Adam Lowe - The No Code Way to Create Native Gutenberg Blocks with Pinegrow

Love them or hate them. Blocks are the future of WordPress. In this presentation, we're going to learn how to build some custom native and completely portable blocks, and we're going to do that with nothing more than HTML css, using pinero's Visual Editor and its WordPress block. My name is Adam Lowe, and I have a small web agency called Peak Performance Digital.

The concepts and the tools that I'm going to show you today are the same ones that I use for my clients. If you've been on my YouTube channel, you've probably seen that I create a lot of videos about Pine Grow. I actually started doing it as a way to document and train my team. It didn't take long though for Pine Grow and some of their users to discover my videos and start asking for more content and tutorial.

Then when Pine Gro started working on their WordPress plugin, they invited me into their closed beta and they allowed me to show off some of what they were building to generate interest from the WordPress community. Now, just so that there's no confusion, no, I don't work for Pine Grove. I don't get paid by them and I'm not incentivized by them.

If anything, I'm a pain in their butt, cuz I keep pushing them to do more to fix issues and to make their product friendlier for new users. I'm critical of them in the places where they need it and I praise them where they. . Now, before I go too deep into the why's, in the hows, let's start off with a quick project.

What you're looking at now is a basic installation of WordPress with a 2023 theme, pine Grove's, WordPress plugin, and some dummy posts created by Faker Press. Looking at our plugin list, you can see that there's nothing up my sleeve, and all I have loaded is Pinero's builder plugin. Now let's dive into the builder.

We'll start by clicking new project blocks for an existing site and custom CSS to keep things simple. Now you can see that we're inside the Pinero builder. It already has some content loaded since we chose to create a block project, so I can either keep this or I can delete it. For this demo, I'll keep it.

I'm gonna save this project and give it a name. Then I'm gonna change my workspace just a bit to make it more comfortable for me to use. I'm gonna grab this panel and move it to the left side of my screen. There, that's better. Now let's activate the WordPress builder function. Under project type, I'll select a plugin project.

I'll give it a name. Hello World, and the slug is gonna be Hello World. I'll leave everything else at the default and click save settings. Now let's start by taking a quick look at what we see on the screen. Well, on the left we have a tree view of the HTML document. This concept should be familiar to anyone who's ever written HTML or used a page builder.

You'll notice though that we have these little blue things with the WordPress logo. These are block actions and they tell Pine Grove that we want to do something in WordPress with these elements. For example, clicking on this section element, we'll see that we have a block action assigned when I switch over to the WordPress panel.

You can see that the properties for that block notice here that we're giving the block the name about me, and we're putting it in a section called Cool Blocks. This will be important to remember later when we add it to our page, I'll switch to the image block and we can see that we have block attributes attached to it, telling WordPress that we want to be able to choose our own image.

Same thing here for the H three element, the paragraph element, and the anchor element. Now let's see what happens when we go back to the WordPress menu and tell it to export the plugin. Amazing. Right. Plugin, export. Done. Okay, fine. Let's do something really cool then. We're gonna close this project. We'll head over to the plugin page and see what we see.

And there it is. It's enter Hello World plugin. I'll activate this so we can use it now we'll go to the homepage, edit the page, and we can see that our About Me block is right here. Let's add it to the page. In our block settings panel, you can see all those things that had block actions assigned are now editable.

Let's click on our image to select a new image. I'll change the name

and a learn more link.

We'll click update, view the page, and you can see all those changes took effect. But wait, there's more. See, there are plenty of plug-ins that let us do similar things. So what's so special about this one? Remember when I said that we're going to build a native plug-in? We'll check this out. Let's go back to the plug-in screen and now I'll deactivate the Pine Gro plug-in.

We're gonna leave only the Hello World block plugin that we created with it. So what do you think's gonna happen now when we visit the homepage or try to edit the page with most solutions? In the worst case scenario, we get a bunch of Unformatted HTML code on our. . In the best case scenario, the page would look okay, but we wouldn't be able to do anything with the block.

However, as you can see here, we don't have a problem with the front end and the backend works just the way that it's supposed to, and that's because the blocks that you create with Pine Gro are regular React blocks, just as you would write them by hand. They don't depend on Pine Gro at all. In fact, I'll switch over to my terminal and open up the blocks JavaScript file so you can see that we can make any changes we want, even without the Pinero builder.

But that's silly though when all we have to do is go back to our Pinero project, make our changes, re-export it, and we're done. Let's do that now. I'll just come back to our plug-ins page, reactivate the Pinero Builder plugin. Open up our project and we'll make a few little changes. I'll start with this anchor element.

It's great that we can change the link, but maybe I also wanna change the text to do that. I'll add another attribute and I'll call this more text.

I give it the label that I want to use and I say that I want this property to replace the content of the element. Before I move on, I want to show you this cool link preview button. Clicking on it brings up a code editor window so that we can see exactly what Pine Grow is going to do. This is a fantastic tool for troubleshooting things when they're not working the way that you expect.

I'll close it now. There's one more change that I want to make before we re-export the plugin. This paragraph element is kind of boring. Rather than just add text, I wanna be able to add any kind of block content here in the WordPress block editor. To do that, we're gonna replace this block attribute, and instead I'll use a block inner content Smart.

we'll save and export the plugin, and then we'll see what we can see. Coming over to the homepage, it looks like nothing's changed, but when I go to the block editor, you can see that our block has some issues. This is normal WordPress behavior when certain code changes happen in blocks and clicking this attempt block recovery button takes care of things nearly every time We.

Now, you'll notice right away that we've changed a few things here. First, we now have a block editor inside of our block. I didn't put any restrictions on this, so it'll take just about any kind of block we throw at it.

Coming back to the block properties, you'll see that we now have this anchor text field so that we can change the link. When we click update and view the page, we can see that all those things have taken. . All right, so this was super simple, but I wanted to run you through it to give you some context before we get too far into the weeds.

So with that outta the way, let's get on to what we're going to cover in the rest of our time. I've already given you an intro and a quick demo, so now we'll get into the meat of our topic and understand that it's important to know where I'm coming from philosophically. So that's what we'll tackle next.

Then we'll talk about blocks because there's a lot to unpack here. Next, we'll take a look at some of the custom block solutions on the market today. Some of them you're probably already familiar with. I'll explain why I have standardized on Pine Grove for my agency. I have a few more demos up my sleeve.

Then we'll wrap it up by telling you where you can go from here. So let's talk about what my agency delivers to our clients. The biggest thing here is to understand that we deliver websites that are meant to be supported for a minimum of three to five. Because of that, the tools that I choose from my client are a lot different from the ones that I would choose from my own personal projects or just for playing around.

Most of my clients are medium and large companies that have internal or external marketing teams with several editors on website every day, making changes to the content. They usually have a style guide in place and the content editors aren't touching the page layouts or coming up with creative looks for their pages and posts.

In fact, the general layout and style of the site may not change at all in its lifetime. Aside from some small tweaks here and there, this might be a lot different than what you're used to, especially with smaller and newer companies. But it's the reason why we do things the way that we do them. It's also not uncommon for a lot of companies to build a website then not touch it or even look at it much at all.

But again, that's not the case with my clients. They use the websites every day and they see it as a key business asset. In fact, it's not uncommon for some of the sites that we build and manage to have hundreds of page views per minute and several million users per month. Don't get me wrong though. I'm not saying that the sites that I build and manage are high traffic or mission critical.

In those cases, completely custom solutions probably would be more appropriate, but they also aren't tiny sites that would be tolerant of extended downtime or frequent bugs either. These reasons drive the decisions that we make about the tools that we use and how we architect our solutions. Some of the things we look at include stability, maintainability, flexibility, dependencies, product longevity and support, and how all the products adhere to standard.

Stability is pretty self-explanatory. We want to make sure that the websites don't break when they update, when WordPress updates or when another product gets updated. We also need to ensure that the product doesn't slow down the website either by generating too much code, running messy queries, or loading unnecessary junk.

When a side note, it's worth mentioning that blocks and page builders do typically increase the load on your server compared to PHP files. It's not terribly noticeable for things like blog posts or lightly formatted pages, but when you start getting into block themes and complex layouts on sites with heavy traffic, that's when you'll probably notice a difference.

Maintainability is the next thing we look at. The tools we choose need to be well documented or at least standard enough that a reasonably capable developer can figure. These tools get used by a host of people from website administrators to support staff and content editors. If the tool isn't usable by any of these stakeholders, we have a problem.

Next is flexibility. No tool is going to accommodate every use case, but we need to select the ones that at least follow the 80 20 rule. Workarounds and limitations are a fact of life, so we just need to make sure that we select our tools with a good understanding of our capabilities and their limitations.

Dependencies are a huge challenge with any development project, and WordPress projects are no except. for each dependency you add, you run the risk of impacting your stability and maintainability. Not only do you have to worry about your tool breaking, but now you also have to worry about how it interacts with all of its dependencies.

We see this all the time with Elementor and its add-ons. Elementor updates something and breaks one of the add-ons. An add-on gets abandoned and suddenly you have to rebuild half your project to fix an unpatched issue with the abandoned plugin. These are all very real and very problematic problems, so when we do introduce a dependency, we have to be extra cautious about making sure that each of those products meets our criteria.

Since we need to support our projects for three to five years, we also need to be sure that the tools we use will be around and well supported for that same. Because of this, we stay away from the new shiny tools on the market and save those for our personal projects and demos. Instead, we rely on time tested solutions from companies that have a proven track record.

And lastly, we look at how well the products we choose adhere to standards. As much as possible, they should use standard processes, APIs, and they shouldn't try to circumvent the way that their dependent platforms work. Let me tell you a little cautionary. In August of 2019, oxygen released version 3.0, and we started using it for our client projects.

There were some red flags, but I was drawn in by the power and the flexibility that it offered, and I chose to move forward anyway. In terms of stability, things were looking pretty good. Version 2.0 had a plan breaking change that was supposed to future proof it. Thankfully, that's been mostly okay ever since the same goes for maintainability.

Documentation and community support have been great and I had no problem getting my teams to support projects built. The first big red flag though was that it purposely broke WordPresses theme system, which also caused issues with other plug-ins and tools that relied on it. Even doing something as simple as adding a function to a child theme was impossible on oxygen without resorting to another third party.

Add-on to managed scripts. Flexibility was a mixed bag in terms of what it let us build and deliver. The product was insanely flexible. However, there were so many usability problems that it almost required third party add-ons to. The same went for components. While oxygen had tons of flexibility for raw HTML elements, it lacked more advanced components like tabs, accordions, sliders, and other interactive components.

And it didn't have a built-in way to create them yourself without coding your own JavaScript. Between the UI, issues with oxygen, the add-on for components and the need for script managers, we ended up installing about five to seven oxygen related plugins, all from different vendors. On every oxygen project we.

That led to dependency problems that really started to rear it's ugly head in 2021 and 2022. By that time, the oxygen ecosystem was no longer the new hotness and many of the plug-in vendors stopped regularly updating their add-ons. More and more I saw the tools that we used on projects being abandoned or causing problems with oxygen updates.

By mid 2022, we had a rewritten, at least some major part of every oxygen project we had. Then of course came the break dance fiasco where oxygen's parent company announced that they had secretly been working on a new page builder project for the last three years. While they insisted that oxygen was going to remain supported, the writing was on the wall, and both third party vendors and a large part of their community quickly jump shipped to new tools.

For my part, we had already been exploring alternatives to oxygen when all those went down. Smaller sites were being built with cadence and generate blocks and more complex sites were being built with pine gross teams and blocks. I also kept a close eye on bricks, since that's what all the WordPress communities seemed to be gravitating to.

But once I started looking at it critically, I realized that it was nowhere near ready for the types of projects that I work on or. Let's crack open your history books for a minute and talk about the history of WordPress page building. In the early days, we either had to choose a theme that had the look and feel we wanted, or we had to code it by hand with php.

That opened the door for early page builders like WP Bakery and a visual composer to let non coders have more design control. Before long page builders started to evolve and add the theme building capabilities that we see today in tools like Elementor, beaver Builder, oxygen, and. While all this was happening, WordPress was throwing its resources behind Gutenberg and the block editor.

So now WordPress users are left with a split On one hand, you have some incredibly powerful and easy to use page builders that sit on top of WordPress and replace or supplement their editor. On the other hand, you have a ton of new block plugins, and some of these block plugins are so powerful that they're almost like coding a site with regular HTML elements in CSS only with the visual controls instead of a text.

and now WordPress is pushing the concept of blocks everywhere and block themes. A few years ago, I was in the blocks or garbage camp, but as they've evolved, so is my thinking, don't get me wrong, I still think that most of the built-in WordPress blocks are very limited. They're best used for editors in blog posts where content is what matters the most.

So what do we do when we have a need that the build in WordPress blocks can't handle? Well, this leads us to the age old build versus by dilemma. And here we have some. We can find a suitable block plugin, but that introduces bloat dependencies and stability concerns. We could use a general purpose block tool like generate blocks, cadence, or green shift.

Those are all grade tools, but again, they introduce dependencies and block editor complexity. If you don't mind getting your hands dirty with a little code, the options open up a little bit. Here we have tools like a CF, Metabo and others that help you create your own blocks with basic html, C Ss, and a bit of markup.

These solve a lot of problems, but again, we have the issue of dependencies plus the need to code the HTML and manage your c s s separately. Finally, you could build your own native blocks by hand using React. This is definitely the cleanest solution, but it's also the most complex, and that is where Pinero comes into play.

It lets you build those native blocks using a visual interface without the need to learn, react, or even touch the HTML and c s s if you don't want to. Yes, it has a learning curve, and yes, it has a complicated interface, but that said, it's much faster and easier than coding blocks by hand, and it gives you most of the same benefit.

For anyone not familiar with Pinero, they've been around for a long time. The product first shipped in 2014 and they've been steadily growing ever since. The reason most WordPress users haven't heard of it is because until now it's been a standalone desktop application rather than a traditional WordPress plugin when the desktop, it's an incredibly powerful platform agnostic HTML and CSS editor.

It also happens to have the ability to create WordPress blocks and themes among other. The WordPress plugin is nearly identical except for the pieces that don't make sense as a server-based application. Things like integration with your desktop code editor, all this is possible because the application is coded in Node J and it runs as a native desktop application using NW js.

What that means for US users though, is that it's completely cross-platform, and that fixes and features on one platform naturally become available every. So that begs the question of how it stacks up against my product selection criteria. In terms of stability, it's rock solid as we've seen. There's no need for the builder to be loaded on your website to use the blocks.

And if you choose to use the desktop version, there's no need to load anything extra and WordPress at all. It uses standard WordPress functions and follows their developing guidelines. What that means is that your blocks will keep working as long as WordPress doesn't break their own backward compat.

And if you know anything about WordPress, you know how well they support backward compatibility well beyond what most other products do. Now, let's take a look at maintainability. There's a ton of documentation on the Pine Gro website, and because it's all using standard WordPress functionality, almost anything can be solved with a Google search when it comes to training developers and support teams.

This is where it falls down a bit. It is a complicated product, and it's not ubiquitous enough for most people to already know how to use it. So yeah, you'll need to invest a bit in training. Pinero's flexibility is amazing. You can use just about any CSS or JavaScript framework you can think of, and you can build almost anything you need.

The best part is that it's all HTML and css, so Code Nimitz Designs and other things can just be copied and pasted right into your project. The biggest limitation I've found, though is with the block controls, they're pretty basic, and Pinero doesn't offer a built-in way to stylize them, customize them, or add new.

Dependencies are a non-issue with Pine Gro. The only dependencies you need to worry about are the ones that you choose to introduce yourself. Product longevity and support is also not a major concern. They've been around since 2014 and although it's a small team, they've been exceptionally responsive and support has been great here.

My biggest concern is that it doesn't have as big a community as other products and community support is pretty limited to their forum. Lastly, sticking with standards is one of their core principles. In some cases, this is even to their detriment, where they insist on not building features and helpers that don't map directly to established APIs, functions and standards.

That's a lot of jabbering from me, so let's jump back into WordPress and build a few more blocks. This time, we're going to grab a project from Code Pen and turn it into some WordPress blocks. As you can see here, we have a pretty basic layout that gives us the opportunity to create a variety of blocks without getting too complic.

Over in Pinero, we'll create a new blank project with plain H T M L and c s s. I'll give this a name and call it Company X, and we'll activate that WordPress module with the default settings.

Now we're ready to rock and roll. Looking at Code Pen, I can see that the HTML was written in Pug and the CSS in SaaS. If I were using Pinero desktop, I could take these as is and bring them in, but the WordPress plugin doesn't support those yet. So I need to switch to compiled H T M L in and c s s in Code Pen.

I'm gonna grab this entire HTML section and I can put it in Pinero in a few different ways. For simplicity's sake, I'm just going to use this insert menu, paste my code in this little blocks. I'll grab this label and drag it into my H T M L tree. Next, we'll come back to Code Pen and I'll copy the c s s in Pinero.

I'll open our project panel so we can see all the files in our project. We'll expand css, right click on style CSS, opening code editor, and just paste that right in there. Now I'll close the code editor window to give us some more room for the preview. If you're wondering what all these lines are, they're just an overlay showing us the CSS grid.

I can toggle them on and off using the Visual helpers menu icon, but they're showing right now because I have the grid wrapper selected in the tree. If I come back to the tree view and select a different element, you can see that they're no longer in the way. Cool. So now we have this layout and we're gonna get to work making blocks out of it.

Let's expand this so we can see what we have and then collapse some of the subsection. The first thing that I want to do is to create a block that's gonna hold our c s s grid layout wrapper. Since I know that WordPress blocks don't have what we need to do that, I'm gonna add another div to our layout, and I'm gonna put this whole thing inside of it, and this is where I'm gonna create our block.

So I'll come over here to the WordPress actions and click on block. I'll give this an ID and a name, so I will call this layout wrapper and we'll call this. Company X layout for category. I'm gonna come down and say custom and I wanna give this a custom category of Company X. Since we don't have this category created yet.

I'm gonna say register category and we will say Company X is called Company X and I'm not gonna bother with an icon rather than using the default icon. For this block though, I'm gonna head over to Remix Icon. and I will find one that seems appropriate, like layout grid. I'm gonna copy this SVG code, come back here to our icon in the block and paste that SVG code right in there.

Now that we have our block created, I'm gonna take this diviv and I'm gonna turn it into the inner content block. So this is where we're going to be able to add all these other blocks that we create. Let's go ahead and do that right now. So I will create a block for each one of these. In order for our category ID to show up, I just need to export this once and now you see it's right there.

We'll save the project and export it one more time. Now we'll come over to our plugin screen. We'll activate that plugin. We'll go to our homepage, edit the page, and instead of showing the regular template, I want to change this template over to, uh, blank, and that's gonna allow it to go full screen. Let's just update this and save it so we can see what we're seeing in there.

You can see now that it's going side to side full screen. We'll edit this again to delete that about meat block. And now I'm gonna add the blocks that we just created. So all the way down here at the bottom, you can see that we have our company X section and our company X layout block. And it also has the icon that we already assigned to it.

So let's click on that to add it, and then put the footer in. Just remove that, get rid of our paragraph. And now inside of company X layout, I'm gonna add all the other ones. So let's do these one by one. Masthead hero features, call out blog, and. . You can see all them right in here. I'm just gonna drag them inside.

And there we have it. Now you'll see here that none of these are styled yet, and that's because we haven't told WordPress to use our style sheet. So we'll save this. View the page so we can see where we are. And now let's go back to Pinero.

One of the things that WordPress does with blocks is each block can hold its own style sheet. By default. WordPress uses its own theme styles to show everything, and each block is gonna hold its own styles, its own JavaScript, all those things that only get loaded when that block is on the page. That's great for reducing bloat, but it's kind of a pain in the butt when we have a ton of blocks that we're creating.

So how are we gonna fix this? Well, let's come right back here. And since we already know that this block is going to hold everything else, I can just add the style sheet to this one block and it's gonna load on the page. That means that all the other blocks are gonna be able to pick up on it and see it.

So to do that, we just come back to our block and we say more options. And under style I say CSS style, css. And what I'm writing there is just the. To the CSS file. We'll save this, export it. We'll come back to the front end. And now you can see that we have our styles showing. You'll notice here that these links are green and that the styles in here look a little bit different than they do inside the Pinero Builder, and that's because WordPress is layering on its own styles on top of these blocks.

Whereas here in the Pinero editor, we're looking at just the things that we've applied, not the things that WordPress is adding to it. So let's go ahead and right click on one of these and inspect so you can see what it is that's happening. And you can see on this anchor link, our color white that was assigned in the style sheet is being overridden by the WP Block style.

So in order for our white to show through, we're gonna need to make that just a little bit more specific. So we'll do that as we go through the project. We'll do an edit page to take a look at what we. And for the most part, things look okay. It's also important to notice that inside WordPress, not only does the front end have its own styles, but the block editor also has another style sheet.

So it just gets more and more confusing as things go along. But again, these are WordPress issues and just things that you'll come to learn how to deal with. down here though, we've got some interesting stuff happening that shouldn't be happening, and you'll see that our layout at the top mostly looks okay, but we're gonna come down here and our blog layout looks funny and our footer layout looks funny, looks nothing like it does here in the builder.

And the reason for that is let's come over to the tree view and take a look. When we click on the blog section and look at our styles, we're gonna see, we have this rule here that applies the grid column one, negative one, two masthead hero features call out and the blog ID and the footer id. Well, those IDs don't exactly work the way that we expect them to inside the block editor.

So rather than have them as IDs, we're going to add classes to. So I'll just do that right here inside of our editor, and I'll say, this is going to be a blog and we're gonna add that, you know what, I'm not even gonna create a rule for it. I'm just gonna come over here and add a class a blog, and I'll do the same thing with footer there.

Now I'll come over to our styles and we will edit this and I'm just going to change it to add blog and. . So it all still works the same. We've just modified the rules a little bit. We're gonna export that project Refresh, block recovery. Block recovery. And now you see that things are stacked up the way that they're supposed to.

That doesn't do anything to fix the editor with, but that's a whole different story. There are some code snippets that you can use to make the editor with inside the block editor a little bit more realistic. By default, it's something like 600 pixels, which is great for writing. Not so good for laying out a page, but let's update view page and everything looks good.

Now we're gonna take these blocks and we're going to start customizing them. We're gonna start here on the mast head, and the first thing that I wanna do, Is take this company X and I wanna change this to the site name. So I'm just gonna look for the WordPress action of site name, and now it's gonna replace company X with the name of the website.

I don't want this to be an H one, so I'll just right click on that and say Transform, and we're gonna change that over to a span. Next thing I'll do is I'll select our menu and I've got the unordered list, which is at the root of the menu, and I'm gonna add a menu action to it. And we're gonna give this the location of primary, and I want to register this menu as primary menu.

All right, there we go. Now what this is gonna do, it's actually going to create a, uh, custom walker nav that goes through and looks at the list items and the anchors and the structure that we have in here, and it's going to build a WordPress menu based on the structure of the first list item. All right, so that's enough for the masthead.

Next thing that I wanna do is start working on the hero. And down here in this hero section, there's two things. First thing I wanna do is to be able to change this text, and then I wanna change the background image. So we'll start with the text and I'll just right click on this, first of all, and change this over to an H one.

And you can see that the H one doesn't have any styling to go along with it. So here I'm going to want to, uh, create a rule. So I'll come over here to our Styles panel, and I'm going to say that I want, um, everything that's in our hero intersection, that's in H one. I'm gonna create a style for it. In the style CSS file, I'm gonna select this and I'm just going to make that white.

So now we're gonna look and we can see that it's written the CSS for us. If we want to look at the whole style CSS file, we've got it right in here and down there at the bottom you'll see that it added our rule for us right there. Now, hopefully this will have enough specificity that's going to overrule anything that's inside a word.

Um, so the next thing that I want to do is I want to, uh, change the background image. And to do that, I need to find out where that background image lives and it lives right here on the section. So under the section, I'll come back to our WordPress actions and I'm gonna add a block attribute. And for the block attribute, I'm gonna say background image.

And we're going to say image size should be. and that we're going to use it as the background image. All right, now let's save this, export the plugin. We'll come over to our page and edit the page. And you can see that we need to attempt block recovery. So we'll do that there and I'll do it over here.

We'll update few page, and there we have it. You can see now that we have a menu. When I go over to menus, we've got a menu that's been created over here called Menu One, and I need to actually create this new menu. I'll create this as my menu. I'll assign it, the primary menu, say Create. We'll add some pages to it, home, blog about contact, and what we do.

Here we go. Save that. And there we have them right there at the uh, menu location. We'll come back here to edit the page. And when we click on our block, we now have the ability to change the background image. So let's select that and click on our coffee beans. Here we go. Update and we're good there. Uh, however, I don't have a way to change the text probably because I forgot to add a block attribute there.

So yeah, I changed it to an H one, but I never actually added the block attribute. So I'll do that here inside of the WordPress. Block attribute title. Call it hero title, export the plugin. Refresh in there. Now I can change this. Update a page, view our page, and there we have it. Now what are we gonna do about these links?

Let's go ahead and quickly fix that and to fix that, we'll come over to our Pine Gro. I'm gonna click on one of the links and we're gonna create a more specific c s s rule. So we'll do that by coming over here to our create new c s s rule builder. And this time I'm gonna say Masthead menu ally A create this rule in the style Out CSS file.

We'll select that new rule and give it a color of white save. Export. Come back to our builder. Or our page refresh. And there we have it. Now our menu items are white the way that we want them. Cool. Next, we're going to work on this feature section, and we're gonna have a little bit of fun with this one.

And we could do nested box again, but we've already done that once in the demo, so we're gonna change it up a bit. This time we're gonna do custom post types, so here we're gonna expand. , and we already have this nice little thing here where we've got our wrapper and three articles here on the wrapper.

I'm gonna come over to our WordPress actions and I'm gonna create a show post loop, and this is gonna create a loop, a loop on the div, and I'm gonna set, our repeated item is going to be this article. I'm gonna select these other two, and I'm gonna say, do not export those to WordPress. So now we're just gonna see them in the, in the builder, but it's not actually gonna do anything with them.

Not gonna create blocks with them, not going to add them to our block. Back in our loop, we're gonna change this loop type to custom query. And under post type, you can see that I have a few post types already available to loop through, but I don't have a custom one. So now let's do. We will clear out our search here and we're gonna do register, and this time I want to add a register post type action, and I'm going to call this company X Features.

Company X features and Company X feature. If you've ever created a post type before, you'll know that you want to give it a plural name as well as a singular name. And that's exactly what I'm doing here. I wanna create this as public, and I wanna say that it's going to support a title. Uh, I wanted to support the editor and I wanted to support, uh, thumbnail.

There we go. Um, show ui. Yes. So I do want this to show. And for menu icons, I want this to be dash icons. Coffee menu positions going to be 20, and I think that's all I really need. Awesome. So let's save this, export it just so that it saves all this stuff. And now I can come back into the loop properties and under post type I've got my company X features post type that I can loop.

The other thing that I wanna do here is I wanna do order by and change this created type up to create a type down. And I only want to show three posts in here. So let's come in and range of items to show is gonna be zero to two. Okay? Now that we have the loop built, we need to tell it what we're going to display in there.

So let's come in here and I'm gonna take my image. And for the image I want to display my featured image. And I'm gonna do that as a medium size featured image. And for my P I'm gonna make this the post content. Alright, save export, and we're gonna go to the dashboard and see what we have there. Awesome.

Now we've got this called Company X features. We've got a new post type. Let's create a few posts in there that we can show on the. Tastes great. Yu Coffee featured image is going to be that one. We'll add another one, less filling. Give it an image. Let's use that one. Why not publish

And great way to start the day. And publish. Awesome. Now let's take a look at our site on the front end. And there we have it. We've got our featured image for all three of those and the content for each of them in there. Okay. Now we've already figured out how to do a block action to change some text.

We've already done a block action to change the background, so we're not gonna mess with that anymore. Now we're gonna come down and we will do a, uh, lube action for the blog. . So let's open up Pinero again, collapse that section and open up our blog. And let's see how this is structured right now. So we've got an inter wrapper, we've got a grid, and this is kind of interesting.

It looks fine in the builder, um, but it's not really structured the way that we want to for a loop. See, what we really want is we want. Single item that we can just loop through over and over again. So we're going to take these guys out, just remove them, and we're gonna create our loop right here. So I'm gonna take this item and I'm gonna do show posts and we'll change this from main loop to custom query.

Post type is going to be post and order by create a date down. Post per page here. I just wanna show five. Uh, I'm not gonna bother paginating results and I think I'm okay with these loop properties the way they are. Uh, cuz by default it's going to take our items container and loop through this whole thing.

But as we're gonna see in a minute, this isn't gonna look very good. Um, but let's go ahead and just attach some things to it anyway so that we can see what it looks like on the front. . So we'll come down to the image and I'm gonna make this our, uh, post featured image down to our text. And here we've got our summary and I'm gonna make that our excerpt, this anchor or read more button.

I'm gonna make this the post link and I also want to add a post title in there. So I will come back up here and we'll do insert H two and I will put post title on. . Okay, now let's save this. Export the plugin, come back to the front end and refresh. And you can see that it's grabbed everything, but the formatting is just abysmal.

Um, so a few things going on here. We definitely wanna change the way these buttons look. We want these H two s to be much smaller than that. Um, and then we want something that's going to add a little bit of separation between each of these posts. Let's take care of it. . So we'll come in here and we've got our H two and this is something that we want to tackle.

So we can come in and just create a rule to take care of that and we'll say, uh, this is going to be our.blog. H two, create that rule. And we want this size to just be two res. . That's fine. Okay, so there's that. Next thing is we're gonna take care of this, read more text that's green right now because of our WordPress styles.

And we're just going to add some specificity there. So I'm gonna say our blog, uh, anchor button is going to be color of white, and that's not exactly what I want cause it's also affecting this. So let's delete. And instead we're going to call this our, um, blog grid anchor button, color white. All right.

That takes care of that. And let's add a little bit of a box shadow around here. So we'll come over to shadow. We'll add a shadow, do one pixel, uh, zero, one pixel. Uh, blur is gonna be three pixels, size with zero and color. I'm gonna make this black. 10% choose and all right, that gives us a little bit of something and I'm okay with that.

All right, save it. Export and yeah, there we go. Now we're looking a little bit better. These things are butting up right against each other, so we're gonna need to add a tiny bit of space in there. Um, so this isn't really the way that I want to do it. Um, but for our purposes today, I'm just going to do it by adding a bottom margin of two rems.

And you know what, while we're here, let's take care of a few, few other things as well. Um, so for this, we're going to make the image, uh, take up the whole space. So let's take our image wrapper and create a rule for it. Blog grid, um, blog image. And we're going to make that a display of flex. And you can see that it's already stretched that out.

Uh, we're gonna take our image, blog image img, and for that we're going to tell it that we want object fit of cover. And our object position is going to be 50% and 50%, which is going to center it horizontally, vertically. Um, and the last thing we want to do is find our more stories button and come over to our properties and just change the link to that, to point to the blog.

Save, export, refresh. And there we go. Now it's looking a little bit more like a blog list. We can click on any of these and it's going to go over to our posts. Looks good.

And there are more stories. Takes us to our blog archive page. All right, so I'm not gonna bore you by doing any more stuff. Um, everything else we've pretty much covered. Uh, I could add more and more menus down here in the footer if I wanted to. I can add all sorts of other elements and more actions to it.

But for our purposes, I think we've covered enough already. We've done a couple loops, uh, we've changed some background images, we've changed lots of properties. We've done menus. And, um, what else have we done? Oh, yes, we created a custom post type, uh, inside of Pinero. So that was an awful lot inside of one demo.

I hope you got a lot out of this if you've made it this far. I appreciate your attention and dedication, and if you want to learn more, I've got a few resources for you. The easiest thing to do is to just go to my personalPage@adamlowe.io. There you'll find links to all these resources. And then my YouTube channel that has a bunch more videos about WordPress blocks, themes, and pinero inform.

I also recommend checking out full site editing.com. They have lots of great information about blocks, block themes, and how to use theme dot Jason to manage your WordPress styles. The Kinta blog always surprises me with their quality content. This article building Custom Gutenberg Blocks is particularly good.

And lastly, if you wanna check out Pine Gro, just go to pine gro.com. Thank you again for joining me today, and I'd love to connect with each of you on social media. Just go to adam lowe.io and you'll find all of my inform.