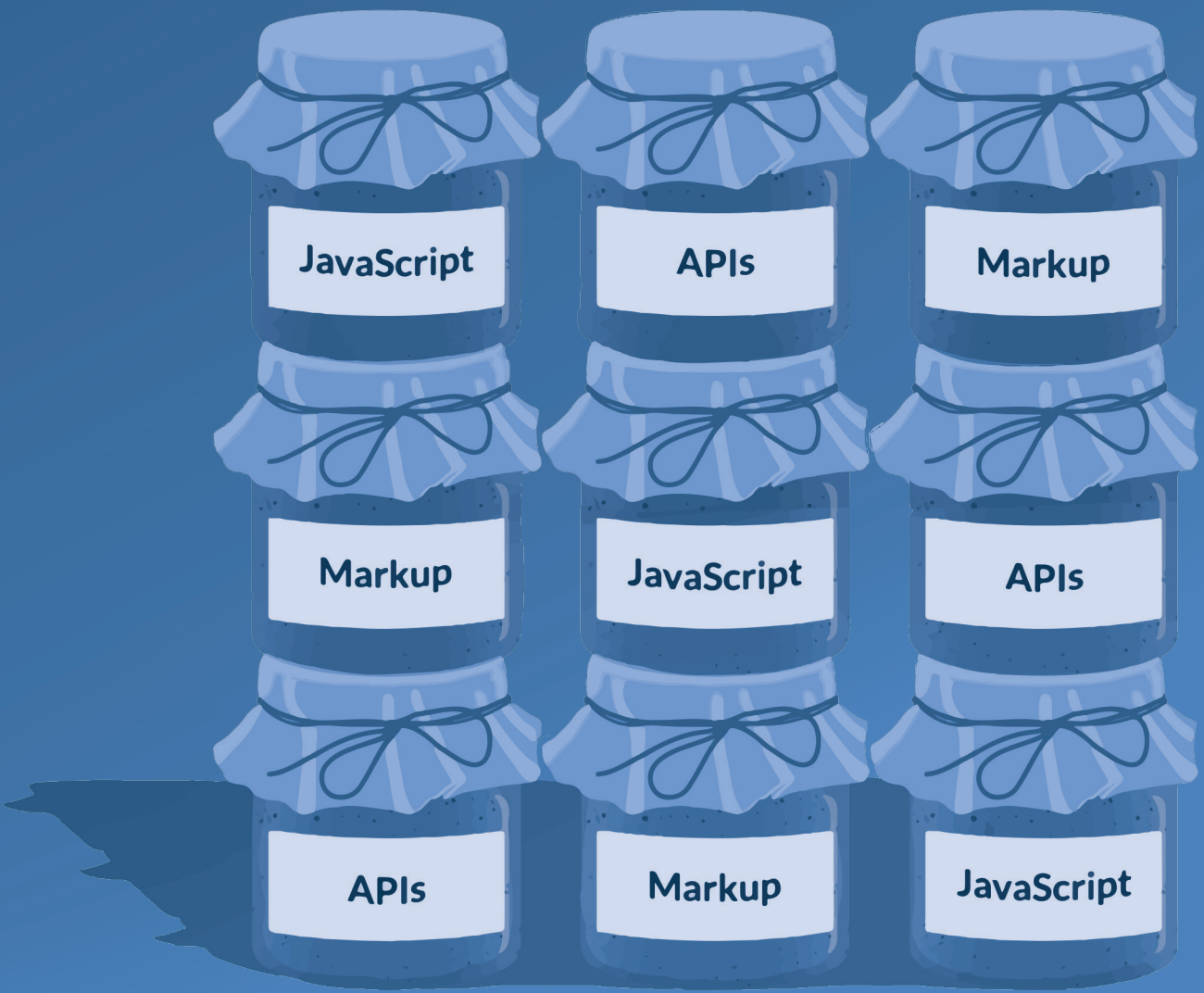
Scrivito

# JAMSTACK FÜR WEBSEITEN




## SCHNELLER, SICHERER UND EINFACHER ZU ADMINISTRIEREN

Scrivito CMS White Paper
New Enterprise SaaS CMS

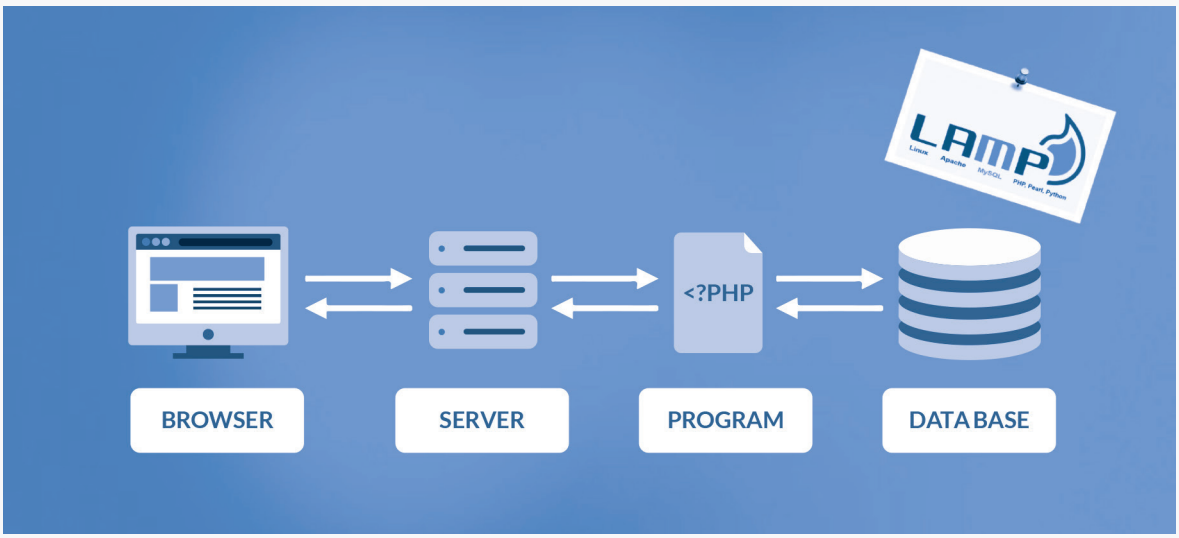
# JAMSTACK FÜR WEBSEITEN

## SCHNELLER, SICHERER UND EINFACHER ZU ADMINISTRIEREN

In den frühen 2000er Jahren wurden Webserver für die Auslieferung von HTML-Seiten massiv weiterentwickelt und optimiert. Heute sind sie eher ein Teil des Problems. Die serverbasierte Webtechnologie ist nicht mehr ausreichend performant, zu komplex, zu teuer, zu wartungsaufwendig und zu unsicher. Seit einigen Jahren setzt sich eine neuer, serverloser Technologie-Ansatz durch: Jamstack. Dieses Whitepaper erläutert, warum wir von dieser Technologie noch sehr viel erwarten können.

Vor 15 bis 20 Jahren war es sinnvoll, möglichst viel Last auf die Webserver zu verlagern. Die Endgeräte waren leistungsschwach und bestenfalls geeignet, um statische HTML-Webseiten anzuzeigen. Die Server-Technologie setzte sich im Wesentlichen aus Linux, Apache, MySQL und PHP zusammen (LAMP-Stack). Daran hat sich nicht viel geändert. Wenn ein Benutzer heute eine Webseite anfordert, wird die Datei nach vielen Interaktionen zwischen Datenbank, Backend-Code, Server, Browser und Cache-Ebenen verarbeitet und im Browser bereitgestellt. Der Server macht die Arbeit und der Benutzer muss warten, bis der Server fertig ist. Vielleicht ist noch ein Load Balancer beteiligt, der die Seiten-Anforderungen an einen von mehreren Webservern weiterleitet.






Eine Idee aus den 90er Jahren: Wenn ein Benutzer eine Webseite anfordert, wird die Datei nach der Interaktion zwischen Datenbank, Backend-Code und Server verarbeitet und dann im Browser zur Verfügung gestellt. Die anfänglich sehr einfache Struktur ist jedoch aufgrund vieler neuer Anforderungen an die Website inzwischen sehr komplex geworden.

## Die neuen Herausforderungen für Websites

Aus den statischen HTML-Seiten der Anfangsjahre sind inzwischen dynamische Web-Applikationen geworden. Anstelle einer einfachen Server-Infrastruktur wird ein komplexes Geflecht aus Datenbanken, Business-Diensten, Video-Angeboten und Caching-Prozeduren verarbeitet. Je komplexer dieses Geflecht ist, umso schwerfälliger ist die Performance, umso höher ist der Aufwand für die Wartung und umso mehr Störquellen und Angriffsflächen müssen abgesichert werden.

## Komplexe Business-Anforderungen

E-Commerce-Funktionalitäten, mobile Anwendungen, Konfiguratoren zur Produktauswahl, Funktionen für die Terminkoordination, Einbindung von Sprachassistenten bis hin zu kompletten Online-Vertragsabschlüssen sind heute typische Anforderungen. Dafür müssen eine Vielzahl unterschiedlicher Content-Pools, Produktinformations-, CRM- und ERP-Systeme mit immer größeren Informationsmengen eingebunden werden. Die Applikationslogik ist in die Website integriert und mit den Bearbeitungsprozessen im Backoffice verbunden. Nachfolgeprozesse und folgende Bearbeitungsschritte sollen immer weiter automatisiert werden.

## Probleme mit der Performance

Die Leistung der Website ist von entscheidender Bedeutung für den Erfolg der bereitgestellten Inhalte. Laut einer Google-Analyse[1] verlassen 53% der Besucher eine mobile Website, wenn sie nicht nach drei Sekunden geladen wurde. Allerdings dauerte es bei 70% der analysierten mobilen Zielseiten mehr als fünf Sekunden, bis der visuelle Inhalt des sichtbaren Teils einer Webseite angezeigt wurde. Durch die immer komplexeren Web-Applikationen verlangsamt sich allerdings die Ladezeit. Während die Server früher nur einfache statische HTML-Seiten erstellen mussten, sind jetzt Datenbankabfragen und diverse andere Schnittstellen zu bedienen und für jeden Besuch muss das HTML "on the fly" erstellt werden. Dies ist ein viel langsamerer, komplexerer Prozess als die Bereitstellung statischer Inhalte.

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

## Aufwendige Server-Infrastruktur

Da für jeden Besucher ein neuer Seitenaufruf generiert und ausgeliefert wird, erfordert dies eine leistungsfähige Infrastruktur. Sie muss zudem ausreichend groß ausgelegt sein, um auch bei Verkehrsspitzen performant zu funktionieren. Zur Absicherung der Verfügbarkeit sind zusätzliche redundante Server, Datenbanken, Entwicklungs-, Test- und Produktionsumgebungen usw. notwendig. Das ist relativ teuer und muss auch verwaltet werden.

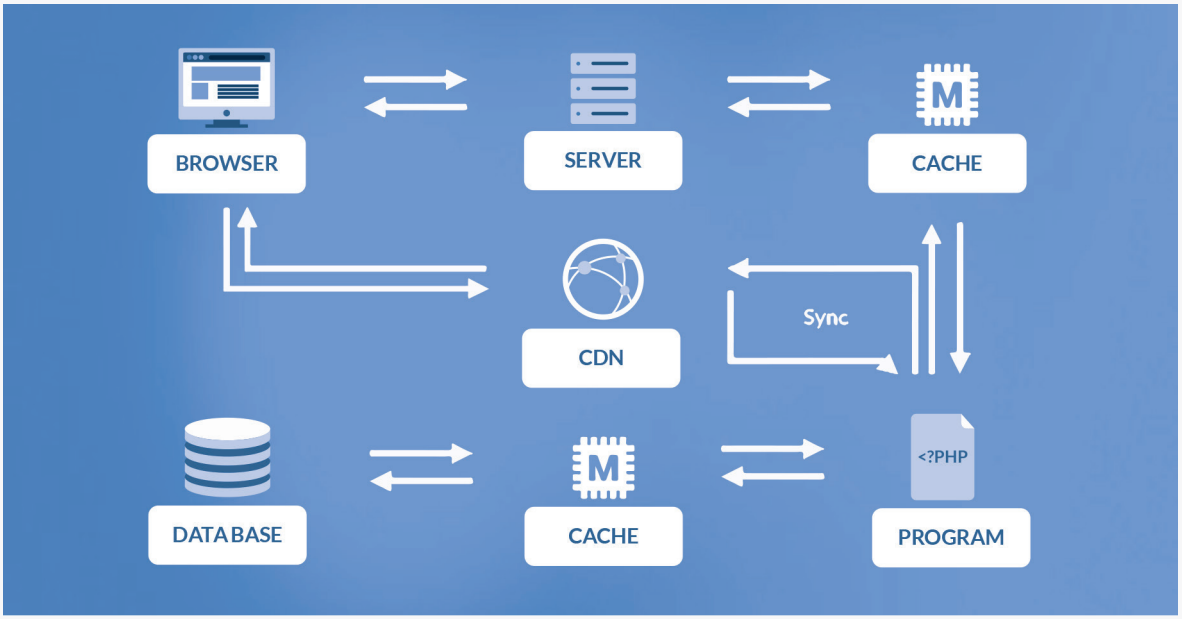
## Angreifbare Applikationen

Herkömmliche Web-Applikationen bestehen aus vielen Serverkomponenten, sind komplex und dadurch leicht verwundbar. Vor allem die Plug-ins von Drittanbietern zeigen viele Schwächen. Sie sind direkt an das Core-Framework angebunden und können die gesamte Website gefährden. Über 70% aller WordPress-Installationen im Netz sind nach einer Studie von WP WhiteSecurity[2] angreifbar. Die sehr häufigen Sicherheits-Patches (allein 542 Patches für Wordpress in 2018[3]) führen zu einem beträchtlichen Administrationsaufwand. Diese Wartungskosten gehören zu den wesentlichen Kostentreibern herkömmlicher Content Management Systeme.



[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

# DIE SERVERBASIERTE ARCHITEKTUR IST ZU KOMPLEX GEWORDEN




Um dynamische Webanwendungen sicher und performant zu halten, ist die serverbasierte Architektur sehr komplex geworden.

## Datenbanken

Einfache Standardkonfigurationen für Datenbanken entsprechen nicht mehr den aktuellen Leistungsanforderungen. Dies erfordert einen hohen Handlungs- und Verwaltungsaufwand (Tuning, Redundanzen, Migration im CMS, Schema-Anpassung, Backups, Patches, Updates, Indizierung usw.).

## Cache-Systeme

Ohne Caching ist die Leistung schlecht. Die große Herausforderung besteht darin, den Cache über mehrere Server und redundante Datenbanken hinweg konsistent und gültig zu halten. Dies erfordert eine ganze Kaskade von Maßnahmen (Datenbankreplikation, Inhaltskonsistenz, Cache-Invalidierung usw.).

## Application Server

Application Server erfordern ein hohes Maß an Verwaltung. (Back-ups, Sicherheits-Plug-ins auf dem neuesten Stand halten, usw.).

## CDN

Ohne Content Delivery Networks (CDN) sind große Videos und Bilder nicht ohne weiteres verfügbar. Das Herunterladen dieser großen Dateien nimmt Zeit in Anspruch, und je weiter sie vom Endbenutzer entfernt sind, desto länger dauert der Prozess. Herkömmliche CMS sind dafür nicht ausgelegt, was einen erhöhten Aufwand (Synchronisierung der Inhalte, Versionierung usw.) und eine schlechte Leistung bedeutet.

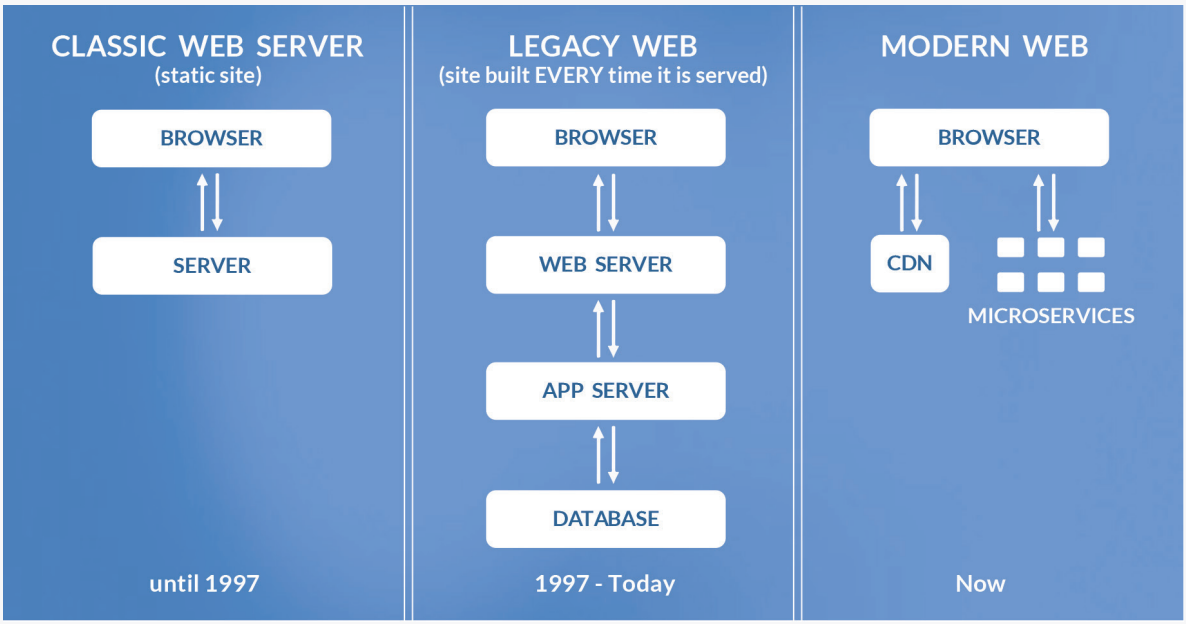
## Server

Die Themen Skalierung, Redundanz für Server und Load Balancer, Entwicklungs-, Test- und Produktionsumgebungen, Firewall-Konfiguration, Patches, Updates usw. sind allesamt zeit- und kostenintensiv.

# DER TREND

## JAMSTACK ALS NEUE ENTWICKLUNGSARCHITEKTUR

Der Jamstack-Ansatz (JavaScript, API und vorgerendertes HTML-Markup) ist erst wenige Jahre alt. Es ist eine neue Art, wie Websites und Applikationen erstellt werden. Die Inhalte werden nicht mehr auf dem Server ausgeführt und für jeden Aufruf neu generiert, sondern als Jamstack-Seiten lokal im Browser ausgeführt. Dank JavaScript und APIs können im Browser auch dynamische Applikationen verarbeitet werden.




Der Jamstack-Ansatz durch JavaScript, Web-APIs, Mikrodienste, CDN und Pre-Rendering vereinfacht die Architektur erheblich, verbessert die Leistung und reduziert die Kosten.

### Verlagerung vom Server in den Browser

Der alte Grundsatz, möglichst viel Last auf die Webserver zu verlagern, ist überholt. Die Endgeräte stellen heute ausreichend Leistungsressourcen für die Ausführung von Web-Applikationen zur Verfügung. Das schließt auch mobile Endgeräte ein.

### JavaScript-Frameworks

Google, Facebook u.a. haben auf Basis von JavaScript komplett neue Web-Frameworks wie die JavaScript-Bibliothek React entwickelt und später als Open Source freigegeben. React ist seit seiner ersten Anwendung 2011 bis heute das beliebteste und am schnellsten wachsende JavaScript-Framework. Durch die Verwendung solcher Webframeworks wird eine

neue Entwicklungsarchitektur geschaffen, in der Anwendungen nicht mehr an bestimmte Betriebssysteme und Webserver gebunden sind.

## Moderne Web-APIs

Von jedem JavaScript-Client aus können moderne Web-APIs eingebunden werden, um E-Commerce-Funktionalitäten wie Zahlungen oder Abonnements, Business-Applikationen mit Zugriff auf CRM- und ERP-Systeme oder beliebige Dienste von Drittanbietern zu realisieren. Für die Bereitstellung der Web-APIs sind keine Webserver mehr notwendig.

## Serviceorientierte Microarchitektur

Weil die Web-Funktionen nicht mehr im Server verwaltet werden müssen, können Websites um Microservices herum gestaltet werden. Ein Microservice übernimmt eine eng definierte Aufgabe und wird unabhängig von anderen Microservices aufgerufen, ausgeführt und wieder beendet.

## Aktuelle Browser-Technologien

Durch Jamstack wird die Logik vom Server in den Browser verlagert, wo die Seite über APIs und Microservices dynamisch ausgeführt werden. Moderne Browser können auf der Basis von JavaScript mit vielen APIs interagieren sowie komplexe und dynamische Anwendungen ausführen. Die Browser werden so zu Mini-Betriebssystemen.

## Entkoppelte Architektur

Die Aufgaben im Backend (Inhalte erstellen, verwalten und speichern) werden von der Darstellung auf einem Endgerät (Frontend) getrennt (Headless oder auch Decoupled). Der einmal erstellte Inhalt kann für beliebige Devices genutzt werden. Die Funktionen im Frontend und die Bereitstellung einzelner APIs können jetzt isoliert voneinander betrachtet werden.

## Pre-Rendering von HTML-Markup

Im Jamstack-Ansatz wird das HTML der Website nicht mehr von traditionellen Frontend-Webservern erzeugt, sondern die Seite wird vorkonfiguriert, über ein Content Delivery Network (CDN) verteilt und im Browser des Benutzers angezeigt. Alle weiteren Aktivitäten finden im Browser statt, denn die Seiten enthalten JavaScript und React-Code, die auf APIs zugreifen und nach dem Rendern der Seite ausgeführt werden.

# BESONDERE HERAUSFORDERUNGEN FÜR DEN JAMSTACK-ANSATZ

Der Übergang vom serverorientierten LAMP-Stack zum neuen Jamstack-Ansatz hat sich als praxistaugliches Konzept bewährt. In den letzten Jahren wurde eine Vielzahl von Jamstack-Webprojekten umgesetzt. Das Konzept setzt sich sowohl bei kleineren Websites als auch bei komplexen Web-Anwendungen mit Tausenden von Seiten durch. Dieser Umstellungsprozess darf aber auch nicht unterschätzt werden, weil hier eine komplett neue Architektur zum Einsatz kommt.

## Einstieg in eine neue Entwicklungsumgebung

Der Umstieg auf Jamstack ist zunächst mit einem Unsicherheitsfaktor verbunden. Es wird ein neues Entwicklungsparadigma eingeführt mit neuen Regeln und neuen Prozessen. Diese Umstellung dauert in der Regel mehrere Jahre und erfordert zunächst den Betrieb von parallelen Strukturen. Das ist für die IT aber nichts Neues und es liegen umfangreiche Best Practices für einen solchen Prozess vor.

## Bezos's „API Manifesto"

Eine interne Mail von Jeff Bezos[4] aus dem Jahr 2002 ist legendär: Alle Teams bei Amazon erhalten darin die verbindliche Anweisung, ihre Daten und Funktionen künftig nur noch über Serviceschnittstellen zur Verfügung zu stellen. Ohne Ausnahmen. Die Umstellung auf diese formalen APIs machte den Mitarbeitern zwar kurzfristig das Leben schwerer, aber Amazon war dadurch in der Lage, seine Systeme wesentlich effizienter zu betreiben und ermöglichte u.a. den Start der öffentlich zugänglichen Amazon Web Services.

## Know-how und Change-Prozesse

Mit dem Aufbau der neuen Architektur ist auch bei den Mitarbeitern neues Know-how zu JavaScript, APIs, Microservices usw. gefordert. Die bestehende und vertraute Systemarchitektur wird durch eine zukunftsorientierte Cloud-Ausrichtung abgelöst.

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

## Mit Web-APIs kommunizieren

Bei Jamstack muss jede Information per API verfügbar sein. Zunächst gab es im traditionellen Webserver-Umfeld für ein Programm, das in einem Standardbrowser ausgeführt wurde, keine Möglichkeit, mit einer Web-API außerhalb der eigenen Domain zu kommunizieren. Erst mit der Verfügbarkeit von JavaScript-Frameworks als vollwertige Laufzeitumgebung und neuen Standards, beispielsweise für Autorisierung und zustandsloser Authentifizierung, war es möglich, jede moderne Web-API von jedem JavaScript-Client aus zu erreichen. Dazu gehört u.a. die Anwendungsprogrammschnittstelle RESTful API zur Kommunikation zwischen verschiedenen Diensten.

## Authentifizierung, Autorisierung und Accounting (AAA)

Ein entscheidender Schlüssel für eine neue und sichere Client-seitige Authentifizierung ist die Durchsetzung des offenen Standards OpenID Connect. Technologien und Identitätsanbieter wie OAuth und Auth0 ermöglichen einen zentralen Single Sign-On, was inzwischen von allen großen Plattformen unterstützt wird.



## Ökosysteme für Entwicklungsumgebungen

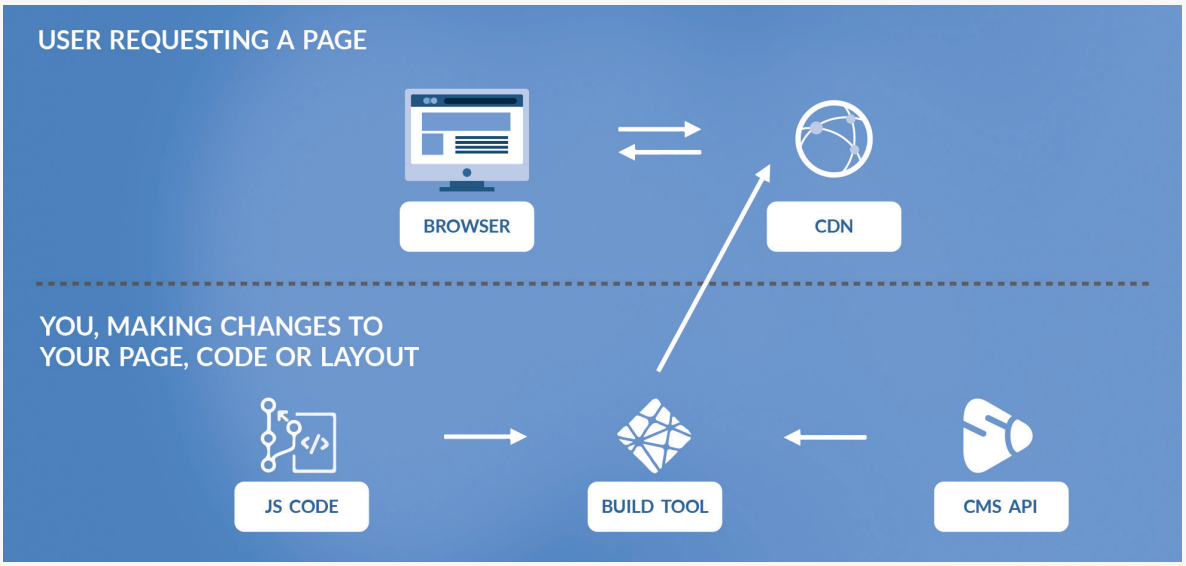
Neben der Entwicklung neuer Web-Frameworks erforderte der Jamstack-Ansatz weitere Web-Komponenten. Durch die Trennung von Backend und Frontend konnten sich Frontend-Architekturen, Browser-APIs, HTML- und CSS-Standards unabhängig voneinander weiterentwickeln. So haben sich z. B. ganze Ökosysteme für vorgefertigte APIs zur Authentifizierung, für E-Commerce, die Suche usw. entwickelt und auch umfangreiche Bibliotheken für sehr spezielle und wiederverwendbare Microservices.

## Einschränkungen beim Pre-Rendering

Zunächst gab es beim Pre-Rendering des HTML-Markups Einschränkungen bei einer zu großen Anzahl von Seiten. Durch die Größe wurde der Erstellungs- und Bereitstellungszyklus sehr lang. Inzwischen sind die Tools für das Rendering der Seiten so schnell geworden, dass auch Tausende von Seiten innerhalb weniger Minuten verarbeitet werden können. Der Vorteil Cloud-basierter Strukturen ist zudem die Möglichkeit der Parallelisierung durch Lambda-Prozesse. So können bei Amazon Web Services (AWS) in bis zu Tausend Prozessen gleichzeitig Tausende von Seiten parallel gerendert werden. Zusätzlich nutzen Content Management Systeme wie Scrivito von Infopark die Möglichkeit, einzelne APIs in den vorgefertigten Seiten nach dem Aufruf sofort auf Änderungen zu überprüfen, um z. B. News zu aktualisieren.

## Content Delivery Network (CDN)

Ein geografisch weltweit verteiltes CDN mit Standorten in der Nähe der Benutzer liefert die vorgefertigten Inhalte der Seiten sehr schnell aus und sorgt für eine gute Performance. Entscheidend ist dabei, dass das CDN über eine ausreichend große Anzahl an Verteil und Endknotenpunkten mit redundanten Standorten verfügt.




Ein global verteiltes CDN mit Standorten in der Nähe des Benutzers liefert die fertigen Inhalte der Seiten sehr schnell und sorgt für eine gute Performance. Native CMS-Cloud-Lösungen wie Scrivito haben die Anforderungen des CDN konzeptionell bereits integriert.

# JAMSTACK FÜR CONTENT MANAGEMENT SYSTEME

Jamstack-Seiten können über ein Content Management System (CMS) gesteuert werden, das als Headless/ Decoupled CMS bezeichnet wird (Backend und Frontend sind getrennt). Die CMS-Backend-Anwendung kann vollständig in die Cloud verlagert werden, so dass keine eigenen Server oder Laufzeitumgebungen mehr zu betreiben, zu administrieren und zu konfigurieren sind. So haben die Anwender die Chance, sich stärker denn je auf die Inhalte und die dafür erforderliche Businesslogik zu konzentrieren. Das setzt allerdings voraus, dass das CMS mehr als nur reine Schreib und Leseprozesse unterstützt.

## Flexibel und erweiterbar

Durch die Jamstack-Logik lassen sich beliebige Business-Anforderungen mit Hilfe von JavaScript React, Web-APIs und Microservices flexibel umsetzen. Das CMS muss solche individuellen Erweiterungen und Veränderungen allerdings erlauben, z. B. indem spezifische Aufgaben durch selbst entwickelte oder vorgefertigte Komponenten abgebildet werden.

## Vorgefertigte Komponenten

Headless/Decoupled CMS wie Scrivito erlauben es Anwendern, spezielle Funktionen durch vorgefertigte Komponenten (Widgets) umzusetzen. Diese Widgets reichen von Building-Block-Elementen für Überschriften, Listen, Bilder usw. bis hin zu Funktionsbausteinen für User Interfaces. Solche gepflegten und ausgetesteten Komponenten verfügen bereits über einen hohen Reifegrad, was eigene Entwicklungszeit einspart.

## Pre-Rendering von Bildern

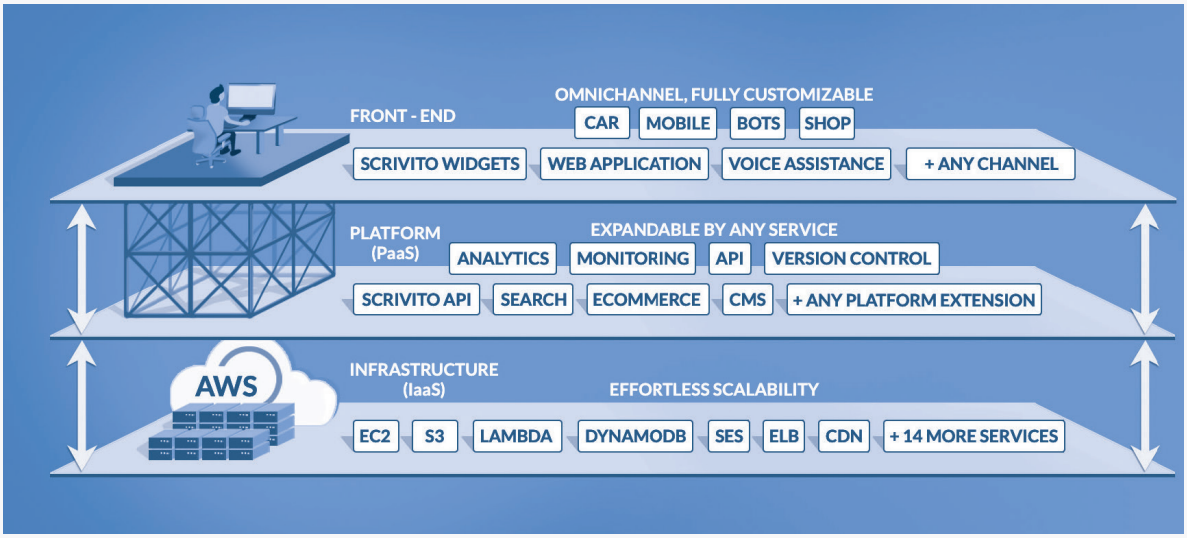
Mit der steigenden Vielfalt an Endgeräten steigt auch der Bedarf nach einer angepassten Content-Darstellung. Durch die Verknüpfung von Headless/ Decoupled CMS mit einem Content Delivery Network (CDN) können z. B. standardmäßig mehrere Auflösungen eines Bildes vorgefertigt und über das CDN für beliebige Endgeräte bereitgestellt werden.

## Content-Verwaltungslogik

Durch immer komplexere Business-Funktionen entsteht schnell eine unübersichtliche Vielfalt an Inhaltselementen. Veraltete und nicht gepflegte Elemente wie Preislisten können allerdings zu erheblichen Problemen führen. Deshalb sind CMS-Funktionen zur Versionierung mit Regeln zur Freigabe, Pflege und Aktualisierung erforderlich.

## 100% Cloud-Lösung

Die Cloud-Infrastruktur ermöglicht eine elastische Skalierbarkeit für Lastspitzen und saisonale Lastschwankungen, wie z. B. das Weihnachtsgeschäft im Handel. Native CMS-Cloud-Lösungen machen zudem ein „CMS as a Service“ mit einer Abrechnung nur für die tatsächliche Nutzung (Pay-per-Use-Modell) möglich.




Kopflose / entkoppelte CMS-Systeme wie Scrivito sind von Natur aus für die Cloud konzipiert. Sie nutzen verfügbare Ressourcen und Webdienste wie Big Data, Technologien der nächsten Generation (z.B. Sprachsuche, maschinelles Lernen, KI) und Cloud-Dienste (z.B. Lambda-Funktionen, Amazon S3, Amazon EC2, DynamoDB, CloudFront oder das Amazon API Gateway).

# FAZIT

## JAMSTACK IST SCHNELLER, SICHERER UND KOSTENGÜNSTIGER

Der Jamstack-Ansatz hat einige entscheidende Vorteile gegenüber der herkömmlichen Server-Infrastruktur. Er ermöglicht eine höhere Performance, verbessert die Sicherheit, senkt die Kosten, ist leichter skalierbar und liefert ein besseres Nutzererlebnis.

### Vereinfachte Architektur

Die vielschichtigen Anforderungen von Webservern, Load Balancern, lokalen Caching-Mechanismen, Kapazitätsplanung und verschiedenen Redundanzschichten werden durch eine drastisch vereinfachte Jamstack-Architektur abgelöst. Die Ausführungslogik wird vom Server in den Browser verlagert. Performance und Verfügbarkeit sind dadurch von der Server-Infrastruktur entkoppelt.

### Bessere Performance

Der Benutzer muss nicht mehr warten, bis der Server mit seiner Arbeit an immer komplexeren Web-Applikationen fertig ist. Jetzt können vorgefertigte HTML-Seiten über ein CDN sehr schnell bereitgestellt werden. Bei interaktiven Seiten und zusätzlichen Benutzeranforderungen liefern einzelne Microservices weitere Ergebnisse, ohne dass die Seite komplett neu geladen werden muss. Die bessere Performance ist nicht zuletzt ein SEO-Kriterium für das Ranking bei Suchmaschinen.

### Höhere Verfügbarkeit

Die einfache Architektur und die Auslieferung statischer HTML-Dateien in Verbindung mit JavaScript führen zu einer sehr hohen Stabilität und Sicherheit. Schließlich sorgt ein globales CDN zu jeder Tageszeit für eine zuverlässige Verfügbarkeit. Das CDN ist ein weiterer Vorteil gegenüber einer eigenen Server-Infrastruktur als Single Point of Failure (SPoF).


## Größere Sicherheit

Anstelle verwundbarer Server und unsicherer Plug-ins reduziert die drastisch vereinfachte Jamstack-Architektur die Angriffsflächen. Deutlich weniger Code (low code) vereinfacht die Qualitätsüberwachung und erhöht die Stabilität. Einzelne Microservices im Webbrowser laufen unabhängig voneinander und gefährden nicht mehr das Gesamtsystem.

## Geringere Betriebskosten

Die neue Struktur führt zu erheblichen Vorteilen bei Betrieb und Wartung, da keine Server mehr vorgehalten und gepflegt werden müssen. Software-Updates und Sicherheitspatches entfallen. Ein hoher Vorfertigungsgrad von Funktions-Widgets verringert außerdem den Entwicklungsaufwand. Schließlich lassen sich Cloud-basierte CMS ausschließlich auf der Basis tatsächlicher Nutzung abrechnen.



## Businesstauglicher

JavaScript, APIs und Microservices ermöglichen neue Businessangebote. Die Web-Applikationen werden dynamisch und die Services interaktiv. Selbst erstellte oder bereits vorgefertigte Microservices liefern benötigte Businessfunktionen wie Produkt- und Tarifvergleiche oder Möglichkeiten für Self-Services.

## Besseres Nutzererlebnis

Durch die Jamstack-Performance erhalten Besucher auch bei Webseiten ein App-ähnliches Nutzererlebnis (User Experience). Die Geschwindigkeit und Interaktivität als SEO-relevante Eigenschaften sind sowohl beim Erstaufruf als auch bei der Navigation zwischen den Seiten auf einem höheren Niveau. Das bessere Nutzererlebnis ist ein messbarer Umsatzfaktor, da die Absprungrate kleiner wird und die größere Interaktivität die Verweildauer erhöht.

# JAMSTACK SETZT TRENDS UND ENTWICKELT SICH WEITER

Der Jamstack-Ansatz mit einer JavaScript-Bibliothek wie React, Web-API und vorgerendertem HTML setzt sich vor allem bei Web-Anwendungen weiter durch. So wird das ursprünglich für Facebook entwickelte React beispielsweise bei Netflix, Airbnb oder Instagram eingesetzt. Mit React Native ist inzwischen eine Variante für die Entwicklung mobiler Android- und iOS-Apps verfügbar. Ein weiterer zukunftsrelevanter Bereich ist der Einsatz von React 360 für 3D-, Virtual- und Augmented Reality-Anwendungen. Schließlich profitieren Headless / Decoupled CMS von den Jamstack-Möglichkeiten zur Distribution auf viele neue Devices und bei der Umsetzung von dynamischen und interaktiven Web-Anwendungen.



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

D-21-236905, Version 0