Search Off The Record - Ep98

[00:00:09:13] - John
Hello, and welcome to a new episode of Search Off the Record, a podcast coming to you from the Google Search team where we talk all about Search, and, I don't know, maybe have some fun along the way. My name is John, and I'm a Search Advocate here at Google in Switzerland. And today we have Martin joining us. Hi, Martin.

[00:00:31:02] - Martin
Hi, John.

[00:00:32:04] - John
Nice to have you here, Martin, because I have some questions around making websites and SEO.

[00:00:39:22] - Martin
Okay. Okay. What is your question?

[00:00:43:00] - John
I heard there was this thing called lazy loading.

[00:00:46:05] - Martin
Oh, yeah.

[00:00:47:21] - John
I would like to know more about whether or not it has SEO effects, if it's something you have to watch out for, kind of like some best practices of what to do and all of these kind of things. Does that sound like an interesting topic?

[00:01:03:27] - Martin
That sounds like a very interesting topic, and I think we haven't been speaking about this for a while now, so it's probably a good refresher for those who have heard us speak about it before, and for those who have never heard about it, it's probably a good primer. Yeah.

[00:01:16:16] - John
Woo-hoo! Okay, so what is lazy loading? Is it something like lazy developers do?

[00:01:24:22] - Martin
No. The idea with lazy loading is to only load resources when you need them, rather than load everything at the same time and then making it like a little bit slower for everything that you're loading and also using all sorts of resources, like battery power and network, traffic, these kind of things.

[00:01:41:18] - John
Okay. But I thought some of that happens automatically in the browser. Like, if you open a page and you look at kind of the Network tab, you have that waterfall diagram and it's like, "Oh, it does five and then it does five more." These kind of things. Is there more involved?

[00:01:57:25] - Martin
Well, so that's the chunking that HTTP used to be doing in HTTP/1, but there's more than just that. So, even if you look at this kind of chunking, eventually it will start to load, let's say, images or videos that are at the very bottom of the page, even if you never, ever get close to the bottom of the page, it will be loaded unless there is lazy loading implemented in some way.

[00:02:25:14] - John
Okay, so it's basically something. If you have a long page or a really big page, then it makes sense to kind of separate things out.

[00:02:35:14] - Martin
Yeah, I would actually say pretty much all pages can benefit from this, especially longer pages. That is true. But, even if it's not a super long page, it can probably gain a bit of performance and a bit of, let's say, leniency from the browser by using lazy loading.

[00:02:55:15] - John

Cool. So you mentioned performance, I suspect that's a big aspect of this as well, and also, especially on things like mobile browsers, you don't want to download everything if you don't need it. Are those kind of the primary drivers behind lazy loading?

[00:03:13:07] - Martin
Yes, those are the primary drivers. As I said, you want to avoid work that yields nothing. And also, if you have lots of non-critical resources, as in images that the article would be fine without if necessary, then loading them just doesn't make that much sense because it keeps the browser busy with things that it won't need in the end, maybe, unless the user navigates to the part of the website where this resource is necessary.

[00:03:39:28] - John
Okay, kind of taking a step back, is this something that has gotten more popular over the years? It feels like, for a while, lazy loading was something lots of people talked about. Is it basically just working nowadays or how do you see that?

[00:03:58:16] - Martin
I think it has become a little more prevalent when we build more complex websites with lots of content, and you can save on a lot of resources. Notice that I say resources, even though it is mostly used for images and iframes and maybe videos. It can be used for all sorts of things. Let's say you have a stock quote ticker thing and you actually have to pay for the API calls that you make. You might be able to save some of them. If people are not scrolling to this ticker, then you don't have to load it so you can implement this yourself. There used to be a time where you had to implement this yourself. As websites grew more complicated, performance took a bigger hit from all these resources being employed in a website. People looked for ways to improve performance for users, as well, and then they kind of had to come up with their own ways. They had to be creative to find ways to detect if a resource is possibly visible to a user or not and deciding when to load them and deciding how to load them. That has been a phase or an era where lazy loading libraries have proliferated. There's like so many of them still to this day, I don't think very many of them are still maintained. A few of them probably still are maintained because they do very specific custom things. But a couple of years ago, and I can't remember how long ago that was, but a couple of years back, browsers got a native attribute for images and iframes, the loading attribute. There, you can specify lazy, which makes the browser take care of the lazy loading for you, rather than having to use some sort of JavaScript API to do that. I think that became more widespread, so more people are using it. I know that Felix Arntz, from a sister team here at Google, he contributed to WordPress, so WordPress uses image lazy loading now, I think, by default. As that is spreading, the discussion gets a little quieter because it's kind of like, "Oh yeah," they just put on the loading attribute, set it to lazy. You're done. It's not as tricky anymore.

[00:06:09:17] - John
Oh okay. So less JavaScript, more HTML seems like a good move.

[00:06:16:27] - Martin
I think that's a fantastic move, actually.

[00:06:19:00] - John
Okay. Why doesn't the browser just do this by default for all images? Like, how do you see it with regards to these images? Why wouldn't you just set it for everything?

[00:06:32:16] - Martin
Ah, that's a fun one. Actually, I got an email from Dave Smart about this a couple months ago--a couple of years ago, I can't remember--but the content management system that we are using for onesie for developers.google.com/search is doing exactly that. It defaults all images to lazy loading, which is not great. You could ask, "But why? I mean, that's great, no?" No because, if every image gets lazy loaded, that means that images that are immediately visible or should be immediately visible are also being lazy loaded. There is a caveat to lazy loading, where browsers have something called a resource scanner. They look for images, for instance, specifically, and they might understand like, "Oh, images are very visual, so they are probably very noticeable when they are missing," so they try to start loading images as early as possible. As they are going through the HTML, they see an image they would "would" immediately start loading it so that it is available for the user to see. If you have a hero image, which is the very top of the page--so you will definitely, or most definitely see it, as you come to the page--they would do that unless there is a loading=lazy because loading=lazy tells the browser, "Oh, no, no, only load it when it's necessary." It kind of parses the entire page, gets everything ready, loads the non-lazy loaded resources, and then looks at the lazy loadable things and only then would recognize, "Oh, the header image, the hero image needs to be downloaded now," downloads it, and then eventually it pops into the page. That also means it probably moves things around if people are not specifying width and height of the image, and that would actually be a pretty jarring experience. So not great.

[00:08:29:02] - John
Oh, okay. With regards to SEO, is the performance side the primary aspect where you would see that?

[00:08:38:18] - Martin
Yeah.

[00:08:39:22] - John
I imagine that would go into Core Web Vitals?

[00:08:42:26] - Martin
Correct. So, if you're not using lazy loading where you should, that will probably impact some aspect of the Core Web Vitals. Most likely Largest Contentful Paint. Well, that actually is also most impacted most likely. Largest Contentful Paint, that is actually most likely also impacted if you use it for images where you shouldn't use it because that means that the painting happens later than it could happen or should happen. It does reflect in the Core Web Vitals. Performance is the primary concern that lazy loading addresses.

[00:09:17:05] - John
And, for Largest Contentful Paint, is that always an image? Do you know? How does that work?

[00:09:24:04] - Martin
It doesn't have to be an image. It likely is an image. But, if you have most of your article copy, for instance, coming in client-side rendered from an API that is slow and you're not doing this efficiently, then there's likely a large block of content popping in pretty late, and that would also impact the Largest Contentful Paint. So it doesn't always have to be images. But, if you are using lazy loading on an image that is immediately visible, that is most likely going to have an impact on your Largest Contentful Paint. It's like almost guaranteed.

[00:09:58:23] - John
Cool. Okay. Does it affect indexing in any way, or ranking? How does that kind of play in?

[00:10:09:09] - Martin
It does influence things a little bit. So, if you are using the native lazy loading, it doesn't really have that much of an impact because, after all, you have an image with a source attribute where the image is linked and then it's just loaded a little later. It has some rendering implications, and it has some Core Web Vitals implications, which goes into ranking, but is a tiny, minute factor in most cases. There are exceptions, but most cases it doesn't matter that much. Indexing, on the other hand, especially if you're using a custom technology to do lazy loading, like a library or some JavaScript that you wrote yourself, then very likely this can impact indexing if it's not done correctly, in the sense of we've seen multiple lazy loading libraries, for instance, that use some sort of like data.source attribute rather than the source attribute. And then, if there are problems with the library, it might end up actually not loading the image at all. Loading, in this case, means putting the actual source image into the source attribute. If it's not in the source attribute, we won't pick it up if it's in some custom attribute.

[00:11:25:08] - John
Okay.

[00:11:26:10] - Martin
So, yeah, that can have indexing implications.

[00:11:30:05] - John
Do you know why people would use custom libraries for lazy loading? Is that common, or how does that work?

[00:11:40:00] - Martin
I think it's still somewhat common because, number one, if you have a working setup, why would you change it? Like, people might not have updated their website for the last couple of years, or maybe have updated it but haven't actually touched the technology that powers the website. That's especially true for themes that you might have installed five years ago that did not trust, or did not use, or did not know about the lazy loading attribute that is in HTML. Number one. Number two is you can do things that the native lazy loading can't do. For instance, you might want to load a low resolution preview either from a low resolution preview that has been saved somewhere at a different URL, or from a data URL that you put into the source and then swap it out for the high resolution version as the user goes there. That simply isn't possible with just HTML these days. You would have to use some JavaScript to make that happen. The other thing is, what if you are not using images or iframes? Those are the only two elements that lazy loading is implemented in the browser for. If you want to lazy load videos, or if you want a lazy load, I don't

know, article content that comes from an API, or if you have your stock ticker, or if you have your comments, these kind of things--if you want to lazy load those, you very likely have to rely on JavaScript to do that as well.

[00:13:03:18] - John
It's unlikely that all of these custom libraries will go away because there's a lot, I guess, you can do with the native lazy loading, but there's still things where people might decide, "Well, actually, I want to go further."

[00:13:22:13] - Martin
Yeah. I mean, the custom experience might be driving you to a library. There's a very simple stock implementation. It's a bit like using JavaScript to do fancy things where content transitions rather than reloading the page, right? Links do the trick, they load the page anew and then show the necessary content, but you might want to use JavaScript to do fancier things. I know that the View Transitions API is coming, which will make things a little easier there, but it's still not baked in default behavior, so you would have to do something to get custom behavior. To be fair, I'm not sure, but I'm pretty hopeful that most of the libraries will actually do this properly and will not break things for Search if you use them. If you use a custom library, that doesn't mean that you have a problem. It just means there is potential for a problem.

[00:14:17:25] - John
Cool. How would you recognize if there's a problem? Like, if you look at your site and your developer say, "Oh, we implemented lazy loading, everything is better now," and you're an SEO, how would you check to make sure that it's actually implemented in a reasonable way or in a way that doesn't affect SEO?

[00:14:39:04] - Martin
That's a lovely question. And the easiest way probably is to go to Search Console, use the URL Inspection Tool and look at the rendered HTML. If the rendered HTML--ignore the screenshot--if the rendered HTML looks like it contains all the image URLs in the source attribute of an image tag, if it has all the content that you lazily loaded, if it's like a custom widget or if it's custom content that you load, like if you use, I don't know, a thing to load comments under an article, then if the comments are there, you will be fine. If the comments are missing, then it might be that the lazy loading is implemented in a way that does not work properly in Googlebot. And then you might want to have a look at why that is and how you can fix that.

[00:15:25:00] - John
From a practical point of view, would you just copy and paste the rendered HTML into a doc and then search for the image tags and go through it like that?

[00:15:35:12] - Martin
Yeah, I think that's the easiest way. You can use the little search icon in the Search Console itself, but I guess it's easiest to just copy and paste it out.

[00:15:44:15] - John
I guess, if you're an SEO and your developers just implemented lazy loading, it's good to make sure that you have some knowledge of HTML so that you kind of understand what is happening on the page for this. Is there a way to kind of recognize when there's an issue without like knowing all of the details of HTML?

[00:16:06:23] - Martin
Yes, so if you know that you have specific content that you want to rank for and that is lazy loaded, then if you're not showing up for the relevant queries, it might be that the content is just not there. So then you can spot check that. The other alternative is, if it's images specifically, you can check if the images are indexed. If the images are not being picked up at scale, then that might mean that your lazy loaded images are not properly implementing lazy loading. With images, it's less likely because you probably just use the browser version and that should be just fine. But, if you are not using the default lazy loading for images, then that could be the reason why your images are not showing up in the index.

[00:16:52:02] - John
Cool. I think that's super helpful. Another topic that's kind of, I guess, related is infinite scroll. Is that a kind of lazy loading or how would you separate that out?

[00:17:05:21] - Martin
I mean, it kind of is. There is a philosophical difference, I would call it. The philosophical difference being lazy loading is loading non-critical resources of a page, let's say images or additional content, like comments and these kind of things, so it's the same page content. It's just parts of the same page content. Whereas, infinite scroll loads additional content and kind of like makes the page infinite in terms of content, like all the content lives on this page now. It has

its own challenges and potential pitfalls, but it is somewhat related, and it's, again, an approach or technique to progressively load content rather than load it all in one go. So it is somewhat related, but it has slightly different aims or goals, I would say.

[00:18:00:05] - John
I guess, if you're implementing it yourself or with a JavaScript library, it would be similar in that you look at the viewport and see, "Oh, should I be loading something here?" and then it goes off and does it.

[00:18:13:23] - Martin
Yeah. It can create a little frustrating situation where you scroll for five minutes, then something happens, you go to another page, you go back, and then you've lost your spot. So you want a way to express in the URL where in this kind of like infinite stream of content you are. But, yeah, there are libraries that help with that, I think.

[00:18:37:23] - John
Cool.

[00:18:38:25] - Martin
Excellent. Yeah. So does that answer your lazy loading questions?

[00:18:43:08] - John
I think. Pretty much. Yeah. I think, fundamentally, it sounds like, for the most part, sites don't need to worry about this nowadays because they're native solutions. If a developer does something custom or uses a special library, it's easy to check in Search Console to see if it's actually working. And, since it primarily affects images, I think you said, you can also just focus on the images and see how they're kind of embedded in--well, not embedded--findable in image search. You mentioned videos as well. How would that work there?

[00:19:31:03] - Martin
Yeah. Videos are quite resource-intensive. It oftentimes makes sense to load a poster image, and then only load the video as it becomes available in the viewport. I believe there are ways in HTML to do that with the poster attribute, but I know that a bunch of people are having some sort of custom implementation to do this kind of thing so you're only starting to load the video as it becomes available. But then, for very large videos, you likely use things like streaming anyway. It's a lot more complicated than with images, but it is definitely possible to do this as well, to not load the video immediately. Unless, of course, it's like a header or a hero video, and you want to autoplay that as the user comes onto the page, then lazy loading doesn't really make sense there. The other thing is some people do it for privacy reasons, that they only load external video content when the user consents to it. That's kind of lazy loading, I guess, because you're not loading a non-critical resource for the page immediately, so you could call that lazy loading, I think. I don't know. Would you call that lazy loading?

[00:20:39:23] - John
So you basically show the thumbnail and have a text like, "Click here to activate the video."

[00:20:46:13] - Martin
Yes. Or like a Play button or something, but not actually load the video immediately.

[00:20:51:14] - John
Yeah. Okay. The other thing I was thinking of, as we were going through this, is what about small images? It sounds like this is mostly something for big images, but if you have like decorative images that you use in a corner of your boxes or like those kind of things.

[00:21:09:08] - Martin
Oh, that's an interesting one. That's not super trivial because there is a fuzzy border in between what is and isn't a decorative image. I mean, for instance, hypothetically, hypothetically, you could use data URLs for most of your images, and that would be a terrible idea because you're just making the HTML really large and hard to parse. But I guess decorative images, like custom bullet points or little icons here and there, you would most likely not put them into the HTML as an IMG as an image tag. You most likely would use CSS to kind of display them, but make clear that this has no semantic meaning. So, if my article is about, I don't know, the fish of the Great Barrier Reef, then the images of the fish of the Great Barrier Reef are not decorative. They are actually explanatory and part of the main content, so semantically they are part of the article; whereas, a little fish icon somewhere is probably not semantically part of the article content. It is more likely just decorative, and then I would load it through CSS. Wwe have mentioned that CSS images are not necessarily picked up for indexing, right?

[00:22:21:19] - John

Yeah. Okay. But, if you load it with CSS, is that also lazy loading or is that basically loaded immediately?

[00:22:30:28] - Martin
I wouldn't call it lazy loading because it's not telling the browser or it's not doing it so that you're loading it as you need it. It is going to be loaded pretty much instantly when it's found in the CSS, I believe, but it is decorative, so it's outside of the semantic image situation, so I don't think it's lazy loading if you put it in the CSS. I mean.

[00:22:54:19] - John
I mean, since these are small, it's not like it's going to have a big effect anyway.

[00:22:59:03] - Martin
That's true. Yeah. If you load like a three megabyte decorative image, then ask yourself, why are you doing that?

[00:23:05:19] - John
Every morning. Yeah.

[00:23:07:00] - Martin
Every morning, into the mirror, "Why am I loading a multi-megabyte decorative image?"

[00:23:14:06] - John
Oh, my god. Okay.

[00:23:15:13] - Martin
Oh, boy. Okay.

[00:23:16:26] - John
Well, this was super interesting. Thanks for joining, Martin.

[00:23:20:10] - Martin
Thanks for having me, John.

[00:23:21:12] - John
Well, that's it for this episode. If people want to chat more about lazy loading images, where would you send them?

[00:23:28:05] - Martin
Probably into the forums, so our Search Central Help Community, or onto LinkedIn. Yeah, I think that the forum is probably the best place to discuss lazy loading with us.

[00:23:40:28] - John
Would going to web.dev also be a place?

[00:23:46:01] - Martin
Oh, yeah, I think web.dev has lots of documentation on lazy loading. We also have some documentation on our site as well, but web.dev is probably the best place.

[00:23:55:00] - John
Thanks a lot, Martin, and thank you, folks, for listening in, and goodbye.

[00:24:00:07] - Martin
Bye-bye.

[00:24:04:23] - John
We've been having fun with these podcast episodes. I hope you, the listener, have found them both entertaining and insightful too. Feel free to drop us a note on LinkedIn, or chat with us at one of the next events that we go to if you have any thoughts. And, of course, don't forget to like and subscribe! Thank you, and goodbye.