

Understanding HTML Structure

Hello, and welcome to the first lesson of the Divi Stylist Academy third module. In this module, we will start to learn how to write our custom CSS code. It is not possible to learn CSS properly without understanding the HTML language structure. And that is what we are going to focus on in this lesson.

HTML - the foundation of every website

So first, it may seem obvious to some of you - but I do need to make it clear - HTML is the foundation of every website you see online. No matter the tool or the way it has been built. So if we are using WordPress and Divi - we are using HTML. But just to take a tiny step back. When we enter the domain name to the browser - the browser connects with the server. And on the server - we have our PHP code, all the files used to build the website - PHP WordPress functions, Divi PHP templates, our database, my SQL database - all that PHP code is handled by our servers. These are the server-side languages. And once the server figures out what we ask it to do with the PHP - it outputs the HTML version of the code and sends it to our browser. Now, the browser is responsible for reading that HTML - and CSS and JavaScript code - the HTML may include - these are client-side languages. In other words - when we are using the builder - we are basically asking Divi (and WordPress) to generate that HTML code for us. Let's take a look at this layout - this would be a simplified HTML version of it. A div with an `et_pb_module` class, which is inside a div with an `et_pb_column`, which

is inside `et_pb_row`, and so on. I know that looking at the code for the first time might be a bit overwhelming, but don't worry, we'll figure it out step by step.

HTML document

We want to understand the HTML version of your page that is rendered by the browser. You don't need to know how to write what I'm showing you, but you will find it helpful if you can see it clearly and understand - what is what? So this is a base structure for any HTML document. HTML uses a sandwich-like structure. The main `<HTML>` tag hold two elements the `<head>` and the `<body>` tag. The `<head>` of the document is where you'll see your website title, everything that needs to be rendered first, so links to scripts, styles, tracking code (like Google Analytics or FB pixel), and also `<meta>` tags, which hold information visible only to search engines, like your website title and description. Now `<body>` tag contains all the content that is displayed on the page, including text, images, videos, or whatever else. HTML5 introduced a lot of new semantic tags, like `<nav>` `<article>` `<header>` `<section>` or a `<footer>` tag, and you might see them when you inspect your Divi website, but what you see the most is probably a non-semantic `<div>` tag.

HTML tag

Looking closer at the HTML tag, the opening tag: This consists of the name of the element (in this example, `p` for paragraph), it's wrapped in opening and closing angle brackets. This opening tag marks where the element begins or starts to take effect. In this example, it precedes the start of the paragraph text. Next, we have the content: this is the content of the element. In this example, it is the paragraph text. And lastly - the closing tag: this is the same as the opening tag, except that it

includes a forward slash before the element name. This marks where the element ends. Failing to include a closing tag is a common beginner error that can produce peculiar results. The element, the HTML element is the opening tag, followed by content, followed by the closing tag.

HTML nesting

And like I said: HTML uses the sandwich-like structure and elements can be placed within other elements. This is called: nesting. If we wanted to state that Divi Theme is very cool, we could wrap the word very in a `` element, which means that the word is to have strong(er) text formatting: `<p>Divi Theme isvery cool</p>` There is a right and wrong way to do nesting. In this example, we opened the `p` element first, then opened the `strong` element. For proper nesting, we should close the `strong` element first, before closing the `p`. This would be an example of the wrong way to do nesting: `<p>Divi Theme isvery cool</p>` The tags have to open and close in a way that they are inside or outside one another. With the kind of overlap in the example here, the browser has to guess what was your intent. This kind of guessing can result in unexpected results.

Now, if we go back to the HTML generated by Divi I think it should look much clearer. We have a `div` with an `et_pb_section`, that is a parent container, a parent `div` of an element `div` with a class `et_pg_row` and inside a row, we have a `div` element for our column with class a `et_pb_column`, there is a `div` module inside and it holds the `<p>` tag with paragraph text. So hopefully that is helpful, in the next lesson we'll look into HTML tags in more detail.