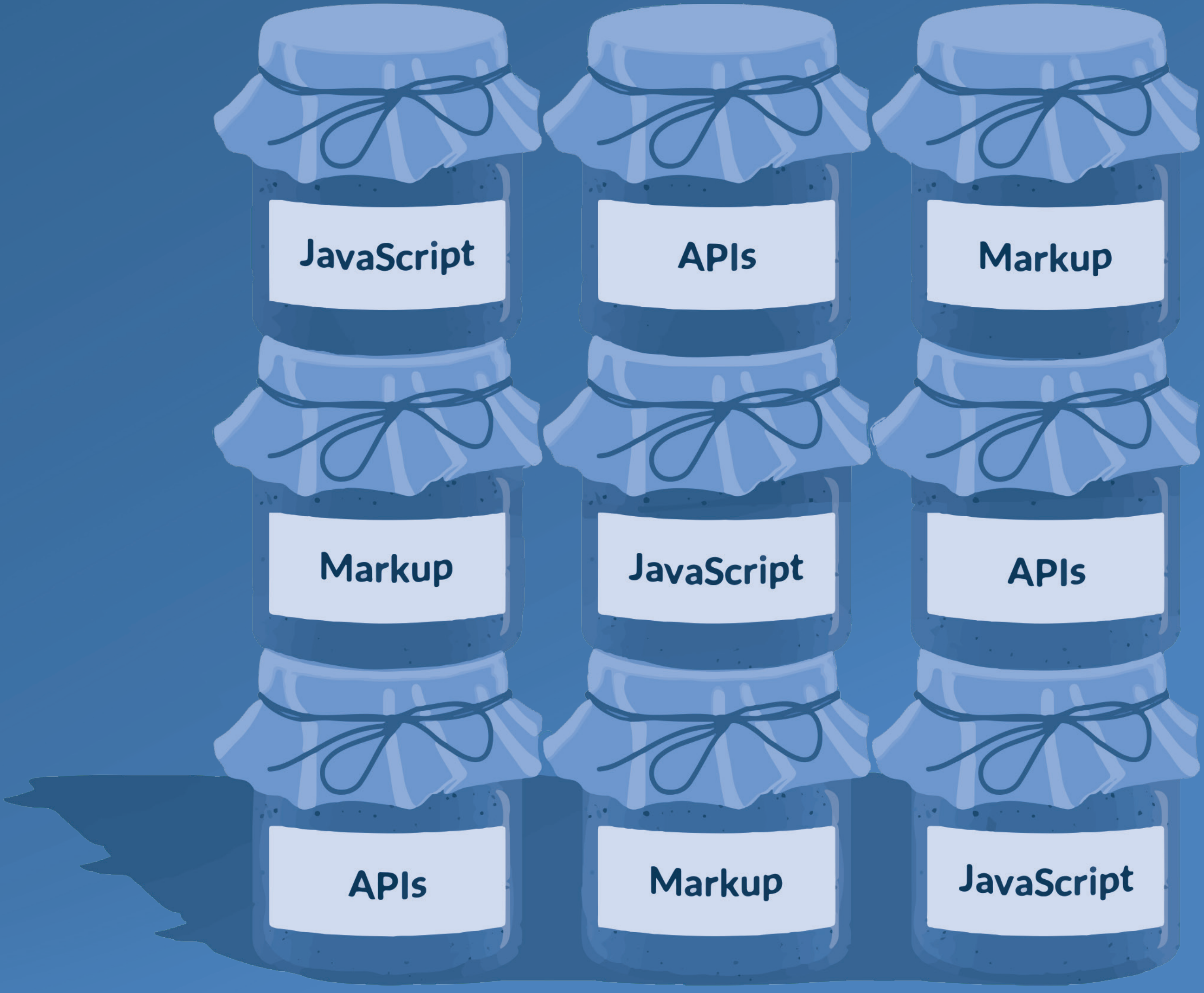
Scrivito

JAMSTACK FOR WEB PROJECTS



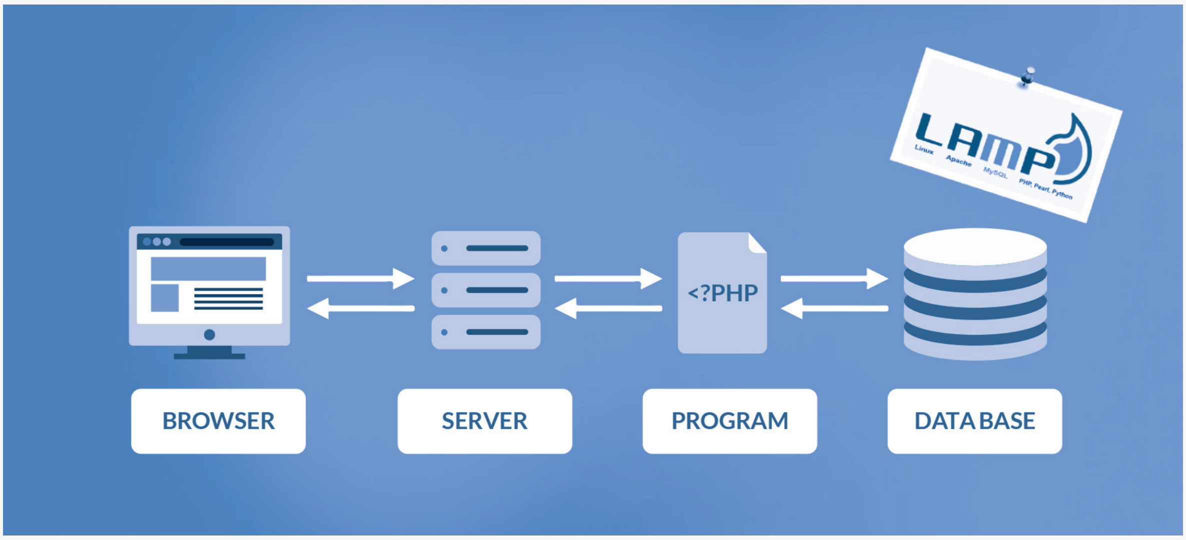
FASTER, MORE SECURE AND EASIER TO ADMINISTER

Scrivito CMS White Paper
New Enterprise SaaS CMS

JAMSTACK FOR WEB PROJECTS

FASTER, MORE SECURE AND EASIER TO ADMINISTER

In the early 2000s, web servers were developed and optimized in leaps and bounds to deliver HTML pages. These days, this has become a part of the problem. Server-based web technologies are no longer sufficiently performant, they are too complex, too expensive, too high-maintenance and too insecure. Over the past number of years a new, serverless technological approach has become established: Jamstack. This whitepaper explains why we can expect a lot more from this technology. 15 to 20 years ago, it made sense to locate as much work as possible on the webserver. User terminals were inefficient and at best suitable for displaying static HTML webpages. The server technology was mainly composed of Linux, Apache, MySQL, and PHP (LAMP stack). Not much has changed. When a user calls up a web page today, the file is processed and displayed in the browser after interaction between the database, the backend code, the server, the browser, and the cache. The server is doing all the work and the user has to wait for the server to finish. There may also be a load balancer involved, which redirects the page requests to one of several web servers.



An idea from the 90's: when a user requests a web page, the file is processed after interaction between the database, back-end code and server and then made available in the browser. However, the structure which was initially very simple has now become very complex due to many new website requirements.

The new challenge for websites

The static HTML pages of the early years have now become dynamic web applications. Instead of simple server infrastructure, a complex network of databases, business services, video and image services, and caching procedures is needed. The more complex this network of procedures is, the more cumbersome the performance becomes. It also means that the maintenance and security effort increases as the areas of interference and those open to attack expand.

Complex business requirements

E-commerce functionalities, mobile applications, product selection configurators, appointment coordination functions, language assistant integration and complete online contracting are all typical requirements today. For this, a large number of different content pools, product information, CRM and ERP systems must be integrated with ever-larger volumes of information. The application logic is integrated into the website and connected to processes in the back office. Subsequent processing steps need to be automated even further.

Performance problems

The performance of the website is critical to the success of the content provided. According to a Google analysis[1], 53% of visitors will leave a mobile website if it has not loaded after three seconds. However, 70% of the mobile landing pages analyzed took more than five seconds to display the visual content of the visible part of the webpage. Increasingly complex web applications, however, mean the loading time slows down ever further. While servers in the past only had to generate simple static HTML pages, database queries and various other interfaces are now required and HTML must be created on the fly for each visit. This is a much slower, more complex process than providing static content.

[1] https://www.thinkwithgoogle.com/marketing-resources/data-measurement/mobile-page-speed-new-industry-benchmarks

Extensive server infrastructure

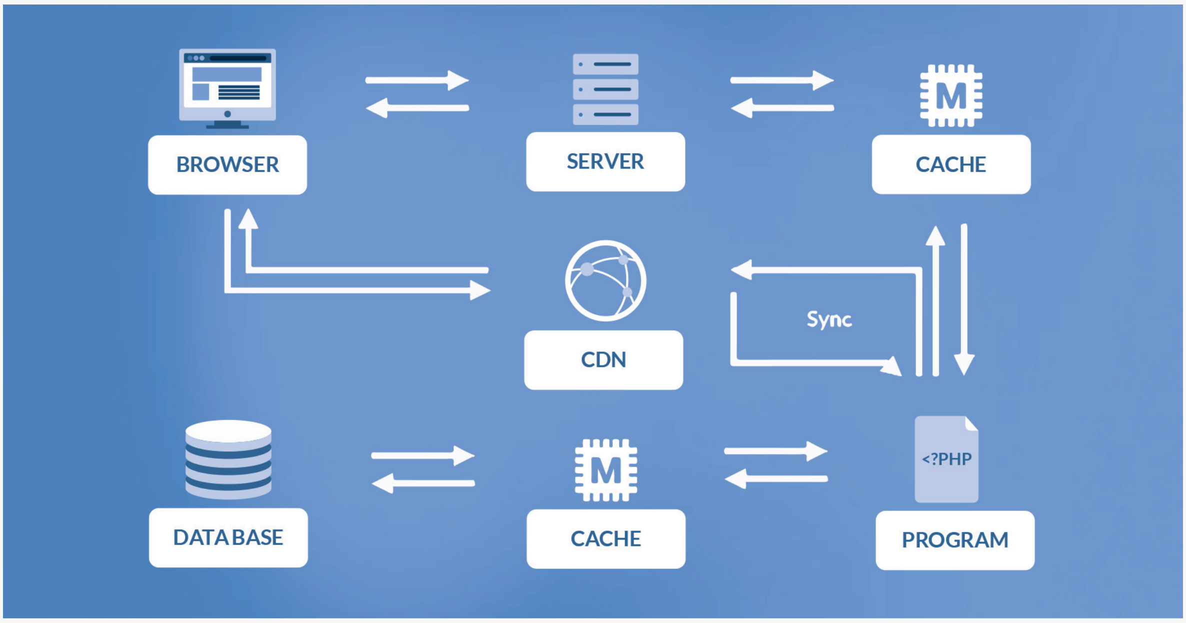
As a new page view must be generated and displayed for each new visitor to a page, efficient infrastructure is needed. It must also be powerful enough to perform well even during peak traffic. To ensure web availability, additional redundant servers, databases, and environments for development, testing, and production, etc. are required. This is relatively expensive as it was typically planned well in advance for an estimated peak usage. Additionally, all this equipment must also be administered and maintained.

Vulnerable applications

Traditional web applications are complex and vulnerable as they consist of many server components. Plug-ins from third parties are particularly vulnerable. They are directly connected to the core framework and can thus endanger the entire website. According to a study by WP WhiteSecurity[2], over 70% of all WordPress installations on the internet are vulnerable. Frequent security patches (542 patches for WordPress alone in 2018)[3] require a considerable administrative effort. Maintenance costs are among the main cost drivers of traditional content management systems.

[2] https://www.wpwhitesecurity.com/statistics-70-percent-wordpress-installations-vulnerable/
[3] Nadav Avital, Imperva, Blog, “The State of Web Application Vulnerabilities in 2018”, January 2019

THE SERVER-BASED ARCHITECTURE HAS BECOME TOO COMPLEX



In order to keep dynamic web applications secure and performant, server-based architecture has become very complex.

Databases

Simple default configurations for databases no longer meet current performance requirements. This requires a lot of action and administration (tuning, redundancies, migration in the CMS, schema adjustment, backups, patches, updates, indexing, etc.).

Cache systems

Without caching, performance is poor. The big challenge is to keep the cache consistent and valid across multiple servers and redundant databases. This required a whole cascade of action (database replication, content consistency, cache invalidation, etc.).

Application server

This requires a high degree of administration (back-ups, keeping security plug-ins up-to-date, etc.).

CDN

Without Content Delivery Networks (CDN), large videos and images are not readily available. These large files take time to download and the further they are away from the end-user the longer the process takes. Conventional CMSs are not designed for this, which means increased effort (keeping content in sync, versioning, etc.) and poor performance.

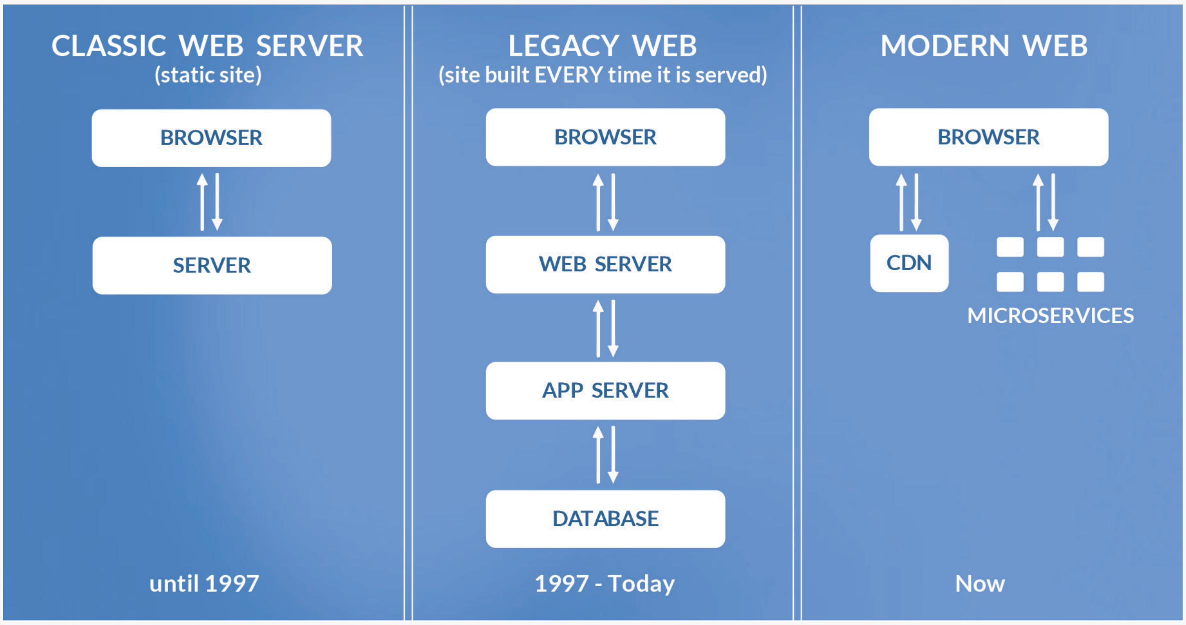
Server

The issues of scaling, redundancy for servers and load balancers, development, test and production environments, firewall configuration, patches, updates, etc. are all time-consuming and costly.

THE TREND

JAMSTACK IS THE NEW DEVELOPMENT ARCHITECTURE

The Jamstack approach (JavaScript, API, and pre-rendered HTML markup) is just a few years old. It is a new method of creating websites and applications. Content is no longer processed and generated afresh for every visitor on the server. It is generated locally in the browser as Jamstack pages. Dynamic applications can also be processed in the browser thanks to JavaScript and APIs.



The Jamstack approach through JavaScript, Web APIs, microservices, CDN and pre-rendering greatly simplifies the architecture, improves performance, and reduces costs.

Relocation from the server to the browser

The former principle of loading as much of the work as possible onto the webserver is outdated. Today's user devices have more than enough resources to run web applications. This also includes mobile devices.

JavaScript frameworks

Google, Facebook, and others have developed completely new web frameworks based on JavaScript, such as the JavaScript library React, and then released them as open source. React has been the most popular and fastest-growing JavaScript framework since its first application in 2011. The use of such web frameworks creates a new development architecture, where applications are no longer tied to specific operating systems or web servers.

Modern web APIs

Modern web APIs can be integrated for any JavaScript client to implement third-party services such as business applications with access to CRM and ERP systems as well as eCommerce functionalities such as payments and subscriptions. Utilizing web APIs can eliminate the need for additional servers.

Microservices architecture

Because web functions no longer need to be managed in the server, websites can be designed around microservices. A microservice undertakes a narrowly defined task which is initiated, carried out and terminated independently of other microservices.

Current browser technologies

Jamstack relocates the logic from the server to the browser, where the pages are dynamically generated through APIs and microservices. Modern browsers can interact with many APIs using JavaScript, as well as execute complex and dynamic applications.

Decoupled architecture

The tasks carried out at the backend, such as creating, managing and saving content, are separated from the presentation on a frontend device (so the architecture is headless, or decoupled). Once it has been created, content can be used for any device. The functionality taking place in the frontend and the deployment of individual APIs can now be handled in isolation.

Pre-rendering of HTML markup

In the Jamstack approach, the website HTML is no longer generated by traditional frontend web servers, but rather the page is preconfigured, distributed via a Content Delivery Network (CDN), and displayed in the user's browser. All other activity takes place in the browser because the pages contain JavaScript code that access APIs and is executed after the page is rendered.

PARTICULAR CHALLENGES FOR THE JAMSTACK APPROACH

The transition from the server-oriented LAMP stack approach to the new Jamstack approach has proven to be a practicable concept. In recent years, a large number of Jamstack web projects have been implemented. The concept applies to smaller websites as well as complex web applications with thousands of pages. However, this conversion process should not be underestimated, because a completely new architecture is used.

Stepping into a new development environment

The switch to Jamstack is initially associated with a degree of uncertainty. A new development paradigm is being introduced, with new rules and new processes. This conversion usually takes a number of months and initially requires operating parallel structures. However, this is nothing new for IT, and there are extensive best practices available for such a process. Additionally, as the process is better understood, the conversion speeds up drastically, saving vast amounts of time in the end.

Bezos's "API Manifesto"

An internal email written by Jeff Bezos[4] in 2002 is legendary: all of the teams at Amazon received the binding instruction to make their data and functions available only via service interfaces in the future. Without exception. The transition to these formal APIs made life difficult for employees in the short term. However, Amazon was able to operate its systems much more efficiently and it enabled, for example, the launch of publicly available Amazon Web Services.

Know-how and change processes

The construction of the new architecture means that new know-how about JavaScript, APIs, microservices, etc. is also required by employees. The existing, familiar system architecture is to be replaced by a future-oriented cloud alignment.

[4] Source: Amazon Web Services (AWS), "Global Infrastructure", August 2019

Communicate with web APIs

Using Jamstack means that every piece of information must be available via an API. In a traditional web server environment, a program running in a standard browser would not have the ability to communicate with a web API outside its own domain. Only with the availability of full-featured JavaScript frameworks and new standards such as authorization and stateless authentication was it possible to reach any modern web API from any JavaScript client. This includes, for example, the RESTful API for communication between different services.

Authentication, Authorization, and Accounting (AAA)

A key to new and secure client-side authentication is the enforcement of the OpenID Connect open standard. Technologies and identity providers such as OAuth and Auth0 enable a central single sign-on, which is now supported by all major platforms.

Ecosystems for development environments

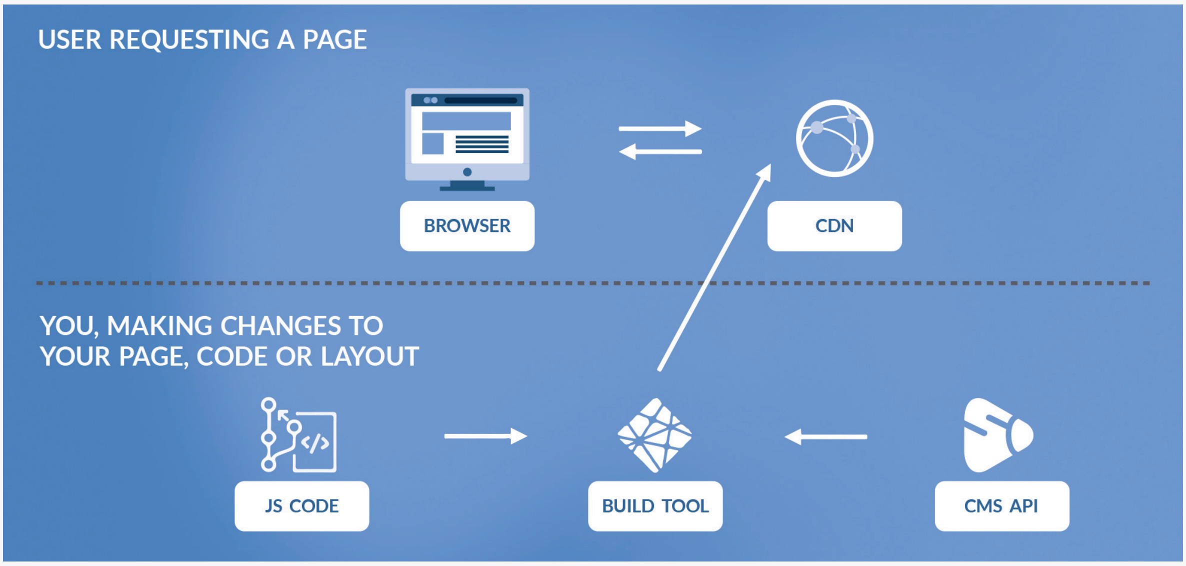
In addition to developing new web frameworks, the Jamstack approach requires more web components. By separating the backend and frontend, frontend architectures, browser APIs, HTML and CSS standards could evolve independently. For example, entire ecosystems are developed for ready-made APIs for authentication, for e-commerce, search and so on, or large libraries for very specialized and reusable microservices.

Limitations of pre-rendering

There were initially limitations in pre-rendering HTML markup for too many pages. Because of its size, the build and deploy cycle was very long. In the meantime, rendering tools have become so fast that thousands of pages can be processed within minutes. The advantage of cloud-based structures is also the possibility of parallelization through Lamda processes. For example, Amazon Web Services (AWS) can render thousands of pages simultaneously in up to a thousand simultaneous processes. In addition, content management systems such as Scrivito have the ability to update content after prerendering to include the latest changes to all content, so that the delivered pages are always up to date.

Content Delivery Network (CDN)

A geographically distributed CDN with locations close to the user delivers the ready-made content of the pages very quickly and ensures good performance. Crucially, the CDN has a sufficiently large number of distribution and end nodes with redundant locations.



A globally distributed CDN, with locations close to the user, delivers the ready-made content of the pages very quickly and ensures good performance. Native CMS cloud solutions like Scrivito have already integrated the CDN requirements conceptually.

JAMSTACK FOR CONTENT MANAGEMENT SYSTEMS

Jamstack pages can be controlled via a Content Management System (CMS) known as a headless / decoupled CMS (where the backend and frontend are separate). The CMS backend application can be completely relocated to the cloud, eliminating the need to operate, administer and configure your own servers or runtime environments. This gives users the opportunity to focus more than ever on the content and the business logic required for it. However, this assumes that the CMS supports more than just pure writing and reading processes.

Flexible and extendable

The Jamstack logic allows the flexible implementation of any business requirement using JavaScript, React, web APIs and microservices. However, the CMS must support such individual extensions and changes, e.g. by mapping specific tasks to self-developed or pre-built components.

Ready-made components

Headless / decoupled CMSs like Scrivito permit users to implement special functions using ready-made components (widgets). These widgets range from building block elements for headlines, lists, images, etc. to functional modules for user interfaces, e.g. forms. These sophisticated, tested components already possess a high degree of maturity, which significantly reduces development and maintenance time.

Pre-rendering images

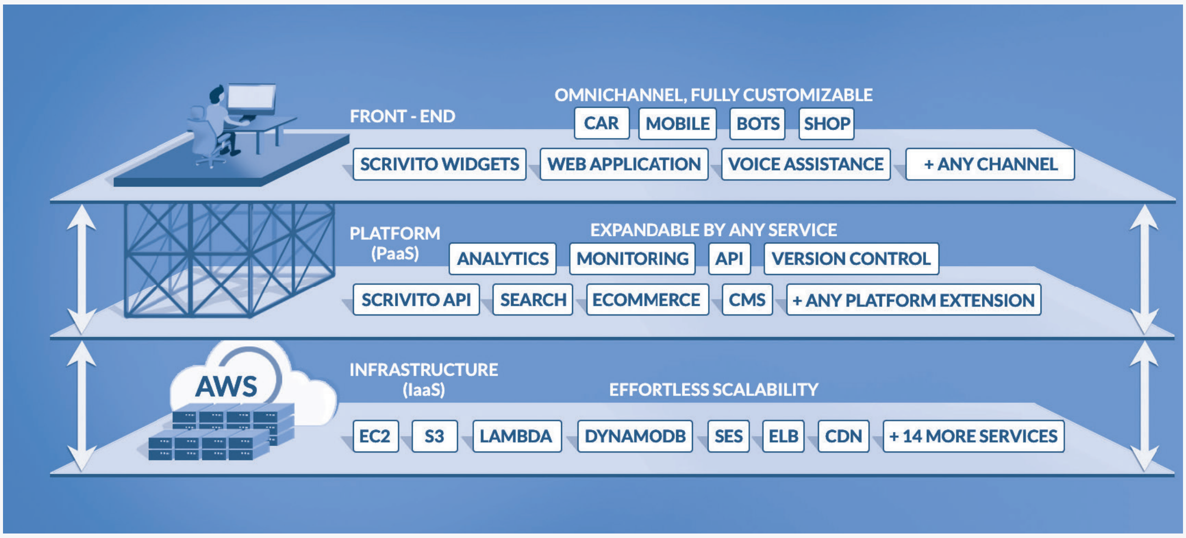
With the increasing variety of devices in use, the need for adaptable content presentation increases. By linking a headless / decoupled CMS with a Content Delivery Network (CDN), you can, for example, have multiple resolutions of an image prescaled and provided via the CDN for any devices as a standard feature.

Content management logic

Due to increasingly complex business functions, a confusing variety of content elements can quickly arise. However, outdated and poorly maintained items such as price lists can lead to significant problems. Therefore, CMS functions on versioning, with rules on release, maintenance and updates are required.

100% cloud-based solution

The cloud infrastructure enables elastic scalability for traffic peaks and seasonal load fluctuations such as Christmas traffic. Native CMS cloud solutions also make a “CMS as a service” possible with billing only for actual use (pay-per-use model).



Headless / decoupled CMSs like Scrivito are natively designed for the cloud. They leverage available resources and web services such as big data, next-generation technologies (e.g. voice search, machine learning, AI), and cloud services (e.g. Lambda functions, Amazon S3, Amazon EC2, DynamoDB, CloudFront or the Amazon API Gateway).

CONCLUSION

JAMSTACK IS FASTER, MORE SECURE AND CHEAPER

The Jamstack approach has several key advantages over the traditional server infrastructure. It enables higher performance, improves security, lowers costs, scales more easily, and delivers a better user experience.

Simplified architecture

The complex requirements of web servers, load balancers, local caching mechanisms, capacity planning and various redundancy layers are replaced by a drastically simplified Jamstack architecture. The execution logic is relocated from the server to the browser. Performance and availability are thus decoupled from the server infrastructure.

Better performance

The user no longer has to wait for the server to finish working on increasingly complex web applications. Now, ready-made HTML pages can be delivered very quickly via a CDN. With interactive pages and additional user requests, individual microservices deliver more results without having to completely reload the page. What is also hugely relevant is that better performance is an important SEO criterion for search engine ranking.

More business efficient

JavaScript, APIs, and microservices enable new business capabilities. Web applications become dynamic and the services become interactive. Self-created or pre-packaged microservices provide required business functions such as product and tariff comparisons or self-service capabilities.

Higher availability

The simple architecture and the delivery of static HTML files in conjunction with JavaScript lead to very high levels of stability and security. A global CDN ensures reliable availability at all times of the day. Making use of a CDN is another advantage over having your own server infrastructure as a single point of failure (SPoF).

Greater security

Instead of vulnerable servers and insecure plug-ins, the drastically simplified Jamstack architecture reduces attack potential. Significantly less code (low code) simplifies quality monitoring and increases stability. Individual microservices in the web browser run independently and no longer jeopardize the overall system.

Lower operating costs

The new structure has considerable operational and maintenance advantages since servers no longer need to be operated or maintained. Software updates and security patches are eliminated. A high degree of prefabricated and reusable functional widgets also reduces the development effort. Finally, a cloud-based CMS is only billed on the basis of actual usage.

Better user experience

The performance of Jamstack also gives visitors an app-like user experience on websites. Speed and interactivity, as SEO-relevant properties, are at a higher level both when a site is first called up and during navigation between pages. A better user experience is a measurable revenue factor, as the bounce rate gets smaller and the greater interactivity increases the retention time.

JAMSTACK IS SETTING TRENDS AND BEING DEVELOPED FURTHER

The Jamstack approach, with a JavaScript library such as React, web API, and pre-rendered HTML, continues to thrive, in web applications in particular. For example, React was originally developed for Facebook and is now used with Netflix, AirBnB, and Instagram. With React Native, a variant for the development of mobile Android and iOS apps is now available. Another future-oriented area is the use of React 360 for 3D, virtual and augmented reality applications. Finally, a headless / decoupled CMS benefits from the Jamstack distribution capabilities for many new devices and the implementation of dynamic and interactive web applications.

Scrivito

JustRelate Group GmbH, Kitzingstraße 15, 12277 Berlin, Germany
www.scrivito.com

D-21-236905, Version 1