

## **GUIDE:**

# How to integrate your design system with your CMS



# Table of Contents

<b>The essential parts of a design system</b>	<b>3</b>
<b>The design system is a part of your build process</b>	<b>4</b>
<b>How to couple the CMS to the design system</b>	<b>5</b>
<b>The integration points</b>	<b>6</b>
Header, footer, menu	
Layouts	
Page templates	
Page components	
Forms	
<b>Front-end frameworks and performance</b>	<b>8</b>
<b>Bonus from Gjensidige</b>	<b>9</b>

# HOW TO INTEGRATE YOUR DESIGN SYSTEM WITH YOUR CMS

Learn how you can integrate your CMS with your design system, including your build process, integration points, and front-end vs. back-end rendering of UI components.

## THE ESSENTIAL PARTS OF A DESIGN SYSTEM

A design system brings together the visual and functional elements of your organisation in one place. With everything from pattern libraries and style guides together under one roof, you can keep your brand and communications consistent, no matter how varied your digital experiences.

The shape of your design system will vary depending on your digital experiences, communications and team, but it will most likely include a:

- Sketch library
- Pattern library
- Style guide
- Organisation principles
- Templates
- Best practices
- UI Components delivered as a code piece (i.e HTML, CSS and other front-end frameworks)

# THE DESIGN SYSTEM IS A PART OF YOUR BUILD PROCESS

Different design systems offer different options, with some you need to import all the code into your project, and others enable you to include the parts you need. The latter option is the best when it comes to creating a slimline system that isn't bloated with unnecessary code.

We won't dig into the nitty-gritty in this guide, but popular tools to look out for when focusing on front-end development include Webpack, LESS, Sass, Babel, and Nodejs.



# HOW TO COUPLE THE CMS TO THE DESIGN SYSTEM

The link between a CMS and a design system is the HTML. This means that to couple your CMS with your design system you must adapt HTML in the CMS templating where it is needed.

So in which scenarios does the CMS need to be flexible? Let's work through some examples.

- Your design system supports multiple columns and responsiveness – how can this be adapted to your CMS?
- Your design system supports optional use of padding – how can this be configured in your CMS without hardcoding?
- Your design system may handle advanced interaction, i.e. form validation – how can the CMS form builder adapt to this without changing the design system?
- Server vs. client rendering – does the CMS support fluid transitions for a fast initial response and rich user experience on the client side?

# THE INTEGRATION POINTS

## Header, footer, menu

The header, footer and menu make up the shell of your site and must be integrated to fit with the design system. Once these are customised for your CMS, you will be able to dynamically build the menu content and specific parts of the footer and header.

## Layouts

Layouts are used to organise the structure of the various component parts on a page. They are configurable, which means you can adjust padding, background, number of columns, and distribution (like 70/30). It is great if your CMS has support for layouts, but if not you will have to hardcode and make more page templates. Just bear in mind that hardcoding will limit flexibility for web editors.



## Page templates

Your CMS might be where you code and manage your page templates, in which case, you will get even more flexibility if the CMS has dynamic page templates that can be configured by web editors. And don't forget that templates can be combined with layouts to produce different results without having to code.

## Page components

Made up of elements and widgets, page components can be inserted into your layout or page templates. These might be helpful apps like a calculator, article list or calendar. It's these components that go deep into the design system, connecting the UI with functionality in the form of buttons, animation, or interactive elements. Most of your design system integration will come in the form of these page components. The page components might be built using React or other front-end frameworks.

## Forms

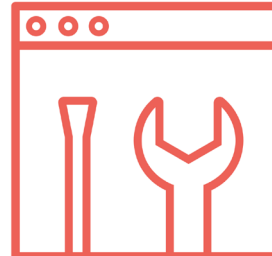
To make online forms work well with your design system, you need a flexible form builder that allows for customisation of the HTML output. Plus, to make sure they fit the design system, the component itself might have to be adapted. Remember that error handling, form layout, and validation are essential to the form too.

# FRONT-END FRAMEWORKS AND PERFORMANCE

Finally, you should consider the alternative between front-end rendering vs. back-end rendering in relation to your design system. Popular front-end frameworks include React, Angular, and Vue, and which one you should choose depends on the technology used in your design system.

Today more and more of the experience is delivered in the browser, which can often have a negative impact on performance - especially initial loading. You must decide if all functionality will be placed in the front-end or if you should opt in for a mixture where you deliver back-end rendered markup and handover to front-end frameworks after loading. The latter will give better response time for users.

It is possible to pull this off if your CMS supports server-side Javascript, then you will be able to generate markup servers-side, using for instance React, and pass this to the client for a smooth handover.



We mention this because relying on pure front-end delivery will affect performance and search engine indexing, which in turn will impact [SEO](#) scoring and accessibility.



# BONUS

## 5 quick tips for building a design system from Gjensidige

Gjensidige is a Norwegian mutual insurance company that has built a dedicated design system for their digital channels. We asked digital editor in chief, Torstein Aas-Hansen, to share the company's experiences and best practices for building a design system, and here are their five essential quick tips:

1. Put time and effort into working with the design system, and collaborate across functions and departments.
2. Find inspiration from others, but make a design system that suits you.
3. Start modestly, with a design system that is easy to build and can be used by many people in different settings.
4. Create a roadmap, telling you where you want to be in six months, and what you have to do to get there.
5. Ensure a multidisciplinary ownership to the design system in the entire design and developer environment.

# FREE DEMO OF ENONIC XP

Get a free demo and  
test Enonic in the cloud.

[REQUEST DEMO](#)