Contenful - The Missing Manual

Jon D Jones

Contents

Welcome

Welcome, to my 4th book!. By purchasing a copy, you have officially become a bona fide legend! Within these pages you will learn about all the essential steps that are required to build an epic website that is powered using Contentful CMS.

As of writing, this is the first book on Contentful CMS that has been published. Where this book differs from the majority of the other CMS books is on its focus. The intention of this book is not to be a glorified content editing manual. Instead, the intention of this book is to be a reference for developers that explains the complete end-to-end process that a team will need to follow in order to successfully deliver a headless web project. Also before you worry… yes, it also contains all the code samples and implementation details you would expect it to.

When it comes to starting and successfully delivering a new CMS powered project, there are two fundamental elements. The first is the code. Knowing and understanding the ins and outs of the programming language that you are using.

The second element in a headless project is being able to define the overarching architecture. Without fully understanding the role a headless CMS plays in the process, the capabilities you should expect it to provide (and the capabilities it does not cover), the technologies required to deliver all the proposed features and how you can budget for these things, you are setting yourself up for failure. Not getting these fundamentals right is a recipe for disaster.

The biggest project failure that I ever worked on occurred around 2015. At the time, I had been working in the industry for over a decade, I had helped deliver a lot of enterprise-level websites and I knew what I was doing. I was brought onto a team as a contractor to help deliver a website for a large global brand most readers will have heard of.

The overall brief from the client was simple, an e-commerce site that made them money within x amount of time for x budget. They had just got an existing company to create a new platform for them. This new platform met 90% of their business use cases, however, some aspects of it had issues. The relationship with the existing vendor had deteriorated badly and they wanted us to finish off the project for them.

The lead architect on the project didn't take this brief. Instead, as a team, we were forced into recreating another brand new website. This new website needed to work within some utopian architecture. Instead of using the licensed software the client had purchased, we were briefed to re-invent the wheel and built custom solutions that took effort to build and which diverted effort from the features the client actually cared about. On top of this, these custom solutions also increased their operating costs, they were paying twice for the same capabilities!

Neither of the two new sites ever saw the light of day. All the efforts by lots of people were discarded. I feel one of the main reasons why this project caused so much controversy was because of the difference in opinions between the two sides about what needed to be built. The client expected us to fix their new website within a few months of effort and to help them make money. Instead, we spent months rebuilding the brochureware part of the project to work within a new architecture. This only compound the tensions, as the brochureware aspect didn't help them make any money.

This story highlights what can happen when there is communication breakdown around architecture. In a headless project, there is a greater chance you might find yourself in a situation like mine. Fail to scope a project correctly and the end operating costs could be thousands of pounds more per month than you planned. Not having a good understanding about which features you can expect to get out-of-the-box and which you will need to custom build yourself, can mean disaster. In the worst-case scenario, you might miss deadlines by months or years.

This is why one of the most fundamental aspects of a headless CMS build, is understanding what you need to deliver. After you have a clear and accurate understanding of your client needs, you will decrease the chance of getting things wrong. This will then allow you to realistically assess which features you can offload to the CMS (or another tool) and which feature you will have to build from scratch. My recommendation is to always try and use something ready-made as in general it will make everyone's lives a little easier. Its somewhere between these pillars of understanding and delivery that project success can be found.

This is why the focus of this book is for technical readers. The book not only covers how to use the CMS, it also covers the different options and approaches that you can pick from when building a headless website powered by ContentFul. Depending on the implementation route that you pick, will also have an influence how you configure Contentful. Understanding how all the bits of the puzzle fit together will help increase the odds that your project is a success.

Simply understanding how the CMS works is not enough. There are also a number of ancillary tooling that you may also need to consider, in order to deliver anything meaningful. For instance, how are you going to host your website? Are you using JAMStack? What about search?

In the real world, regardless of how perfectly you write some code, if that code does not address the fundamental business problem a company is trying to solve and you deliver it within budget, do not be surprised if your technical masterpiece never gets used.

The aim of this book is to allow you to build a website for any client, whether it be enterprise or personal, using Contentful CMS. With the introduction nailed, let us get to the good stuff!

Developer Prerequisites

One cool aspect of a headless project is the freedom that it gives you as a developer. Contentful is classed as software-as-a-service (SASS), meaning you will not need to install the CMS locally yourself. Instead, you will access Contentful using an internet connection and a web browser. The great thing about SASS is that you are free from the responsibility of patching, or upgrade the CMS whenever a new version is released. As a developer, all you need to focus on is building a righteous website by talking to an API.

This means that the prerequisites required to start a headless CMS project are minimal. Basically, you need the ability to write code, push code to source control and run your website locally. Like the majority of developers, I use Visual Studio code as my IDE of choice. VS-Code is the most used IDE by developers, its free, and it has a great extension library. What more can you want?

You are free to use whatever IDE you want, as long as you can install NPM packages you will be able to do everything outlined within this book. If you want to know which VS-Code extensions I use to optimize my development flow, I recorded a video outlining everything which you can find here.

I use Github as a code repository combined with the command line to pull and push things. I recommend that all developers invest the time and learn how to use Git from the command line as it's a skill that will last your whole career and it's a skill that will make you more productive. Those really are the only two essential tools that you need to get started!

If you are anything like me, one annoyance when reading any programming book is being able to access the included code samples. Having to manually try and copy and paste code out of a book is annoying. Additionally, code samples that are spread over many pages are hard, if not impossible to understand.

To make your life a little easier, I have included all code samples within this book within an accompanying GitHub repository. You can access this repository at:

https://github.com/jondjones/contentful-the-missing-manual

If you want a quick way to search the code contained within this book I recommend bookmarking that link!

The final handy resource to mention is the Contentful forum. As you go through this book, if you get really stuck trying to implement something, I recommend you head over to the Contentful forum which is available here and ask for help over there. Failing that Stackoverflow is always your friend.

Why use a SAAS headless CMS?

I frequently get asked all sorts of questions about CMS development through my work, my website, and my YouTube channel. Out of these questions, there always seems to be a subset of developers who tend to think pretty negatively about headless CMS development. The majority of this negativity tends to come from folks who have a history within CMS and fail to see the benefit of switching.

For some reason, many older developers can be very opposed to even considering a headless CMS solution. Typically, these herds of disgruntled developers tend to come from a web development background and are not very familiar with the whole Javascript/Typescript ecosystem. They also tend to already have a "favourite" CMS

I have found that one of the main reasons why certain people adopt this mindset is based on some he-said/she-said horror story that someone once told them in a pub. "Oh, I don't want to use headless, because I know someone, who said such and such a client couldn't preview anything!"

Yes, these horror stories do exist for Contentful, however, these types of stories also exist for every language, framework, package and tool that has ever been created.

The simple truth is that the team implementing a project has the biggest influence on whether that project is successful or not. If the team does not really understand the system they are using, they will configure and implement it incorrectly. This in turn can create a bad experience for content editors, which leads to a horror story.

Whenever I encounter a real-world horror story, I always question the storyteller about the challenges the team had during the project. It is only when you question, that you learn the truth. "Oh yeah, the lead developer quit halfway through and the client made us deliver the site a month before the agreed deadline, so it was rushed"

The point I am trying to make is that if you do online research about headless CMS you will quickly find good and negative opinions on it. It is very easy to find stories that will collaborate with your thinking.

One of the reasons why some folks can have misconceptions about headless CMS is that enterprise-level solutions, like the CMS systems provided by Adobe, Optimizely, or Sitecore, approach the challenge of CMS from a different perspective. These enterprise-level CMS solutions tend to provide a lot more capabilities than just content editing. These CMS systems, fall into the bracket also known as digital experience platforms (DXP).

DXPs are a digital tool that can be compared to a swiss-army knife. The reason why these tools exist is mainly due to the CMS sales process. Over time, as all the different CMS companies kept trying to outdo each other, more and more capabilities got added to the CMS. After many years of this competition, the CMS systems eventually did so much that the category of software they resided in changed. It went from CMS to DXP!

If you think of this in terms of pure sales, this is understandable. In order to get a competitive advantage a sales team needs to have features that their competitors can't match. This is why building new features is a key aspect for all software technology vendors.

Your affinity towards tools that do everything will determine which side of the fence you stand on. A headless CMS tool will fundamentally approach the problem with a different mindset compared to DXPs. DXPs try to solve all the problems, while headless tools tend to simply solve a single problem. This is why you can find a lot of conflicting articles online about which approach is best.

In order to try and explain both camp's positions, let us consider some of the main justifications that always seem to get mentioned in these debates:

Sub-par features: Each technology company normally has an insane amount on their product to-do list. With constant pressure to work on the next ticket, a CMS vendor will typically build a feature as a minimal viable feature (MVP). The typical development cycle is to build a feature that ticks the minimum number of required requirements so the team can start the next ticket as fast as possible.

If the vendor's customer base reacts favorably to a feature, it might be improved later on, however, if it doesn't get high adoption then that feature will likely never be touched upon again. This product development cycle continually repeats. After a few years, the CMS system starts to become composed of more and more features. Some good features and some not-so-good features.

When a potential new customer evaluates a new CMS, they will usually have a tickbox of things that the CMS has to do. The simple fact is that when a CMS has lots of features, it will get considered for more opportunities.

In a lot of instances, a customer may pick a CMS because of a certain feature. Sadly, more often than not, the company ends up never using that feature. In other scenarios, because the feature is average it might not give the benefit the customer expected, so they stop using it.

A great example of where this buyer's remorse can occur is when picking a CMS because it offers A/B testing. Several CMS systems ship with pretty bad out-of-the-box AB testing capabilities. If you are serious about experimentation, you will want to buy the best tool possible and spoiler alert, no in-build AB feature provided by a CMS makes this list!

This story highlights the pro vs against DXP debate. The pro DXP'ers will negatively judge any CMS that does not ship with A/B testing capabilities. The against DXP'ers will say, why buy a feature that is not great, which does not give them the best business outcome possible, and which they are then forced to maintain in the future?

To re-cap, the pro headless camp thinking is that best-of-breed software will give the team access to the best tool possible, which should create a better overall system!

More code to maintain: The issue with a DXP that has lots of capabilities, is that the chances that the system needs to be upgraded more frequently increase.

DXP-powered websites will typically be more reliant on that single DXP platform. Meaning the next time you need to upgrade your CMS, the team will have a higher chance your upgrade will take longer as you have more code to maintain.

The pro headless camp will say that by splitting responsibilities into different tools, there will be less need to upgrade software and when you upgrade the impact will not be as severe.

Admin effort: If you sit in the pro-DXP camp, your preference will be to minimize the amount of admin effort involved in the procurement process. It takes time and effort to research, onboard and maintain a new software vendor. The process will involve lawyers, finance people, training and more. With a DXP you have a single contract to agree upon with a single vendor, however, that contract will typically be a lot more important and in-depth!

If you agree with MACH and headless, you will want to start your project ASAP. You will not want to wait and rely upon some big contract being signed before you start the project.

Many of the new SAAS tools allow you to get going on a Freemium tier without signing a big contract. All of the tools that I mention in this book will allow you to create an account and get up and running with that tool in under 15 minutes. Most of the time you will start your project in development before going to production. You can access the Freemium tier while getting the lawyers to sign contracts in parallel. Speeding up your project timelines.

Lastly, if you find a certain feature is not useful, you do not need to worry about signing an enterprise license. You can try it before you buy.

Yes, there is going to be more admin in headless, the tradeoff is starting a project sooner.

Upgrades: In order to get access to a new feature released by a DXP vendor, you are typically forced to upgrade something. The upgrade process could involve changing a package, a database, an executable, a framework, or a combination of all of the above.

I know from first-hand experience, upgrades can take blood, sweat, tears, and arguments. The worst upgrade project I worked on took over six months!

The pro-DXP camp will consider upgrades as good because they feel like having the power over when they should upgrade is less risky. They will not want a vendor changing anything without their knowledge, they will need to do everything themselves in-house.

When you use a headless tool that is also SASS, like Contentful, you never need to worry about upgrades. The thinking within the pro-headless camp is that as a business their focus is on doing whatever they do best. The purpose of their company is not a software maintenance business. They understand that they are not experts on the tool in question and that the vendor will likely do a better job of maintaining the tool they created as they understand it better. They want to focus on growing their business, rather than maintaining software.

Hosting The arguments around hosting is the same as the ones made above. Do you want to manage the hosting responsibilities in-house, or, do you want someone else to take the hassle away? When you use a SAAS tool, you do not host the software. Hosting, patching of servers, and dealing with 24/7 support and up-time is transferred to the SAAS vendor.

MACH Vs Traditional?

I think the above points help to showcase one of the most misunderstood aspects of building a headless website powered by a CMS, responsibilities.

Traditionally, a CMS/DXP did everything for a developer, in terms of features and in many cases infrastructure. A DXP will typically include things like a search and personalization engine, it will provide ways of managing frontend members, and it will be customizable so you can include your own custom screens. In the new world, this way of thinking has evolved.

The intention of Contentful is to be a CMS that can provide you with everything you need to create and deliver content, nothing more. This is why when you decide to use a headless CMS, typically the story is not just about picking a single tool, it is about picking a combination of technologies that can work together. It is this power to combine tools that will allow you to build a kick-ass website.

When you go to a restaurant, you will want to get the best meal possible, based on your liking. To do this you will pick a starter, a main course, and a dessert from the menu of options. The same is true when creating a website powered by Contentful.

After picking Contentful as your CMS, you will also need to decide how you want to build your head. What other technologies will you need to use? Javascript, Typescript, Swift, React, Kotlin, Vue, NextJs, Flutter, or, Gatsby.

What other tools may you also need to use? Does your site need to provide a search engine? If so, you will likely need to pick additional kick-ass SAAS tools. Will Algolia, Optimizely, Netlify, Big Commerce, or Cloudinary make the list? These choices can seem endless. Do not overlook the fact that at the start of your project, you will also need to learn about the best SAAS tools in the marketplace.

A lot of headless naysayers will quickly point out that Contentful CMS is rubbish as it doesn't provide a search engine. Yes, this is true… however, when you typically pick a DXP CMS that has its own search engine, you typically end up with two OK products, rather than two great products.

In the SAAS, best-of-breed world, if you want to provide a search engine, you would likely consider using Algolia. If you want to create a membership area, Auth0 provides a great way to manage log-in. Need a provider to host your website? Netlify will integrate nicely

This mix-and-match philosophy is where I believe the true power of this new approach comes into its own. Headless turns the challenge the DXP herd has on its head. Big, monolithic code base are notoriously hard to maintain. This is why the DXP vendors who have been in the market for 20+ years, typically find it harder and harder over time to release new features quickly. By slowly adding more features to the CMS, they limit their ability to be agile.

The same principle is true for your project. The more composable you can make your architecture, the easier it will be to pivot later on. ContentFul is very clear in its intentions. To be the best CMS in delivering content via an API. This intent is not to be the best member management tool or the best A/B testing tool. If you want these features you need to either build or, or, what is more likely buy it.

SAAS tools also provide some other less obvious benefits. When going with a traditional DXP-powered CMS, you will also need to regularly maintain the software. Taking this route, you are faced with an issue. Do you take the responsibility to host the site in-house and potentially reduce your hosting costs but increase your maintenance responsibilities? This comes with the burden of maintaining the software. Expect to regularly invest time upgrading and patching the software and the infrastructure. Also, how do you deal with bursts in traffic? What happens when the site falls over in the middle of the night? You need teams to maintain the servers, provide support, deal with scaling issues, etc…

In SAAS, the vendors will manage the tool for you. As the tool lives in the cloud, updates and patches will be managed for you. The tool will be configured so it can be scaled if you get a spike in traffic. It will also come with DDOS protection so you do not need to deal with malicious attacks. Do not forget there will be 24/7 maintenance and support included in the background. There will be a team of people making sure everything is running OK. Maybe that's the reason the monthly SAAS bill can seem quite expensive at first glance!

Granted, headless and SASS are two different topics that I just munged together, however, when you go online most articles you find will often do the same. Assuming I have convinced you about the benefits of headless and SASS over traditional CMS systems, the next question is why should you pick specifically Contentful?

Why use Contentful CMS?

When it comes to using a headless CMS you are faced with a lot of options. To prove just how many there are, you can head over to Jamstackers.org. Jamstackers.org publishes a list of pretty much all of the headless CMS solutions that are currently available within the marketplace, both free and commercial.

This list can be filtered by `Open` and `Closed` source projects. Projects are then displayed based on the number of stars it has on Github. Obviously, Contentful CMS is on the list, however, as Contentful is not open source and not on Github it has no stars. This means that you will have to scroll for a while before you find it in that list.

Before I start sharing tips on how to build a website powered by Contentful, it is prudent to call out some of the reasons why I think it is a good option, what exactly makes Contentful stand out?

According to G2.com's Best Headless CMS award, Contentful is ranked as having the largest market share in the mid-market category with 36% adoption. It also has an impressive grip on the small-business market with 33%. based on G2, Contentful has pretty solid all-around user satisfaction scores.

In terms of CMS functionality, Contentful has all the capabilities that you would expect a CMS to ship with. Key features for content editors include:

- Content modelling capabilities with 9 different field types including rich-text, text, number, date, time, location, media, boolean, JSON object and reference type.
- Content scheduling
- Content versioning
- Publishing workflows
- Environments
- Role-based user permissions with multi-factor authentication
- Internationalization/multi-language with fallbacks
- Content tagging
- Content feedback with tasks & comments

For developers, you will get access to:

- Querying for content via GraphQL

- Querying for content via API endpoints
- 8 SDKs to render content, including Javascript, Python, /NET. PHP and Java
- Import/export via CLI
- Webhooks

Yes, the majority of headless CMS systems listed on JamStackers will also have these capabilities, true. A feature matrix is definitely handy to point out what Contentful can do, however, it does not really highlight the reasons why you should pick Contentful CMS.

Personally, I think these three features, in particular, make Contentful stand out:

Marketplace: One of the main reasons why I consider Contentful as one of the top headless CMS contenders is because of its partnership network.

Contentful has a marketplace with over 70 apps. You can view the marketplace online here. Within this list, you will see tools for analytics, commerce, artificial intelligence, asset management, optimization, search, video, etc…

The majority of other headless CMS solutions do not have their own marketplace. The ones that do, tend to ship with limited apps that are created by the CMS vendor themselves. These apps are focused on adding slight improvements to the CMS, like adding a new content type.

One example of this is Strapi CMS. Strapi has one of the largest marketplaces with over 90 apps, however, 95% of these extend what Strapi can do. In the Strapi marketplace expect to see tools like UUID field, TinyMCE editor, and a Preview Button.

If you compare this to the Contentful marketplace, you will notice the difference in the scale of integrations. Contentful apps include adding A/B testing capabilities with Optimizely, adding e-commerce functionality with BigCommerce, adding a DAM with Cloudinary, backing up your site with Dropbox, or improving your teams' communication with Slack.

Contentful provides apps to integrate with partners, including, Brandfolder, Bynder, Cloudinary, CommerceTools, Dropbox, Frontify, Google Analytics, Jira, Optimizely,Shopify, Smartling, Typeform, AWS, Algolia, Bitbucket, CircleCI, Elasticsearch, GitLab, Heroku, Mailgun, Slack, Travis CI, and Twilio.

If you can leverage the marketplace, you will end up getting a project up and running with a lot less effort compared to other solutions.

Environments: Any developer worth their salt (does anyone know why salt is so valuable?) understands that when it comes to shipping bug-free software, we need a process that involves a level of isolation.

In order to prevent disruption to production/customers, the development team need to build a feature in development and when those features are completed and tested, only then should it get merged into production. When it comes to CMS and content authoring, these requirements do not change.

This is where the Contentful `environment` capability can help. For us developers, it is probably easier to consider a Contentful environment as a feature branch in Git. To avoid messing up production content during development, use a Contentful environment to create a separate space/area where you can change things in the CMS without impacting production.

When you are finished working on your task, you can copy your changes back into the production CMS branch and finally delete the Contentful feature branch.

The next time you need to create a new feature in code, you can quickly log into Contentful, and create a new feature branch based on the production branch. This process will take about a minute and it will mean you will have access to an environment that contains all the latest content updates which you can use and update without impacting production.

As long as you use the same name to create these environments like `development`, you will not need to update anything within your code. It will all just magically work!

The main and default environment that gets created whenever you create a new space within Contentful is called 'master'. Whenever you create an additional environment, you can pick the branch to branch from. In the free tier, you can create 3 environments.

We will delve into creating a new environment in detail later, for now, you can read more about it here.

Quick to get going: One issue with a traditional DXP, like Adode, is that it can take months before you are given access to an environment. You need to get everything agreed upon before you can create anything. As Contentful has a free tier, you can get a production-ready website up and running with Contentful in under 20 minutes. In terms of pace, this can not be beaten!

The last thing I like about Contentful is that I find the UI nice to use and using some templates you can find online, you can get a working site up and running in under 10 minutes.

One reason why you can move so quickly is that there are a number of starter kits and themes available for Contentful.

You can use sites like Jamstack themes and Themeforest. Pick a theme, log into your COntentful account and all the content will be created for you. Hook the process up to Netlify and you have a production-ready website with pages and content, ready online in minutes. For someone from my background, where getting a basic site up and running would useful take a few weeks. Getting that time down from weeks to minutes is amazing!

The true price of Contentful

Before we get to the good stuff, I think its worth covering an area that can cause a lot of controversy when picking Contentful, pricing.Some developers do not consider financial impact as part of project planning, however, it is an often unescapable necessity. The simple fact is that Contentful CMS is a bit of software that is created by a business. Like all businesses, in order to survive and thrive that business needs to make money.

The reason why I feel I need to highlight this at the start of the book, is because a lot of teams fail to consider this when they are picking the CMS they want to use on a project. One of the reasons why a lot of teams are drawn to Contentful is because it has a free tier. A lot of people make the mistake of thinking that because they start off on the free tier, Contentful will be free forever, this is a mistake.

For small brochureware websites with limited requirements, its possible to build a website powered by Contentful for free. As you would expect from any Freemium service, there are restrictions within the free tier. During the early stages of development when it is just you are your team building stuff, it is unlikely that you will bump into one of the tier restrictions. Hitting this limit can often happen just after a site is launched. When your requirements hit a certain threshold, your only options will be to pay for the premium service.

Contentful provides two types of paid tiers, teams and premium. If you are an enterprise-sized organization you will have a big laundry list of requirements around what a CMS vendor must provide. It is almost guaranteed that an enterprise- size company will need guarantees around uptime with financial penalties if an SLA is breached. Other considerations might be a significant number of bums in seats who can access the CMS and the amount of content that needs to be created. Often large companies will also require real-time access to support. This list could go on and on.

If you are building a project that requires the capabilities offered by Contentful premium tier, you will need to get in contact with Contentful to get a custom quote. After contacting Contentful, an account executive will get in contact with you. Between the two of you, you can go over your requirements and the AE will be able to cost it up. As the price will depend on certain levers, Contentful can not advertise a flat price for enterprise costs on their site. I do know that pricing can start from roughly $4000 a month.

For the smaller companies, a developers lack of planning around potential future operating costs can be critical. The teams tier costs $489 a month, so $5868 annually. Promising a client that Contentful is a free CMS, then 6 months later invoicing them for 6K can be a very unwelcome surprise. This type of unplanned news can be a make or break relationship. This price increased can also be compounded if your project uses other SASS tools. Typically, if you project reaches the level where you need to scale the CMS, it is often likely that you will also need to scale the hosting and other SASS tools used within your project. That promise of free operating costs, can very quickly jump to operating costs of 10K a year if you are not careful.

This is why before undertaking a project that uses Contentful, I strongly recommend you think about pricing and tiers. I think Contentful is a great CMS which is worth the money, however, it is a mistake to not make the client aware about the risks around potential increased operational costs before you start writing code.

This communication is key in order to establish a long and happy relationship with a client. Some companies will not be able to afford the price increase from free to 6k a year. When a company is forced into a corner, their only option might be to cancel the project, re-platform, and pay someone else to rebuild their website and then dump you like yesterdays leftovers. This is why it is important to take the time and consider what you will need to likely use in the future before committing to Contentful. We all love Freemium, until we hit its limits.

I point this dilemma out from personal experience. On one project, we identified Contentful as a great platform for a client. Based on their needs the free pricing tier gives the client everything they had asked for. Shortly after releasing the website, the client asks for some bigger and better capabilities on the site. Even though we had made the client aware of the limits at the start of the process, they still had an unhappy shock when we told them about the cost increase.

In the remaining part of this chapter, I will explain what limitations you need to consider and how those limits could impact your software budget.

Tiers: The first thing to make a client aware of is the jump in price between the tiers. You can start to use Contentful CMS for free, however, when you need to scale there is no small step between the free pricing model and the next tier. As mentioned, the jump goes from free to 6k a year. This is not insignificant. Some companies might be easily able to afford this price hike, some might not.

You will need to consider if this price is something the client can live with. If you compare this yearly cost to a self-hosting CMS solution, like Wordpress or Umbraco CMS, take into consideration everything. Self-hosting software wil always seem cheaper at first glance, however, you also need to take into consideration the invisible costs of the time required to patch, upgrade, monitor, host and support that software. The benefit of SASS is that this is taken care for you and that price is also included within the monthly fee.

Features: In order to successfully plan a headless project, determining the capabilities of your project at the start is essential. The scope of most projects will include a lot more than simply creating some screens or pages that render content.

Typical features that we all know and love could include, e-commerce journeys, secure member areas, or site searches. These are all examples of extra features that a typical website is expected to offer. An important point to understand is that just because your website does a lot of things, this does not mean that you should expect your headless CMS to handle everything for you.

Contentful is a tool that adheres to the best-of-breed mentality. As a product, its focus is on providing content editors with a great editing experience and it does this really well. Contentful does not try to pretend that it does everything for everyone.

This limitation around product offering is a very different approach compared to most of the traditional CMS systems in the market. Traditionally a CMS has promised to do everything for everyone. To emphasis this point, it is worth pointing out that there is even a completely different category of CMS systems that fall into a category of software called the digital experience platform category (DXP). CMS systems within this category include Adobe and Optimizely.

The aim of these DXP tools is to be a one-stop shop for everything you might need. For example, Contentful does not offer member management capabilities, whereas a CMS like UmbracoCMS or even WordPress provides both.

Despite what each CMS vendor's marketing hype might try to make you believe, one approach is no better than the other. Like most choices in life, each CMS platform has its strengths and weaknesses. There is no perfect fit for everyone.

Instead, determining the best CMS for your project will be determined by a number of factors. These include what it needs to do, the budget, the size of the team, the number of developers, and its budget. Combining all these factors together will influence which type of CMS is right for you.

In the new world, you will potentially need to select a different SASS tool in order to deliver each core feature that is offered by your website. Meaning, the monthly cost for Contentful might not be the only cost that you may need to consider. If you want to provide a great search in your site with faceted filtering, you may need to pay for a tool like Algolia. If you want to build a members portal, you may need to introduce Auth0 into your stack. If you need an asset management solution to edit images, you might want to consider Cloudinary. Do not forget you will also need to host your website so you might want to consider Netlify.

Each of these tools has free and paid tiers. Assuming you wanted to build a website with all the tools listed above, if you need the enterprise tier for all of them, expect to pay 10-30k a month. If you work on a small project, the price could be completely free. This is why when using SASS tools, having an exact understanding of which tiers you need to use for each tool that you need to use is key. The difference in price points can get very expensive very quickly if you fail to plan accordingly!

This pricing might sound stupidly expensive, however, the theory is that by picking the best tool for each job and combining them, your team will get access to way better capabilities compared to a CMS that tries to do everything. This is where the best-of-breed mantra comes into play.

The point that I am trying to be crystal clear about is that Contentful is aimed at content editing and content delivery. If your specification defines any additional functionality, you may need to purchase additional products.

In simple financial terms, the more products you use, the greater the chance your monthly operating expenses could increase. This potential impact on the projects operating costs is the main reason why I recommend that you get a clear picture of what your website needs to do at the start. On my projects, I always aim to get this clear understanding before committing to using any SASS tool on a project.

Having the ability to pick the best product for each feature, means you can create a much better overall experience and you are not constrained to using sub-par features. It also means that all the hosting costs, upgrading costs, and support costs for that tool are taken care of for you for one monthly price.

If you are interested in this topic and you want to learn more about other headless SAAS tools there is an organization called the Mach Alliance that you should be aware of.

MACH is short for micro-services, API-first, Cloud-native SaaS and Headless. This group of individuals defines a group of SASS technologies, which include Contentful, which are known to play nicely together.

One issue I have with some of the recommendations on the MACH alliance website is that the recommendations also include digital agencies. Obviously, you can pay these agencies to design and build a headless website for you. If the aim of the group is to promote technology, why promote companies that make money from using those technologies? For me, it muddies the water a little bit too much.

Just because an agency is not on that list, that does not make them worse at implementing headless websites compared to those on the list. It also means that those agencies are more likely to try and influence technologies that they know and use, which could stunt the adoption of new tools in the future.

I know from personal experience the agencies listed on the MACH alliance page are just as happy to use a DXP tool to create a website. If you check my CV, you will notice that I have previously worked at one. As long as an agency can make money, they will build a project, however, the client wants.

One good thing about the MACH alliance website, is that it can give you an indication about what tech works together nicely. If you are struggling to plan a headless project, the MACH alliance website can be a good place to learn about what else you need to start thinking about. I will also be covering these considerations as we continue through this book.

Number of content editors Another element that will influence most SASS vendors' pricing is access to the tool. How many people from your team need access to the CMS? Within the Contentful free tier, you get a maximum of five content editors. If you work in a large team, with a handful of developers, testers, project managers and a few clients, be aware that you will hit that limit quickly. One workaround is to share login details, although this can get messy. If you have lots of people who will need access to the tool and governance and cadence is really important, this will likely be a reason why you will need to use the paid-for tier.

Environments: Within Contentful, it is possible to create multiple environments. Environments are really handy for development, testing and user acceptance.

When writing code, it's much easier to target a staging environment, so you can change content models and create pages without impacting the live site. For testing new code changes, it's easier to sign something off in isolation, before it is merged into production. Why risk new code potentially breaking existing behavior when you do not need to? For approving new content changes, it is much easier to update things on staging and test that they work and look OK before merging those changes to the production pages.

In the free Contentful tier, you are allowed to create three environments. For most projects, a development, a QA and a production environment will be enough. If you need more, you will need to pay more.

Another thing that I should point out is the limitations around copying content between environments. When you create environments, you can not copy content between them within the UI yet. The process of copying content requires a developer to use the Contentful CLI tool. Using this tool, I have hit limitations when copying deeply nested content. The tool can fail to transfer some of this content. This error is not a limitation of the free plan, however, something to be aware of when doing content migration! We will also cover this CLI tool later in the book.

Governance and roles: As part of the free pricing plan, you will only have access to two different editing permissions, `admin` and `editor`. If I am being honest, for day-to-day usage during a build, the `editor` role is a bit useless because it does not have permission to publish content. From my experience, during project kickoff, I have typically needed to give everyone on the team `admin` access.

Another limit to the `edit` role is access to environments. As mentioned above, when using Contentful CMS, I do not recommend performing all of your development work against the production environment. Instead, I recommend creating content in a staging area first and then pointing your local website to that staging environment. Using this approach means that you can create code and test it without impacting content editors.

In the free plan, the `editor` role can only access the `production` environment. If you want to allow a content editor the ability to edit content in a different environment, you will need to make them an admin. This means that on every project I typically need to give all content editors the `admin` permission. This means those editors can do everything, including delete the website! If you do not like that prospect, you will need to use the premium tier where you get access to more granular permissions!

Number of supported Languages: If you need to create a multi-language site, the free plan will only allow you to create three languages. If you expect the site to support more languages in the future, this will be a limit that will force you to upgrade to the next tier.

SLA: A lot of companies think they need an SLA so they opt to pay for the premium license, however, that statement is not necessarily true. Within the enterprise tier, you will obviously get a much better SLA compared to the free tier. Note that I said SLA and not level of hosting quality. The underlining infrastructure Contentful uses under the bonnet on the free plan is the same as the premium plan. This means that regardless of free or premium, you get exactly the same underlining hosting service.

The SLA is simply a money-back guarantee your company could get if things go wrong. In most cases, it will be cheaper to not have an SLA. If in the future you think that you will need an SLA then expect to pay more. Depending on how much SLA you need, may mean you need to use the enterprise tier.

For most clients, I advise them to not worry as much about SLA as they might have when they used a traditional CMS. If you created your website following a JAMStack architecture, with static site generation, access to the pages contained within your website is decoupled from the CMS being "up". Atomic deployments mean that your website will work if the CMS is down. We will cover that statement in more detail throughout this book, so do not worry if that statement does not make complete sense yet. The takeaway is that if you are considering Contentful, consider if you really need SLA first!

Content: The final capability that can influence the type of tier that you need to use is an obvious one, the amount of content that you need to model. Within the free, or, premium tiers, you can model up to 48 different content types. In the free tiers, you can create up to 25K worth of records using those types. In the premium tier, this can be increased to 50K if needed. Anything larger than this will require the enterprise tier. If you know that in the future your project's content needs will scale a lot, be prepared. I think this is the main influencing feature that can catch people out from premium to enterprise!

Out of all of the projects that I have built with Contentful, these are the main points that clients have flagged as causing an unexpected increase in operating costs. Contentful is a great CMS, however, I recommend knowing these points before picking it.

For this book I am obviously assuming that you can see the value of Contentful CMS and you are committed to powering your website with it. The next step is to decide how we are going to build our headless website.

Additional reading and resources

- Environments FAQs
- Multiple Environments
- Contentful Community