The Design-to-Development (D2D) Contract

> This page describes how to organize and name components and their elements in order for the Style Forge to work properly and provide developers with concise, comprehensive and consistent theme file

The **D2D Contract** enables Figma Design Assets to be turned into React Web Components in the Design System. The spirit of the agreement is for designers and developers to collaborate on both the design and the dev so that what gets produced in Figma produces a stable theme.

In order create React components, there are some **naming and structural rules designers must follow when creating design assets in Figma so that the Style Forge (SF) parser properly converts** the component's attributes into usable code.

The D2D contract is both explicit and implicit in nature.

While the contract does demand that the design of components follow specifics in regard to order, structure, naming, hierarchy, and limitations, it's also understood that the contract requires human input in the process and reasonable flexibility as no two components are exactly alike in shape, size, behavior, variation, or features.

General Guidelines

Designers and Developers should get together at the start to discuss their approach when designing Net-New Components or Making Major Updates to Existing Components

Get together and align the creation of a component so that naming and structure, properties and attributes are ironed out at the start of the process since design drives the code.

Together the team can apply the proper rules to component naming of layers, hierarchy, and configuration so the theme schema and output is valid and produces the data quality needed for Web components.

Contract for Designing Design System Components for the SF

There's no "one" way to design system components, but there are standards to follow in order to support development of DS components. It's a balancing act getting design assets to satisfy both designer's needs, and development requirements.

Basics

- Work in **branches** and pair with other designers to match expectations, and to iterate, evolve and validate the design and usability requirements of the component. Whatever is designed **must** work for designers.
- **Pair with Development early** on to analyze, improve, and align design thinking often. Whatever is designed **must** work for Development.
- **Look to leverage the design of existing components as a template/guide** that ensures consistency and alignment on things like: Naming conventions and schemas, layer structure, order and organization etc.
- **Design as flat, intuitively, and semantically as possible, with the least complexity** to support both designers and developers aka the SF. Proper use of Figma Frames, Shapes, Auto Layout, Sections and custom delimiters to hide or show nodes or style data.
- **Designers think like a developer and vice-versa.** Understand that whatever you are designing has to ultimately output specific styles, CSS, and code Dev can use. For example, designers should be familiar with HTML and CSS, browser constraints, viewports and platforms, the Flexbox model underpins Figma Auto Layout, that sometimes an extra frame may be needed to contain certain elements may be necessary to provide spacing, margins, padding information, or how elements interact and behave with other elements on the screen, as well as the bars of accessibility and usability DS users and experience consumers expect.

- **One page per component.** Pages usually include basic usage doc, usage-specific color and type tokens on the same page, and follow known schema and naming conventions. Depending on the component, these pages can also contain subcomponents, e.g. Type Ahead's menu, focus elements for Checkbox etc.
- **Consistency matters** when designers are used to working fast and messy, design system components require order and finesse, deliberateness and intention. There are many different ways to achieve the same outcome in Figma, so ensure that whatever your process is, results in a predictable outcome for the theme it's trying to generate. Test components for usability, how they appear in Figma Search and for the experience who will be using these components.

Specific Component Organization and Semantic Naming Rules for Designers

All Components Must be Variants

> **Components must be variants** in order to be parsed by the SF

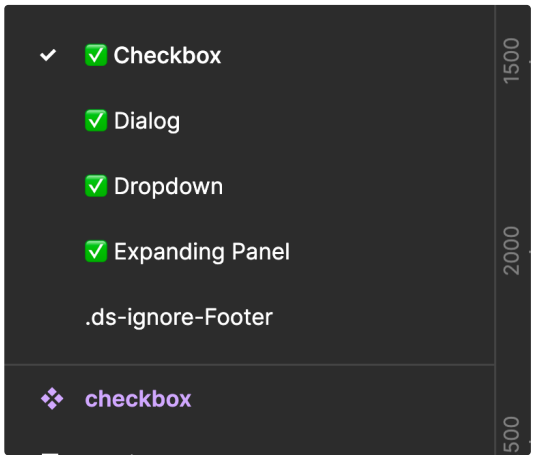
The Style Forge can only process components that are created as variants in Figma. Only variants are elements with a `COMPONENT_SET` type are handled by the SF's **Variant Parser**.

Component Names

Name of the COMPONENT_SET should be fully descriptive by itself, out of current context. The object name in the output theme is depending only on the name of COMPONENT_SET.

If there is multiple sets for the same component type, they should have different names, for example *button, button-link button-header etc.*

Component names should match the variant name, what's in Storybook, and the Component Set:



The Checkbox page has a component set named Checkbox

Property Names

Every combination of properties should be accounted for in a COMPONENT_SET. This is because during development we are creating a contract to consumer, that a component supports *properties x, y, z* with *values a, b, c*. In this case, every combination of properties should be supported, as there is no fallback scenario in place.

If this would result in designing useless variants, it would be good to think about splitting the COMPONENT_SET into multiple sets.

Property Name style is keep to one or two words max, and to initial-capitalize each.

Property Values

Values of the properties should be fully descriptive. The way we are defining component in the theme is by the values of properties it has in Figma. This means that button>medium, primary, idle is fine, but introducing boolean variable (for example if the button has an icon) would lead to name button> medium, primary, idle, false, which is not very descriptive. Instead it should be something like `no-icon`.

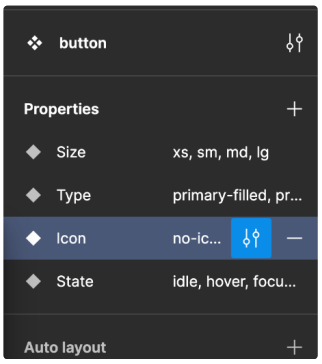
Always consult names and values of properties with your friendly neighborhood developer. It will save time and issues in the long run.

Property Value style is to keep to one or two words at most, and to initial-capitalize each word.

Order of appearance for Property Name/Value Pairs

Once Property Value pairs are established the SF expects them to come through in this order (if applicable):

1. Size
2. Type
3. Shape
4. *n* Variant(s)
5. State



Button Properties follow a set order of appearance the SF expects.

Master Components & Variant Instances

We are still using Masters and Instances to manage components and create variant sets.

A Component Master contains every possible permutation of a variant set.

For a button, this means, its label, border, left and right icons are all in the Master, while it's variants may contain all or a subset of these attributes. Masters are what Figma has long recommended to enable global maintenance for a variant set, and are foundational.

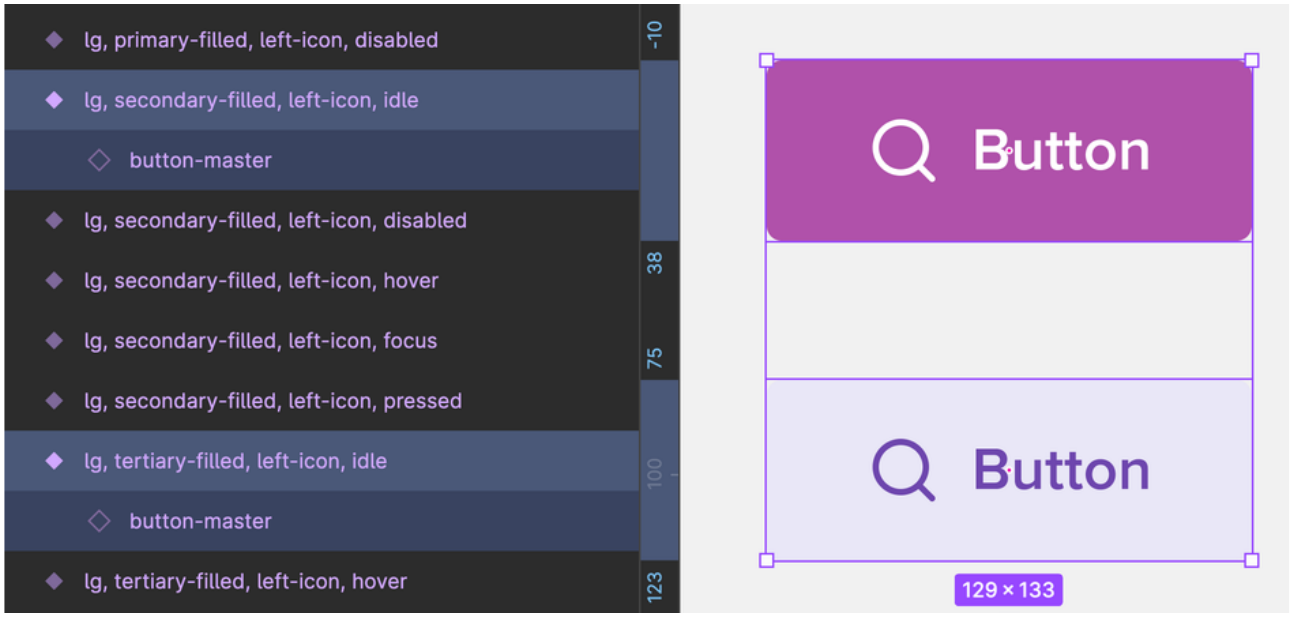
> We are keeping watch on changes to this paradigm and advances in the prototyping feature, as Figma has suggested recently (Q1 2023) that masters may have performance issues in complex prototypes.

Master Components are intentionally left outside of their corresponding variant set because a Master cannot be used in Figma prototyping, only its Instances as variants contained in a frame, are supported.

By leaving it out as a so-called "Partial Component" it retains its baseline resizing constraints, auto layout, layer names and properties which is helpful when making global changes to a whole batch of variants, vs. making point updates one variant at a time.

Identical Naming of Component Masters in Variant Instances to Support Content Statefulness

Instances in a component variant set must be identically named so that a change to one button label, extends to all variants when swapped for another using the same instance. Designers sometimes refer to this as having "sticky labels" meaning that one label change in one variation, e.g. Send, changes them all.

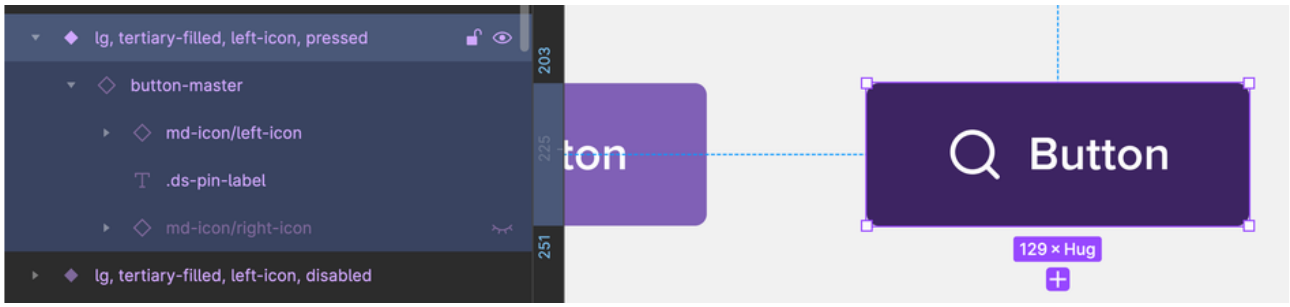


The button-master is identically named in all instances/variants of the button for each size of the button. Changing the button label to “Send” in one, changes them all for that size/type variant.

Icons and the md-icon Prefix

Icon elements should be named semantically, not with the default name of the icon.

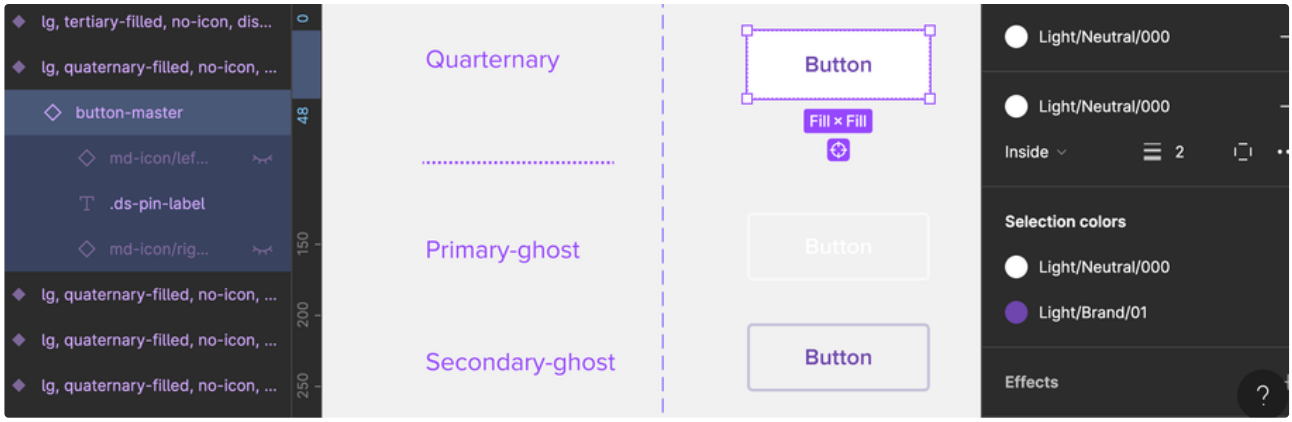
Icon layer names should have the `md-icon/` prefix. The specific icon, its color and dimensions are processed by another parser called the **Icon Bridge**.



The search icon is properly prefixed and semantically-named vs. specifically named

One with Borders, All with Borders

If one variant of a component has a border, **then all variants** of that component must have borders defined. This is to ensure that in both Dev and Figma Prototyping, there isn’t a visible “jitter” produced by unequally-sized components, e.g. one variant with a border and another without when a user interacts with it in the case of the button, through *idle-to-hover-to-focus-to-pressed-to-disabled* states.



The *Quaternary-Filled* button has a 2px border defined with the same SFckground fill, to enable the 2 *Ghost* variants. For variants such as the *Plain* variant the border still must be defined but at 0% opacity. This is in to ensure the SF gets consistent data for all variant types in COMPONENT_SET.

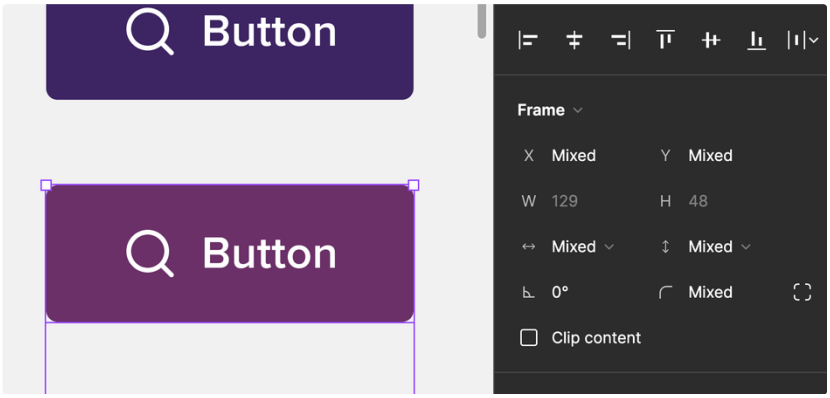
Shapes & Frames

Certain components require a specific Shape layer in order to produce styles in the theme output.

Often times a designer may need to use a frame to contain an object, e.g. icons to the left or right of a label, so that the correct Auto Layout settings can be produced in the theme.

Identical Bounding Boxes Dimensions

The dimensions of the containing bounding box of every variant in a COMPONENT_SET should be identical so that variants used in Auto Layout and in prototyping, have consistent spacing, padding, margins.



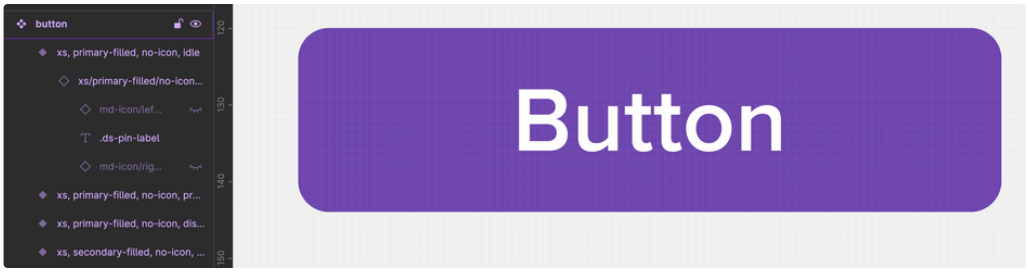
Regardless of the variant, all buttons in the same LG size category and same label text are 129px by 48px and have 12px minimum padding around the icon and label.

The .ds-pin prefix & Included Elements/Element pinning

You are able to elevate an element in the output theme by prefixing it with `.ds-pin-`. This action will create a separate object with name according to what is after the prefix, in the output theme. You should be consistent with this pinning, if you decide to to elevate label in a component, it should be elevated in all versions of the component (where applicable).

Any element in Figma: Layer, frame, shape that has a `.ds-pin-` prefix will be parsed by the Style Forge and added to the theme.

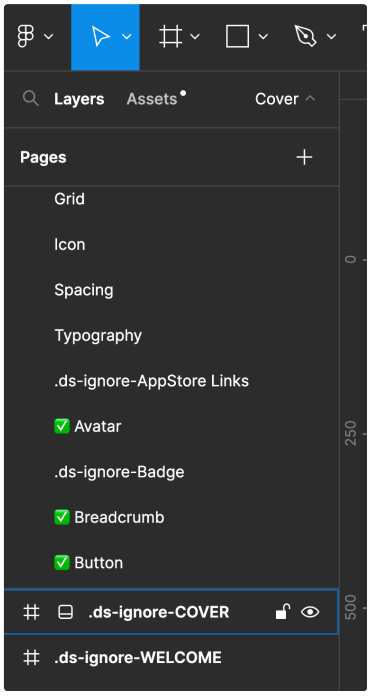
Use the `.ds-pin-` prefix when you need to explicitly force a node and associated styles to appear



In the layers panel on the left, the button's label is `.ds-pin-` prefixed to ensure it's style and color attributes are included with the Button node in the theme.

The .ds-ignore prefix & Hidden Elements/Elment Ignoring

Any element hidden within a **Figma Section** or with **underscore or period**, `.ds-ignore-` prefix is ignored by the SF. This element and any child element will not be processed.



The AppStoreLinks and SFdge Pages, as well as the Cover and Welcome page Frames, will not be parsed by the SF.

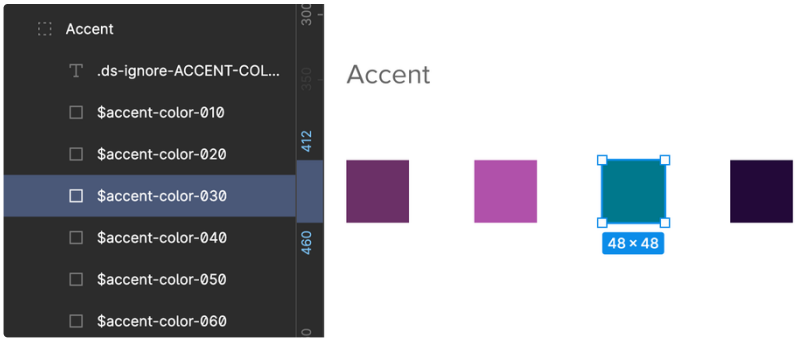
Anything such as a Page, Layer, Frame Name, Group, etc. leading with `.ds-ignore` will be ignored for the purposes of Bridge Adapter. It will not be searched through for tokens, for components

> Use the period, underscore or `.ds-ignore` `_ds-ignore` when there is something in the design of a component that's purely for Designers, to keep in-progress design artifacts, and/or are not ready to be processed by Development.

The $ variable prefix for Color and Type Style/Token Elements

In Figma, tokens are taken as variables, to be used consistently to drive color, type, etc.

All elements with a `$` prefix will be processed with a **Token Parser** and included in the SF output.



Styles are defined as variables using the `$` prefix so they can be parsed by the SF.

Note: Use of a Numerical in Color or Type Token Elements

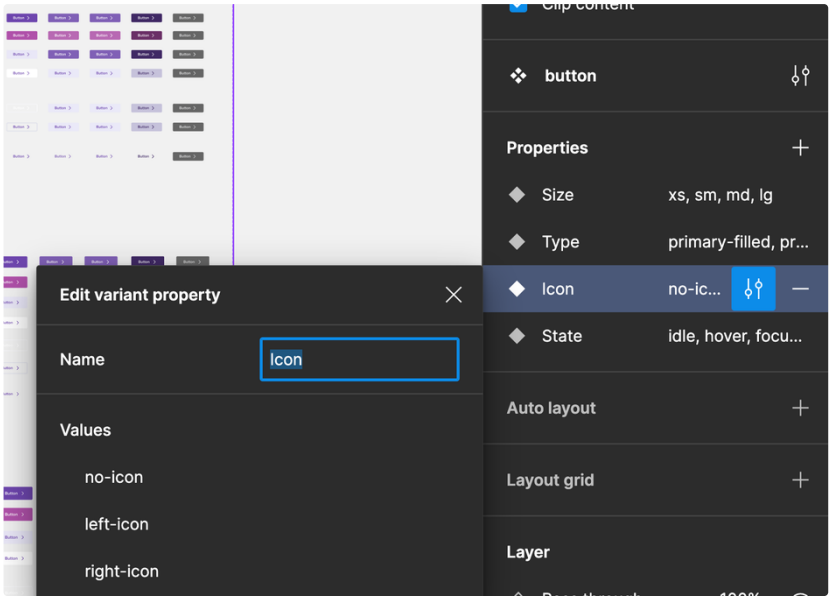
Numerical Indexes can be used so long as they are not leading a name, vs. trailing as seen in the example above for Accent color definitions.

Leading Emojis

A **green checkmark emoji** preceding a component page indicates it is complete, meaning there are design assets for a designer to use in mockups, and there's also a matching React Web Component.

Boolean Operators Not Yet Supported

The use of Figma's support of On/Off, True/False Property/Value pairs is very useful to designers, but for DS components booleans are not yet supported. Designers and Devs should get together to align on alternative property/value naming.



For the button, a version with an icon or without (*i.e. no-icon*) would usually be a boolean on its own, but it's grouped with a Property named *icon* that has values, *no-icon*, *left-icon* and *right-icon*.

These standards are always evolving. To get the latest, sync up with us on Slack, at #design-system-community, or email us at coreUX@monster.com to discuss your project needs from the DS.

Misc. Elements

These Elements are rarely if ever used by designers in components but are included here as informational and for transparency.

Config Elements

Anything prefixed `.ds-config-` is processed inside the SF's **Config Parser.** For now it concerns only fallback fonts, which aren't natively supported in Figma but are necessary for development purposes. You can see this in the main component library on the page for *Typography > Fall back Fonts*