http://ctrlclickcast.com/episodes/htaccess-primer

CTRL+CLICK CAST #18 Jonathan Penn on .htaccess

[Music]

Lea Alcantara: You are listening to CTRL+CLICK CAST. We inspect the web for you! Today we're talking about .htaccess with Jonathan Penn. I'm your host, Lea Alcantara, and I'm joined by my fab co-host:

Emily Lewis: Emily Lewis.

Lea Alcantara: This episode is brought to you by Hover. Hover takes away all the hassle of registering domains, no crazy upsells and no complicated interfaces. They also provide a crazy amount of top level domains to choose from including the new .link domain extension. Emily and I are in the middle of a company rebrand right now, and some of their domain suggestions from our keyword search actually helped us narrow down some choices. So register your next domain with Hover using the promo code HTACCESS to get 10% off your first purchase.

Emily Lewis: CTRL+CLICK would also like to thank Pixel & Tonic for being our major sponsor. [Music ends] Hey Lea, anything new in your world?

Lea Alcantara: Not much except the weather is getting better.

Emily Lewis: [Laughs]

Lea Alcantara: And I feel like I'm planning barbecues and stuff that's outdoor related these days.

Emily Lewis: Yeah, I think the people who live in Seattle, in that part of the country, they get really excited at the end of spring and summer.

Lea Alcantara: How true.

Emily Lewis: [Laughs] Because you actually get good weather for a number of months, right?

Lea Alcantara: Yes. It's a little bit weird though as a kind of a new person in Seattle how locals react, because I don't think it's even that much warmer.

Emily Lewis: [Laughs]

Lea Alcantara: But I feel like the moment the sun comes out, everyone takes up their beach clothes.

Emily Lewis: [Laughs]

Lea Alcantara: And I'm like I'm still in my jacket and like pants and I've got a scarf on, like I don't understand. [Laughs]

Emily Lewis: Everyone has got shorts and flip flops on. [Laughs]

Lea Alcantara: Yeah, exactly. It's just like, "Oh, it's just because the sun is out." [Laughs]

Emily Lewis: Yeah, I think I'm one of those people. It's why I live in New Mexico, it is the sun, and I'm lucky that it's pretty much sunny here all the time. The main drawback, spring and summer are my least favorite seasons because spring is basically like wind season.

Lea Alcantara: [Agrees]

Emily Lewis: It's like 50-mile an hour winds, not gusts, just the wind is 50 miles an hour, and then we have 70-mile an hour gusts, and it's the desert so lots of dusts and so it shoves the dusts and pollen way up into your sinus cavity really well. [Laughs]

Lea Alcantara: Oh lovely, lovely.

Emily Lewis: And then it goes straight into like incredibly intense heat. So right now I'm just waiting for fall. [Laughs]

Lea Alcantara: Oh well, I'm looking forward to summer. [Laughs]

Emily Lewis: Yeah, well, hopefully I'll be up to visit you I think in July or August. I'll get to enjoy it by then.

Lea Alcantara: Oh, that would be so fun.

Emily Lewis: Yeah. So anyways, I'm really excited for today's episode because I think we've wanted to discuss .htaccess for like two years, right?

Lea Alcantara: Yeah. I know, totally.

Emily Lewis: We've been trying to find the right person to talk about it, and so our guide for today's topic is software engineer, designer and author Jonathan Penn. Jonathan has a unique gift for making highly technical topics not only understandable, but interesting, which is why I begged him to tackle .htaccess for us today. So welcome to this show, Jonathan. Thanks for joining us.

Jonathan Penn: Thanks. I'm glad to be here.

Lea Alcantara: So Jonathan, can you tell our listeners a bit more about yourself?

Jonathan Penn: Yes, I live in Akron, Ohio where our weather is not too good at the moment either that way. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Jonathan Penn: We're having some cold problems over here, and yeah, I keep myself busy trying to chase my two small children. I'm half of the executive team managing them. It doesn't work all the time, they never do what I say.

Emily Lewis: [Laughs]

Jonathan Penn: And yeah, I'm a musician, other fun stuff, and I type a lot. [Laughs]

Emily Lewis: So you're a musician like an instrument or a singer.

Jonathan Penn: Yeah, both. I do guitar and piano and sing, and I've written my own stuff, and I play out here and there, helping people out.

Emily Lewis: We have a lot of guests who are in the tech field that are musicians.

Lea Alcantara: Yeah.

Jonathan Penn: Yeah, I've noticed, they overlap. [Laughs]

Lea Alcantara: [Laughs]

Jonathan Penn: With the technical people and yeah, it's fun stuff.

Emily Lewis: It's got to be like some part of the brain is stimulated by both and it appeals to people who are musically and technically inclined. I don't fall along those lines though, but Lea, you're a singer.

Lea Alcantara: Yes, and I mean, I know it. Like I used to, well, way back when I was a teenager, I used to play the clarinet, and I was part of like honor bands and all of that crazy, nerdy stuff. [Laughs]

Jonathan Penn: [Laughs]

Lea Alcantara: And I just feel like music is very technical. Reading music, basically you're kind of subconsciously or consciously counting in your head, like the entire time while like you're performing and so you have to be very precise.

Emily Lewis: [Agrees]

Jonathan Penn: And hyper critical of music you hear on the radio anybody around you.

Lea Alcantara: Yes.

Jonathan Penn: And all that stuff. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: Yes, it's like we're bred to be working on the web.

Jonathan Penn: Yeah. [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: Yeah.

Emily Lewis: So one of the reasons why I wanted to talk about .htaccess on the show is that it was one of those files that I didn't even know existed until I started working with ExpressionEngine, and so it's one of those things that I think maybe people who fall into the kind of job description like I do, which is primarily design and front end development that when you start working with a CMS, then .htaccess becomes something that you might use to like rewrite your URLs. Like in ExpressionEngine, we use .htaccess to remove the index.php that's automatically generated in the URLs, and so it was one of those things, and to this day, that's kind of the extent of my knowledge of it. It doesn't go much further beyond that.

Jonathan Penn: [Laughs]

Emily Lewis: And so can you describe kind of what .htaccess is? What purpose it serves, especially beyond what I was mentioning with rewrites?

Jonathan Penn: Yeah. Oh, it's a big topic. [Laughs]

Lea Alcantara: Oh yeah. [Laughs]

Emily Lewis: [Laughs]

Jonathan Penn: At its core, .htaccess is a special configuration file for the Apache web servers specifically, and most people using content management systems like ExpressionEngine, WordPress or things like that, most people will experience the Apache web server. But there are other servers out there, like I'm not aware of any others that use the .htaccess file to configure things, but like if you're running on a Windows box, there are different ways of doing that. So this is just a file with commands to tell Apache what to do, and in this case, these commands help Apache know when a URL is requested, what should you do with it. Usually, Apache just says, "Oh, I'm going to go and look and see if there's a file on the disk that kind of matches the directory structure of the URL that you passed in." But with the .htaccess file, you can do things like turning on the rewrite engine and say, "Well, no, every part of this URL I want you to take this and pass it to a script instead."

Like in ExpressionEngine's case, the index.php file, and then ExpressionEngine can access the URL because it has a complete access to Apache's environment, and then it can make decisions based on that, routing to different pages that are stored in the database, maybe you have media files somewhere. I'm not sure if ExpressionEngine does this because I haven't done much work with it myself, but there are other content systems that let you even set up access controls that say, "Well, this set of URLs below this level, only people in this group can access it and stuff like that."

Lea Alcantara: [Agrees]

Jonathan Penn: So that's kind of the basics of the file itself.

Emily Lewis: And it's one of those things that at least in my experience with rewrites that I also had to learn a little bit of regular expressions.

Jonathan Penn: Yeah. [Laughs]

Emily Lewis: And when I say learn a little bit, I didn't learn anything, I just copied and pasted from what other people had done successfully. So when you do rely on it, let's say for just rewrites with .htaccess, is regular expressions like the only way that you can prompt a rewrite, or is there something else that you could use?

Jonathan Penn: You know, there might be another way. I tend to reach for the regular expressions because I know them. [Laughs]

Lea Alcantara: [Agrees]

Jonathan Penn: And so I think there are simpler forms of saying everything under this directory or if you have a pass name, you can just say everything here should be over here, but regular expressions usually are what's used because you have more complex needs for maybe rearranging parts of the URL.

Lea Alcantara: [Agrees]

Jonathan Penn: Let's say like you want to swap. If you're used to nesting in a certain way with different dates and now you want to remove the day. However you want to do that, regular expressions are really useful for that.

Lea Alcantara: Yeah, I think the first time I started using some regular expressions or having to look up regular expressions is when you're working with the CMS, oftentimes you're moving from an older system to a newer system, and sometimes the older system has a crazy URL like query, question mark, equals or whatever, you know?

Jonathan Penn: Yeah.

Lea Alcantara: And it's like random numbers and then the actual slug that is readable and make sense.

Jonathan Penn: Yeah.

Lea Alcantara: And so with the regular expression, you have to pretty much say, "Okay, remove the query completely plus whatever these other things, and now move it from the farthest slug to the first slug, and then add the word blog in front of it.

Jonathan Penn: [Laughs]

Lea Alcantara: Those kind of things, and it's funny because you can talk it out, but you kind of have to figure out how do you translate what I just said into regular expressions, and I feel like for those that are like new to .htaccess or regular expressions, it's just trying to figure out what the common language is in the first place to say, "Okay, move this from here to here with that." How does that even translate to regex.

Jonathan Penn: Yeah. It's a challenge, and I do want to make it clear too that if you have trouble and struggling to understand regular expressions, you're not alone. There's a joke among other computer scientists saying that, "Hey, if you think you have a problem and you reach for regular expressions to solve it, now you have two problems." [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: Right. [Laughs]

Jonathan Penn: Because it's just complicated. It's a complicated specification to manage.

Emily Lewis: Is it its own language, or is it just like another language that's just written a certain way?

Jonathan Penn: It is. It's kind of. You can call it a symbolic language to describe patterns that should be matched.

Emily Lewis: Okay.

Timestamp: **00:09:42**

Jonathan Penn: That's literally what a regular expression is as used by these tools like .htaccess, there's an engine under the hood that interprets the expression and then runs it. It creates this thing called the state machine. It runs the text you're trying to match to it through the state machine, and then you get an answer, did it match or did it not, but then on top of that, you can use as you've probably discovered, you can wrap certain parts of the pattern that you're matching in parenthesis and those turn into the special capture groups that then, using the dollar sign ($) and the numbers, you can say, "Well, the first capture group goes here, and the second capture group goes here." And then you can basically build anew. It's like a template of what your text used to be. I captured this. Everything after the query string that looked like a slug, that was the capture group 1, so I'm going to put capture group 1 here in the output.

Emily Lewis: Now, when it comes to regular expressions kind of creating two problems,[laughs], what were your own resources for learning it when you first started picking it up?

Jonathan Penn: Unfortunately, brute force. [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Jonathan Penn: Regular expressions have their start in shell scripting languages like Perl. They've become popular in many language, Ruby and JavaScript now have them. You can use regular expressions in the browser through JavaScript. They grew up in these lower level text processing languages, but they're so useful to express patterns in a very tight format that they just cropped up everywhere. Even though they introduce a lot more complexity and sometimes if you find yourself getting too clever or if you feel really clever after you've written a regular expression, you might want to rethink it because in six months when you come back and look at it, it's just hard.

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Jonathan Penn: Our brains are not regular expression parsers. [Laughs]

Lea Alcantara: [Agrees]

Emily Lewis: Right.

Jonathan Penn: It's really hard to figure that out.

Emily Lewis: Are there any resources that are available now that might be useful for someone wanting to get a little more comfortable with regular expressions?

Jonathan Penn: Yeah, and when we were talking about this podcast earlier, I went and did some checking to try to find a good resource for people who don't have necessarily like a low level programming background so that your use of regular expressions wouldn't get too deep, but it would help with things specifically like this with rewriting, and I found a video by Lea Verou, and I think I'm pronouncing her last name incorrectly, which she presented at the 2012 O'Reilly FluentConf, and she gave a really great walk through, step by step about how you compose an expression as a pattern to look for in some sort of input string, and she wrote a neat web tool that you can even follow along in

her talk to play with it. Probably the best thing to do when you're learning regular expressions is if you have something that you want to match, and then you have a pattern to match it with, and you just keep trying this until you start learning how the syntax works.

Emily Lewis: [Agrees]

Jonathan Penn: As you go, it starts to click and then you can use it in real situations.

Emily Lewis: Now, to take a step back to talk a little bit more about rewrites, are there any general best practices for doing URL rewrites in .htaccess?

Jonathan Penn: Best practices, do you mean like search engines or do you mean like technically?

Emily Lewis: I mean, I hadn't really thought about search engines. I wasn't sure of that. I mean, I'm aware of search engines appreciating kind of keyword-friendly URL structures, but I was thinking just like in your .htaccess file, is there a way of writing rewrite or writing your regular expressions that's better than the others. I mean, you mentioned keeping your regular expressions not too clever.

Jonathan Penn: Yeah, shorter is better obviously, and the fewer the rules is best because really you want to write the stuff for the humans, the poor humans, who have to go in and maintain, it could be yourself, it could be other people. It's funny, even though the show is about .htaccess, if your content management system supports handling its own redirects, I would recommend starting with that.

Emily Lewis: Oh.

Jonathan Penn: And you can say, "Oh, my content used to be here, but now it's here," because that's usually in your domain where you can handle it and you understand it. The rules are clear. It's part of a larger tool that does that. It's not necessarily as fast. The way it works is the .htaccess file is read on every request that comes in, and it's processed in and handled before the scripting engine even starts up to run to process what it should do and generate the content. So its speed was a

concern, you could do all your rewrites in .htaccess and it's very, very fast. If your concern is like I have a whole bunch of things I'm worried about that I need to move and I'm afraid of this file getting too big, just use your content management system to manage the renaming for you.

Lea Alcantara: I feel like that leads to some issues or concerns maybe with accessing the .htaccess itself with security because do you need to do any type of configuration to say, "Okay, CMS, you\re allowed to read and write an .htaccess file."

Jonathan Penn: [Laughs] Yeah. I know that WordPress writes an .htaccess file for you, and I would assume that ExpressionEngine does this too, and usually leave it as it is because it's putting in the right rules.

Lea Alcantara: [Agrees]

Jonathan Penn: Like actually, you can use .htaccess to do access control of its own like say, "This files are not allowed to be read." And by default, you don't want .htaccess files to be downloadable over a web browser.

Lea Alcantara: Yeah.

Jonathan Penn: Because then they can read. There might be things that an attacker or someone nefarious could divinate from that file about your web server and then try to exploit somehow. So a lot of the things, the defaults that are generated by many content management systems are sufficient, and then you'd go in and make changes as you need to.

Emily Lewis: Now, what about protecting that .htaccess file? I don't even know if this is possible, but like could some sort of machine or search engine or a bot access the .htaccess file on the server somehow?

Jonathan Penn: That's a great question. There are two level, or how do I say it, not levels, but there are two forms of accessing files sitting on the web server. The one is a browser making a request, and then Apache tries to serve the request, and by default, any physical file that's in your directories in the document route can be sent back over the internet to the browser and then it gets downloaded or displayed, depending on what the file is.

Lea Alcantara: [Agrees]

Jonathan Penn: So what you can do is you can tell Apache not to send back certain files. You specify it in the configuration language, and you just say, "Deny, no one should have access to this. There's no browser in the world including search engines should be able to spider this and download it." But that does not say that the file is unreadable.

Emily Lewis: [Agrees]

Jonathan Penn: So if your content management system lets users browse files or edit files on your web server, they can still see that as long as it's all part of the same directory that they have access to. So it's just going to be clear that the .htaccess file stops Apache from sending the file over the internet from a simple URL request, but there may be other ways of reading the files like through the normal CMS, whatever scripts you have running that do things on the web server. There are different sets of permissions you would use like the low level Unix permissions to say this file is readable or this file is not readable, things like that.

Emily Lewis: So that deny statement, does that go in the .htaccess file or is that something that's done on the server itself somewhere?

Jonathan Penn: It goes into the .htaccess file. There is like these little configuration blocks and you would just say, "All right, for this list of files or for files that match this pattern," and you can just say

.htaccess is one of the files, and you would just say, “Deny from all,” and that would mean anybody coming in from any browser anywhere trying to pass in the URL for .htaccess to download it, it would get a 404.

Lea Alcantara: What I find a little weird though is why isn’t that default?

Emily Lewis: [Laughs]

Jonathan Penn: [Laughs]

Lea Alcantara: Why isn’t that like default configuration from Apache? There are certain things that are just default behavior, so why we would have to specifically say, “Browser, don’t show .htaccess.”

Jonathan Penn: That’s a great question, and I think actually it brings up the larger picture of how these configurations bubble up. It is a default from many installations of Apache. The problem is if you’re on a shared hosting service.

Lea Alcantara: Oh yeah.

Jonathan Penn: Somebody else is maintaining it, they might have at their administration level, at the root level with the root configuration file that you can’t get access to, it’s locked away in a directory that you wouldn’t see, it might have rules to expose .htaccess for some other technical reason that your needs don’t matter because you’re on a shared environment. It’s a different kind of a game there. But you can override that locally. You can say, “Oh yeah.” And there’s no problem saying, “Oh, all my .htaccess files from here on down the directory structure, all of them should be denied.” That’s fine. I mean, even if the server is already configured, otherwise, the safety of just doing that is better.

Lea Alcantara: And I think with my limited knowledge of .htaccess shows that you can have multiple .htaccess files so you can say, like the permissions cascade down like within your own server. So if you put something at root, that's going to apply to every subdirectory, right?

Jonathan Penn: Yeah, and then you can override in sub-subdirectories if you want to, which can be a high cognitive load depending on how you structure it so be careful.

Lea Alcantara: Yeah, yeah, yeah.

Jonathan Penn: But that's a really useful feature. You can get as complex as you need to for certain goals. The dilemma though because Apache can't respond to a web request until it processes all the .htaccess files up to tree that it needs to, to make sure that it's serving…

Lea Alcantara: Oh yeah.

Jonathan Penn: It's either denying or allowing or whatever it needs to do, so that means that assuming you have a 10-level deep nested directory structure with ten .htaccess files, every request to the deepest directory has to load all ten files just to double check. I mean, for most people, that's not a big deal. I mean, when it's time to scale up because you're really, really popular, you're getting tons and tons of hits, you're probably going to need a developer operations team anyway.

Timestamp: **00:20:03**

Lea Alcantara: [Agrees]

Jonathan Penn: There are other issues going on, but that's something to be aware of. Like Apache, it has to load the file every time, and that's what you want because just in case something change, it's double checking.

Emily Lewis: You talked a little bit about protecting the .htaccess file with self denying a browser from displaying it. Does the .htaccess also provide a solution for actually prompting browser request to then authenticate a user to then view that request?

Jonathan Penn: Yeah, yeah. It's that old school way that pops up the alert that says, "Hey, type in a username and password."

Emily Lewis: [Agrees]

Jonathan Penn: Which isn't used very often nowadays because we like to present login forms within a branding of some sort. We don't really get a chance to change the branding of that popup. It's part of the original specification of HTTP to say, "Oh yeah, you can log in this way." And there is a way to do it with .htaccess, although I don't recommend it because you have to generate a file in a certain format and you have to have passwords encoded and hashed in a certain way, then you specify that file has to be hidden somewhere where no one can accidentally browse it because then otherwise there are ways of hacking the system that way then. In general, I would just say use your content management system's access controls because that's what the tool is for. It's within the domain of what you're working in. Yeah, it just works better that way.

Lea Alcantara: I do believe, and this is something I learned in the last ExpressionEngine conference that I went to, especially if you're doing Version Control and you've got a staging server that you show your clients that you can also restrict IP addresses so that only your client's IP can view this particular like URL, and if your IP doesn't fit, then no one can see that.

Jonathan Penn: Yeah.

Emily Lewis: Can you do that with .htaccess?

Lea Alcantara: Yes, I believe so.

Emily Lewis: Oh.

Jonathan Penn: Yeah, you can. You can, and that works really well, and in some ways, it's kind of like authentication of its own kind. [Laughs]

Emily Lewis: Yeah. So Lea, that situation you just described, is that like to protect a staging site from search engines or just anybody?

Lea Alcantara: I think both. I think part of the presentation that I saw, it was saying that search bots to spider the web to make sure that search engines find your site can be really aggressive and if you don't hide a staging server, it might actually think that's the real client site, and it might start showing up in Google unless you like really lock it down either with your CMS and just close it down or whatever, or if it still needs to be accessed for whatever reason that just restrict the IP, so only your client's IP can see it.

Emily Lewis: [Agrees]

Lea Alcantara: Yeah.

Emily Lewis: So continuing on with some of the other potential uses that .htaccess offers, can .htaccess do anything with like server error messages? We had an episode last year, I think it was last year.

Lea Alcantara: Yeah.

Emily Lewis: Where we talked about some of the error pages that you can force ExpressionEngine to present depending on the URL that's accessed, but beyond what the CMS can do, can .htaccess allow you to have any specific messages if this certain URL is requested, Jonathan?

Jonathan Penn: Yeah, you can. For any status code that could be interpreted as an error, you can specify a different HTML or you can even point it to a script if you want, and I believe a lot of CMSs do that automatically. When they generate the .htaccess file for you, they're telling Apache, "Hey, if something goes wrong, tell me about it because I want to figure this out." So most of the time, you just want to use the content management systems because that brings in the browsing, the sitemap, the navigation, all that stuff.

Emily Lewis: [Agrees]

Jonathan Penn: The one situation where you would have to use a text file or an HTML file with a custom error message is for a 500 Error, which just usually means the server couldn't even run the script.

Lea Alcantara: [Agrees]

Jonathan Penn: Like if for some reason, somebody left a syntax error in the PHP code of ExpressionEngine, well, there's no way that an error page could be generated so Apache will go and find – I forget what the default error is, but it's very unhelpful just to plank away page with some block text's very disconcerting error and you can specify that error instead and say something a little softer of communication to the user, "Oh well, something went wrong in our end. We've been notified. We're looking into it." You can say things like that.

Emily Lewis: [Agrees]

Jonathan Penn: And then you wouldn't necessarily need all of the dynamically generated branding parts and navigation parts to help the user because there's nothing you can do.

Emily Lewis: Right. Now, another thing that I mentioned at the very beginning that my first introduction to .htaccess was the indes.php rewrites for ExpressionEngine, but then a little later, I

started writing HTML5 media like audio and video, and I learned about having to specify MIME types, and that .htaccess fit in with that a little bit. Can you talk to our listeners about how .htaccess can tell the server how to handle these different file types?

Jonathan Penn: Yeah, yeah, and part of the reason you have to do that is because file extensions are a very poor way of determining what's inside the file and how it should be interpreted.

Lea Alcantara: [Agrees]

Jonathan Penn: And so there's a lot. I mean, .doc, is that a Word document? Did somebody just named their text file that for fun? You know?

Lea Alcantara: [Agrees]

Emily Lewis: Yeah.

Jonathan Penn: There could be a whole bunch of reasons, and so the MIME types were coordinated as a way of being able to say, "Okay, with this, we have a type like a document type, and then you can have a Microsoft Word as a subtype." And then you specify all these different types for different things like images and you can have PNGs, image is JPEGs, and for videos and things like that. So what happens out of the box is Apache has a bunch of these MIME types all ready to go with certain extensions just to make it easy. When we write files and throw them up on our web server, we usually know what we're doing. We name things .jpeg for JPEGs, things like that, and so Apache will automatically send back the file with the proper content type header saying what the MIME type is and then the browser can figure it out. But for some video files, like, let's say you're using the Ogg Vorbis video codec, and you have the .ogg extension, I don't know if Apache supports that out of the box, but let's say it doesn't, and for some reason when you try to view that file, instead of loading up the video player, the browser just shows the big string of "Aargh" [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Jonathan Penn: Now, we've all seen that.

Lea Alcantara: Yeah.

Jonathan Penn: And that's where the server doesn't know what to do, probably sent it back with the content type of text plain, and the browser said, "Okay." So what you're doing is you're helping the browser and the server coordinate more and you're telling Apache in the .htaccess file, "If you see the .ogg extension, it's of this MIME type." Then the browser is telling it doesn't know how to use that MIME type, but by and large, we've got some really good coordination between server and browser vendors, and yeah, that's how that works. So if you have a custom file type, you set the MIME type to make sure the browser knows what to do.

Emily Lewis: Now, another thing that I understand that .htaccess can be used for is like for caching to sort of optimize what maybe your CMS output is.

Jonathan Penn: Yeah. There are ways, and this also dives deep into the realm of the HTTP protocol, so there are ways that servers can tell browsers, "Hey, this hasn't changed." Or to be proactive and say, "This won't change for the next – I don't know – like ten days." So it's possible to go into the .htaccess file and explicitly set these headers for certain file types or just for files that matches certain patterns. So let's say you had a directory full of images. You can say that, "Oh, these files aren't going to change." All of the images in this directory need to be sent back to the browser with a certain cache header that says, "Don't expire cache for two weeks."

Emily Lewis: [Agrees]

Jonathan Penn: And then the browser could extract that.

Emily Lewis: Is this one of those things that's kind of like the authentication you talked about earlier where it might be smarter just to use what your, for example, CMS offers in terms of caching, or is the .htaccess, because it's called as like the first access a good way of going?

Jonathan Penn: Well, yeah. On that, it depends. As with most things, it depends. I like the idea of using the CMS because, again, it's within what you know, why get outside that unless you have a good reason to. The situation where you might need to is for speed. If you're serving up a lot of content, that you have to get out like large files, and media files are the great examples where you do this, and they're not generated by CMS, so using an .htaccess file to specify that would be good.

Lea Alcantara: Yeah.

Emily Lewis: [Agrees]

Jonathan Penn: For the most part, if your content system has rules where you can set a flag to say, "Well, I'm not going to change this for two weeks," and then it generates the right headers to tell the browser, that's within the tool. It's stored with your data. It's all in the same place, then that's generally better.

Lea Alcantara: I know that we use Cache Control for this podcast because the MP3, once they're out there, it doesn't need to be changed every single day. My question is, and this is like whenever I applied it, I just use whatever the example was, and it seemed like this random date. Because you mentioned like expire in two weeks or you can say it to expire in two weeks, you can expire in one day, you can expire in 30 days. How do you know? [Laughs] How do you choose like whether it's good to expire in 30 days or one day or one week or like two years?

Jonathan Penn: That is why server management is a career unfortunately. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Jonathan Penn: The general rule of thumb is you measure the number of request. What you're worried about too when you're doing caching like this is you're worried about repeat request from the same browser.

Lea Alcantara: [Agrees]

Jonathan Penn: If you have a thousand people who make one-time visits and they all download the stuff, the caching doesn't help.

Timestamp: **00:30:03**

Lea Alcantara: Yeah.

Jonathan Penn: Now, granted, there are proxies and others, there are endpoints along the way, or I guess they're not endpoints, there are midpoints along the way between your server and the browser, they might listen to those cache headers to decide and that it might affect other people too in a useful way, but by and large, what you want to use these cachings for is if the same person is downloading the same MP3 file over and over again, that might mean they have a podcast player problem. [Laughs]

Lea Alcantara: [Laughs]

Jonathan Penn: But in that situation, you would just check to see how often do they download it. If it's within the span of a month, then maybe setting it for 30 days is great, and for things like MP3s, you're not updating them ever.

Lea Alcantara: Yeah.

Jonathan Penn: As far as I know.

Lea Alcantara: Exactly.

Jonathan Penn: Yeah, yeah.

Lea Alcantara: Yeah.

Jonathan Penn: And this gets a little outside of the .htaccess, but if you're worried about large media files and content caching, that's when I would just say, "Well, you can look into other services like Amazon S3.

Lea Alcantara: [Agrees]

Emily Lewis: [Agrees]

Jonathan Penn: Or things that just, you know. If they get the same request and it gets downloaded again, that's their problem. This is when you're worried about your own server's load and if you see a pattern of the same people accessing the same files within a certain range of time, that gives you at least a window of time to experiment with to see if you can cut down some of the cost to your server.

Lea Alcantara: So this kind of leads towards speed situations, when we're downloading and things like that, I do believe we spoke a lot about different ways to compress things in ExpressionEngine a few episodes back, and one of the ways you could do that is adding some chunks of code in your .htaccess to configure gzipping and expires headers which is what we've just talked about. Are there more ways that .htaccess can help speed up a site?

Jonathan Penn: That's a good question. Not really.

Lea Alcantara: [Agrees]

Jonathan Penn: Not by itself. Maybe you could. One thing I've seen, I have not done as much work with ExpressionEngine. I have done some work with WordPress, and there's a plugin you can use

with that. Like when you make a request to a WordPress page, it will generate the HTML and stick it in a temporary directory, and then you can use rewrite rules to tell Apache, "Well, if there's a file here with this name, read that. If there's not, talk to WordPress." And then WordPress generates the file and every future request will talk to the file until you delete the file because you want to bust the cache and regenerate it.

Lea Alcantara: Yeah.

Jonathan Penn: But that's scary. It gets into some conflicts. [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: Sure.

Jonathan Penn: You have to be careful with that kind of stuff. But other than that, I mean, gzipping, cache control, I think the real bottlenecks are not where the .htaccess file normally are effected, it's where your CMS' content generation is, and then if you're really worried, if you have hundreds of thousands of users trying to access the same content, you might want to look into a content delivery network like Amazon CloudFront.

Lea Alcantara: [Agrees]

Jonathan Penn: Push the content, CloudFront takes the content that you have, and actually distributes it out closer geographically to where the people are, so it speeds up access for them. It takes the load off of your server so that people aren't just reaching into your poor little box that's wilting under the strain.

Emily Lewis: If you're using something like you suggested like a content delivery network, should you just not even worry about putting things like the Cache Control in your .htaccess, or the gzip and expires headers? Like does that become moot at that point, or does it become redundant?

Jonathan Penn: That is a good question. For my usual mode of setting this up, it's just to turn on gzip because that is really useful especially on mobile networks where you're dinged for bytes that you download, and the speed of the download and the actual payload over the network is more important than anything else. So turning on gzipping is fine. As far as, yeah, maybe like setting a week for cache expiration for media files, letting your content management system handle the rest. Most of the time you don't have to worry about that, and then if you have a content delivery system in the mix, well, depending on the network I guess too, there are rules that you can set to say, "Oh, you should check me every once in a while." [Laughs]

Lea Alcantara: Yeah, yeah.

Jonathan Penn: And that's beyond the scope of .htaccess. If you already have a CDN, content delivery network, set up, I would ignore .htaccess. Don't worry about it.

Lea Alcantara: Interesting.

Emily Lewis: [Agrees]

Jonathan Penn: Except for gzipping. That's still a good idea.

Lea Alcantara: I have a question about gzipping. Just like the deny accessing .htaccess, why do we have to turn it on? I feel like everyone I speak to say, "Gzip your site. Gzip your site." So why isn't that on by default?

Jonathan Penn: Well, again, we are at the mercy of the shared hosting solutions. [Laughs]

Lea Alcantara: Okay.

Jonathan Penn: Because depending on their goals, gzipping uses CPU energy.

Lea Alcantara: Okay, yeah.

Jonathan Penn: Or every time you request, it has to do the zipping.

Lea Alcantara: Okay.

Jonathan Penn: It's not really that bad, but if you're a shared hosting service trying to pack 500 low-traffic sites under one box, it might not meet your needs to spend the CPU cycles by default in and of itself.

Lea Alcantara: [Agrees]

Jonathan Penn: I'm not quite sure otherwise. I tend to just turn it on because it makes a lot of sense, and I also run my own servers, so I buffer for that expense. [Laughs]

Lea Alcantara: [Agrees]

Emily Lewis: [Laughs]

Jonathan Penn: It's part of my planning.

Emily Lewis: Now, you mentioned a couple of times that it may be better to rely on your CMS or the CDN to handle some of the different needs, but if you chose to rely on .htaccess, are there any potential issues or concerns you should be aware of before you turn to .htaccess for handling your redirects or your caching or even the authentication. Is there anything that you should be concerned about and at least aware of to check for?

Jonathan Penn: I would check to see if your existing tools do what you need first.

Lea Alcantara: [Agrees]

Jonathan Penn: Because the .htaccess file, when you break it, it's very hard to debug why.

Lea Alcantara: [Agrees]

Jonathan Penn: I mean, there's a whole host of historical decisions that were made that make it what it is today, and you kind of have to be historian to dig back to understand some stuff to unwind it. So usually, yeah, I mean, if your content management system generates it for you, it knows what it's doing. These tools are maintained for this purpose so that you don't have to think in this low level. Most of the time, just stick with what you've got, and then there are times like if you switch content management systems, that's often when you have to get in and mess with the .htaccess file yourself.

Lea Alcantara: [Agrees]

Jonathan Penn: Because the new management system probably won't know how to rewrite the old links.

Emily Lewis: [Agrees]

Jonathan Penn: So you're going to sit down and do that magic yourself and then let the new system handle it from there on out.

Lea Alcantara: So another common thing that .htaccess is used for that I've played around with is to access server-side settings like increasing PHP memory limit.

Jonathan Penn: [Laughs]

Lea Alcantara: I feel like almost all the time I've had to increase it, and I get annoyed that it isn't given to me immediately. Because I mean, obviously, if I could change it, that I had access to that. But usually, these types of things require a hosting situation that allows you to access something like php.ini or has settings in place that allows you to alter these settings via .htaccess. So are there any reasons why you would use .htaccess over php.ini when changing these types of settings?

Jonathan Penn: If you have access to the PHP configuration file, that means that you're probably the administrator, and then you just make the change there.

Lea Alcantara: Yeah.

Jonathan Penn: Well, it changes the setting for every site, for every PHP site you're hosting.

Emily Lewis: Oh.

Jonathan Penn: That's why a shared host sets it at a lower value. I mean, the reason they're setting it low is because they're thinking, "Hey, I'm going to pack X number of sites on this box. I want to just to, let's say, by default, I'll use this amount." And the fact that they let you go in and change it later, you can disable the ability to change it later in .htaccess file, and I'm surprised they don't do that considering that everybody could have increased it and then cause problems.

Lea Alcantara: [Agrees]

Emily Lewis: [Agrees]

Jonathan Penn: But yeah. I mean, if you don't own, if you're not running the server yourself, or if you have two different sites with different needs, then doing it in .htaccess is the normal to do it. That's for site-specific goals. That's what it's for.

Lea Alcantara: [Agrees]

Emily Lewis: [Agrees]

Jonathan Penn: And in this case, if you're not running the server yourself, you have your own goals, so you set the file and make it work.

Emily Lewis: And is there anything that we should always, no matter what, include in .htaccess file? Even if you're not doing rewrites or cache control or anything like that, should there be something that's always in it?

Jonathan Penn: Yeah, the configuration option to deny access to the .htaccess file itself. That should be at the top. Just for safety, say, "Hey, I don't know how the main server is set up, but we're just going to make sure no one can access this."

Lea Alcantara: So the other question would be, are there certain things that we should double check before adding in an .htaccess file?

Jonathan Penn: Yeah, just double check your CMS if it's got what you need or if you can get a plugin that does what you need to, just use it, and don't worry about it. If you think you need to drop into the .htaccess file, you're now making management decisions that might require more hosting knowhow.

Lea Alcantara: [Agrees]

Jonathan Penn: Like once you reach that point, maybe you need to think through your strategy of should I involve maybe higher end developer operations person to help me at least audit my set up. I mean, if you want to drop down to .htaccess to do security management, why? If there is a legal reason and you need to help manage your accountability in that, then you probably would want to pay someone to audit your setup just to double check.

Lea Alcantara: Sure.

Jonathan Penn: It's worth it because you don't want to get sued in case you have a client's confidential documents hidden away somewhere. You see, for the most part, you don't really have to go in there except for those times when you're switching CMSs. When things that your CMS can't do, you have to reach down beneath like the 500 Error or the 500 Status Code Error.

Timestamp: **00:40:02**

Emily Lewis: [Agrees]

Jonathan Penn: The things when the CMS can't run and can't figure out what to do, you're kind of stuck. That's when you've got to go down to the lower level.

Lea Alcantara: Awesome. We have spoken a lot about .htaccess. So for our listeners, we just want to tell you that all the links to the resources that we talked about will be in our show notes. But before we finish up, we've got a rapid fire ten questions so our listeners can get to know Jonathan a big better. So are you ready?

Jonathan Penn: I hope so. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: It's going to be cool. It will be fine. First question. Mac OS or Windows?

Jonathan Penn: Mac OS.

Emily Lewis: What is your favorite mobile app?

Jonathan Penn: The SMS app. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: What is your least favorite thing about social media?

Jonathan Penn: [Laughs] Recruiters.

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs] Oh, what profession other than yours would you like to attempt?

Jonathan Penn: Drummer. [Laughs]

Lea Alcantara: What profession would you not like to do?

Jonathan Penn: Trumpet player. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs] Who is the web professional you admire the most?

Jonathan Penn: Eric Meyer.

Lea Alcantara: What music do you like to code to?

Jonathan Penn: Anything with a floor to floor driving beat that drums out with no lyrics.

Emily Lewis: [Laughs]

Lea Alcantara: Cool.

Emily Lewis: What's your secret talent?

Jonathan Penn: Oh, there are so many. Which one do I pick?

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Jonathan Penn: Now, I think my secret talent is I am very good at improv in spite of looking awkward in public. [Laughs]

Lea Alcantara: Nice.

Emily Lewis: [Laughs]

Lea Alcantara: What's the most recent book you've read?

Jonathan Penn: A Handmaid's Tale by Margaret Atwood.

Emily Lewis: Oh.

Lea Alcantara: Oh, I love that book.

Jonathan Penn: I just finished it.

Emily Lewis: Lastly, Star Wars or Star Trek?

Jonathan Penn: Oh no.

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Jonathan Penn: Okay, well, oh man. I think Picard shot first. I'll just put it at that.

Lea Alcantara: [Laughs] Nice.

Emily Lewis: [Laughs]

Jonathan Penn: Okay, I'll choose Star Trek. I grew up on Star Trek, but I fell in love with Star Wars too.

Lea Alcantara: All right, so that's all the time we have for today. Thanks for joining us.

Jonathan Penn: All right, thank you.

Emily Lewis: In case our listeners want to follow up with you, where can they find you online?

Jonathan Penn: I have a blog, well, for iOS development, which is what I do a lot now, at cocoamanifest.net, and then my company is rubbercitywizards.com. That's where we do a lot of things, contribute to open source and talk about what we're doing.

Emily Lewis: Great. Thanks again, Jonathan. It was a pleasure talking with you today.

Jonathan Penn: All right. Thanks.

Lea Alcantara: [Music starts] We'd now like to thank our sponsors for this podcast, Hover and Pixel & Tonic.

Emily Lewis: We also want to thank our partners, Arcustech, Devot:ee and EE Insider.

Lea Alcantara: And thanks to our listeners for tuning in! If you want to know more about CTRL+CLICK, make sure you follow us on Twitter @ctrlclickcast or visit our website, ctrlclickcast.com.

Emily Lewis: Don't forget to tune in to our next episode. We rescheduled with Jina Bolton to talk about refactoring web interfaces. Be sure to check out our schedule on our site, ctrlclickcast.com/schedule for more upcoming topics.

Lea Alcantara: This is Lea Alcantara …

Emily Lewis: And Emily Lewis.

Lea Alcantara: Signing off for CTRL+CLICK CAST. See you next time!

Emily Lewis: Cheers!

[Music stops]

Timestamp: **00:43:09**