Jamstack

The modern way to build Websites and Apps that delivers better performance

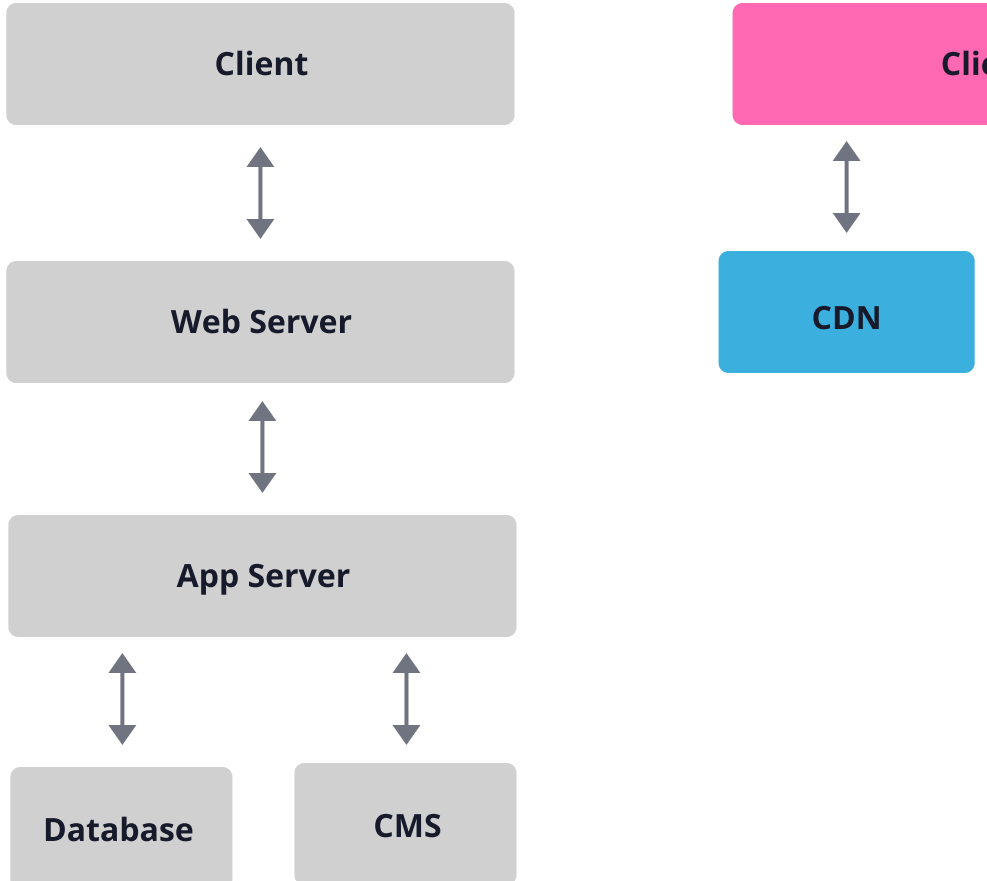
What is Jamstack?

- Jamstack is an architecture designed to make the web **faster**, more **secure**, and **easier to scale**.
- It builds on many of the tools and workflows which developers love, and which bring maximum productivity.
- The core principles of pre-rendering, and decoupling, enable sites and applications to be delivered with greater confidence and resilience than ever before.

The future is highly distributed

- Jamstack is the new standard architecture for the web. Using Git workflows and modern build tools, pre-rendered content is served to a CDN and made dynamic through APIs and serverless functions. It is for example easy to use github actions to build and deploy the front. Technologies in the stack include JavaScript frameworks, Static Site Generators, Headless CMSs, CI/CD and CDNs.

- The work was done during the build, so now the generated site is stable and can be hosted without servers which might require patching, updating and maintain.



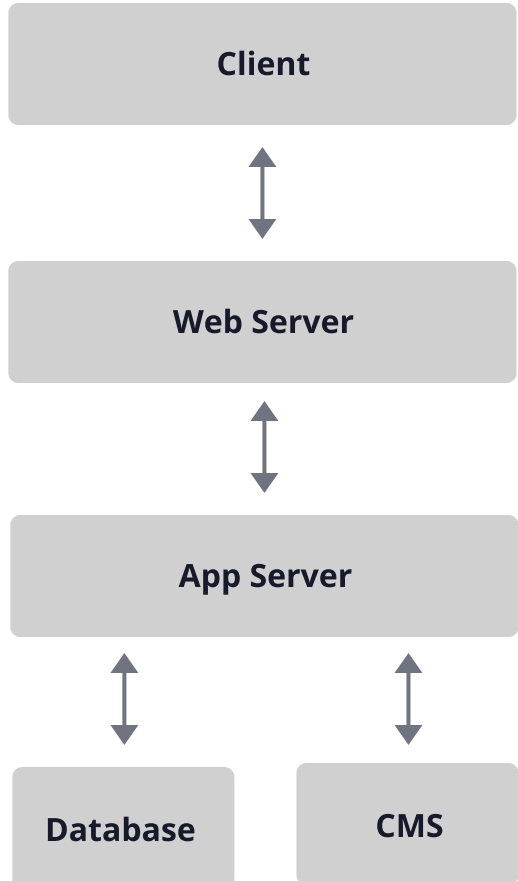
Talk outline

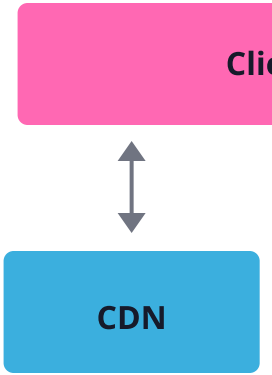
1. Jamstack
2. What is Jamstack?
3. Why Jamstack?
4. Best Practices
5. Comparing static code generator approach
6. A vision => Mulder: an extended Scully powered by DSLs
7. References
8. Learn More

Why Jamstack 1/6?

Security

- The Jamstack removes multiple moving parts and systems from the hosting infrastructure resulting in fewer servers and systems to harden against attack.
- Serving pages and assets as pre-generated files allows read-only hosting reducing attack vectors even further. Meanwhile dynamic tools and services can be provided by vendors with teams dedicated to securing their specific systems and providing high levels of service.



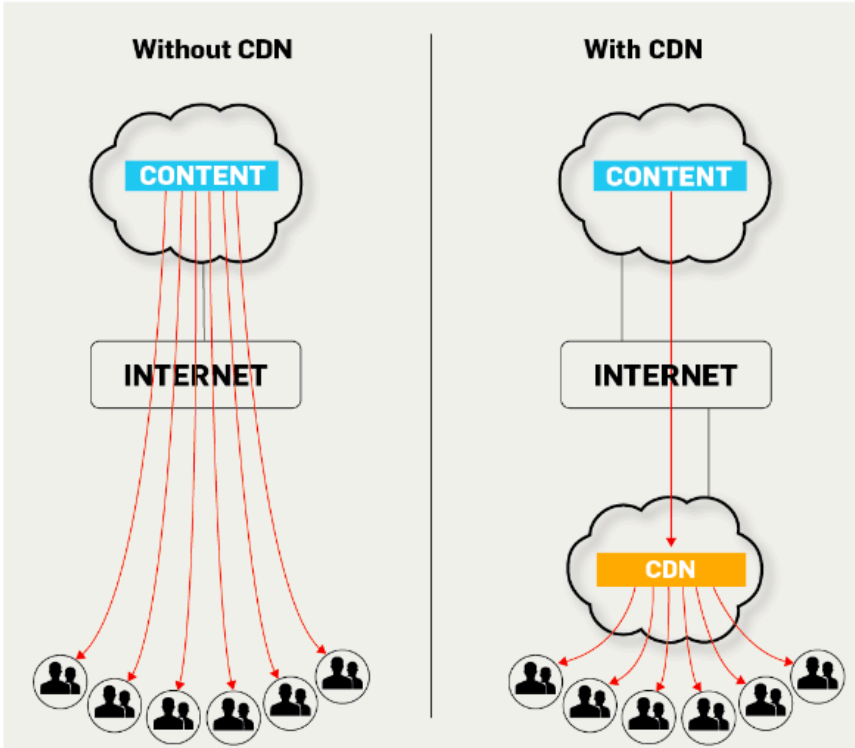


Why Jamstack 2/6?

//

Scale

- Popular architectures deal with heavy traffic loads by adding logic to cache popular views and resources. The Jamstack provides this by default. When sites can be served entirely from a CDN there is no complex logic or workflow to determine what assets can be cached and when.
- With Jamstack sites everything can be cached in a content delivery network. With simpler deployments, built-in redundancy and incredible load capacity.



Why Jamstack 3/6?

Performance

- Page loading speeds have an impact on user experience and conversion. Jamstack sites remove the need to generate page views on a server at request time by instead generating pages ahead of time during a build.

- With all the pages are already available on a CDN close to the user and ready to serve, very high performance is possible without introducing expensive or complex infrastructure.

Why Jamstack 4/6?

Maintainability

- When hosting complexity is reduced, so are maintenance tasks. A pre-generated site, being served directly from a simple host or directly from a CDN does not need a team of experts to "keep the lights on".
- The work was done during the build, so now the generated site is stable and can be hosted without servers which might require patching, updating and maintain.

Why Jamstack 5/6?

Portability

- Jamstack sites are pre-generated. That means that you can host them from a wide variety of hosting services and have greater ability to move them to your preferred host. Any simple static hosting solution should be able to serve a Jamstack site.
- Bye-bye infrastructure lock-in.

Why Jamstack 6/6?

Developer Experience

- Jamstack sites can be built with a wide variety of tools. They do not depend on the proprietary technologies or exotic and little known frameworks. Instead, they build on widely available tools and conventions. As a result, it’s not hard to find enthusiastic and talented developers who have the right skills to build with the Jamstack. Efficiency and effectiveness can prosper.

Best Practices 1/6

Entire Project on a CDN

- Because Jamstack projects don’t rely on server-side code, they can be distributed instead of living on a single server.

> Serving directly from a CDN unlocks speeds and performance that can’t be beat. The more of your app you can push to the edge, the better the user experience.

Best Practices 2/6

Modern Build Tools

- Take advantage of the world of modern build tools

> It can be a jungle to get oriented in and it's a fast moving space, but you'll want to be able to use tomorrow's web standards today without waiting for tomorrow's browsers. And that currently means Babel, PostCSS, Webpack, and friends.

Best Practices 3/6

Automated Builds

- Because Jamstack markup is prebuilt, content changes won't go live until you run another build.

> Automating this process will save you lots of frustration. You can do this yourself with webhooks, or use a publishing platform that includes the service automatically.

Best Practices 4/6

Atomic Deploys

- As Jamstack projects grow really large, new changes might require re-deploying hundreds of files.

  Uploading these one at a time can cause inconsistent state while the process completes.

> You can avoid this with a system that lets you do “atomic deploys,” where no changes go live until all changed files have been uploaded.

Best Practices 5/6

Instant Cache Invalidation

- When the build-to-deploy cycle becomes a regular occurrence, you need to know that when a deploy goes live, it really goes live.

> Eliminate any doubt by making sure your CDN can handle instant cache purges.

Best Practices 6/6

Everything Lives in Git

- With a Jamstack project, anyone should be able to do a git clone, install any needed dependencies with a standard procedure (like npm install), and be ready to run the full project locally.

> No databases to clone, no complex installs. This reduces contributor friction, and also simplifies staging and testing workflows.

Jamstack

System

Site Generators

A List of Static Site Generators for Jamstack Sites

Filter 355 Generators

All Languages

All Templates

Comparing static code generator approach

A first step: Static Site Generator (SSG) like Jekyll, Hugo

- **Jekyll** is a simple, blog-aware, static site generator perfect for personal, project, or organization sites. Think of it like a file-based CMS, without all the complexity. Jekyll takes your content, renders Markdown and Liquid templates, and spits out a complete, static website ready to be served by Apache, Nginx or another web server. Jekyll is the engine behind GitHub Pages, which you can use to host sites right from your GitHub repositories.

- **Hugo** is one of the most popular open-source static site generators. With its amazing speed and flexibility, Hugo makes building websites fun again. We use it for the DiverSE site and Gemoc Site.

Comparing static code generator approach

A first step: Static Site Generator (SSG) like Jekyll, Hugo

Pros

- Open source
- Huge community
- Plenty of themes, plugins
- …

Cons

- Developing a new plugin that consume micro-services is a bit paintful
- Do not fully embrace the notion of WebComponents

Comparing static code generator approach

A second step: Gatsby, VuePress, Scully,

- Angular, VueJS and React as well as now svelte are excellent frameworks that are very well documented to promote the development of the front end based on the notion of Web component. This notion of Web component is based on the literature around the notion of software component and aims, among other things, to promote reuse, composition and maintainability of the various elements of an application. It was therefore attractive to rely on these frameworks for the development of the front end while allowing server side rendering for the generation of the static pages necessary to respect the JamStack approach. This is exactly the idea behind Gasby, scully or vuepress based on React, Angular and VueJS respectively.

- The huge interest of these frameworks is to have the best of both worlds between a static site *à la* Hugo or Jekyll and the use of Web components which ease the reuse off-the-shelf components from the Angular, VueJS, React communities or to develop easily your own Web components by benefiting from all the support of modern IDEs like VSCode for the use of these frameworks. It is the case of Gatsby based on React, VuePress based on VueJS and Scully based on angular.

Comparing static code generator approach

A second step: Gatsby, VuePress, Scully,

- The developer can then decide whether the content should be static and therefore pre-rendered when passing through the CI or dynamically calculated by consuming the data source when loading the page and benefit from Angular, React or VueJS for rendering.

Pros

- Excellent architecture
- Easy to develop, to maintain
- Could reuse lots of existing components
- Easy to create you own theme
- …

Cons

- When consuming static or dynamic data, JSON or other structured data are overused

A vision => Mulder: an extended Scully powered by DSLs

> Good heavens, to doubt it is more than a failure of the imagination, it's a failure to recognize the limits of our own stupidity. The nascency of our science, the rudiment of our tools. We listen, we search, we look for a sign as if our eyes and ears are good enough, our brains large enough, or egos small enough. *Cigarette Smoking Man*

- https://mulder-jamstack.github.io/

A vision => Mulder: an extended Scully powered by DSLs

Why scully ?

- I'm a fan of Angular over React or VueJS, I am not able to explain it, maybe it's because they built the framework on top of the TypeScript language features making the handling of the framework abstractions extremely simple and intuitive. As a result, I built *Mulder* on top of Scully.

Mulder: Empowering the JAMStack with DSLs

- The idea of *Mulder* is to provide a set of web components and scully plugins to consume data structured within DSLs that are parsed directly into TypeScript either at the time of passing through the CI for static data or at the time of loading the data at the time of loading the page for dynamic data. The scenario for the enduser is the following, any *txt* file hosted on github, gitlab, hackmd that can be updated on the fly by the user. This txt file is parsed on the fly by a combinatorial parser within the application in order to be consumed in the component it uses.

A vision => Mulder: an extended Scully powered by DSLs

- Mulder therefore offers sample components for Markdown, a survey markup language, an online conference agenda and provides documentation on developing your own DSLs.

- As a result, github, gitlab or hackmd could become your headless CMS. They manage for you authentification, role-based management, Web IDE for txt file editions, txt file history, … and Scully and Mulder can help you to design your front.

References

- Excellent web site on Jamstack
- Excellent book on Jamstack
- Excellent video introducing Jamstack

Learn More

Documentations · GitHub · Showcases