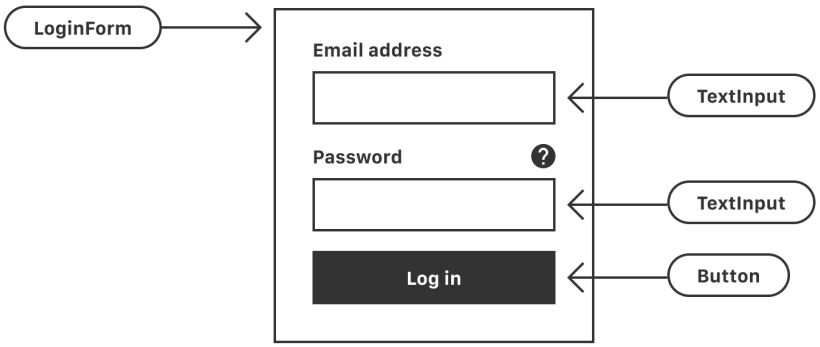
Component composition

Introduction

UDS is a modular system by design, and one of the core architecture design patterns that enables this is component composition.

The idea of component composition is relatively simple: build more complex, more use case-specific components out of smaller, simpler, more generic components. For example we might *compose* a login form out of a pair of text inputs, and a button.



In this case, the simpler components (text inputs and buttons) can be reused across many composite components. For example we could build an email capture form using the same basic building blocks. This pattern helps avoid complex components being overloaded with additional props to trigger new behaviours—instead we can just compose a new component for a new use case.

It's worth emphasising that this is not something novel introduced by UDS, in fact it is advocated in the official react docs. The most well-known framing of the composition pattern for UI components comes from atomic design. You might also like to read Subcomponents by Nathan Curtis

> Implementers own their own destiny, and our systems exist far more to equip and accelerate than control or prevent that destiny.
>
> **Nathan Curtis**

Composition with UDS

A core part of the UDS toolkit is the UDS Base component library. These components are atomic in the sense of atomic design, and form the foundation or *base* for composing all UI components built in UDS. Because these base components are built using React Native and are themable, they can be used as user interface building blocks for all brands and all platforms.

This composition pattern can also be used to create component libraries within UDS. For example, TELUS web components, leverage composition of UDS Base components to create brand-specific user interfaces. The TELUS `Callout` component is composed of `Typography` and `Icon` components from UDS Base. In this way we can create families of component libraries sharing UI primitives, but composed to meet a specific domain's functional needs and expectations.

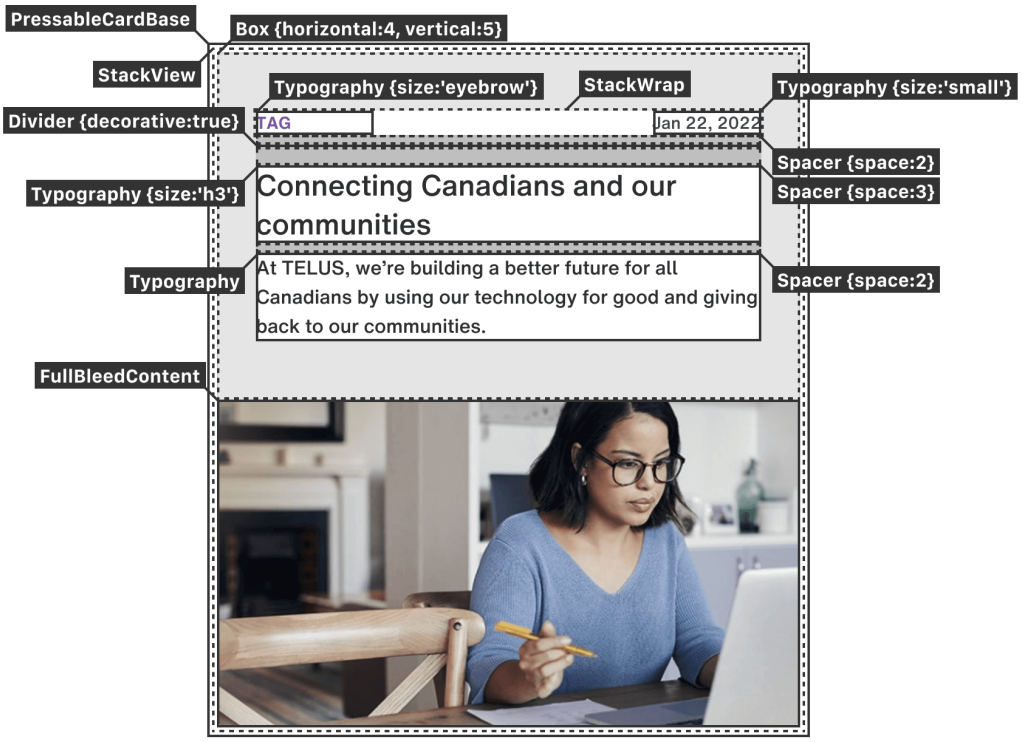
Component libraries created from UDS Base components in this way can be:

- specific to a brand, or cross-brand
- web-specific or multi-platform

It's always good to think about component composition when creating components. Designing simple, reusable building block components allows you to create many complex, composed components cheaply. As components become larger and more specific, they naturally become less reusable. But that's OK! Sharing the atomic building blocks means teams can compose domain-specific UIs quickly with UI consistency baked in.

A practical example

In this example we show how the `StoryCard` component (Allium web-only) has been composed out of UDS Base components. You can see the StoryCard component in the Allium docs and find the source code on GitHub.



Design

It's worth noting that this method of composing complex components out of smaller simpler components is also considered good practice for designers using Figma. For details, see composing components in Figma.

Edit this page