"You Want Me to Do What?": The Trials and Errors of Coding

Jessica Kreul

In this article, Jessica Kreul explores her personal experiences learning about computer coding by focusing on the genre research and repeated practices that were needed to engage in this new genre.

When I chose to add a Digital Rhetorics course to my schedule for spring 2023, I had no idea that I was going to be expected to code. I thought to myself: I'm an English major! What could coding possibly have to do with English studies? When I enrolled in the course, I figured that we would be discussing the ways that rhetorical strategies are used in online spaces. This is not terribly far off from what we discussed in class, but it certainly doesn't encompass everything that the class has to offer. So, although I approached the idea of coding with a lot of skepticism and the belief that I could never figure it out, I've ultimately realized that there are many similarities between my coding and writing processes. In other words, I can consider coding to be a new kind of literacy. It has its own rules and strategies, as well as a range of learning resources that I had to figure out how to engage with. While I would definitely still call myself a novice coder (a thing I never imagined I would say), I've learned to see my coding process as similar to struggles I sometimes have when writing. Both include multiple rounds of revision accompanied by agony, doubt, hopefulness, and finally, triumph.

Read My Lips: I Don't Code

I had the expectation that coding was going to be too difficult for me to learn because of my antecedent knowledge of the topic. Having **antecedent knowledge** is often positive, like when your experience in doing one thing translates easily to a new task that is similar. For example, imagine you are learning a new game and you come to find that it's pretty similar to one you've already played. In situations like this, your foundation of knowledge, or, at least, your awareness of possible connections between two activities, can be useful as you continue to learn about a topic and progress your skills. But prior knowledge (which is what Ambrose et al. call this kind of knowing) can also be problematic when the things you already know get in the way of new kinds of learning. Unfortunately, this is something like what happened to me. My previous experiences with seeing code and hearing it explained made me resistant when I began to interact with and attempt to understand it myself. I was expecting it to be incredibly difficult. This is because I had seen code being produced by experts who had studied it exclusively, and I never dreamed that it would be something I could begin to learn in a semester. Of course, I am only learning the basics of coding. However, the same codes that I now see as simple, I stared at with confusion a couple months ago because I believed that I would never understand them. For example, when I first opened the application that I would be working with to practice coding, which is called Brackets, I could not even begin to decipher what I was looking at (Figure 1). It looked just as complex as the coding I had seen before taking the class, and, honestly, I wasn't sure that I was willing to learn it. In fact, I was incredibly doubtful I would end up succeeding in the course and that terrified me. Despite my worries, I had committed to taking the class, and, therefore, I committed to learning how to code. After a few weeks of taking the course and having done some surface research and practice coding myself, this page was no longer as daunting to try and interpret. I can look at that code and understand what lies within the boundaries of the web page thanks to the <body> element and the fact that any of the code that appears above that element will not appear on the web page. I know that the <h1> element specifies what the main heading, or title, of the web page is, and the <h2> element specifies a subheading.

Prior (Antecedent) Knowledge

"[P]rior knowledge will not support new learning if it is insufficient or inappropriate for the task at hand. [Additionally], research indicates that inaccurate prior knowledge . . . can distort new knowledge by predisposing students to ignore discourse or resist evidence that conflicts with what they believe to be true" (Ambrose et al.).

Knowing the basics of computer coding has allowed me to pick apart seemingly complex code strings to find some sense of clarity. So, while my

Figure 1: The introduction page on the Brackets application.

antecedent knowledge surrounding coding was not necessarily incorrect, it did alter my perception because I did not have the additional context that I needed to properly situate that knowledge. Had I known what I would be able to achieve through my efforts, I don't think I would have viewed coding as something outside my realm of capability. Yet acquiring this new skill and shifting my mindset didn't come without a process of studying and experimenting, as well as going through a range of emotions that took me from horror to actual enjoyment of what I was learning.

Let's Learn About Code . . . I Guess

In order to learn how to code my own web page, I needed to first conduct **genre research**. This work is accomplished by researching the information and tools that are required or would be useful in creating a text within a particular genre. For me, this meant consulting the book, *HTML & CSS: Design and Build Web Sites* written by Jon Duckett. Before learning the specific codes that I would need to create a text for my class assignment, it was an absolute necessity that I understood what the purposes of the different elements were and how they generally operated. This meant recognizing the general structure of code and how it would translate and appear on a web page. Within a single web page, there are multiple elements that are surrounded by defining tags. Those tags are the codes that tell the computer exactly how to read and render what is situated within them; the tags will not show up on the web page, only what is written inside of them does (Duckett). While I understood what the text was getting at, I still didn't know how

to translate this idea into what it would look like in action. Once I came across a couple of images that showed exactly this process, I compared the text to the image to see how the tags defined and affected the text within them. These images helped me see that the <body> element outlines the boundaries of the page and whatever resides between them will appear in the browser window after it has been rendered. Within the bounds of this element are more elements that define headers (<h1>) and paragraphs (<p>). I use headers and paragraphs constantly in my academic writing, so the familiarity of these terms certainly helped me grasp the purpose and order of these elements—they were describing a structure that I was used to. I began to feel more confident moving forward, but I was still conscious of how I could best learn the process and did not want to overwhelm myself before I attempted to code.

Now that I had an incredibly basic understanding of what tags I needed to simply create and define the boundaries of a web page, it was time to try it out for myself. It would not have been beneficial for me to keep learning codes without actually practicing the basics first—there are simply too many codes to learn and memorize before even putting them on a page. My research needed to be a simultaneous effort of interpreting knowledge and then immediately putting it into practice. Luckily, I did not have to search hard for an application that was capable of rendering code, as my professor told us that we would all be using Brackets. This tool is incredibly simple to use; the only complex aspect is the code that will eventually be entered by the user. Although, I did have to conduct some outside research of my own to get better acquainted with the application and appreciate its many affordances before I could begin to play around with my new coding knowledge.

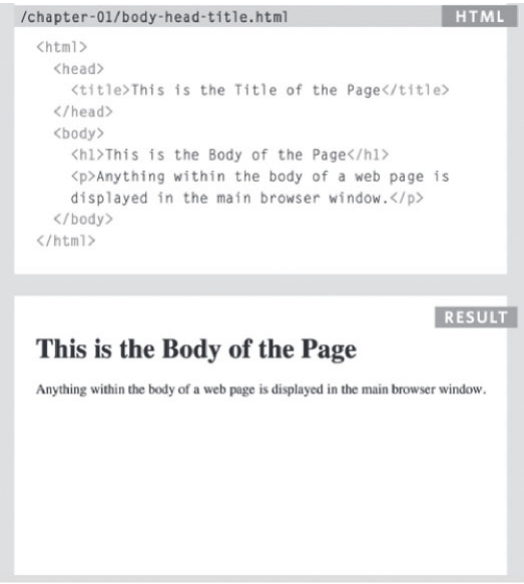

Figure 2: This image is from the book *HTML & CSS: Design and Build Web Sites* and shows code and how it would render in a browser.

One of my first coding mistakes remains a vivid memory of frustration. I was struggling to set up a brand-new page that was capable of interpreting HTML, which is a specific language used for tagging different areas of text. I had attempted to enter similar codes from the images in Figure 2, however, I did not define the page correctly as being HTML, so it would not allow

```
<html>
        <head>
            <title>My First Web Page</title>
        </head>
        <body>
            <h1>Welcome to My First Web Page<h1>
                <p>This is an HTML page that I created in Brackets.</p>
            <p>I have absolutely no idea how to code, but I feel like I'm doing it?</p>
        </body>
</html>
```

Figure 3: My first attempt at coding using HTML.

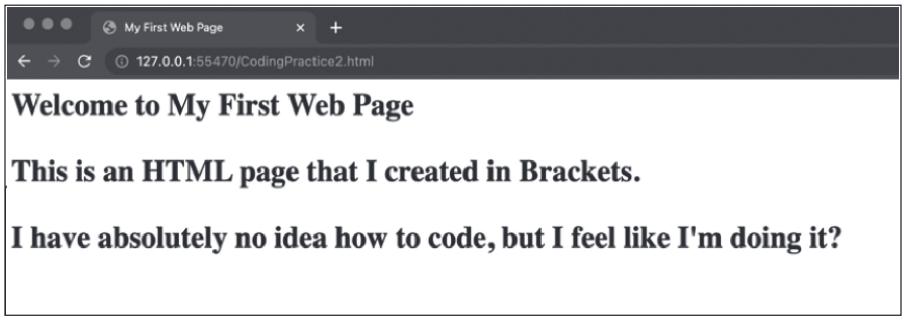

Figure 4: My first attempt at coding using HTML after it has been rendered.

the browser to open and display the text. To try to figure out what was going on, I headed to my favorite place to watch other people do the exact skill that I'm struggling with and explain it in more concise and clearer terms: YouTube. I'm a visual learner, so YouTube is my best friend. Within twenty seconds I had the solution to my first obstacle, which was two simple clicks at the bottom of the Brackets file. All this research to figure out what coding is, how it is structured, how it appears, the tools that I needed, and a few other details had to come together before the action of coding took place. Because I was able to find a solution so quickly, I did not have enough time to get so irritated that I needed to step away. My determination to learn was still going strong.

Coding as a Genre

Earlier, I mentioned that my process of learning code is like my writing process. However, the genre of HTML code, as it appears on the computer screen, is unlike the typical writing genres that I am used to. It's highly specific, and there is not a lot of room for error. If a mistake does slip through and the computer still manages to read the code, there is a good chance that the

mistake will reveal itself by altering a different element further down in the code. I suppose this is somewhat like writing an essay where one might make a statement and then contradict themselves later in the paper. However, there is a key difference that makes coding a genre that requires extreme precision. If I miss a comma in my paper, the likelihood of my intended meaning being lost is slim—the reader will carry on. If I miss a specific character or bracket required in a tag, then the element will not render correctly, which can affect the appearance of the code and ruin it. In Figure 3, there is a mistake in my code where I did not complete the element by using a slash in my closing tag, so all the text underneath it appeared as a heading instead of the paragraph style, which is set in a smaller font. You can see how this rendered incorrectly in Figure 4. Brackets is handy because it highlights the mistake in red and lets the user know that something is not working. Once I consulted the book again and this was fixed, I was able to successfully render my page and it appeared correctly in the browser for the first time (see Figure 6). The feeling was glorious; I could not believe how cool it was to see my work in action, and I wanted to code more. Although I now have the instinct to make sure I have opening and closing tags to surround my text, I did not realize how crucial this step was until I experienced the frustration of having written a block of code that was going nowhere.

While sometimes I have to go back and figure out the error in my tags, occasionally the entire element is wrong. This can happen when certain tags may have worked once before but have since evolved, or it can happen when I attempt to enter a code that my application and/or browser does not recognize. I came across this discovery while conducting genre analysis, which may sound similar to genre research but actually has a key difference. According to the ISU Writing Program, **genre analysis** examines the genre as it appears in different forms, as each variation of it may contain different features. Genre research allowed me to understand the overall basics of how code is written and how it appears once it has taken its final form on a web page, in addition to the tools needed to be successful. My genre analysis of coding was my exploration into the different ways that codes are written for particular purposes. For example, when I wanted to add a video to my web page, I needed to consult the Internet because I did not have any idea how to make that happen. Luckily, I discovered endless websites that provided me with an answer to my problem, except that they all proposed different codes. This is where the frustrations of trial and error come into play. How was I supposed to know which code would work with both my application and my browser?

There was no way to know until I tried the code, and it ended up working out. Figure 5 shows the multiple codes for inserting videos that I

```
<html>
        <head>
            <title>My First Web Page</title>
        </head>
    <body>



        <video width="auto" controls>
            <source src="img/bobmarley.mp4" type="video/mp4">
        </video>


        <video src="bobmarley.mp4" width="320" height="240" controls></video>


        <video width="320" height="240" controls>
            <source src="bobmarley.mp4" type="video/mp4">
        </video>



    </body>
</html>
```

Figure 5: Three different codes for embedding a video.

found on different websites. They are all supposed to code the same video, but the video only appeared once because only the top code worked. Although I wanted to be content with my success, it was to my benefit to understand why some codes worked for me and others did not. Studying the **genre conventions**, or features, of the code proved to be the best course of action to give me a better idea of what my application favored. Codes must be written in a certain order, similar to the way that words connect to form sentences that can be understood; there are rules. However, going from one website to another, that order started to change, and certain elements appeared in one code but not in another. Despite how similar they all looked, I needed to see which specific codes were rendering on my page, memorize the elements present and their order, and then look for similar codes when attempting to try more things later.

Where Coding Meets Writing

All of this research contributed to writing what some may consider to be simple code. Now, as an English major, I write almost daily. So, it might seem kind of silly that I am so proud of a single page that contains a header and a couple sentences (Figure 6). But as I learned more about how to do different things with code, I realized that I was actually starting to like the processes of coding, both in the research that I was doing to learn how to do different things and also the satisfaction I felt when the code that I wrote loaded and functioned properly. Through all of this activity, I think I went

Figure 6: My first attempt at coding using HTML corrected.

from reluctant learner to a proud (if still novice) coder, and I might even have a new hobby.

I have come to see coding as another way of writing and communicating. Writing code and writing as an English major are different, yes—in many ways. But coding is a really useful tool for making different kinds of texts that I didn't have access to before. For example, before this class I could have written the content for a web page but would have had to find someone else to actually create it. Now, I can do both! I have managed to compare my experiences with coding to writing in multiple ways so that the activity better fits in with what I am already used to doing despite the differences between these two genres. I am constantly writing papers because of my major, and coding has found a way to fit right in. Both require research at varying points throughout the process, whether it is looking up how to use a word correctly in a sentence or how to use code to change a font color. I have written a terrible sentence or two in my academic career, and I have written multiple incorrect codes in my incredibly short coding career. There are many ways that I can write a sentence and many ways that I can write a code, but it is about finding the best means to achieve my goal. I can then store that knowledge so that it gets easier each time I sit down at my laptop, whether gearing up to write *or* code. With both, I am constantly improving my skills and adapting my antecedent knowledge. Before I took this Digital Rhetorics course, my antecedent knowledge told me that I would have to study computer coding for years and then *maybe* I would understand, and even then, I was doubtful. Now, because of my experience with coding, my antecedent knowledge tells me that I don't know everything, but I am capable of finding answers, learning, and doing.

Works Cited

Duckett, Jon. *HTML & CSS: Design and Build Web Sites*. Wiley, 2011.

ISU Writing Program. "Key Terms and Concepts." *Grassroots Writing Research*. Illinois State University, www.isuwriting.com/key-terms/.

Ambrose, Susan A., et al. *How Learning Works: Seven Research-Based Principles for Smart Teaching*. Kindle ed., John Wiley & Sons, Inc., 2010.

Jessica Kreul is an undergraduate English major at Illinois State University. She hopes to progress her writing skills before graduating and entering a career in editing. Besides her love of reading and writing, Jessica also enjoys working out, playing Animal Crossing, and watching movies.