jQuery and Divi in Practice

Hello, and welcome to the final lesson in this module, where I would like to give you a few practical tips on working with jQuery in Divi. First, we'll do a short reminder on where we can add the jQuery code in Divi, and then I will share a few useful examples and things to consider when implementing custom JavaScript in your Divi websites.

Adding JS to Divi Theme Options

So first, let me show you three places to add your jQuery code in Divi and explain how they are different. First option: Divi Theme Options, Integration tab. We can include a script tag and insert custom code right here inline. Or, we can link to an external file. Let me give you an example. That is how we could import an external JavaScript file using a script tag.

And we can also write JavaScript inline like this. So here we have opening script tag and closing script tag, and inside our jQuery document.ready function with a little alert. So if I save that now, we can see browser gives us that alert that JavaScript is working. Now the integration tab gives us a few different places to include the code: inside the head or the body, which would run globally on every page of our website.

Or we can use this section to include it only for blog posts.

Adding JS inside a Code module

The second place we can include JavaScript is a Code module.

The syntax we would use here is the same: a script tag with an external link or

most likely) an inline jQuery code. The difference between using integration tab and a code module is that the code from the module will only run in places where that code module is added. If we add a module on a page, the code will run on that specific page only. If we include a code module in one of our Theme Builder templates, the code will run only in places where that template is currently assigned to. So these are options Divi provides us.

Adding JS inside a child theme

And another way to add jQuery, I want to show you, which is not specific to the Divi theme, is of course adding the code through your child theme.

We can enqueue you any number of JavaScript files inside our child theme. We only need to call the wp_enquque_script function here. We would define the script name, the URL, the path that holds the file. So in our case that's the 'js' folder with a scripts.js file inside our theme directory. We can add jQuery dependency, meaning that this will always be loaded after the main jQuery file. We can include a version number. And that last parameter is here to... we can choose if we would want to load that script in the footer or not. So true means it will load in the footer. WordPress codex is the best place to learn exactly how to use WordPress functions. But I did provide a sample child theme that includes a JavaScript file, which you can download and use as a starting point.

jQuery(document).ready Syntax

Now here, inside the file, we do not need the script tag anymore. We can simply start writing our code without that opening and closing script tags. Now you might have noticed that I haven't used the shorthand version of the document ready function.

The one I used when working with sample HTML structure in previous lesson, I am using a full jQuery document ready function, and then dollar sign, because

that is needed, because of the way WordPress loads the main jQuery file. So that would be my first tip: always use the full document. ready syntax to make sure your code works.

Modifying Text strings

Now let's explore a few code examples and ideas on how to use jQuery and Divi. First, modifying text strings. It could be something generated by WordPress, Divi theme or a plugin, a text string, not editable in the dashboard. Like the "Older Entries" link here on the blog page. And yes, I know that we could use a translation plugin to edit these various text strings this way.

And that would be probably a recommended approach, especially for multi-lingual websites, but for simple sites using one language, we can use jQuery to quickly change some words on the front end. So to be able to change the word behind that link, we have to be able to target that the same way as we would target that with CSS.

So that link doesn't have any special class, but it is inside a div with a class "pagination". And that Older entries would have a parent of "alignleft" and Newer entries, I believe has a parent of "alignright". So we can target that link by listing the parent container. So here, dollar sign and then our selector, the same as in CSS.

So class pagination, alignleft, a link, and we can use text method. And instead of Older entries, let's say "Previous articles". Now, let's save our JavaScript file. And now we see that previous articles text right here. Now let's take care of the next entries link. So that will be the same, but with "alignright" class. "Next articles". Lovely. So now we have previous and next articles links instead of the standard, older, newer entries.

Using JS with Ajax-generated content

But, what is working fine on our default archive pages will not be as easy to achieve on a page where the blog module is used. Let me show you. Here, for my category page I am using a Theme Builder template with a Blog module and that JavaScript code worked for that previous articles link.

But once we go to the second page, it stops working. And the reason for that is that the Divi is using Ajax to load the second page of our blog list. So to fix that, we have to make sure that this code is loaded not only once the document is ready on the page load, but also when Ajax function is completed.

And luckily there is Ajax.complete method in jQuery we can use, but first we would want to include these two lines of code inside our own function. So we can simply define function. And then any name, for example, renameLinks. And here, inside curly brackets, we define our function.

So when the code is inside a function, it doesn't do anything yet. It's just setting the function. So now to make sure that the function is actually run by the browser, you just have to call it like this. And next, we can call it again when the Ajax complete function is completed. So it looks like this and here, our renameLinks. Now, if we check that category page with a Blog module on the second page of our links, that function is run again. So that would be useful thing to remember when working with Ajax generated content. Especially on the Blog module.

Simple jQuery Read more / Read less

Now, another example I would like to show you is a simple show and hide action. So I have a text module and a button. When I click the button, it shows me another text module. And then I can click again to hide it. So the simplest approach we could take to recreate what's happening here is with that type of

structure where we have that first text module, the second additional content, and I've added a CSS class of dsa-more-content and two buttons.

One with dsa-read-more. And the next with the dsa-read-less CSS class. So, what needs to happen at the beginning is these two elements that additional content and read less button needs to be hidden. So we can do that easily with the hide() method.

We can specify more than one selector in jQuery, the same as we would in CSS: with coma. Okay. Let's save that. Let's check my page. So that's a good start, but now nothing happens when I click the button. Now with jQuery, we can attach our own click event to any element. So I've added a CSS class of dsa-read-more to the button. Dot here for a class, the same as in CSS and on click, we will perform a function.

And here inside curly brackets, we can declare what we want to happen when that element is clicked. So we want to show the dsa-more-content and we can use the slideDown() method for the animation.

Okay. Let's see how that looks. It is working, but then there is that instant page refresh because I haven't added any URL to my read more. If I used a hash symbol, I would get that scroll to top effect, which is also unwanted. So if we are working with the click event on a button, we have to use that "preventDefault" method to make sure that the default action (so going to a specific URL or that scroll to top with anchor link) doesn't happen.

Button Click event with preventDefault

So here, to that click function, we will add an event name and preventDefault like this. Now it's working. There's no default action attached to that button. Only the thing that I want to happen. Okay. So that is a good start. Now, other than showing that more content, I also want to show that dsa-read-less button,

but maybe without the slide down, just with simple show.

And I also want to hide the read more. So instead of defining the class again, I can use "this" selector because I'm already working with that element: $(this).hide();.

Okay. The read less is visible, read more is hidden. And now we could attach a similar function click function to that read less button. It actually worked now because of the page refresh again. So, but we do not want the page to refresh. We want our own function. I will copy that. What I have for the read more and for the read less, we would slide up our content, a dsa-read-more would be visible. And this, the "read less" would get hidden. So super simple approach.

Targeting multiple elements with the same function

Now, the problem we may encounter is when we decide to use that on more than one element. So let me show you if we would duplicate that in multiple places.

This is not the action we would want to see. Right. So for that, we have to make sure that what we are defining here, these selectors, what's happening on click, are relative to the thing that we are clicking. Right? So instead of targeting all elements with the dsa-more-content, we have to be more specific and see how that "more-content", that particular one we want to hide, is, what's its relation to the thing that we are clicking. So here, if we inspect our page, the read more link is a button module. But it's actually inside that et_pb_button_module _wrapper, et_pb_module DIV. We would have to first look at the parent and then find the parent sibling, right on the same level.

We have that, et_pb_text module with dsa-more-content class. So it's a sibling of a parent container. Instead of that simple class. First, we are looking at this then parent siblings. And we want to find that class within the siblings of the

parent container. And then we use the slide down method.

So the same thing would need to happen here. And now what's the relation between read more and read less. It's actually a bit more complicated than just a sibling because the button module, wrappers, the container, the parent DIVs are siblings. But the thing that we are clicking links are inside. So the relation between the read more and read less is that we have to look at the parent, next, the sibling with the et_pb_button_module_wrapper class, and then find children, a child element with dsa-read-more.

So here again, this parent siblings, but not any sibling, sibling with a class et_pb_button_module_wrapper.

And that we can find inside or use children, but find would be fine. So find method looks inside the container and we ask it to find that dsa-read-less, a CSS class inside that wrapper, which is a sibling of a parent. So that's the relation between that read less and read more and it would be the same other way around.

But here, we want to find the dsa-read-more CSS class to show that, okay. Let's now test that .You see now they work independently because we are not just targeting all the CSS classes, the same classes, but the ones that are in the correct relation to the thing that we are clicking. And that whole show and hide is based solely on jQuery.

Show/Hide based on CSS class toggle

We haven't add any CSS to make it work here, but I wanted to show you a different approach we could take. I've built that footer template in the Theme builder, and I created a fixed section. So that section: position fixed with, I put that in the middle of that page. I made sure that it's 300 pixels wide. So in the sizing section, I changed the width to 300 pixels. I made sure that my row inside

is full width, I've added some padding and here in my Row, I have a text module, a button module, and also an icon module right here. So if we look at the, I have a text, icon and a button and the icon module is positioned absolutely. So position absolute at the top right corner with some offset to make sure that it is outside of the section here. These are the settings that I've used and now I've added two CSS classes to the icon. I've added dsa-side-toggle, and also to the parent section, I named it dsa-side.

What I would like to do is simply hide that. And when I click the icon, that section would slide right, and then if I click again, it would slide back behind the viewport. Right? So it is quite simple to achieve with jQuery. We would only need to toggle CSS classes and the rest can be done with CSS alone.

So for that, I would also use a click function and I will use my dsa-side-toggle CSS class. And on click, we don't need that event because the icon module is not a button. So there is no default action that would have happened on click. So I do not need that additional event with preventDefault method. On click, I want this to get a new CSS class. I will use toggleClass() method to toggle "opened" class. Meaning: the first time I click - that class will be added, and then when I click again, that class will be removed. And what I would also want is to add that opened class to my dsa-side, to the section itself like this. Okay. Now, if I update that and preview that page. We can see here on the site that that opened class is added to the icon module and the section as well. When I click it gets highlighted that there is a change here. So the class gets added and hidden. Now we can use these CSS classes to make sure that our section behaves in a way we want, whether it's opened or not.

By default, it would be closed. Meaning my dsa-side, the container, the section would have position left negative 300 pixels because I want to move it outside by its width. So left minus 300 pixels. Now it's right there behind, but once it has the "opened" class notice, there's no space here. That's targeting one

element with two CSS classes, position left would be zero.

So now that is working, there is no smooth transition here. We can fix that by adding a transition to that dsa-side selector as well. So here transition all 0.3 seconds ease-in-out. You can use any type of transition you like, but now that would be animated. And another thing we could do is to... well, we could change the icon when it's opened, but we can also rotate it since it's an arrow when it's open, I would rather have these arrows pointing in the other direction.

Right. So we can again, use that 'opened' class on our dsa-side-toggle. So .dsa-side-toggle.opened so when it is opened, we want to target the icon inside, which is a span with a class et-pb-icon.

And here we can use transform property. And the rotate. Rotated by a 180 degrees. Let's see, we could also add a transition, but it's not as important here for, for the arrow. So simple as that. Let's see, let's see how we've added three CSS declarations and one line of JavaScript to create that you know, fixed, hidden section. And using that approach, we could do any type of animation.

We have to think what is the state of the element before it's opened and what is the CSS that affects it when it is visible? That is just based on that simple jQuery function that toggles the CSS.

Modifying HTML structure with each()

And one last example I want to show you involves modifying the HTML structure. So this is a Blog module, and I would like to display that is in that alternating style with image on the side all the blog details on the other side.

And then for the next one other way around. The problem is that the structure of each of the posts on the blog list... we have the content for each post in the article container, but then our image, our featured image is a link and all other elements are also right here as a sibling to that link. So.

What I would like to see here is a container that holds these three together, the title, post meta, post content, and a read more button, if it would be here. So all of these elements, if they would have a parent container, we could place that parent next to the image. And then alternate that for each of the posts we could achieve that with CSS alone, but it would be kind of tricky and it will be just much easier to achieve if we had that parent container here. So let's try adding that with jQuery, I've added an ID of dsa-blog to that Blog module. So we want to tell each of the articles inside, et_pb_post CSS class article inside my dsa-blog.

I have to kind of loop through all of them similar as with the multiple show and hide buttons, but there's no click event here. That's why we can use the each() function. It would look like this: first, our target so ID dsa-blog, and inside we have an article or any container with a class et_pb_post.

And now I want to perform something for each of these. I can do that with each method and then function. So similarly to click, I will be able to use this to find elements within each post and do something with these elements. There is a jQuery methods called wrapAll(), and I want to wrap all the children of each of posts, except for the featured image.

So going back here, the future image has a class of entry featured image URL. So for each of my blog posts, I want to target that particular element children, which are not, which do not have that entry-featured-image-url. So all the children, except for that CSS class we'll wrap them all. So wrapAll, and here in brackets, we would define the container we want to wrap them with.

So that would be a Div with a class of blog-wrapper, for example. Something like that and then closing div. Okay. So that should work. We are for each of et_pb_post's inside the dsa-blog ID, we'll look at each post children and all of the children, which are not this CSS class, we'll wrap them with a DIV.

If I refresh...

Styling a new HTML structure with CSS

We can see that Div with a blog-wrapper here. Next to the image. And since now we only have two containers, we can use Flexbox to align them. So as with Flexbox, we would need to target the parent container. So that would be a #dsa-blog .et_pb_post display flex. So that's a great start right now. Our "blog-wrapper" could use some spacing and maybe a different background. So let's add some padding and background, light, gray, something like that. And now we could change the order of Flex for alternating elements. So again ETP et_pb_post but :nth-child(2n) and here we would change the direction.

So flex direction, row reverse. Okay. And now we can force the image to match the height of our container. Our image is that entry... or we can target the image tag. So again, image inside #dsa-blog .et_pb_post. Height 100% and when forcing the height, we have to make sure that object-fit is changed to cover because we don't want to stretch that item.

And that space in here is the margin on that entry-featured-image-url. Let's add our #dsa-blog parent, and remove that margin bottom. Okay. So this is the CSS I wrote for, for that Blog module. And as you can see, it, wasn't a lot of CSS and now we have that nice looking Blog layout. But the problem, I will copy that, the problem would again, be on that second page. Because here, there is no container, right? Our jQuery only work on the first page load and not for the second. So here, let me just add to that CSS right here. And we would need to do a similar thing with that document Ajax complete function. And here I added a function, called that function and then call that again, inside Ajax complete, but we do not need to do it this way.

We could do like repeat ourselves. And just add all this again, inside here. Okay, so I'm sure you see how it would be problematic for longer functions. That's

why I do recommend this approach with naming your function and then calling it and call it again this way. If you want to edit something, you only edit it in one place.

Whether here, if I want to edit something, I would need to edit that part and that part as well. But it's just to show you that it should work. So that would be our first page. And on the second page, the wrapper div is still here working correctly. Hopefully these few code examples will give you an idea of how you can implement jQuery in your own projects.