name |
|---|---|---|
| 12 | 16 | Paragraph I X Small |
| 14 | 20 | Paragraph I Small |
| 16 | 24 | Paragraph I Medium |
| 18 | 28 | Paragraph I Large |

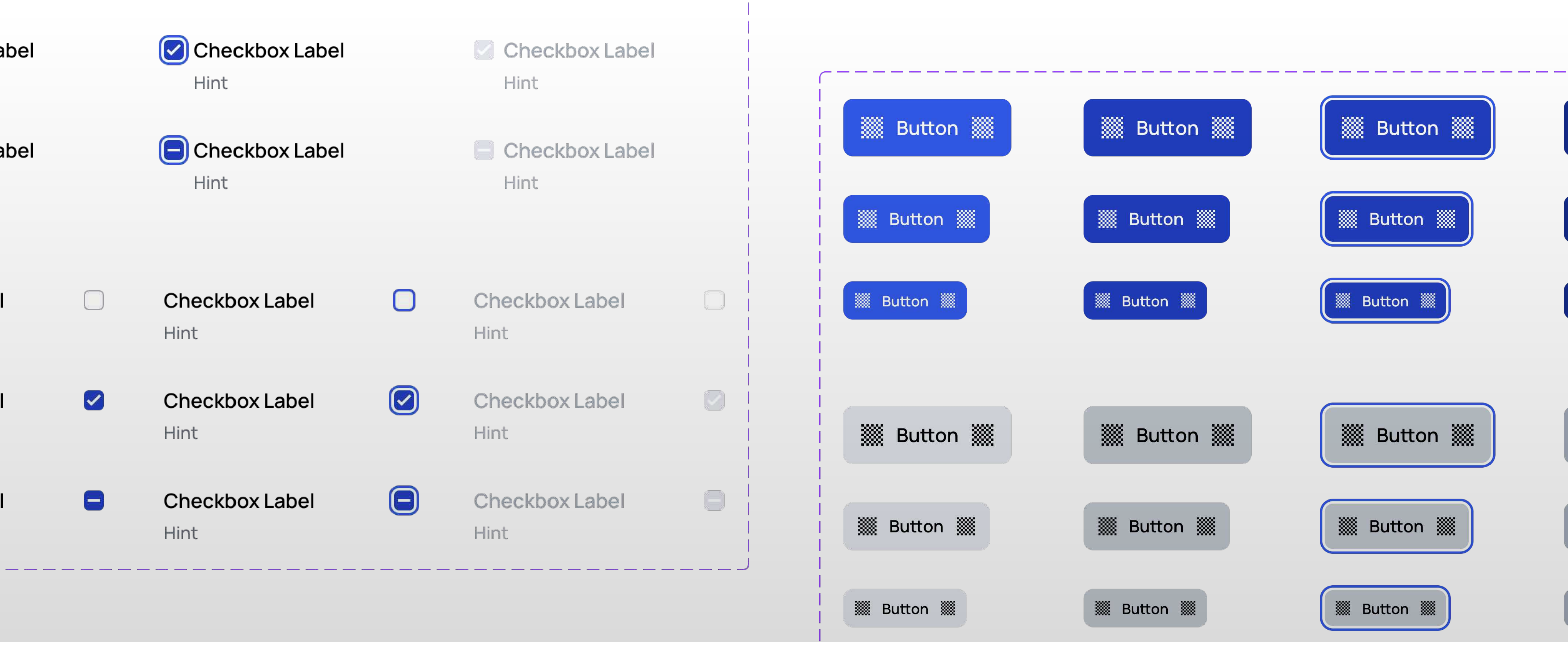
For X Small and Small Headings, we also use a ratio of 1.4. Starting from Medium, we reduce it to 1.3. It is important not only to follow the formula but also to see how each value will look in practice. If necessary, you can adjust it.

| Size | Line height | Name |
|---|---|---|
| 20 | 28 | Heading I X Small |
| 24 | 32 | Heading I Small |
| 28 | 36 | Heading I Medium |
| 32 | 40 | Heading I Large |
| 36 | 44 | Heading I X Large |
| 40 | 52 | Heading I XX Large |

For the Display role, the value 1.3 is perfectly suitable.

| Size | Line height | Name |
|---|---|---|
| 56 | 72 | Display I Small |
| 72 | 96 | Display I Medium |
| 92 | 120 | Display I Large |

Once you have set all the sizes for the text styles, you need to test them and remember that you can always go back and change them.



# Components

This section of the book explains how to create the most commonly used components in design systems using Figma. We will go over the process of creating components like buttons, inputs, and checkboxes.

# Introduction

## Overview

Components are reusable elements in your designs that ensure consistency across the design system. You can create components from any layers or objects, including buttons, icons, layouts, and more.

There are two main aspects of a component:

❖ **Main component:** Defines the properties of the component.

◇ **Instance:** A reusable copy of the main component. Instances are linked to the main component and automatically update with any changes made to it.

## Creating components

Creating components in Figma is simple. Follow these steps:

1. **Design the element:** Start by creating the element you want to make into a component, such as a button or icon.
2. **Select the element:** Click on the element you wish to convert into a component.
3. **Create the component:** Right-click on your selection and choose 'Create Component' from the context menu, or use the shortcut.
4. **Name the component:** Give your component a clear, descriptive name for easy identification later.

## Using components

Here's how to use a component in Figma

**Locate components:** Open the Assets panel on the left side of the Figma interface to view your components.

**Insert component:** Drag a component from the Assets panel into your design.

**Modify instances:** Customize properties of the component instances, such as changing the text or swapping an icon, without affecting the original component.

# Advanced component features

Figma includes advanced features to make components more powerful:

### Variants

Group related components, like a button with different states (default, hover, active), to keep them organized.

### Auto layout

Use Auto Layout to make components adjust to different content sizes, perfect for responsive design.

### Interactive components

Add predefined interactions and animations to components, ideal for prototyping and testing user interactions.

# Learn more about components

### Guide to components in Figma

🔗 help.figma.com/hc/en-us/articles/360038662654-Guide-to-components-in-Figma

### Create and use variants

🔗 help.figma.com/hc/en-us/articles/360056440594-Create-and-use-variants

### Explore component properties

🔗 help.figma.com/hc/en-us/articles/5579474826519-Explore-component-properties

### Name and organize components

🔗 help.figma.com/hc/en-us/articles/360038663994-Name-and-organize-components

### Create components to reuse in designs

🔗 help.figma.com/hc/en-us/articles/360038663154-Create-components-to-reuse-in-designs

### Create interactive components with variants

🔗 help.figma.com/hc/en-us/articles/360061175334-Create-interactive-components-with-variants

### Create and manage component properties

🔗 help.figma.com/hc/en-us/articles/8883756012823-Create-and-manage-component-properties

### Prepare for variants

🔗 help.figma.com/hc/en-us/articles/360055471353-Prepare-for-variants

# Auto Layout

## Overview

Figma's auto layout feature is a powerful tool for designers, making designs more dynamic and flexible. It automatically handles the positioning and resizing of elements, making your workflow faster and more responsive. Learning to use auto layout can greatly improve your design's consistency and scalability.

## What is auto layout?

Auto layout in Figma allows designs to automatically adjust to content changes, similar to CSS flexbox. It helps elements resize, reposition, and align based on the rules you set, making it ideal for creating responsive designs that work across different screen sizes.

You can apply auto layout to frames and components, so your designs can expand, shrink, and rearrange as content changes. This is useful for adding new layers, handling longer text, or keeping everything aligned as your design evolves.

Some ways to use auto layout:

- Create buttons that resize with the text
- Build lists that adapt as items are added or removed
- Combine auto layout frames to create entire interfaces

## Key features of auto layout

### Dynamic resizing

Elements in an auto layout frame can resize automatically based on their content. This means buttons, cards, and other components adjust their size as text or images are added or removed.

### Alignment and distribution

Auto layout lets you control the alignment and spacing of elements. You can easily set padding, margins, and spacing between items, keeping your layout consistent throughout your design.

### Direction and order

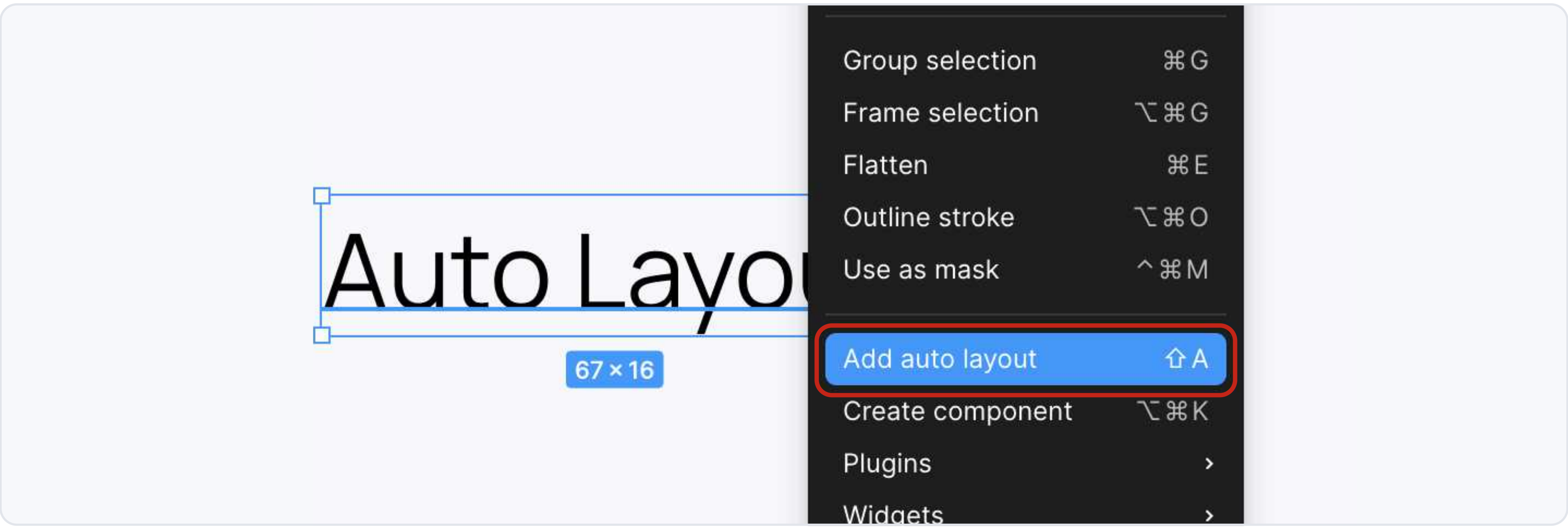
You can set the layout direction (horizontal or vertical) and control the order of elements. This makes it simple to stack items or place them side by side.

### Nested auto layouts

Auto layout frames can be nested, enabling complex and responsive layouts. This feature is ideal for creating detailed designs that adjust to different content sizes and screen dimensions.

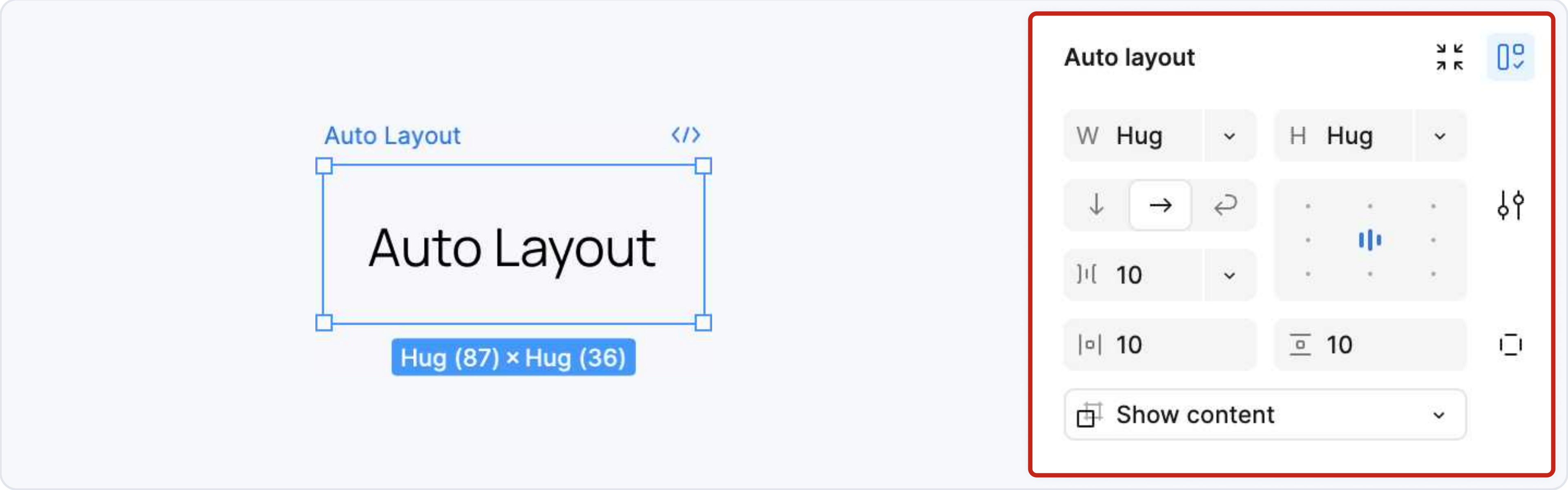
## Creating an auto layout

Select the elements you want to include, then right-click and choose **Add auto layout** from the menu. This will group the selected elements into an auto layout.

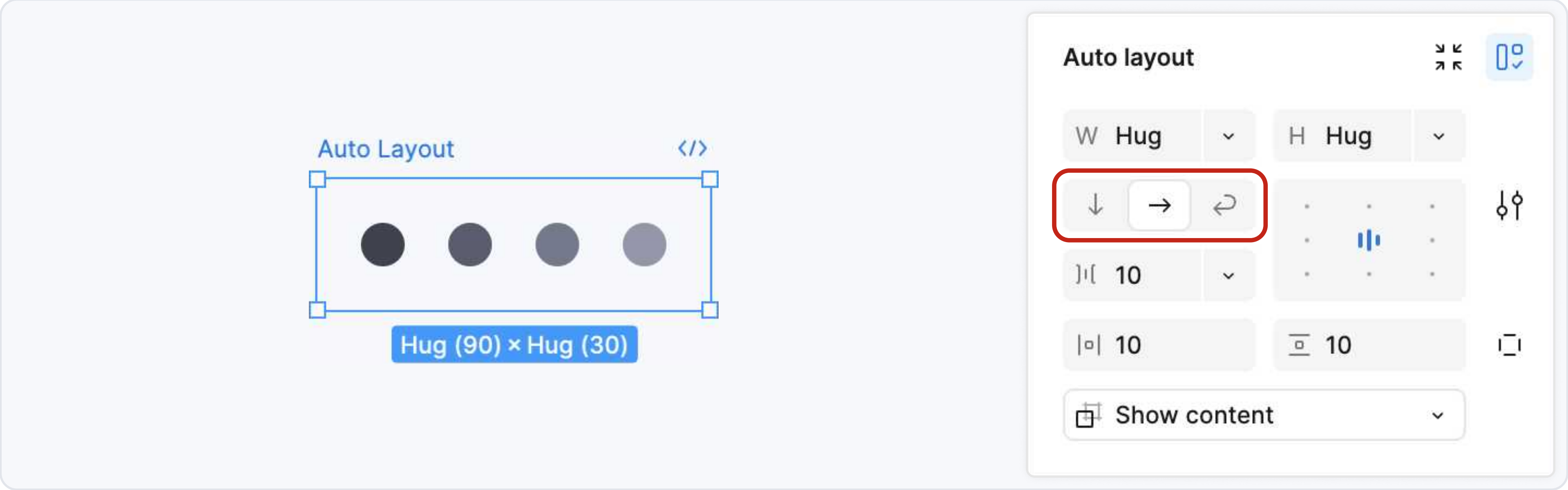


# Properties

In the properties panel, you can adjust the direction (horizontal or vertical), spacing between items, padding, and alignment. These settings control how the elements look and behave in the auto layout frame.



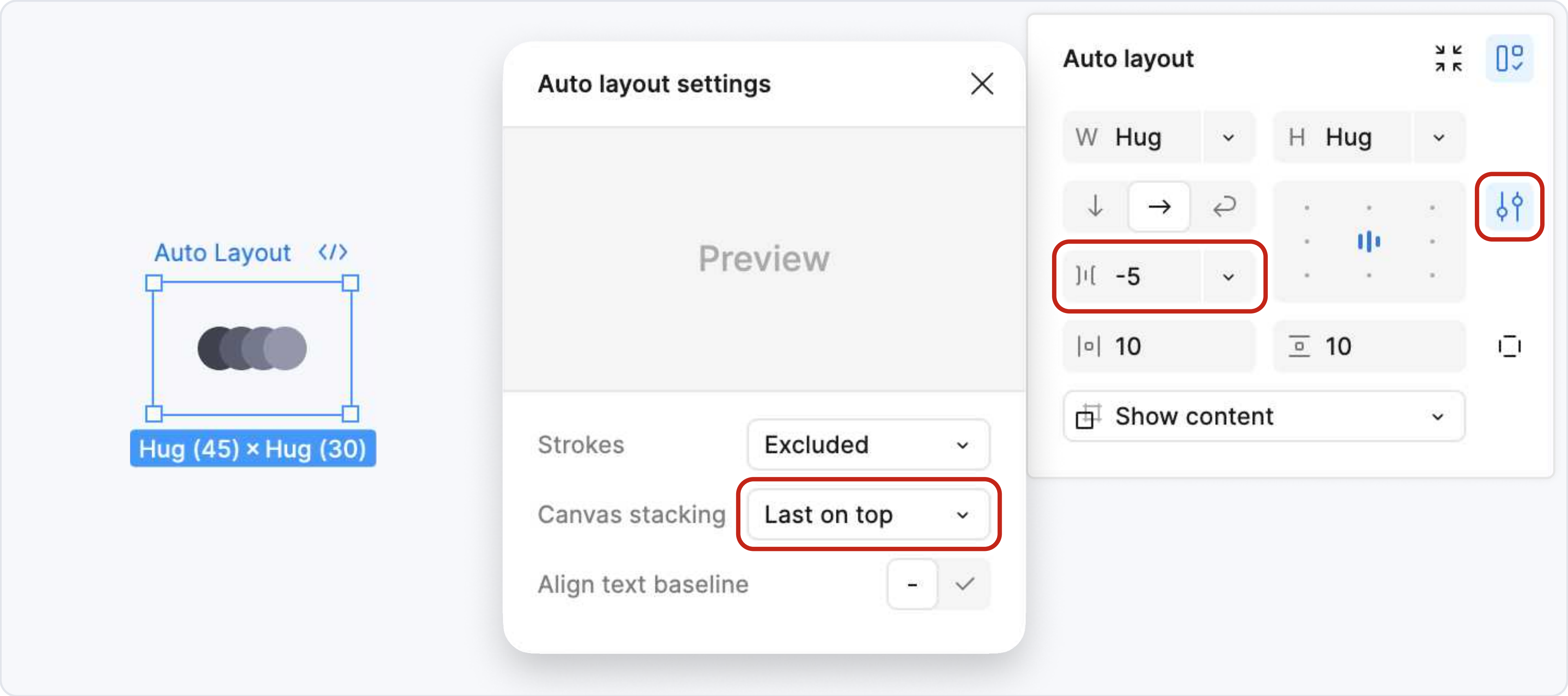
## Direction



Direction controls how the auto layout frame will arrange items.

- **Vertical:** Items are arranged up and down (y-axis), like in a list.
- **Horizontal:** Items are arranged side by side (x-axis), like a row of icons in a tab bar.
- **Wrap:** Items are arranged in rows and columns, moving to the next line when needed, like in a photo gallery.

## Canvas stacking order



Auto layout settings

Strokes: Excluded

Canvas stacking: Last on top

Align text baseline: -

W Hug

H Hug

-5

10

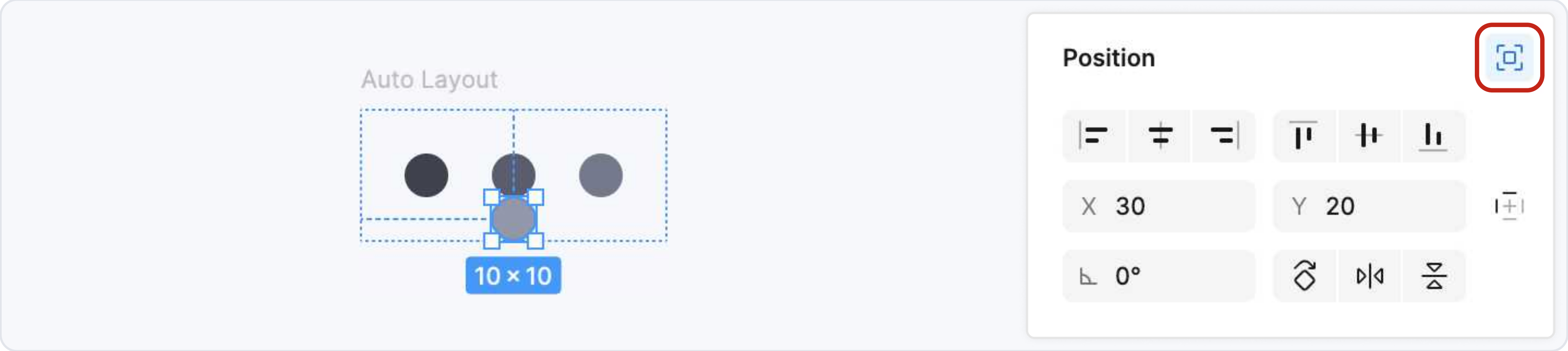
10

Show content

When layers overlap with negative spacing, the last object (right-most or bottom-most) will be on top by default. To change the order, select the auto layout frame, click the settings button in the right panel, and choose:

- **First on top:** The first layer in the stack will be on top.
- **Last on top:** The last layer in the stack will be on top.

## Ignore auto layout (Absolute position)



Position

X 30

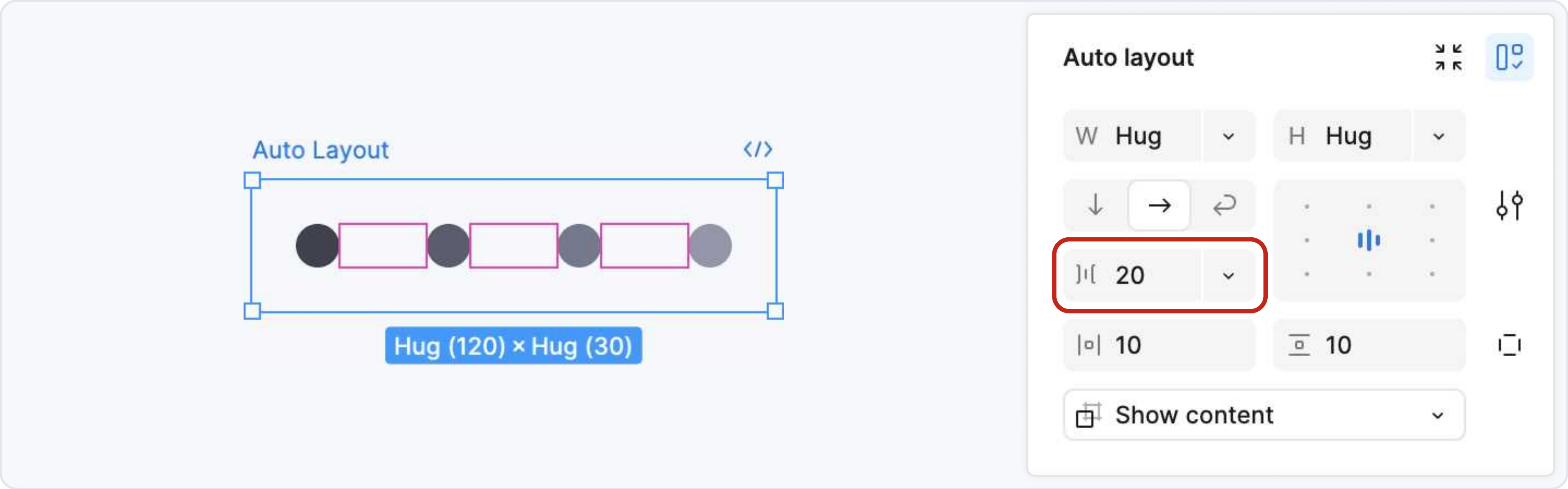
Y 20

0°

An object with “Ignore auto layout” stays inside the frame but doesn’t follow the auto layout rules. You can position it exactly, like with absolute positioning in CSS.

To turn this on, select the object and enable the option in the right panel.

## Gap between items

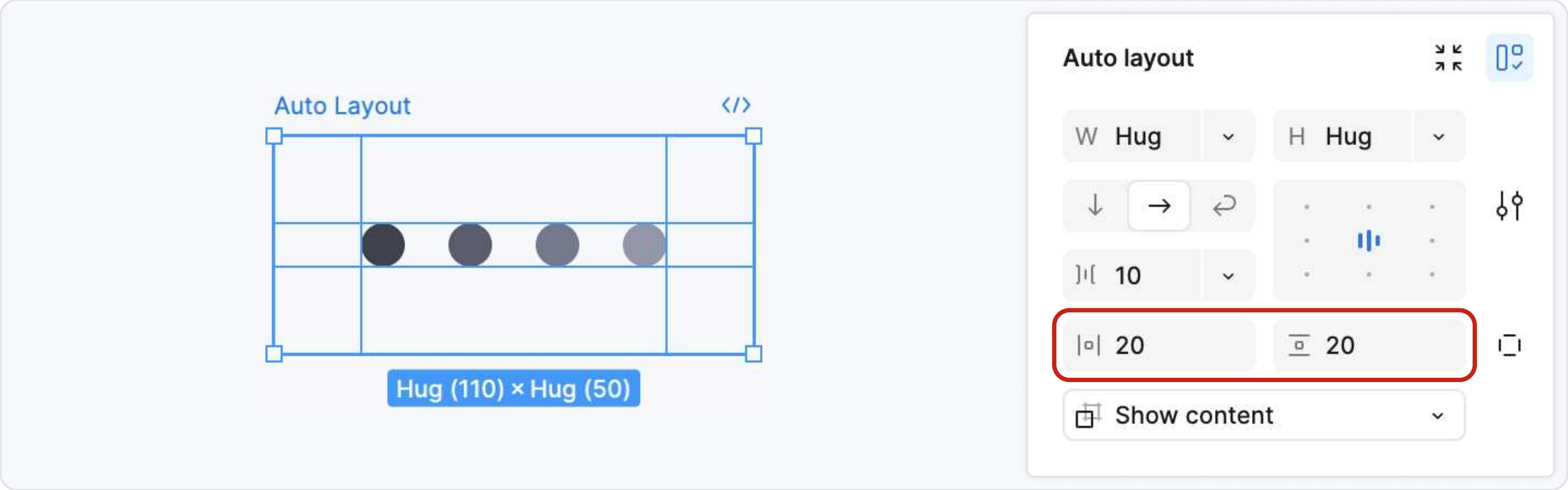


Use gap between items to set the distance between objects in an auto layout frame.

There are two settings:

- **Auto:** Automatically sets the largest possible gap between items. Type "Auto" or select it from the dropdown menu.
- **A specified gap:** Choose a specific distance by entering a value.

## Padding



Padding controls the space between the edge of an auto layout frame and its child objects. You can set padding equally on all sides or customize it for top, right, bottom, and left.

To adjust padding:

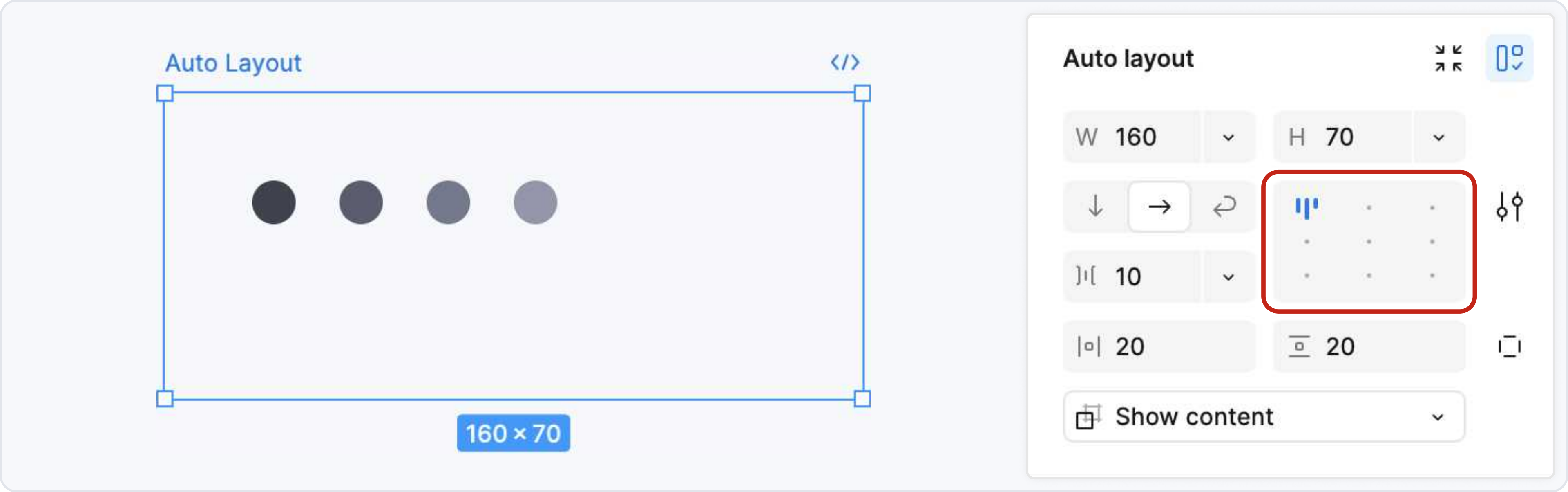
- Use the controls on the canvas: Select the frame, hover over it to see pink handles, then click or drag these handles to change the padding.
- Or use the spacing fields in the right panel.

## Alignment

### Set alignment on child objects

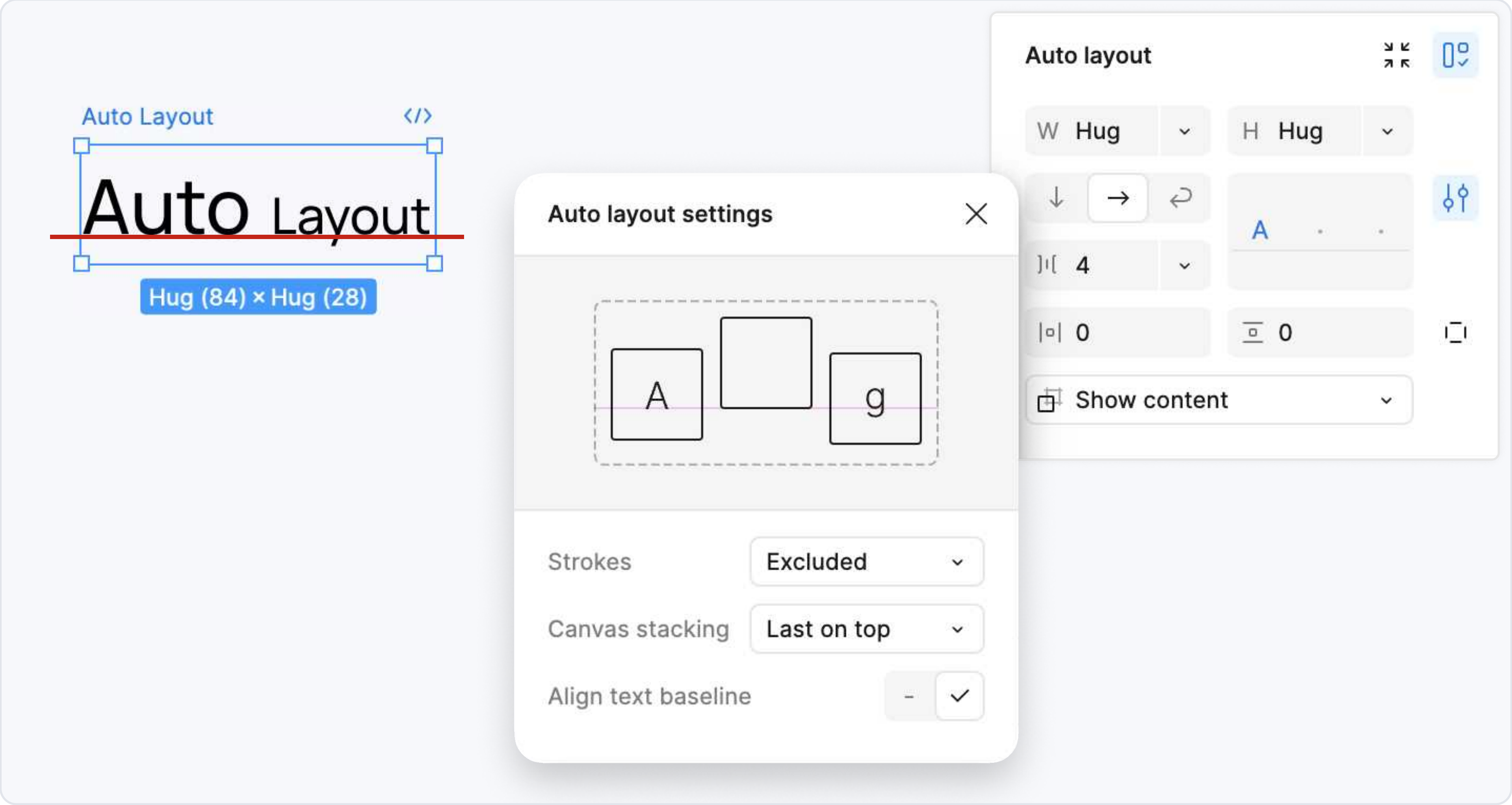
Set alignment on child objects within an auto layout frame by choosing how they align. The direction and gap between items will affect your alignment options.

Unlike a regular frame, you can't align objects individually. Instead, you set the alignment for all child objects on the parent auto layout frame.



### Text baseline alignment

When layers of different heights are vertically centered in a horizontal auto layout, they align in the middle. However, if you have text layers of different sizes or text with an object like a button with an icon, you might need to align their baselines instead.



Auto layout settings

Strokes: Excluded

Canvas stacking: Last on top

Align text baseline: ✓

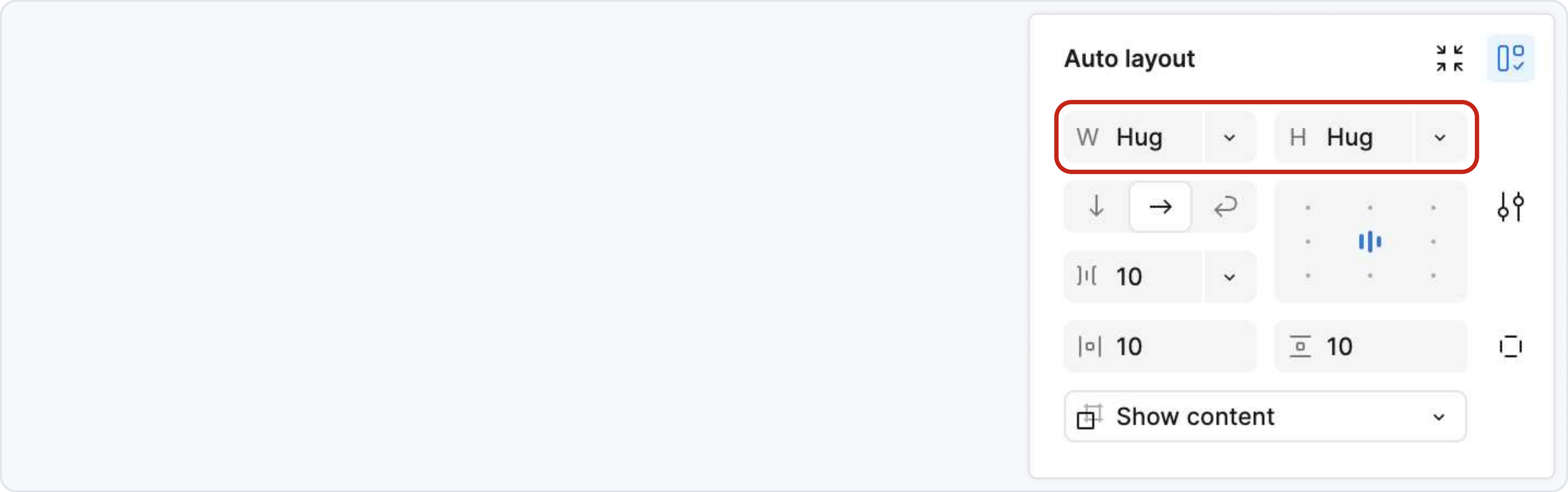
## Resizing

One of the key features of auto layout is its ability to control the size of objects within the frame. You can set resizing behavior for the parent auto layout frame to adjust when its child objects change. Width and height can be resized separately using the dropdown menus in the right panel.

You can set elements to:

- **Hug contents:** Resize to fit the content.
- **Fill container:** Stretch to fill the available space.
- **Fixed size:** Keep a set size.

These options control how elements resize within the auto layout frame.



Auto layout

W Hug

H Hug

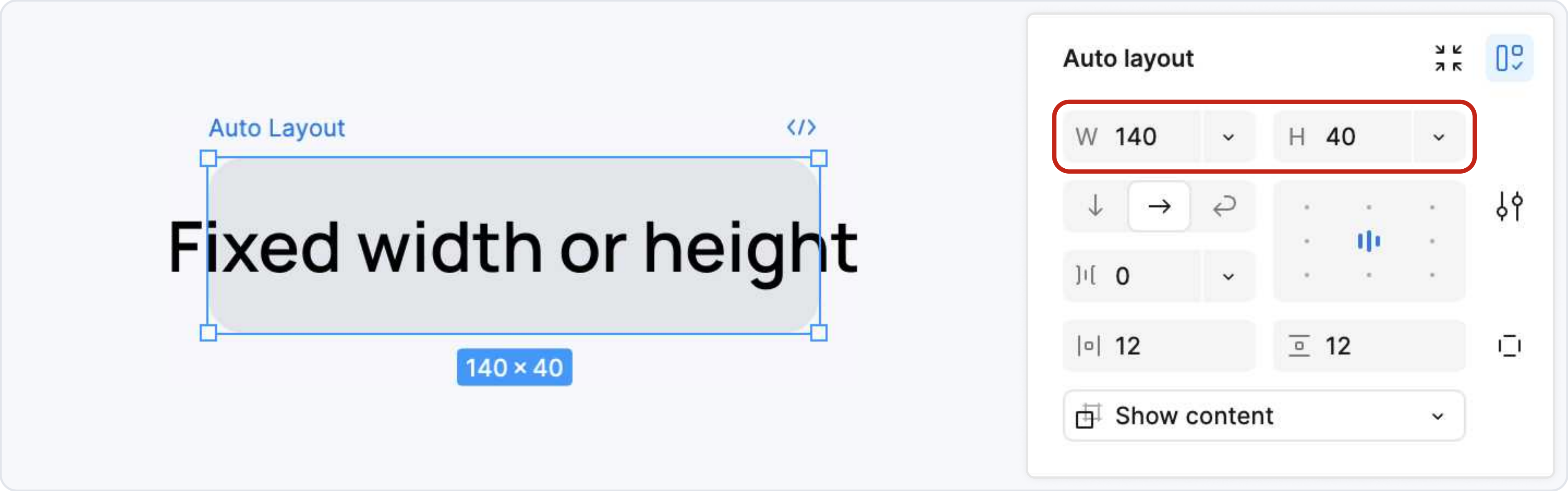
10

10

10

Show content

### Fixed width or height



Auto layout

W 140

H 40

0

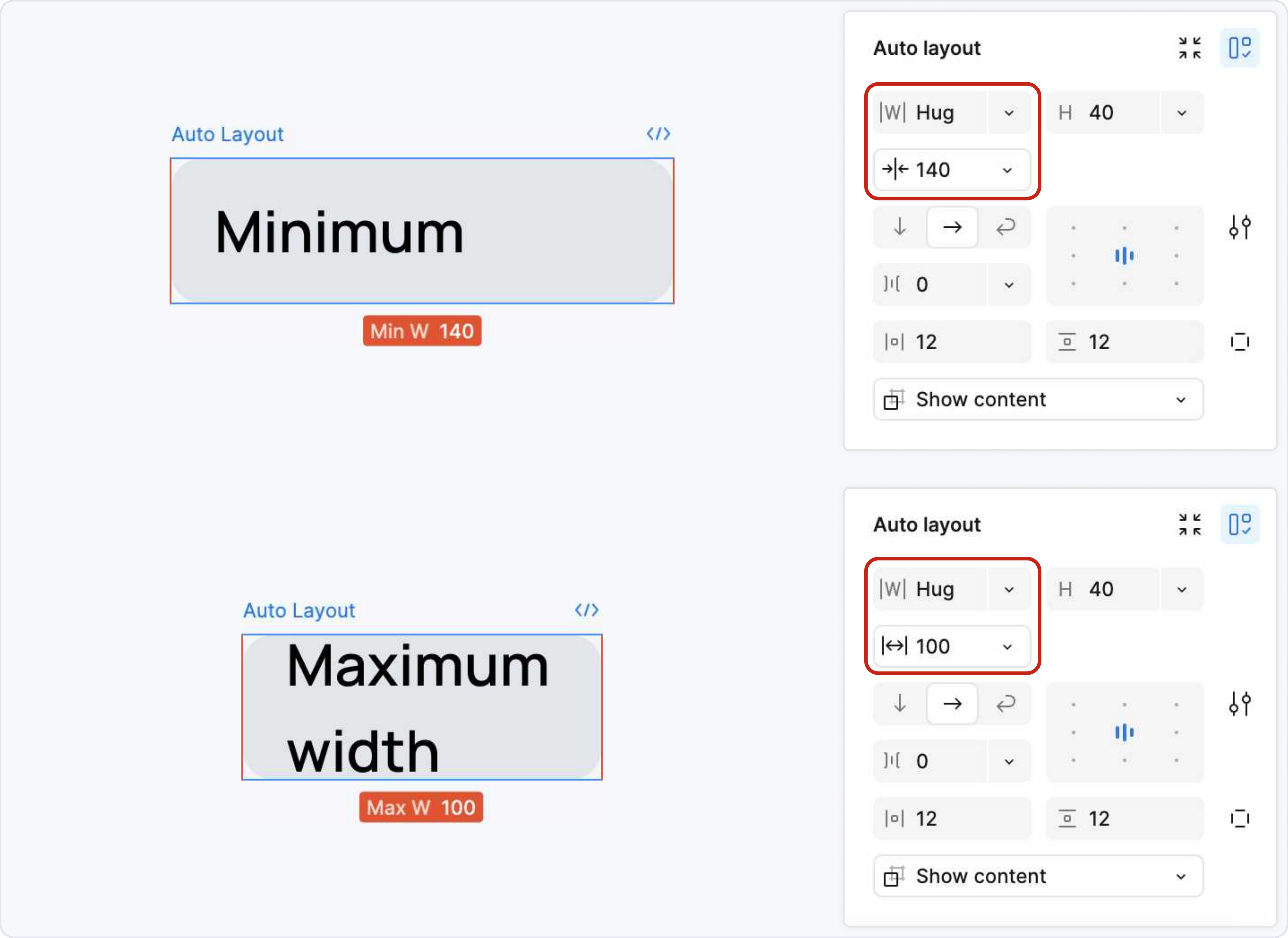
12

12

Show content

When an auto layout frame is set to "Fixed" width or height, its dimensions stay the same no matter what's inside. The frame won't adjust if the content, like a text string, changes in size.

## Minimum and maximum dimensions

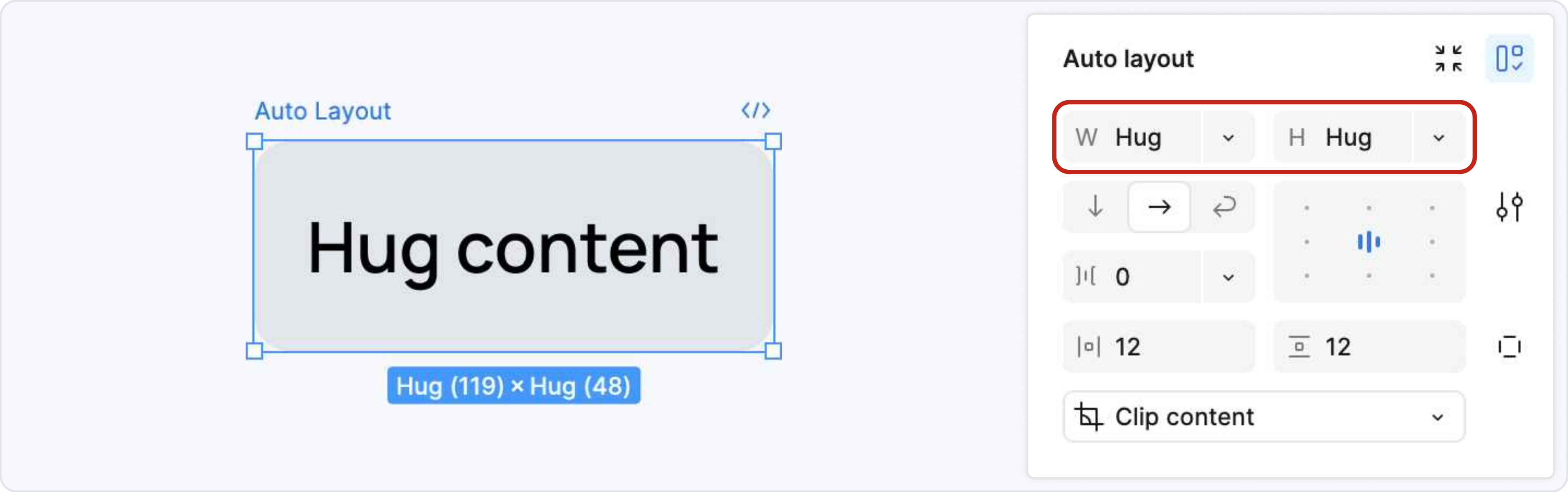


You can set minimum or maximum width and height for an auto layout frame and its children.

- In the **Width dropdown**, select ”Add min width” or ”Add max width”.
- In the **Height dropdown**, select ”Add min height” or ”Add max height”.

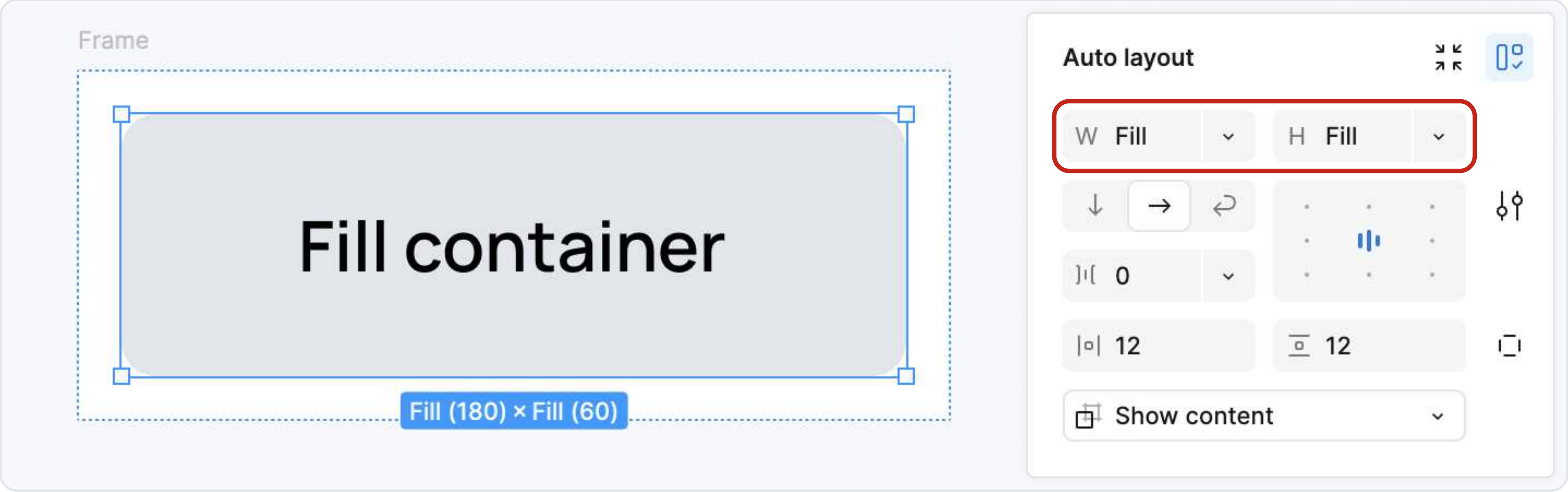
After selecting, a new field appears where you can enter a value or choose a number from the dropdown.

## Hug content



Set an auto layout frame to Hug content to make it resize based on its child objects. The frame will adjust to the smallest possible size that still fits the objects, while keeping the padding value.

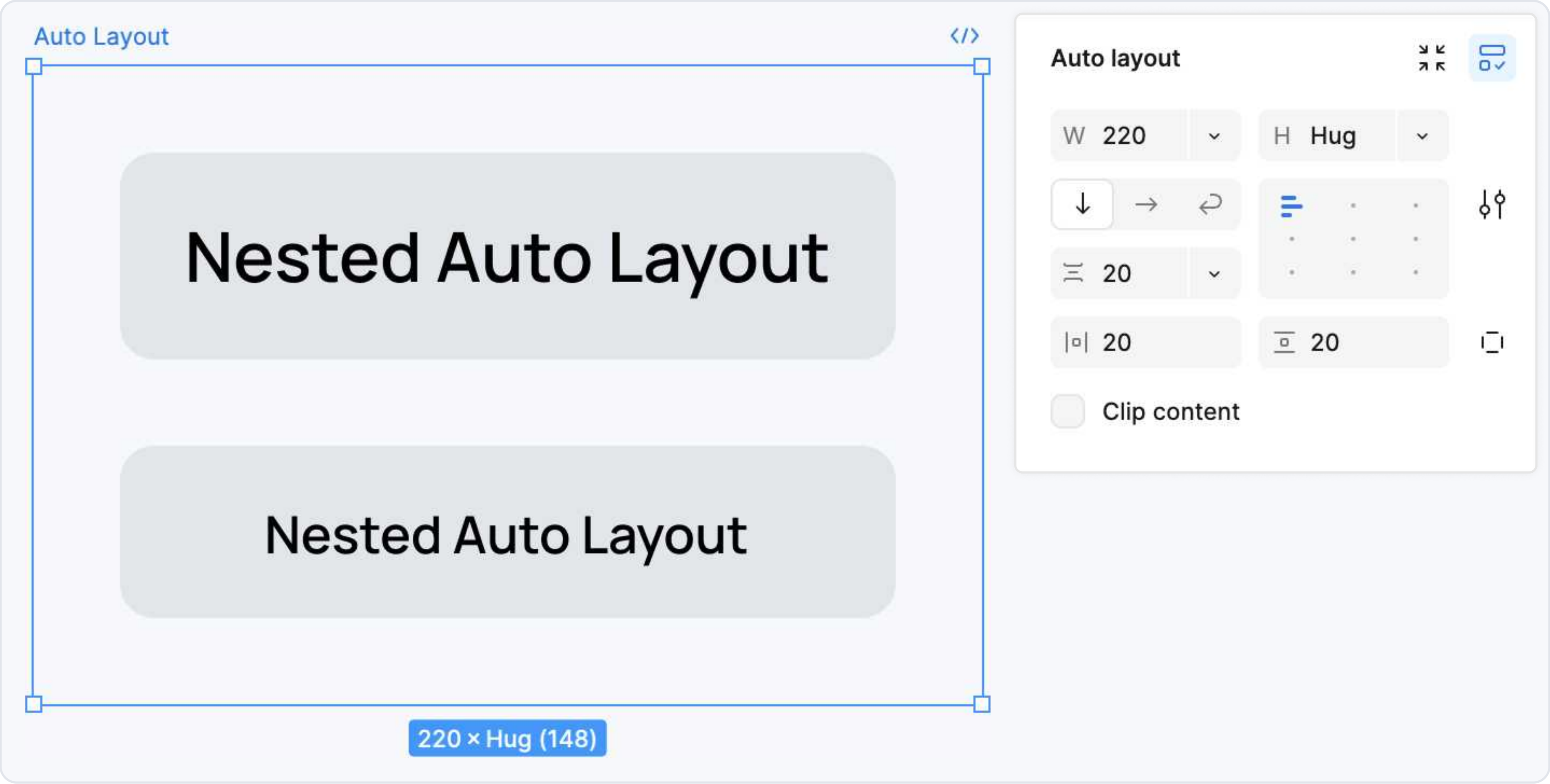
## Fill container



Objects in an auto layout frame set to Fill container will stretch to match the width and/or height of the parent frame. Child objects will also be set to Fill container if you manually resize them to fill the parent frame's width.

# Nesting auto layouts

To make more complex layouts, you can nest auto layout frames within each other. This lets you create designs that adjust to different content sizes and structures.



## Learn more about auto layouts

### Explore auto layout properties

[help.figma.com/hc/en-us/articles/360040451373-Explore-auto-layout-properties](https://help.figma.com/hc/en-us/articles/360040451373-Explore-auto-layout-properties)

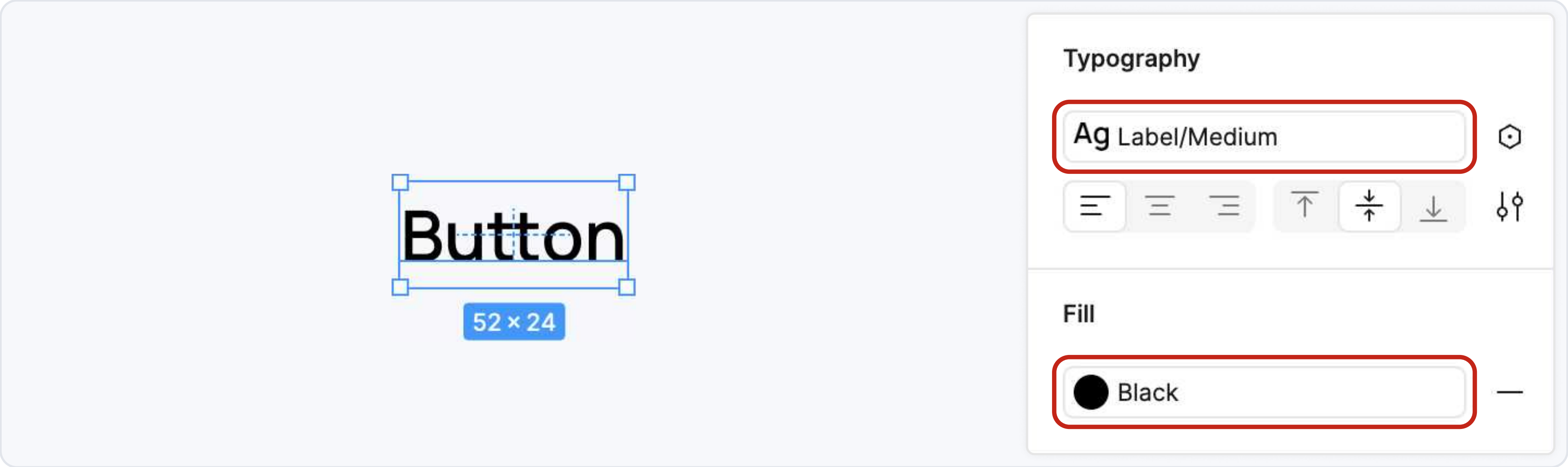
### Add auto layout to a design

[help.figma.com/hc/en-us/articles/5731482952599-Add-auto-layout-to-a-design](https://help.figma.com/hc/en-us/articles/5731482952599-Add-auto-layout-to-a-design)

# Button

## Create a universal button

1. Let’s start with the text “Button” and assign it the “Label/Medium” text style.



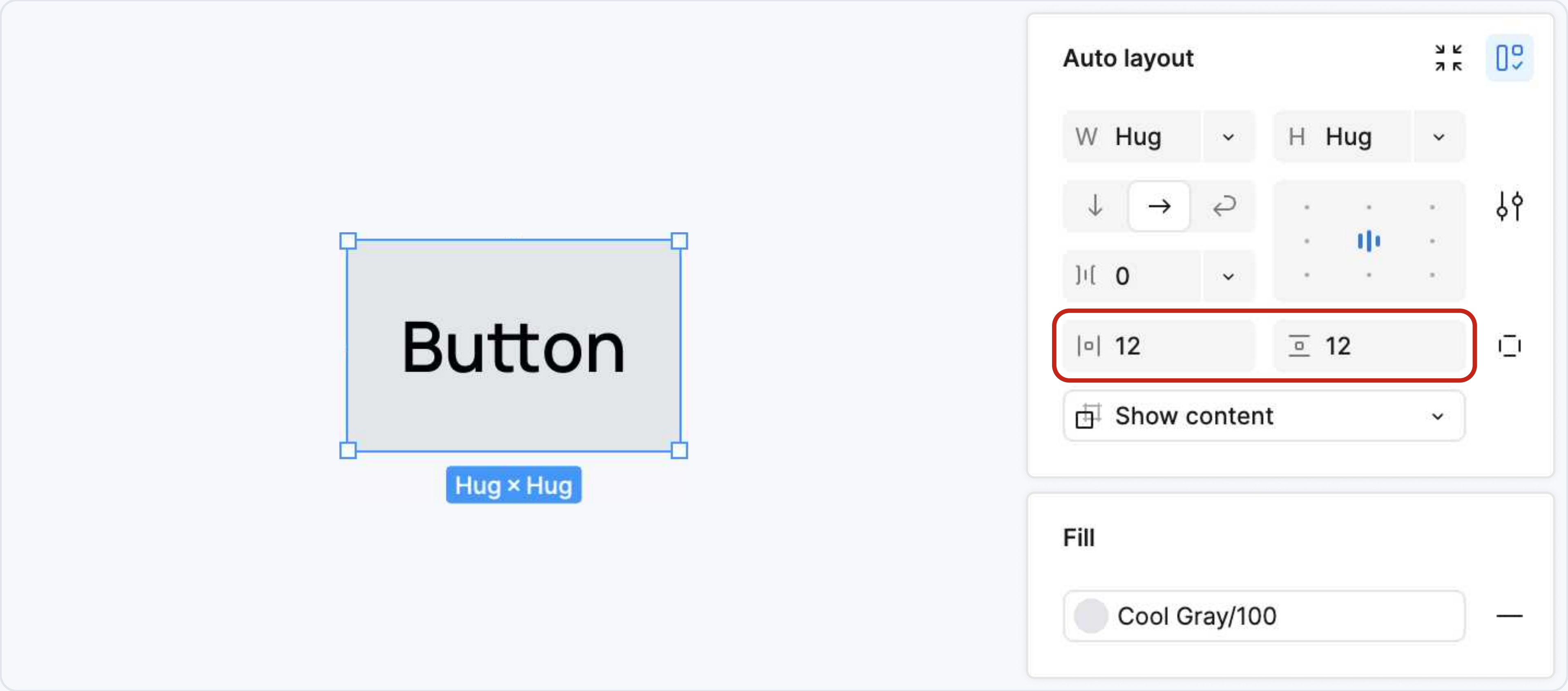
Typography

Ag Label/Medium

Fill

● Black

2. Next, create an auto layout for the text with 12-pixel spacing.



Auto layout

W Hug

H Hug

0

12

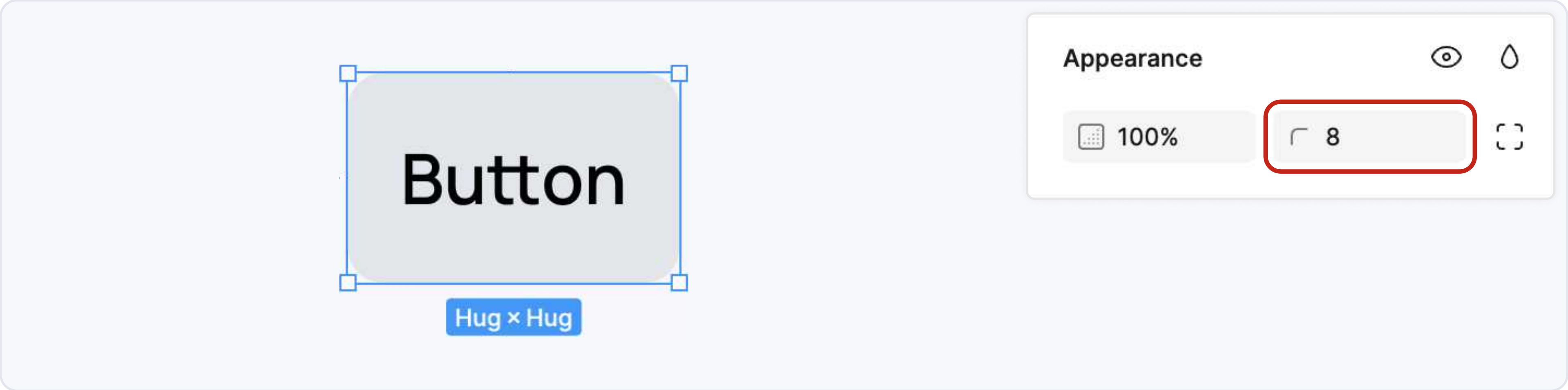
12

Show content

Fill

Cool Gray/100

3. Set the corner radius to 8.

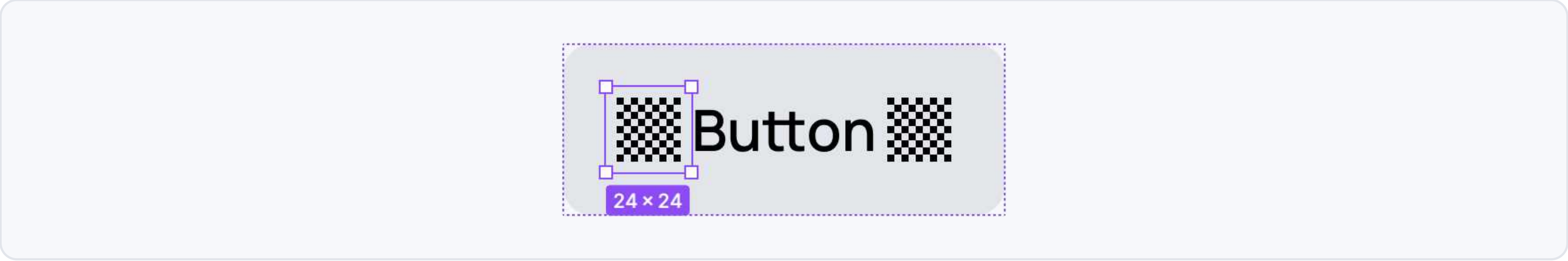


Appearance

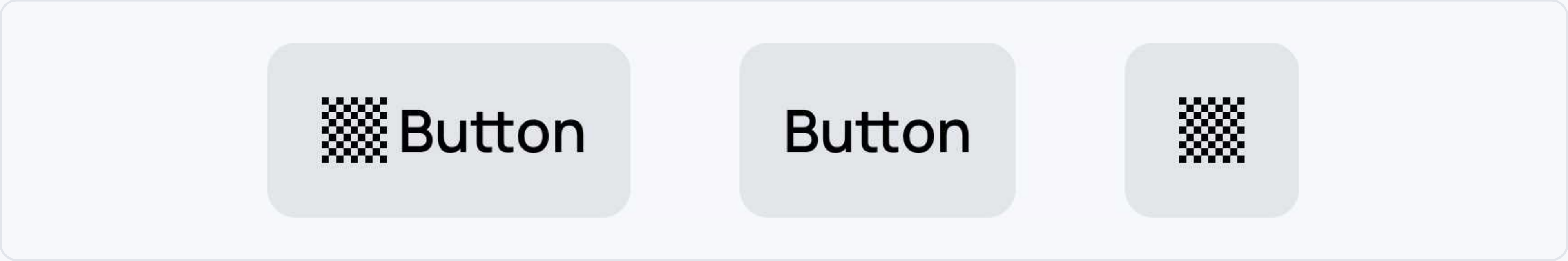
100%

8

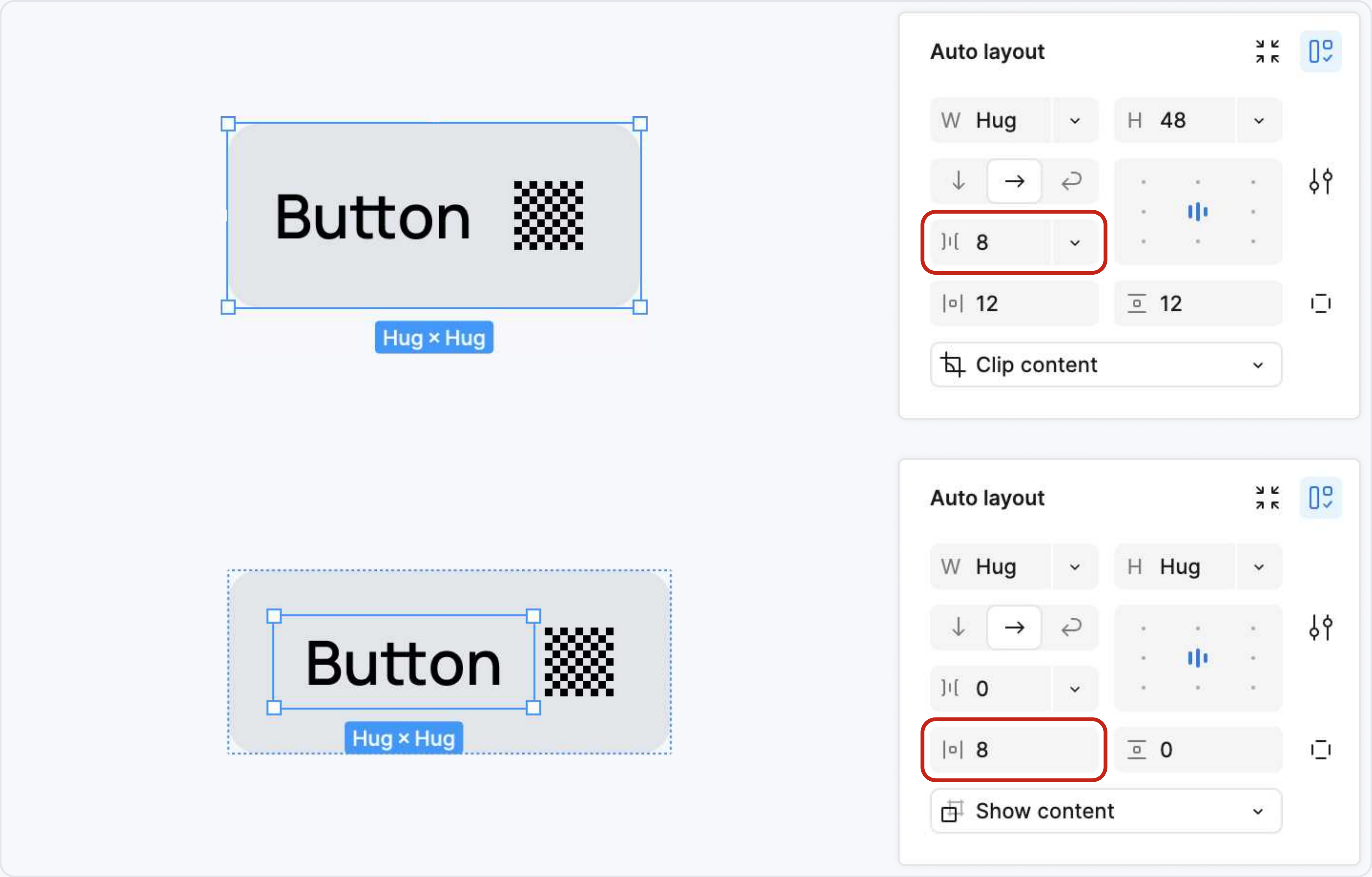
4. Add default icons to the edges to make the button more adaptable. Set the icon size to 24x24px to match the text size.



5. This way, we gain the ability to manipulate the button. Display either icons, text, or a combination of both.



6. The only thing missing here is the spacing between the icons and the text. If we simply set the spacing between the text and the icon to an auto layout value, then in the case of a combined display, our text will be aligned to one of the edges (Option 1). Therefore, I suggest creating a separate auto layout for the text and setting the spacing there (Option 2).



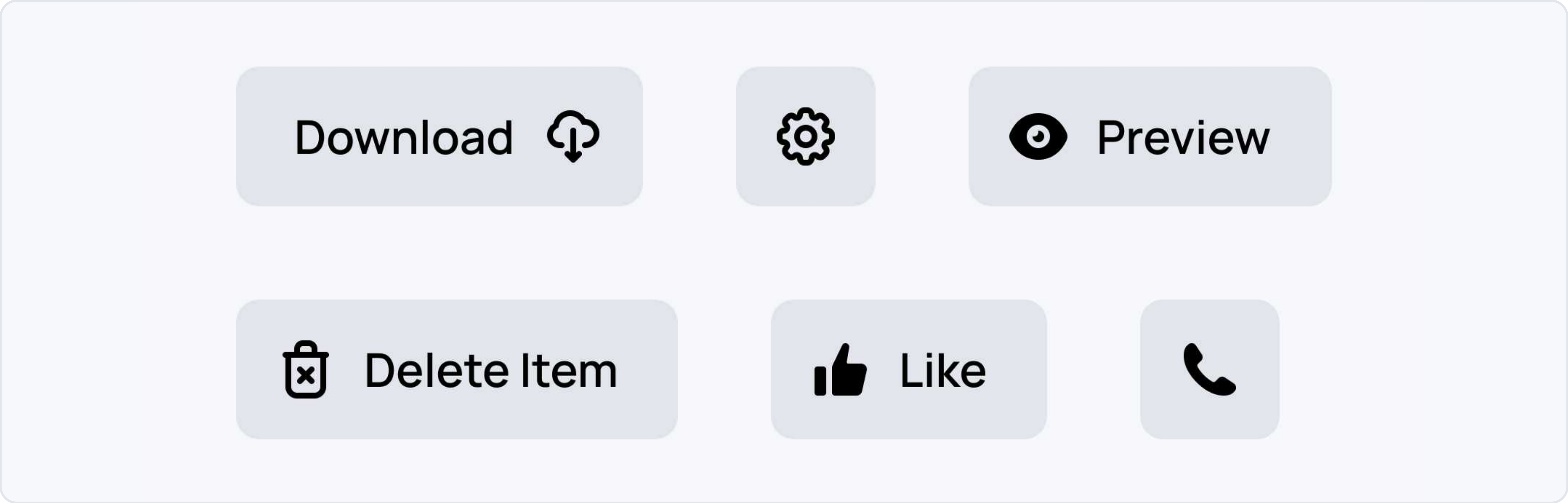
Auto layout

W Hug
H 48
8
12
12
Clip content

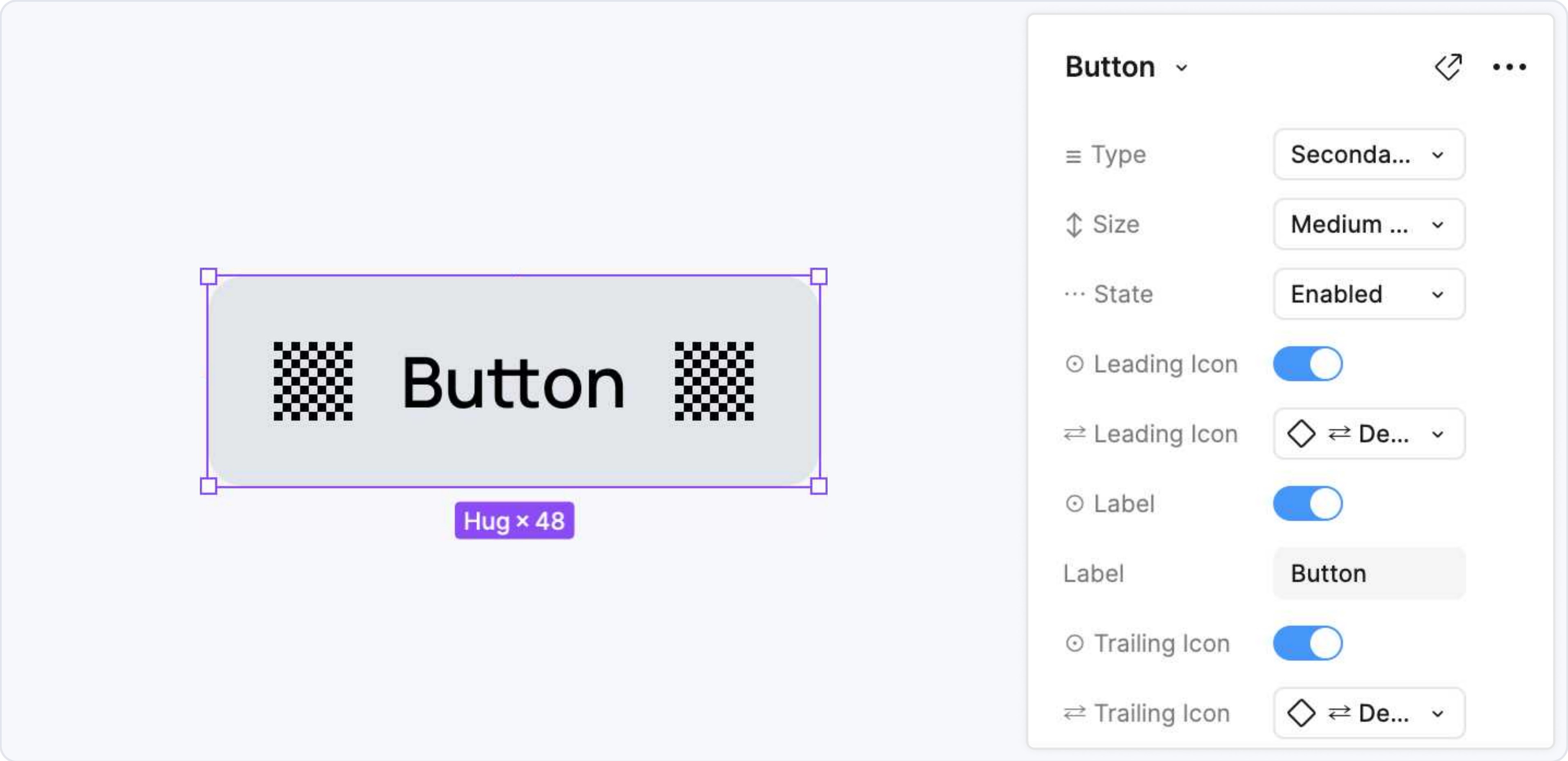
Auto layout

W Hug
H Hug
0
8
0
Show content

The result is that our button looks harmonious in any state.



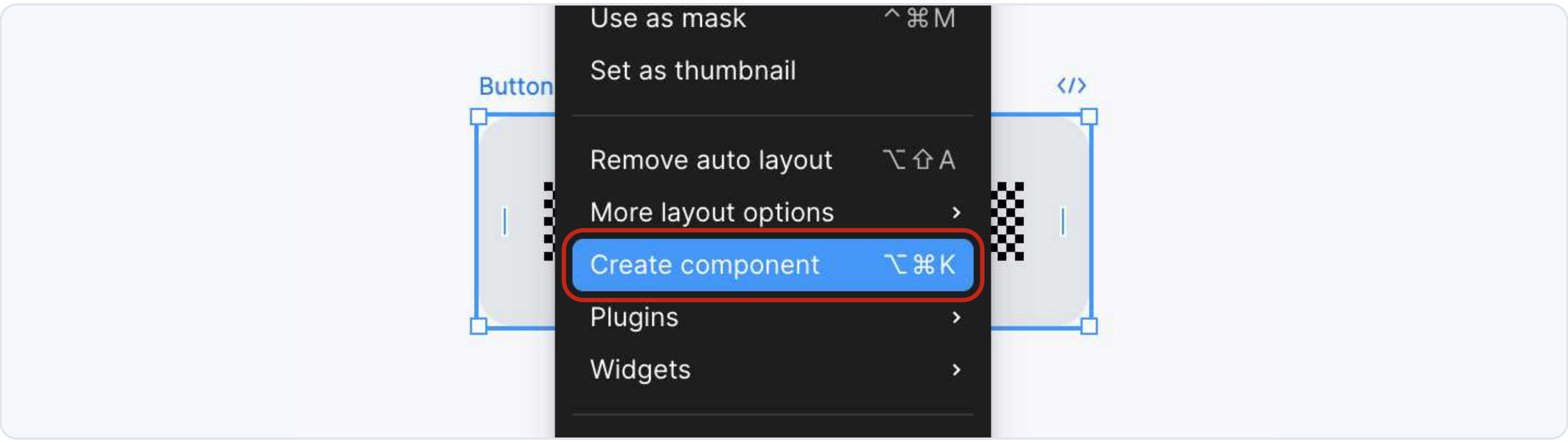
7. After we have created our button, we need to implement user-friendly controls for it, allowing us to configure it in one place.



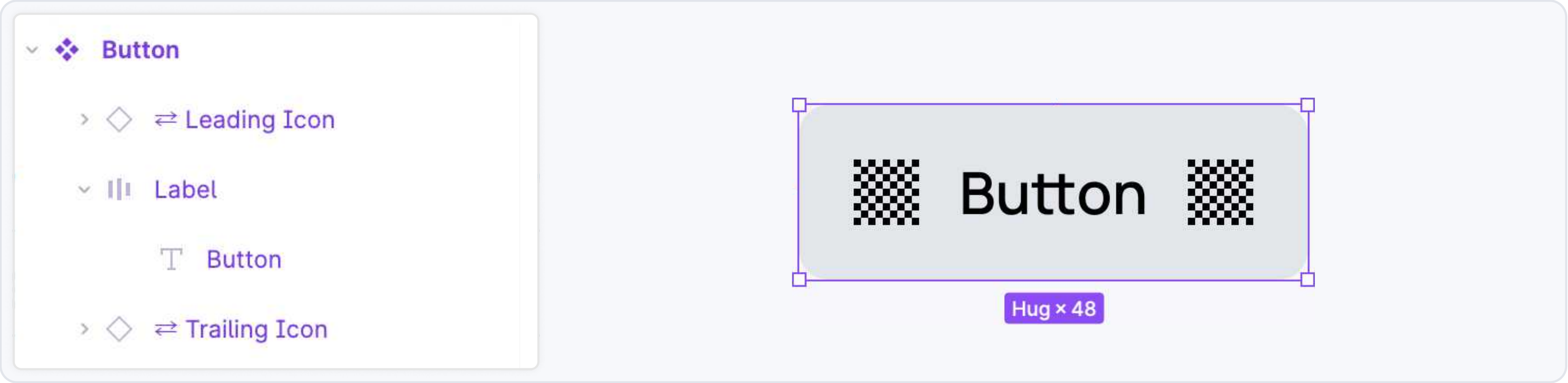
Button

| | |
|---|---|
| Type | Seconda... |
| Size | Medium ... |
| State | Enabled |
| Leading Icon | ● |
| Leading Icon | ◇ ⇄ De... |
| Label | ● |
| Label | Button |
| Trailing Icon | ● |
| Trailing Icon | ◇ ⇄ De... |

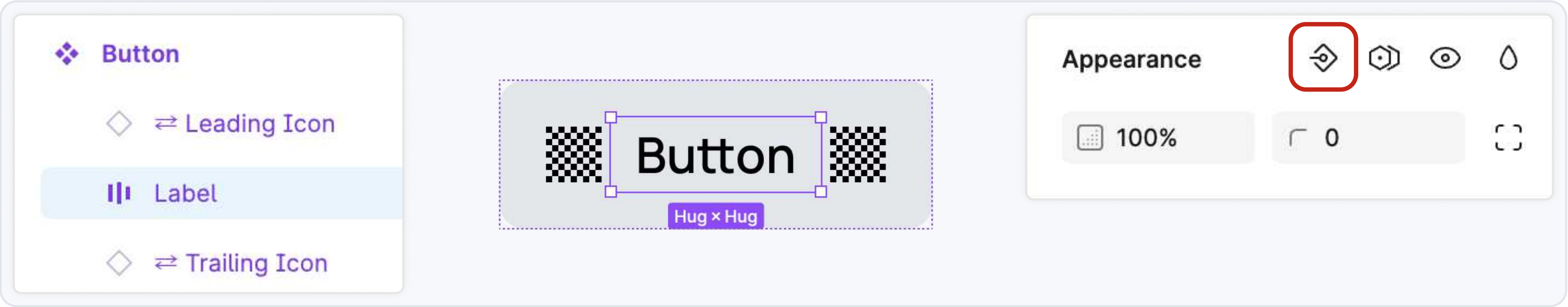
8. To do this, create a component.



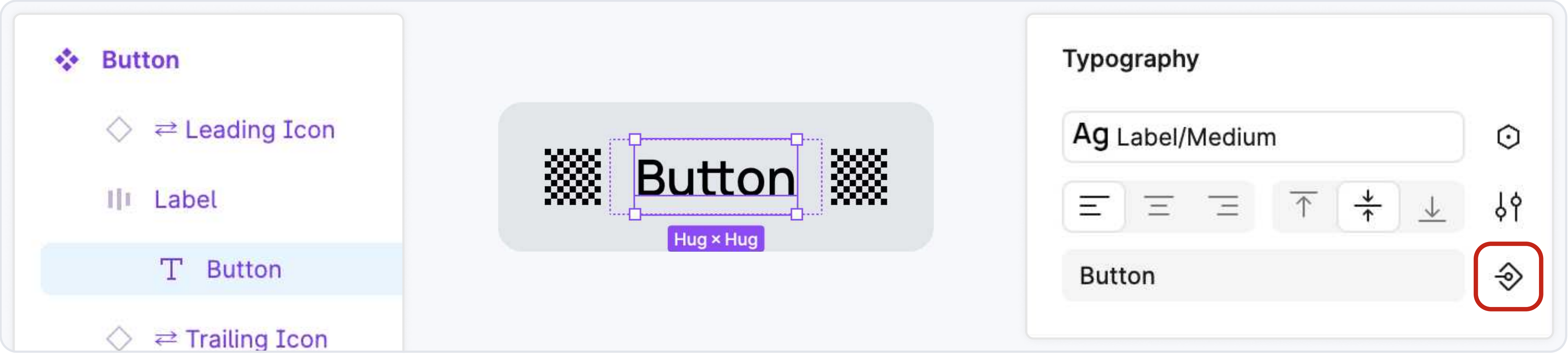
9. Before you start creating properties, I recommend naming all your layers in the component. This will help you navigate the structure more easily. When naming layers, I use a special symbol “ ⇄ ” for elements that can be replaced with similar ones, like icons.



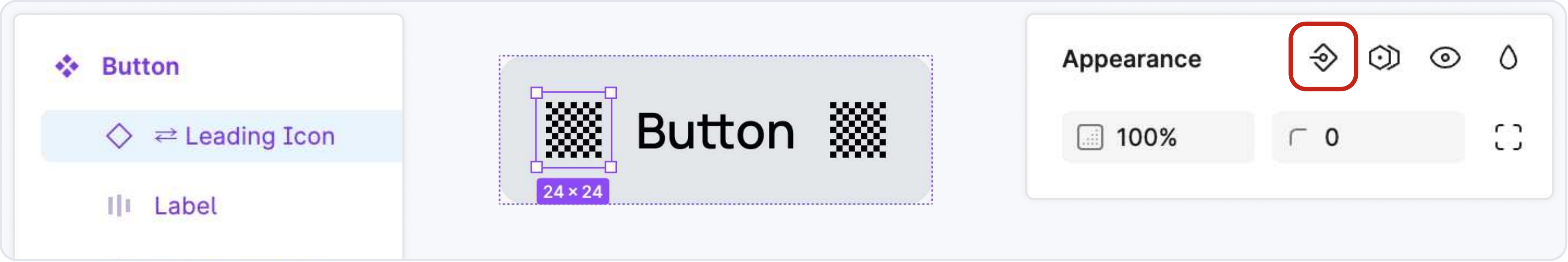
10. For the text “Button”, we need to create 2 properties. One for showing/hiding the text. This is created for the auto layout.

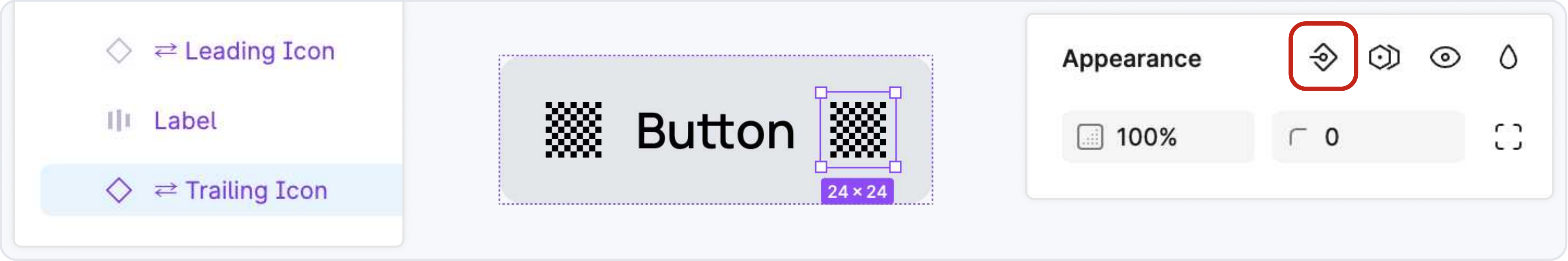


The second one is for changing the text itself. We create it for the “Button” text.



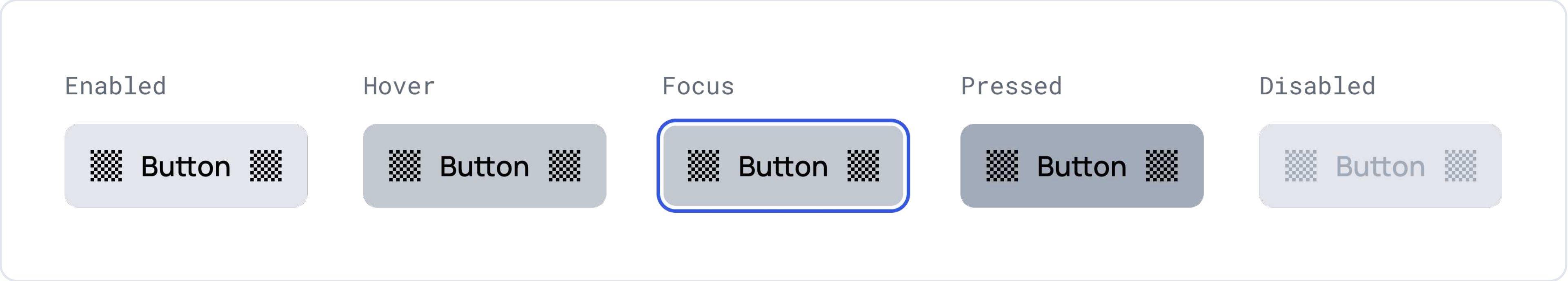
11. For icons, we also need to create properties for showing/hiding them.



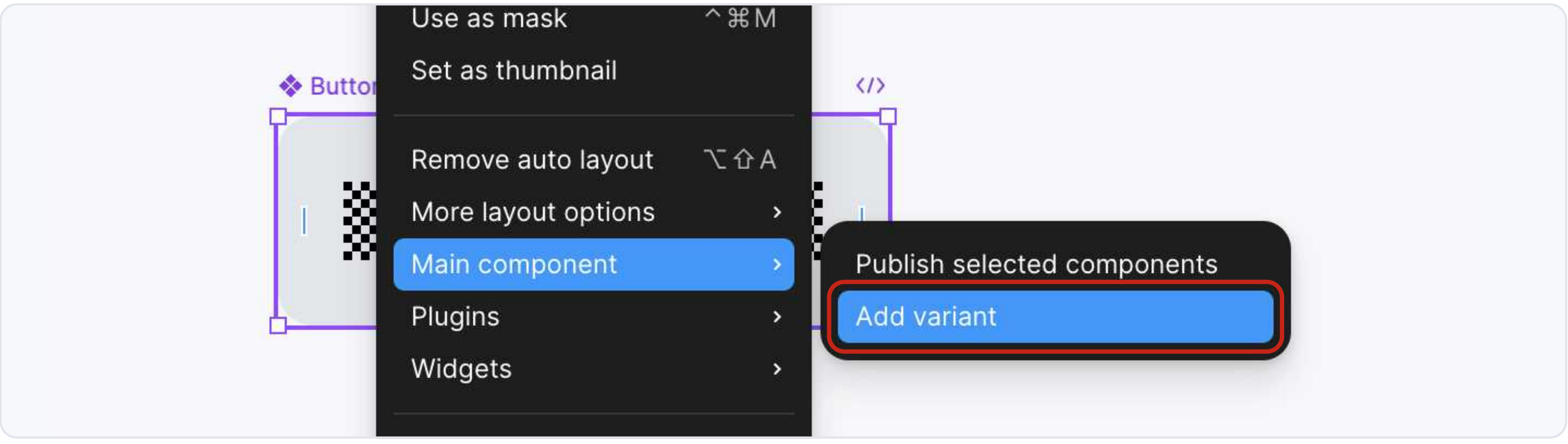


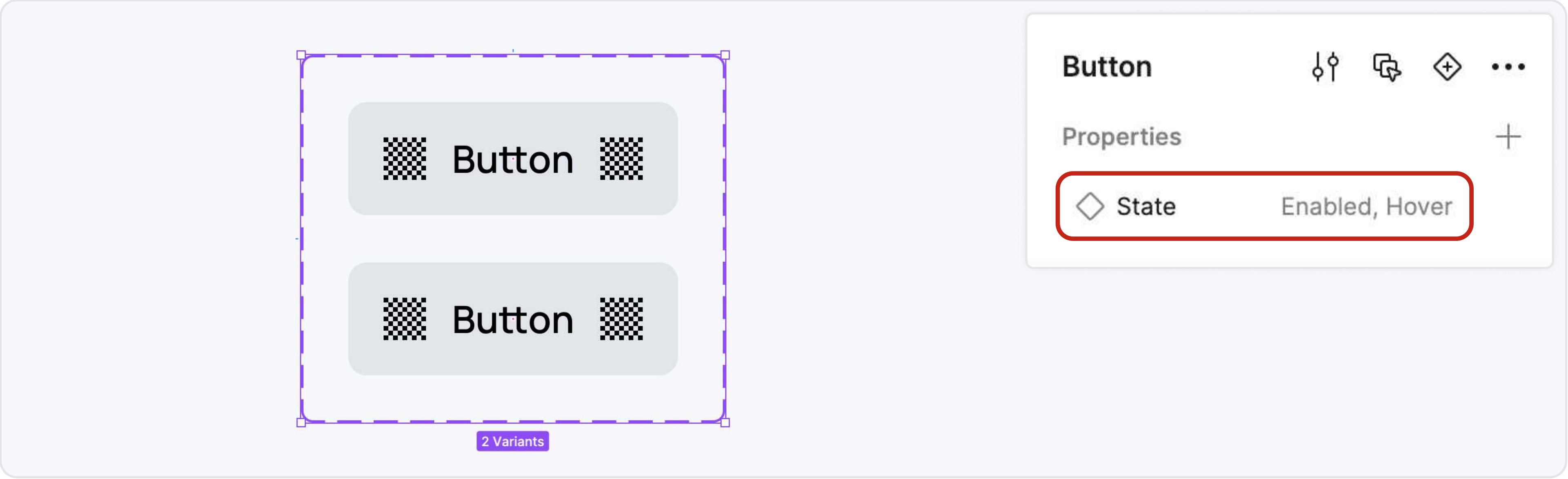
12. After setting up the properties of the component, move on to creating the button states:

    - Enabled
    - Hover
    - Focus
    - Pressed
    - Disabled

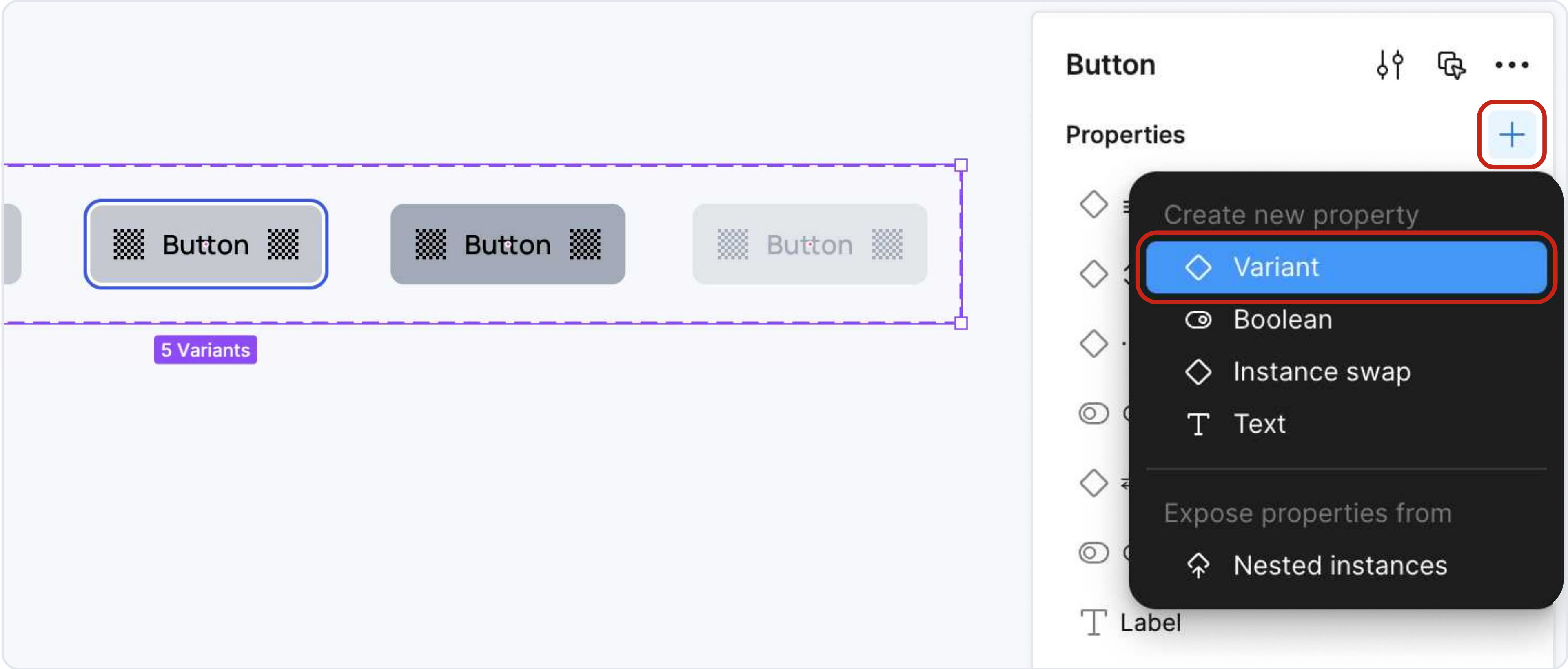


To do this, add a variant to our component and name it "State".

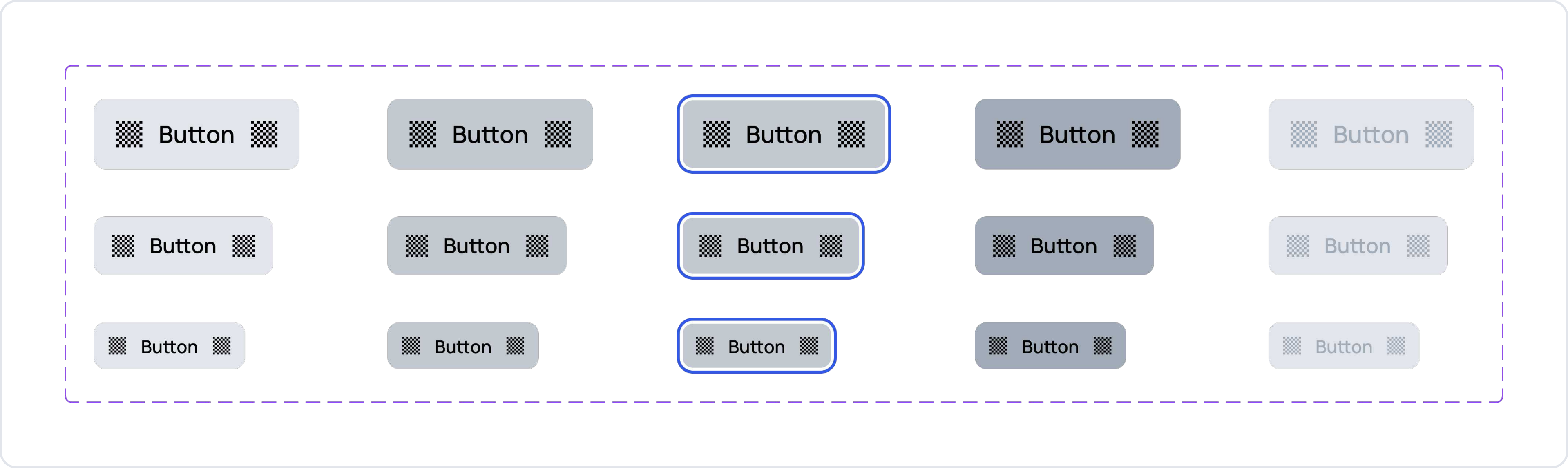




13. After creating all the button states, we can add a new property called “Size”.



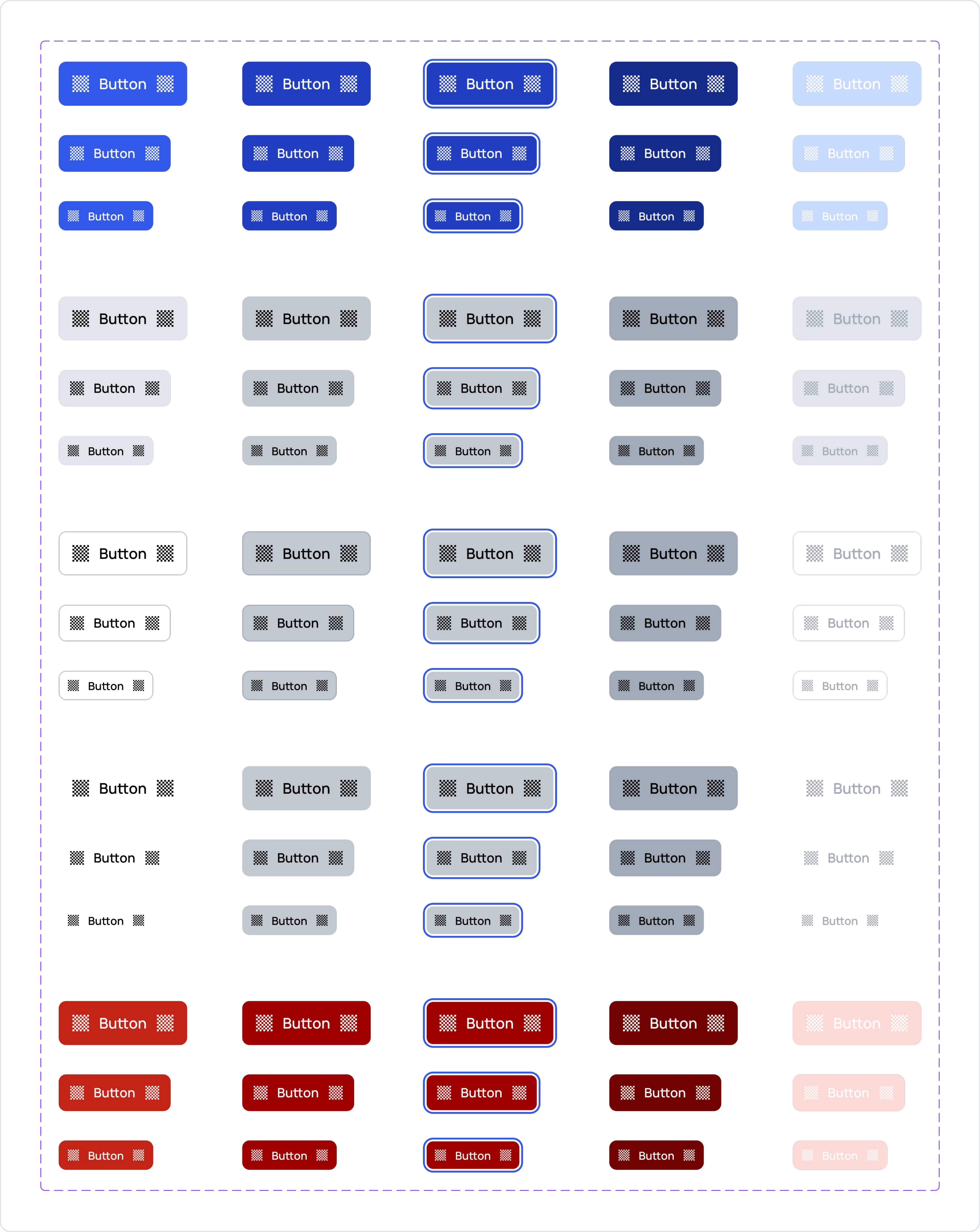
14. Next, you can select and copy all our variants and change the size for each state, or you can copy one state, change its size, and then add all the states.



15. In the same way, you can add types for the button:

- Accent
- Primary
- Secondary
- Tertiary
- Danger
- Ghost, etc.

16. In the end, you may end up with the following structure:

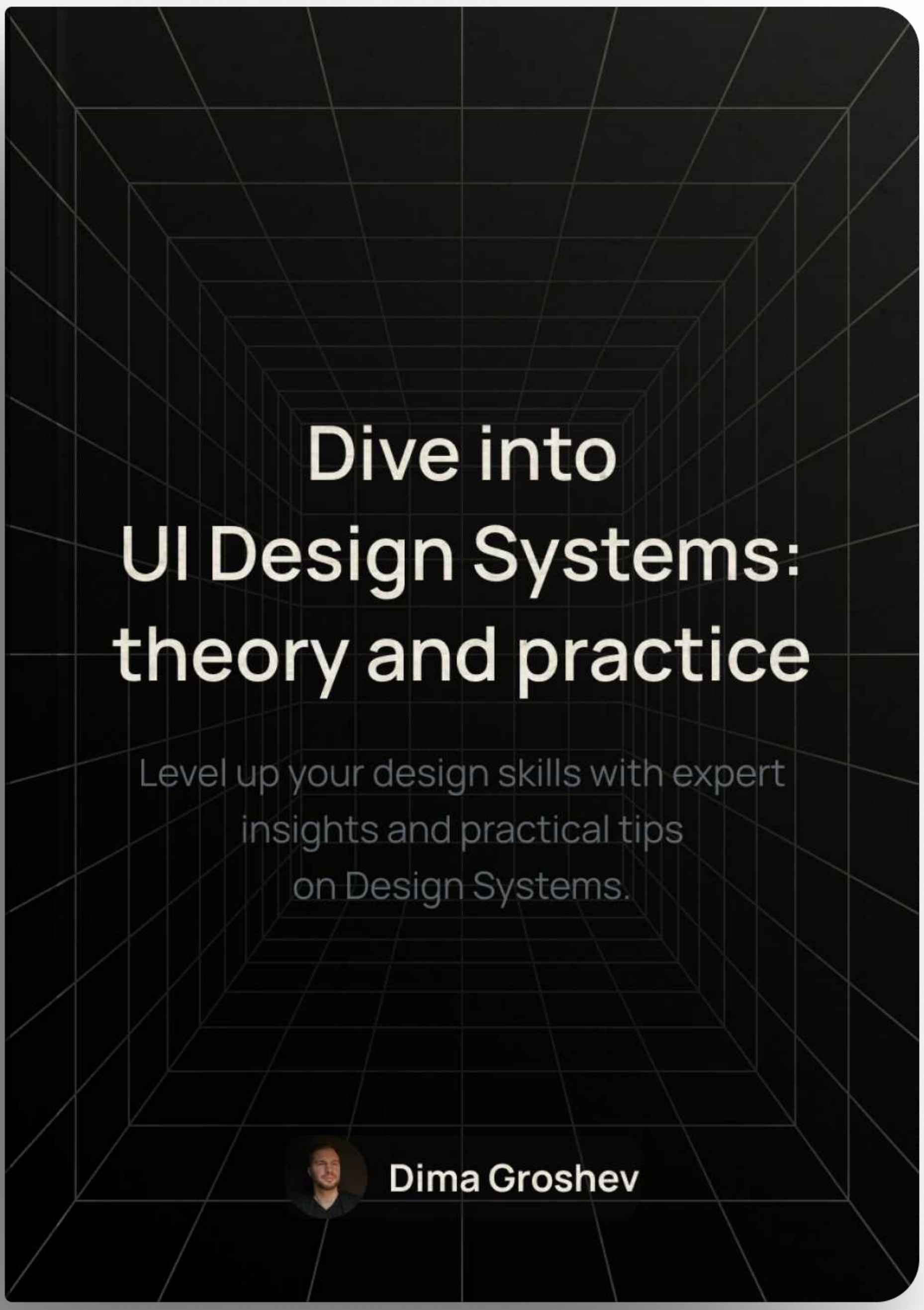


# Thank you for reading the free sample of "Dive into UI Design Systems: theory and practice."

I hope you enjoyed it! If you have any questions about the chapters, feel free to email me at hello@123d.one.

To get the full version of the book, visit
https://123d.one/products/design-system-book

Thank you and happy reading!


Dive into
UI Design Systems:
theory and practice
Level up your design skills with expert insights and practical tips on Design Systems.
Dima Groshev

Get full eBook