# CMS • A BRIEF INTRODUCTION



Using a CMS (content management system) for our web projects will take our sites to a whole new level and will allow us to... well, manage our content. This is not to say that all web projects will have to make use of a CMS ~ it is about finding the best solution for our project's mission. There are many options out there - the courageous might build their own :D But for now, let's talk CMS.

## When to use a CMS

### consider all angles

The move to working with a CMS is a challenge and you might feel a little anxious. But don't worry, take your time and consider your options and ambitions, and as you know ~ we're here to help so talk to us :)

Before you set out to build any site, be that for a client or for your MP - it will be important to answer certain questions: not only which CMS is best but whether a CMS is the right direction in the first place. Not all scenarios will have a clear demand for a management system though many will do.

### questions to ask for client projects

While a client project scenario does not strictly apply to all our web projects - the following will still serve as pointers to various aspects of our projects.

#### the site admins

? who will maintain the website?

tech support / server / software updates

? what data process is needed?

admin users / GDPR / backups

? how are the technical costs covered?

hosting / license / domain registration

## the content publishers

? who is responsible for content updates?

in-house team / web team

? how will publishing be managed?

schedule / editorial control / promotion

? training for editors

manual / help files / tutorials / in-house

In the case of any web project - my advice would be to look at the whole picture and consider not only the needs of your project but also your own skills and learning journey. This is the perfect project to get stuck into and learn. Which way exactly is what you'll have to decide :)

## A CMS might be best for your project if...

- your content strategy plans for phased updates of content which would benefit from scheduled publishing.
- you are planning to offer location based details, feature maps and offer specific means of contact.
- your site will have multiple users who will contribute in some way to the site's content.
- you are planning to use social media for promotion and to integrate third party content.

## Your project might not call for a CMS if...

- your content strategy plans for a slow cycle of updates. And you fancy the

challenge and work of manually updating content (+ have the time for this).
- the planned amount of content, once published, will not require frequent/ any updates.
- the site has a single purpose and set expiry date.
- your plans for design/interaction call for experimental techniques or approaches which might be best done within a fairly static site.

## A quick word before you start

If you've already worked with any kind of CMS before ~ easy peasy ~ you'll likely know enough to play and experiment and figure out what works best for your project. Even if you've only worked with a CMS as content editor/publisher, you'll likely know quite a bit about how a CMS works from that perspective and what kind of functions your project would need.

But if not, if you're completely new to working with a CMS, especially new from a design/dev angle, then this will be a steep learning curve! — well worth all the work and battles through code but not an easy ride.
**You will break things.** You **will** see the white page of death and all **will fall apart.** But that is part of the journey and how you'll learn — **you will get there in the end :)**

# What is a CMS anyway?

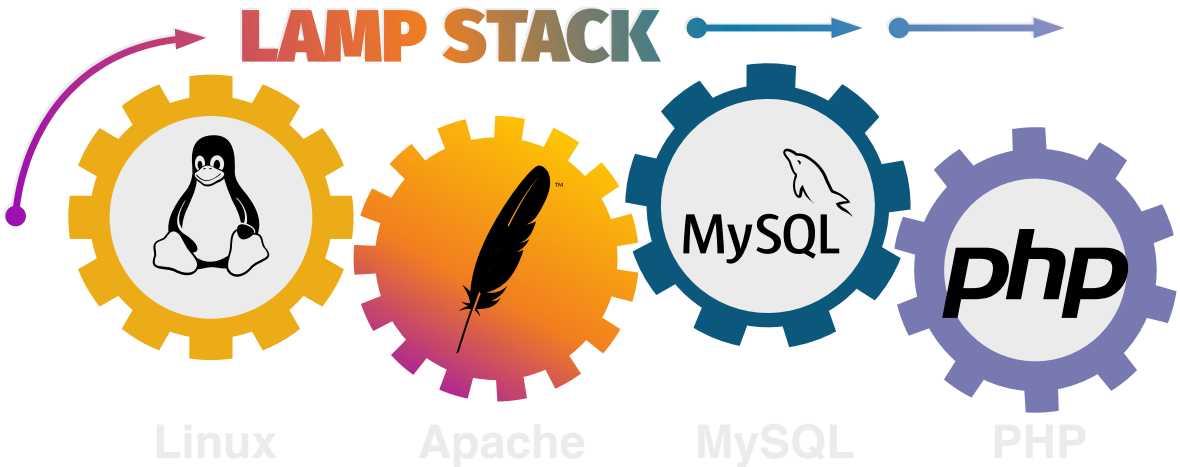
## definition

A CMS is a piece of software that runs on the server to facilitate the creation, editing and overall management of digital media content. It includes an interface for the management of content and the function to deliver this content to the final output. Uses vary from simple scripts for forms etc to bigger solutions for webmail or forums.

The main advantage of using a CMS for a given website is that it will allow the user to modify the online content without the need to code. An interface facilitates creation, modification and deletion of content, updating the database with any changes and delivering the final result to the end user.

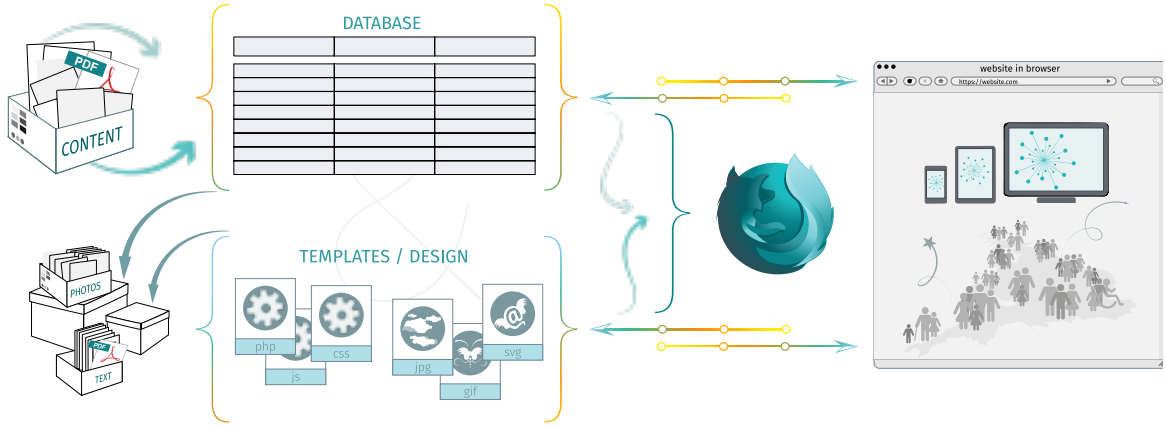
## components

CMSs come in different shapes and sizes and can use different technologies. One of the most common is the **LAMP** stack which is comprised of four open source components: Linux (operating system) + Apache (HTTP Server) + MySQL (database) + PHP (programming language).



# separations

When you started learning to code, you were considering the separation of content (HTML) and design (CSS). Broadly speaking, the CMS follows the same principle but adds the database (content) to the mix and uses templates (design). Of course, there's much more to it, but this is one of the core points.




## database

1. store website content
2. store user details
3. store permissions / access to functions

Instead of the content being directly in the markup, you'll store all details in the database and use the templates to pull in the required content.

## templates

1. markup for all content elements
2. content structure via template parts
3. presentation of content as designed

Templates can be written in various languages and will result in an HTML output which will use the CSS (± JS) for presentation.

# Types of CMS

## What are the options?

Looking into the various flavours of CMS out there, you'll soon realise that there are many, many differences and just as many different opinions. It is a matter of assessing the project, its mission and the final target group and weigh this against the intricacies of the CMS. There is no easy answer and one size does not fit all.

From a practical angle, there are open source options, proprietary solutions as well as SaaS (Software as a Service). Each has their own strengths and advantages, both philosophically and technically.

## Quick note

The links to the various different CMSs listed here are not endorsements per se. I've included links to those most popular or used in the past by our students for their MPs.

### difficulty level

As most of you are new to web design/dev, I thought it'd be helpful to rate the different technical setups by difficulty so you can gauge the level of challenge ahead. Hover the icon to see detail / read text. Hope it's useful :)

## What are the technical differences?

Since their conception, CMSs continue to evolve and there are various technically different systems you should consider. The decision which will fit your project will depend on whether you'll be primarily serving web content to the usual browsers on various devices, or whether your project is more complex and aims to deliver to the IoT (internet of things).

### traditional CMS

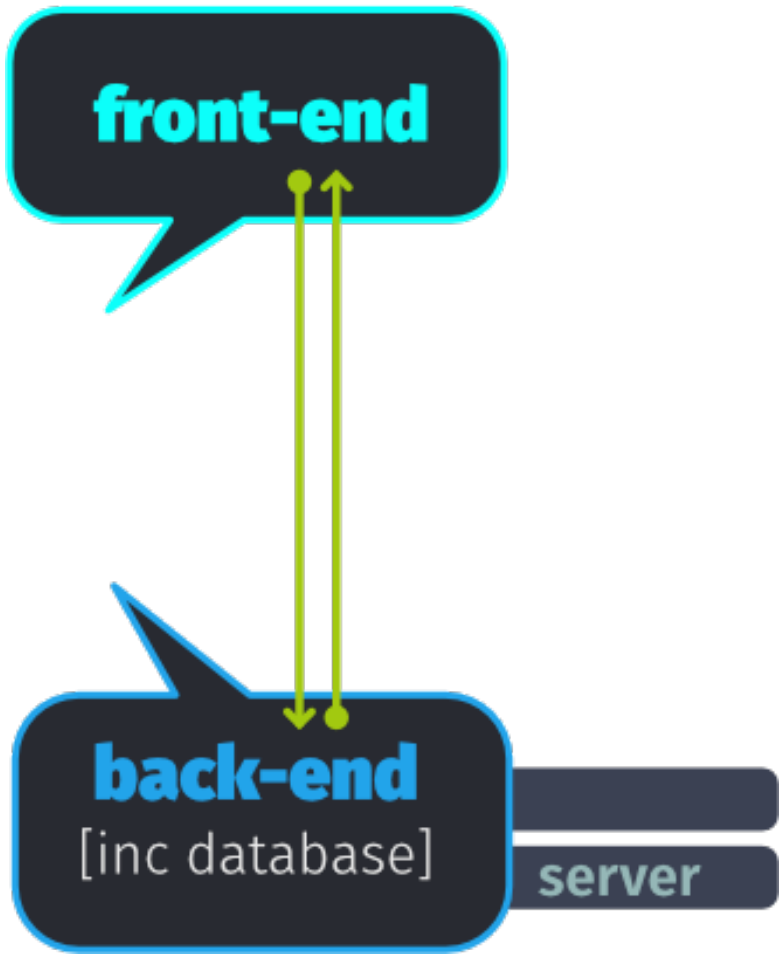
1. **integrated back/front-end**
   DB for content + GUI for managing/publishing content

2. **content stored in DB on server**
   data structure set by CMS, accessed via templates; usually delivered to browsers

3. **what do you need to know?**
   HTML/CSS + additional programming language to edit templates

## Open source options:

- WordPress
- Drupal
- Expression Engine
- Text Pattern
- Joomla
- Typo3



## flat file CMS

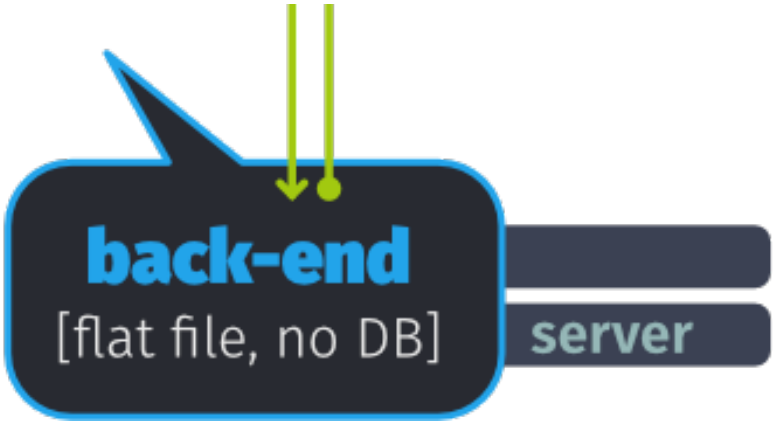
1. **integrated back/front-end**
   flat file for content + GUI for managing/publishing content

2. **content stored in static files on server**
   data structure set by CMS, accessed via templates; usually delivered to browsers

3. **what do you need to know?**
   HTML/CSS + additional programming language to edit templates

### Open source options:

- Typemill
- Grav
- GetSimple
- Kirby
- Flextype
- Monstra




## headless CMS

1. **independent back-end**
   lack of the presentation layer
   (no 'head' = no front-end)

2. **content stored in DB on server**
   data structure designed for purpose, accessed via API for output;
   can deliver to various systems/apps (IoT)

3. **what do you need to know?**
   HTML/CSS + additional programming language to edit templates
   + JavaScript framework

## Open source options:

Strapi

directus

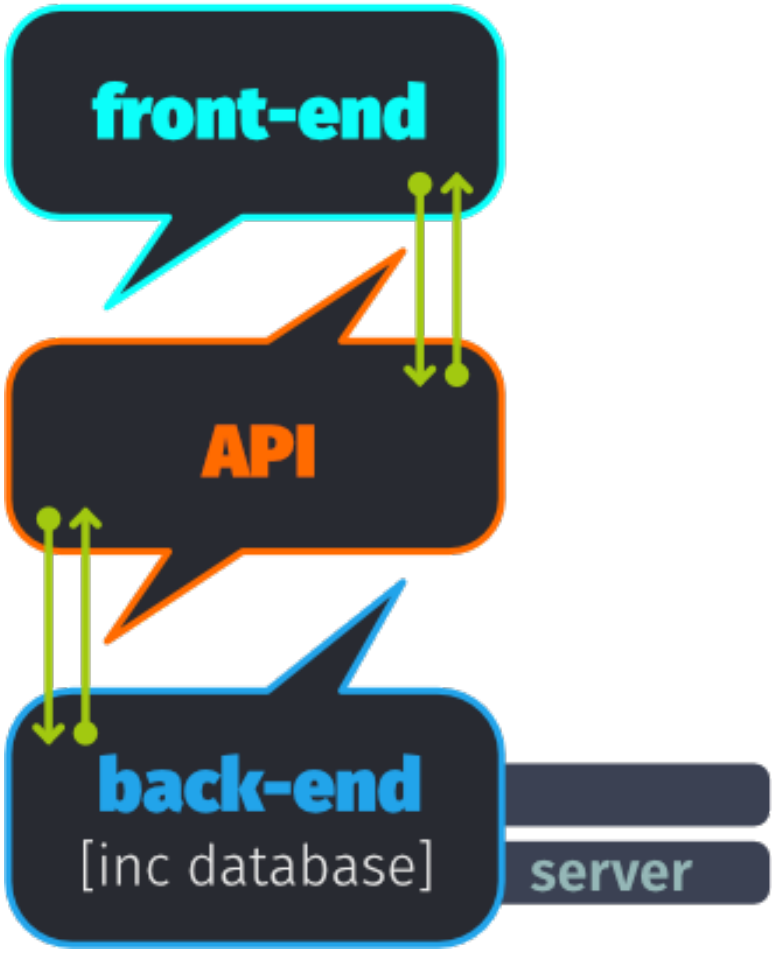
Cockpit

Headless WP + React

## Licensed options:

Perch Craft

Agility Butter




## serverless CMS

1. **independent back-end (no front-end)**
   lack of the presentation layer

2 **content stored in DB on cloud server**
data structure designed for purpose, accessed via API for output

3 **what do you need to know?**
HTML/CSS + additional programming language to edit templates + NodeJS (JavaScript framework)

## Open source options:

Webiny

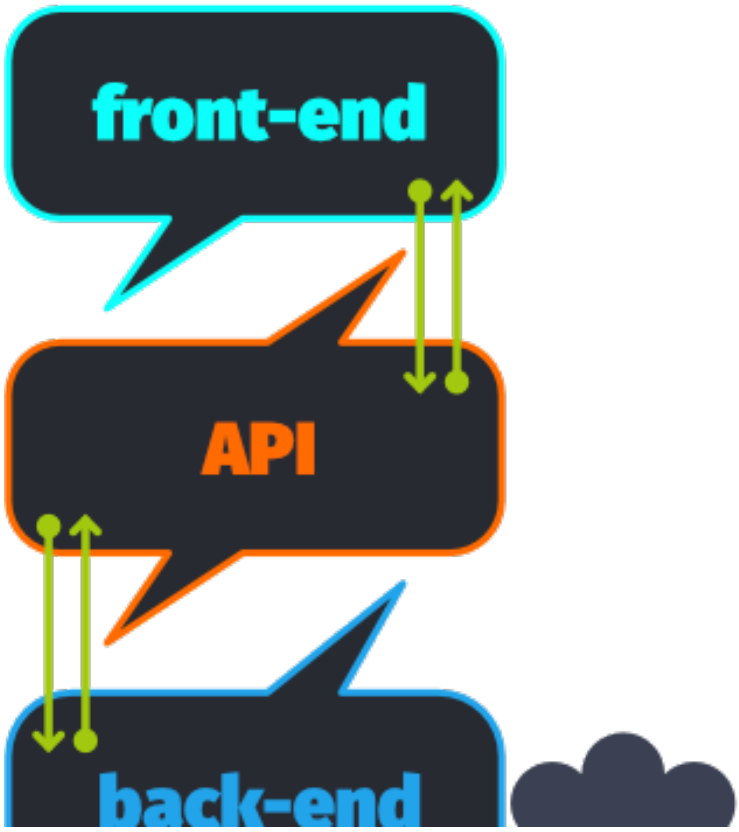
serverless

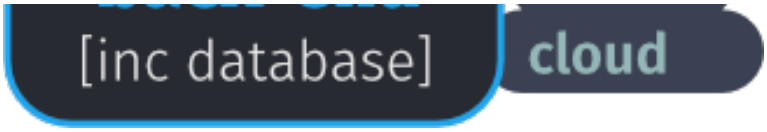
## Frameworks:

NodeJS
(JavaScript runtime)

architect
(build serverless web apps on AWS)

AWS Amplify
(end-to-end AWS solution)






## useful reading

Rachel Andrew: Designing For Content Management Systems

Dom Nicastro: 14 Rules for Selecting the Right Content Management System

webdesignerdepot: How To Choose The Right CMS

Paul Boag: 10 Things To Consider When Choosing The Perfect CMS

w3c: W3C CMS platform selection - full status update