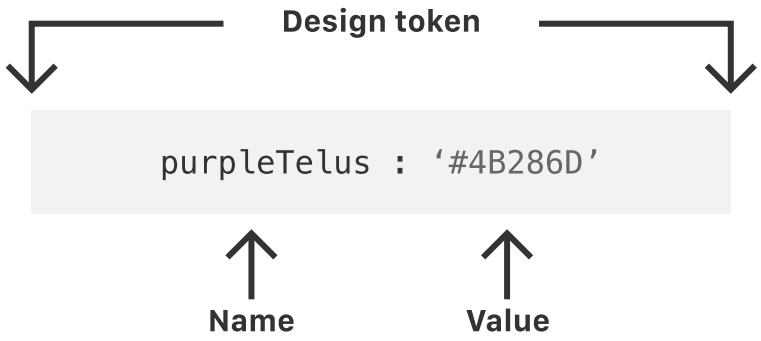
Design tokens



A design token is nothing more than a name and a value that together communicate a design choice (for example, a colour name paired with an RGB colour value). Any design choice that you can represent as a concrete value can be codified as a design token. In relation to atomic design, design tokens are subatomic: they represent even smaller design choices than atoms.

Importantly, design tokens are a source of truth: they represent distilled design thinking in an unambiguous way that is both machine- and human-readable.

The concept of design tokens is relatively new and there's still much development in this area. The great thing about design tokens is they're building bridges and forming a common language between designers and developers.

Design specification

One of the most fundamental ways we can decouple monolithic design systems (like legacy TDS) is to abstract the visual design decisions from the component implementation. That is, we want to separate what something does from how it looks.

Of course, these two parts of the design process are intrinsically linked and certain visual design decisions will need to be made from the outset; however, there are benefits in considering aesthetics independently of function.

- **Reducing "bikeshedding" before initial release.** Striving for perfection is of course important, but not before you've learnt whether this component performs as it should.

- **Continuous improvement.** Styling updates will always be required in the future. They could be subtle refinements or potentially more fundamental brand updates.
- **Reducing the reliance on developers to make changes on behalf of designers.** With a little knowledge about how to make a commit, there's no reason why designers couldn't make changes for themselves. This becomes even easier if tooling is created that gives designers a UI to make the adjustments. Automated processes could bring designers into the developer's world by generating branches, saving changes, running tests, and creating pull requests (PRs) for developer review.
- **Brand-agnostic components allow for themeablity.** Brand-agnostic means making component code more reusable in a wider variety of contexts; for example, with rebrands, experimental brands, acquired sub-brands, or partner brands. Theming also allows for the same components to be visually tweaked based on different contexts (for example, light and dark mode), or maybe different views to accommodate not just multiple screen sizes, but also viewing distances from the screen.
- **Platform-agnostic components keep more teams happy.** The final benefit is most significant for omnichannel projects. Abstracting design decisions for component implementation allows us to reuse and re-apply our visual specification in any software language or platform. Updates to design tokens in one place can be propagated everywhere.

The main takeaway? Design tokens, if organized and managed appropriately, can enable all of the powerful features listed above.

Design tokens in UDS

Although all design tokens represent design thinking, it's useful to distinguish between *design options* and *design decisions*.

Design options

Design options define the constraints for a brand experience. They set out all the valid options that a designer creating a brand experience can choose from when creating that experience. These options could include spacing, colour, or type settings. There's no implication of how these values might end up being used. An example of a design option might be "include San Felix Green in the brand colour palette":

```
color.greenSanFelix: '#1F5C09'
```

UDS encodes and versions these design options for a given brand into something called a brand palette. See the palettes guide for a deeper dive into working with palettes.

Design decisions

Design decisions represent an active application of a value from the set of design options onto a UI or UIs. An example of a design decision would be “make button borders San Felix Green, or:

```
button.borderColor: color.greenSanFelix
```

UDS encodes and versions these design decisions into something called a theme. See the themes guide for a deeper dive into working with themes.

Decoupling with design tokens

Design tokens allow us to separate design specification from the component implementations. This separation allows for the design specification part of the codebase to be distributed across multiple platforms. While the majority of TELUS outcome teams use React, many other technologies are also being used from Angular to Drupal, Swift, and Kotlin. Design tokens can be used by all teams on all platforms.

Adding semantic version control to collections of design tokens enables teams to stay in sync. Brand teams are able to iterate on design tokens, and downstream consumers can pull in those new changes safely, enabling continuous improvement of design.

Edit this page