Government of Canada | Gouvernement du Canada

Canada

Design System Project Team

Sprint # 3 Review

November 1-16, 2021

Sprint goals

1. Set up the environment for the Alpha documentation site
2. Create proof of concept for a tech agnostic design system – rules and limits
3. Create proof of concept for design tokens
4. Draft research-design workflow: How do we collaborate on creating research questions?
5. User needs identification exercise

Alpha documentation site environment

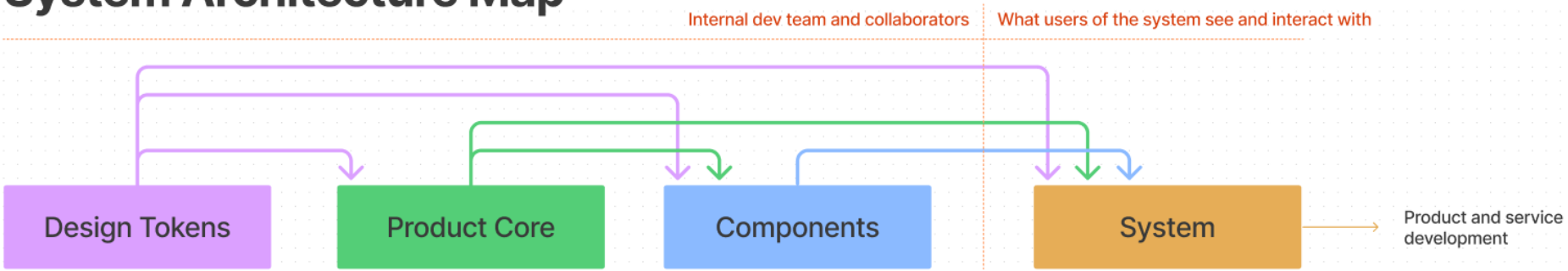
Experimental alpha site made with **11ty** that uses **NetlifyCMS** for content publishing.

- Not the final documentation environment
- Will allow the team a place for experimentation and testing proposed solutions

https://github.com/cds-snc/alpha-design-system-documentation

Architecture draft proposal

System Architecture Map



- Modular pieces with distinct purposes
- Each designed to be swapped/reworked without impacting the others to reduce future debt
- Design tokens set a foundation shared across the system and future products
- Product core contains styling framework
  (discussions ongoing as to creating one or using 3rd party)
- Components that are designed to leverage tokens and styling framework and be used together or on their own

Component architecture discussion

Ongoing work to separate out pieces/structure of components and define:

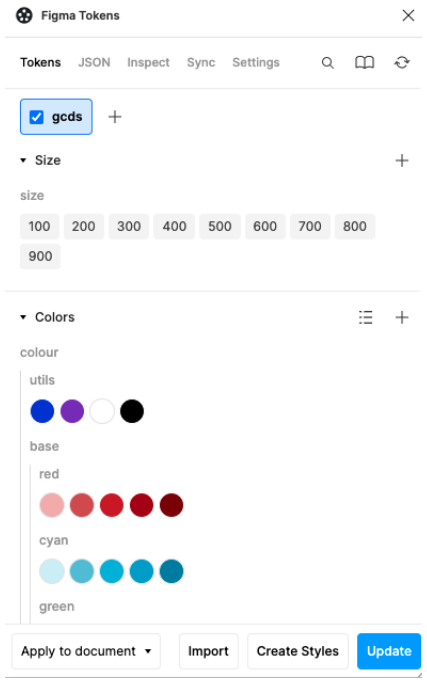
- HTML
- CSS
- Data layer
- Java script interactions

This sets us up to facilitate **interoperability** with other frameworks so teams can **integrate** and maintain components more easily in their **distinct environments** (ReactJS, Angular, ViewJS, etc.)

Proof of concept: Design tokens

Figma Tokens is an open source plugin required to use design tokens in Figma.

- Figma Tokens can sync design tokens with GitHub
- There are a few limitations (but improvements are coming):
  - synchronisation can cause unwanted overrides
  - lack of support for bilingualism



Proof of concept: Design tokens

Style Dictionary is an open source tool that lets **developers** generate tokens programmatically and/or manually:

- Some foundational tokens could rely on a pre-programmed scale or formula to generate extra tokens
- There could be ways to create formulas with some accessibility rules baked in

tokens > color > JS base.js > ...

```
const Colour = require( "tinycolor2" );

const palette = {
    "red":       { h: 1,   s: 75, v: 75 }, // #BF3230 hsv(1, 75%, 75%)
    "cyan":      { h: 194, s: 70, v: 85 }, // #41B6D9 hsv(194, 70%, 85%)
    "green":     { h: 102, s: 80, v: 70 }, // #4EB224 hsv(102, 80%, 70%)
    "yellow":    { h: 35,  s: 65, v: 90 }, // #E5A750 hsv(35, 65%, 90%)
    "grey":      { h: 0,   s: 0,  v: 40 }, // #666666 hsv(0, 0%, 40%)
    "blue_grey": { h: 212, s: 30, v: 50 }, // #596B80 hsv(212, 30%, 50%)
}

// Use a reduce function to take the array of keys in palette
// and map them to an object with the same keys.
module.exports = {
    "colour": {
        "utils": {
            "blue":   { "value": "#0535D2", "type": "color", "description": "Used for hovered links" },
            "purple": { "value": "#7834BC", "type": "color", "description": "Used for visited links" },
            "white":  { "value": "#ffffff", "type": "color", "description": "Used for regular backgrounds" },
            "black":  { "value": "#000000", "type": "color", "description": "Use as the default text color" }
        },
        "base": Object.keys( palette ).reduce( ( ret, colour ) => {
            return Object.assign( {}, ret, {
                [ colour ]: {
                // generate the shades/tints for each colour
                    "100": { "value": Colour( palette[ colour ] ).lighten( 35 ).desaturate( 1 ).toString( "hex6" ), "type": "color", "description": "35% lighter and 1% desaturated" },
                    "300": { "value": Colour( palette[ colour ] ).lighten( 10 ).desaturate( 10 ).toString( "hex6" ), "type": "color", "description": "10% lighter and 10% desaturated" },
                    "500": { "value": Colour( palette[ colour ] ).toString( "hex6" ), "type": "color", "description": "Regualar tint/hue" },
                    "700": { "value": Colour( palette[ colour ] ).darken( 10 ).saturate( 10 ).toString( "hex6" ), "type": "color", "description": "10% darker and 10% saturated" },
                    "900": { "value": Colour( palette[ colour ] ).darken( 20 ).saturate( 20 ).toString( "hex6" ), "type": "color", "description": "20% darker and 1% saturated" }
                }
            } )
        }, {} ),
    }
};
```

Stakeholder research insights

1. **Communications and IT work in silos in government,** creating two consulting services for supporting the current design system product.
2. **The design system is split into two touch points with two support channels,** causing a split in communication for contribution and consistency within the system.
3. **The IA specification and the current WET codebase aren't flexible enough to fit all the current design needs for the web and Government of Canada,** causing a need for one-on-one support for public servants with specific questions and requests.
4. **There are two different design processes.** This helps create two levels of design outputs: experience, and product level for the design system. However, this causes siloed communication around unified priorities like research, accessibility, bilingualism, and documentation.

… more insights to come. Report available for processes & next steps

User needs identification

Jamboard exercise

Write concrete tasks, describing what users need a GC design system for.

2 sessions, over 100 attendees:

- developers
- designers
- web communication advisors
- others/all users

User needs identification

Next steps:

1. Review and refine user needs
2. Integrate community input in our list
3. Prioritize areas of focus for first ite

Decisions made this sprint

Design tokens

- **Design tokens dictate the visual experience only (styles, typography, spacing), and not the component behaviour.** Design tokens provide a layer of decisions that we don’t want people to do over and over again.
- **We create rules around interaction and behaviour through the components themselves**. Depending on context, we assume implementers will need more flexibility when it comes to component behaviour than design token values.
- Design tokens are not just for the web, they are also for other devices.
- Design tokens offer enough identity constraints to enable developers and designers to create a user experience based on the specific needs of their users.

Priorities

- The Design system is meant to set certain rules. Decisions will be made around where these lines are hard and where they are flexible. We will prioritize **tools** (enablers) **over rules** (constraints).
- The Design system must have the **ability to evolve**. Governance processes will exist to let teams **easily contribute back** when they discover gaps where the Design system doesn't meet their needs.