Website Developer Explanation

Conner Brinkley

Vice President of Graphics and IT

Modern Nook Design

12/31/14

Virtual Enterprise is truly a one-of-a-kind class. You can get a lot out of it if you choose to do so, and, oh my, did I choose to do just that. With that being said, I am not interested in the business aspect so much as I am the IT. Almost five months ago, I was granted the (what I thought was) relatively simple task of creating the company website, but it gradually evolved and became more challenging than I anticipated as I learned about more languages and emerging website design trends. Aided by opinions and suggestions from various family members, friends, and fellow Modern Nook Design employees, I, alone, fully designed and coded the website from scratch. For obvious reasons, I cannot go over every little detail that went into the design and coding process. However, I am going to discuss several important methods and sources that I used to enhance my knowledge of the vast world that is code, as well as the innumerable challenges I had to overcome. In the near future, I wish to use the experience I gained from coding my first website in this class to stay relevant in the ever-growing IT industry. Now, let us take a dive into the "short" five month development process.

Throughout the first week or two in August, I learned the basics of the simple website language, HTML. I truly believe that the best way to learn code is to get your hands dirty and try out different combinations to see what works and what does not. Not really having a clue of where I was going, I coded the skeleton of our homepage. Which included the <html>, <head>, <body>, and a few <div> tags. Over the entire course of learning this language, I noted some HTML 5-specific tags and attributes. I then implemented some into the website, not really worried about older browsers at the time. If you take a look at my code through "Inspect Element" you might notice I used <section> tags often. I used these more or less to define how the entire body was broken up into parts that were 800 pixels high. Granted, not all <section> tags were, but a vast majority were this tall. I wanted most of the sections to cover the entire height of the window, so I felt like this height would be enough to cover (most) windows fully. This was still very early in the development stages.

My next discovery was Twitter's very helpful framework for web developers, Bootstrap (which is an approved advanced resource). As I read up on this and watched many YouTube videos of Bootstrap examples (while simultaneously learning CSS3), I was (almost) certain I wanted to make a mobile-friendly website. I did not think it would be that difficult, as Bootstrap makes this option easy for anyone who knows how to properly use it. Being the highly-experienced developer of three weeks (sarcasm intended), I made the bold decision to try and use Bootstrap's mobile-friendly features, such as the collapsing navigation bar (navbar). What I mean by collapsing is that the navbar readjusts according to the width of the browser. This plan worked out until I added more and more and found other ideas that I wanted to implement into the site. For example, when I tried to add our company logo into the top navbar, it just did not fit, or even look right when it did. Although the collapsing navbar did not work out, my hopes for a mobile-friendly site were not crushed just yet. I quickly found another alternative, which was the off-canvas navbar. I was in love with this idea, the only problem was that I did not have a clue on how to create it.

I researched and found that to create a button that would move the entire wrapper <div>, jQuery was a must. Not knowing jQuery at the time (and slightly intimidated to learn it), I thankfully found an alternative. I watched a video on YouTube by YouTuber, devtips, and he explained how you can use a checkbox in HTML to achieve the same goal. I tried this out, and it worked. To go in a little more depth, I coded a simple checkbox outside the wrapper <div> and then coded its <label> inside. I then changed some CSS, and masked the checkbox with a button (nav-toggle). Finally, I added a pseudo class on the checkbox of ":checked", and linked it to the wrapper to give it a specific margin-left and transition. In the end, the wrapper moves right to reveal the hidden navbar whenever the nav-toggle button is clicked (or checked).

I finally gave up on the idea of creating a mobile-friendly site when I was scouting out website trends and stumbled upon parallax. I did not know much about it at first, but I knew that I was determined to include it in my website. Parallax is the design in which certain elements move at a different speed than others. It is used in many types of media, but in website design, the background is usually the element that moves slower than the rest of the content. This is the point where I knew that I could not go on without learning JavaScript and/or jQuery. It was the bridge that turned my website from a basic beginner site, to a complex beginner site. I researched how to write parallax functions for many days until I came upon one that stood out. It was a demo by YouTuber optikalefx. He explained, in extreme depth, a function that enabled background images to move slower than the <div> or <section>. Just how slow it moves depends on the margin between how high the <section> or <div> is and the height of the image. For example, if the <section> is 800 pixels high, and the image is 1,000 pixels high, then the image has 200 pixels of movement. He explained all of his code well, and it was very understandable. I included this design with the home page and about page. At the end of the function, he used a requestAnimationFrame() loop. This render loop refreshes the display at a speed up to 60 frames per second (fps). Unfortunately, with the very high quality images that I used, not everyone will be able to experience the website at this frame rate. Computers with newer and more advanced hardware will be able to render at higher fps. The laptop with which I coded clocked a speed of a smooth and constant 40 fps. However, my older desktop only maxed out at an inconsistent 20 fps on lower quality images. This problem did not concern me too much, being that I was targeting newer browsers in affiliation with newer computers. The website is at its best when displayed on these premises as well as a 16:9 ratio monitor.

To really make the parallax stand out, I came up with the idea to have a smooth animation whenever the scroll event was triggered. Since I already had anchor links, I wanted this “smooth scrolling” idea to be included in that as well. I took to Google and found a few good tutorials and examples. After I included that feature, I wanted to add a search bar so that returning customers could find a package quickly and easily. I coded a sidebar on the right (the opposite of the navbar) where the search bar was going to go, this time using jQuery for the toggle button. Sadly, after spending a great amount of time coding the sidebar, I learned the difficult lesson that the search bar was much more work than I originally had thought. I then abandoned the idea and just added extra buttons on the navigation bar to make the entire site more accessible. If you look through my "main.js" file, you can find code that was commented out which originally enabled the search sidebar to be toggled. I kept this, along with other commented out code, just in case I decided to salvage what I had and transform it into something else of use.

Once I really started to learn the basics of jQuery, I used it whenever I had the opportunity. Originally wanting to avoid using it at all, I ended up with over a thousand lines of jQuery (which seems like a lot to me). I used it to create arguably one of my favorite aspects of the website—the fade in content. It was relatively simple (just a function with .fadeIn() and .fadeOut()), and I used it with the contact information, Virtual Enterprise disclaimer, catalog information, and order form. Not only did I use jQuery for fade effects, but I also created animations with it. On the about page, I created a sliding animation for the commercial whenever it was clicked. The effect is the same as unlocking a phone. I gave the play button a left value with .animate() and had a transparent black background fade out whenever it was clicked. In the end, the play button slides left and reveals the commercial (almost like unlocking a phone). Another favorite feature is the hover effects on the home and contact pages. I linked two <a> tags (links) and gave a hover effect for both when you only hover over one. For example, when you hover over the button that says catalog on the home page, the shopping cart icon (catalog) on the navbar also has a hover effect too to symbolize that the two are linked and go to the same place. The same technique is used on the contact page where hovering over the word Twitter on the navbar also gives the hover effect for the big Twitter button on the page. The jQuery method I used to achieve this most likely is not the simplest of ways, but it worked for me. I just gave the buttons a .mouseenter() and .mouseleave() event and then defined the CSS properties. It was a simple concept for me, but it came with the price of writing a great deal of CSS properties in jQuery (which is not very time efficient). Ultimately, I am not complaining, because learning jQuery helped improve my website immensely as well as my knowledge of coding in general.

One specific issue that I had to overcome was the Google map on the contact page. Coding the map itself was relatively easy, considering that Google thoroughly explains how to code and customize the map. On the contrary, the problem that I ran into was that the map was not rendering fully whenever the page loaded. I was stumped, because I did not know the reason behind this issue. After many hours and several forums on stackoverflow.com, I finally did pinpoint the reason, but I did not have a clue on how to fix it. In short, the reason was because I had the map in CSS to display:none. I set this, because I was planning on using a .fadeIn() effect. The map was not fully rendering before the page hid the map in the background. I changed this CSS setting, but that defeated the whole purpose of where I wanted to place the map. Fortunately, I thought of a solution. I kept the new CSS display, but I gave the map an opacity of zero and called the class "contact_hide". I then added the following code in an onload jQuery function, "$('#email-popup').fadeOut(500).removeClass('contact_hide');". What this accomplished was that once the map had time to render properly, it was faded out and given the full opacity. That way, whenever someone wants to see the other contact information, the map will fade in and be rendered correctly.

A second major issue that I solved was emailing the order form once submitted. To accomplish this, I had to learn a fraction of another language that I was not initially planning to use, but it was the only way to send the form. I learned a bit of PHP, more specifically the PHP mail function. This was not too terribly difficult to learn, being that I just had to define variables (such as the package that the customer wants), and write, "mail($to, $subject, $message, "From:".$from);". To do this, I had to define each input's name, and then set a variable equal to $_POST[name];. After that, I set each variable equal to another variable called $message (which is the message being emailed). Finally, the $to variable was our own email. The only problem that occurred was that the email took several minutes, sometimes hours, to receive. After a day of trying to solve this, I came up with the only reason that made sense. It took so long, because the email had to cross from our domain to Google's mail server (gmail). Thankfully, GoDaddy gave us a free domain email with our domain and hosting plan. I made an email account on our domain called mnd@modernnook.com. Once I set the PHP mail function to mail to that address, the order form was sent and received instantaneously, because the email stayed in our domain. Now, whatever is input on the order form will show up in our email in an instant once it is submitted. Afterwards, the PHP redirects the user to a "Thank You!" page. One thing I wish I had time to do would be to learn and write PHP validation rules so that I can filter what comes through and what does not. I did actually write a rule that forces the customer to input something instead of leaving important fields blank. All it does is check if any important fields are blank before it sends. If any of them are, then it redirects to a page that shows required fields. If I had an extra month or so, I wanted to learn how to use AJAX to dynamically load the "* required fields" indicator without refreshing instead of redirecting to a new page. Although I did not originally want to learn PHP, I am glad that I did, because of the extra experience in coding languages.

One statement that I made earlier was of the poor frame rate. I found a solution to this online by accident, but it was already too late. I tested it out, and the solution really helped, but it interfered with the parallax design. To summarize, it would have required me to redesign the entire website. Although that was not going to happen, I thought the idea was very clever. The solution was to use a technique called GPU acceleration. When most people view websites, their computer is using mainly the CPU, or processor. However, an experienced developer can add "dummy" code that does not really change anything to the display, but it forces the computer to use its GPU, or graphics card. The code would be some form of CSS translate3d set to (0, 0, 0). Once the computer starts using the GPU alongside the CPU, the frame rate is much more reasonable.

The last feature that I added was code to enable the navbar to be dynamic. Since the navbar is mostly made of anchor links, I thought this would be a nifty feature. It highlights whatever anchor link you scroll to, without having to click it. As you scroll down the page, the links are dynamically given the class of active according to the section you are on. I wanted to do this ever since I created the navbar, but I was restricted due to my lack of knowledge of jQuery. With Stack Overflow by my side, I used a dynamic function that automatically changes the active class (hover effect) relative to the section in the page content. Of course, with coding, whenever you add a new feature, a thousand new problems surface. Well, that may have been a hyperbole, but there were actually two problems that occurred. I am not going in great detail, but one was just a conflicting CSS error that was not a problem until I added that function. All I had to do was change the active class a bit and add "!important;" to the end of the rules. The second was the carousel slider on the contact page. The side arrows were not working, but I just removed the "href='#myCarousel'" and changed it to "data-target='#myCarousel'". The reason was because the dynamic function that I used mistook the arrows for an anchor link. The good news is that it was the last error I had to fix before I finalized the website.

The past five months have been a bumpy road. I look back at it now and realize just how lucky I am to have had this opportunity to learn code, especially since I hope to have a career in computer science someday. I have encountered problem after problem, but I overcame them in order to create something of which I am truly proud. Perhaps in the future, I can extend my new knowledge of code by learning about app design and programming to make a Modern Nook Design app that would make up for the not mobile-friendly website. I had a little less than five months to learn and utilize four different coding languages, all while trying to keep cross-browser compatibility and equality. I can honestly say that my final product is better than I initially envisioned.

All resources used:

<u>Learning Purposes</u>

- W3Schools
- Forums on Stack Overflow
- Many YouTube tutorials
- Several other websites with tips and tricks with HTML, CSS, JavaScript, jQuery, PHP, and website design (css-tricks.com, paulirish.com, etc.)

<u>Used in the Website</u>

- Bootstrap CSS/JavaScript/Glyphicon CDN (high end framework)
- Google Fonts CDN
- Font-Awesome CDN (extra free symbols)
- Modernizr (older browser support)
- jQuery CDN
- Tweenmax jQuery plugin CDN (smooth scrolling)
- Scroll-To jQuery plugin CDN (anchor link smooth scrolling)
- Google Maps API
- Google Images
- Paul Irish requestAnimationFrame API (vendor prefixes)
- Parallax demo from YouTuber, optikalefx
- css-tricks.com smooth scrolling snippet