

















Building Single Page Applications With **dotCMS**



	Key Takeaways	04
	What is a Single Page Application?	06
<ul style="list-style-type: none">• Progressive Web Apps• The Rise of the SPA		08 08
	Single Page Applications vs. Multi-page Apps	09
	The Benefits of Using SPAs	10
	The Limitations of SPAs	11
	When to Use (& NOT to Use) a Single Page Application	13
	How dotCMS Helps Developers & Marketers Build SPAs	14
<ul style="list-style-type: none">• Developers: Framework Freedom• Marketers: Content Authoring as it Ought to be		15 15
	How dotCMS Delivers Personalized Content via SPAs	17
	How DYNAMIT Uses dotCMS to Build SPAs (Without Javascript)	18
	Frequently Asked Question About SPAs	20
	dotCMS: The Ideal SPA Solution	22
	About dotCMS	23
	About DYNAMIT	25
	References	26



Digital Transformation Never Stops

Technological breakthroughs are always happening, leading to the emergence of new touchpoints, devices, and digital experiences that slowly sculpt consumer expectations. For the sake of growth and relevance, **ambitious brands are always evolving their customer experiences** in line with those expectations, molding every touchpoint in a way that —hopefully— differentiates their brand in the market.

Single Page Applications (SPAs),

are one of the new breakthroughs and touchpoints being used to help brands deliver fast, personalized customer experiences on the web.

Users on LinkedIn, Facebook, and Gmail are already using (and loving) single page applications; they just aren't necessarily aware of it. Part of the beauty of an SPA is that it streamlines the user experience in such a way that **the user barely notices they're fetching and browsing data without leaving the page** — data that would otherwise be presented on multiple pages, with each one having to load individually.

As we will come to discuss, SPAs aren't the best fit in every situation, but when it comes to developing lightning-fast user experiences within a single page, building Progressive Web Apps (PWAs), or creating offline experiences, using a single page application is a no brainer.

In this whitepaper, we'll:

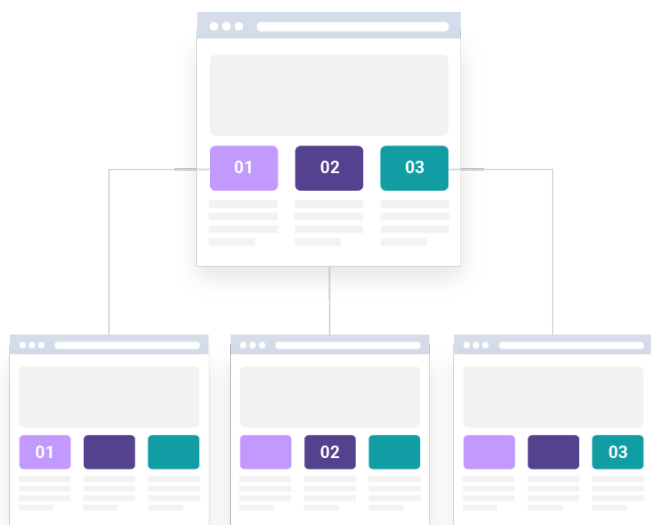
- Define single page applications
- Explore the benefits and the limitations
- And show exactly how dotCMS enables brands to develop SPAs without having to type a single line of Java



Key Takeaways

SPAs

No page refresh required



MPAs

Whole page refresh required



01. What is a Single Page Application (SPA)?

A single page application, or SPA, is a **type of web application that operates entirely from one page**, sometimes with an “infinite scroll” user interface. Further, SPAs don’t require the entire page to reload when the end user clicks or scrolls on a page element to fetch new data or to execute an action.

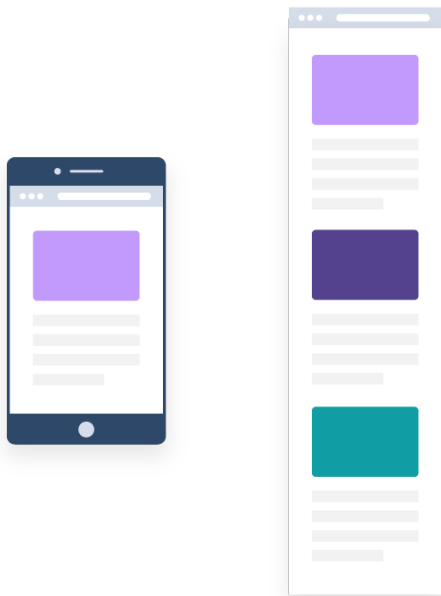
02. Are SPAs Better than Multi-page Applications (MPAs)?

It depends on the use case. SPAs have many benefits, including **speed and the ability to deliver offline experiences**. However, MPAs are more secure, and are easier to optimize for search engines.



PWAs

An SPA built to provide a native mobile app experience



04. How Does dotCMS Help Build and Manage SPAs?

With its API-first and decoupled architecture, **dotCMS can headlessly distribute content wherever it needs to go, including to an SPA.** Meanwhile, marketer-friendly content authoring tools help facilitate ongoing content creation for SPAs. That's because features such as content previews, inline editing, and personalization rules are not disrupted when marketers are working on SPAs, enabling marketers to work without relying on the IT team.

03. What are Progressive Web Apps (PWAs)?

A Progressive Web App is (almost always) a type of SPA. **It's a browser-based application that's built to mimic a native mobile application,** and can be used like one via a web browser. The mobile version of Twitter's website is a good example of a PWA in action.

05. Can dotCMS deliver Personalized Content to SPAs?

Yes, unlike other content management systems, which are limited to page-based personalization rules, **dotCMS can adapt to SPA environments by providing server-side personalization.**



What is a Single Page Application?

A single page application, or SPA, is a type of web application that

operates entirely from one page, usually with an “infinite scroll” user interface.

Further, SPAs don't require the entire page to reload when the end user clicks or scrolls on a page element to fetch new data or to execute an action. They're **built with Javascript** and can be developed on a multitude of frameworks including Angular, Vue, and React.

Just like any other website, SPAs are accessed through a web browser, but the significant difference is that SPAs possess the ability to deliver a more dynamic user experience, and at a faster speed.

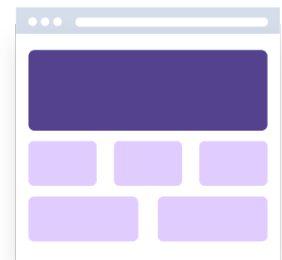
Traditional

New requests for information reload the whole page.



SPA

Only requests the new pieces of information.





A popular SPA example is the Facebook feed that we all know and use. When you click the 'Like' button, or when you request to see the comments on a Facebook post, **the entire page isn't reloaded**. Instead, the page loads in the relevant data without a full page refresh being required.

Another example can be found in Gmail. You'll notice that **Gmail doesn't load entire pages from a server** when you compose a new email, **it dynamically updates the current page instead**. The web browser remains on the same page, but new content has been served to the end user.

Both the Facebook Feed & Gmail are examples of Single Page Apps

Additionally, SPAs—in the form of Progressive Web Apps — can be used to build offline experiences, as users can load content that is cached even when their internet connection is lost. With an PWA, data can be cached, even two-three steps ahead of the customer's current location on their journey. That way, even if their internet connection is lost, or if the brand's servers do down momentarily, the customer experience can continue.

Progressive Web Apps

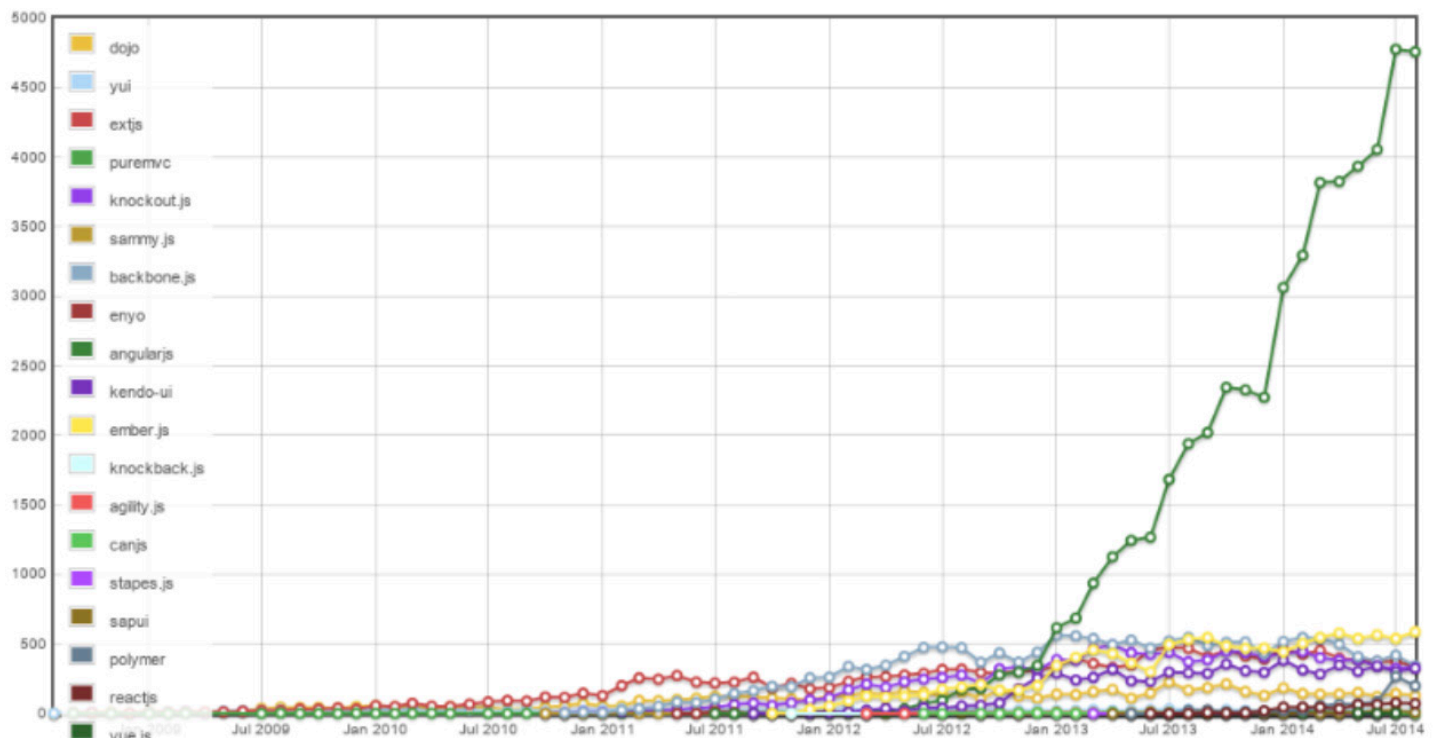
A Progressive Web App is (almost always) a type of SPA. **It's a browser-based application that's built to mimic a native mobile application**, and can be used like one via a web browser. The mobile version of Twitter's website is a good example of a PWA in action, while Google Docs is one example of an PWA that takes advantage of browser APIs and browser-side caching to work offline when necessary.

When it comes to the actual delivery of content, **SPAs are much faster than traditional web applications** (i.e. multi-page applications, which we will come to discuss later in this whitepaper) since they can execute page navigations and logic within the browser itself instead of bouncing back to the server to retrieve the page information. This

is achieved by leveraging AJAX, a tool that enables the SPA to exchange information with the backend servers, which then loads this information into the app without having to incur a full page refresh.

The Rise of the SPA

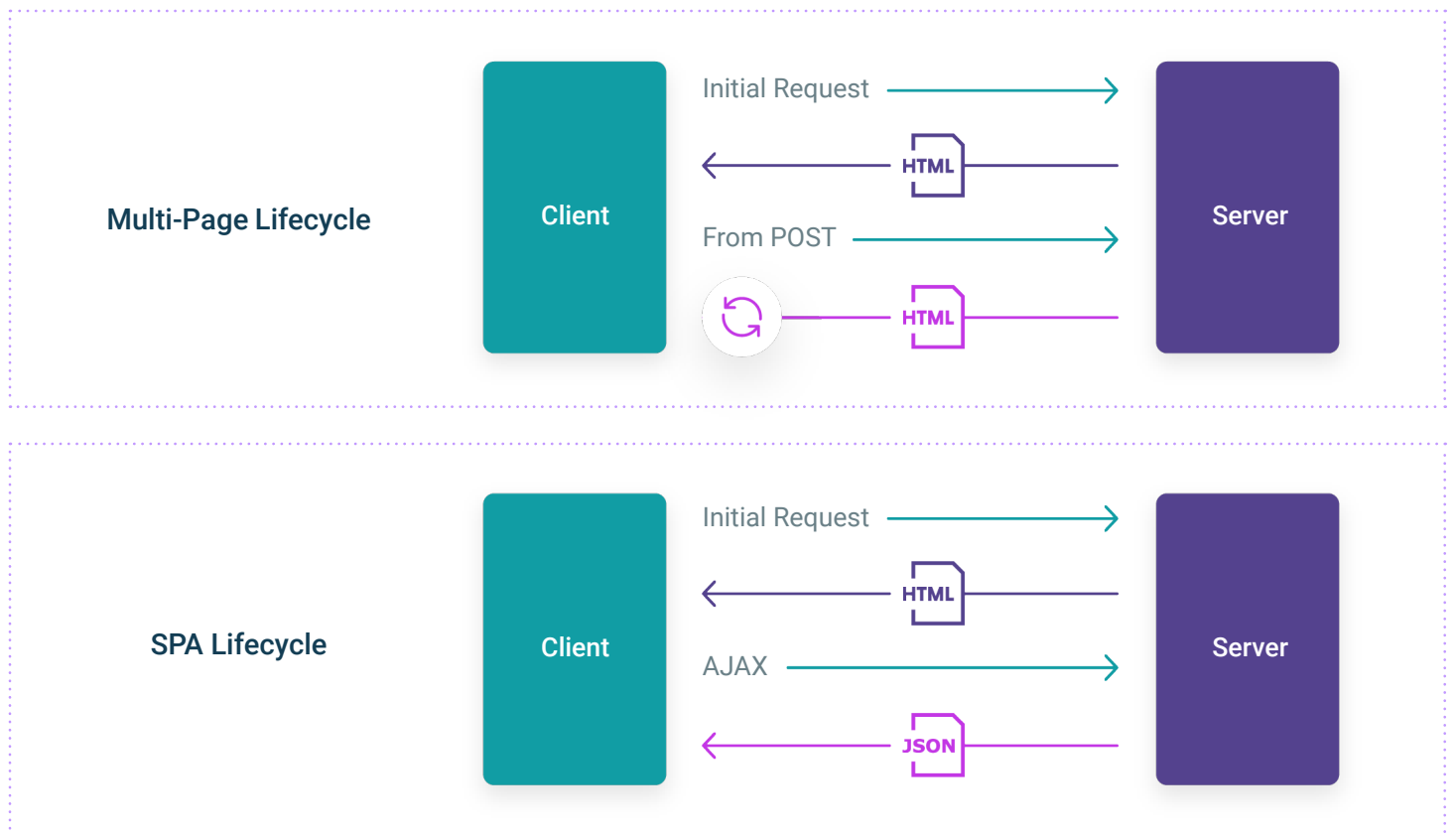
There is growing evidence that shows the growth of SPAs across industries. While the world largest digital organizations continue to roll out SPAs (here's looking at you, Twitter, Netflix and Google) **the graph below shows a sharp rise in developers becoming proficient in Angular JS**, a framework that has plays a fundamental role in developing single page applications.¹





Single Page Applications vs. Multi-page Apps

To understand the difference between SPAs and multi-page applications, here's a useful graphic taken from an article written by Mike Wasson on the Microsoft Developer Network.²



As the above graphic illustrates, **multi-page applications** provide a more traditional way of delivering web experiences. Each time the end-user wants to navigate to a different page, **a request is sent to the server which in turn renders a new page to update both appearance and content.**

At one time, before the advent of SPAs, these multi-page applications were much larger, therefore requiring much longer load times. It also enlisted the use of complex user interfaces in order to function properly.



The Benefits of Using SPAs

Single Page Applications have a host of benefits, which is why they're continuing to rise in popularity. Here they are at a glance.

01. Speed

SPAs provide a **faster-perceived load time**. Most of the resources (ex: HTML, CSS, Scripts) are loaded once throughout the lifespan of the application.

02. Relatively Easy to Build

Because SPAs are **housed in one HTML document**, developers can build themes and template faster.

03. Adaptability

SPAs are mobile ready. In a SPA dev environment, **you can use the same backend components**, including the content from your web-based application, for your mobile application.

04. Streamlined Development

To develop a SPA is **simple and straightforward**. No coding is required to render pages from the server. As a matter of fact, you can kick-start your development without utilizing any server at all.

05. SPAs Can Operate Offline

SPAs are able to **cache any local storage with dispatch**. SPAs send one server request and then saves all the data that it receives. The SPA then retains this data so it can be used over and over again.

06. Debug SPAs with Chrome

Since you can see all the code in one place, **you can inspect individual page elements**, which will have their own unique ID, and apply the appropriate fixes.



The Limitations of SPAs

An end-user might wonder why every website on the web isn't an SPA. After all, the benefits are fantastic. However, SPAs don't fit every scenario, and here's why:

01. SEO Optimization

You are **limited** to how many keywords you can put on one page. However, this is something that is being continuously addressed, so eventually, it will be resolved.

02. Initial Slow Loading Speeds

Depending on the size of the SPA, the **initial data request can be slow to download**.

03. JavaScript Reliance

SPAs are developed using JavaScript. If users have disabled JavaScript, they won't be able to use the system. Also, only developers who have **sufficient knowledge of JavaScript** can only develop SPAs.

04. Security

In comparison to multi-page applications, **SPAs are less secure**. Attackers could, theoretically, inject malicious coding (client-side) into the application, affecting the end-user.

05. Difficult to Edit

With a regular headless CMS, the content authoring experience (if there was one in the first place), is **disrupted** in the case of an SPA. However, dotCMS enables marketers to use traditional content authoring experience including WYSIWYG editing, online editing, and content previews.

Learn more about...

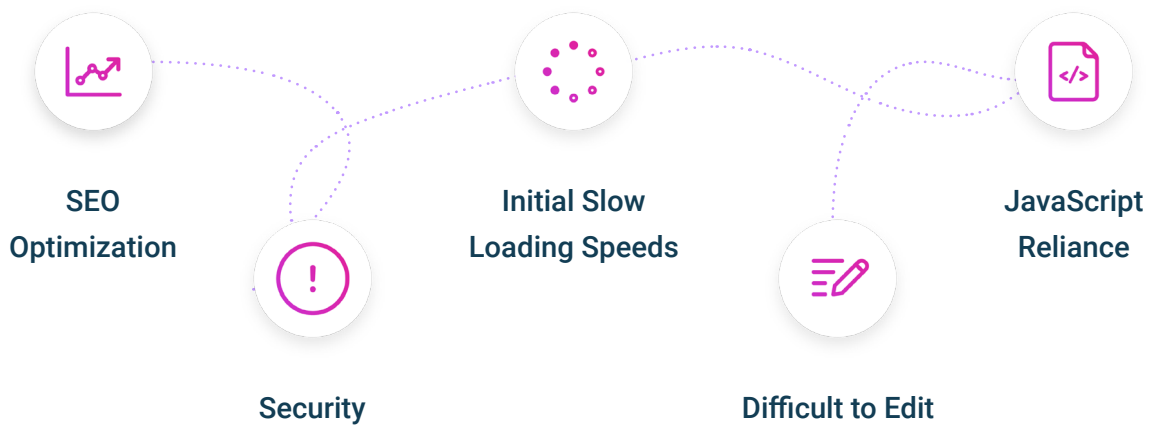
How dotCMS gives marketers and developers the best of both worlds through its [Hybrid CMS solution](#).



Benefits of Using SPAs



Limitations of SPAs





When to Use (& NOT Use) a Single Page Application

Speaking at dotCMS Bootcamp 2018⁴, John Hartley, associate team director at DYNAMIT explained that SPAs shouldn't be used for every application development scenario.

“SPA isn't always going to be the right answer. If you have a developer that [says] every site that we do should be SPA, ask them why. [If they say], it is the latest and coolest thing to do, then that's not really a good business case.”

Hartley further explained that if your application is going to see a lot of client-side interaction, state management and a considerable amount of data that needs to persist, then **SPA is the best approach since it provides faster page load speeds.**

On the other hand, if you are developing a web-based application that is going to deliver static content, for instance, a blog, then going down the SPA route is not ideal. Introducing static content will affect the load time on a SPA since users will need to download and run the JavaScript payload before viewing any content.³



How dotCMS Helps Developers & Marketers Build SPAs

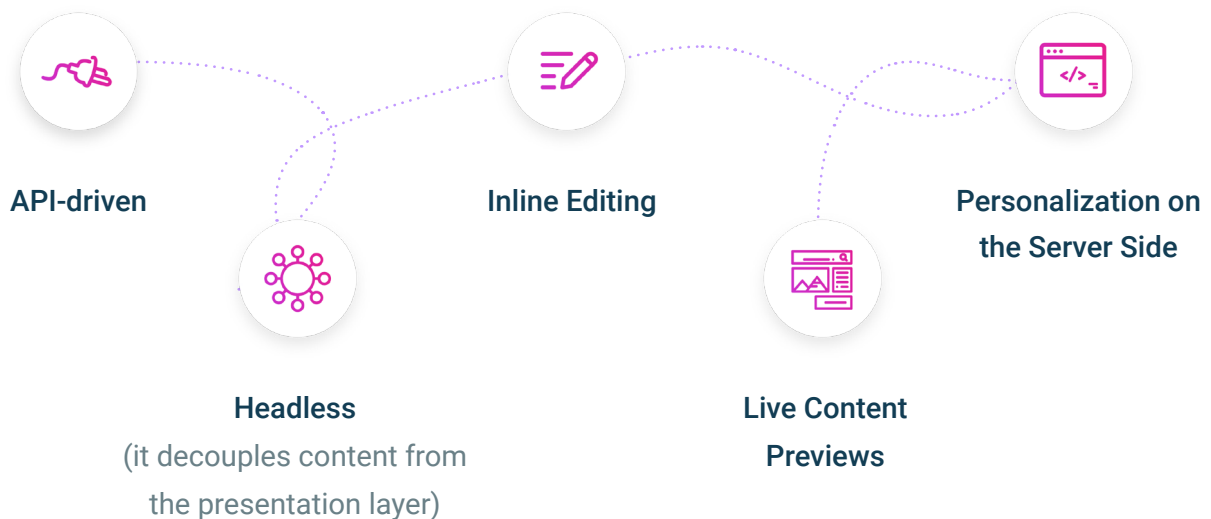
With a pure headless CMS solution, there's no front-end presentation layer. For developers, this setup is great as it allows them to have **more freedom and flexibility when it comes to content delivery and presentation**. For instance, they can go ahead and built SPAs just the way they envisage them. But for marketers and business users who aren't proficient in JavaScript, this is a major disadvantage.

Without WYSIWYG editing, drag-and-drop functionality, and content previews, marketers have a hard time building websites, landing pages, and microsites. They just click publish, and hope for the best.

In this tricky scenario, any changes to an SPA will require the marketing team to lean heavily on the IT team – creating a bottleneck. To make matters worse, most content management systems have personalization rules that aren't compatible with an SPA, because those rules are page-based.

Thankfully, dotCMS has a solution to all of the below.

dotCMS solutions



As well as providing developers with an **API-first solution that is framework agnostic**, dotCMS also serves marketers by giving them a **content authoring environment** they can thrive in.

Developers: Framework Freedom

In other words, dotCMS serves both sides of the company, by **providing developers with an API-first solution that is framework agnostic**, and by giving marketers the content authoring environment they can thrive in.

Furthermore, dotCMS provides Layout as a Service (LaaS)⁵, which takes the best of CMS-driven experiences, which includes easy templating, personalization, rule and permission-based content delivery and server-side contextual rendering workflow, and combines this with the **developer friendliness of Content as a Service**.

dotCMS automatically assigns each content type a unique ID, which enables the developer to pinpoint to

a specific content type that they want to display on the SPA. If you want a single piece of content for your SPA, you can command dotCMS, via the framework of your choice, to only extract that piece of content. As mentioned previously, dotCMS is compatible with many frameworks including Angular, Vue, React, Ember, Backbone, and Aurelia.

Thus, developers won't be tied or restricted to a tightly coupled presentation layer that is usually found in a traditional CMS and they will have the **freedom to develop any type of application with the tools of their choosing**. Developers can also build and experiment with the latest modern applications and implement them into dotCMS.

Marketers: Content Authoring As It Ought To Be

dotCMS provides **Layout as a Service (LaaS)**⁶, which takes the best of CMS-driven experiences, including **easy templating, personalization, rule and permission-based content delivery and server-side contextual rendering**, and combines this with the developer friendliness of Content as a Service—which enables headless content delivery.

The result is a marketer-friendly experience that empowers marketers with inline editing, content previews, and drag-and-drop page building.

The introduction of dotCMS 5.0 was also a boon for marketers. The major release embodied dotCMS' 'NoCode' philosophy, which seeks to give enterprise marketers a codeless (and autonomous) experience.



dotCMS 5.0 brought about the following enhancements, all of which can be leveraged in an SPA environment:



**Drag-and-Drop
Page & Layout
Editor**

Layouts are now tied to the page itself, making it easy to make changes to the layout without having to leave the page editor.



**Drag-and-Drop
Workflow
Builder**

Building on the 'Four Eyes' approval feature from dotCMS 4.3, the dotCMS 5.0 workflow builder allows for multiple workflow schemes per content type.



**Drag-and-Drop
Content Type
Builder**

Users can now drag and drop from a list of fields to build out content types. Adding rows, columns, and changing the order of fields makes it easier to build out new content types, especially forms.

This latest release continues to follow dotCMS's long-term goal of becoming the most user-friendly and easy-to-use headless CMS offering on the market.



How dotCMS Delivers Personalized Content Via SPAs

Thanks to the API-driven environment in dotCMS, you can deliver personalized content to your segmented audience through a single page application.

To do this, you'll need to collect and aggregate the targeting information within your single page app and pass that information back on subsequent queries to the REST API. By doing this, you can mimic the functionality of the queries generated by the "Pull Personalized" tooling in dotCMS—which include tags and personas—to retrieve content.⁷



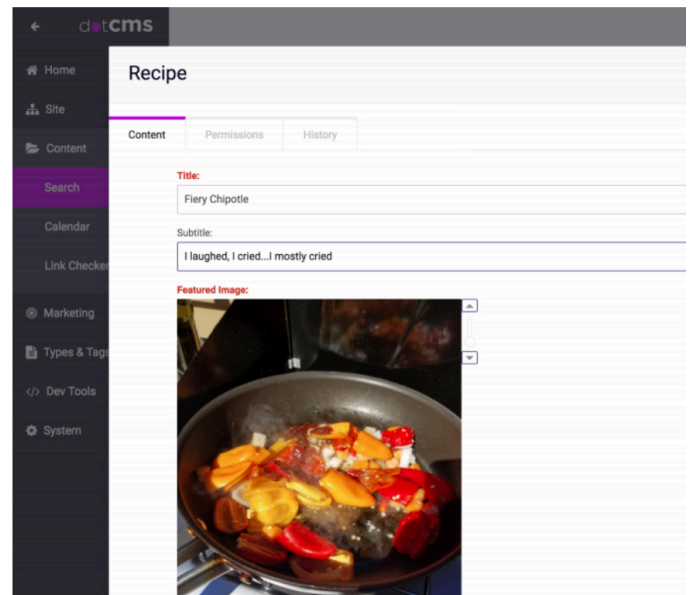


How DYNAMIT Uses dotCMS to Build SPAs (Without JavaScript)

At dotCMS BootCamp 18, associate team director John Hartley of DYNAMIT shared how his company utilizes dotCMS to build and manage SPAs. Below, you'll find a [step-by-step guide on how DYNAMIT uses dotCMS to create a single page application that sells hot sauce](#).

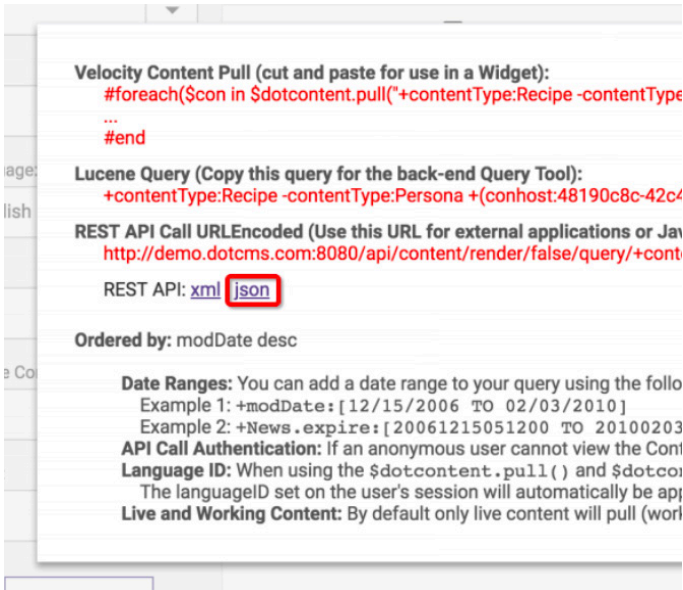
Step 01

The layout and content were created using dotCMS's drag-and-drop composition and WYSIWYG editor, respectively. Page elements like Titles, Subtitles, and a Featured Image.



Step 02

Once the presentational layer has been set up, the JSON file was extracted via the Search function that is available on the dotCMS editor. The search query pulls all the content showing all the velocity templates (dotCMS's built-in front-end development templates), the Lucene query, as well as the hot sauce recipe content, as shown below.



Step 03

Clicking the “JSON” link on the screenshot above, will show you the JSON file.

Step 04

In developing their SPA, DYNAMIT utilizes **Vue CL1, which provides a simple way to run online commands and create new SPA applications.** The Vue CL1 framework assists users in routing the content and layout listed in the JSON file by commanding dotCMS to populate specific content details through selecting the content’s unique ID.

Step 05

Once the SPA has been routed and configured, the **SPA is then connected to the dotCMS API environment.** Further customization can then be applied through CSS.

Producing a Slug URL and ID URL

With Vue CL1, DYNAMIT was able to push their dotCMS-built SPA to two output lengths, an actual ID URL and a slug URL.

ID URL:

<http://samplesite.com/recipes/3fdc86d5-76fe-48c2-9ed9-24dba3c7c78b>

Slug URL:

<http://samplesite.com/recipe/fiery-chipotle>

The difference is that the ID URL enables the Vue CL1 to pull content and utilize it in your SPA component. And the slugify URL makes the SPA more easier to index on a search engine and also makes it easier to read and share by your end-user.



Frequently Asked Question about SPAs

01. Will My File Size Be Larger if I Develop an SPA?

This is partially true. You will have many JavaScript files which will be bounded into one. To address this, you can use chunking or code splitting wherever possible to reduce the file size and reduce the number of HTTP requests.

02. Will My SEO Be Affected If Use an SPA?

This is a fairly common issue, and we've already addressed this as one of the main disadvantages of running a SPA in this whitepaper. Fortunately, there are a number of ways around this issue. You can use a server-side rendering or pre-rendering tool. Or, you can also use a router, like Vue CL1, to automatically generate a slug URL to coincide with your ID URL, as demonstrated by DYNAMIT.

03. Is There a Way to Remove Hashtags in URLs?

Back in October 2018, Google's John Mueller announced at PubCon that their search engine will not favor URLs that contains a "#" (hashtag)⁸, with the exception of basic anchors such as href="#anchor". Thankfully, there are several ways around this. For instance, you can use a router to generate a slugify URL for your SPA.

04. What About Accessibility Considerations for an SPA?

All SPAs must follow the WCAG 2.0/2.1 (Web Content Accessibility Guidelines), which defines how to make content more accessible for people who have disabilities and impairments.⁹ This includes visual, auditory, speech, physical, cognitive, language, learning, and neurological disabilities.



Here are a couple of **best practices and protocols** that will make your SPA even more accessible, and provide a better user experience:

Bypass blocks:

These are essentially blocks that sit on top of a page in a SPA, and they are not visible until you tap into it and it says “skip to content”. The primary purpose of these bypass blocks is to **skip over the main navigation to the exact page of where you want to go, in half the time.**

Learn more about...

How dotCMS [facilitates website accessibility](#).

Focus management:

This is an area where we need to pay a lot of attention, especially if you are in the initial stages of building an application. As users interact with your SPA, via a keyboard or screen reader, they will be using a lot of functions including drop-down boxes, forms, and clicking on several links on the SPA. **It is vital that you replace the user’s focus whenever the user interacts with your SPA.** That’s the biggest part to focus management: you have to make it very clear where someone is going.



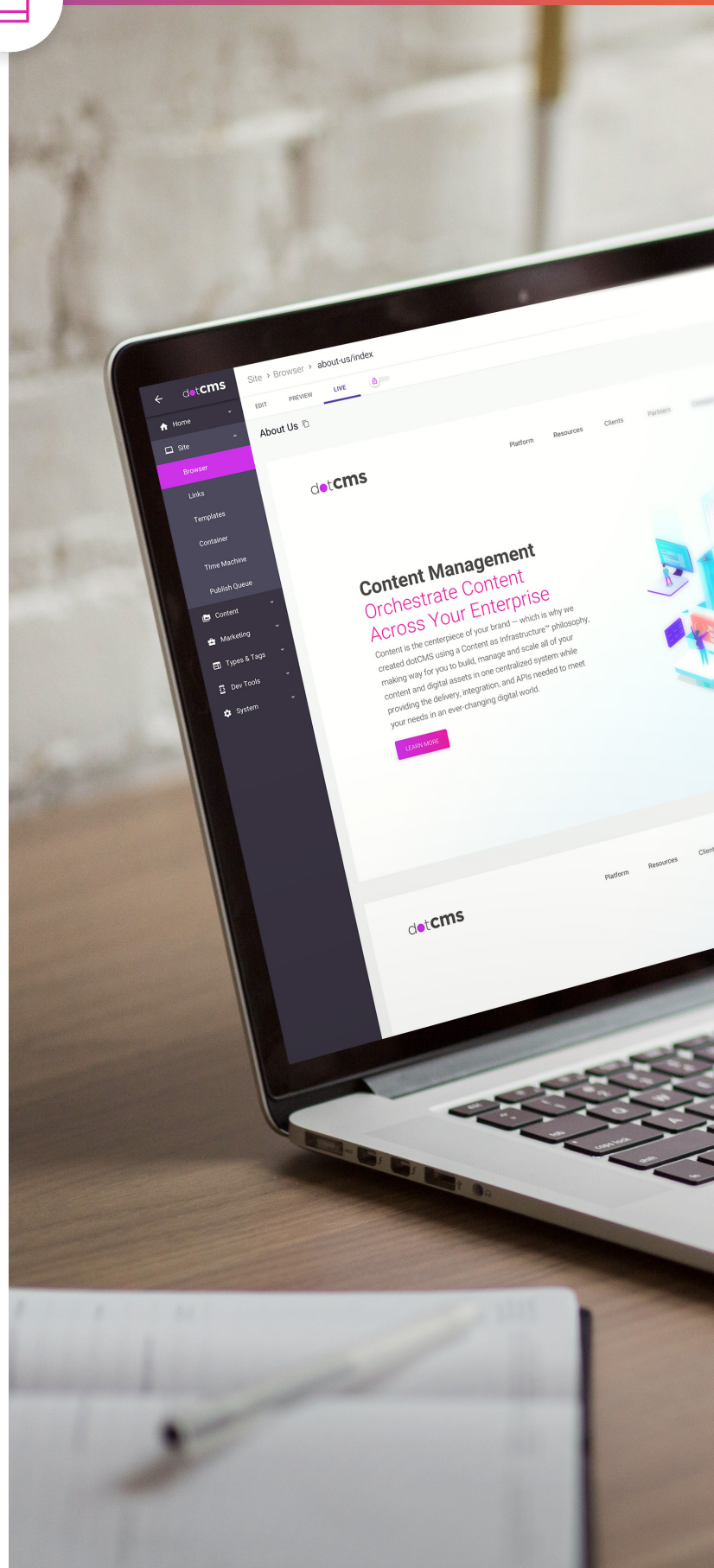
dotCMS: The Ideal SPA Solution

SPAs and Progressive Web Apps (PWA) are growing in popularity. In recent years, organizations such as AliExpress, The Washington Post, and Forbes have created their own PWAs using SPA technology.

The only problem is, not every CMS is built to serve both developers and marketers when building and managing these dynamic experiences. **dotCMS** on the other hand, **gives developers the framework freedom they need to craft excellent digital experiences, while giving marketers the content authoring tools they've become accustomed to.**

**For more information
on how dotCMS can help you
build your SPA,**

[get in touch with us today.](#)



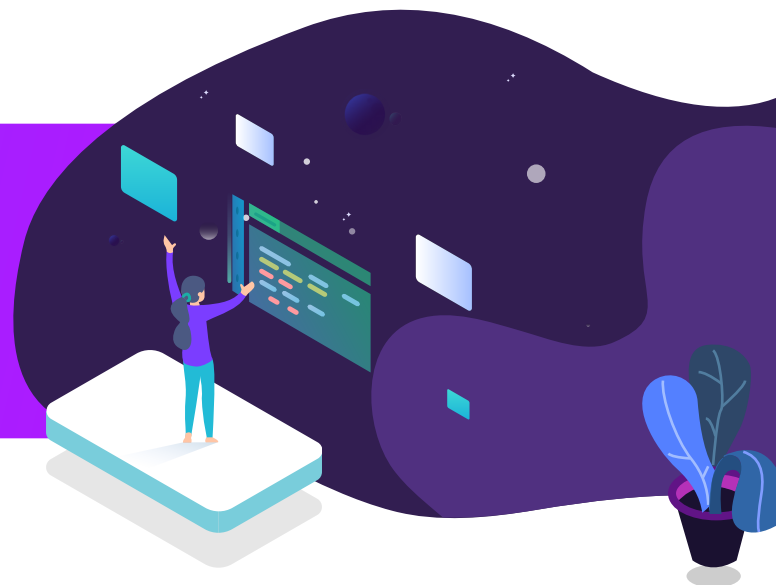


About dotCMS

Is an open-source Java, customer experience orchestration hub for companies that want to drive business outcomes with their websites and other content-driven applications. dotCMS provides the technology to deliver connected and continuous customer experiences that business teams can orchestrate. **Extensible, scalable, and with headless content management capabilities, organizations can rapidly build their Digital Experience Platform and drive innovation** while their marketing and business teams drive customer experiences for every touchpoint, in every customer journey, on any device – all from a single system.

Founded in 2003, dotCMS is a privately owned U.S. company with offices in Miami (Florida), Boston (Massachusetts), and San José (Costa Rica). With a global network of certified implementation partners and an active open-source community, **dotCMS has generated more than a half-million downloads and over 10,000 implementations and integration projects in over 70 countries.** Notable dotCMS customers include: Telus, Standard & Poors, Hospital Corporation of America, Royal Bank of Canada, DirecTV, Nomura Bank, Thomson Reuters, China Mobile, Aon, DriveTest Ontario, and ICANN.

SCHEDULE A
dotCMS demo at dotcms.com

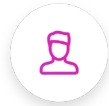




Contact dotCMS



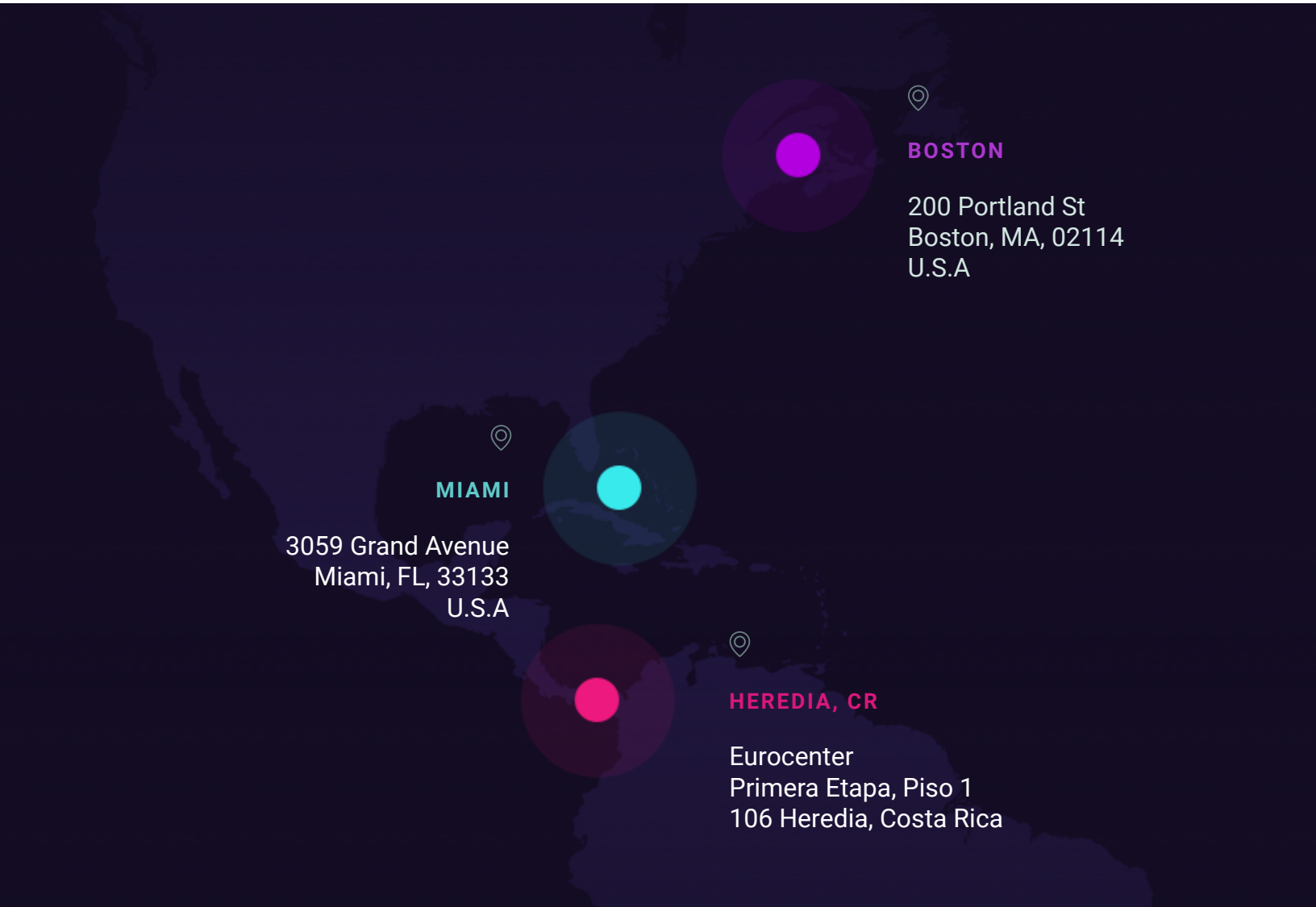
Web:
dotcms.com



Phone:
+1 - 305 - 900 - 2001



Email:
info@dotcms.com





About Dynamit


Dynamit creates web and mobile applications, serving its clients as strategists, creatives and engineers. In collaboration with its clients, Dynamit creates the strategies and leverages the tools and technologies necessary to solve complex business problems in today's digital age.


Dynamit blends the best of both marketing and IT—form and function—to help clients tackle some of their toughest challenges.

Dynamit believes that

-done right-technology should simplify, making life easier for clients and their customers.

DYNAMIT

 Web:
dynamit.com

 Phone:
+1 614 538 0095



References

- 1 "Single- Page Web Apps with Angular JS are Immensely Popular" <https://aimconsulting.com/insights/blog/single-page-web-apps-angular-js-immensely-popular-right-business/>
- 2 "Single-Page Applications - MSDN - Microsoft." <https://msdn.microsoft.com/en-us/magazine/dn463786.aspx>
- 3 "What is a single-page application? - Exoscale." <https://www.exoscale.com/syslog/what-is-a-single-page-application/>
- 4 "What Is a Single Page Application? - CMSWire." <https://www.cmswire.com/digital-experience/what-is-a-single-page-application/>
- 5 "Beyond Headless Content: Layout as a Service in dotCMS.", <https://dotcms.com/blog/post/beyond-headless-content-layout-as-a-service-in-dotcms>.
- 6 "Beyond Headless Content: Layout as a Service in dotCMS.", <https://dotcms.com/blog/post/beyond-headless-content-layout-as-a-service-in-dotcms>.
- 7 dotCMS Documentation: Pull Personalized Content, 2018 <https://doc.dotcms.com/docs/latest/pull-personalized-content#CustomPersonalizedPull>
- 8 "Google Says Do Not Use Hashtags In URLs." <https://www.seroundtable.com/google-no-hashtags-in-urls-26537.html>
- 9 "Web Content Accessibility Guidelines" <https://www.w3.org/TR/WCAG20/>