SPARKBOX

# Enterprise Design Systems Thinking

Ben Callahan

@bencallahan

# Work

Gap

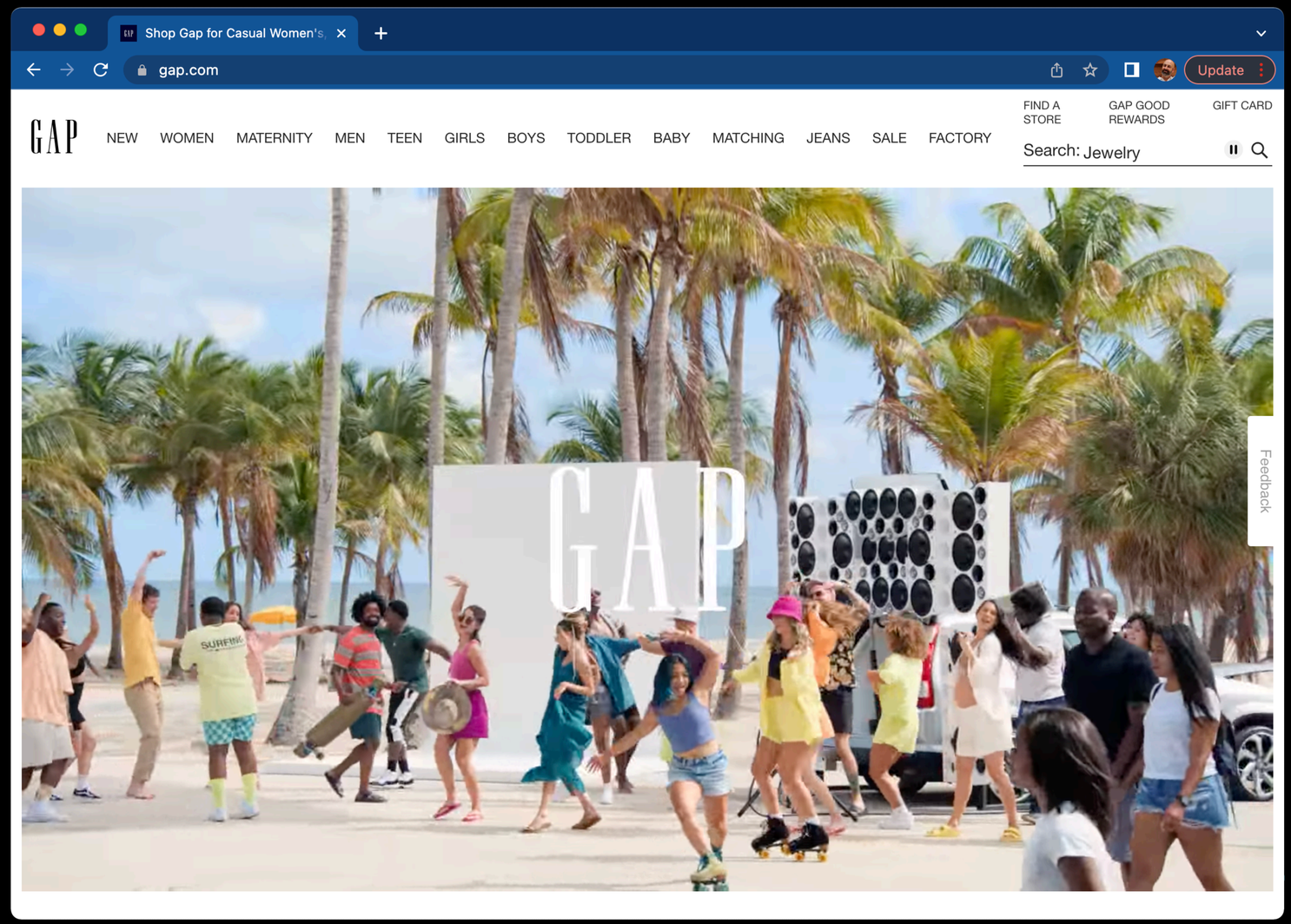
Stanford University

CarMax

University of Georgia

Docusign

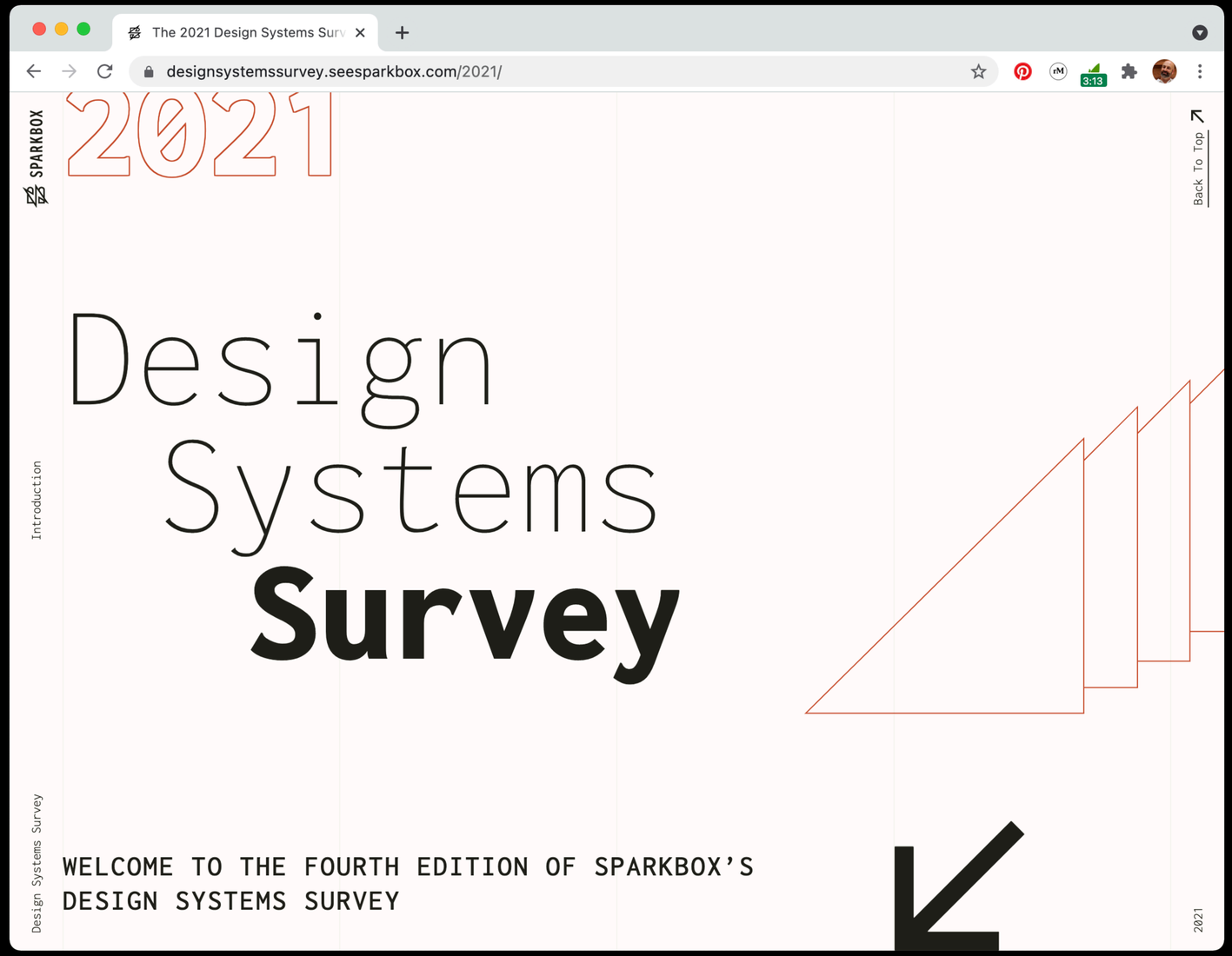
(and many more!)



# Research

2018–Today

https://bit.ly/sparkbox-ds-survey



There is no one way.

I want to show you
how to find your way.

# Today

Anatomy of a Design System

Maturity Model Overview

Origin Stories

Framework for How to Mature

Framework for Maintaining Stability

Design System Culture

# Connect with me

ben@heysparkbox.com

bencallahan (on Twitter & Mastodon)

https://bit.ly/connect-with-ben

# A little context, one challenge, & one success.

a quick story...

# Takeaway #1

There is still a lot of confusion about design systems

# Takeaway #2

Teams are not fully realizing the benefits of a systematic design practice

# Context

**Audience**

↑

**Brand**

identity, mission, vision, values, positioning, brand colors, etc.

**Audience**

↑

**Printed Materials**

↑

**Brand**

identity, mission, vision, values, positioning, brand colors, etc.

**Audience**

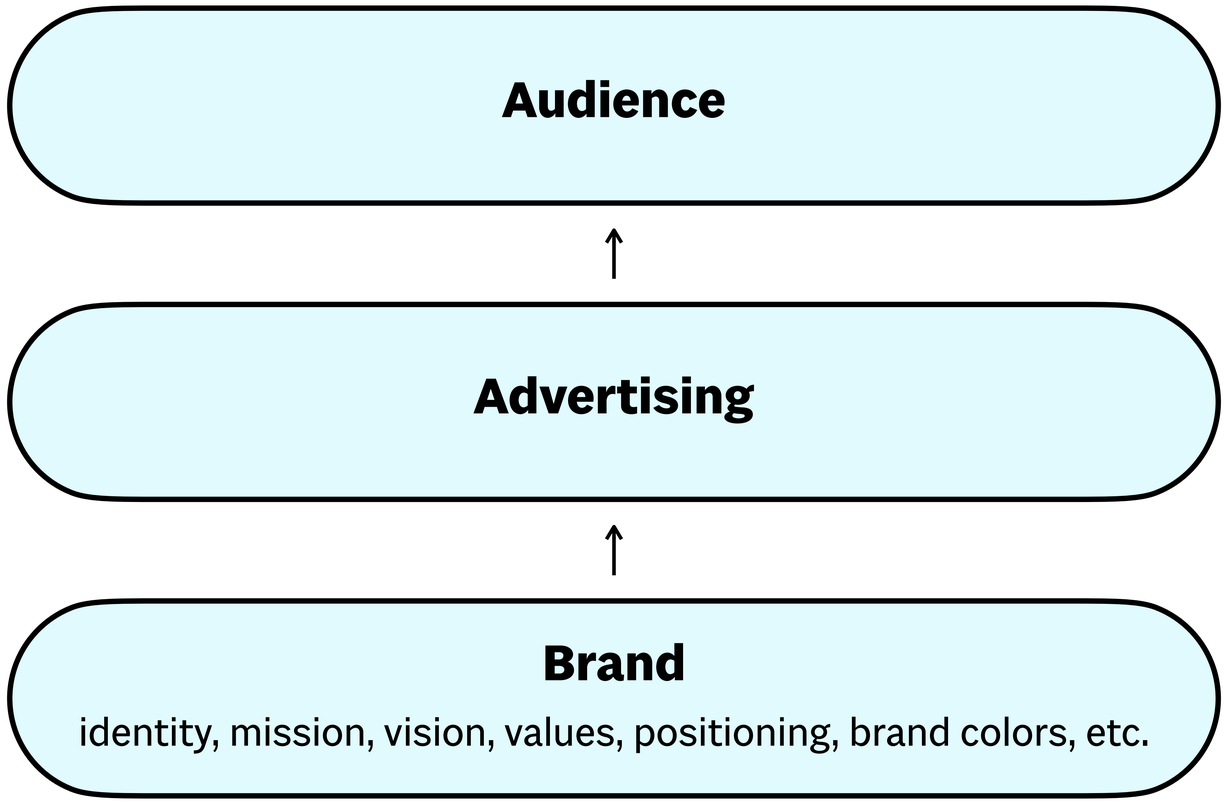
↑

**In-Store Experiences**

↑

**Brand**

identity, mission, vision, values, positioning, brand colors, etc.


Audience
Advertising
Brand
identity, mission, vision, values, positioning, brand colors, etc.

**Audience**

↑

**Digital Experiences**

websites, native applications, kiosks, etc.

↑

**Brand**

identity, mission, vision, values, positioning, brand colors, etc.

**Audience**

↑

**Digital Experiences**

websites, native applications, kiosks, etc.

↑

**Design System**

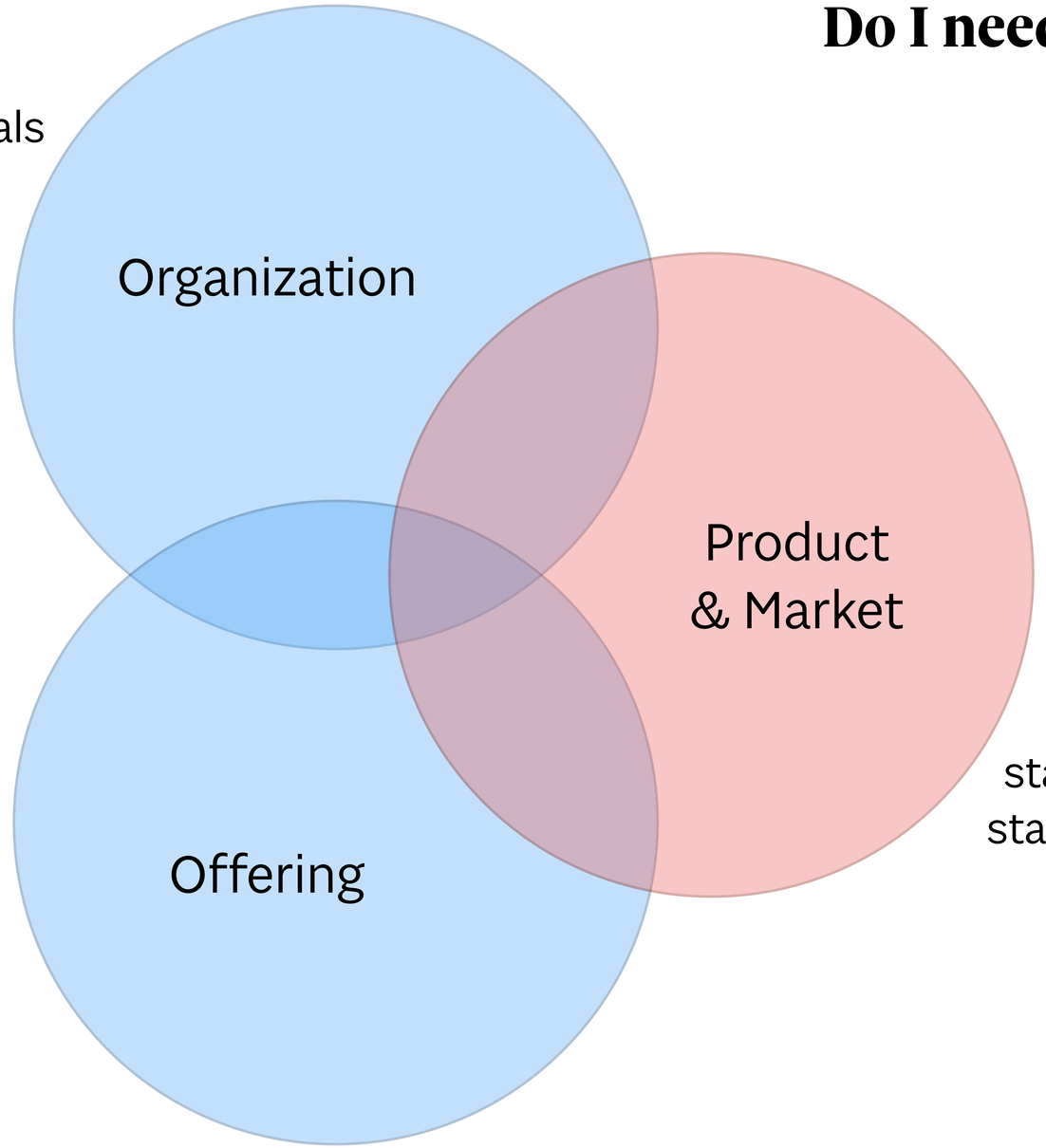
↑

**Brand**

identity, mission, vision, values, positioning, brand colors, etc.

# Do I need a Design System?

number of teams
maturity of individuals
complexity of org



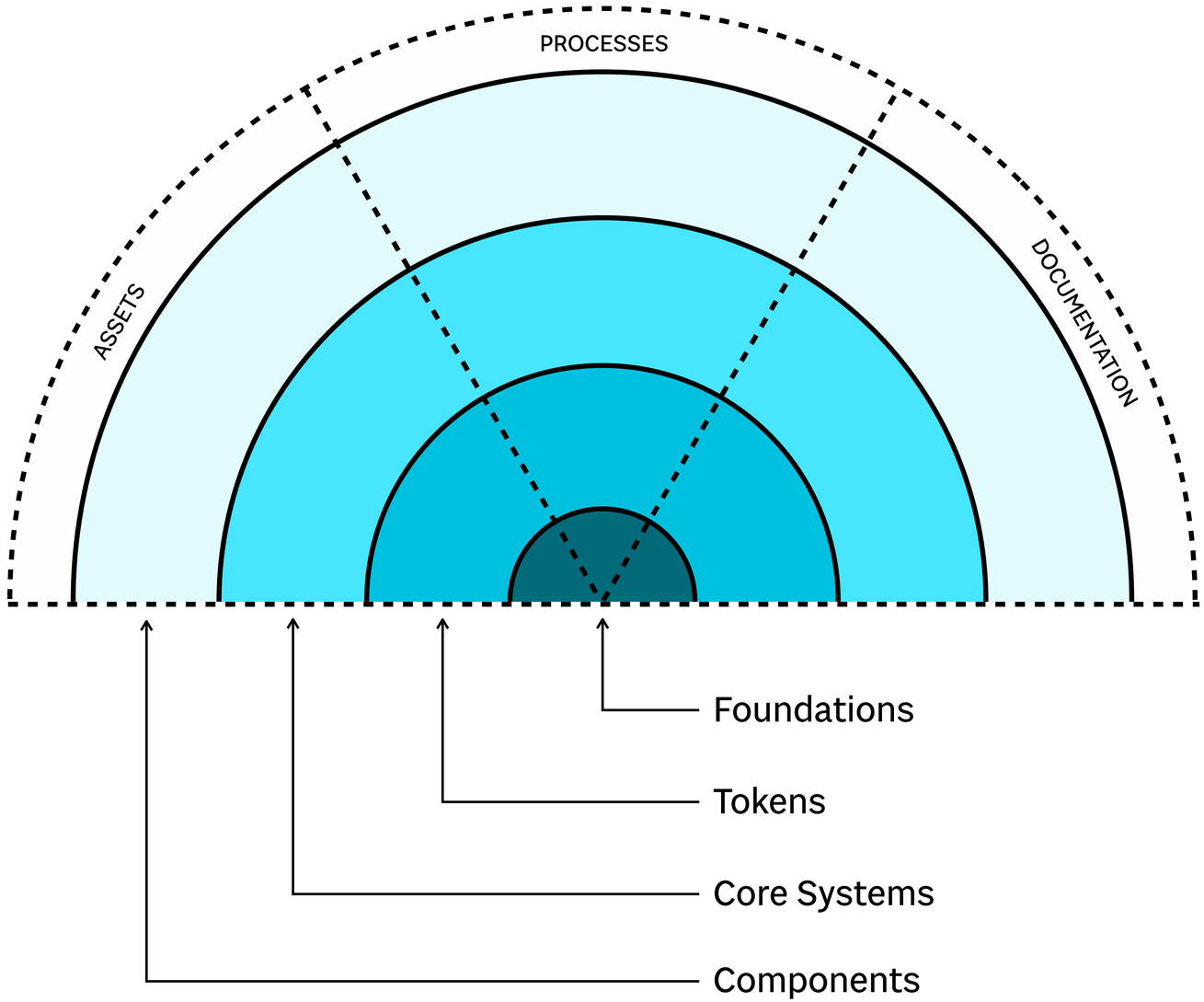

Scale

Longevity

stability of the market
stability of the product

number of products
number of channels

# The Anatomy of a Design System



The **Foundations Layer** is a subset of the organization's brand to be used in the design and development of digital interfaces

design principles

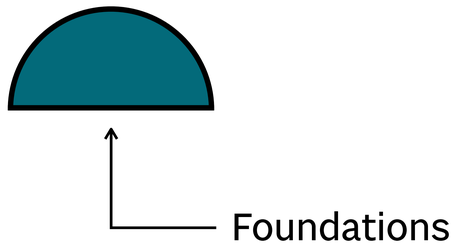
iconography guidelines

logos and typefaces

illustration guidelines

sound guildelines

voice and tone




photography guidelines

motion guidelines

layout guidelines

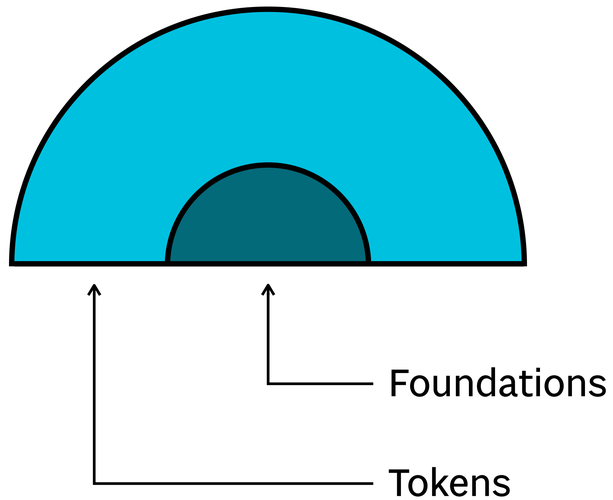
elevation guidelines

how we think about color

The **Tokens Layer** is a set of codified design decisions based on the Foundations Layer

[token-name] = [token-value]

color-primary-brand = blue

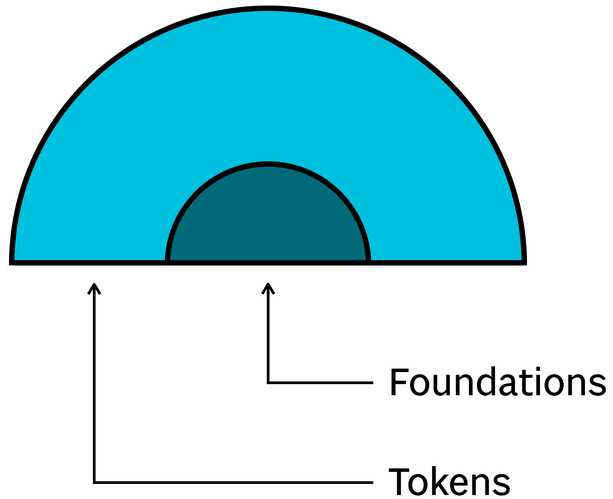


The **Tokens Layer** is a set of codified design decisions based on the Foundations Layer

consistent design

ease of change

multiplatform support



The **Tokens Layer** is a set of codified design decisions based on the Foundations Layer

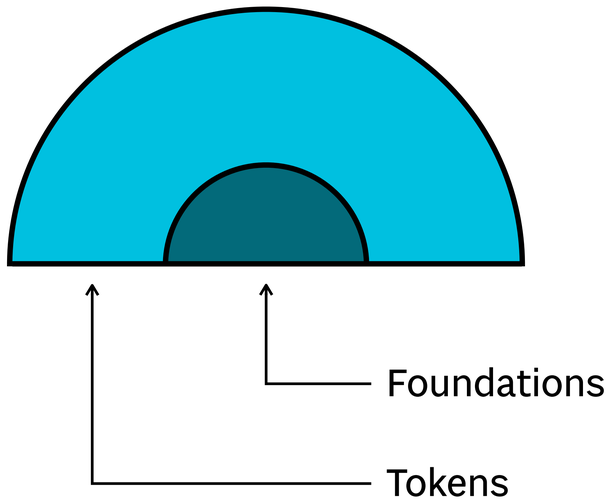
type

color

spacing

elevation

opacity




radii

volume

borders

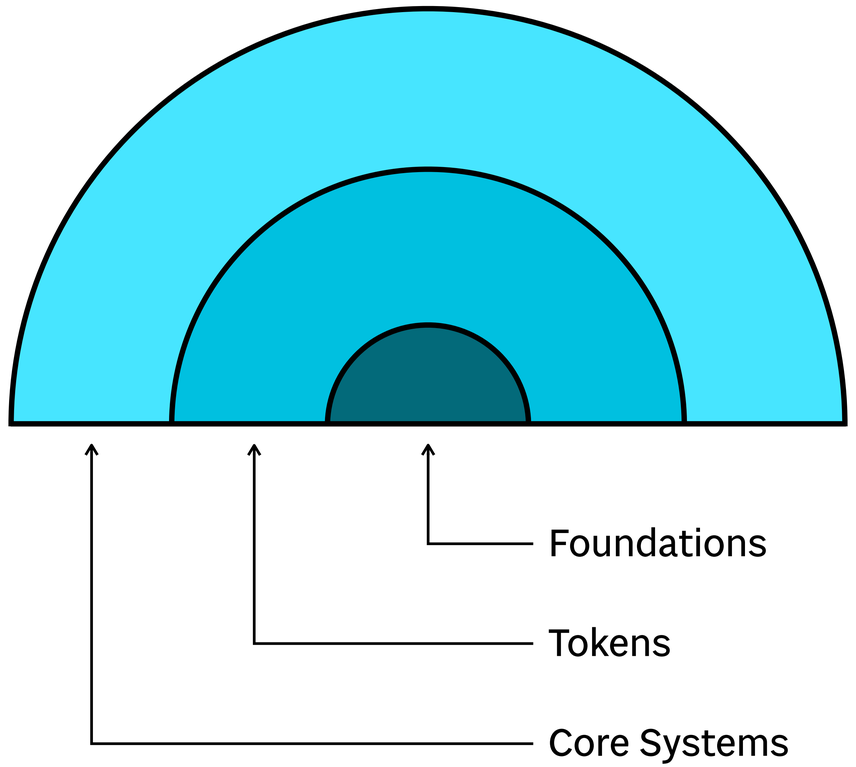
timing

The **Core Systems Layer** is a set of “building block” systems used to solve common interface challenges

iconography systems

layout & grid systems

theming systems

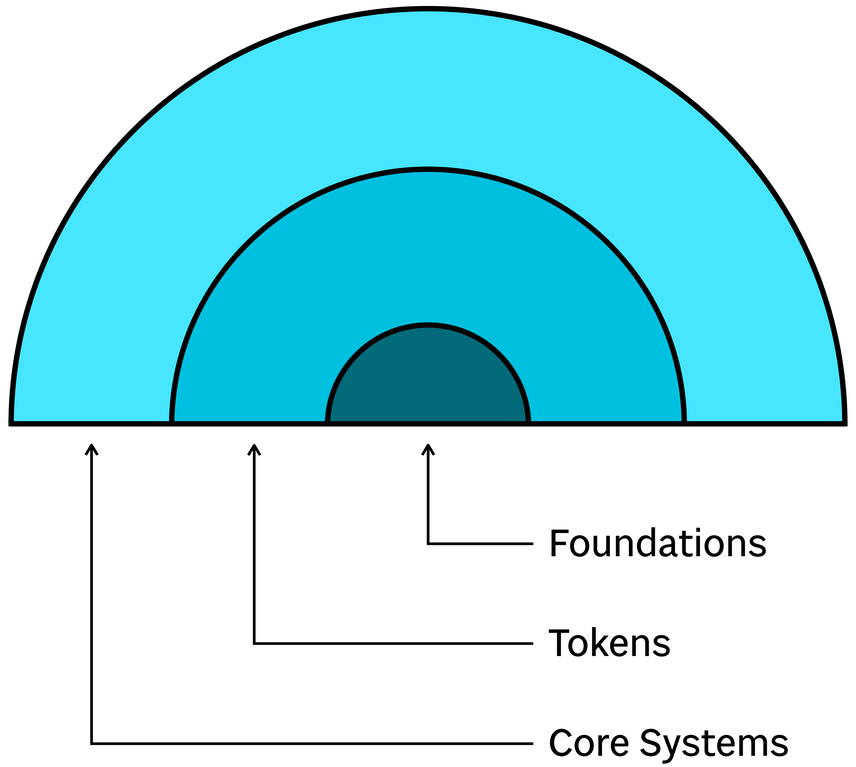



color systems

type scale systems

animation systems

# The Fundamental Layers



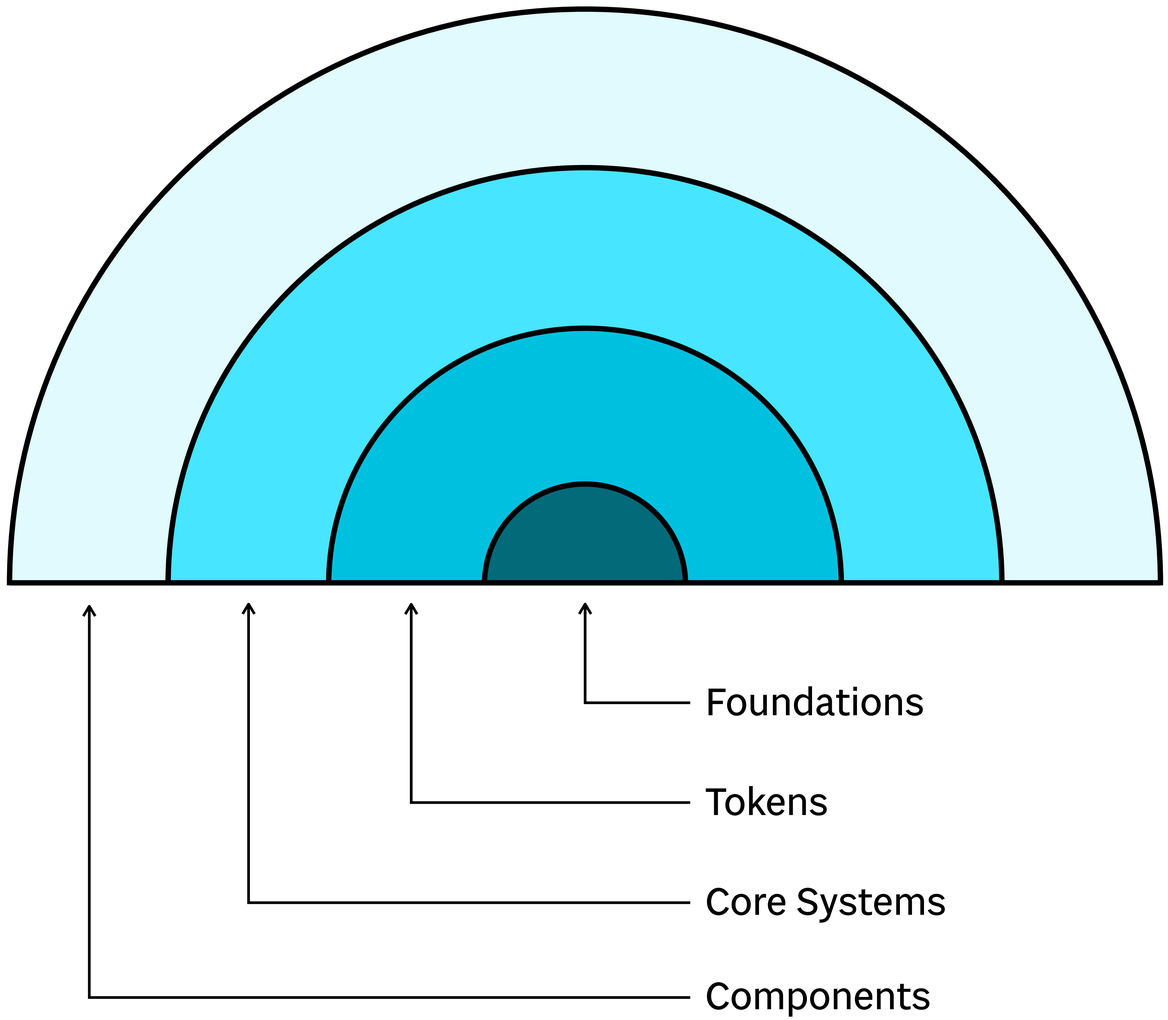
The **Components Layer** contains the reusable parts of a digital interface

notifications

buttons

search bars

text inputs



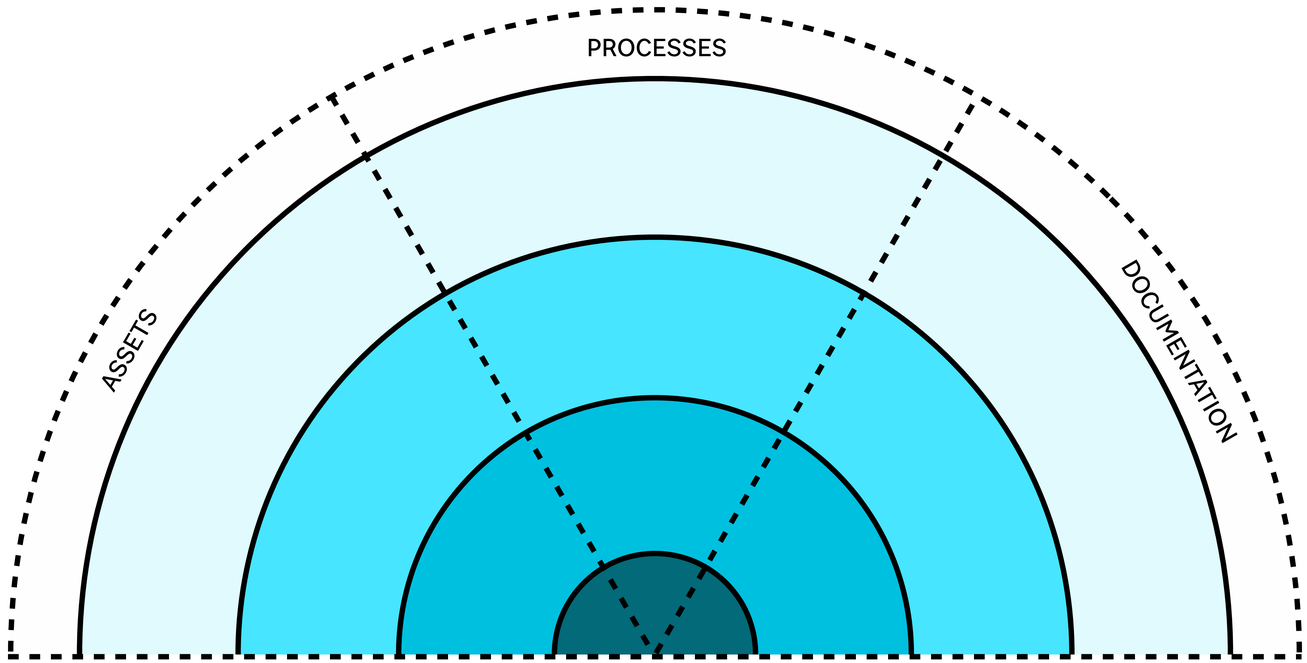

breadcrumbs

tabs

tooltips

radio buttons

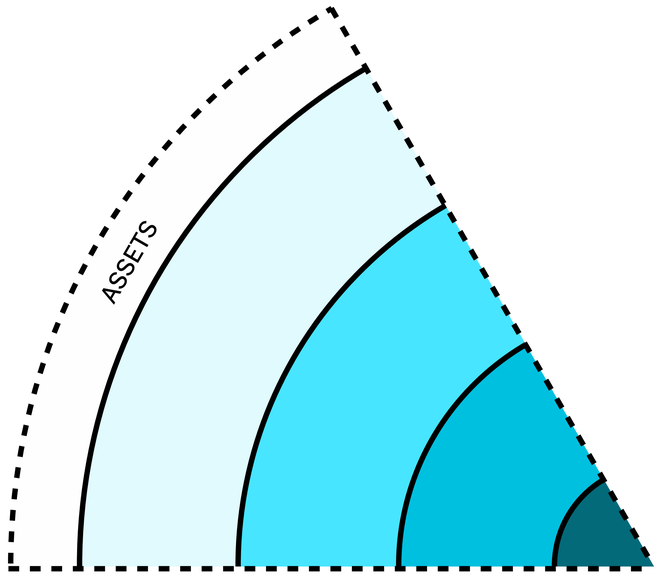
# Processes




## Assets

## Documentation

**Assets** are the tangible things
that make each layer usable




Content files

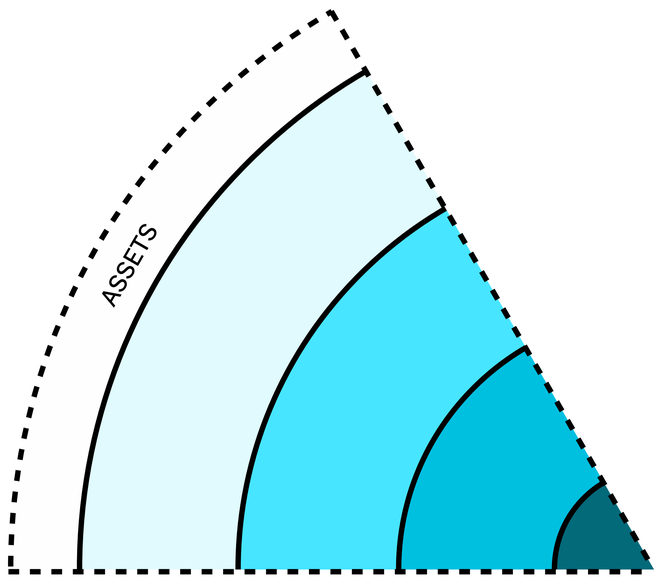
UX & Design files

Development files & Repositories

Image files

Font files

**Assets** are the tangible things
that make each layer usable

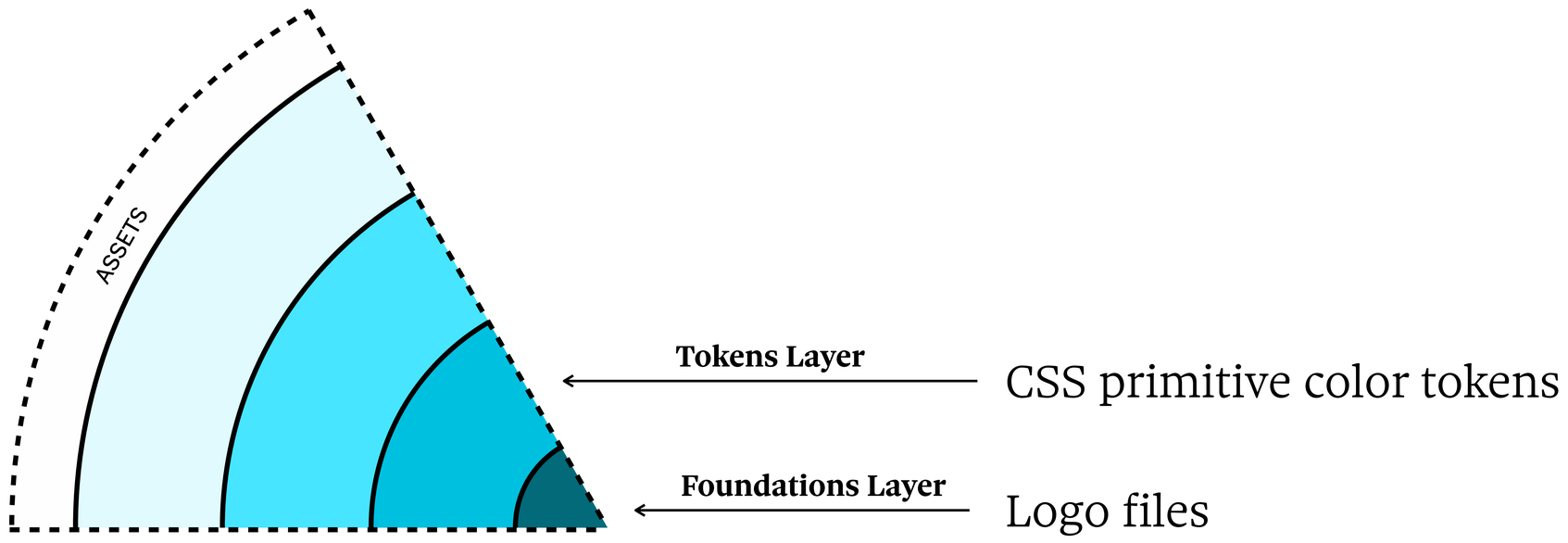



JPG

PNG

**Foundations Layer** ← Logo files

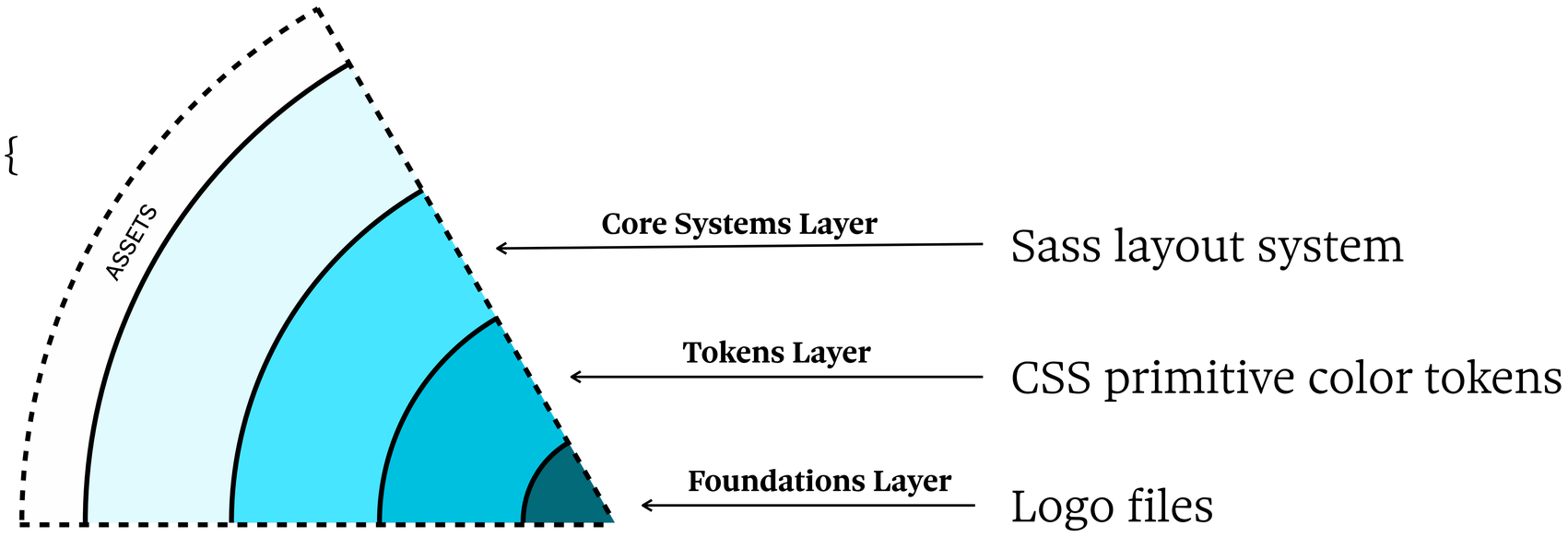
**Assets** are the tangible things
that make each layer usable



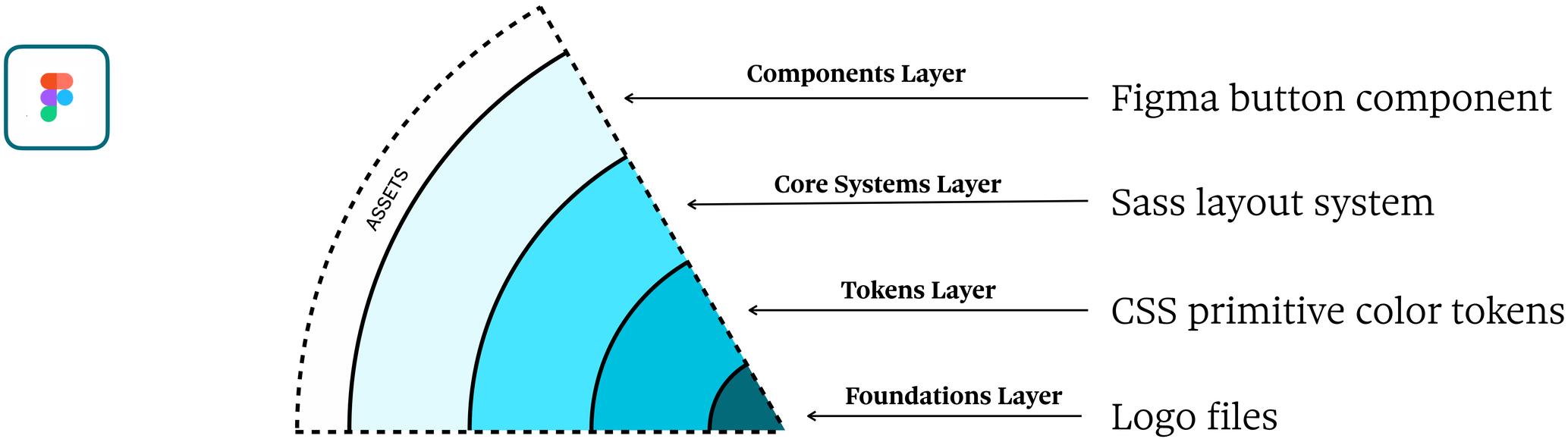

```
--white: #fff;
--gray-10: #eee;
--gray-50: #777;
--gray-80: #333;
```

**Assets** are the tangible things
that make each layer usable

```
@mixin make-row($gutter) {
  display: flex;
  flex-wrap: wrap;
  gap: $gutter;
}
```



**Assets** are the tangible things
that make each layer usable



**Processes** define human behaviors as
we work on and with the design system

Communication

Governance & Testing

Contribution

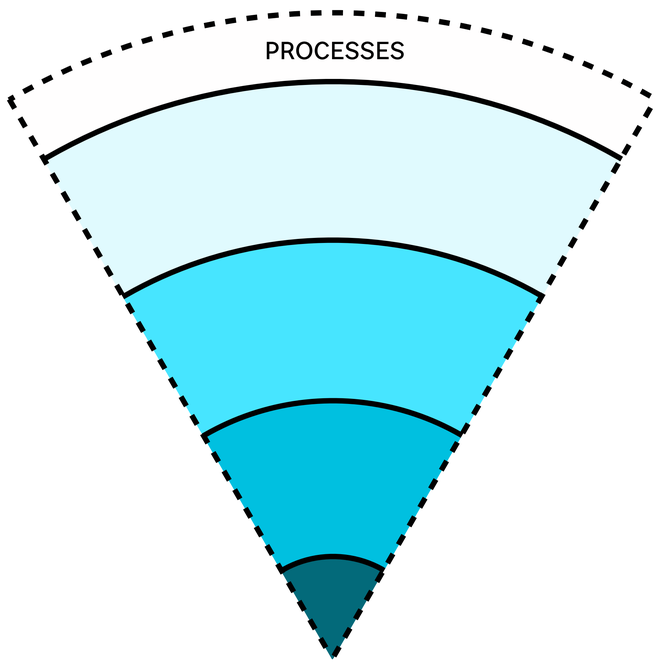
Synchronization

Deprecation

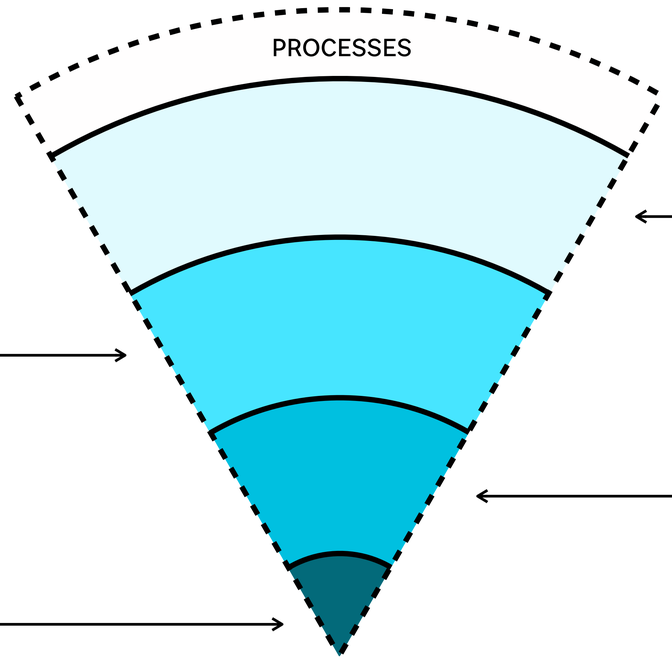
Release & Versioning

Extension & Variation

Onboarding



**Processes** define human behaviors as
we work on and with the design system




**Components Layer**
Accepting component modifications from subscribers

**Core Systems Layer**
Deprecating an old core system

**Tokens Layer**
Modifying the value of a token

**Foundations Layer**
Communication between Brand and DS teams

**Documentation** explains why something is the way it is and a breakdown of when and how to use it

What do we believe about this?

Why do we believe this?

What is this thing?

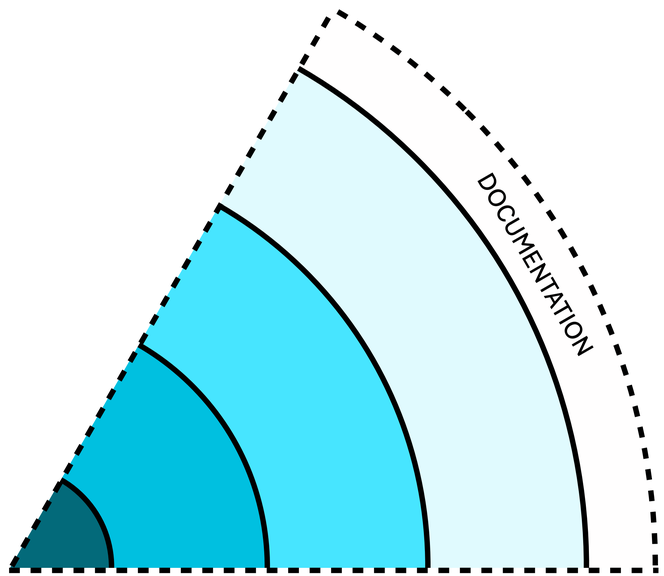
How do I use or migrate to a thing?

When should I use (and not use) a thing?

Who owns a thing?

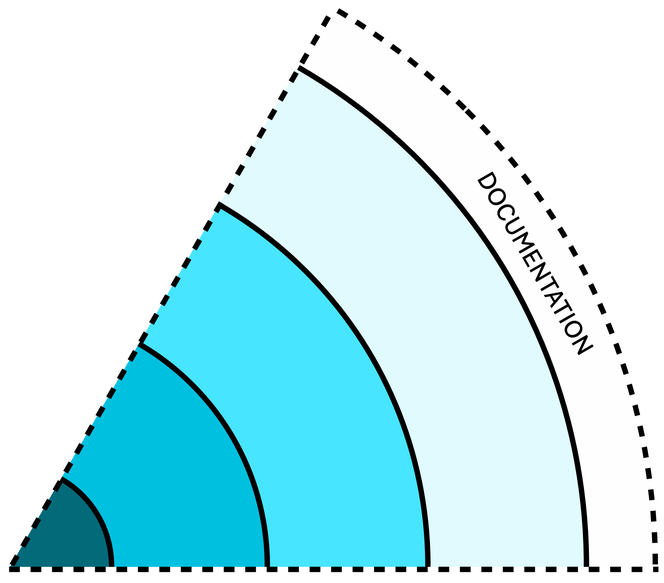
Where does a thing live?

Examples of a thing in use



**Documentation** explains why something is the way it is and a breakdown of when and how to use it

Point of view on color → **Foundations**



Color – Carbon Design System

carbondesignsystem.com/guidelines/color/overview/

Carbon Design System

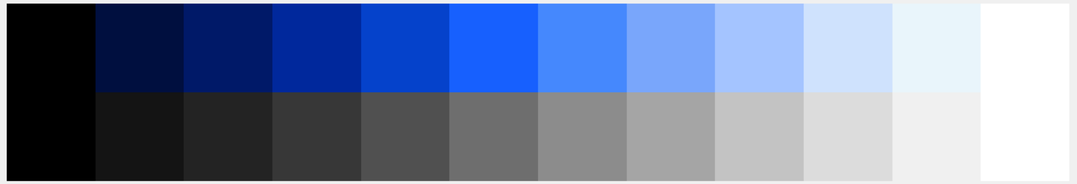
All about Carbon
Case studies
What's happening
Designing
Developing
Contributing
Migrating
Guidelines
2x Grid
Accessibility
Content
Color
Icons
Pictograms
Motion
Spacing
Themes
Typography
Components
Patterns
Community assets
Data visualization
Help
GitHub

Overview | Usage | Code | Implementation

# Color anatomy

Carbon's default themes are derived from the IBM Design Language color palette. The neutral gray family is dominant in the default themes, making use of subtle shifts in value to organize content into distinct zones.

The core blue family serves as the primary action color across all IBM products and experiences. Additional colors are used sparingly and purposefully.



Alert Colors

## Layering model

Colors in the neutral gray palette are layered on top of each other to create depth and spatial associations. The layering model defines the logic of how colors stack on top of each other in a UI when using the Carbon themes. Aspects of the layering model are built directly into the themes, color tokens, and components.

The layering model differs between the *light* and *dark* themes.

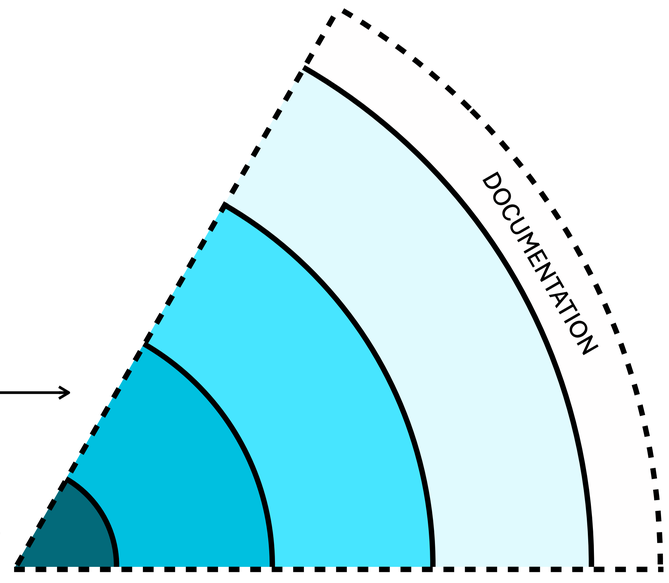
- In the light themes, layers alternate between White and Gray 10 with each added layer.
- In the dark themes, layers become one step lighter with each added layer.

Point of

**Documentation** explains why something is the way it is and a breakdown of when and how to use it

Explanation of how to name tokens → **Tokens Layer**

Point of view on color → **Foundations**



Explanation of how to

Point of

Color – Carbon Design System

carbondesignsystem.com/guidelines/color/overview/

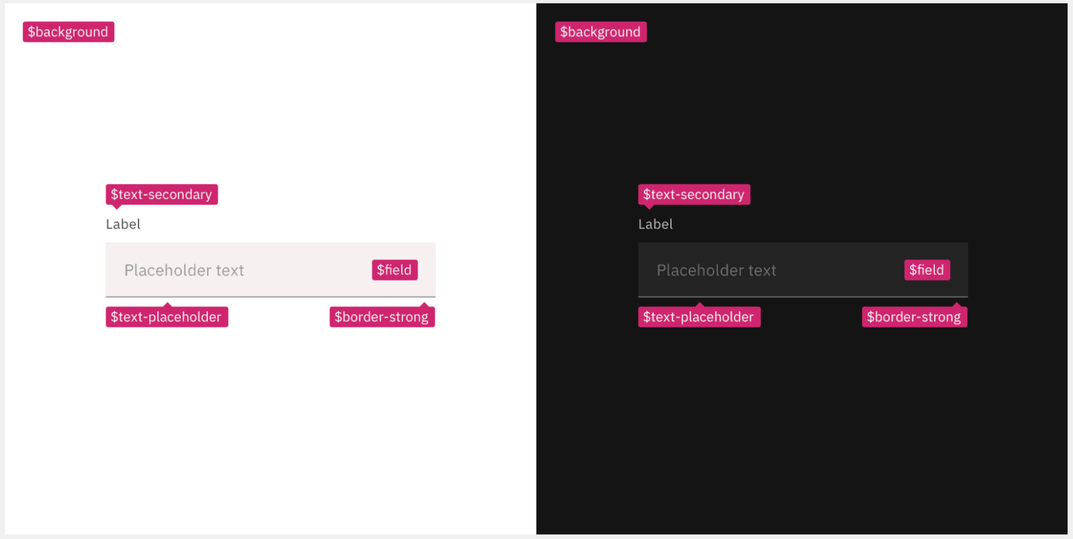
Carbon Design System

All about Carbon
Case studies
What's happening
Designing
Developing
Contributing
Migrating
Guidelines
2x Grid
Accessibility
Content
Color
Icons
Pictograms
Motion
Spacing
Themes
Typography
Components
Patterns
Community assets
Data visualization
Help
GitHub

Overview | Usage | Code | Implementation

## Token names

For quick reference, the role of a token is represented in the token name itself to help you correctly apply tokens. The first part of the token name references the general UI element the color is being applied to, like `background`, `text`, or `border`. The second part of token name will specify its unique role within the element group like `$border-subtle` or `$text-primary`. Additionally, some tokens include aninteraction state at the end, like `$background-hover`.




Color tokens for components are the same across themes as shown by this text input using the White theme (left) and Gray 100 theme (right).

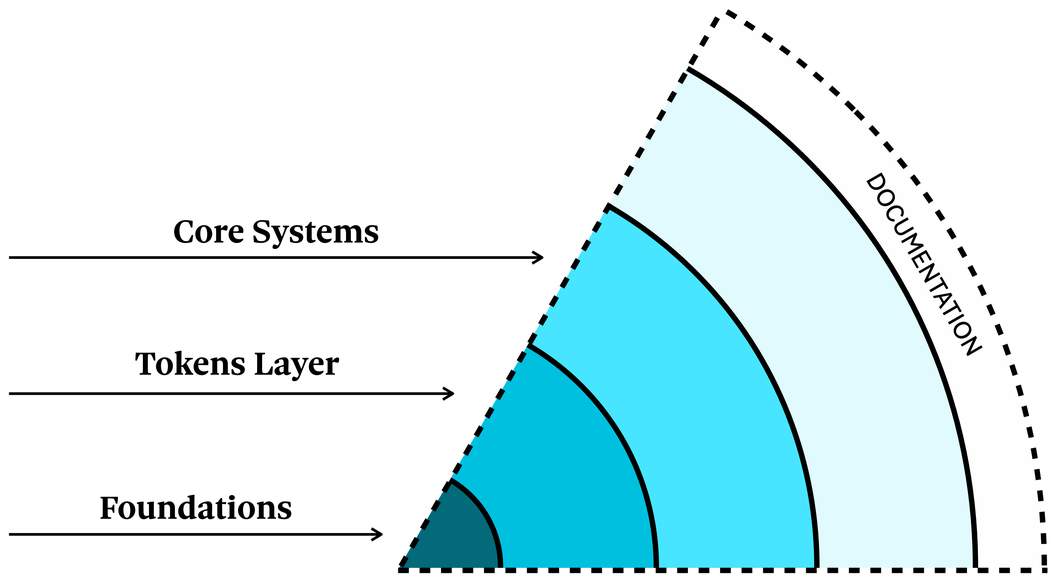
## Core tokens

**Documentation** explains why something is the way it is and a breakdown of when and how to use it

How to use the theming system

Explanation of how to name tokens

Point of view on color



How to use the th

Explanation of how t

Point of

Themes – Carbon Design Syste

carbondesignsystem.com/guidelines/themes/overview/

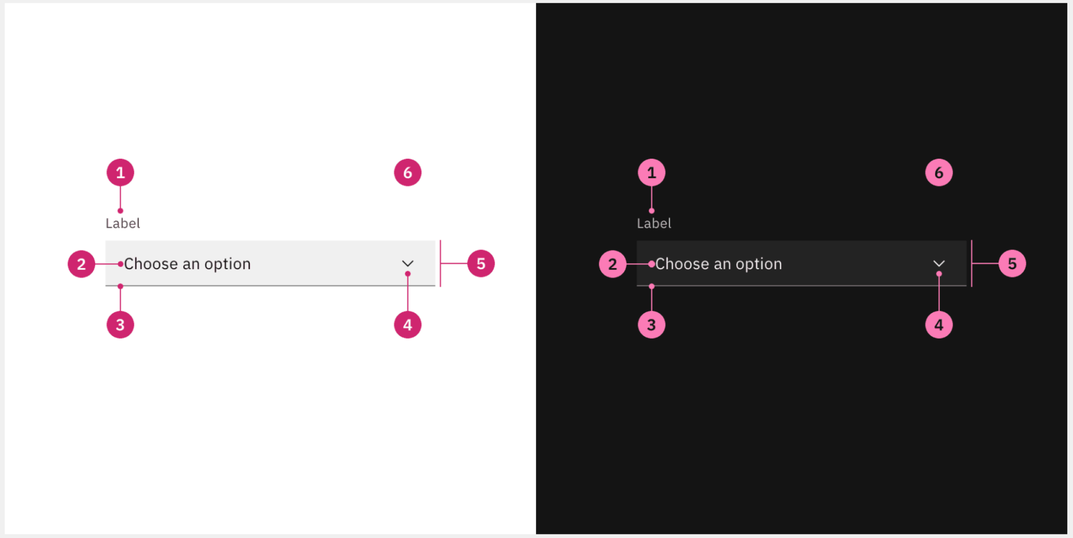
Carbon Design System

All about Carbon
Case studies
What's happening
Designing
Developing
Contributing
Migrating
Guidelines
2x Grid
Accessibility
Content
Color
Icons
Pictograms
Motion
Spacing
Themes
Typography
Components
Patterns
Community assets
Data visualization
Help
GitHub

Overview
Code

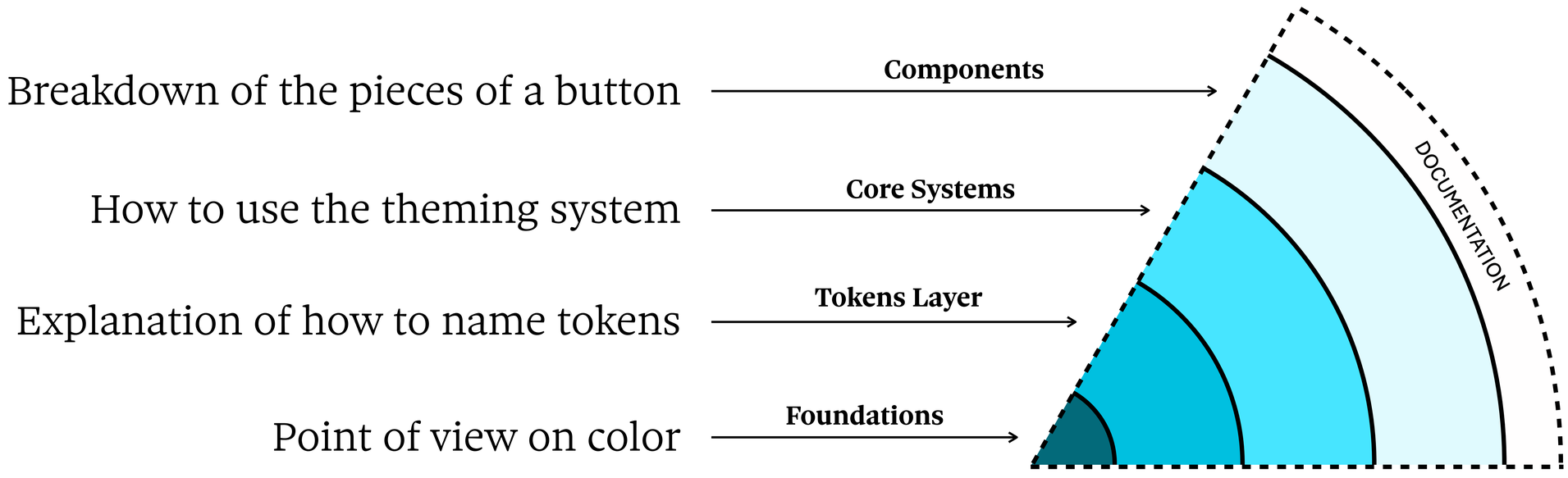
# Theming applied

The following example demonstrates the relationship between the different themes. Each theme shares the same variables and roles, with only the value changing for each individual theme.




| Key | Token | Role | White theme value | Gray 100 theme value |
|---|---|---|---|---|
| 1 | `$text-secondary` | Label color | Gray 70 | Gray 30 |
| 2 | `$text-primary` | Primary text color | Gray 100 | Gray 10 |
| 3 | `$border-strong` | Border bottom color | Gray 50 | Gray 60 |
| 4 | `$icon-primary` | Primary icon color | Gray 100 | Gray 10 |

**Documentation** explains why something is the way it is and a breakdown of when and how to use it



Breakdown of the piec

How to use the th

Explanation of how t

Point of

Buttons – Carbon Design Syste

carbondesignsystem.com/components/button/usage/

Carbon Design System

Usage Style Code Accessibility

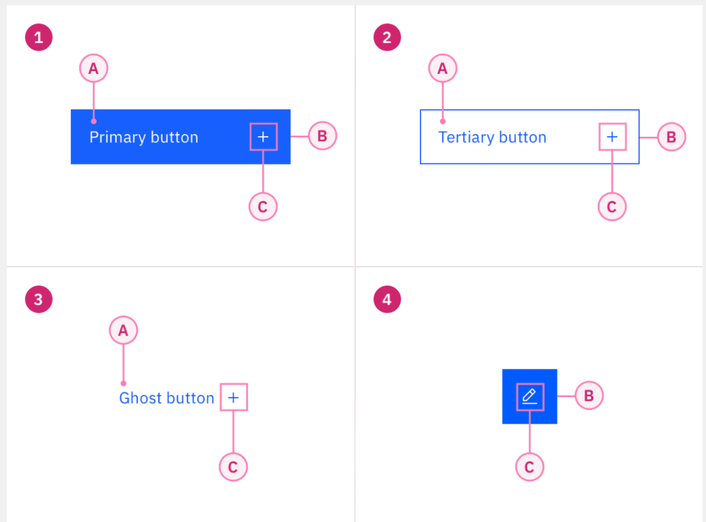
- Breadcrumb
- **Button**
- Checkbox
- Code snippet
- Content switcher
- Data table
- Date picker
- Dropdown
- File uploader
- Form
- Inline loading
- Link
- List
- Loading
- Modal
- Notification
- Number input
- Overflow menu
- Pagination
- Popover
- Progress bar
- Progress indicator
- Radio button
- Search
- Select
- Slider
- Structured list
- Tabs
- Tag
- Text input
- Tile

# Formatting

## Anatomy

A button's text label is the most important element on a button, as it communicates the action that will be performed when the user interacts with it. In a contained button the text is always left-aligned, not centered. By default Carbon uses sentence case for all button labels.

If a text label is not used, an icon should be present to signify what the button does.

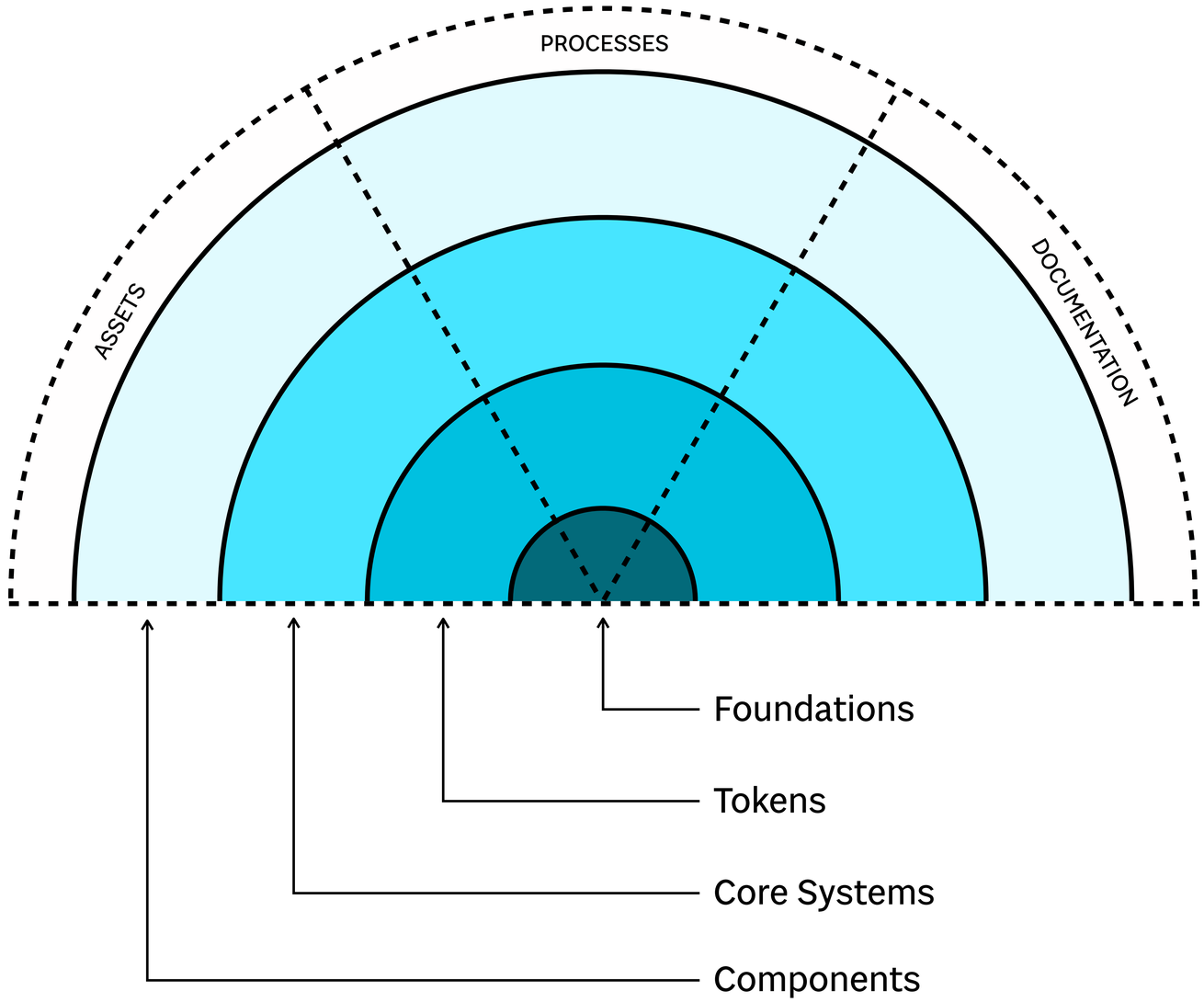



**1. Contained button**
A. Text label
B. Container

**2. Outlined button**
A. Text label
B. Container

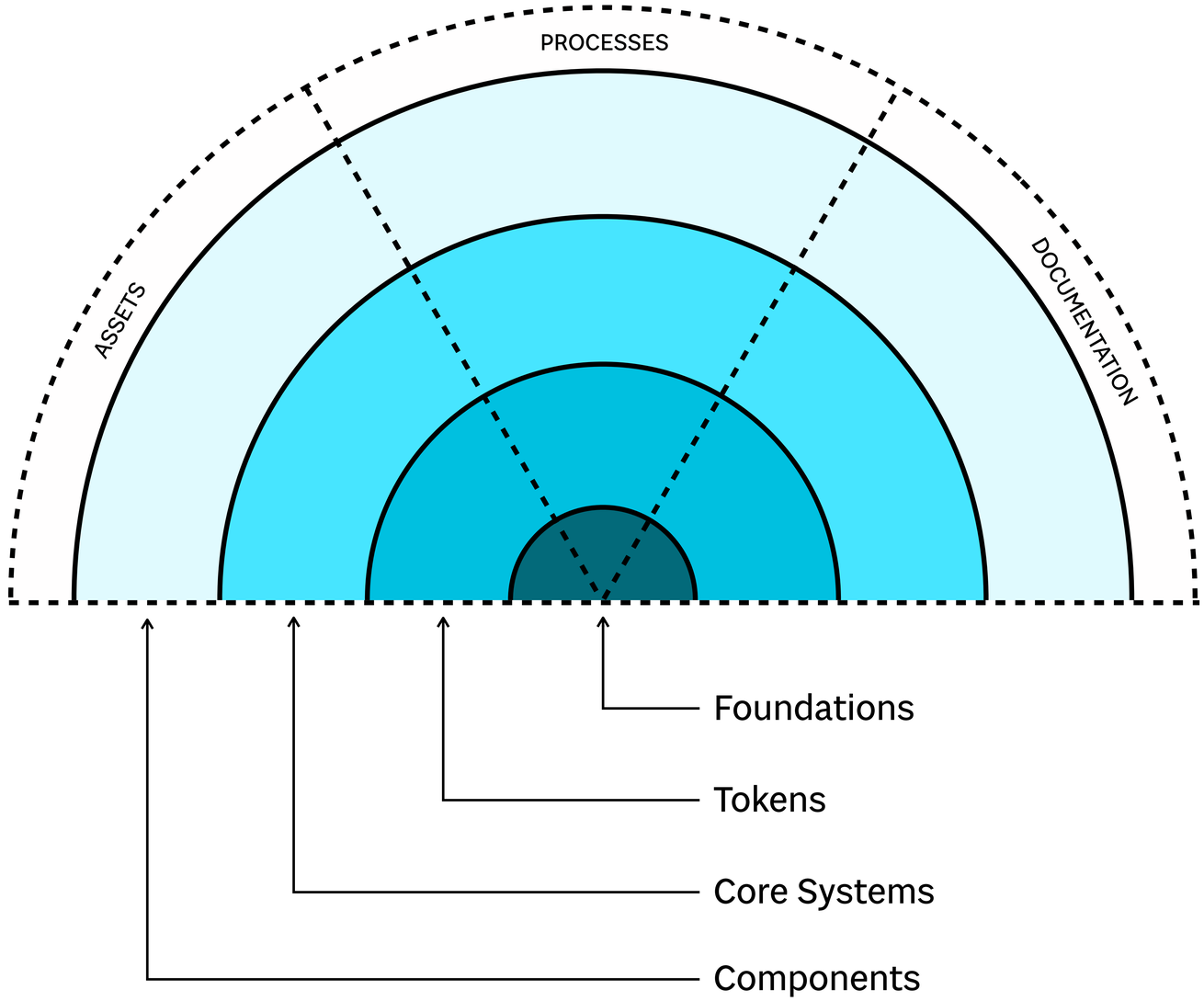
# Takeaway #1

There is still a lot of confusion about design systems



# Takeaway #2

Teams aren’t realizing the full benefits of a systematic design practice

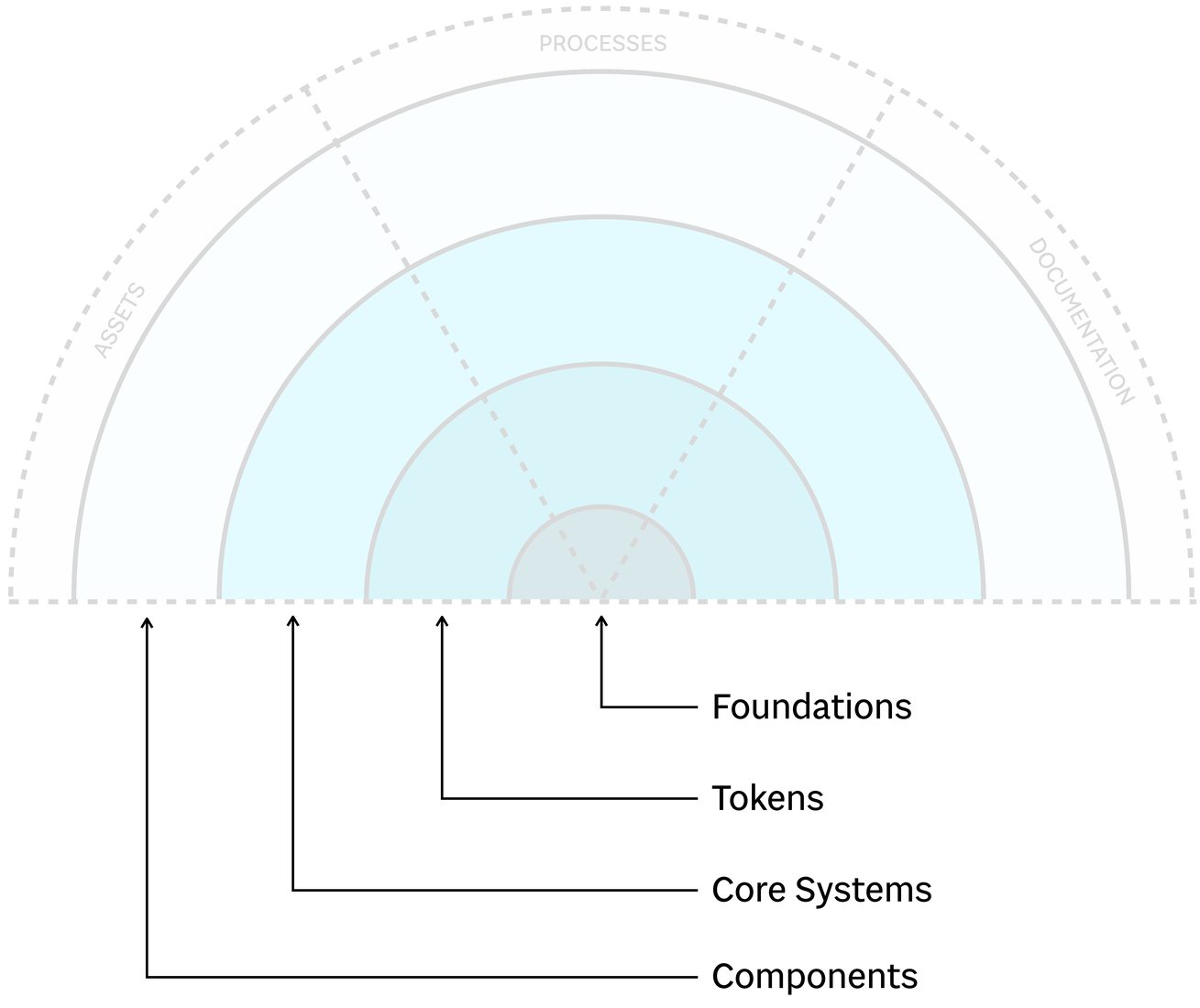


## Consistency

Individuals say, “The buttons in my product all look the same”

Departments say, “All the digital products we offer look and feel the same”

Cultures say, “Every interaction a customer has with us aligns with our brand”

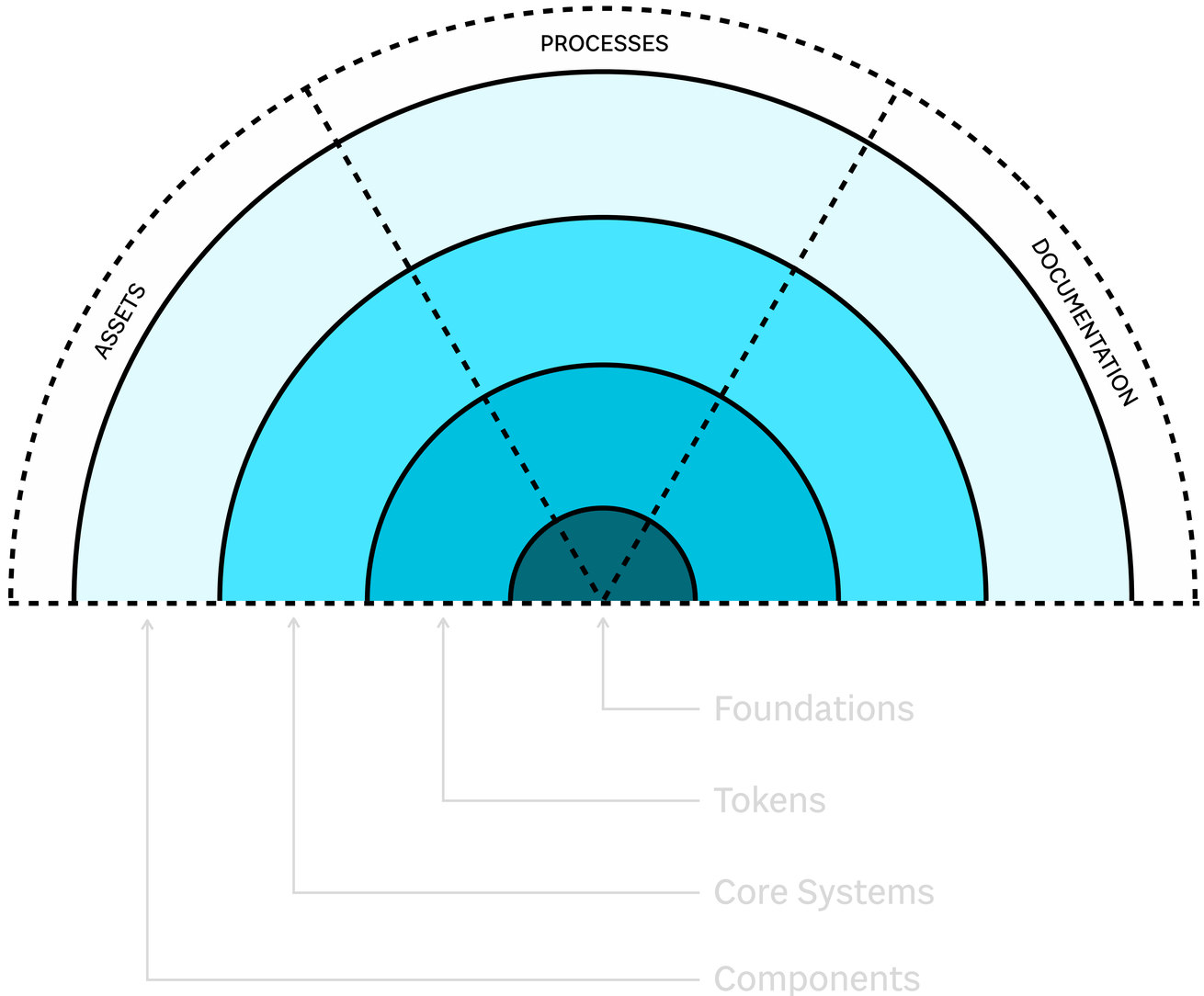


## Efficiency

Individuals say, “I can design or build a feature faster”

Departments say, “Our product organization gets to market faster”

Cultures say, “We understand each other”

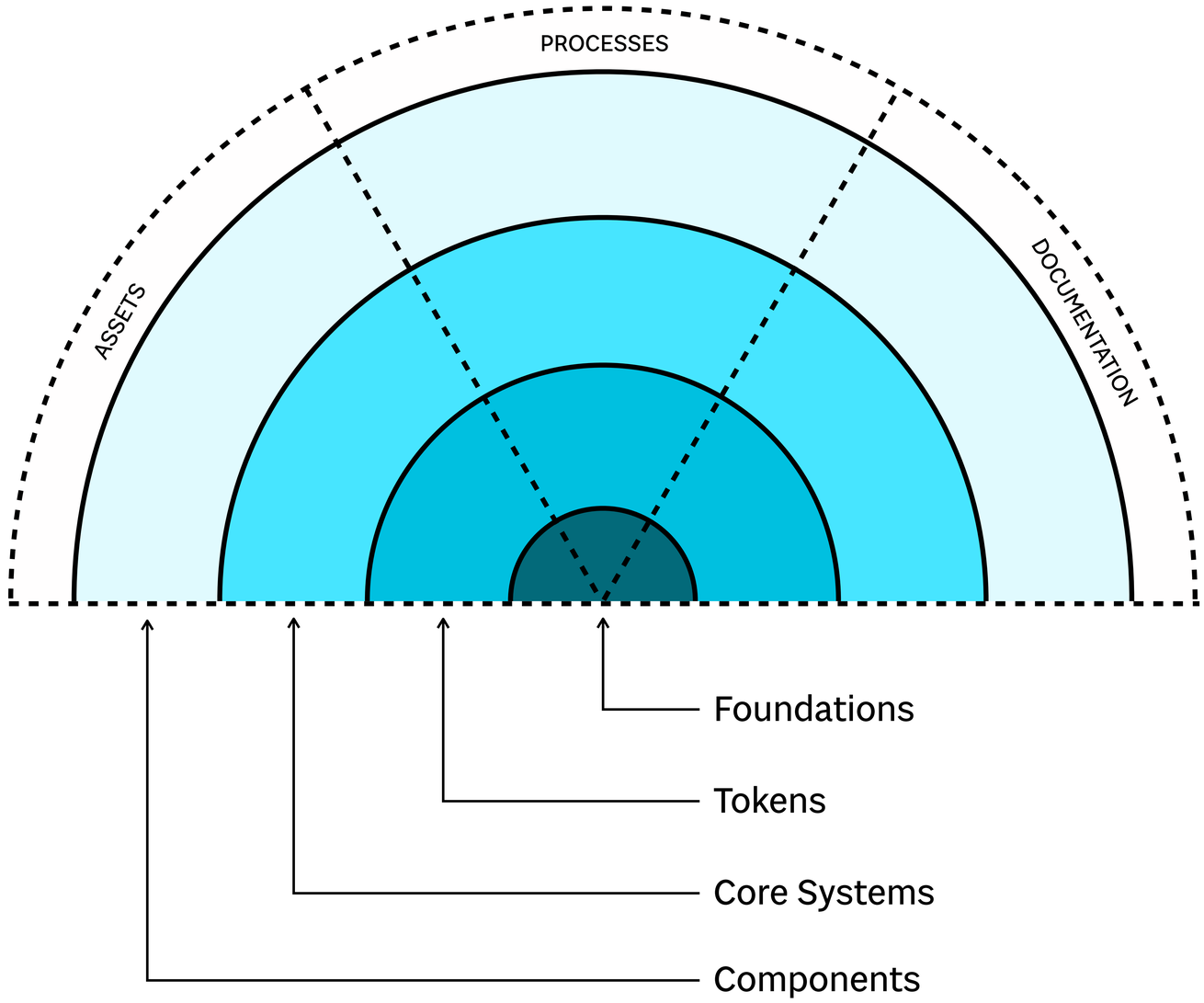


## Meaningful Efficiency

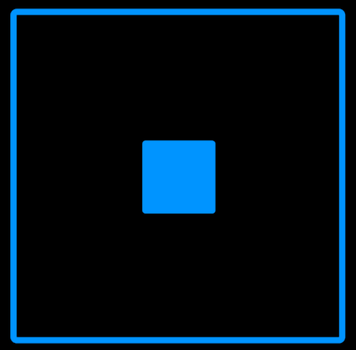
We understand each other

## Authentic Consistency

Every interaction a customer has with our company aligns with our brand



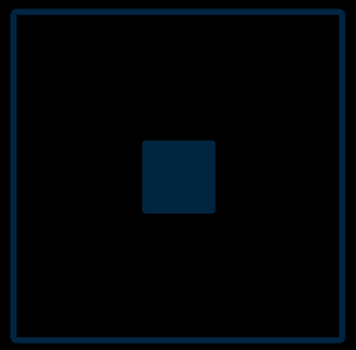
# Maturity Model Overview



## Stage 1

Building
Version One

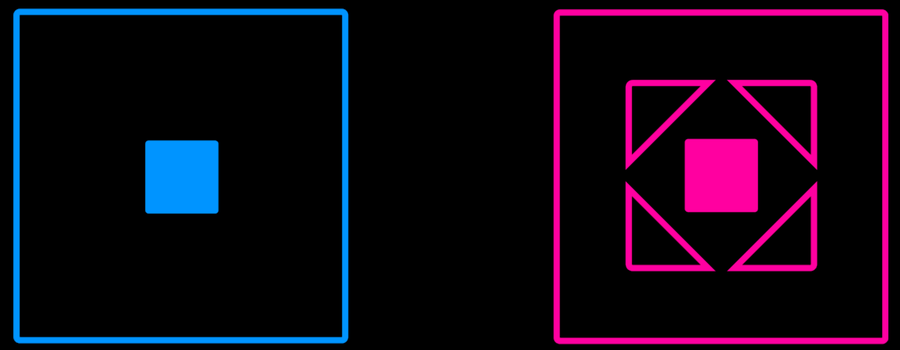
Focus: Discover the right first version



Stage 1

Building
Version One

Transition: Release v1

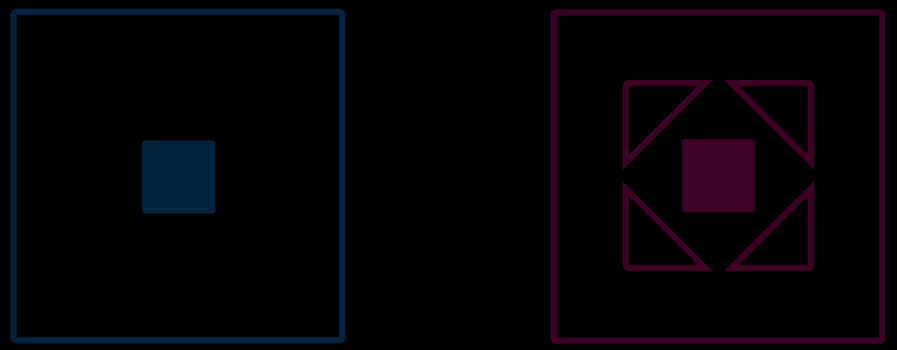


Stage 1 → Stage 2

Building
Version One

Growing
Adoption

Focus: Make it easy and obvious to adopt

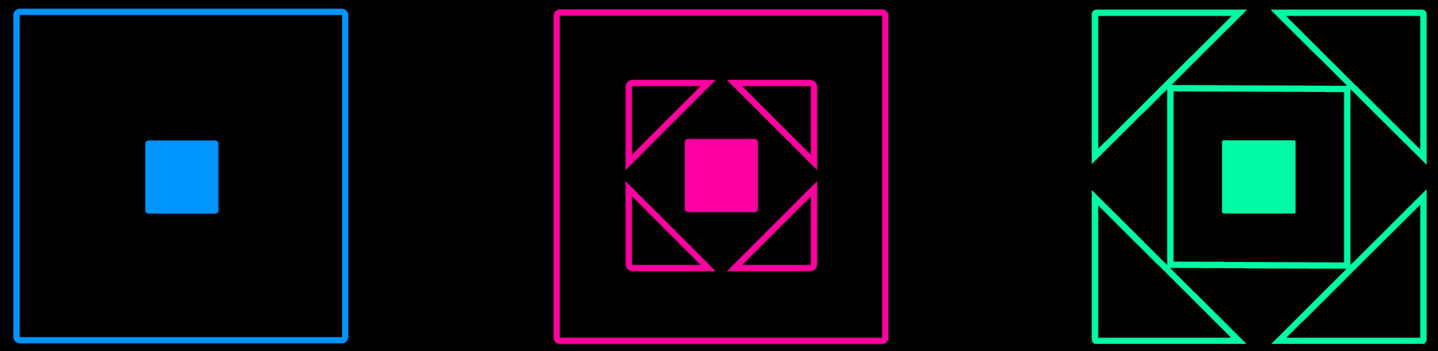


Stage 1 → Stage 2 →

Building
Version One

Growing
Adoption

Transition: Your focus shifts to challenges other than adoption



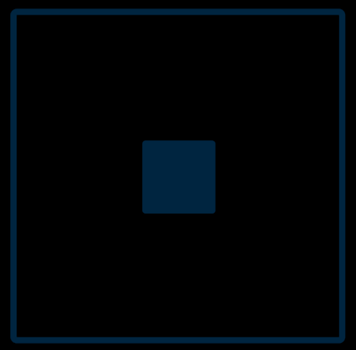
Stage 1 → Stage 2 → Stage 3

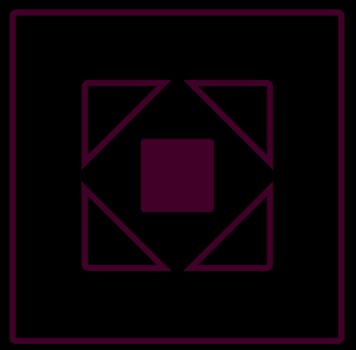
Building Version One

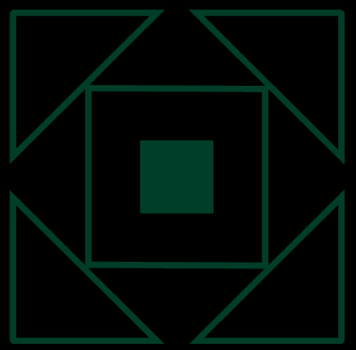
Growing Adoption

Surviving the Teenage Years

Focus: Build a product team and mindset







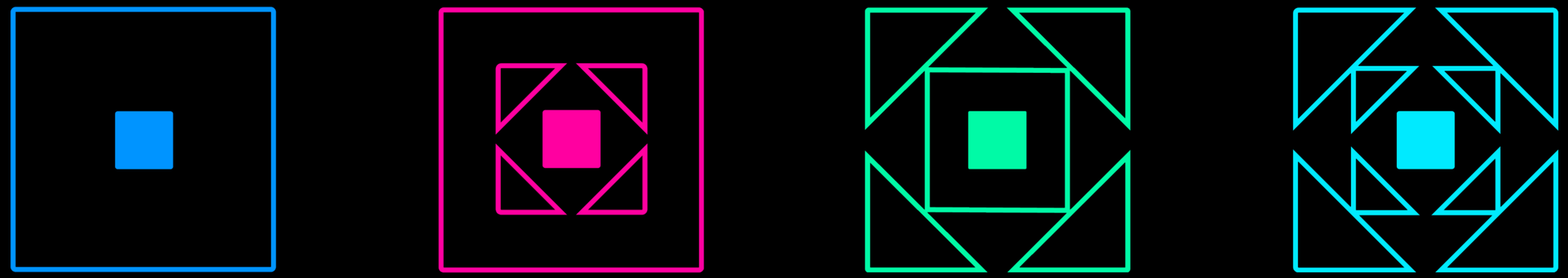
Stage 1 → Stage 2 → Stage 3 →

Building
Version One

Growing
Adoption

Surviving the
Teenage Years

Transition: The organization sees the system as an influential leader



Stage 1 → Stage 2 → Stage 3 → Stage 4

Building Version One

Growing Adoption

Surviving the Teenage Years

Evolving a Healthy Program

Focus: Mature into stable leadership

# Origin Stories

Origin Stories

# Top Down

Executive leadership was involved, aware,
and actively supportive of our first release.

Origin Stories

# Grassroots

Executive leadership was uninvolved, unaware, or unsupportive of our first release.

Origin Stories

## Top Down

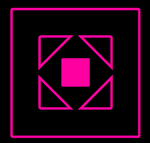
- More visibility
- Formal budget
- Sanctioned team
- Broad scope
- High pressure to deliver

## Grassroots

- Less visibility
- Informal (or no) budget
- Unsanctioned team
- Narrow scope
- Low pressure to deliver

Top Down ⟷ Grassroots

Top Down



Grassroots

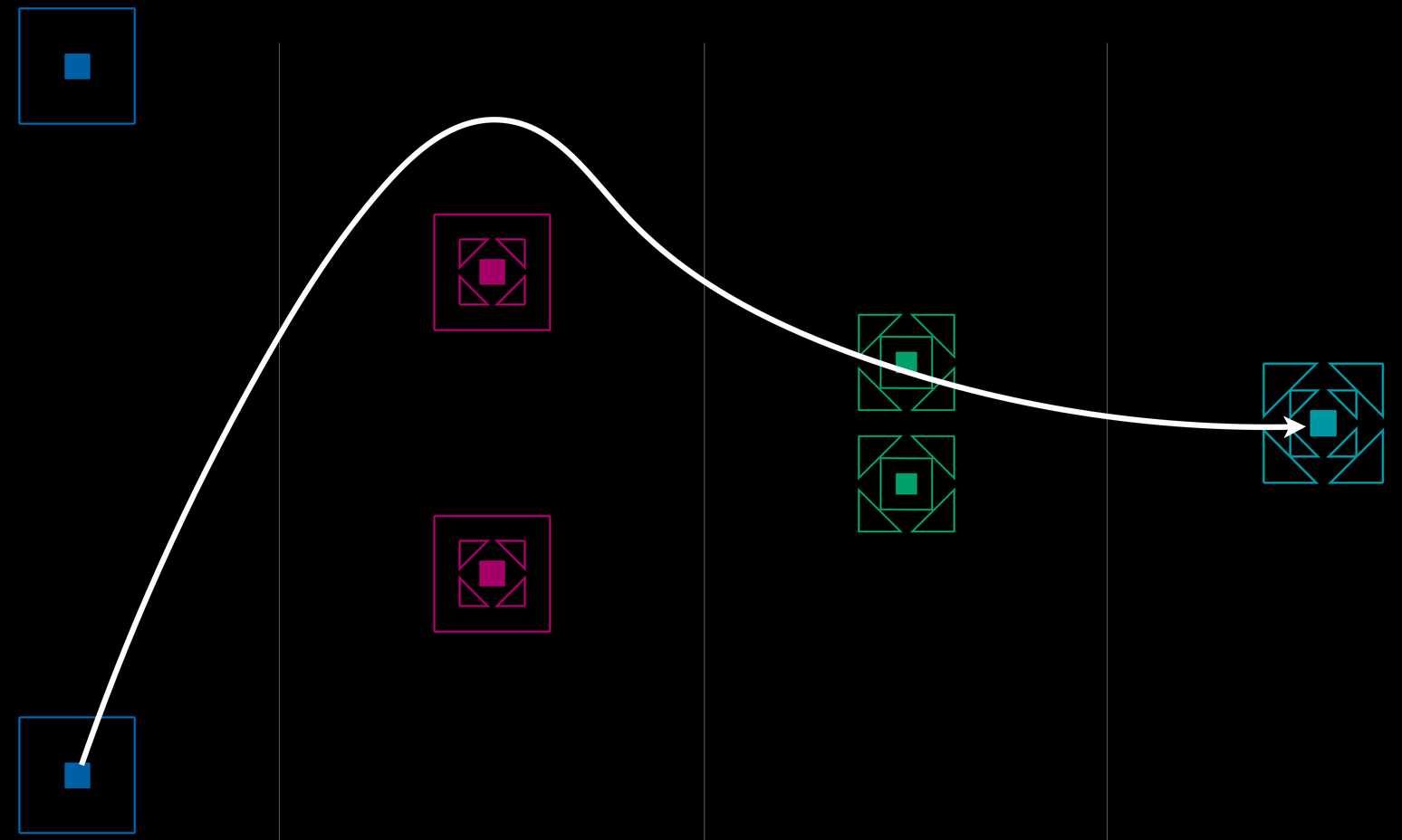
Stage 1

Stage 2

Stage 3

Stage 4

Top Down



Grassroots

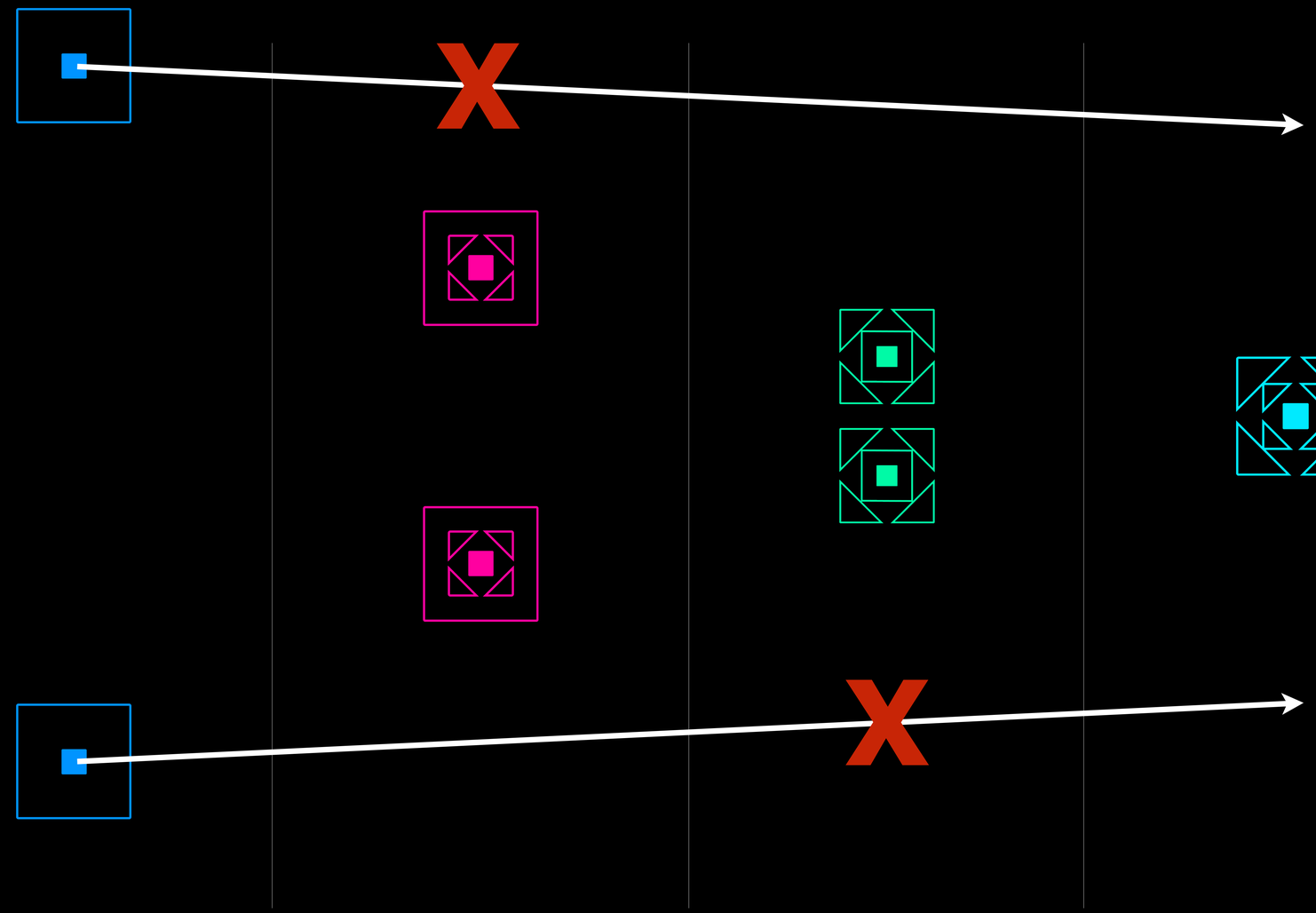
Stage 1

Stage 2

Stage 3

Stage 4

Top Down



Grassroots

Stage 1 Stage 2 Stage 3 Stage 4

# Identify your origin story and stage

| | Stage 1<br>Building Version One | Stage 2<br>Growing Adoption | Stage 3<br>Surviving the Teenage Years | Stage 4<br>Evolving a Healthy Program |
|---|---|---|---|---|
| Top Down | | | | |
| Grassroots | | | | |

# What is next for you to mature?

# How to Mature

# How to Mature

## Educate

- One-way interaction
- Create a shared understanding of what a design system is
- Cast the vision for why a design system is needed in your organization

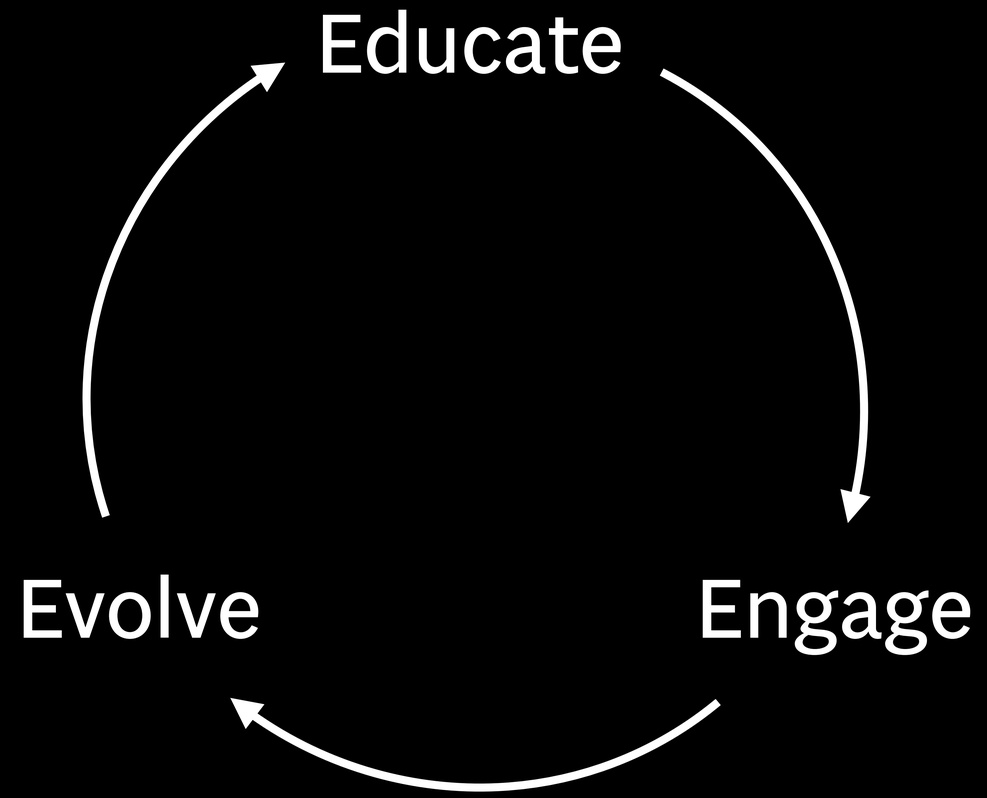
## Engage

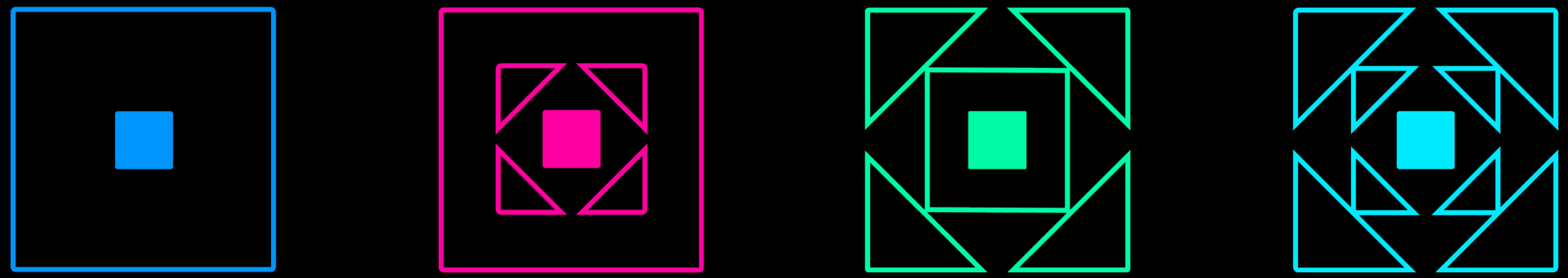
- Two-way interaction
- Learn from your users
- Give them a say in the way your system grows
- Recruit adoptors and contributors

## Evolve

- Team effort
- Make the system better
- Iterate and test your design system assets, documentation, and processes

# How to Mature





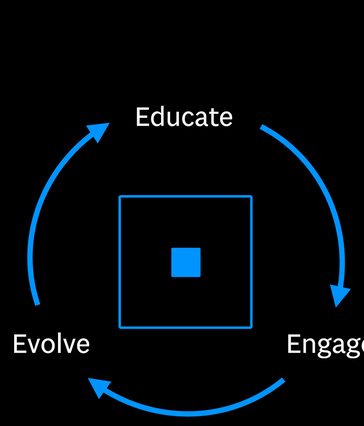
Stage 1 → Stage 2 → Stage 3 → Stage 4

Building Version One

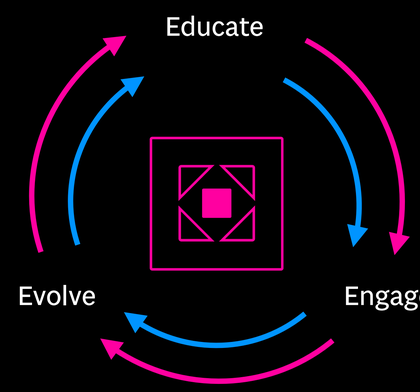
Growing Adoption

Surviving the Teenage Years

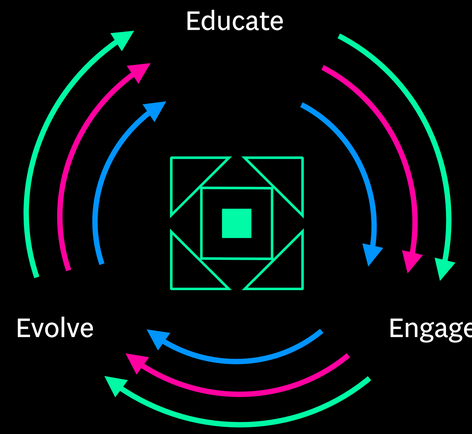
Evolving a Healthy Program



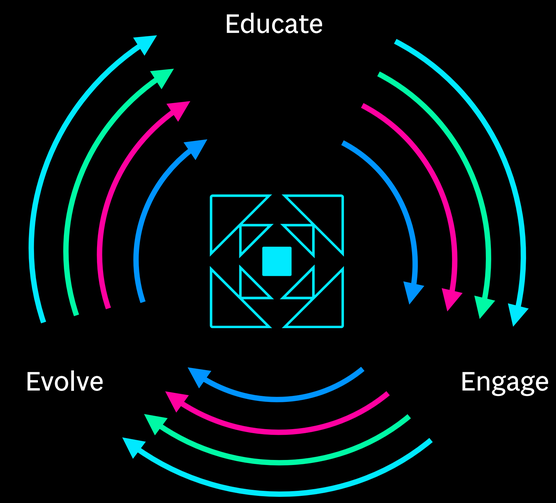

Stage 1




Stage 2




Stage 3




Stage 4

## Stage 1: Building Version One

### Educate

- Define design systems in the context of your organization
- Share benefits, costs, approach
- Cast the vision

### Engage

- Discover what will make your first version desirable
- Identify early adopters who may help test

### Evolve

- Iterate toward a first official release
- Focus on the fundamental layers
- Build doc site
- Write usage documentation for each discipline

## Stage 2: Growing Adoption

### Educate

- Demo the system and each release
- Share adoption stories
- Share the roadmap
- Share adoption goals and metrics

### Engage

- Gather feedback
- Collaborate on roadmap
- Pair with new subscribers
- Support existing subscribers
- Ask for support

### Evolve

- Create “How to Adopt” documentation
- Strategically add valuable features and documentation
- Strategically fix issues
- Automate capturing “adoption” metrics

## Stage 3: Surviving the Teenage Years

### Educate

- Capture and report on appropriate metrics
- More sophisticated communication strategy
- Share refined docs for system use
- Define and explain governance and contribution process

### Engage

- Ambassador programs
- Host subscriber teams
- Assist in onboarding new hires
- Accept contributions
- Solidify long-term leadership support

### Evolve

- Better define the processes around the system
- Continue to add documentation, and assets
- Improve everything!

# Stage 4: Evolving a Healthy Program

## Educate

- Explain the design system's role in new initiatives
- Demonstrate how the system is making product work better

## Engage

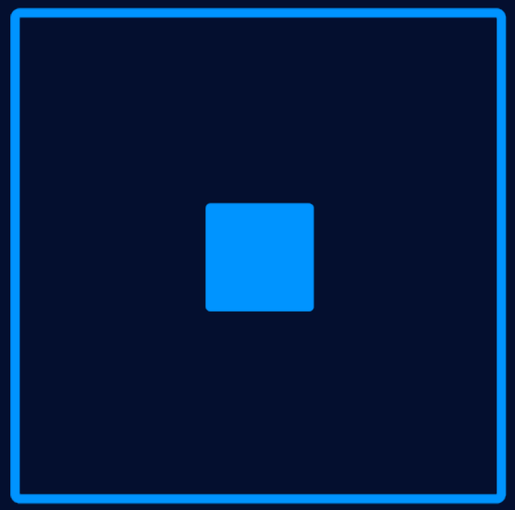
- Continuously interact with subscribers to understand new needs
- Actively participate in big picture convos with leadership

## Evolve

- Ensure the system is the best way to design and build digital interfaces
- Proactively add new assets as industry tooling shifts

# Break your next steps into the 3 E's

| | Education | Engagement | Evolution |
|---|---|---|---|
| Task 1 | | | |
| Task 2 | | | |
| Task 3 | | | |



# Stage 1

## Building Version One

# Discover the right version one.

Stage 1: Building Version One

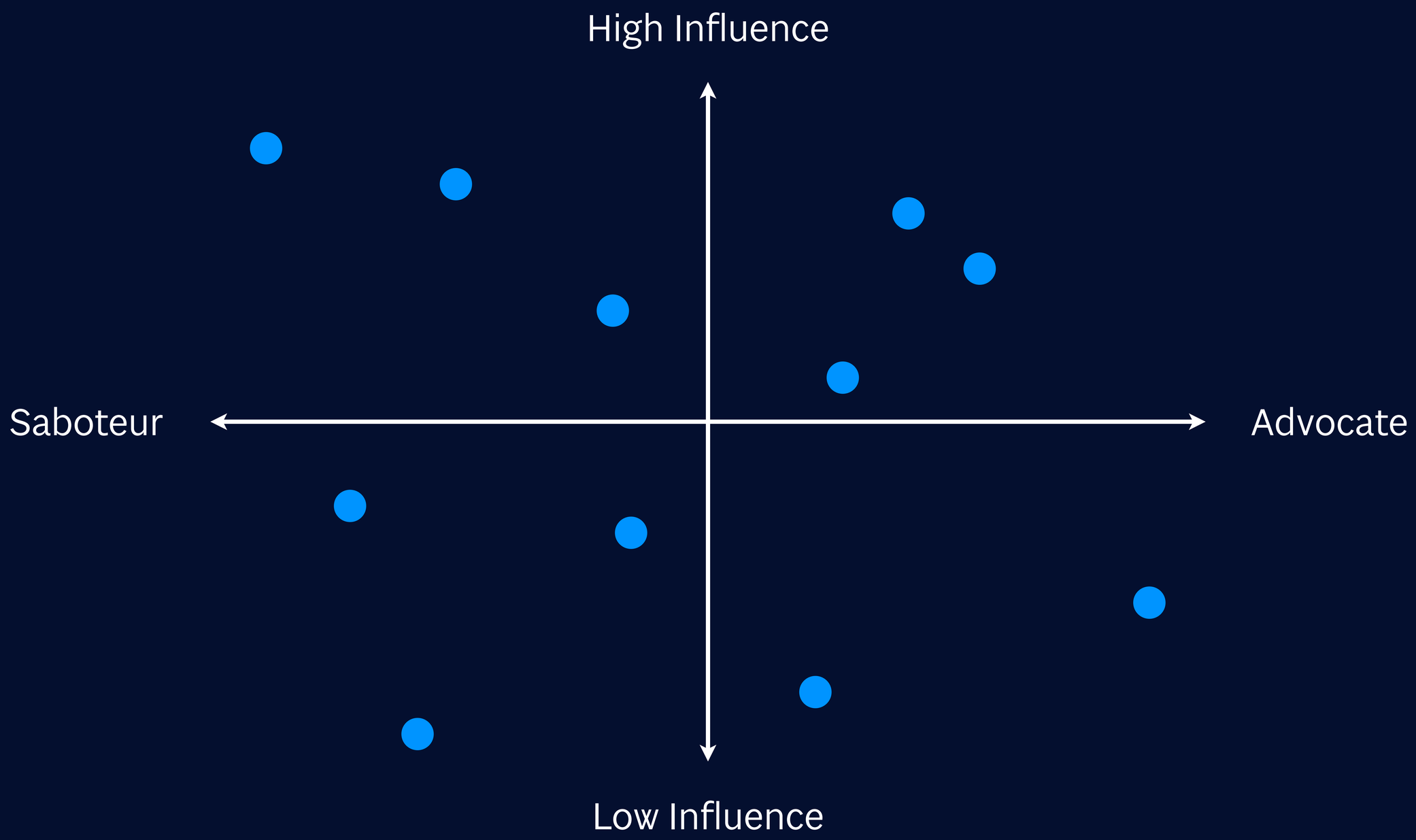
# Top Challenge

Find a balance between releasing quickly and showing value

# How to Succeed

Shape the project to show value to the right potential subscribers.

# Know your audience.


High Influence
Saboteur
Advocate
Low Influence

# Imagine one year from today...

Stage 1: Building Version One

# Define Subscriber Groups

## Example 1, Product or Feature Driven Groups

Group 1: E-commerce Web

Group 2: iOS Shopping App

Group 3: Checkout Flow on E-commerce Web

Stage 1: Building Version One

# Define Subscriber Groups

## Example 2, Discipline Driven Groups

Group 1: UI Designers

Group 2: UI Devs

Group 3: Product Owners

Stage 1: Building Version One

# Define Subscriber Groups

## Example 3, Discipline by Product Driven

Group 1: UI Designers working on E-commerce Web

Group 2: UI Devs working on Checkout Flow of E-Commerce Web

Group 3: UX Designers working on iOS Shopping App

Stage 1: Building Version One

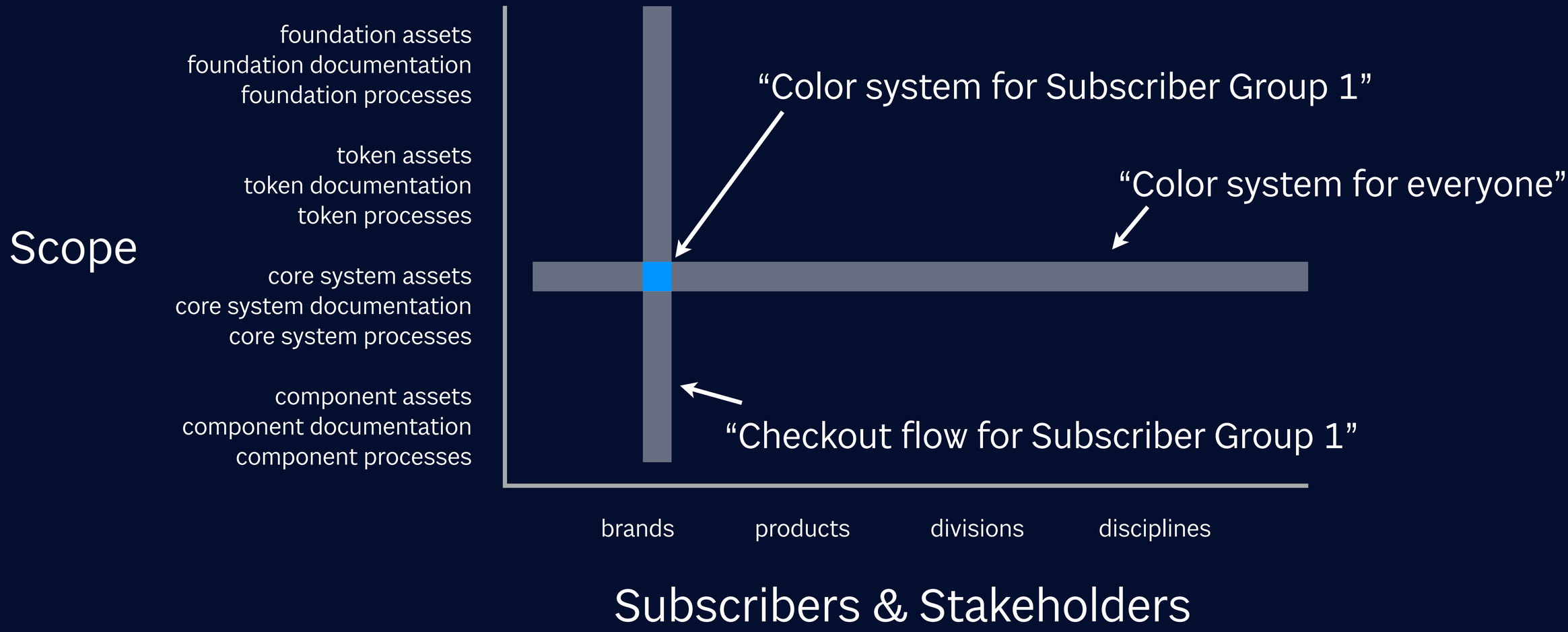
# Understand Subscriber Metrics

Learn how they measure their own success

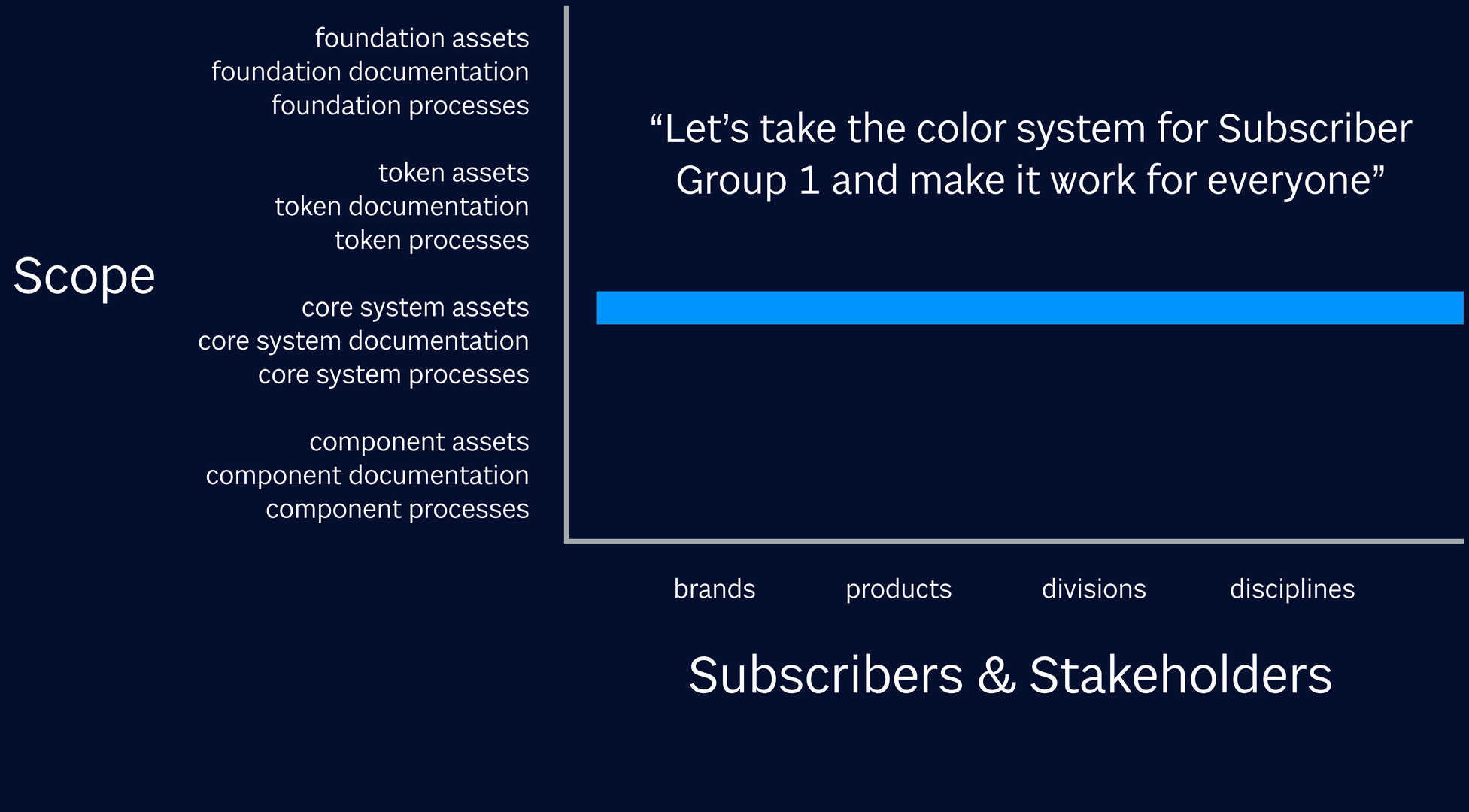
Capture baselines before they adopt the system

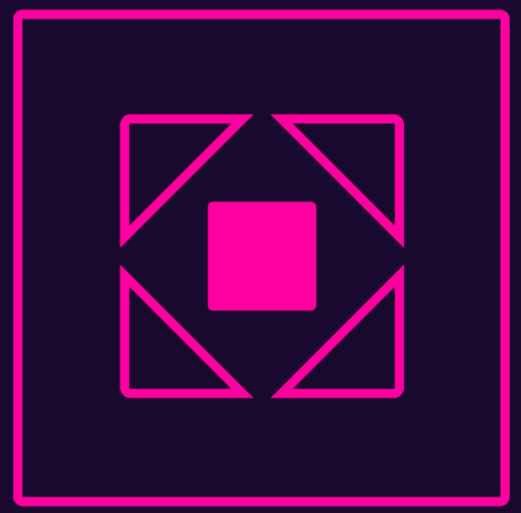
# Discover the right version one.

# Stage 1: Building Version One



# Stage 1: Building Version One





# Stage 2

## Growing Adoption

# Build the system people want to use.

Stage 2: Growing Adoption

# Top Challenge

Evolve the system to show value to potential subscribers and stakeholders

# Stage 2: Growing Adoption

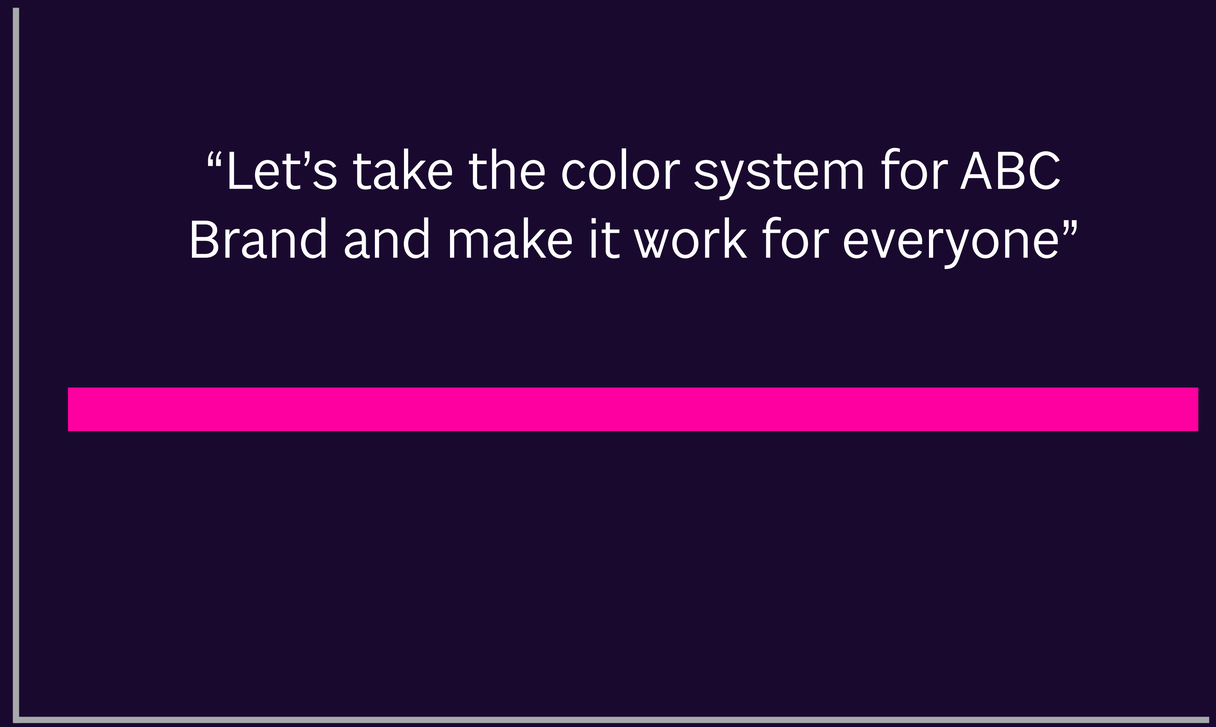
foundation assets
foundation documentation
foundation processes

token assets
token documentation
token processes

Scope

core system assets
core system documentation
core system processes

component assets
component documentation
component processes



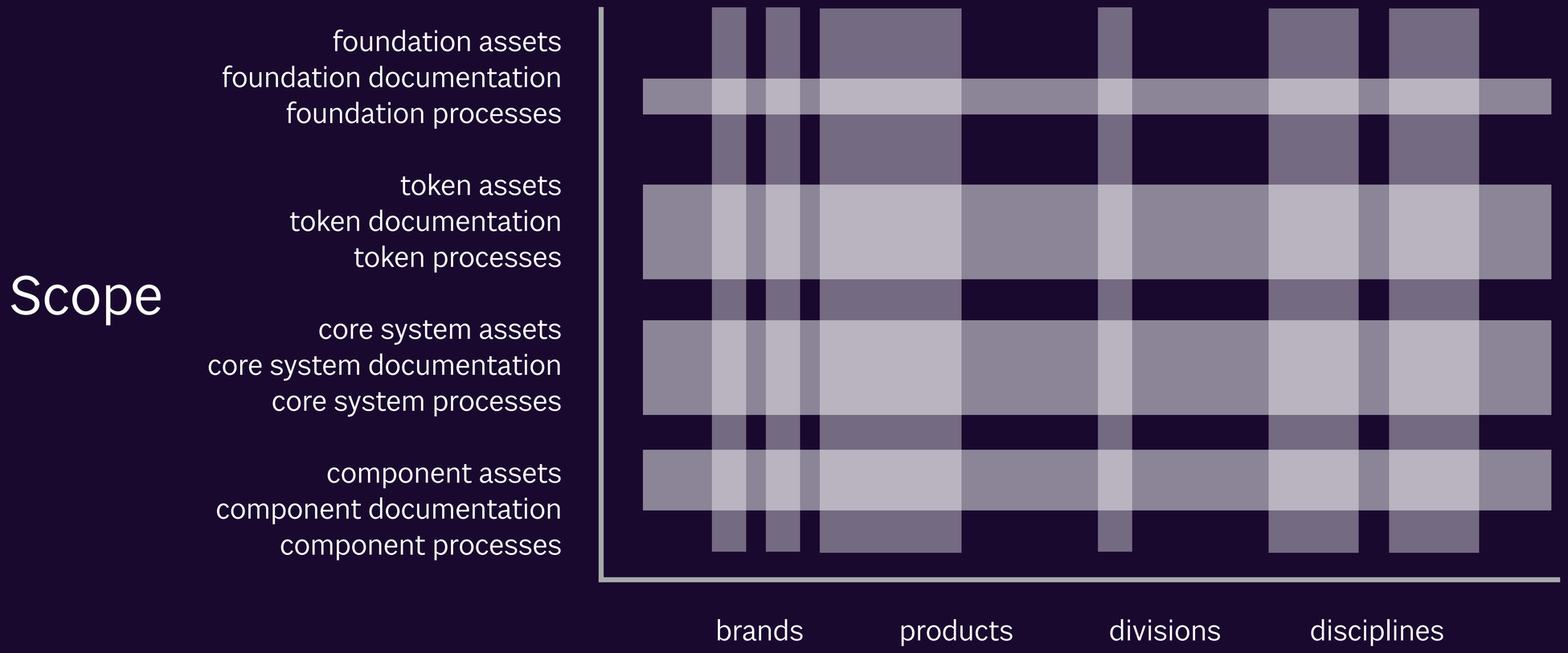
"Let's take the color system for ABC Brand and make it work for everyone"

brands products divisions disciplines

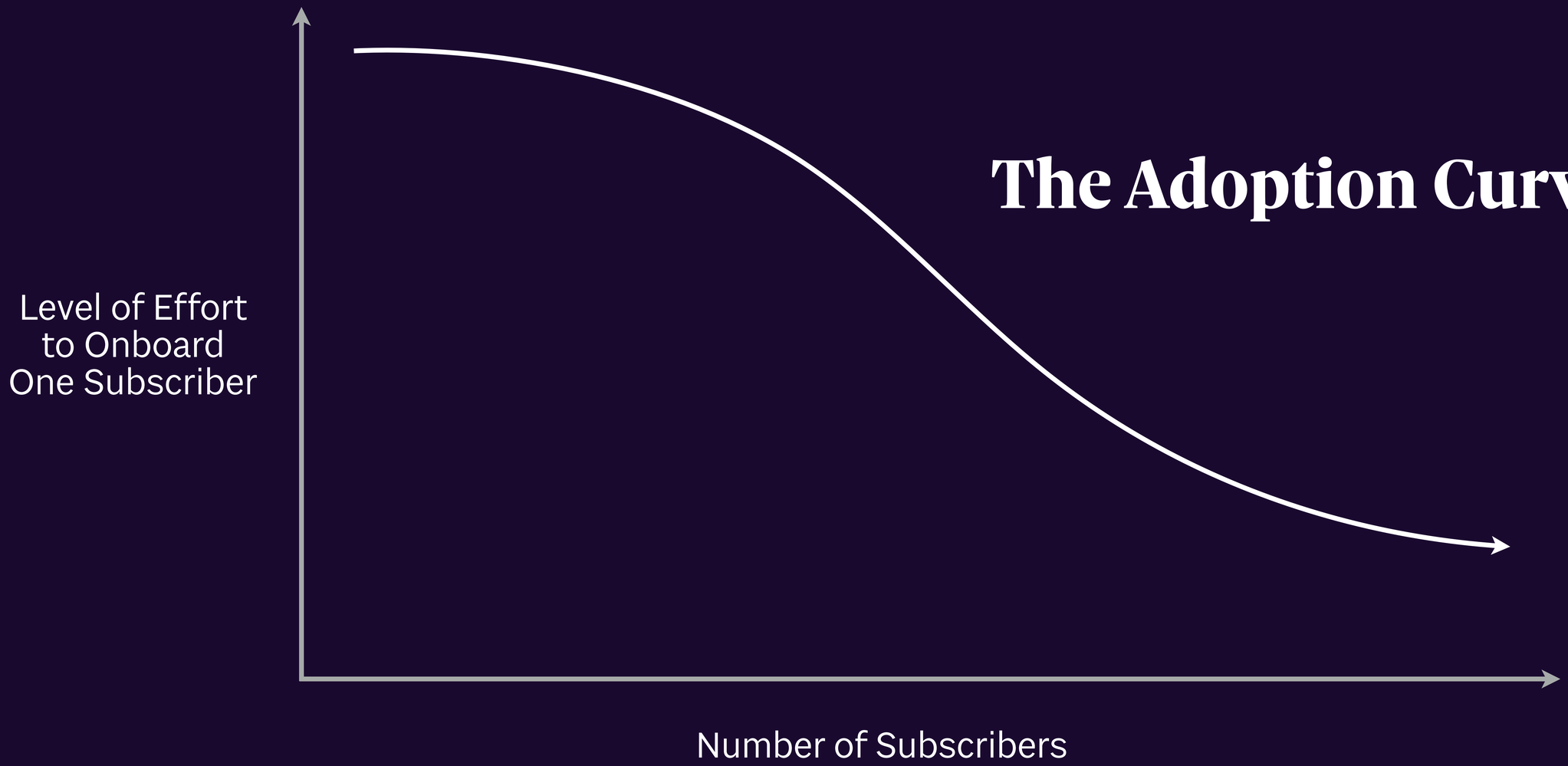
Subscribers & Stakeholders

# Stage 2: Growing Adoption

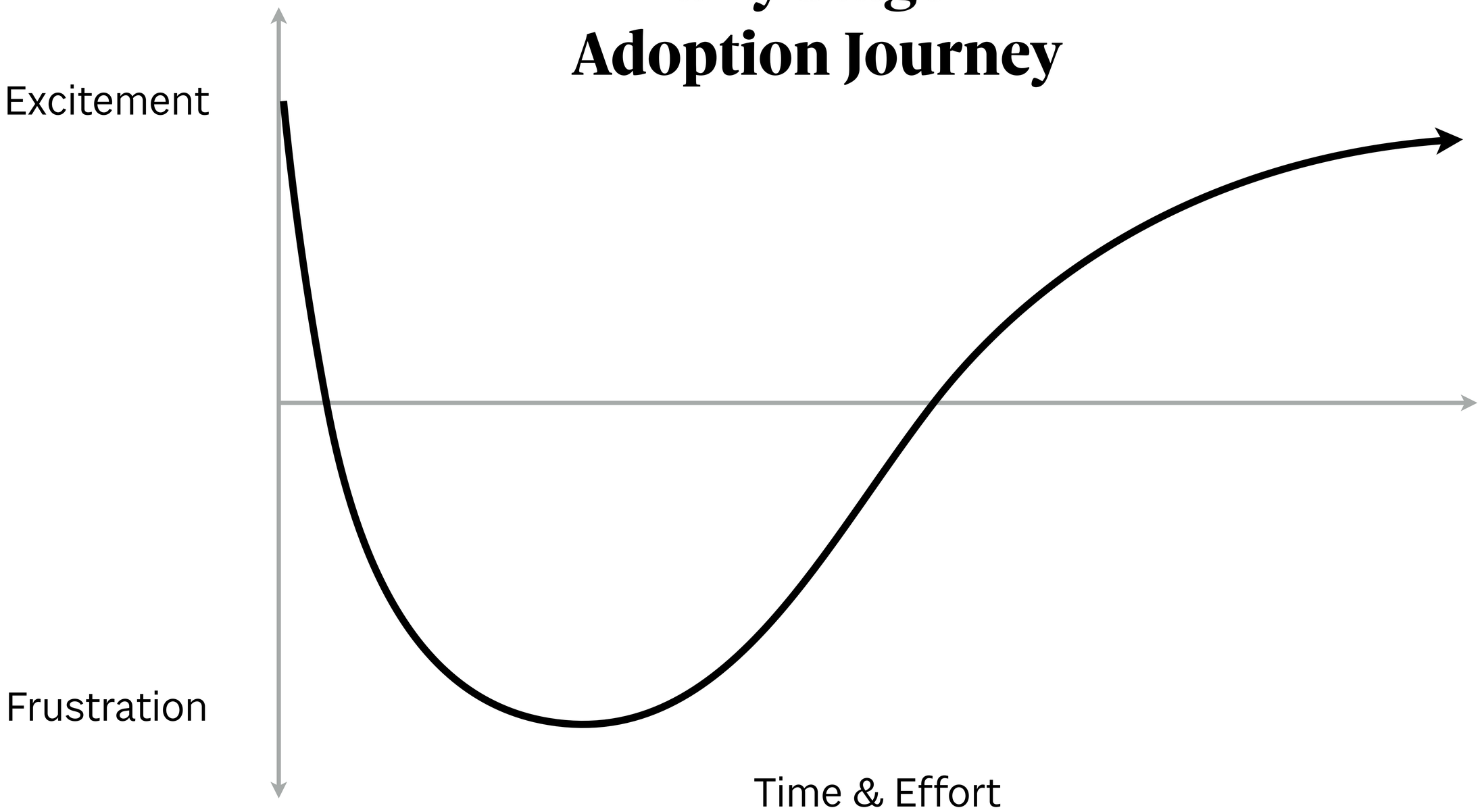
# Stage 2: Growing Adoption



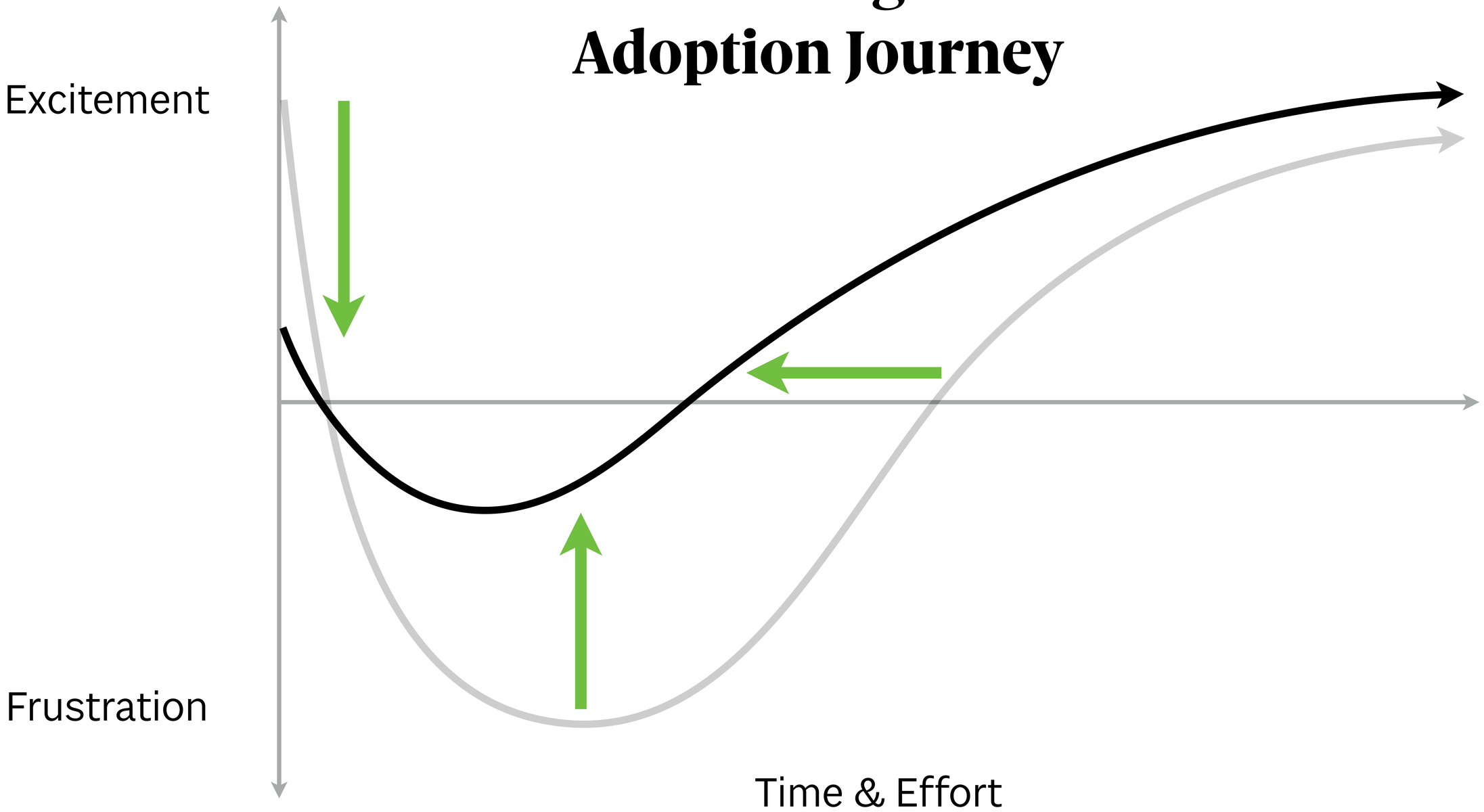
Stage 2: Growing Adoption



# Early Stage 2
# Adoption Journey



# Late Stage 2
# Adoption Journey



Stage 2: Growing Adoption

# Define the Adoptable System Parts

If you’ve used The Anatomy of a Design System model, the intersection of a layer and a part is adoptable.

Stage 2: Growing Adoption

# Define what Adoption Means

This seems easy, but it’s not

## Stage 2: Growing Adoption

One of your subscribers uses the grid and spacing core system from your design system to handle layout in their product's production code, but they don't use your color or elevation systems.

## Stage 2: Growing Adoption

One of your subscriber teams uses everything your design system offers, but only in the sign-up flow of their product.

## Stage 2: Growing Adoption

The UX folks on one of your product teams use the design system to prototype and test, but the developers on that same product team aren’t using the engineering assets you provide.

## Stage 2: Growing Adoption

One of your subscriber teams uses a version of the design system that is two releases old.

## Stage 2: Growing Adoption

The devs on one of your product teams starts with your design system components, but modifies them to fit a specific use case.

Stage 2: Growing Adoption

# Define an Adoption Path

Give your potential subscribers a path to follow that helps them adopt in a healthy way.

## Stage 2: Growing Adoption

Level 1: Understand and apply foundations layer

Level 2: Use Core Systems assets for layout, type, and color

Level 3: Use all parts of the Fundamental Layers

Level 4: Use all parts of all Components relevant to the product

Level 5: Contribute feedback, documentation, content, UX, design, code to the system using our contribution processes

Stage 2: Growing Adoption

# Define Health Metrics

Articulate health scores for the metrics that are important to you.

Stage 2: Growing Adoption

# Health: System Version

Grade A: Current Version

Grade B: Current Version -1

Grade C: Current Version -2 or less

Stage 2: Growing Adoption

# Health: Product Coverage

Grade A: 75% + of the Product uses the system

Grade B: 50%–74% of the Product uses the system

Grade C: 25%–49% of the Product uses the system

Grade D: 0%–25% of the Product uses the system

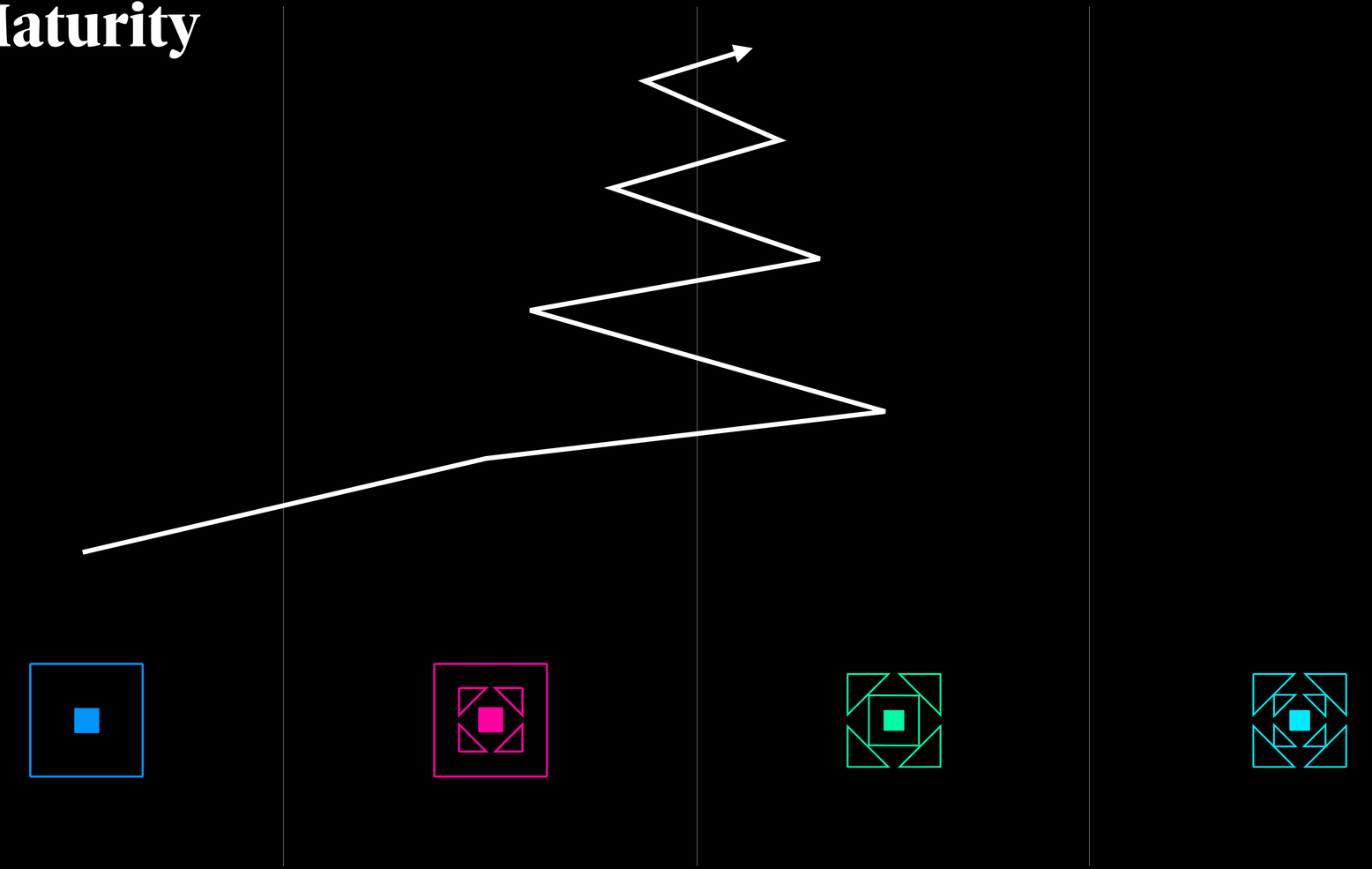
Stage 2: Growing Adoption

# Maintain Adoption Levels

Maintaining adoption is about maintaining trust with your subscribers.

Communication, consistency, transparency.

# Teeter Totter Maturity

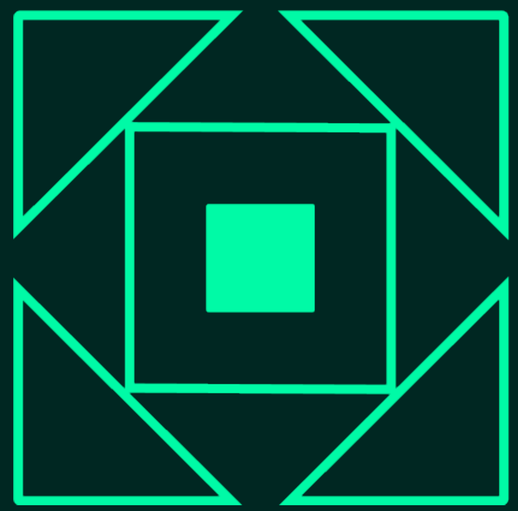


**Stage 1**

**Stage 2**

**Stage 3**

**Stage 4**



# Stage 3

## Surviving the Teenage Years

# Keep your promises.

Stage 3: Surviving the Teenage Years

# Top Challenge

Prove the system does what you said it would do.

Stage 3: Surviving the Teenage Years

# Design System Metrics

Correlate promised benefits with increased adoption of the system.

## As _____ changes

Usage of the design system documentation site

Usage of specific design system assets across disciplines

Usage of specific versions of the design system

Percentage of products with Level 1 Grade A adoption

Percentage of UI designers using design system assets

Percentage of UX designers using design system assets

Percentage of QA engineers using design system assets

Percentage of UI developers using design system assets

Percentage of production code generated from the system

Engagement of subscriber teams

Modification of assets from their defaults

## We see _____ improve

Speed to market with new features

Efficiency of testing new ideas

Speed of onboarding new employees

Quality of digital products

Age of bug or feature requests in products

Velocity of product or feature teams

Usability of products built with the system

Conversion of products built with the system

Accessibility of products built with the system

Carbon footprint of products built with the system

Product or feature team member happiness

Stage 3: Surviving the Teenage Years

# Add Disciplines to Your Team

UI Designer

UI Developer

UX Designer

Subscriber Support

Metrics/Analytics

Quality Assurance

Content Lead

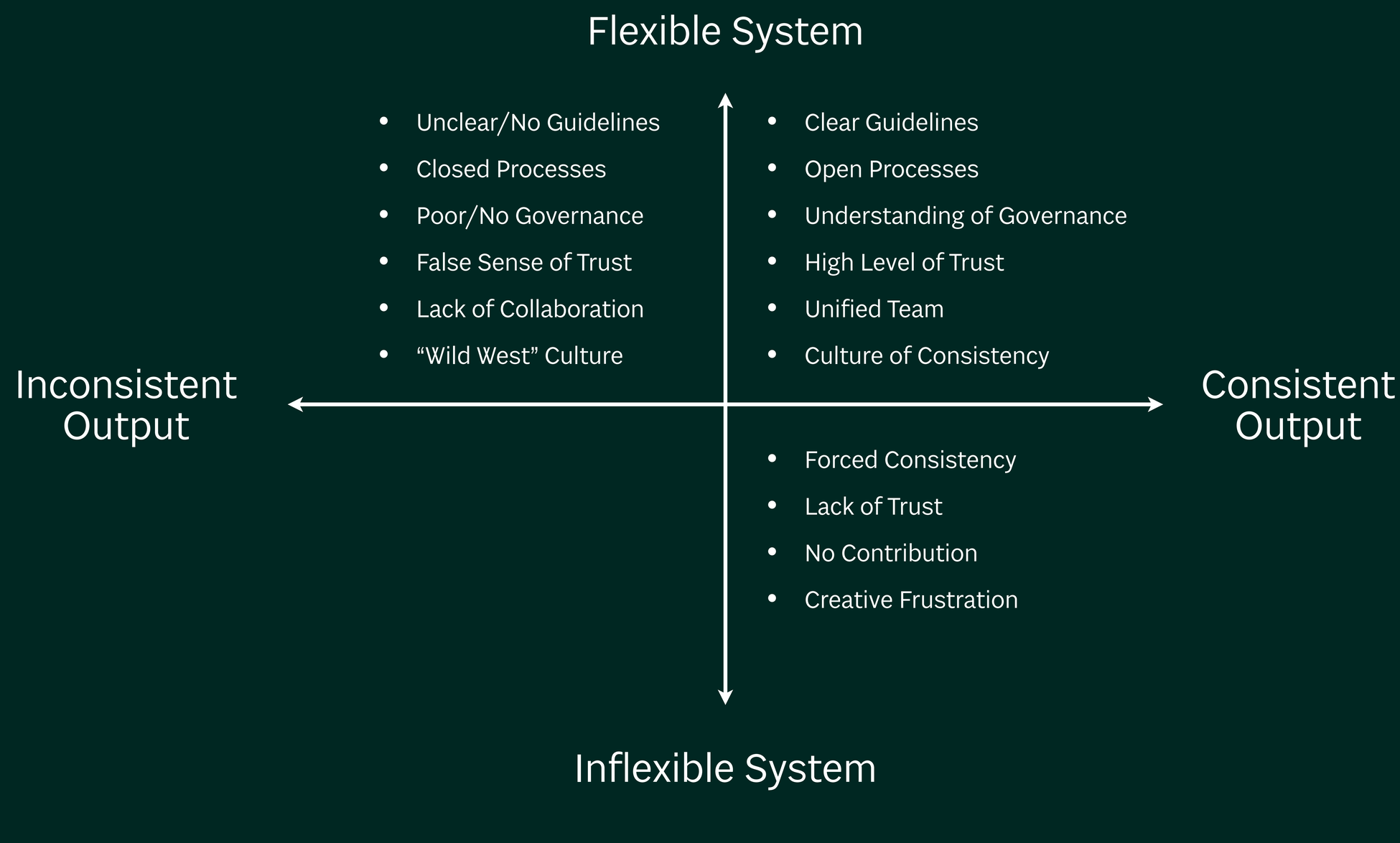
UX Writer

Product Owner

# Stage 3: Surviving the Teenage Years

**Consistency** ⟷ **Flexibility**

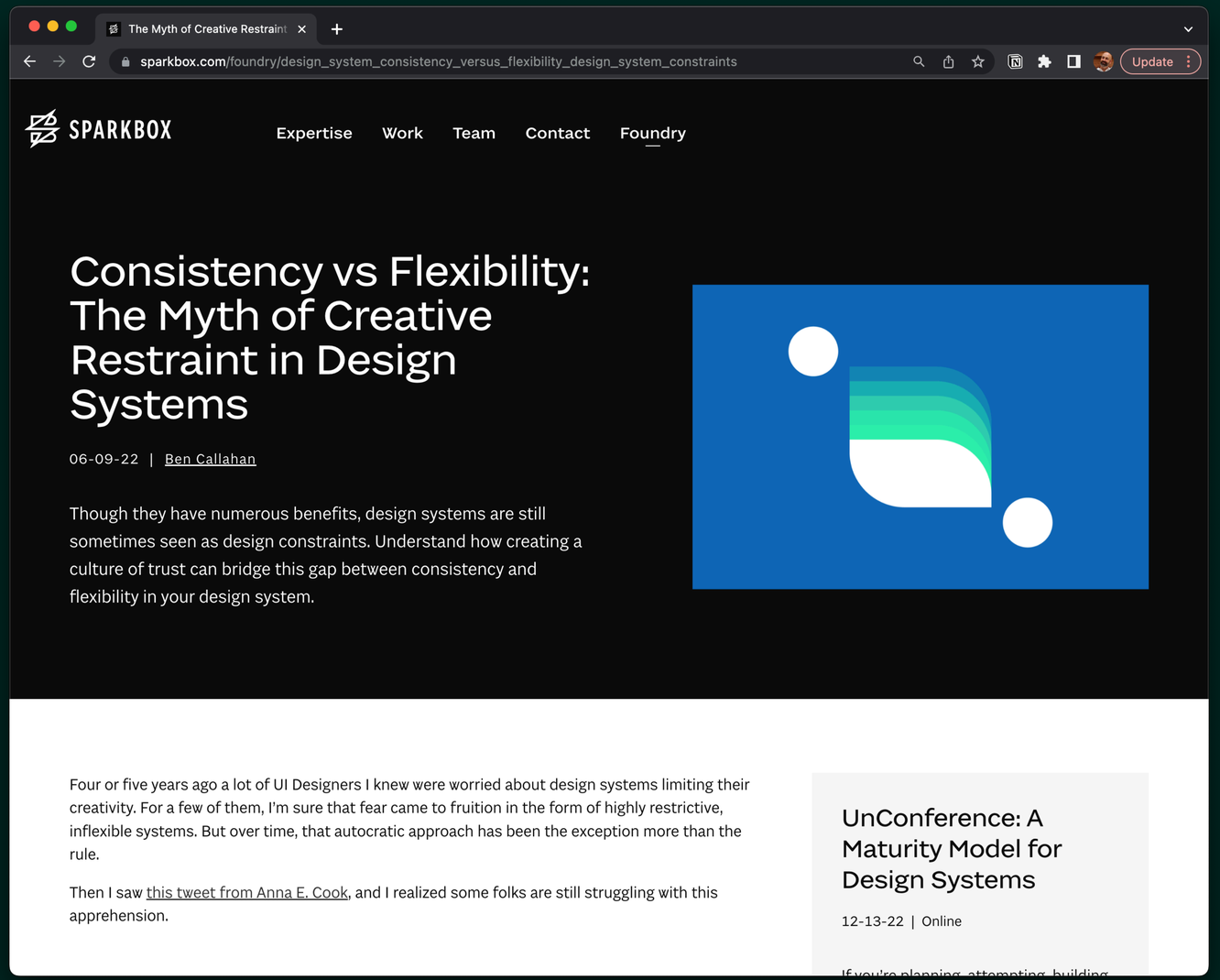
# Stage 3: Surviving the Teenage Years



Stage 3: Surviving the Teenage Years

# Flexibility & Consistency

Your system needs to enable creativity while your culture encourages restraint.




https://bit.ly/ds-flex

Stage 3: Surviving the Teenage Years

# Selling Design Systems

“Turning something from an idea
into a reality
can make it seem smaller.
It changes from unearthly to earthly.

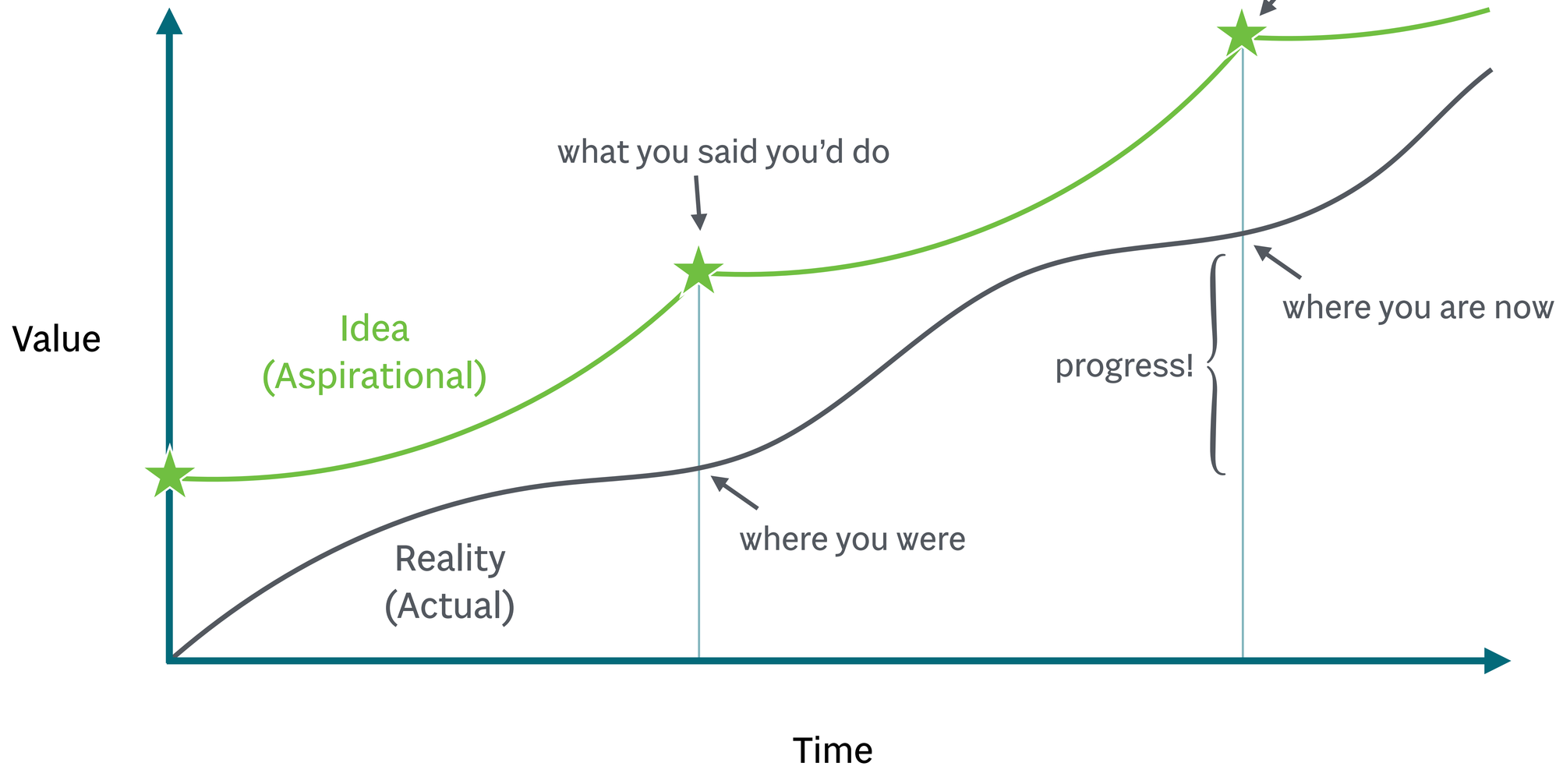
The imagination has no limits.
The physical world does.
The work exists in both.”

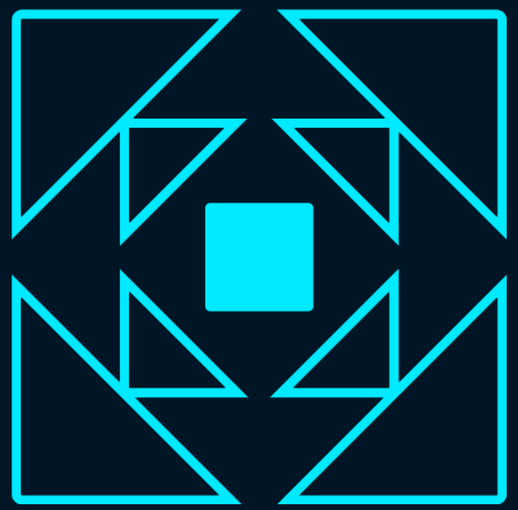
*~ Rick Rubin, from The Creative Act: A Way of Being*

SPARKBOX

# Selling a Design System

Idea vs Reality





# Stage 4

## Evolving a Healthy Program

Stage 4: Evolving a Healthy Program

# Consistency of Behavior

All the things you've been doing through the first three stages are precisely the behaviors needed to reach Stage 4.

# Consistency of Behavior

This consistent behavior means the system is trustworthy and predictable.

Stage 4: Evolving a Healthy Program

# The Proactive DS Team

The characteristic I see in Stage 4 teams that seems to be missing in Stage 3, is proactivity.

# Codify the Maturity

One risk: Stage 4 is reached because the DS team has some really talented people.

# System Stability

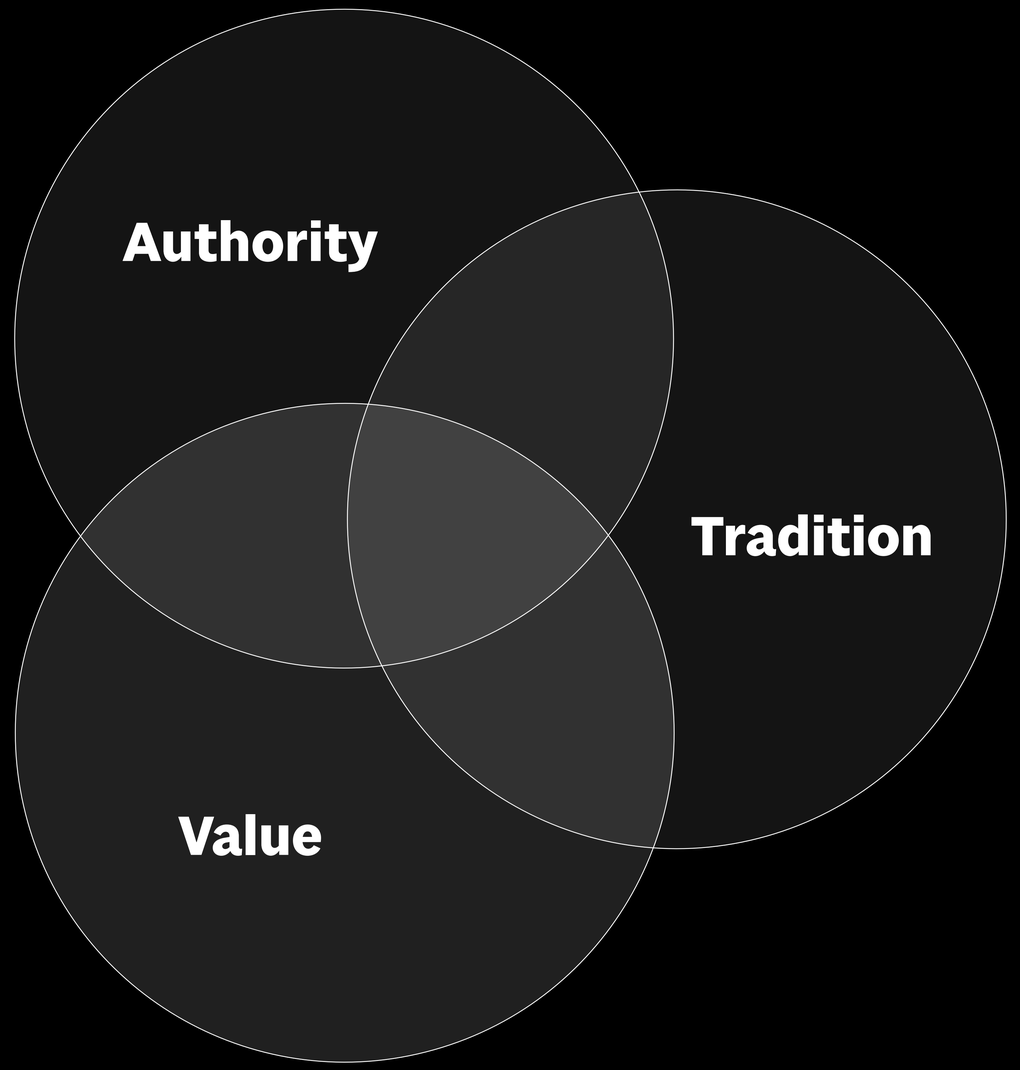
Everything is always changing.

System Stability

# Destabilizing Forces

- Changes in the market
- Changes in the direction of the product
- Changes in support for the product
- Changes in leadership
- Changes in the structure of the organization
- Changes in the design system team members
- Changes in the structure of the design system team
- Changes in the tooling supported
- Changes in the subscribers to the design system

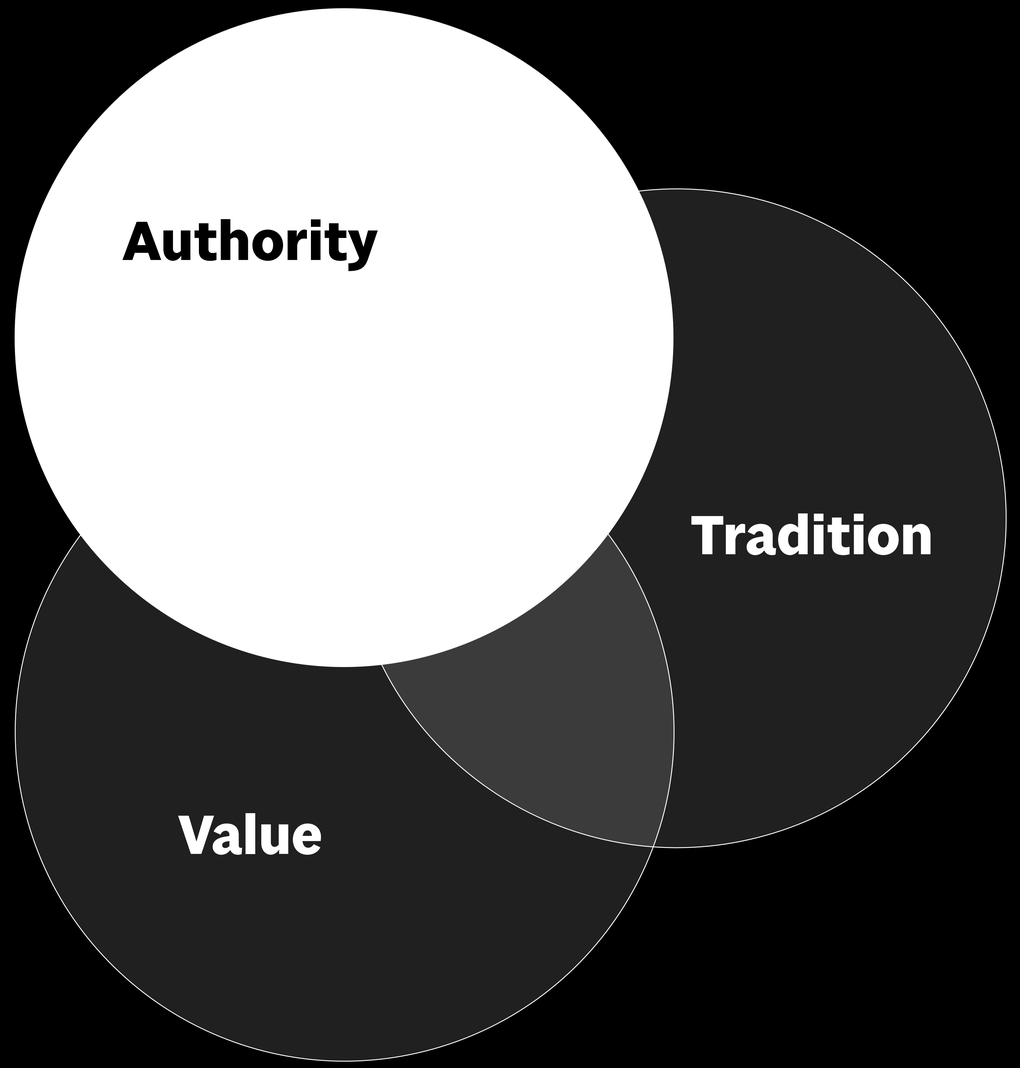
# System Stability



System Stability

# Authority

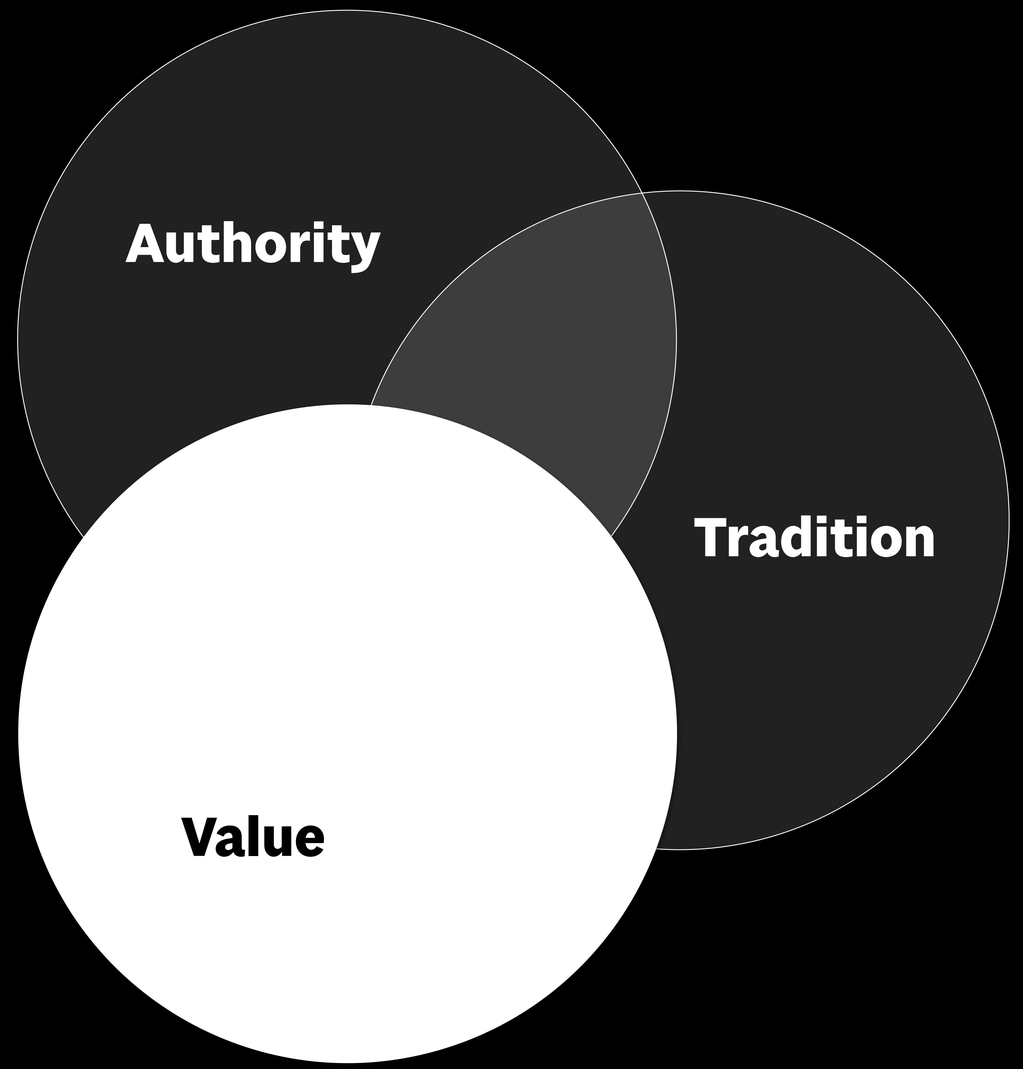
Leadership actively supports the design system team and their goals.



System Stability

# Value

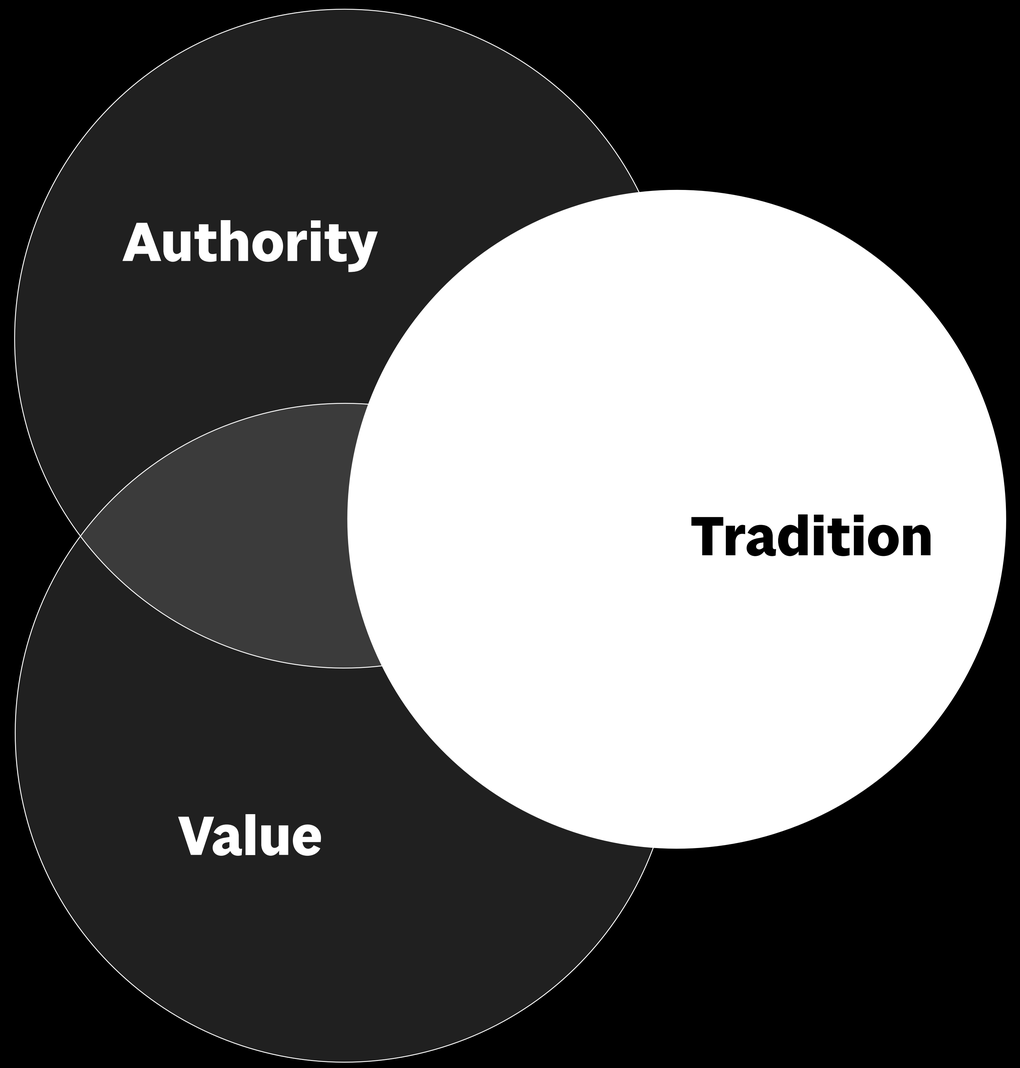
The system offers true value to those who use it.



System Stability

# Tradition

The system has become
“the way we build interfaces”
inside the organization.



System Stability

# Tradition

Authority

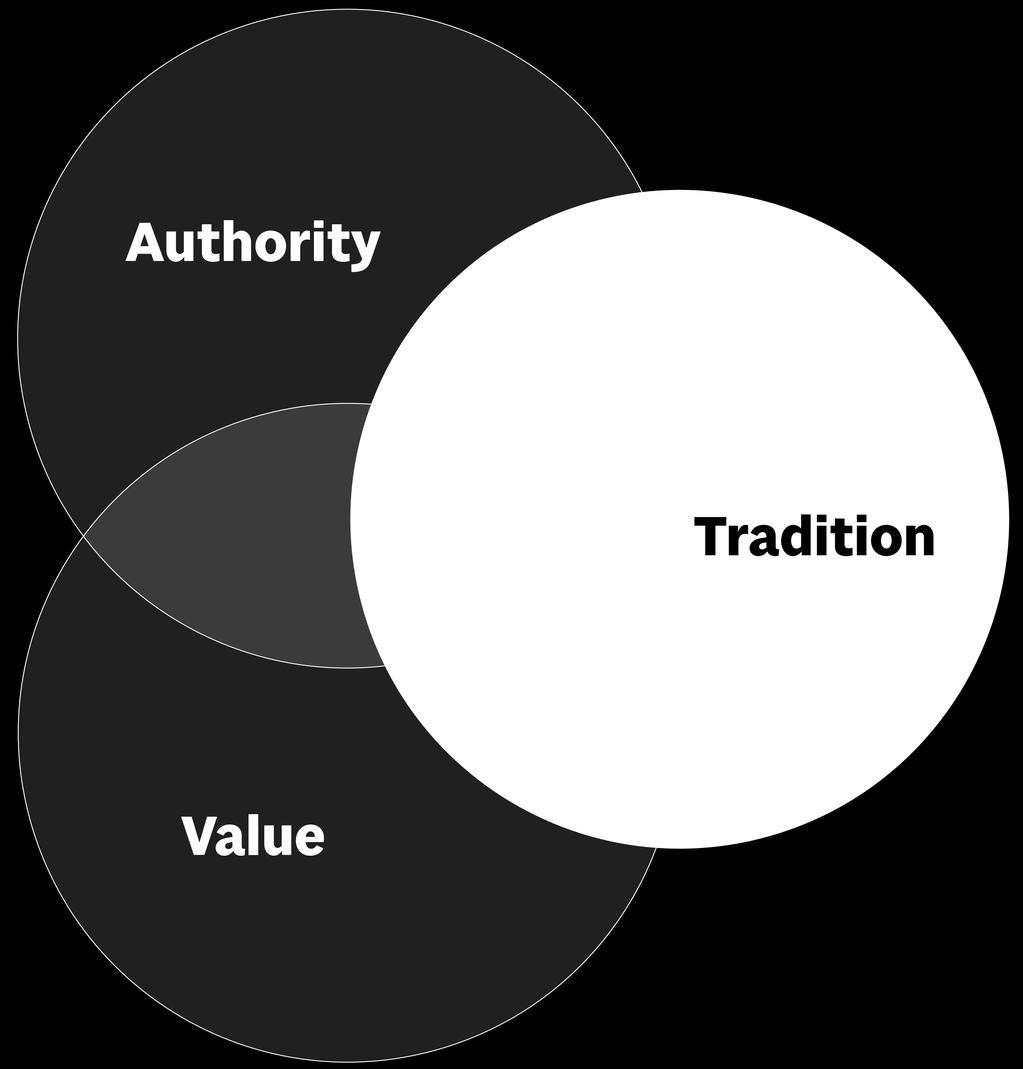
Value

\+

Time

---

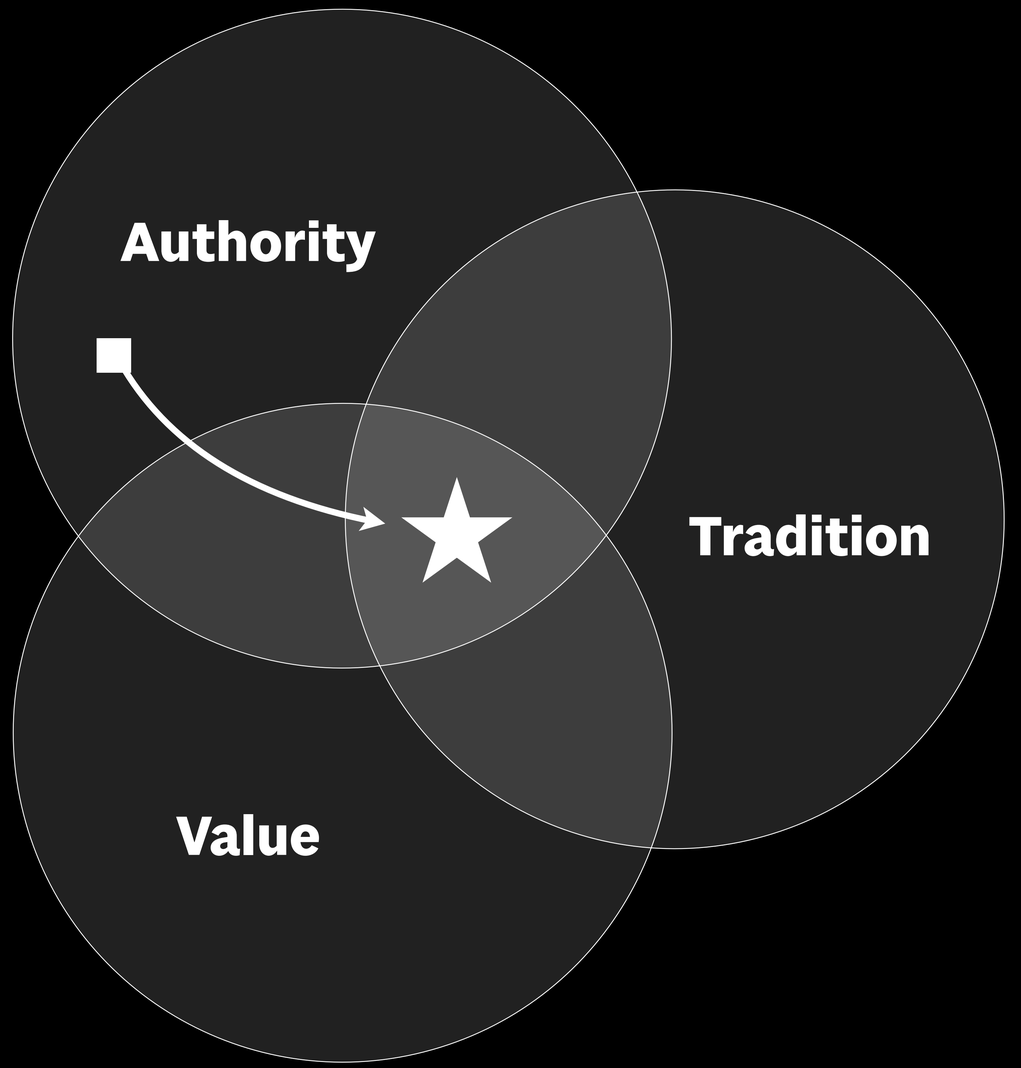
Tradition



System Stability

# Top Down

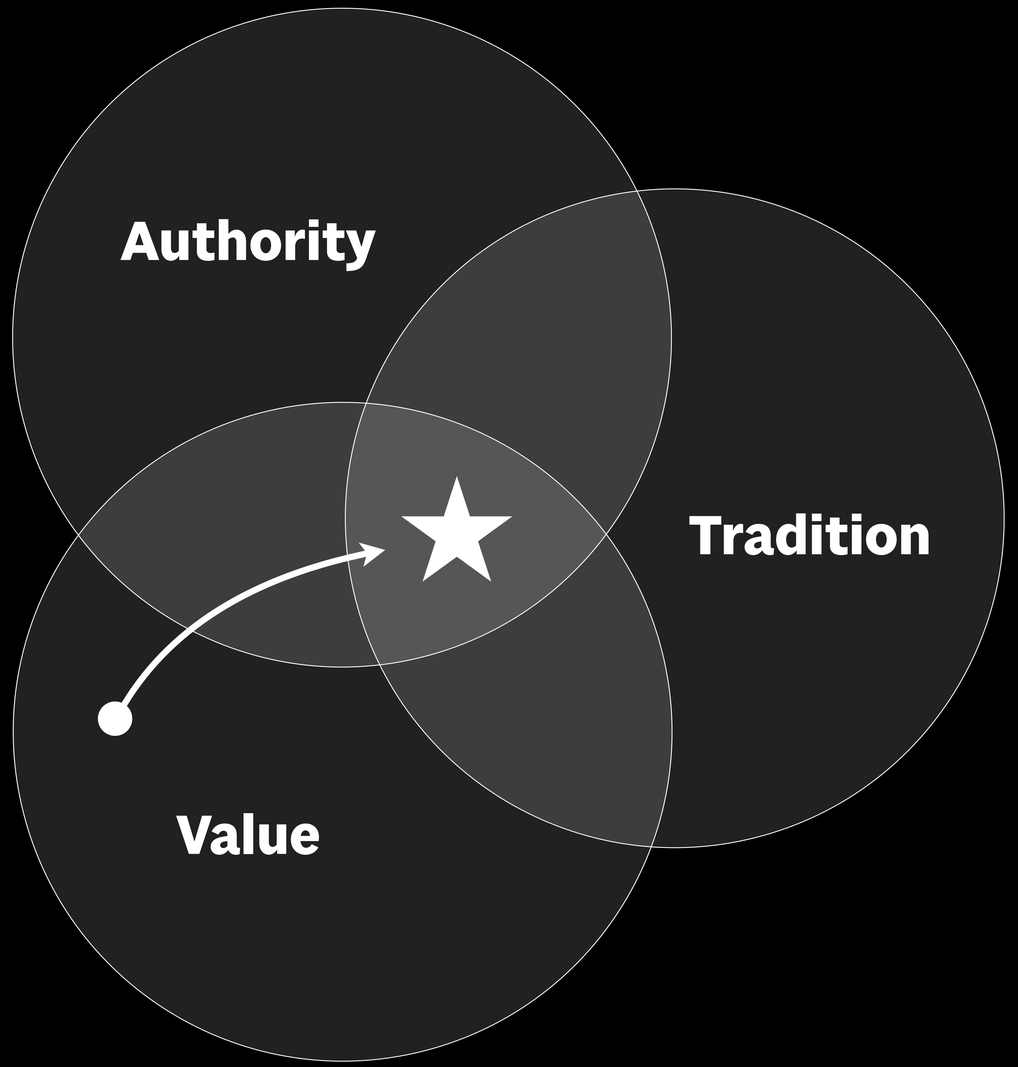
The system was initiated by leadership.



System Stability

# Grassroots

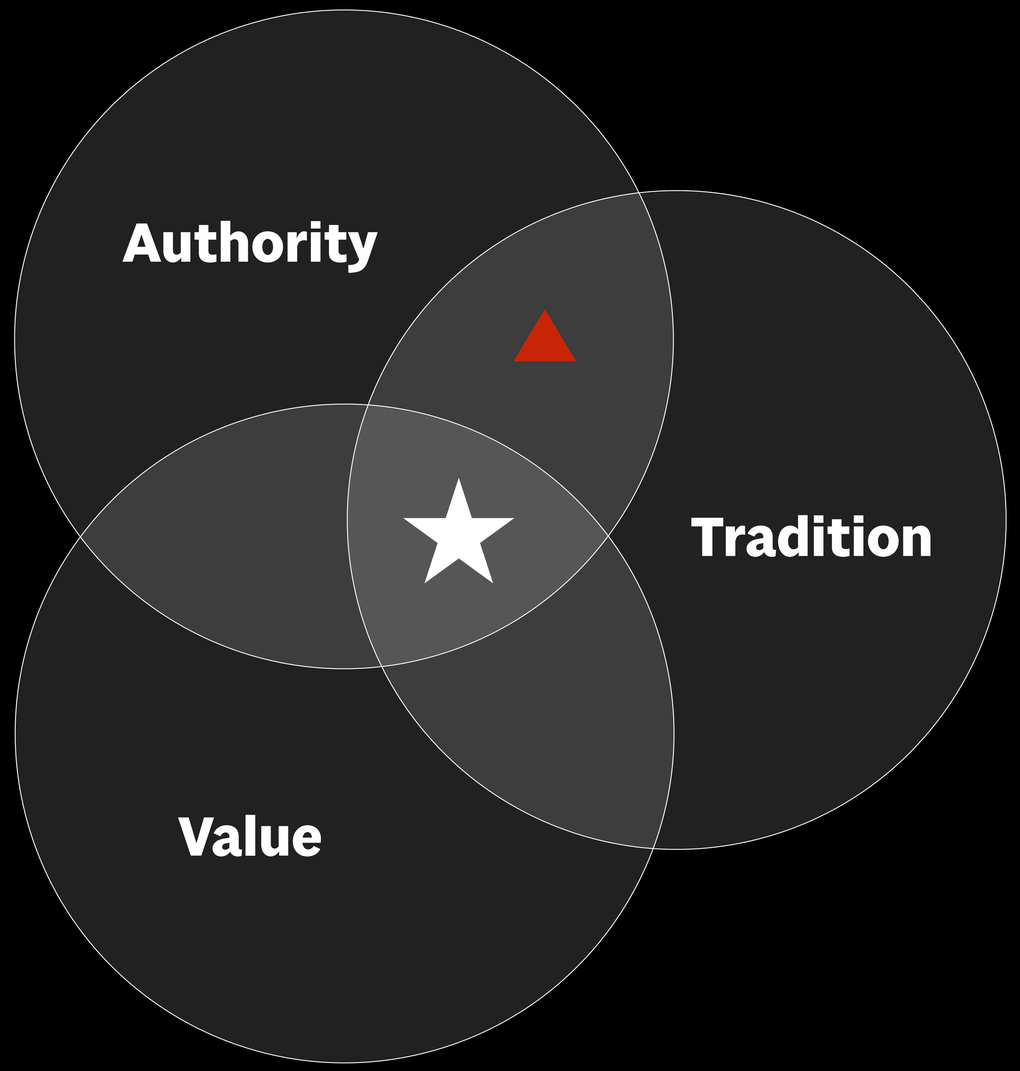
The system was initiated by individual contributors.



System Stability

# Regression

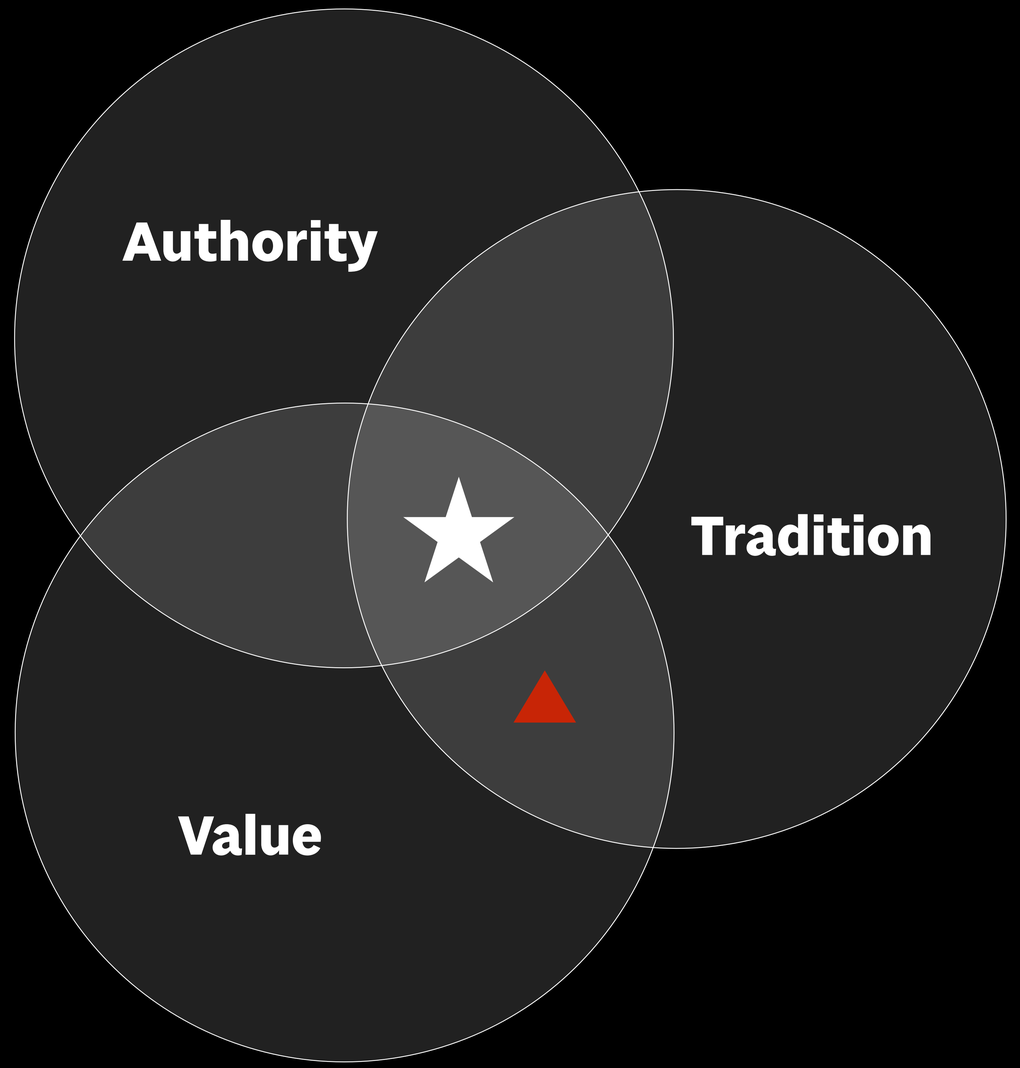
Once you've gained a stabilizer, you have to work to maintain it.



System Stability

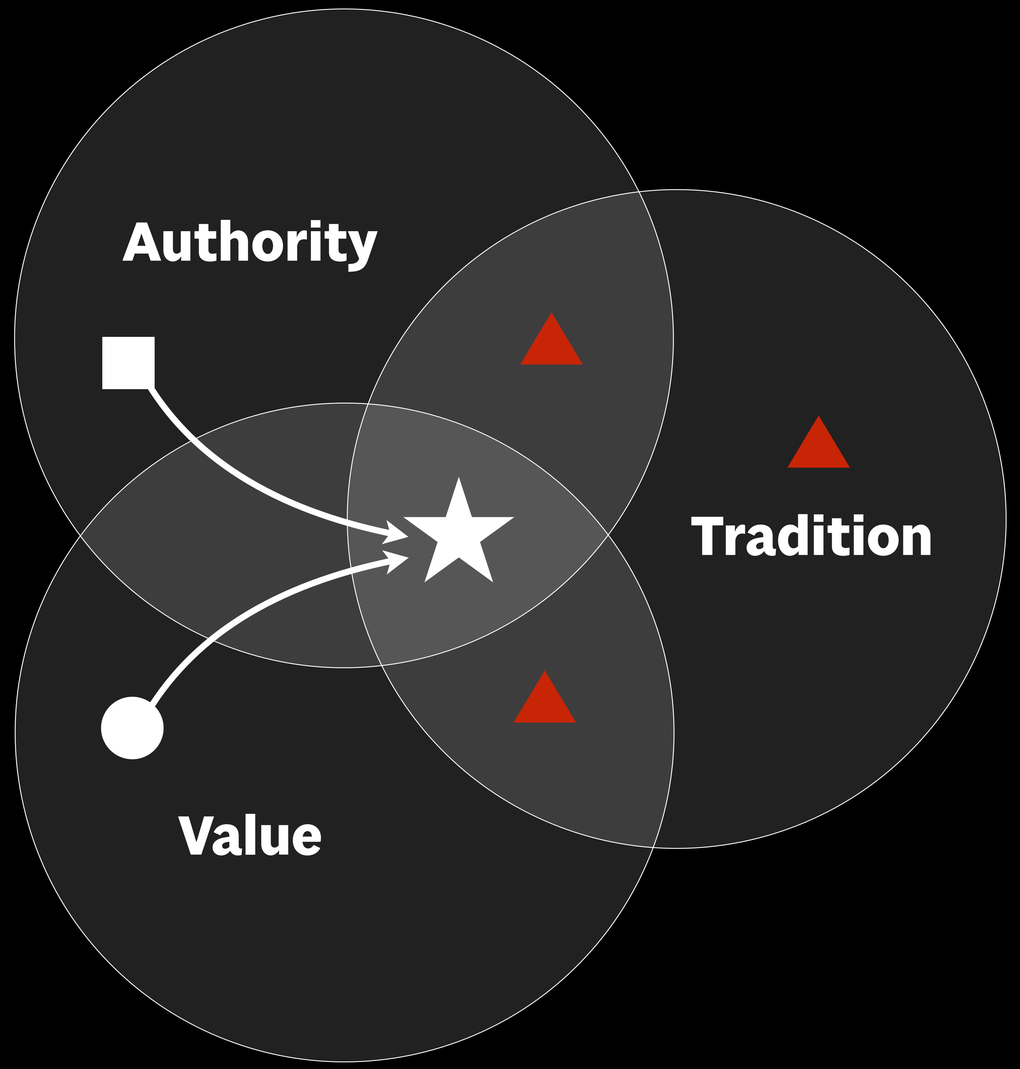
# Avoiding Authority

Selling to leadership is tough, but it’s part of the job.

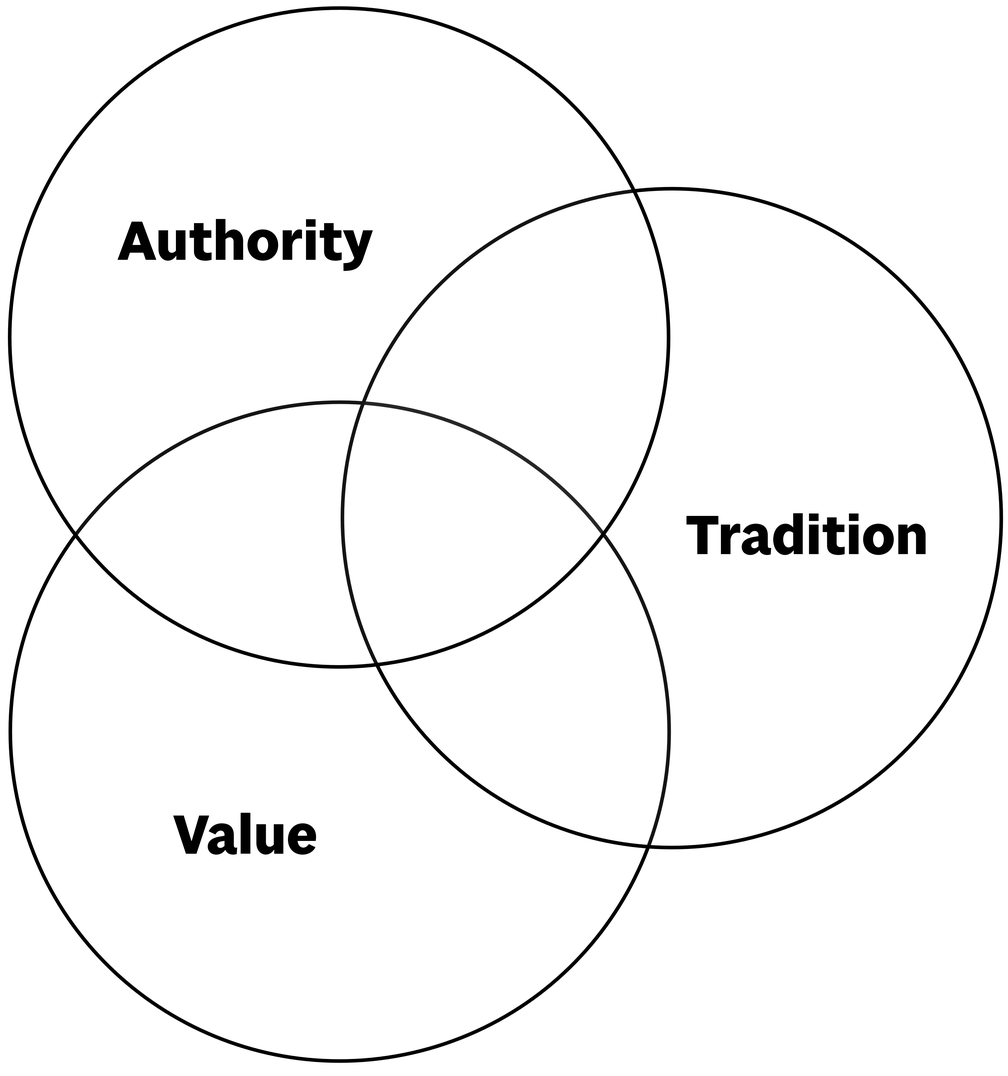


# System Stability

- Top down start
- Grassroots start
- Stable
- At risk



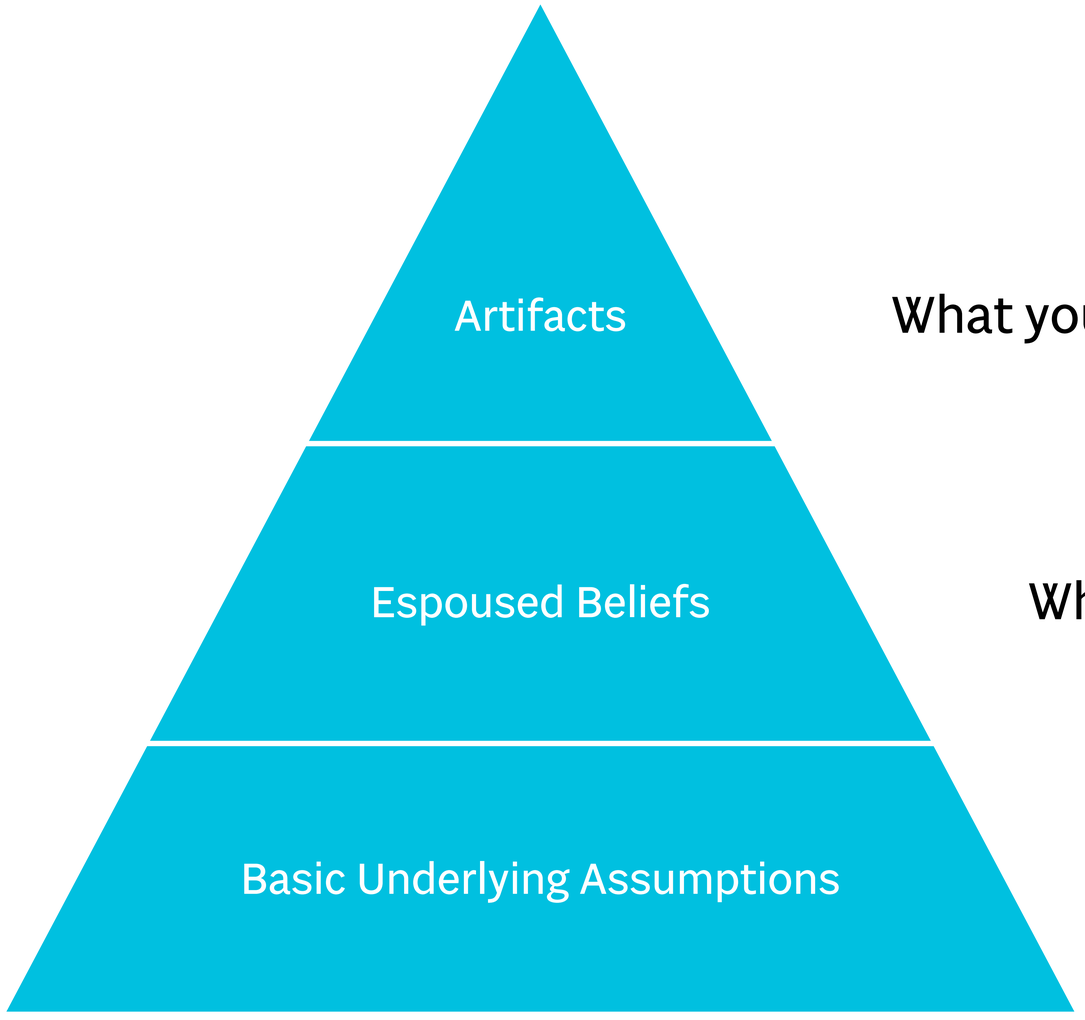
# Identify your stabilizing forces


Authority
Tradition
Value

# Design System Culture

# Cultural incompatibility.

**Organizational Culture** is
made up of three layers




What you see

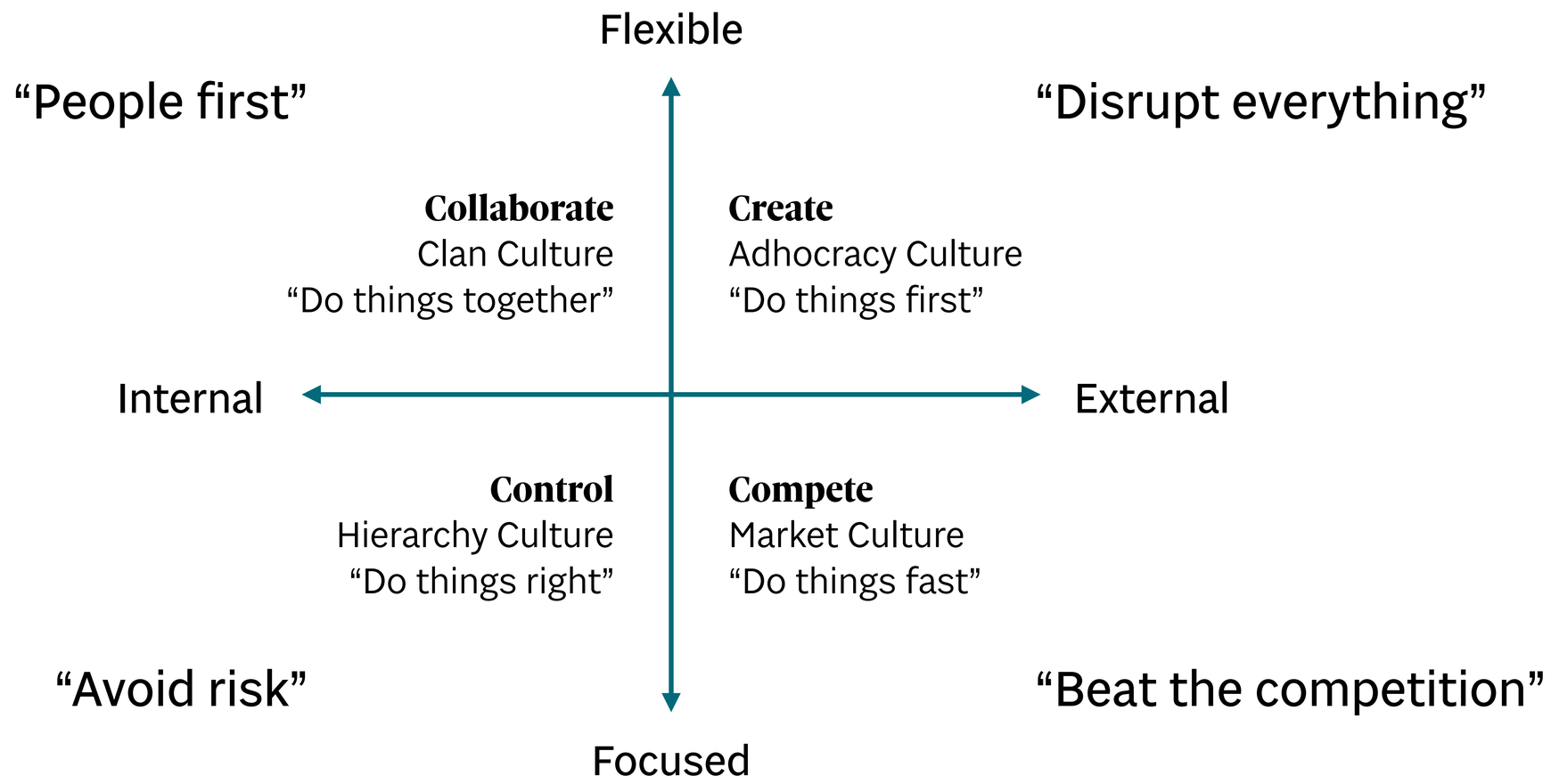
What people say

What people believe

**Experience**

http://www.scheinocli.org/

There are four primary types of **Organizational Culture** in the competing values framework




https://www.thercfgroup.com/files/resources/an_introduction_to_the_competing_values_framework.pdf

Design system subcultures are most often
**Collaborative** or **Controlling**

**Collaborate**
Clan Culture
“Do things together”

“I believe the scope of my work cannot be accomplished in any way other than to partner with teams who have caught the vision”

**Control**
Hierarchy Culture
“Do things right”

“My product teams haven’t created consistency, so I’m here to make sure they do going forward”

The spectrum from collaborative to controlling
helps us identify **healthy** and **unhealthy** behaviors

Unhealthy
Collaboration

**Collaborate**
Clan Culture
“Do things together”

**Control**
Hierarchy Culture
“Do things right”

Unhealthy
Control

## Prioritization

Tries to prioritize everyone’s needs (serves no-one well)

Primarily based on **subscriber** needs

Primarily based on **organization** needs

Ignores input from subscribers (serves no-one well)

The spectrum from collaborative to controlling
helps us identify **healthy** and **unhealthy** behaviors

Unhealthy
Collaboration

**Collaborate**
Clan Culture
“Do things together”

**Control**
Hierarchy Culture
“Do things right”

Unhealthy
Control

# Adoption

Unsustainable
over-support through
the adoption process

Primarily driven
by **relationship**

Primarily driven
by **incentive**

Unsustainable
non-support through
the adoption process

The spectrum from collaborative to controlling
helps us identify **healthy** and **unhealthy** behaviors

Unhealthy
Collaboration

**Collaborate**
Clan Culture
“Do things together”

**Control**
Hierarchy Culture
“Do things right”

Unhealthy
Control

# Extension

Extreme flexibility
(extension of the
system is overused)

Intentionally flexible
(extension of the system
is **encouraged**)

Necessarily restrictive
(extension of the system
is **discouraged**)

Extreme restriction
(extension of the
system is forbidden)

The spectrum from collaborative to controlling
helps us identify **healthy** and **unhealthy** behaviors

Unhealthy
Collaboration

**Collaborate**
Clan Culture
“Do things together”

**Control**
Hierarchy Culture
“Do things right”

Unhealthy
Control

## Change Management

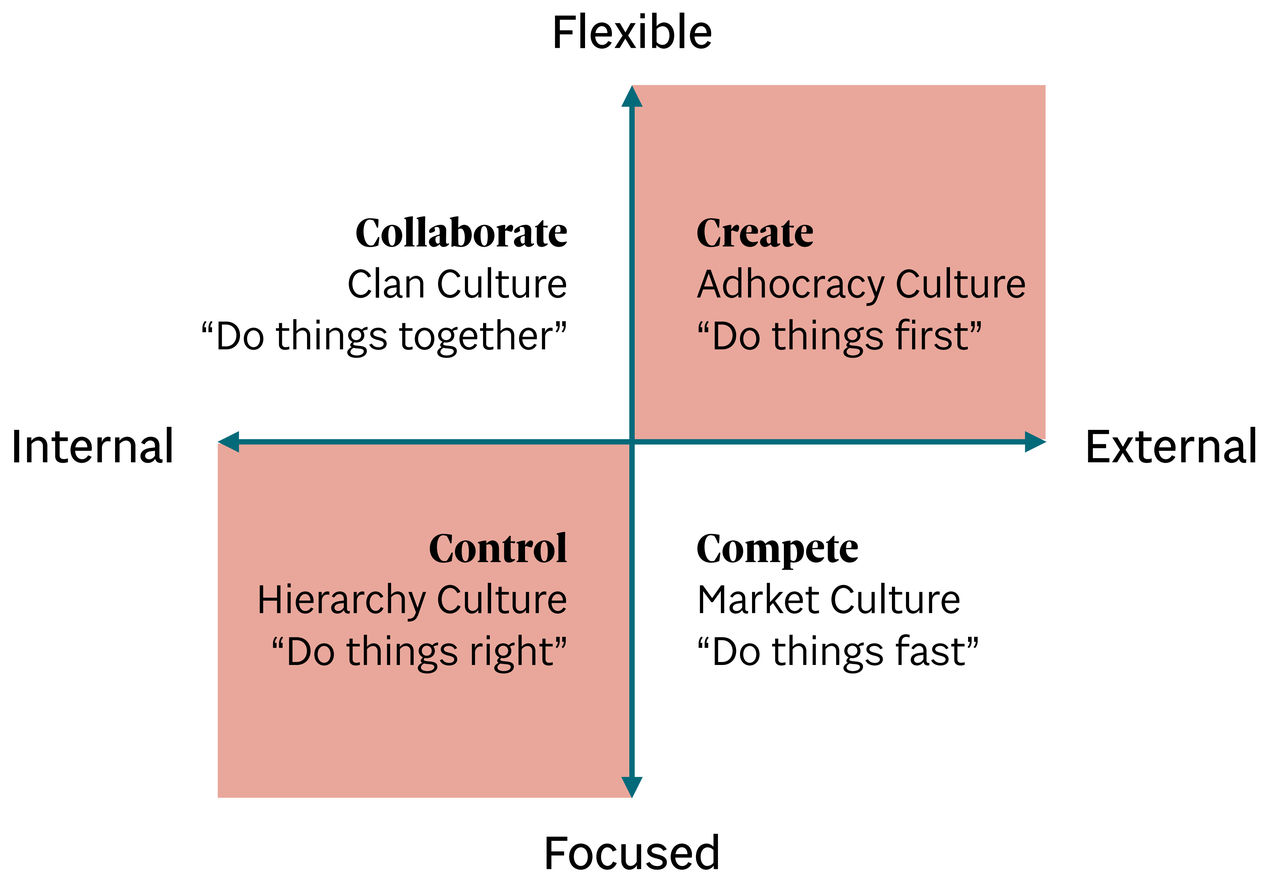
Extreme accommodation for subscriber workflows (decision paralysis)

System primarily adapts to the workflows of the subscribers

Subscribers primarily adapt to the approach of the system

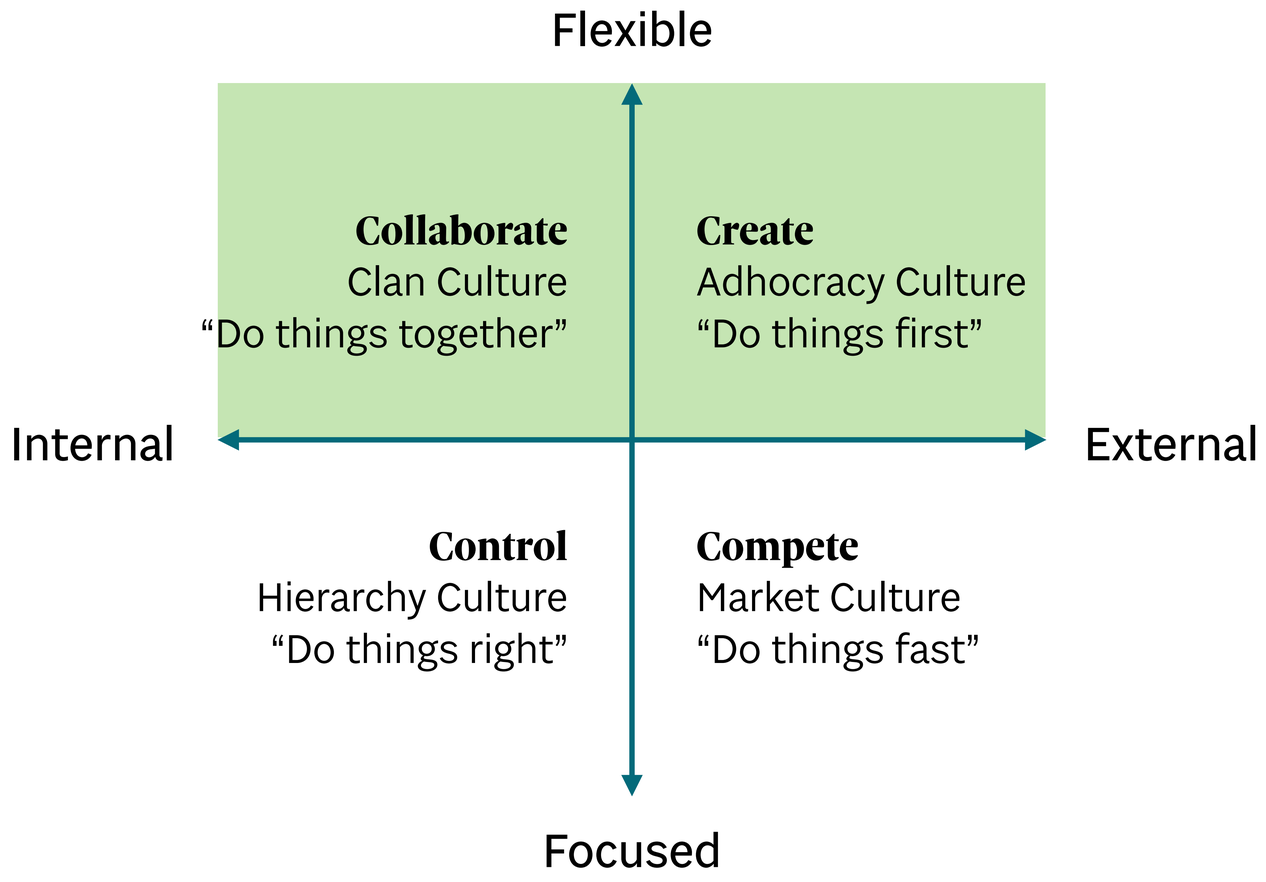
Lack of accommodation for subscriber workflows (regular rework)

There are four primary types of **Organizational Culture** in the competing values framework



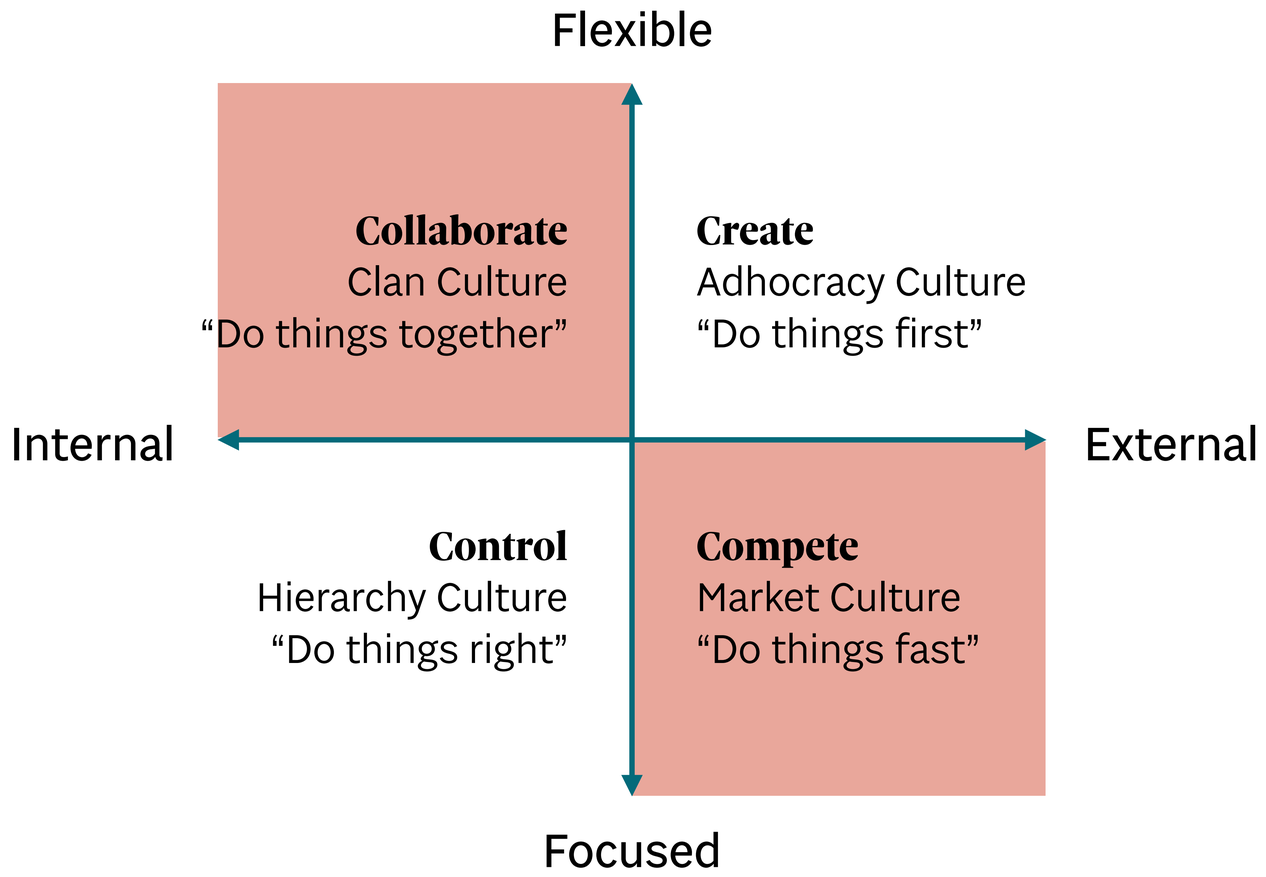

https://www.thercfgroup.com/files/resources/an_introduction_to_the_competing_values_framework.pdf

There are four primary types of **Organizational Culture** in the competing values framework



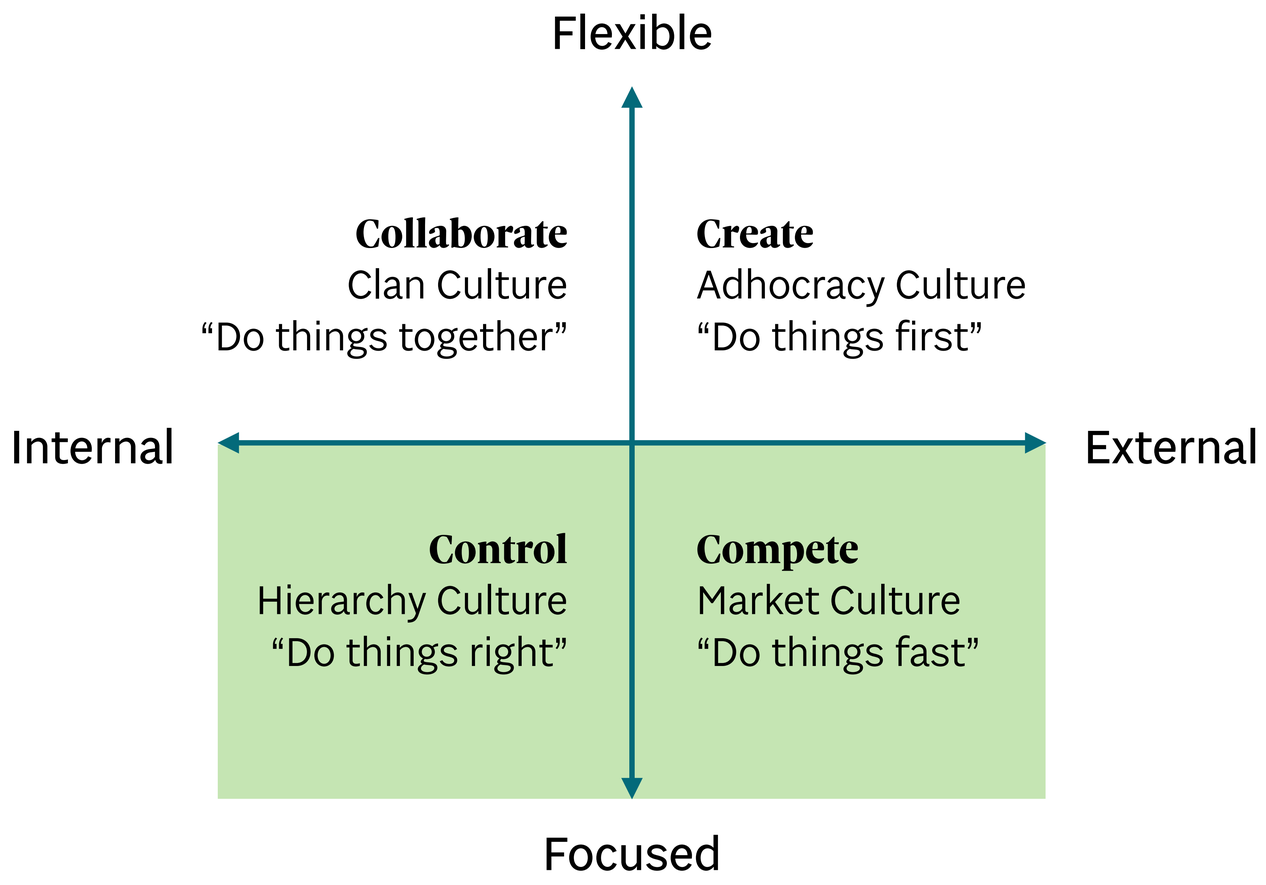

https://www.thercfgroup.com/files/resources/an_introduction_to_the_competing_values_framework.pdf

There are four primary types of **Organizational Culture** in the competing values framework




https://www.thercfgroup.com/files/resources/an_introduction_to_the_competing_values_framework.pdf

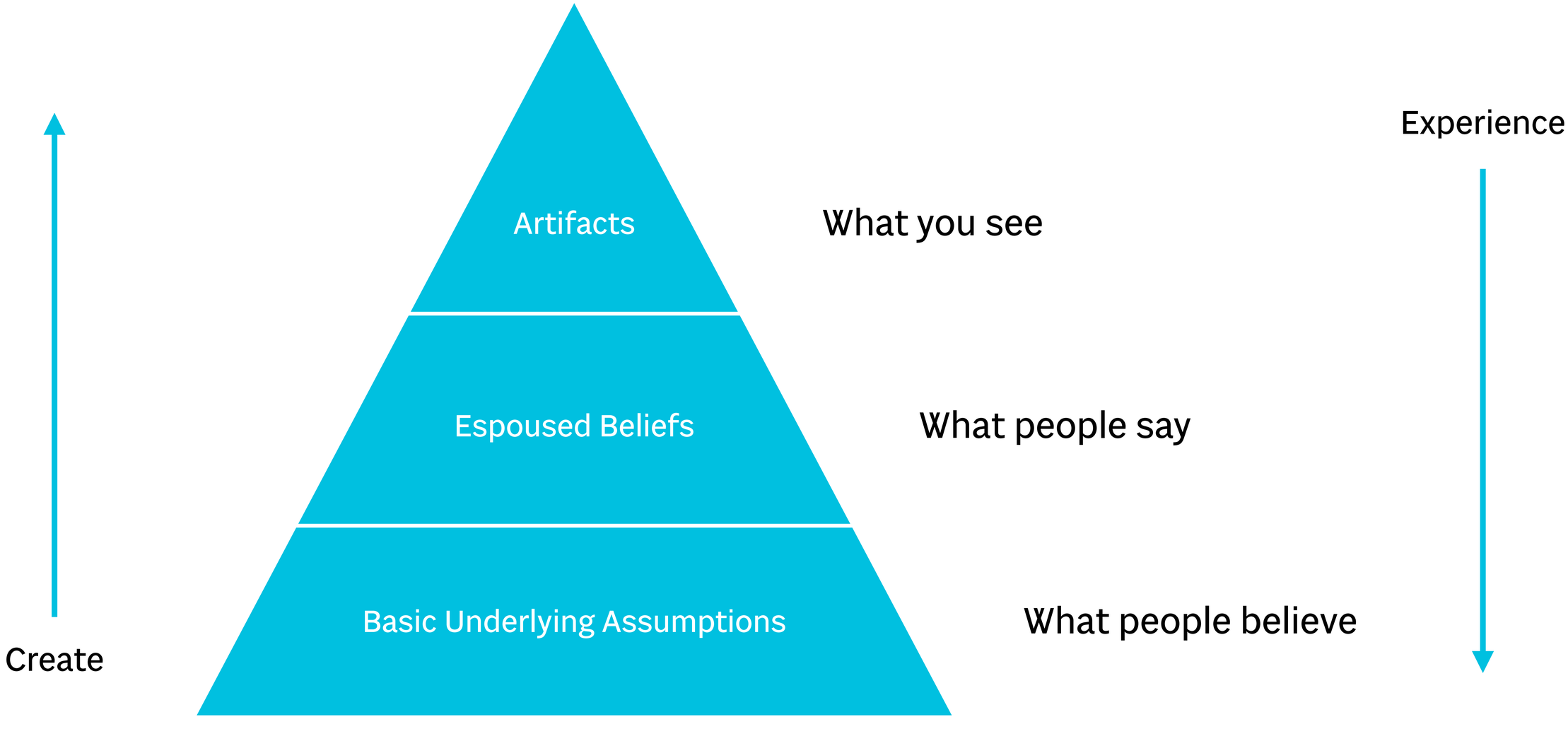
There are four primary types of **Organizational Culture** in the competing values framework




https://www.thercfgroup.com/files/resources/an_introduction_to_the_competing_values_framework.pdf

# Can the design system subculture change the culture of the organization?

**Organizational Culture** is
made up of three layers




http://www.scheinocli.org/

# Your design system won’t make your products more consistent.

# People will make what they want to make.

People will make what they want to make.

Your job is to change
~~People will make~~
what they <u>want</u> to make.

# Final Thoughts

# Start how you want to end

If you don’t model early on how you expect to interact with your subscribers, it will be difficult to change over time.

# It takes time

Design systems are only valuable over a long period of time.

This is not a quick fix.

# Measure adoption first

Without trustworthy adoption metrics, you can’t measure anything else.

Next, make a financial case for your system.

# Plan for using your gains

If you don’t make a plan for how to use the efficiency gains you will see, they will be used to cram more work in.

# Discoverability is king

Design system doc sites without good discoverability will suffer.

If people can’t find what they need, they’ll make their own.

# Transparency vs Parity

Your system will never fully be in sync.

Transparency on the state of things is how you manage expectations.

# Resources

SPARKBOX

# 2022 Design Systems Survey

## Our fifth year of the Design Systems Survey results are available here

https://bit.ly/sparkbox-ds-survey

SPARKBOX

# DS Study: Better Code, Faster

Our team ran a study to determine the impact a design system can have on efficiency.

https://bit.ly/ds-better-faster-code

SPARKBOX

# Design System Calendar

If you manage a design system, this simple calendar subscription will remind you regularly of critical things to be considering.

https://bit.ly/ds-sb-calendar

SPARKBOX

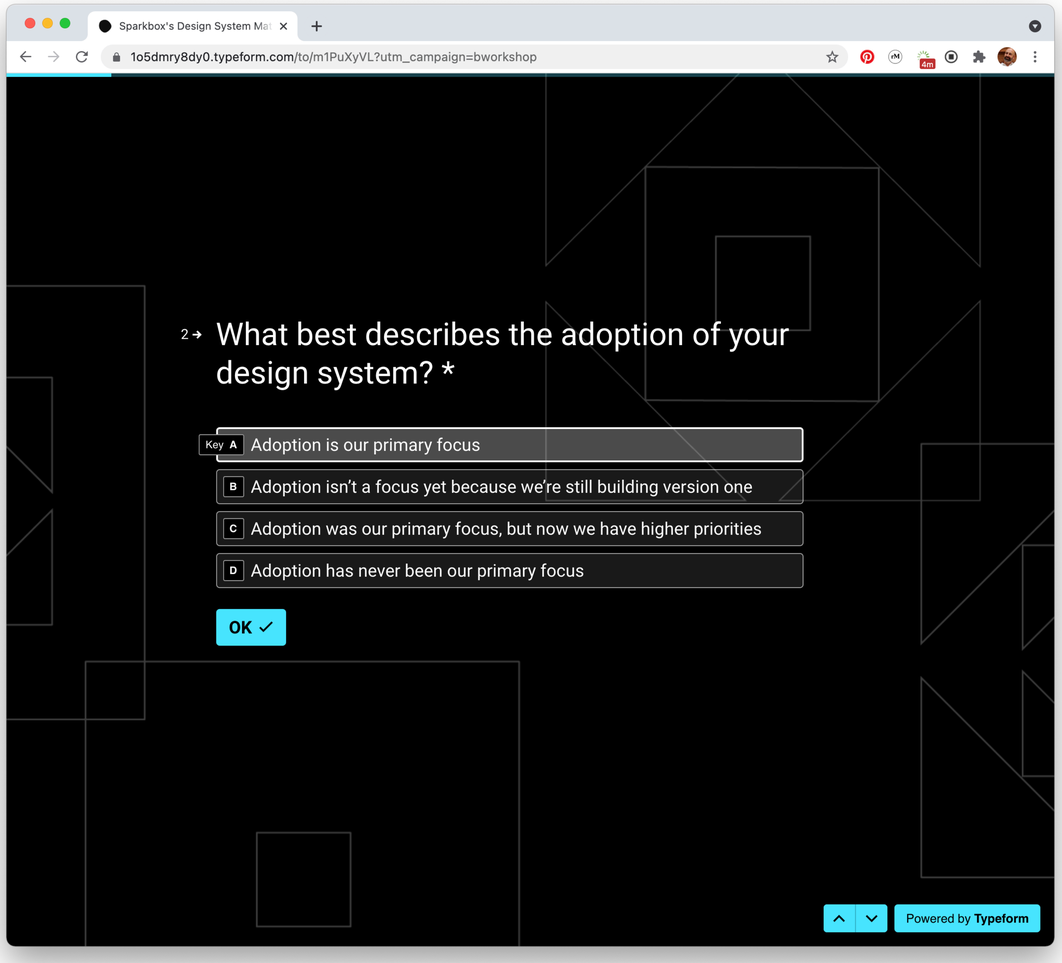
# Selling Design Systems

A big part of education and engagement is being wiling to be a salesperson for your design system.

https://bit.ly/selling-design-systems

# Maturity Assessment

We've designed a Maturity Model Assessment that will help you identify your stage and offer customized advice on next steps.




https://bit.ly/ds-maturity-assessment

# Connect with me

ben@heysparkbox.com

bencallahan (on Twitter & Mastodon)

https://bit.ly/connect-with-ben

SPARKBOX

# Building a Better Web

Sparkbox partners with complex organizations to create user-driven web experiences.

sparkbox.com