



Government
of Canada

Gouvernement
du Canada

Canada

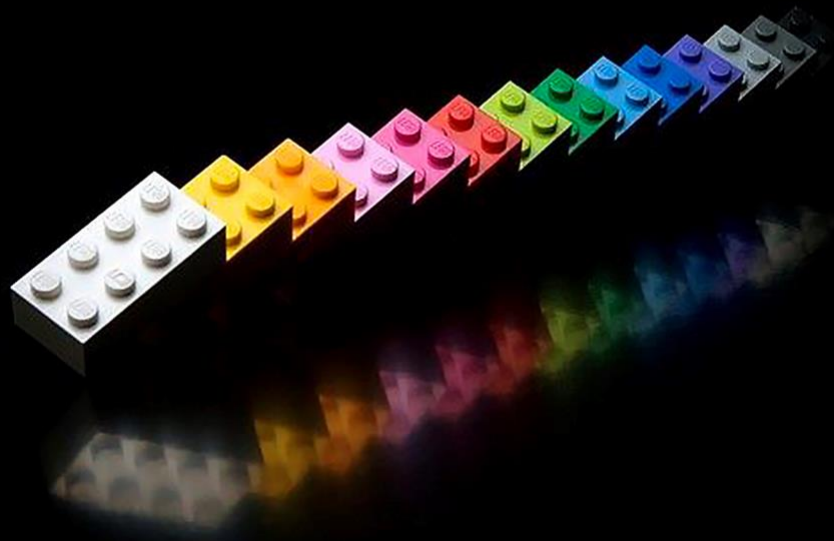
Design system MVP

Use case

[GC Design Slack](#)

dto.btn@tbs-sct.gc.ca

XXXX, 2022

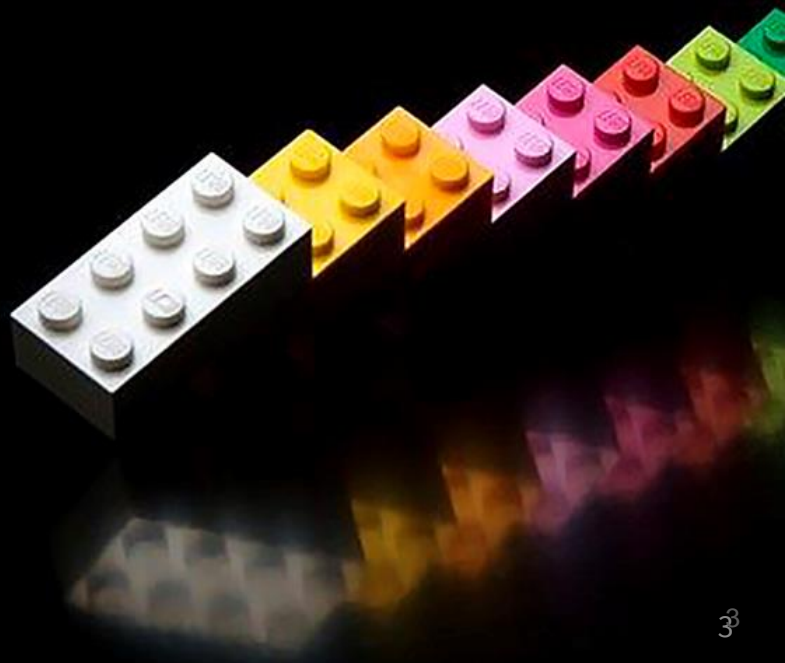


- Defining a design system
- MVP deliverables and approach
- GC design system MVP
 - Easy and fast installation and creation (npm)
 - Documentation that integrates guidance on code, design, content, style and policy elements together in one place
 - Specific accessibility guidance (including for cognitive needs)
 - Design tokens
- Next steps

What's a design system?

A design system is a collection of **flexible and reusable designs, code elements, and guidance supported by research.**

It gives teams the building blocks they need to design their online presence in a way that supports task success and trust among users.

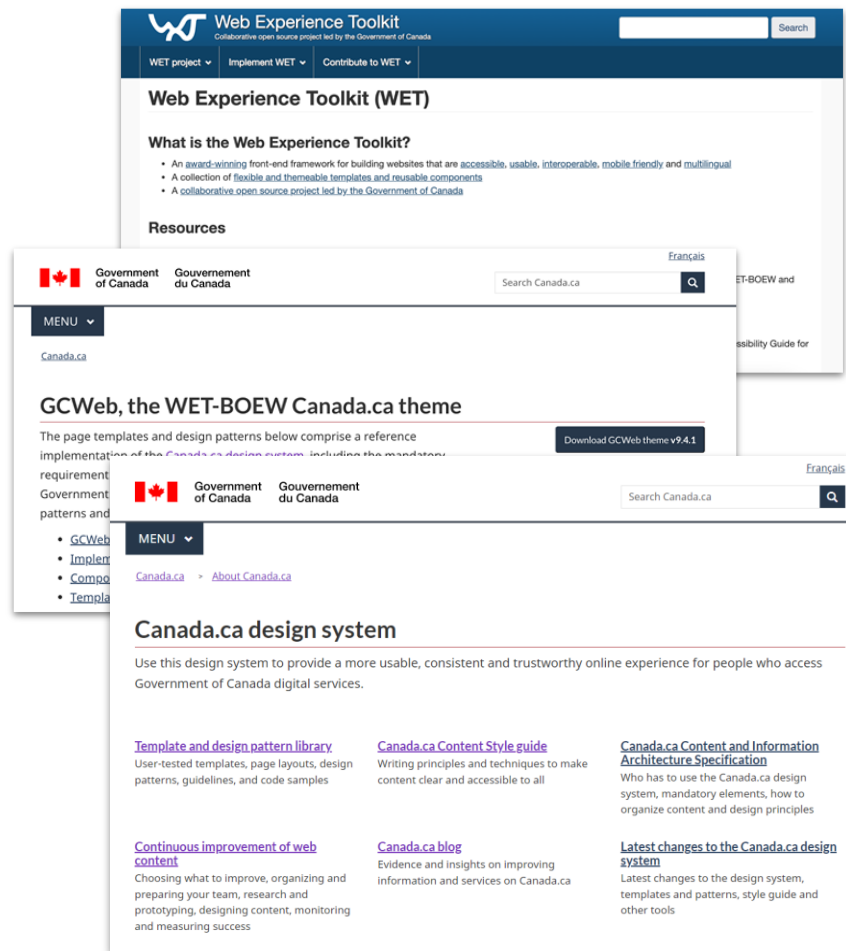


GC design system

An accessible, consistent experience is predictable. It makes people feel safer. It builds trust.

Investing in a design system creates **integrated guidance** using **modern, consistent code**. It means we can:

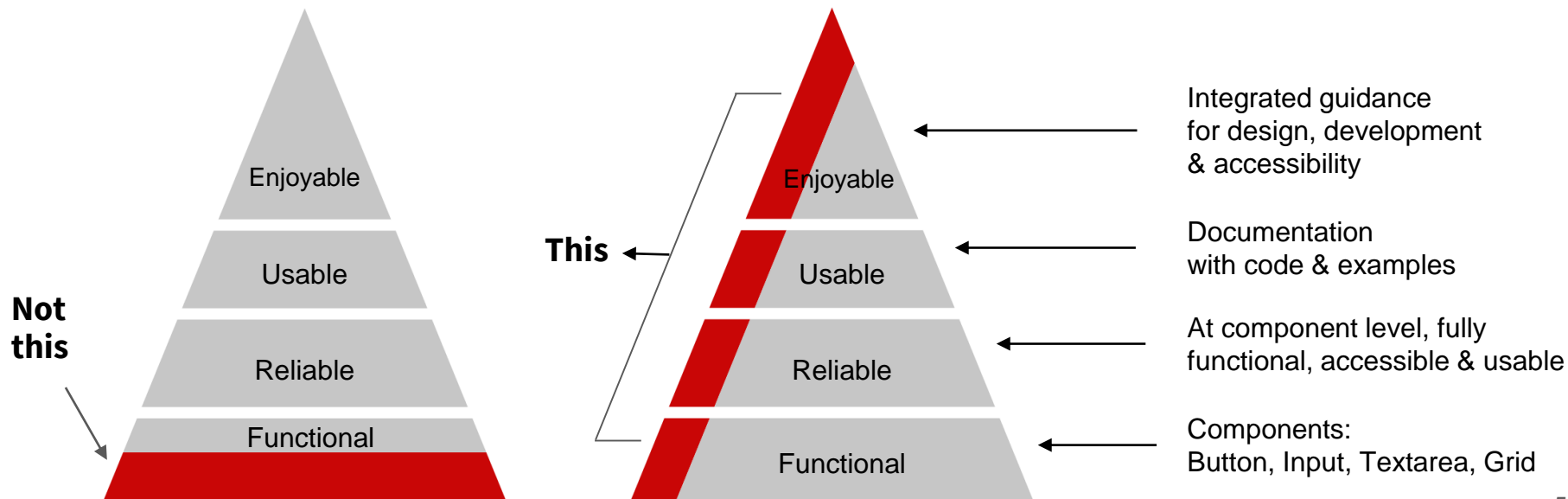
- create new services without creating tech/design debt
- speed up design and development time to respond more efficiently to the evolving needs of the public
- improve accessibility



MVP Approach

Our approach to developing the MVP targeted two goals:

1. Provide Value Sooner
2. Enable Feedback For Next Iteration



MVP deliverables



Technical

- Basic Token system
- Basic UI components
 - Front-End tech agnostic
- Code system designed for modularity

Documentation

<https://design.alpha.canada.ca/en/>

- Foundations
 - Tokens - typography
- Component pages
 - Code & advice
- Guidance:
 - For developers & designers
 - For accessibility & usability

Foundations

Installation

Benefits of npm for installation

- Ease of use for publisher and developer
 - Free
 - Open source
 - No registration or login required
- Single commands to publish or download

Currently supported frameworks

- JavaScript
- React
- Vue

Installation

Install from npm

```
npm install gcds-components
```

Supported frameworks

The gcds-component library works in multiple frameworks.

JavaScript

Place the following code in the <head> element of your site.

```
<script type="module">
  import { defineCustomElements } from '/node_modules/gcds-components/loader/i
  defineCustomElements();
</script>
<link rel="stylesheet" href="/node_modules/gcds-components/dist/gcds/gcds.css">
```

All gcds-components should now be ready to use in your site.

React

Place the following code in the index.js file of your app.

```
import { applyPolyfills, defineCustomElements } from 'gcds-components/loader';
import 'gcds-components/dist/gcds/gcds.css';

ReactDOM.render(...);
```


Foundations



Settings for foundational elements that create the overall brand

Structure:

- What it is, why it's useful
- What it looks like in **code** and the **semantic hierarchy** (if applicable)
- Detailed **specifications**
- Guidance on **how to use** the element and **how to customize** it in unique situations

Typography tokens

Typography is the presentation of text. It includes fonts, sizing, and spacing as applied to the style, arrangement, and appearance of letters, numbers, and symbols.

 [GitHub](#)  Figma — coming soon

Typography design tokens

Default fonts, and styled size settings make digital products responsive, predictable, and readable across devices and platforms.

Decisions about typography values (fonts, sizes, weights) are built into typography tokens in the GC Design System for a unified design.

Font defaults

Font family tokens:

- \$gcds-font-families-heading
- \$gcds-font-families-body
- \$gcds-font-families-monospace
- \$gcds-font-families-icons

Font sizes are configured with rem units. All font sizes are relative to base size 20px.

Display	Font type	Size (rem)	Size (px)	Weight	Line height
Heading 1	Lato	2.5	40	bold	128%
Heading 2	Lato	2.25	36	bold	144%
Heading 3	Lato	2.0	32	bold	162%

Components

Component pages

- Improving communication between communicators, designers and developers
- Quick access to supporting developer and designer tools
- Experimental local navigation pattern
- Integrated accessibility guidance and considerations
- Distinct sections designed for distinct audiences:
 - General
 - Developers
 - Designers
 - Learners

Component at a glance

“Sticky” On-this-page **local navigation** experiment lets you scan for what you need

On this page

[The basic button component](#)

[How to modify the button component](#)

[What problems buttons solve](#)

[Related components](#)



[Where to place a button](#)

[How to write a good button label](#)

Button

`<gcds-button>`

A button is an interactive object that highlights an important or common action for the person using your product.

 [GitHub](#)  Figma — coming soon

HTML tag promotes **common language** between technical and non-technical users

Access to complementary, supporting tool set encourages **interdisciplinary collaboration**

General audience content

- Brief description of:
 - what it is
 - what it's for
- Visual example (for each type if applicable)
- “Use for” and “Avoid” guidance
- View and copy code options

The basic button component

Use buttons for user-led actions.

Buttons have built in states that change when someone interacts with that button: default, hover, focus, active, disabled.

Primary button

A dark blue rectangular button with the word "Primary" in white text, set against a light gray background.

Use a primary button for the most important action.

Use it for critical actions in a flow or as the default button.

Avoid more than one main call to action on the same page. It reduces impact and makes it harder for a person to know what to do next.

A light gray rectangular button with rounded corners and a thin border, containing the text "View code".A light gray rectangular button with rounded corners and a thin border, containing the text "Copy code".

Secondary button

A dark gray rectangular button with the word "Secondary" in white text, set against a light gray background.

Use a secondary button for supporting actions.

Developer audience

- Provides developers the details required to **adjust components** in **exceptional circumstances**, to meet edge-case context
- Includes the **hierarchy** of attributes and properties
- Links to **Foundations** section for typography, colour and spacing specifications

How to modify the button component

Occasionally, the default options may not meet the needs of the people using your product. You can modify the button component to create a custom button for a particular use case.

Button component type, style, and role

Button type

Set the button type based on how the button will act.

The button-type attribute, accepts the following options:

- button
- submit
- reset
- link

Button role

The buttonRole property, button-role attribute, accepts the following options:

- destructive
- primary
- secondary
- skip-to-content

Learner audience

For people who:

- are new to the component
- use the component infrequently
- want to learn more about other ways to use the component
- want to explore other options

What problems buttons solve

Use a button for important user-led actions like:

- saving, deleting, copying, or downloading
- clearing entered data
- sending a form or request
- giving consent or agreement
- making a binary choice
- entering or exiting a secure area login, a form, or tutorial
- skip-to-content link

In apps, buttons replace text links for things like:

- controlling movement, like going from one screen to another in a flow
- external links (in a way that lets someone know they're external)

Related components

Radio buttons or dropdowns when you can give someone a set of options for single selections.

Checkboxes when you can give someone a set of options for multi selections.

Designer audience

Advice related to:

- interaction design
- content design
- accessibility

Where to place a button

Place buttons in a regular and predictable way to help a person find the thing they want to do and take action.

Avoid making a person needlessly scroll, tab, or type by providing the path to the next action early.

Check the button border's visibility against the surface where you're placing it.

Adding skip-to-content buttons

Let a person skip a cluster of navigation links with the skip-to-content button.

To avoid covering up content, configure the button to push down content so it's not floating. For desktop, place skip-to-content button at top left of the page so it doesn't interrupt the flow.

How to write a good button label

Make buttons readable

Keep label text visible by stating the button's specific action in minimal words.

Use sentence case.

Forms guidance



General guidance with a service design lens

- **Focused on designers** of forms
- Explains **how** to make **components work together**
- Includes **view** and **copy code** links
- Offers **basic direction for developers** who are asked to design forms

Provide a clear path to completion

Only select the form fields you need

Some questions to ask when you're deciding which fields to include on a form:

- Who needs the data?
- What are they using it for?
- Is it essential or just nice to have?

Not a required field? Just remove it altogether. It's probably not that important.

Choose form elements that match your use case

Choose form components that fit the use case for your data needs. Prioritize the user experience.

Consider the mental models of someone completing your form. They might be familiar with answering questions a certain way. Try to do what would be most comfortable for them.

Form component pages

Guidance on the order of fields and messages, as well as best practices, based on research

- **Templated form fields** help ensure accessibility
- **Tips** for writing good labels & error messages
- **Component code** includes default properties for quick, standardized creation

On this page

[The basic text area component](#)

[How to modify the text area](#)

[What problems text areas solve](#)

[Related components](#)

[Size and placement of text area](#)

[How to write good text area labels](#)

[When to use hint text and error messages](#)

Text area

<gcds-textarea>

A text area is a space, usually on a form, for a person to enter long-form information in response to a question or instruction.

 [GitHub](#)  Figma — coming soon


The basic text area component

Use a text area to collect multi-line information from a person.

Text areas have built in states that change when someone interacts with that field: default, hover, focus, active, disabled.

Text area

Text area



Use the text area for longform, multi-line, response collection.

[View code](#)

[Copy code](#)

Under the hood

Design tokens



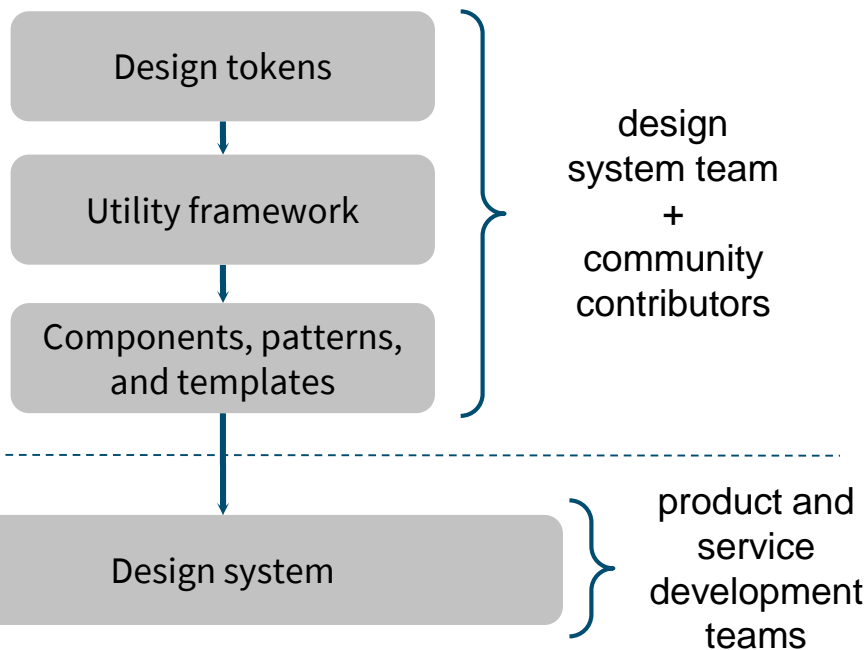
Design tokens are a way of creating **small pieces of reusable code** that centralize decisions about the visual experience – e.g., styles, typography, spacing.

They **codify design decisions** so people don't have to solve common design problems like in-component colour contrast or spacing over and over again.

This improves:

- Consistency
- Resilience (future proofing)
- Connection between humans and code
- Flexibility

GC design system



Product core (utility framework)

consumes the design tokens and generates a style framework

Components are re-usable elements containing tokens that can be used to build interfaces

Design system gives teams all the guidance, design elements, and pre-coded pieces they need to build a cohesive user experience

Next step: Testing

MVP research

Small teams outside AEM

- Compare the ease of use with WET and Canada.ca design system
 - Are we improving anything?
 - Are we solving identified user needs?
 - What is the implementation feasibility?
- Component guidance
 - Too much, too little?
 - Clear and complete?

Tasks

Tasks to test with developers and designers

All

- How easy is it for users to find, understand and use the documentation site?
- Do they understand what can be customized?

Developers

- Install npm package

Designers

- Understand form elements
- Create a basic form

GC Design System Product Team

[GC Design Slack](#) • dto.btn@tbs-sct.gc.ca