# Post-Wordpress web-dev, FUCK IT.

Psst, it’s me. Your Post-WordPress web developer. Yeah, I’ll tell you a bit more about my story.

*\*(intro) What did the subject look like as a healthy and normal person?\**

It all started the day I created my first social media account. Having an account brought along with it the possibility to fill up my gallery with personal pictures, select and list top friends on one’s profile, and fill in the lists asking about one’s favorite movies, foods, and miscellaneous interests.

Nothing was blank on my Hi5 (Balkan Myspace) profile. I was crafting and re-crafting it daily. My favorite part was customizing the wallpaper through copying codes from other platforms in the platform’s hypertext boxes. I went through many different kinds of wallpapers: from clouds to black and white skate parks during sunset. Once I entered my ‘pitch-black phase’, I focused more on the quality of my lists and made sure they showed some good taste.

Styling one’s profile was the most enjoyable part of my user’s experience. Profiles were exchanged amongst users as exemplary references to look at while building one’s own.

Today, I can’t tell if enabling the users to apply aesthetic changes on social media platforms (which were ultimately designed for user connection) was a deliberate aspect of the platform’s design, but I do believe this was the reason why the switch to the currently most dominant one (Facebook), a platform where aesthetic customization is *disabled*, was so easy.

Between this moment of stylizing my wallpapers in 2009 and my bureaucratic situation in 2019 lies a huge gap in my coding practice. Attending the bootcamp school bridged this gap: I started building websites from scratch, not just inserting codes on someone else’s platform.

The first few small-scale websites I built were handmade—meaning they were built with HTML, CSS, and Javascript; basic tools for building any kind of website. Learning how to build a button, for example, already made me realize that the design of this button can rather easily manipulate the user’s mood, akin to how a doorknob of a random door would. Where does the door lead to? Is it locked? What’s lurking on the other side? Should I *open* the door? Multiple, diverse types of buttons leading to a manifold of pages made me wonder about the amount of space a website occupies within various digital infrastructures.

*\*(rising action) Which software affected the subject’s condition?\**

At the time when I started building websites for clients, WordPress was the go-to software to use in order to get a sense of the server-client traffic behind a website. In the beginning, I charged very little and said yes to any project or deadline. The projects were simple; the deadlines extremely tight.

Because I felt rushed, I used theme-building helpers (such as Divi) to meet the client’s expectations in time. The design would highly depend on the limitations and freedoms of the software’s given functionalities: using theme builders that had an easy drag and drop functionality to create the looks of a digital space constrained my creative output. The limited possibilities these builders provided was not satisfying enough to feel creatively fulfilling to someone who wanted to break out of the default grids and blocks, a.k.a: me. Not *seeing* the code that built my decisions felt weird.

After I built two websites with extra builder products, I felt like a scam to the world of web development. I felt like *any*body could do it and that *every*body was doing it. As I was claiming the title of a web developer, I felt like I had to live up to it. But also, I just really liked coding. So I started picking up on PHP language and coding custom themes, after which I finally started feeling like a real web developer (whatever that means).

Identifying myself with this expert software in the company of other web developers made me feel quite shy at first, but upon discovering that the majority of web infrastructure is built with this same software at some point started making me feel average.

As I was building websites, it seemed to me that the software was updating itself according to the user’s needs really fast, and that whatever users lacked in their editing freedom soon became an option in a plugin library: a true catering to their needs. It is true that WordPress software updates not only grow alongside the needs of its user, but also follow the web developer’s imagination, which, when turned into a plug-in, can be uploaded to the WordPress Plugin library. That’s what open-source means; contributors add themes and plug-ins to it, for software improvement. ‘All technology reflects the society that produces it, including its power structures and prejudices.’ — Douglas Rushkoff, *Program or be Programmed*.

I don’t know if being content with theory justifying my software choice was naive, but I was having fun. I was entertained by the underlying irony of the previously explained WordPress cringe.

*\*(climax) How did the subject liberate oneself from the software’s constraints?\**

That’s when I became a Post-WordPress web developer. As a Post-WordPress web developer, I knew what I *didn’t* want from my coding endeavors: plug-ins (unless custom-coded), hamburger menu (unless absolutely necessary), WP default styling (a.k.a. classification and names according to the semantics), a deadline within the month, and having to say yes to the reducing amount of pages as well as meeting the client once a week. I wanted to make rules that would bring me closer to building *light* websites at a *slow* pace—what my friend Clara calls ‘handmade coding’.

With handmade coding, I could build a website like I’d imagined building a space back in the days of studying architecture: hand in hand with the inhabitant or citizen, the user, and their individual taste. Our collaboration would mean meeting and building each other’s differences, on whatever interface necessary.[[1]](#footnote-2)

The *Post* in my title refers to the fact that even though I am aware of the WordPress cringe, it’s the tool I choose to operate with despite its reputation. That in my productivity I have found liberation, not software enslavement; that WordPress will succumb to my needs and accommodate the skills necessary for me to have whilst collaborating with my client.

Here comes an example of my liberation. The following code was written for the client’s portfolio website and in agreement with the client:

#### <details>

#### <summary style="margin-left: 50%">editor</summary>

#### <?php $loop = new WP\_Query( array( 'post\_type' => 'editor', 'posts\_per\_page' => 10 ) );

#### if ( $loop->have\_posts() ):

#### while ( $loop->have\_posts() ) : $loop->the\_post();

#### ?>

#### <details class="secondary-details">

#### <summary class="project" id="editor-project" onclick="editorProjectTop()"><?php the\_title('⊃&nbsp;'); ?></summary>

#### <div class="entry-content">

#### <?php the\_content(); ?>

#### </div>

#### <div class="mode-buttons">

#### <button class="feel-btn" onclick="editorProjectTop()">see, hear, sense</button>

#### <button class="read-btn" onclick="editorProjectTop()">read</button>

#### </div>

#### </details>

#### <?php endwhile;

#### 

#### else: ?>

#### <p><?php echo '<div class=“void"></div>’;?></p>

#### <?php endif;

#### wp\_reset\_query();

#### ?>

#### </details>

The code indicates that there is a button called ‘editor’ on the website, which is one of the professional roles that define the client’s professional history. Clicking on this button opens a list of project titles (‘secondary-details’) in which this role is played out. Each project (‘editor-project’) is also clickable and when clicked opens the content (‘entry-content’) the client added to it in the admin panel of the website.

Content is divided between two reading modes: visual and textual. By default, textual content opens on the first click. Once you reach the end of the content, there is a button that switches between two reading modes (‘feel-btn’ and ‘read-btn’, *btn* being button). If there are no projects assigned to this role, one can choose to show void (‘void’). A void, in my collaboration with this respective client, was represented by a dark blue circle piercing the soft aesthetics of the website. In a nutshell: handmade coding of a fairly light website.

The code explained here is my favorite code written thus far. It speculates on the client’s future, doesn’t limit it to one professional role, nor demands the existing roles to be filled with more professional activities. It was written with the purpose to emphasise the uncertainty any professional role might have. These professional roles could fade out and turn into the designed void (dark blue circle), or they could be emphasized more and be accompanied by other projects in the future.

After passing two interviews for the ‘creative developer’ role at WeTransfer in 2021, I was invited to work on a technical assignment. The challenge was to code a ‘whack-a-mole’ game through which the company could get a sense of my creativity level. There was no framework, language, tool or imagination requirements, but there was one big ask: not to spend too much time on the task. In the span of the single week I was given to develop it, I spent a total of nine hours on it.

Adding to this, I must say my creativity depends on other people. I found and copy-pasted the simplest HTML/CSS/Javascript code of a whack-a-mole (by Ania Kubów) and decided to change the code according to my friends’ answers to the question: *If you could whack one thing for the rest of your life, what would it be?*

One of the answers was Facebook:

#### <body>

#### <header>

#### <div class="display-flex">

#### <p>Facebook killed:</p>

#### <p id="score">0</p><p> times.</p>

#### </div>

#### <div class="display-flex">

#### <p>Time Left:</p>

#### <p id="time-left">666</p>

#### </div>

#### </header>

#### <div class="grid">

#### <div id="1" class="square"></div>

#### <div id="2" class="square"></div>

#### <div id="3" class="square facebook"></div>

#### <div id="4" class="square"></div>

#### <div id="5" class="square"></div>

#### <div id="6" class="square"></div>

#### <div id="7" class="square"></div>

#### <div id="8" class="square"></div>

#### <div id="9" class="square"></div>

#### <div id="10" class="square"></div>

#### <div id="11" class="square"></div>

#### <div id="12" class="square"></div>

#### <div id="13" class="square"></div>

#### <div id="14" class="square"></div>

#### <div id="15" class="square"></div>

#### </div>

#### <footer>

#### <p>

#### [person] undoubtedly wants to whack Facebook all the time, and for a long time. <br>

#### Is your name not [person] <a href="/index.html">My name is not [person].</a></p>

#### </footer>

#### 

#### </body>

This is the body of a web page that was built to show the score (how many times Facebook was whacked); above it one can see the countdown time. Underneath this, a grid of fifteen squares floats in which the logo of Facebook appears randomly, and which the user is expected to click or shoot. The footer of the page asks if the user’s name is [person]’s name, a.k.a. if they wish to be someone else while whacking this evil, Zuckerberg-spawn thing.

The code was adjusted to the [person]’s specific desires in whacking Facebook, which was done in the following ways: even though the timer’s countdown started counting down from 666 seconds, there was, if desired, additional time given for completion of the mission (1.000 seconds). Next to this, the Facebook logo would hang around in one of the grid’s squares, long enough to shoot it. It was rather impossible to miss. Unfortunately, the message at the end of the countdown reminds the user that, despite their astonishing scores, Facebook still exists.

#### function countDown() {

#### currentTime--

#### timeLeft.textContent = currentTime

#### 

#### if (currentTime == 0) {

#### clearInterval(countDownTimerId)

#### clearInterval(timerId)

#### alert('There is no time left to whack Facebook. Despite your score of: ' + result + ' points, Facebook still exists.')

#### }

#### 

#### }

#### 

#### let countDownTimerId = setInterval(countDown, 1000)

I furthermore collected eight answers from my peers to the question *What would you whack?* The results were: a lack of confidence, eczema, architecture (i.e. one school building per day), sea creatures, Gerald Darmanin, applications and/or funding opportunities, and ‘I would whack people who think that rationality is better than emotions and intuitions; I would whack everyone who expects and forces me to be “clear” according to *their* understanding of clarity; I would whack the idea that if you make work you have to be able to write about it; I would whack people who tell me what to do and how to do things without me asking for it; I would whack people who say that they would never drink their own pee if they were in the desert almost dying. *Liars.*’

I loved turning the project into research about what my peers would whack and discovering if my coded reflections on their wishes could get me hired for my creativity level. During the interview with a front-end web developer at WeTransfer, I was told that I was *sooo very* creative.

You can find out the answer to the question whether I got the job or not by reading the following ‘list’ of websites I dream of building, yet never have the time to:

#### <a href="https://whack-what.vercel.app/">This was a technical assignment for WeTransfer, but I didn't get the job.</a><br>

#### <a href="/instagram-poetry/ig\_bore.html">IG bore</a><br><br>

#### <a href="#">Click here to stop scrolling</a><br>

#### <a href="#">Blue Heart Agency</a><br>

#### <a href="#">I Just Wanna</a><br>

#### <a href="#">Button Poetry</a><br>

#### <a href="#">The Writer is Present</a><br>

#### <a href="#">We only got 10 mins to save the content</a><br>

#### <a href="#">Mute everything</a><br>

#### <a href="#">Forever load, but not like that</a><br>

#### <a href="#">Rotating structures, floating points, and similar</a><br>

#### <a href="#">All known blanks</a><br>

#### <a href="#">Working station of profile production</a><br>

It’s not a list in code. Were I to write semantic code, it’d be wrapped in a <ul></ul> tag, followed by a <li> tag around each <a> tag. Then I’d get rid of the line break tags (<br>). The way this list looks like right now doesn’t bother me in the slightest. Any web developer inspecting it might find these codes very simple, because they *are*. And they bring me a lot of joy.

1. <https://motherfuckingwebsite.com/> [↑](#footnote-ref-2)