



Настя Брыкова
Design System Designer

Гайд по построению мультибрендовой системы токенов и иконок в рамках дизайн системы

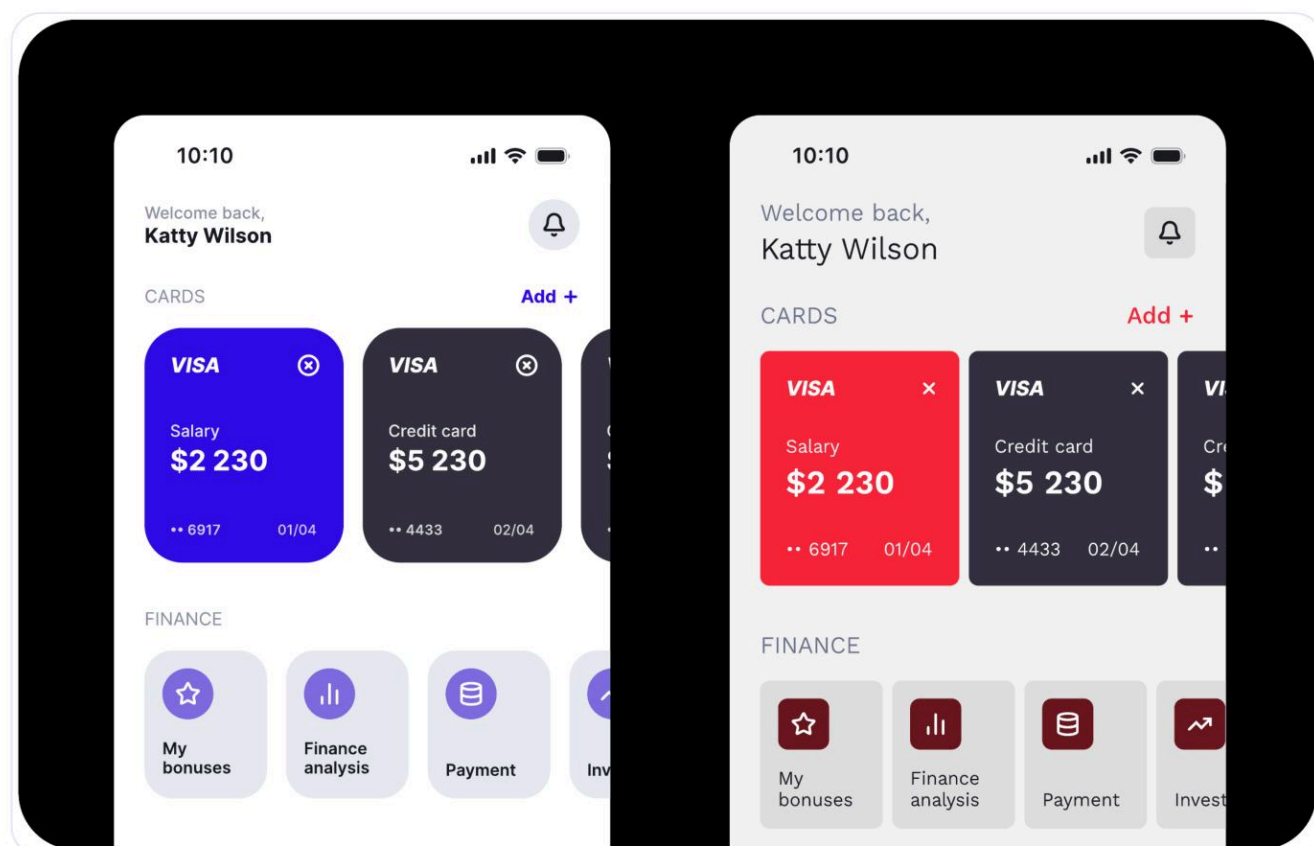
Построение мультибрендовой дизайн-системы

Tokens

Graphic assets

Этот гайд предназначен для продуктовых дизайнеров и команд дизайн-систем, которые работают с несколькими брендами в рамках одного продукта или экосистемы.

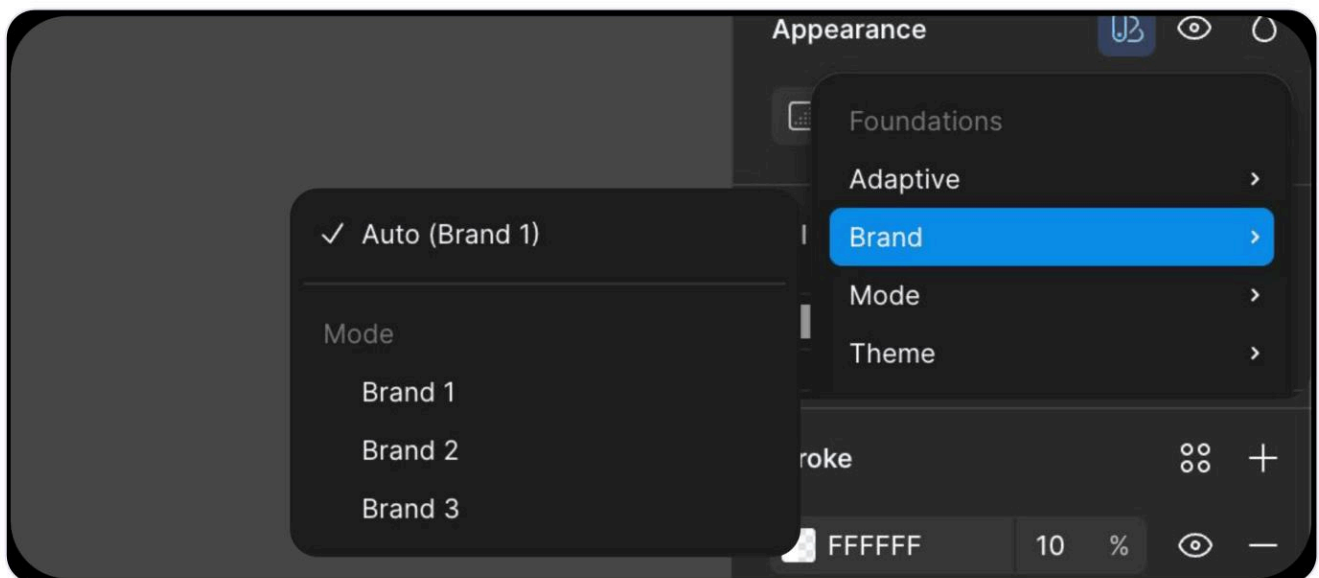
Он будет полезен, если вы создаёте мультибрендовую систему с любого уровня, развиваете white-label-продукт или хотите объединить разрозненные библиотеки в единую управляемую структуру



Tokens

В нашем продукте мы создали гибкое переключение визуала интерфейса за счет параметров Adaptive, Brand, Mode и Theme

- Adaptive переключает параметры текста в зависимости от устройства
- Brand применяет настройки выбранного бренда
- Mode переключает интерфейс между режимами обычного и премиального пользователя
- Theme задаёт светлую/темную тему интерфейса



Чтобы посмотреть интерфейс второго бренда в тёмной теме для премиального пользователя, достаточно задать выбранным экранам следующие значения: Brand – Brand 2, Mode – Premium, Theme – Dark

В гайде мы разберём общую логику построения архитектуры токенов на примере двух параметров – Brand и Theme. Эту логику можно будет легко масштабировать под особенности вашего продукта для других параметров

Основные сущности Variables

Архитектуру токенов мы будем настраивать с помощью Figma Variables. Перед практической частью разберём основные сущности

- **Collection (Коллекция)** — верхний уровень структуры, объединяющий связанные переменные, группы и режимы
- **Group (Группа)** — раздел внутри коллекции для объединения переменных по типу
- **Nested group (Вложенная группа)** — вложенная группа, которая помогает создавать дополнительные уровни иерархии
- **Variable mode (Режим с переменными параметрами)** — набор переменных для определённого контекста, например бренда
- **Variable (переменная)** — отдельный токен с заданным именем и значением

Foundations		Color primitives	
Collections	+	Name	Value
Brand	202	0	FFFFFF
Mode	320	5	F6F6F6
Theme	636	10	ECECED
Adaptive	219	15	DDDDDF
Color primitives	294	20	CBCBCD
Dimension primitives	66	30	B1B1B4
Groups	≡	40	8B8B8F
All	294	50	6D6D71
Brand 1	98	60	505054
Grey	14		
Blue	10		

Иерархия

Далее разберем основы иерархии структуры токенов.

Архитектура токенов строится снизу вверх. На нижнем уровне находятся примитивы — базовые значения цветов, отступов, радиусов и других свойств.

Токены на следующих уровнях используют значения примитивных токенов. Чем выше уровень, тем конкретнее назначение токена: от базового цвета до цвета определённого элемента интерфейса. В такой структуре изменение примитивного токена автоматически распространяется на все связанные токены выше по структуре.

Group Brand	Token name key/primary	Value Theme/key/primary	
Group Theme	Token name key/primary	Value Blue50	
Group Color primitives	Token name Blue50	Value #6099FF	

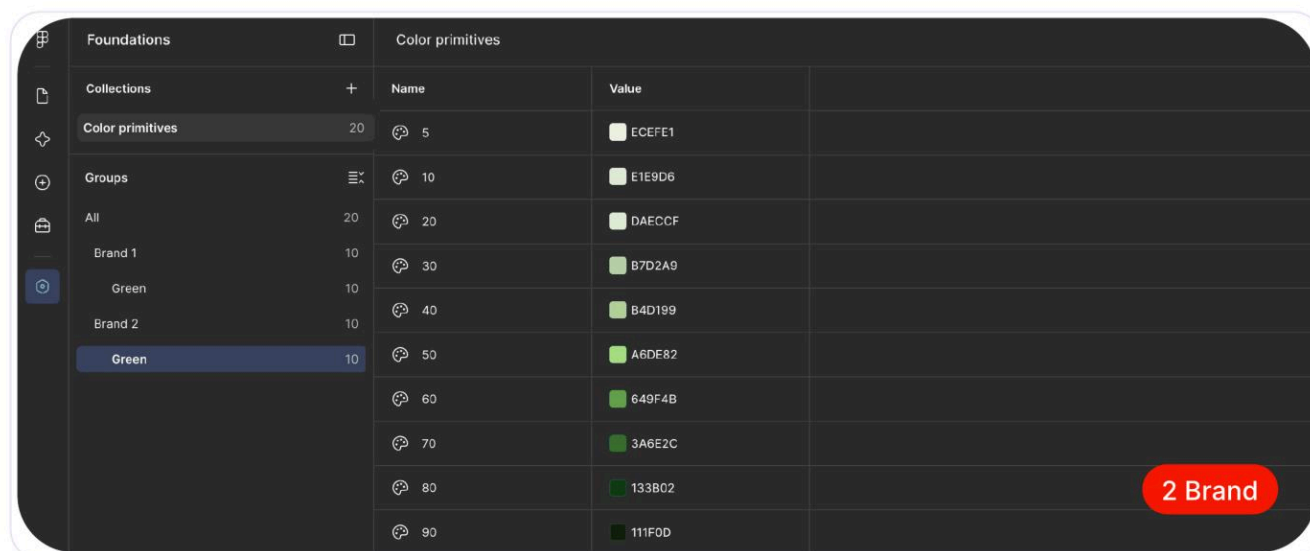
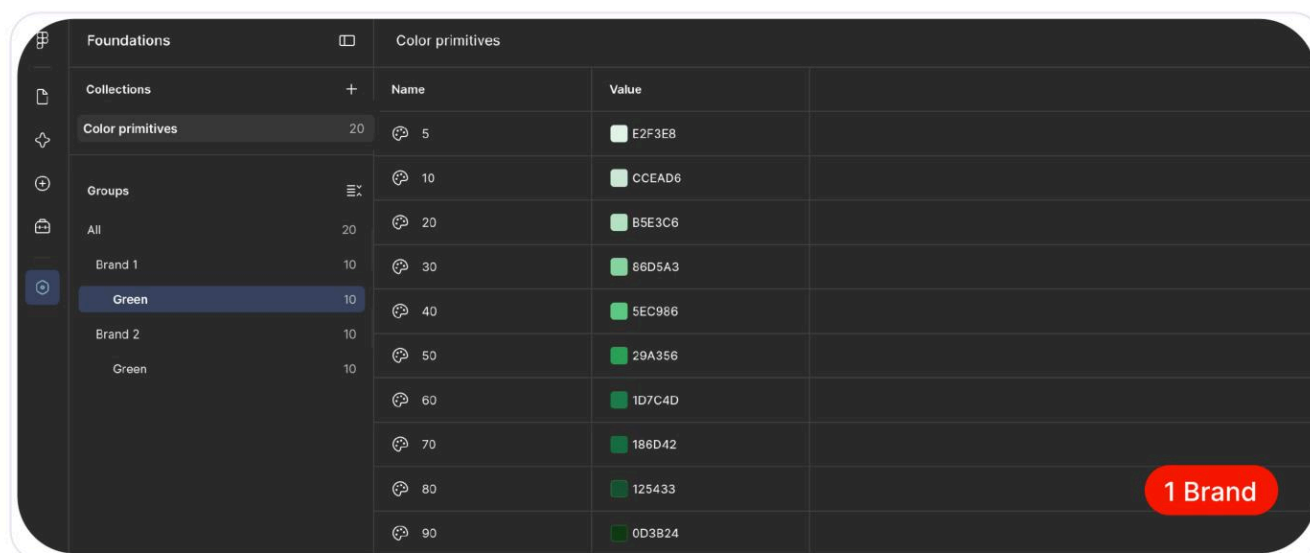
Поэтому важно сначала определить базовый набор примитивных токенов. Допустим, нам нужно переключать только цвета между брендами, то тогда создаем палитры примитивных цветов.

3

Tokens

Коллекция примитивов

1. Создайте коллекцию примитивных цветов
2. Добавьте в коллекцию две группы: Brand 1 и Brand 2.
3. Внутри каждой группы создайте вложенную группу с названием цвета, например Green
4. Создайте внутри нее палитру этого цвета. У нас на примере этот цвет разложен на 10 оттенков
5. Если цвет различается у брендов, задайте для палитры Brand 2 собственные другие значения HEX



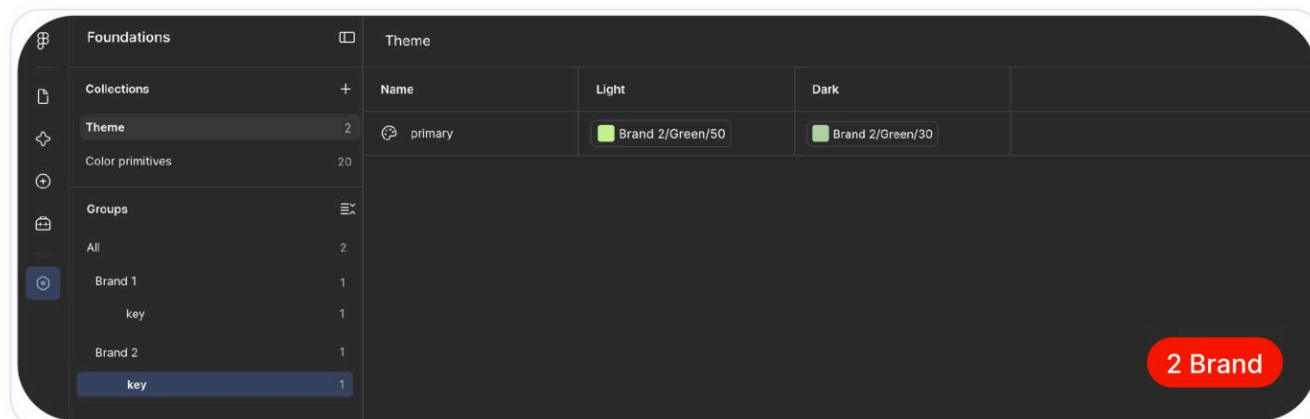
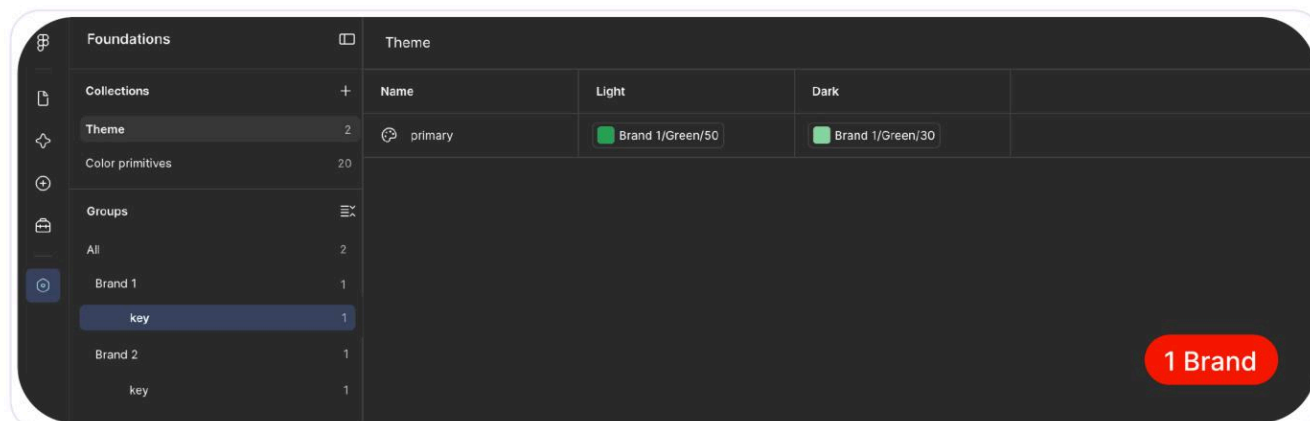
4

Tokens

Коллекция темы

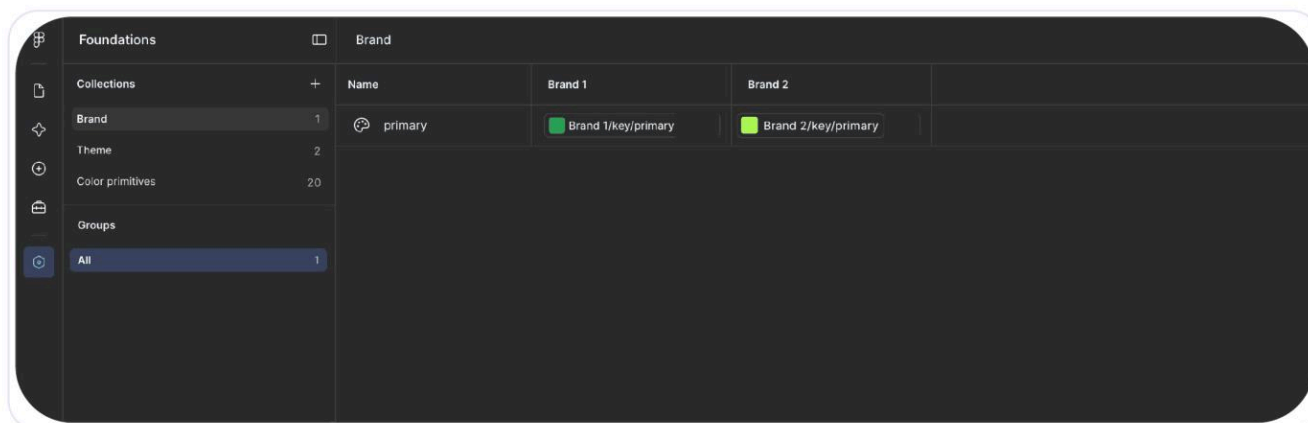
1. Создайте следующую коллекцию – Theme. Она будет отвечать за переключение светлой и тёмной тем
2. Добавьте в неё две группы: Brand 1 и Brand 2
3. Внутри каждой группы создайте вложенную группу по семантической роли цвета, например key
4. Добавьте Variable Modes: Light и Dark
5. Создайте внутри групп семантический токен, например primary, и для каждого режима свяжите его с нужным оттенком из коллекции Color primitives

Важно: семантический токен должен ссылаться на примитив своего бренда. Например, токен из группы Brand 1 – на Brand 1/Green/50, а токен из Brand 2 – на Brand 2/Green/50



Коллекция бренда

1. Создайте следующую по иерархии коллекцию — Brand. Она будет отвечать за переключение между брендами
2. Добавьте Variable Modes: Brand 1 и Brand 2
3. Создайте внутри коллекции группы по семантическим ролям цветов, повторяя структуру коллекции Theme. Например, группу key
4. Внутри группы создайте семантический токен, например primary
5. Для каждого режима свяжите токен с соответствующим токеном из коллекции Theme:
Brand 1 → Brand 1/key/primary
Brand 2 → Brand 2/key/primary



Если бренды различаются не только цветами, но и типографикой, радиусами или другими параметрами, создайте в коллекции Brand соответствующие группы, например, Typography и Radius. Тогда при переключении бренда одновременно изменятся все связанные параметры интерфейса, а не только цвета

6

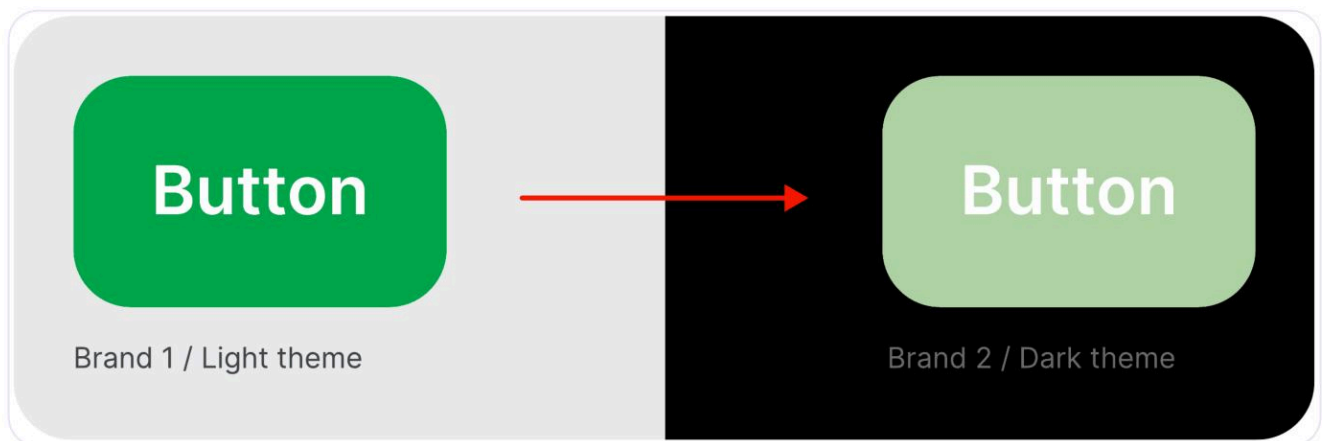
Tokens

Переключение бренда в макете

Чтобы переключить объект между Brand 1 и Brand 2:

1. Создайте любую фигуру в файле, где настроены коллекции переменных
2. Примените к её заливке цветовой токен из коллекции Brand
3. В разделе Appearance на правой панели найдите параметр Brand: Brand 2
4. Так же проверьте переключение темы. Выберите параметр Theme: Dark

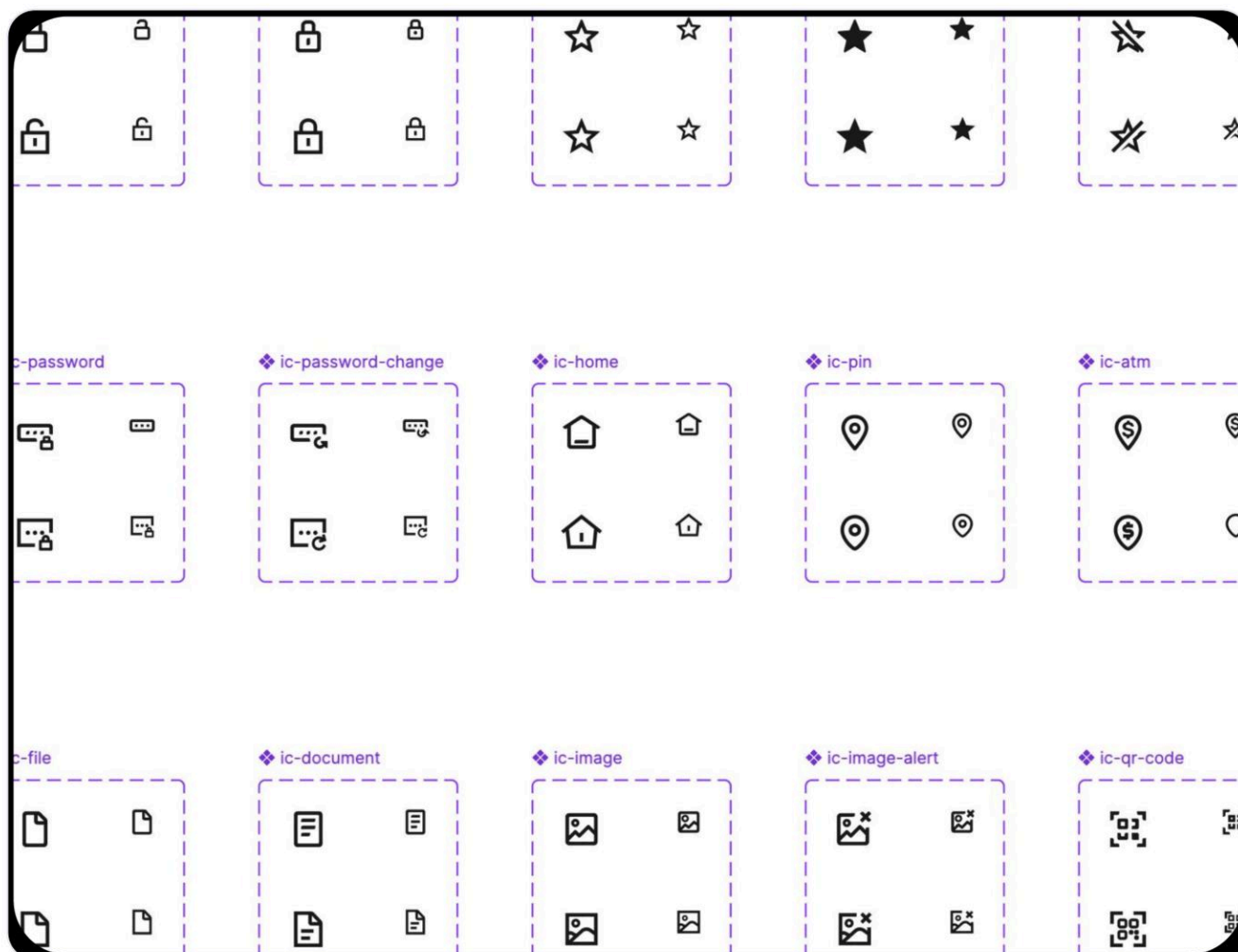
После переключения цвет и другие связанные параметры объекта изменятся в соответствии с настройками второго бренда



После создания и проверки полного набора токенов опубликуйте библиотеку и тогда переключение бренда станет доступно в файлах, к которым она подключена

Графические ресурсы в мультибрендовой дизайн-системе должны переключаться так же автоматически, как цвета, типографика и другие параметры интерфейса. При этом между брендами могут различаться цвет и стиль

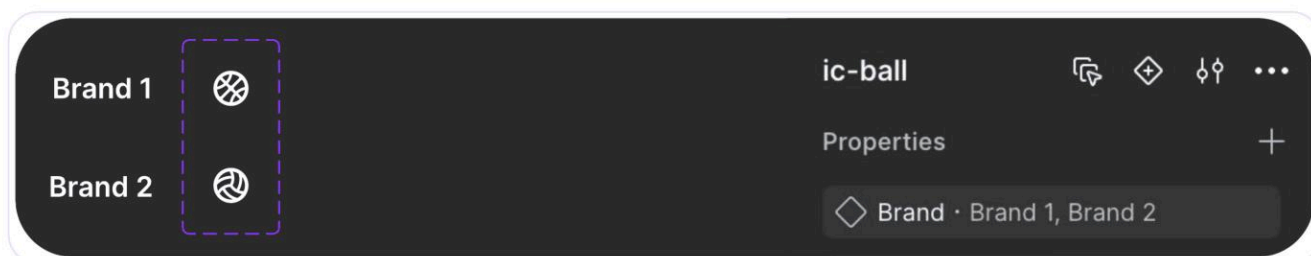
В этом разделе разберём, как связать варианты иконок с коллекцией Brand, объединить ресурсы из разных библиотек и автоматизировать часть работы с помощью Figma MCP и AI-агента. Также рассмотрим особенность Figma, из-за которой цвет иконки может сбрасываться при переключении бренда внутри компонентов дизайн-системы



Настройка Variables

Автоматическое переключение между брендами поддерживают все графические ресурсы: иконки, логотипы, иллюстрации. Как настроить дизайн-систему, чтобы при переключении брендов менялся стиль ресурсов:

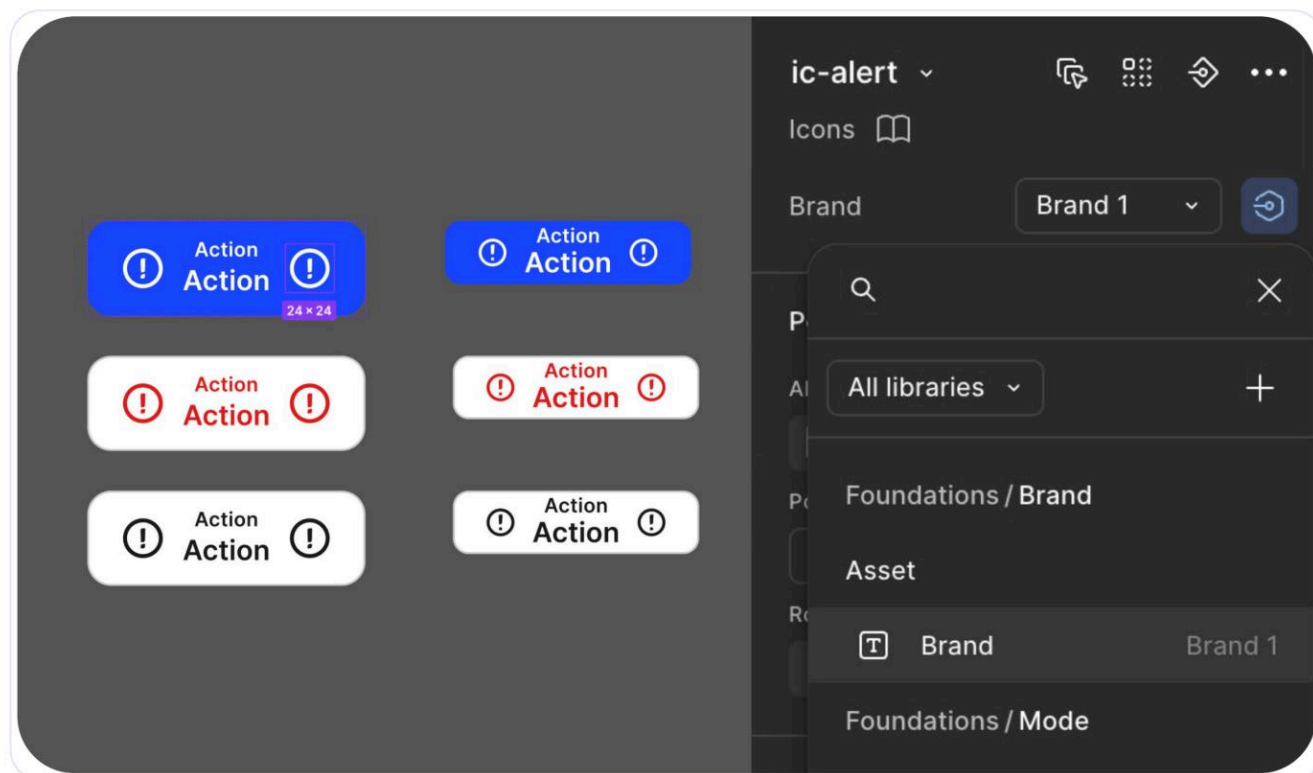
1. Соберите версии одного графического ресурса в общий Component Set
2. Добавьте свойства, которые соответствуют различиям между вариантами: Brand 1 и Brand 2



3. Создайте группу в коллекции Brand и назовите ее, например, Asset. Внутри этой группы создайте свойство для ресурса в Variables в библиотеке с токенами дизайн-системы. Его варианты должны совпадать со параметрами внутри Component Set: Brand 1 и Brand 2

Name	Brand 1	Brand 2
Asset		
 Icons	Brand 1	Brand 2

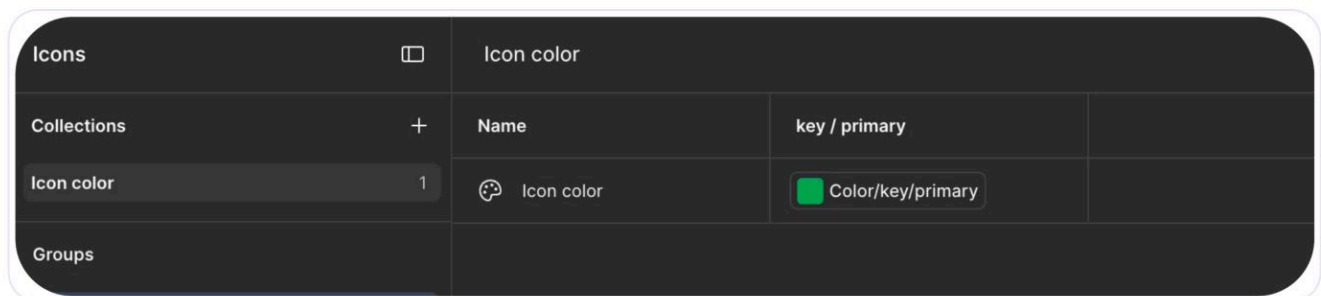
4. В компонентах дизайн-системы привяжите свойства вложенных экземпляров ресурсов к соответствующим variables



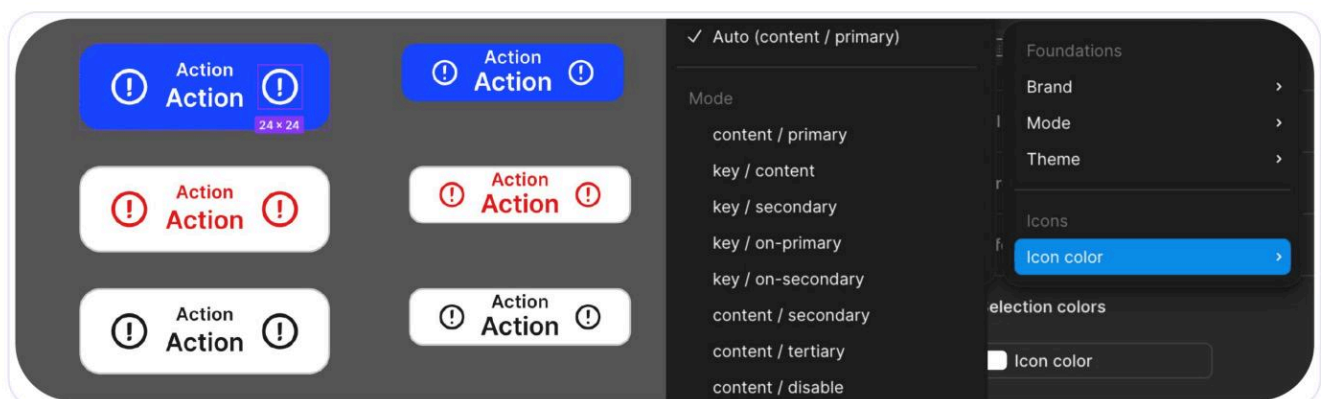
Привязка цвета к иконкам

В Figma есть проблема: при переключении бренда цвет иконки внутри компонента дизайн-системы сбрасывается. Отдельных иконок это не касается. Например, вы сделали иконку внутри кнопки красной, но после переключения с Brand 1 на Brand 2 она снова стала цвета по умолчанию. Чтобы исправить эту проблему, сделайте следующее:

1. Создайте цветовые токены в Variables файла с иконками. Обязательно привяжите туда уже существующие токены из дизайн-системы



4. Переназначьте цвета иконок в компонентах через правую панель инструментов в разделе Appearance. Таким образом цветом иконки будет управлять варианты из Appearance, а не токен в Fill. При переключении вариантов компонента цвет вашей иконки сохранится





Объединение библиотек

Если нужно объединить несколько библиотек ресурсов разных брендов, часть работы можно автоматизировать с помощью Figma MCP и AI-агента.

Зафиксируйте правила обработки ресурсов в отдельной инструкции или skill:

- Как определять ресурсы с одинаковым смыслом
- Как называть итоговые Component Set
- Какие свойства и значения использовать для каждого типа ресурсов
- Какие варианты можно объединять
- Как обрабатывать неполные группы и исключения
- Какие проверки выполнять после изменений

Автоматизация должна работать только с существующими ресурсами: объединять подходящие варианты, нормализовать свойства и сообщать о пропущенных сочетаниях или дубликатах

Такой подход позволяет одинаково обрабатывать новые партии ресурсов, поддерживать структуру библиотеки и снижать количество ошибок при добавлении новых брендов

Чтобы упростить работу — отправьте в LLM эту инструкцию и попросите создать готовый скилл, который будет управлять иконками и библиотекой токенов за вас