The SEO problem with Single Page Applications and how HTML pushstate provides a solution.

The rise in mobile device applications and mobile web useage in general has greatly influenced internet user expectations as to how they want their content delivered. When you're navigating around in your favorite iPhone or Android app, the displays usually transition smoothly. You rarely get a feeling that a given page is being "loaded" on your device. As a result, an increasing amount of web content is trying to follow suit.SEPSEPA popular way for a website to emulate a device application is to load content by way of asyncronous AJAX calls, i.e., javascript injecting new content from a server into a page without the entire page needing to be reloaded in the browser.

A classic example of this is posting a comment on Facebook, where your comment pops up after you submit it, but without the page being reloaded from scratch. Over the past few years this trend has been taken even further, with a notable number of websites now running as Single Page Applications, or SPAs. In a SPA, there is only the initial page load when you arrive at the website. From there, all the different pages within the site are loaded by javascript. You can check out some award winning examples of SPAs **here**.Looking at a few of these SPA sites confirms how much many of these sites strive to emulate the smooth user experience of a good Android or IOS application as different web pages seem to glide into view without the laborious process of a page loading.

So why aren't more of these single page application sites

being built? As a developer, I have been in the room when beautiful, single page application demo sites have been vetoed by project and account managers. The reason why? SEO.

SEO for Single Page Apps

Google's indexing crawlers have traditionally ignored content that is served by javascript. Google considers a single web page as a unique block of content with semantically valid html that corresponds to a unique URL. This makes the page worthy of indexing and, subsequently, ranking. The classic issue with SPAs is that the content on the page changes without the web url changing accordingly.For example, let's say you had an initial website homepage "A" load in your browser. Then once homepage "A" is loaded, you click on another page link that shows you a new webpage, page "B". Only this time, page B

loads entirely by javascript. Google crawlers would only recognize homepage page "A" as a valid page, as it was the result of a traditional, server-side page load. To google, the javascript loaded page "B" doesn't exist. But since website visibility is such a critical component to the success of a given web property, it makes no sense to build a beautiful SPA site if no one can find it in search results.

This problem of Google not indexing javascript pages is a persistent one for SPA sites and even for traditional sites that rely heavily on javascript. Attempts at solutions initially have proved awkward at best.

How Twitter Attempted to Provide a Solution to SPA SEO Twitter, which has relied heavily on AJAX technology, came up with the idea of javascript

loaded urls to have a '#!' url segment to make the url unique to correspond to a given tweeted comment. However, this became known as the "hash bang" tag and it was far from a perfect solution; using '#!' in a url was not in accordance with **World Wide Web Consortium (W3C) standards**, as it didn't truly mark the path to a real resource location. The '#!' also made things difficult for developers since they had to work with these unusual characters in backend developement scenarios. Consequently, the hash bang url was a flop despite being initially recommended by Google, oddly enough, as a quick fix to the SEO, javascript page dilemma.

Another more solid solution was to make sure that if a website was being built as a single page application, a complete duplicate site would be created within the same url to make sure that Google crawlers - working in the background - would index the site pages properly.

Again, this was not a viable long term solution. Creating mirror versions of a given live website gave rise to a maintenance nightmare, as code redundancy is the enemy of the web developer.**Enter HTML Push State**

Push State is a feature of HTML 5's History API. In modern browsers, the main window object now has a child history object that enables a user to programmatically move backwards for forwards in the browser history via window.history.back() or window.history.forward() giving the same behavior as if the user were clicking on a given entry in their browser's history toolbar menu. But particularly relevant for SEO and SPA applications is one of the history API's methods; pushState.

The pushState method takes three arguments, a data

"state" object, a title string, and a url. When the pushState method is called, the url that is passed to it will appear in the browser's url window. So if you're on http://mydomain.com, and call the below pushState method, the browser window will then show http://mydomain.com/new-page.html.

 Up until recently, the only way that javascript could programmatically change a browser url was to change the window.location object; but this would always result in the reloading of a browser window. But with pushState, the url magically updates without the page reloading. A PushState fired url change can then be followed by an ajax call that loads new content onto the page, so we have a new url and new content to match. You can see a fun example of pushState **used here,** courtesy of **CSS Tricks.**

Note that the state object you pass into the pushState function can contain useful data that you can use on the new page. There is a limitation to the amount of data you can use in a pushState state object however, as the serialized version must be under 640k. There is also a wise limitation built into pushState which only allows new urls that are within the current web domain.

HTML Push State Boosts SEO for Single Page Apps

This new feature HTML 5 offers gives a huge boost to SPAs when it comes to being search engine optimized. However, it doesn't exactly qualify as a quick fix. Going back to our previous example, a user sees the new url mydomain.com/new-page.html magically appear via pushState in the browser window, accompanied by new, ajax loaded page content.That's all well and good, but what happens if the user then hits enter and refreshes the

page with that new url?In this case, the server, not javascript, takes over. If the web server can't find an actual, document titled new-page.html, then there will be ugly "404 not found" thrown. Moreover, this 404 would prove that the Uniform Resource Locator known as "http://mydomain.com/new-page.html" does not in fact exist. This means that your work in setting up an SEO compliant Single Page Application site is not yet done. In order for content on mydomain.com/new-page.html to come up in search results, Google and other search engines will insist that your pushState loaded url is not just a javascript construct, but actually points to a valid browser resource.

So all urls that you intend to magically load via pushState will also need to exist as real web pages. Once this happens, Google will then properly acknowlege and index your Single Page Application website as it would

any other site. While this involves more grunt work on the part of the developer, the reward offers the best of all worlds: a SPA site that offers a fast and fluid user experience, SEO compliance, and safe, server side fallback urls that operate regardless of browser versions.Single Page Applications presently comprise only a tiny portion of the web landscape. HTML 5's History API and **associated methods** may change this by allowing SPA's to compete for search engine rankings on an equal playing field with conventionally built websites.

The SEO problem with Single Page Applications and how HTML pushstate provides a solution.

The rise in mobile device applications and mobile web useage in general has greatly influenced internet user

expectations as to how they want their content delivered. When you're navigating around in your favorite iPhone or Android app, the displays usually transition smoothly. You rarely get a feeling that a given page is being "loaded" on your device. As a result, an increasing amount of web content is trying to follow suit.A popular way for a website to emulate a device application is to load content by way of asyncronous AJAX calls, i.e., javascript injecting new content from a server into a page without the entire page needing to be reloaded in the browser.

A classic example of this is posting a comment on Facebook, where your comment pops up after you submit it, but without the page being reloaded from scratch. Over the past few years this trend has been taken even further, with a notable number of websites

now running as Single Page Applications, or SPAs. In a SPA, there is only the initial page load when you arrive at the website. From there, all the different pages within the site are loaded by javascript. You can check out some award winning examples of SPAs **here**.

Looking at a few of these SPA sites confirms how much many of these sites strive to emulate the smooth user experience of a good Android or IOS application as different web pages seem to glide into view without the laborious process of a page loading.

So why aren't more of these single page application sites being built? As a developer, I have been in the room when beautiful, single page application demo sites have been vetoed by project and account managers. The reason why? SEO.

SEO for Single Page Apps

Google's indexing crawlers have traditionally ignored content that is served by javascript. Google considers a single web page as a unique block of content with semantically valid html that corresponds to a unique URL. This makes the page worthy of indexing and, subsequently, ranking. The classic issue with SPAs is that the content on the page changes without the web url changing accordingly.For example, let's say you had an initial website homepage "A" load in your browser. Then once homepage "A" is loaded, you click on another page link that shows you a new webpage, page "B". Only this time, page B loads entirely by javascript. Google crawlers would only recognize homepage page "A" as a valid page, as it was the result of a traditional, server-side page load. To google, the javascript loaded page "B" doesn't exist. But since website visibility is such a critical component to the success of a given web property, it makes no sense to

build a beautiful SPA site if no one can find it in search results.

This problem of Google not indexing javascript pages is a persistent one for SPA sites and even for traditional sites that rely heavily on javascript. Attempts at solutions initially have proved awkward at best.

How Twitter Attempted to Provide a Solution to SPA SEO Twitter, which has relied heavily on AJAX technology, came up with the idea of javascript loaded urls to have a '#!' url segment to make the url unique to correspond to a given tweeted comment. However, this became known as the "hash bang" tag and it was far from a perfect solution; using '#!' in a url was not in accordance with **World Wide Web Consortium (W3C) standards**, as it didn't truly mark the path to a

real resource location. The '#!' also made things difficult for developers since they had to work with these unusual characters in backend developement scenarios. Consequently, the hash bang url was a flop despite being initially recommended by Google, oddly enough, as a quick fix to the SEO, javascript page dilemma.

Another more solid solution was to make sure that if a website was being built as a single page application, a complete duplicate site would be created within the same url to make sure that Google crawlers - working in the background - would index the site pages properly. Again, this was not a viable long term solution. Creating mirror versions of a given live website gave rise to a maintenance nightmare, as code redundancy is the enemy of the web developer.**Enter HTML Push State**

Push State is a feature of HTML 5's History API. In modern browsers, the main window object now has a child history object that enables a user to programmatically move backwards for forwards in the browser history via window.history.back() or window.history.forward() giving the same behavior as if the user were clicking on a given entry in their browser's history toolbar menu. But particularly relevant for SEO and SPA applications is one of the history API's methods; pushState.

The pushState method takes three arguments, a data "state" object, a title string, and a url. When the pushState method is called, the url that is passed to it will appear in the browser's url window.So if you're on http://mydomain.com, and call the below pushState method, the browser window will then show http://mydomain.com/new-page.html.

Up until recently, the only way that javascript could programmatically change a browser url was to change the window.location object; but this would always result in the reloading of a browser window. But with pushState, the url magically updates without the page reloading. A PushState fired url change can then be followed by an ajax call that loads new content onto the page, so we have a new url and new content to match. You can see a fun example of pushState **used here,** courtesy of **CSS Tricks.**

Note that the state object you pass into the pushState function can contain useful data that you can use on the new page. There is a limitation to the amount of data you can use in a pushState state object however, as the serialized version must be under 640k. There is also a wise limitation built into pushState which only allows

new urls that are within the current web domain.

HTML Push State Boosts SEO for Single Page Apps

This new feature HTML 5 offers gives a huge boost to SPAs when it comes to being search engine optimized. However, it doesn't exactly qualify as a quick fix. Going back to our previous example, a user sees the new url mydomain.com/new-page.html magically appear via pushState in the browser window, accompanied by new, ajax loaded page content.That's all well and good, but what happens if the user then hits enter and refreshes the page with that new url?In this case, the server, not javascript, takes over. If the web server can't find an actual, document titled new-page.html, then there will be ugly "404 not found" thrown. Moreover, this 404 would prove that the Uniform Resource Locator known as "http://mydomain.com/new-page.html" does not in fact

exist. This means that your work in setting up an SEO compliant Single Page Application site is not yet done. In order for content on mydomain.com/new-page.html to come up in search results, Google and other search engines will insist that your pushState loaded url is not just a javascript construct, but actually points to a valid browser resource.

So all urls that you intend to magically load via pushState will also need to exist as real web pages. Once this happens, Google will then properly acknowlege and index your Single Page Application website as it would any other site. While this involves more grunt work on the part of the developer, the reward offers the best of all worlds: a SPA site that offers a fast and fluid user experience, SEO compliance, and safe, server side fallback urls that operate regardless of browser versions.SEPSEPSingle Page Applications presently

comprise only a tiny portion of the web landscape. HTML 5's History API and **associated methods** may change this by allowing SPA's to compete for search engine rankings on an equal playing field with conventionally built websites.