Transcript

Jackie

Hey, everybody. It’s Jackie D’Elia with another episode of Rethink.fm. This is Episode 15. I have my very special guest, Morten Rand-Hendriksen who is a senior staff instructor at Linkedin Learning. Hey, Morten.

Morten

Hi.

Jackie

Thanks for joining me.

Morten

Good to be here.

Jackie

For those who don’t know you, and I can’t imagine there is a lot of people that listen to this show that don’t, is there anything else that you want to fill in about what you do?

Morten

Well, my main job is as an instructor at Linkedin Learning and Lynda.com, I also contribute to the WordPress Project in odd and unusual ways. I don't really write a lot of code patches for WordPress but I ask a lot of questions about decisions that are being made. I propose things that should be done. I also contribute to the Community Project, like we're working on something. I'm peripherally involved in a non-hard and heavy code sense in that project.

Jackie

Well, before we jump into that, I've just got a couple of questions for you. How did you get started teaching?

Morten

At random. Let's see. This must have been 2009, 2008, something like that. I was working on a project, just some random client project. My wife was, at the time, doing what eventually became vlogging. She was one of the first people in Canada to make a TV show on YouTube where she went around interviewing people. She was a producer at a TV show. Then she would take all the people that she couldn't get on the show and do separate things on YouTube instead. She got invited to this event for Microsoft. We were like, "Yeah, whatever. We'll go to the Microsoft event."

Then at the event, they were handing out these little business cards with an activation code from new software that no one had tried before. It was an alpha, so it was very buggy. They said, “This is our new web design and development suite, it’s called Expression. We want you to try it.” Then I took one of the cards. I said, “Sure. I’m starting a new project right now, so I’ll try. I’ll just build the project on this new platform so I can really beta test it for me.” I started doing that and immediately ran into some pretty significant problems because it was an alpha application, so hardly worked at all. I was like, “I should really document what I’m doing here so that they know what I’m doing.”

I made a blog that was called, at the time, just Blog.PinkAndYellow.com because Pink and Yellow is my company. I started just blogging about this Expression Web and Expression Design application. I was literally like, “I try to do this, everything explodes.” The funny part was it was you color pick inside the application. Instead of color picking what’s on the screen, it would color pick based on the screen settings for your computers, it was these random colors. It was very odd. I was documenting all these things and just putting it in blog posts everyday. Then after a while, I started getting a lot of comments back. The comments were very odd because they were super specific.

They were like, “So when you experience the screen picking issue, exactly what are you doing? Where are you screen picking, inside the application or outside the application?" I'm like, "That's a really specific question from some random person to ask." I went in and looked at the IP address and then I noticed that almost all the comments on my blog had IP addresses in Redmond which is where the Microsoft headquarters is. I answered back to one of them and I'm like, "Well, maybe you should just call me instead of us having this ridiculous conversation on my blog."

They called me and then I went down there to just show them what I was doing. That got me … Because of that, I became a official beta tester for Expression Web which was their front end publishing platform. From that, I started writing articles for them about Expression Web in addition to my blog. Then Pearson Publishing contacted the Microsoft team and said, "Hey, we want someone to write a book." Then they pointed Pearson at me, so then I wrote a book. Then I wrote another book, and then another book. We released four or five editions of that book. Then finally, I made a video tutorial for Pearson under the Inform It Labels.

If you go to Amazon and search for my name, you'll find all four books. The crazy thing is you can still buy these books even though the software doesn't exist anymore. I keep getting royalties for it and I'm like, "You really shouldn't sell this. This is not something …" I don't think it's even Pearson, I think Amazon acquired a bunch of these books. They just sit at a warehouse. They will just keep selling them until they're gone. It literally started with a blog. It's funny, if you go to my blog at Mor10.com and you go to the first blog post, that's a blog post about me testing out Microsoft Expression Web.

You can actually read the whole thing, see all these comments back and forth, and see when I realize, "Oh, these are people from the team." I think I even left a comment at some point going like, "Oh, you're the actual development team for this software? Maybe we should have a chat." It's somewhere in my blog. You can still see it. It was, like I said, totally random. It all stemmed from my blog. Once I started actually writing proper documentation that wasn't just me ranting about what I was doing in the blog but thinking very carefully about why am I doing these things, how does this actually work, I've just realized that this is actually way more interesting to me than building things is understanding why they are done a certain way and then figuring out how to explain it to other people.

Then from that, I went out to do a course. They asked me to do a course on Expression Web. When they asked me, I already knew that Microsoft was going to shelve it. I'm like, "Yeah. We shouldn't invest any money in this. Just trust me on this. This is not worth your investment. I can't say why but it will not be worth your investment so we should do something else." Then they randomly were like, "Well, you seem to know a lot about WordPress. We have a WordPress course that's outdated. Maybe you should do this because not a lot of people watch these WordPress courses so it's not a big risk for us to let you do that as a first course."

Then I recorded the WordPress essential training course. It was released in the second week of October 2010 I think. Then I got a call three weeks later that said that's the first course to ever earn back its full advance within three weeks or within the first month of release. Since then, it's always been one of the top two or top three courses on Lynda.com.

Jackie

Okay. Well, as somebody who has watched probably almost every course you've made on Lynda …

Morten

That sounds awful.

Jackie

At some point or another. I really have. I really have. Everybody who listens to me or have seen on other podcasts knows I'm a big fan. Just in Lynda courses in general, I mean, I've even shared with you, we were chatting. Ben Long is a photographer. He has got some great courses on Lynda.com. I was hooked on those when I was learning photography. That's always been my go-to source for learning. You did the course for WordPress. I've noticed you've done a huge mix of other types of courses that are outside of WordPress. I'm just curious, what's your mix of courses that you're currently working on? How do you decide what courses will be coming out in the future?

Morten

I'm not actually a WordPress developer. I'm a front-end developer that happens to use WordPress a lot because it makes sense to people I build websites for. Because of that, my proficiency for WordPress is extremely high just because I've done so much work with it. I've been with WordPress for so long that I know how it used to work, why it's working the way it does now. I can see where it's going. My passion, my core passion isn't specifically WordPress, it's just general communication through the web.

You have an idea. You want to somehow convey that idea to other people in a way that they can access and understand whatever method they want to and then they should be able to communicate back to you and using web technologies to facilitate that communication. That's what I'm interested in. A lot of that is just front-end development. Some of it is communication theory and user experience design.

My field of focus is all of that. It's not just WordPress. For the last few years, I've focused quite heavily on WordPress just to build out our library. Now, I think this year, my split is probably something like 60-40. I do 40% WordPress and 60% other things. I'm doing some big things on Java Script. I'm doing some emerging technologies. There is a bunch of other things happening. I'm shifting focus very much to either the leading edge or just past leading edge mark-ups, ESS, Javascript, HTML, see where things are going, how to use things now before they become the next standard, things like that.

Jackie

Yeah. One of your courses was on SVG Icons and switching them out with Fun Awesome. It wasn't a WordPress course. At the end of the course, you had a little movie or two that said, "Here is how you can integrate this into WordPress," which I thought was a great way to present that because on one end, you have an understanding of the topic outside of WordPress because I think it's really important to understand basic HTML, CSS, and Javascript on its own and be able to do something with it. Understanding how all of that worked and how you put it together to swap it out. You did a build process in there. I think you were using SVG. You were injecting SVGs in there.

Morten

Yeah. The course covers four or five different methods for doing the same thing.

Jackie

Yeah. I'll put a link in the show notes for it. I thought it was a great course. I like those courses that go outside the WordPress bubble and get you honed in on your skills on that level. Then you can look at it. I'll jump to another question that I did want to ask you is on the 2017 theme, I've dug in and started looking at some of the code in there. Typically develop using Genesis, so some of the things didn't apply in the same way. There were some really nice bits of code in there that I really liked.

You had done a course using social icons for a menu system instead of maybe using a plugin widget or something. I'm just curious what you thought about 2017 theme as a whole. I know you had one course where you actually pulled out some code out of there for navigation menu that you were swapping out from underscores. I'm just curious what your thoughts are on that.

Morten

The icon system that was built for 2017, so the system that uses SVG Icons not just for the social menu but for anywhere you want to use SVG Icons is a stroke of genius. It's developed by Sami Keijonen from Finland. It is extremely robust and very easy to use. The tricky part is there is four or five different pieces that need to be stuck together for everything to work. You need to know there is a bunch of PHP in two different files, then there is some Java Script that needs to be hooked in and a bunch of CSS. When all these pieces stick together, the actual addition of any kind of icon within your theme becomes very easy. It's built so that you can create custom fallbacks for each individual item.

You can say, "This icon needs a PNG fallback, so I'm going to provide a PNG fallback. And this icon, if it doesn't display, it doesn't matter, so I'm not going to make a fallback for it. This icon needs a text fallback so I'm going to create a text fallback for it." It's a really well-wired system which solves one of the main concerns that a lot of people have around icons which is what happens if people have old browsers because SVG isn't fully supported, which is a whole other topic which is really odd.

There are people in our community that are doing great work, that goes well beyond just WordPress because the solution that Sami created can be ported into anything else. As long as it runs PHP, you can use that same solution. You can also rewire the solution to use other languages if you want to. If you were making some front end Java Script, single page application thing, you could actually use the majority of his code and just convert it into Javascript and still use it. For me, that component of 2017 is really well-built and probably the best example of how to use existing technologies to push modern technologies forward. There are some other things in the 2017 theme that I don't agree with like the inclusion of a video header, which is probably the, pardon my French, but one of the worst things we're doing on the web right now.

If you go into the track tickets and you go into the chats and everything, you'll see me very very aggressively trying to convince them not to do it because it's bad practice. It's honestly something we shouldn't do. The argument that was made against it made for adding video headers in was everyone is doing it already. A lot of themes are doing it so we want to show people how to do it. My constant argument against it was simply, "I don't care if people are doing something that is bad. It's still bad. But this is still something we shouldn't do." There are a myriad of reasons why you shouldn't have a video header, most importantly, just performance.

My concern is when we do things in WordPress we set a best practice standard. WordPress is big enough that when we make a decision like for instance, switch two SVG Icons, number of switching SVG Icons on 27% of the web. Then all of a sudden, that becomes a thing. When WordPress shifts responsive images, that basically … You can watch the graph of websites using

ICG mark-up. It was pretty flat. Then WordPress shifts it and then it eventually had this super sharp line that went up because all of a sudden, it's being used all the time. Then the browser manufacturers go, "We need to handle this stuff right away." Right?

When we then jam video headers into WordPress, we're basically telling people, "Video headers is a good thing to do." It's not. It will never be. It's a bad … It's blingy. It looks fancy but it goes against accessibility principles. It goes against UX principles. It goes against performance. It's just a bad idea overall. The good thing is the original implementation of it was scrapped for something that's far lighter.

YouTube backing was introduced to ensure that people can use an external video service to serve up the videos instead of just hosting them on your self-hosted site — which would be terrible and very foolish to do. Still, it shouldn't be there. Now, it's there. Now, it's built into core so now everyone can do it. It's an open source project. You can't win all battles.

Jackie

Yeah. You don't have to use a video.

Morten

No, of course not. I don't think a lot of people are, but I don't think, on principle, we need to make more careful decisions about what we're doing because it sets examples for people. When things like that get built into core, now it's a feature that people will use. It will never be removed from core. Video backgrounds, which is what this is about, are purely visual elements that have no communicative value or purpose other than making things look good, right? That means if you're adding a video background into anything on the web, you're doing it purely to make your content look fancier.

Then you have to make a decision, is this something that's important enough for the experience that I should force it one everyone who visits or should this be something that only loads if the visitor actively engages with it? You'll see a lot of news websites for instance now. When they embed a YouTube video into their page, you see a screenshot of the YouTube video with a play button. Then you click on the play button and then it loads. Then you get the video container from YouTube. Then you have to click play again. People go, "Why are they doing that?" Well, the reason is if you put a video container from YouTube on your page, you're downloading the video onto that page the second it loads.

They know that so they're like, "No. We're not going to impose on you the download from YouTube because you don't need to buffer the first 30 seconds of that video unless you're going to watch it. So we're going to put a wall in front of it and say you need to interact with that wall to be able to open the video even if it increases the number of clicks necessary." A video background is the opposite of that. You're introducing a video component which is a heavy download and a heavy bandwidth component into the experience even though it has no communicative value. If you're going to do that, you would have to then say, "What is the bandwidth connection of the person who is visiting currently on?" Which we can't do because there is no system for it yet.

You'd have to ask, "Is the person able to choose whether or not to load this video?" They can't. Are we going to defer the loading of the video until every other piece of content on the page is loaded so that it doesn't slow things down? Well, that defeats the purpose of a video header because it needs to be there up the top. There are all these strategic and technical questions that come into play that need to be addressed properly. The team that we're building 2017 took all these things into consideration and implemented it in a responsible way, that's fine.

The problem is when we establish a standard like this, hand it over to other people who will not ask all these questions, will just do this without considering the implications of it. Unlike the team that we're working on, 2017, someone might upload a 50-megabyte video, right, and make it play in the background or make it take over the whole page or force it to play with audio. There is a bunch of problematic things.

Jackie

That even happens with background images.

Morten

Yup.

Jackie

People are publishing content, start uploading images and don't pay attention to the size of them. But before we go onto that, let me just jump back and I wanted to ask you, you had written an article about the need for telemetry in WordPress and understanding who is using WordPress and how they're using it. We had a conversation not too long ago about that.

Some really valid points you made were that most of the people that are making decisions about what we're going to do next and what we're going to add to WordPress aren't necessarily the people that are using WordPress every day. Why you read that article and what do you think this means for WordPress going forward?

Morten

Telemetry projects or proposal stems from just observing ongoing conversations around features and WordPress core, which is every time a new feature is proposed or someone proposes removing a feature or something has changed, there is this conversation around, “Well, are people actually using this or are people to use it? How are they going to use it? Who is going to use it? Is this going to be something that everyone uses or just a few people use?”

It’s a weird conversation because there is a lot of theory going on. People are proposing or not even theory, there is a lot of hypothesizing going on where people are just throwing out ideas and saying, “Well, I think that most people would use it like that,” or “In my experience, this is how people use it.”

There is no concrete data to refer to. Then some user testing is done but the user testing is typically done internally in the active WordPress contributor community. User testing is announced on the Make Blogs which are only accessed by people who really care about WordPress core development. The people that things are being tested on skewer heavily towards advanced users who are involved in the process of building WordPress itself.

As a result, we don’t really know much about the people who use WordPress every day for things like publishing content because we're not reaching out to them. We don't have any way of talking to them. They don't have any simple way of providing feedback back.

There is no button inside WordPress that says, "Provide Feedback," right? You see that in a lot of applications. You have a little fly out, your click button and then you go to user voice or whatever. There is no such thing in WordPress. There is no easy way of providing feedback back to WordPress unless you know how the system works. It all came to a head during the conversation about whether or not to remove the underline button in the WYSIWYG editor. There was a proposal to simplify the WYSIWYG editor so all the buttons that are up in the kitchen sink at the top of your editor, which I am 182,000% for because the structure of the WYSIWYG editor made no sense.

For instance, the drop down that gives you hierarchical content order was underneath the kitchen sink so you have to know how to click the button and then know how to use it. That should really be the first item. The WYSIWYG toolbar got scrambled. I mean, moved things around and reorganized. In that process, questions were asked about each individual button which is justifiable to say, "What is this? What does it do? Should it be there? What should we do instead?" In some cases, we looked at a button and said, "Well, this button follows HTML standards so that should be there. This button injects some garbage CSS into the page itself. That should go, right?"

Then there were conversations about what are these buttons doing? We got to this underline button. The argument that came up was underline is too similar to link underline. Therefore, it's confusing for user experience. To which I responded, "That's a developer problem? That's a problem of the theme developer not accounting for underline being different from the link underline?"

While that is the case, it is not because of the underline functionality itself. It's because of theme developers not accounting for this or not providing best practices. The solution, if that is the issue, the solution is probably to create some packaged style that comes with WordPress that distinctly differentiates it from the regular link underlines. Then theme developers can overwrite that style with their own class. Right?

That would be the way to solve a problem. That's not really on its own, a reason to remove the button. The response to that was, "Well, no one uses the button." At which point, I and other people asked, "Based on what data are you saying this?" Then they're like, "Well, no one used it. I've never seen it used." I kept asking, "Where is the data to support this claim?" Then eventually, you come to this 80-20 rule. Well, 80% of the users need to use this before it's valid. Otherwise, we shouldn't have it in core, which ties back to the WordPress philosophy which is it wasn't designed for 80% of the user base.

The 80-20 rule is a lie because we don't have data on 80% of the users. We don't even have data on 1% of the users. We can't use that as a rule. Yet every single major decision that has been made in WordPress ties to the 80-20 rule and is used by everyone all the time. That's when I said, "Well, if we're going to do this, we need data. What we should do is collect anonymized telemetry about how WordPress is being used. Anonymized telemetry literally means track how many times people click on a certain button or whether or not people click on a button at all or how many themes are being installed and how many plugins are installed but not activated, just things like that.

Then build a system that truly anonymizes the information. The information that's sent from your site doesn't have any information about your IP address or anything else. It just has the core data about clicking. Then it goes into a system that further anonymizes it so that there is no way for anyone to see that behavior and then tie it to you. It would also include things like preventing the system from collecting actual keyboard input so you can't store what people are actually writing. You can just see that they are writing something.

I made a huge proposal about how this could be done to fully anonymize it, how to make it opt in rather than opt out, how to secure it so that if people don't want it, they can jut uninstall the entire feature and then it won't be sitting inside WordPress core. It will actually be removed from the platform entirely, and how to handle the data when it comes in on the back end. Who should have access to it? Who can decide what to collect? The whole menagerie, just full coverage everywhere because I really think we need data.

It's irresponsible to not have this data at this point because when we make changes to WordPress, we're making changes for millions of people, millions, probably 10s or 20 or 30 million people will be directly impacted by any change we make. We can't make those changes without knowing how those people are using the tools. The proposal was made and got shut down by Matt Mullenweg because his argument was it doesn't fit in under the current three focus areas which are the editor, REST API, and Customizer, I think.

Jackie

Okay. But those three things, wouldn't it be really beneficial to find out who is using those and how they're using them?

Morten

Of course. Of course. That argument is nonsense. I'd tell that to his face, that is truly not a valid argument against this at all. In fact, that is an argument for telemetry. I think what he meant was we already have limited resources. Those resources are allocated to working on these three things. If we do this, it will take up too much time, which is really unfortunate but that's the nature of open source is a telemetry project is not as sexy as creating a better WYSIWYG editor. It's harder to get people involved in building it. It's much harder to get people involved in building it if there is no sign off from people higher up that will say, "We want to include this."

Until we get sign off from someone higher up that says, "This needs to be done now," it simply will never happen. That means WordPress development moving forward will distance itself further and further from the real user because the WordPress user base continues to increase. The increase in user base in WordPress is not an increase in people heavily involved in WordPress core development.

It's an increase in people who use WordPress and will never interface with the WordPress community at large. They will never go to a Word Camp. They'll never go to WordPress.org for anything other than to download WordPress. They might never go there at all. They will have no interface with the community. We have zero information about them. They are an entirely black box to us. Unless we have telemetry, that data will never be available.

It's something that needs to be done. I don't have the time or skills to actually build it. I have talked to a bunch of people who are very interested in building it, but at the same time, are weary of starting the project without sign off from someone higher up because this is a huge project that requires … It can't be shipped in bits and pieces.

It has to be done right off. It could easily be something you spend a year on and then it just gets shelved in which case, you've just wasted your time, right. It's really tricky and at the same time, extremely important. Come June in Paris, I'm going to make this a conversation topic with a lot of people to make sure that at least people have considered it in a more careful way and not just written it off as this is something that sounds like it's too much work.

Jackie

One of the questions I wanted to ask you is about the Customizer. We chatted about it. I see there is a push going to where you could edit your content in the Customizer, right. You're doing publishing functions versus configuration functions. I'm just curious. Why aren't they separating those things out and configuring the site or configuring how the site is going to work a completely separate thing than publishing content? I think if you want a WYSIWYG editor for publishing content, that's one separate thing. I don't understand why we're mixing it all together. I'm just wondering, where do you see all that going?

Morten

What do you think? Why do you think that's happening?

Jackie

My reason would be is they want to compete with easy WYSIWYG editors.

Morten

Sure. Ignore that part. Why do you think they want to put it into the Customizer instead of somewhere else?

Jackie

Those people that are building out the Customizer want to add all of those components there. I just think that even just splitting it off, I would love to see it when you open up WordPress, you have three buttons, publishing. It's configuration. It's administration. If you're just a content producer, you have one button to click on, you're in.

You don't load all of this other code that isn't even really being used while you're publishing. You don't need all of that other stuff loaded up in the dashboard. I'm just wondering, I mean if the goal is to compete with Square Space and other platforms that their goal is to make it easier for people to publish content as well, this seems like this is more of an obstacle than it is making things easier.

Morten

Yeah. I can only speculate here. I've observed a lot of these conversations but I can't speak on behalf of anyone else. I can only speak on behalf of myself of course. One of the reasons why this is being tied to the Customizer is simply because the Customizer is already a place where you are making changes to settings and database. You've already solved all the issues around authentication and everything. When you open My Customizer, you're basically opening a virtual space where you can see your website and make changes to it without anyone else doing. You're running this instance of WordPress inside your browser essentially, right.

It's all powered by Javascript. It's super complicated, the way it works. Whenever I see it, you should just go, "Open the Customizer. Then open your developer tools." Just look at the HTML mark-up that the Customizer produces for that pane. It's nuts because there are so many things going on. It's all being manipulated by a Javascript on the fly. It's a very fancy piece of technology that's running there.

I think that the reason why a front-end editing, which is basically what we're talking about, the idea that you can go into a post or page and edit content is being injected into the Customizer is simply because that is already a path that's open. I know that what people want is true front-end editing, meaning you open the front page of your site, you go to a post or page. You can just start editing the content right there on the page. There are solutions in existence already that can do that. I've built some very simple demos of how to do it. One of them is making its way into Linkedin Library right now. There are ways of doing this.

A lot of the ways of doing this are actually tied to the REST API. The Customizer existed way before the REST API. There is still a reluctance to use the REST API for a lot of things, for various reasons. I believe that much of the reason for trying to push all this editing stuff into the Customizer is simply that it's an existing modality people are familiar with or the developers are familiar with. They don't have to build something new. That said, I am firmly in agreement with you that there is a separation of concerns here that needs to be established, which is the Customizer is for customization of the application. That would be the customization of the look and feel of WordPress on the front end.

Whereas content creation and editing is a separate thing altogether. That conversation I've been part of. Generally, people just choose to make the umbrella that is customization broader and say that customization includes customizing the content itself. To be fair, there are actual very clear crossovers there. For instance, let's say you're setting up a new site. You just downloaded WordPress. You set it up. You installed it. You're going to start customizing your theme. Then you go, "Well, I want to have these four pages." Then you go into the Customizer. You start building out your menu. Then you realize, "I don't have the pages."

You have to go to the back in the WordPress and create the pages. Then go back to the Customizer. With 2017 and 4.7, a feature was introduced that allows you to create a new page directly from the Customizer so that you can add it into a menu. Then you can go onto the back end and edit it. The next step of that would obviously be you create the page and then you edit the page. Right now, that editing happens in the Customizer pull out menus. Then it updates on the page. Then that confines it to the Customizer space, which justifies the argument that this is still in the Customizer.

Once you move that functionality over into the preview, so you're not editing the content on the page itself, not in a little panel, the separation of concerns becomes very obvious. At that point, you're like, "That specific editing functionality should exist on the front end." Now, you could, in my opinion, do both. You could do it so that you can create content and manage content from the Customizer. That should be an extension of the front end editing capability. You build front end editing first. Then you migrate front end editing into the Customizer to allow people to continue it in the Customizer rather than forcing people to boot up the Customizer to do the front end editing.

That is a philosophical conversation that has to do with literal design philosophy, “What is customization? What does customization encompass? Is content editing part of customization? Is content creation part of customization? Where is the line between customization and just actual content editing? Who should have access to it? Who should not have access to it?” There is a bunch of very interesting conversations around that. Right?

Jackie

Well, that goes back to just having the data so that you understand how people are currently using things. I would really like to know how many people when they are editing a page or using the visual editor versus the text editor. I have no idea. I know the use the text editor because I like writing HTML but I’m assuming all of my clients would never be in there. But I don’t know.

Morten

To that, there is this very interesting thing that was happening. You know about the Gutenberg Project, right? This project to redesign the content editor and WordPress to make it block-based. What they're doing is something that looks very similar to what Linkedin Post has whereas you start with an editing page. Any time you hit enter, you get a new element and then there is a little plus sign. You click on a plus sign, you can open a thing that you can define, "This is a paragraph or a heading or quote or image or whatever." It's contextual so that you have the different types of content at the level you are working on them instead of in a toolbar at the top. Right?

It's an interesting user experience idea. It has pros and cons. I'm not going to get deep into that. What's important is part of the conversation that happened around the Gutenberg Editor was this conversation around what happens when you hit the return key on your keyboard as you're writing content. This was one of those places where I go, "If there was ever a reason if there was ever a proof of why telemetry is necessary, this would be a good example of that proof," which was a conversation around what happens when you hit the return key.

Where the majority of respondents said, "A Return key creates a line break." Everyone who has worked with a content editor is just like, "Under no circumstances are you ever allowed to make the content editor in WordPress have returned a line break." That goes against all standard conventions for content editing.

That is only the case in a text editor. The text editor is the only place where hitting the return key just returns you to the next line in all content editors. Microsoft Work or Word Perfect or Office 365 or God knows what, everything, a Return key creates a new paragraph. Shift return creates a line break. This is the standard.

Then you got a ton of push back. People who said, "No, it's not." Then we have to actively prove it, literally go into Word and tape ourselves hitting enter just to see how paragraphs are created when they hit enter here. This is not new. This is not something people are making up. This is an industry standard. Then they were like, "What industry? I don't think so. People can learn." It's like, no, no, no. You cannot design WordPress like this. If you ship a content editor in WordPress where enter is a line break, the ramifications will be so disastrous, right, because people would literally just go, "Oh, WordPress is broken now." Then they would just abandon the platform.

It's such an arbitrary decision. It just proves that the distance between a lot of the people who contribute to WordPress and the people who actually use WordPress on a day-to-day basis is so long and so vast that there is a complete disconnect about how things work, right. Then when you realize, "Oh my god, these conversations are happening." Then you wonder, "What other unsaid things are being assumed about how people are using content editors?"

The crazy part is the people who were very adamant about return returning a line break had every reason to be. They're not crazy people. They're not stupid. They're not misinformed. They have every intention of doing the best thing possible for the people who use the application. They simply work in an environment that doesn't have the return key creating a new paragraph because they don't create content.

If they do create content, they write it in Markdown where that isn't the case. Their experience of how WordPress works is fundamentally different from that of other people and because they have no data to cross-correlate their experience, they just don't know that it's … When I say don't know, people go, "Well, that's ignorance." It's not. The world is too complicated. There are too many different people. There are too many different ways of doing things that a single person can't have full knowledge of everything. You have to actively seek out other people that don't fit into your particular bubble and then figure out how they do things and then take their disclosure of behavior as a true statement about the world as they see it rather than some misunderstanding of how things work.

Then see how, "Okay, so I know that I do things a certain way. I know other people do things a completely different way. How can we solve the issue for both of them or is one way more prevalent than the other? Should we then make fallbacks for the other one?"

That's where telemetry comes in. This is why telemetry is so important because if we had telemetry, you would very quickly see that the frequency of people hitting return twice to create … It would be astronomically high, right. You would have billions of double clicks on return. Whereas in reality, you don't because the only place people would do that was if they were trying to make double lines and then it would just collapse on itself and they would stop doing that.

Jackie

All right. Well, I've got one more question I wanted to ask you for sure that I wanted to get into this episode, which is, "What do you think developers who currently work in WordPress need to be focused on in the future? What do they need to be learning now?" I know that's a broad topic depending on what it is that they do. If you're a developer, you work with clients, you build websites or you're creating plug-ins that you want to sell. What do you need to be focused on? What's on your radar?

Morten

Okay. Give you the super general answer first because I get this question all the time about … Two variations of it. As a WordPress developer, what should I learn? The other one is, what describes a good WordPress developer? The answer to both of those questions is the same. You need to learn how to build things without

WordPress. WordPress is a tool in your toolkit but it cannot be the tool you stand on when you do all your work. WordPress is literally a content management system that uses PHP, HTML, CSS and Javascript to create front-end content.

Technically speaking, WordPress works no differently from any other content management system. Your job as a content developer, designer, whatever you are, is to facilitate the communication between your client, WordPress and the person who visits the client's website. WordPress itself is a box you use as an interface between humans and a database. Your main job is to handle the stuff that happens on the client end. How the data that gets put into the database then gets returned out to the end user, that is where all the work is happening. Now that we have the REST API, you no longer have the bounds of WordPress and what WordPress can do to deal with. Right? The limits that WordPress itself imposed on us as WordPress developers have effectively been removed.

What I mean by that is if you wanted to do something crazy in WordPress, let's say you wanted to mix content from the WordPress database with content from some other source, you'd have to go into your theme or make a plugin that would hook into the theme. Then you would have to make some sort of PHP functionality that would then take the contents like it was returned from WordPress and the WordPress query. Then you would have to intermix it with content from a different source. Then sort out how that mixing should occur and then display in an HTML document that's shipped from the server to the browser.

Once you have the REST API, WordPress is just one of those data sources. With the REST API, you don't have to think in terms of a WordPress theme or WordPress plugins or anything else. You think, "I have an application that I built," in whatever language you choose. It can be PHP or ASP.net or RUBY or NODE or God knows what. You build that application. The application sends requests to the WordPress REST API and gets that in return. Then you can take that data and co-mingle it with whatever other data you get from any other source.

Smash it together, display it in any way you want. The REST API throws out WordPress essentially. WordPress becomes a tool that you just reference for data. The reason why we don't want to use it is for the backend because it's easy. It has a good user interface for people who are managing the content itself. They don't have to worry about anything. You can configure the REST API to give you the data you want, in the order you want, in the structure you want. Then you just display it however you want.

What I'm doing at Linkedin Learning right now is building a new path for people because up until this point, we've been building WordPress content that focuses on

WordPress development within WordPress proper. How WordPress itself works, how you extend WordPress with plugins and themes, how you build your own themes and plugins to get it to work the way you want to and how you do things further with it. Now, I'm building another path that says, "Let's treat WordPress as a REST source."

Understand how the REST API works meaning how you get the information you want from WordPress through the REST API and how you capture information and format that data. Once you have the REST API information, you then step up to the next step which is to send authenticator request into WordPress for the REST API meaning you can alter the contents of the database from outside WordPress.

You can build a standalone application that lives outside WordPress and then you can say edit the title or edit the content or add a new user or do whatever you want. That data then gets sent through the REST API back into WordPress in a secure way so that it's still only available to the people with the right authentication level and the right user role. But it's not happening inside WordPress admin anymore.

Once you have those two pieces, you don't need to work with WordPress at all. WordPress just becomes a source for your JSON data and it can do whatever you want. That opens up this whole new world of development that the WordPress community has basically been shielded from which is now you can build standalone applications that live either on the web or on mobile devices or pretty much anywhere else you want.

You're not confined by how the WordPress loop works or how the WordPress meta data works or anything. You could literally use the data in any way you want and mix it in any way you want and display it in any way you want and make whatever requests you think are rational to the REST API. Once you have a good handle on the REST API, you can just learn whatever you want. That means you need to then migrate away from building WordPress themes and plugins into how do I build phone applications or watch applications or audio only applications or whatever it is, or Internet of things. It just opens the world to all these other opportunities.

You very quickly realize that once WordPress is a REST API, the things you can do with it are no different from what you can do with any other REST source. The majority of data sources on the web are REST based. All the things you now learn are applicable to anything else, right. Twitter has a REST API. Amazon has a REST API. Google has a REST API. Everything has a REST API. All that data is now on par with any WordPress site in the world. Where do people want to go? That's where you want to go.

You need to learn the REST API not as an extension of WordPress but as a portal into a standard REST API. Then you need to learn how to use REST data sources to build resilient web applications. Throw the whole WordPress … Think of it as two different things. WordPress is the source so you learn how to work with that thing. Then you learn front-end different around REST API which is something else that is not connected to WordPress. Then WordPress happens to be one of your sources of information.

Jackie

You're going to have some courses I'm sure that are going to help people understand how to do that, right? I know you already have one that I watched already there which was you were using postmen to pull data through the REST API and create a little standalone application. It was pretty basic but you could see that you could easily bring this data in into something completely separate outside of WordPress.

Morten

Yeah. I'm working on a course series now that are three courses so far. They're three courses in development. You have the original course which is the one you talk about, that's just called WordPress REST API that just looks at how does the WordPress REST API actually work. How do you extend it? How do you get data from it? How do you do something with that data?

Now, you say it's very basic. This is the funny part. The REST API is basic. That's the thing. Because once you make a request, you just get a JSON object. The second you understand what a JSON object is, I can give you a simple demo as like, "Here is how you display the title, the featured image, and the content." Right? Then if you want to display something else, the process is exactly the same. There is no difference. There is nothing complicated here. It's not like you need to have extra knowledge to understand how to get categories. It's just in the JSON data. The REST API is very well-built so for every single piece … Every returned item within an object that has further information, so it links to something, the link to that REST route is provided inside the REST response.

If you say, "Give me a post." In the post, there is an attachment, so an image. Then there is a REST URL route to that attachment post so you can go to the attachment post and get that data and return it back in. Once you understand the communication from WordPress to your application, you can do whatever you want. Then you understand how to add in new endpoints. No, sorry. New routes, and also how to augment the data that comes out of existing routes to make its structure the way you want. Then you can get WordPress to basically cough up anything you desire.

Then you can extend it into your custom post types and custom fields of whatever you want.

The next course that is currently in editing and will come out soon is on authentication where I cover different methods for authenticating requests into the REST API. Both cookie authentication that happens inside the REST API itself, sorry, inside the context of WordPress. You're basically piggy backing off the authentication methods into WordPress and how to use OAF too and a bunch of other authentication methods to send requests from outside WordPress in. That's coming out. Then the third part of that course is a larger course that looks at how to build a standalone app that does something that WordPress does not do.

Basically, you build the thing on its own that uses the REST API to both send and receive data in an authenticated way to do something totally different that falls outside the scope of anything that you would think of doing with WordPress and simply uses WordPress as a good interface that is secure because you just use WordPress because it's a secure database. It's a secure management system. You build something that sits next to WordPress that does a bunch of stuff.

From there, what I hope is that people realize once you understand these pieces and you see how it all sticks together, all these other courses in our library and all this other information on the web in general about how to build pretty much anything you want, become applicable to WordPress because we're no longer using WordPress. We're just using the data from WordPress. You don't really need to know anything about how WordPress works to get that to work.

Having said all that, what I'm seeing in the community now, which is not surprising and makes total sense is a lot of the people that are working with the REST API are still thinking about WordPress inside the context of WordPress. Sorry, they're thinking about the REST API inside the context of WordPress. They look at how to build REST-based themes which are really cool because you can build these super snappy themes that load content almost immediately, that just have new ways of interacting with them that are very very fancy, right. You're still confined by how WordPress works. The consequence of that is you often see themes that require you to have specific permalink structures to work or themes that will try to use the permalink that's returned from WordPress. That would be the human readable permalink that's set in permalink settings.

Use that to figure out what type of content this is and then provide the correct template based on that, right. That comes from thinking about the REST API as if it's an extension of how WordPress already works. If instead of doing that, you say, "Forget about all that WordPress stuff. The REST API itself returns to you all this information." When you request a post, one of the elements, one of the properties inside the object for that post is the permalink generated by WordPress.

Another property inside that post is the type of content you got returned. Another property is the link to the REST route, right. You can use that information if you treat the return from the REST API as an actual proper data point. You can create the same functionality without having to do any kind of weird stuff with forcing WordPress to behave a certain way or trying to use rejects to decipher what the URL is or anything like that.

It all sits there but you have to think about it as this is the primary source of information. It's not WordPress. It's not how the theme works. The primary source of information is the JSON object itself. Then everything changes. Then you're like, "I need to learn proper Javascript because I can build all new things with it." Not Javascript in the context of WordPress, just Javascript. I need to learn how to build apps for other applications.

I need to maybe extend into other languages that I previously didn't touch because there are things that you can do in other language that you can't do with PHP. It just opens the door to all these other things. I can see how if we do this right in our community, we can evolve our community away from just thinking about WordPress as this little box or an island separate from the rest of the world and extend it into just a tool on the web that's extremely powerful that we use as a data source.

Jackie

That is awesome. I can't wait to see the rest of the courses that you're going to put out because the first one just had me just going, "Wow. Look at this. You can do some really cool things." I agree, it opens up all the possibilities. It changes the way we publish content completely, right. It's the way you view published content. You have options now. A client just doesn't have to have a website. They could have an application that pulls data and shows it in a completely different way from their website. They could still have the website but they could still have lots of other functionality or combine data, like you said, in lots of different ways. That opens up the whole world for everybody.

People who are considering themselves WordPress developers might want to be rethinking their approach to that and saying they are a developer that's using WordPress. They're using the REST API. They're using a lot of these other tools to create and distribute content, help people do that. I think that's the message.

Morten

If someone is coming into WordPress now, so if you're listening to this and you're just starting out and you're getting into WordPress development, I would strongly encourage you to invest your time in learning how to use REST APIs, not in the context of WordPress. Just focus first on REST APIs in general because this is not new technology. I went to conferences in 2009 that were focused squarely on RESTful APIs.

Learning REST APIs in general and how you build things with REST APIs and then coming to WordPress and treating WordPress as the source of REST APIs, you'll have a huge advantage over old geezers who have been working with WordPress in the confines of PHPs this entire time because if you come at it from the outside in, you'll see exactly what you can do and how far you can go.

You won't be limited by the way that WordPress used to be. You won't be limited by how WordPress themes work or how WordPress plugins work. You're starting out by saying, "I can do anything I want. Now, I want WordPress to give me that data." That's my recommendation if you're starting out. Start with REST APIs and work your way back into WordPress rather than start with WordPress and work your way out. Now, that means you're going to go into a much more advanced field than what WordPress is because REST APIs are much more complex than WordPress on its own. But that's what our industry is now.

You can't go into the web design industry or the web development industry treating it as an easy thing to do. It is not. It's extremely complicated. You need to learn all these things that have nothing to do with what you want. You need to dive into performance. You need to dive into house servers, render images.

You need to know all these things that go well beyond development into content strategy and how teams work together and how content is produced and edited and parsed online and how you link into different types of content and then pull that content out. Then change it and then just hand it over to other applications. Think about WordPress just as a tool in your toolkit, just like anything else and you'll go far with this.

Jackie

Morten, thank you very much for being on this podcast. This has been just awesome. You had a lot of great information that you shared. If folks want to reach out to you, follow you, talk with you, engage with you, how do they reach you?

Morten

I'm on Twitter @Mor10 because that's my name. That's M-O-R-1-0. Mor10, right?

Jackie

Yeah.

Morten

I'm also on Linkedin, of course. You can find me on Linkedin if you just hit me up. Go on. Make a connect request and then type in, "I heard you on Jackie's podcast." Then I'll be like, "Sure. I'll make a connection with you." It's just you get all these requests that are like, "Be part of my professional network," I'm like, "Who are you? I don't know who you are." It's because I have a really hard time remembering people's faces and text. So …

Jackie

Okay. Here is everybody's chance. If you want to connect with Morten on Linkedin, mention Rethink.fm in there and you're going to get connected.

Morten

Exactly. I'll connect with you. I publish a ton of content on Linkedin Post, which is just the publishing platform on Linkedin. It's just like Medium except it's on Linkedin instead. I write content infrequently on my own blog at Mor10.com so some of my more opinionated angry stuff goes there. You should also go to that blog and read the original post because it's hilarious to read it now. You'll be like, “What the hell is Expression Web and why don’t have it?” You realize it’s like Shareware and no one can use it.

I also write stuff for the … What’s it’s called? The Linkedin Learning blog, which is more general, about learning and the industry. I’m speaking about CSS Grids in the context of WordPress at Word Camp Europe in June. If you happen to be traveling to Europe in June …

Jackie

Go listen to you talk about that.

Morten

Well, I discovered today that my talks are the same time as Andrew Nacin’s talk. So there is going to be five people at my talk which means you’ll have ample opportunity to ask questions afterwards. It’ll be on WordPress TV so you can watch it there because that’s one of my big focus areas this year is newer types of the CSS and other technologies. I have a large course on CSS Grid coming out to augment the existing course we have on CSS Grid that was released last year. I’m doing a course on SVG that goes down into hyper detail like, “What does a box look like in SVG code? How can you manipulate this using Javascript and CSS?”

Jackie

You know I'm going to be all over that because I love SVGs.

Morten

I'm going into things like element queries which don't even work yet because I think that's going to be a big thing. I'm trying to poke at new technologies that are coming around. Right now, CSS Grid isn't even future technology. It's truly something that runs in your browser currently right now. When you start thinking in CSS Grid the way that you make layouts on the web and what you can do with layouts changes completely. I want the WordPress community to adopt this right now because there is so many opportunities for doing things in a much more structured way and much simpler way. That's why I have this talk at Word Camp Europe.

There is an article that I'm writing that's going to come along with it that will be published at an undisclosed location in the near future which is huge. I really want anyone listening here to just throw away everything you know about web layouts and just invest all your time in CSS Grid. I'll show you how to do it so everyone can actually get it done.

Jackie

Awesome. Speaking of CSS Grid, Episode 14, which is the episode that aired right before yours did, Rachel Andrew was my guest. We talked just exclusively …

Morten

Yes.

Jackie

About CSS Grid. Anybody who is interested in that can go listen to that episode and definitely follow you for more information as it rolls out.

Morten

Yeah. Just to be absolutely clear, I defer all information about CSS Grid to Rachel Andrew. She is the one that knows how this stuff works. She is the one that's been pushing this from day one. If you need to know how CSS Grid works, that's where you go. You go to Grid by Example or you go to the MDN Network and see how all these articles are sprinting about it. They're fantastic. Then come to me and I'll show you how to think about WordPress themes in this context and how to think about just general layouts. I won't cover every single piece. I'll just tie it together in your practical ways so you can actually use it today.

Jackie

That's going to be very helpful because her videos that she has got on Grid by Example show you how it works. Then you need to take that next step to figure out, "Okay. How am I going to get this into WordPress? How can I take advantage of it in WordPress? How can I do that?" I think that will be some great followup with you. Maybe you can come back on Rethink and talk with us about that.

Morten

Absolutely.

Jackie

All right. All right, everybody, well, thank you. I know we ran over on this episode but it was just way cool to have you, Morten. I'm just totally ...

Morten

That's what happens.

Jackie

In awe. Thank you again for joining me on the show. I hope everybody else has a great week. I'll see you on the next episode.

Morten

Bye.