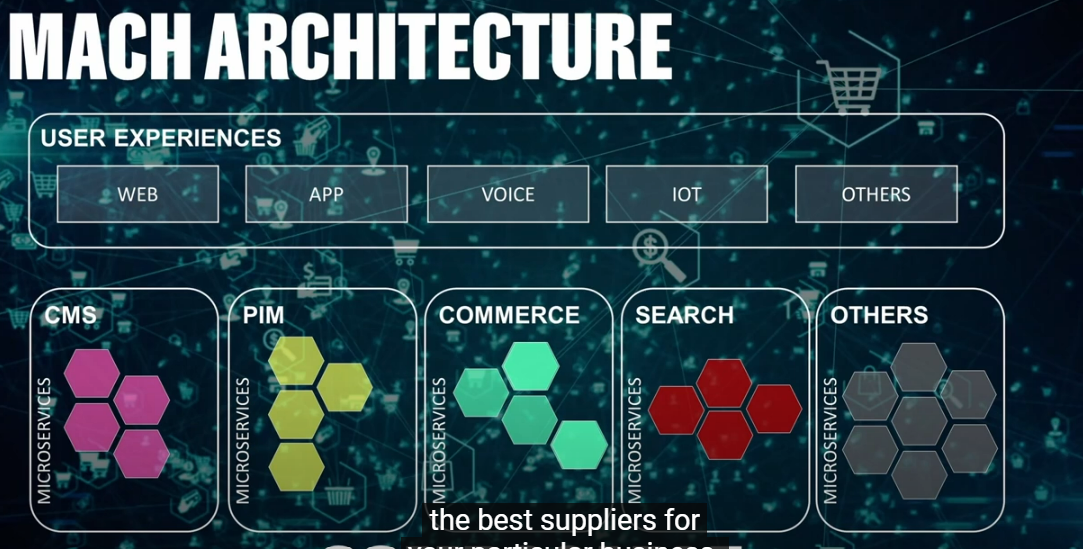
**Why MACH architecture**

The purpose of the monolith is just to create a single channel.

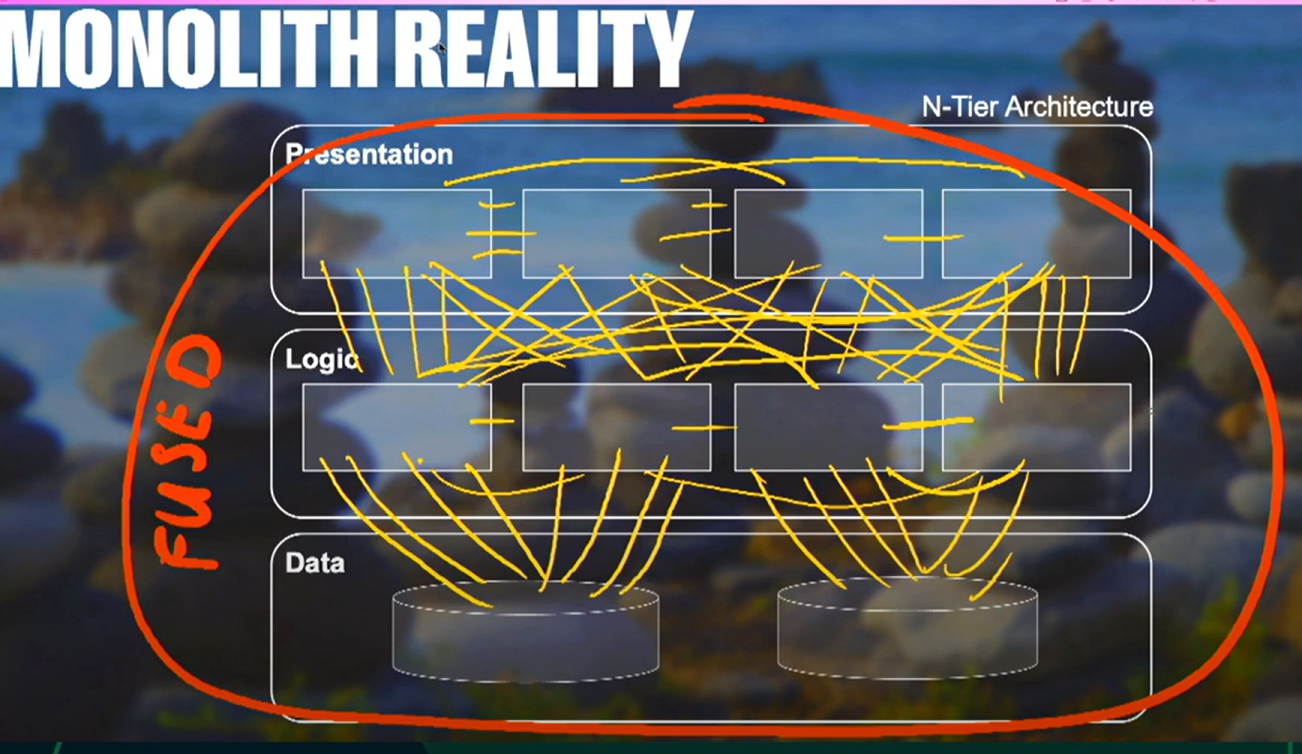
Mach Architecture – it is build for multi-channel or omni channel.



As a case study first go to cloud native, then API first, the go headless (serving content into neutral format irrespective of presentation, delivering it as a API which any technology on UI can rehydrate the way that they want to hydrate it) by content as a service with headless CMS. Headless not just about APIs but complete independence from the presentation, all the way down to the how the data is constructed in the backend, then go to Micro services from Macro/Monolith API (horizontal).

**MACH (Microservices, API-First, Cloud-Native SaaS, and Headless) Matters**

Monolith architecture reality –

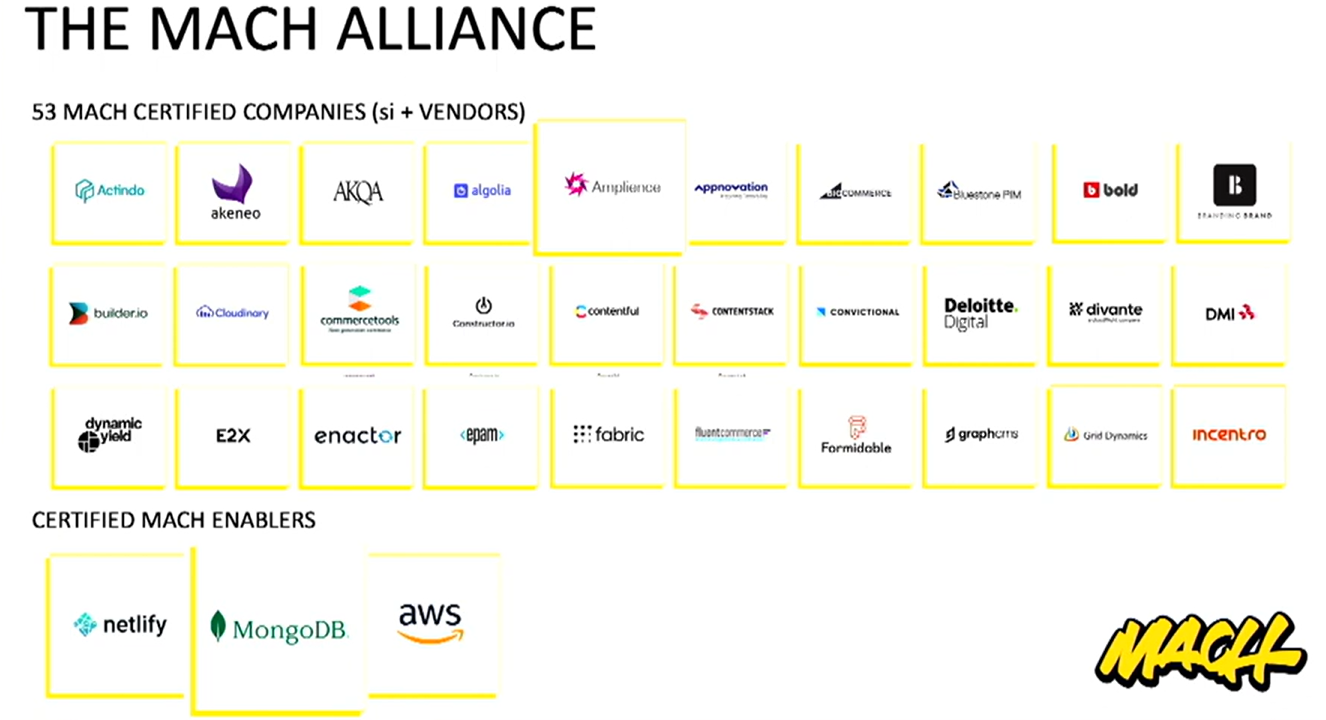


MACH architecture – to move towards more composable architecture.

A screenshot of a computer screen

Description automatically generated

MACH certified companies –



The scalable, interchangeable nature of headless CMS inspires innovation by integrating with new technology seamlessly.

Unfortunately, most widely-adopted content management systems (CMSs) are monolithic—meaning they’re built to cater to traditional websites and assume every consumer will follow the same predetermined pathway. They fall short of delivering the personalized, consistent experiences that modern consumers demand along each zig and zag in their shopping journey.

Different touchpoints for consumer – desktop, email, social, in-store app, mobile, support, kiosk, logistics, iot.

Traditional CMS vs. headless CMS –



This architecture, called “MACH”, is commonly used to build e-commerce sites. It allows you to do “composable commerce”, i.e. build your architecture as if you were assembling legos. We assemble ready-to-use Saas bricks or custom bricks when the needs are too specific. It is then easy to replace a brick if it is depreciated or if the business changes, allowing you to benefit from the latest innovations without having to rewrite the entire application.

Benefits –

Develop your platform faster thanks to high modularity - All of this can be done seamlessly on frontend by being **headless**. When separating the frontend services from the backend services with the business logic, they are decoupled, and it’s thus possible to make the backend services evolve without impacting the frontend too much, once again, by only changing the API contracts.

Build better customer experiences - To support these multiple distribution channels and these new business models, having a headless and API-first architecture is essential. This allows the use, and even the reuse, of different e-commerce services without having to re-implement their behaviors. For example, the same shopping cart or search module can be used on both the mobile version and the web version. Frontend development teams can then focus on building a deep user experience. Building a strong identity and immersive features that will allow platforms to stand out. These websites benefit from the advantages of recent frontend technologies like NextJS, combining the dynamic aspect of a Javascript library like ReactJS with the performance and SEO provided by server-side rendering.

Bring robustness and scalability to its architecture - Finally, a micro-service and cloud-native architecture provides resilience and scalability. Indeed, having micro-services will allow us to scale on demand using only the services that we need most.

What headless CMS means for marketers –

Marketers can’t afford to delay their campaigns. They need to publish as soon as they generate new pages and content. Unfortunately, their development teams are busy, so their campaigns must wait.

Marketers also need to brand refresh their websites to stay ahead of the competition or widen their appeal. But as soon as they come up with their redesigns, their developers tell them to wait in line.

Headless CMS changes this co-dependent dynamic.

While older CMS systems need engineering teams to add new pages or content beyond basic formatting and image uploads, headless CMS enables marketers to make significant changes to their websites without any developer assistance. It allows marketers to integrate with the existing tech stack and continue to iterate and improve the website over time.

Traditionally, the front end of your website (what you see) and the back end (where your content is processed) were bundled together as part of the same single-coded system, which meant that developers needed to recode new pages or significant changes.

Because a headless CMS separates the code from the content, it enables your developers to code the back end as they want, as long as the engine they create processes the content in a way that is transparent to the specific content it processes.

By example of having a racing car –

Let’s say you want to try car racing. You don’t really want to invest a lot upfront, so you pick whichever car is available for you to use so you can test it out. That’s your first car—simple and whatever is available.

As you start getting more and more serious about the sport, you decide to make an investment. You get a second car specially made for you. Better tech, more customized. However, once it’s built, you can’t really improve upon it or upgrade to the newest race car technology. You’re stuck with your good old car. This can be like hard-coded websites or old CMSs that do not use headless API technology.

You decide to go pro. You hire a team of engineers to build a custom car for you—but this time, they use a technology that allows modulation. The parts are easily exchangeable so that as the car technology changes, your car can change and adapt to it. For example, if you want to upgrade to the newest engine boosters or aerodynamics, you merely swap out the old ones and replace them with the new ones.

Headless CMS is like that. Its API technology is like the modularity of the third car. The API separates the car’s performance from the technology that drives it.

An API-driven CMS creates a flexible and powerful website that is fast and made according to your needs.

However, based on your resources and the stage of your company’s growth, you might choose the cheap first or traditional second cars—unwittingly treating content management as a new hobby. Let’s take a look at some of those alternatives.

**Headless CMS alternatives**

No-code website builders

No-code website builders like Wix or Squarespace are similar to the first car. They work well for getting started quickly with zero development support, offering drag-and-drop interfaces and pre-built templates out of the box. But they give you the least amount of control over your content and website design.

You have almost no control over the underlying technology, how you manage SEO, what you can do to speed up your site, and so on. While these tools are great for simple personal websites or small business landing pages, they get in the way of your creativity as your needs become more complex. As your business grows and requires more customization and scalability, you'll likely find yourself needing to hire developers to code a more robust website solution.

Hard-coded sites

Hard-coded sites are similar to a second car—customized to your needs. Except…you discover that you are not the one who’s driving; your developer is.

If you’ve ever joined a smaller startup or a company where developers had full ownership of the marketing website, you might have encountered a hard-coded site.

When you ask for simple changes—for example, you spotted a typo on the homepage copy—your developer needs to make that change. This can be frustrating for both you and your developer, as it creates additional, low-value work for both of you.

Don’t even bother designing new campaigns or web pages.

While hard-coded sites give developers full control over the technology they choose and how they build websites, they don’t give you the flexibility to manage content the way you want. So, you purchase third-party-coded traditional CMSs.

Traditional CMSs

Traditional CMSs are also like the second car, but with more control. You get more control over certain aspects of your website, but when you decide on the solution you want to use, the technology your developer can use is determined by that solution.

WordPress is like this. Developers must use one code-base—PHP—to manage your website, inhibiting them from using the newest web technologies.

Additionally, you discover that it doesn’t give you as much control as you need. As your company grows, you might start to encounter more and more limitations in how far you can push your custom designs, what parts of the site are editable, what control you have over your SEO, how you structure your URLs, how fast your site is, what functionality you can add to your site without relying on too many plugins that impact your performance, and so on.

As stated above, API-driven, headless CMSs give you full control over your campaigns and page designs, and they allow your developers the freedom to [choose the best tech stack for your projects](https://prismic.io/guides/find-the-best-stack-for-your-website).

**How headless CMS benefits your developers**

Here’s the key point for developers to consider: with a headless CMS, the front end of your project is not a part of the CMS any more. A headless CMS has no control over the way the content is presented. The CMS just does what it’s already good at—displaying, managing, and storing content.

In other words, you don't get a website out of the box with a headless CMS. This might sound like a bad thing, but in fact, it isn't. Unlike a "traditional" or "monolithic" CMS (like WordPress or Drupal), a headless CMS focuses exclusively on:

So, what does that mean for your work? It means you can build the front end of the website however you want to, using whatever technology you think is best. Then, in the places you want to include dynamic content (which could be everywhere), you simply set up an API call and retrieve and display the returned data.

Prismic is an example of headless page builder CMS –

<https://prismic.io/page-builder>

<https://my-website-hg.prismic.io/masks/hello.json/>

A screenshot of a computer

Description automatically generated

**Cloud-Native**

The term "Cloud-Native" is used to describe applications that are designed specifically for deployment in the cloud.

Cloud-native apps are also designed to be highly resilient, with each microservice being independently scalable and replaceable. This approach enables organizations to quickly adapt to changing conditions and ensure that their apps remain available even in the event of an outage.

ButterCMS is an another headless CMS example –

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

We're starting to switch to strapi for our cms and svelte as our client. Headless cms is much closer to using wordpress than to building things from scratch.

# Bootstrap: Build Interactive and Responsive Web Apps

Today rows and columns in Bootstrap are flex box based by default, but they do behave a little bit differently than standard flex boxes.

Breakpoints –

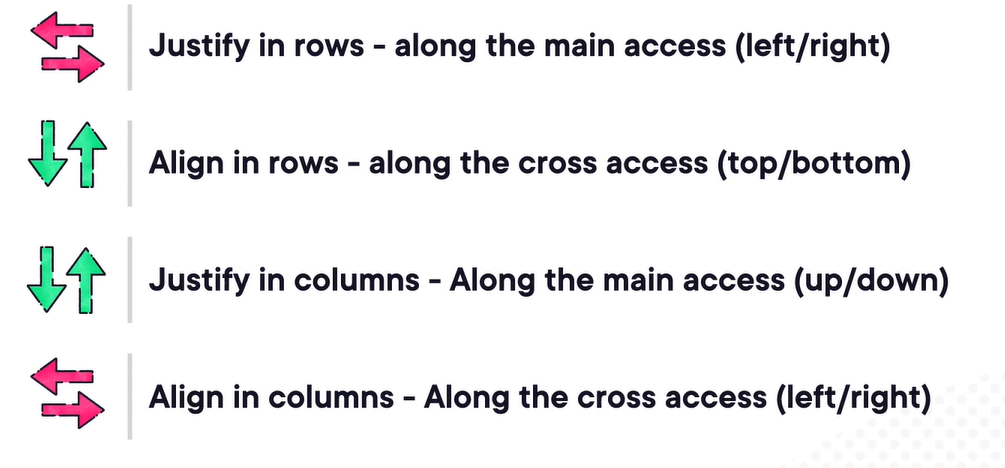
A group of rectangular boxes with text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Justify vs. align –



Justify flexbox columns –

A screenshot of a computer

AI-generated content may be incorrect.

Using d-flex class –

A logo on a white background

AI-generated content may be incorrect.

A computer screen shot of text

AI-generated content may be incorrect.

Color classes –

A close-up of a sign

AI-generated content may be incorrect.

While using the modal, the footer and header should always be fixed and only the content should be scrollable.