Engineering Innovation

A Text Pamphlet on the Basics for Learning Programming, Engineering, and Project Application

Written and Compiled by Chava Alexandra

Contents

Use Your Resources

Utilizing your resources is one of the two most important parts in innovating and learning, with the other being Communication. This is why there are two chapters, one at the present and the second at the end, devoted to this concept. You will indeed learn a lot, especially by merely doing over listening, but there will always be times when you need to know information immediately to save something or someone. This is where knowing how to use your resources comes in handy.

Throughout our lives we learn. Learning is defined as *the act or process of acquiring a skill*, and as *the modification of behavior through practice, training, or experience*[1]. We gain skills every day of our lives through talking, listening, seeing, and doing. Whether you are in the classroom or in the park, at work or at a party, you are interacting with people and objects, which means you are learning things from how to write an essay to how to throw a baseball, from how to turn on a light switch to how to build a couch-fort that won't fall down even if a mastiff sits on it. In most of our K-12 classrooms, we sit and listen to the teacher. If we're lucky, we can listen to and absorb most of what he/she says, and if we're really lucky she/he will let us put the lessons to practice. Our teachers are a resource. At home, our parents, grandparents, older siblings, neighbors, babysitters, guardians, and such, teach us how to dress ourselves, how to operate the television, how to drive a car or do our laundry. Our caretakers are a resource. On the playground, our friends show us how to act, what to say, who to hang out with; they tell us what is "cool" and which classes to take and what sports to play; they teach us how to behave with our peers and we often learn our first swear words from our friends. Our friends and our peers are a resource.

And then there is Google. Google's search engine could be defined by the tagline: *whatever man doesn't know, Google knows*. Dictionaries, thesauruses, atlases, encyclopedias, records, statistics - these are all resources and various versions of these are accessible online, all you have to do is search for them. Knowing how to search, when to search, and for how long to search is Part I of Using your Resources. Part II is being able to discern the reliable, useful resources from the rubbish of fabricated and over-exaggerated products of bored and deranged minds. This ability mainly comes with

practice. Some good guidelines are to note the information's references - does the information you've found come from a reliable source? Also, as often as we use shorthand in text messages, emails, and such now, nothing will change the fact that honor and trust emanate from good grammar - if it doesn't look right, it probably isn't right.

The answers are always out there. The tough part is figuring out the questions. Once you get the hang of figuring out what questions to ask and where to find the answers, you'll be set! And, like most things you learn, repetition is the way to learn it.

How & When

Type of information	*Long explanation - may be easier to have someone explain it* *A quick question - search for a video, how-to guide, diagram* *How to do/fix/build - videos, diagrams, and wiki step-by-step how-to's*
You're stuck	*You need information fast*

Is it a good source?

Reliable Source	*It's trusted by many*
Well-written	*It was checked, reviewed, and edited*

Journaling

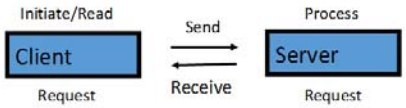
Track your progress and keep track of your ideas, decisions, and movements. Writing things down, much like repetition, helps to cement concepts in your brain, much in the same way that rolling a wheelbarrow across the same dirt path forms a rut, which, in turn, makes it ever easier to move the wheelbarrow along that path. This is only part of what journaling does for you. It also provides you with an excellent resource and reference to consult that is in your own words, that *you* have format and language control over.

There are many things you can journal about, from notes to answers to doodles to ideas. Separating these into two categories helps keep your journals organized and easy to sift through. I suggest having two journals - one to keep your ideas, doodles, project drafts, etc. in, and the other for your notes, questions/answers, and quick references (like $V=I*R$). I do not recommend combining or mixing the two. Your thoughts and ideas will shift as you increase your knowledge, understanding, and accomplishments, thus changing the shape and aims of your ideas. Coupling that with the facts that you wish to keep record of will leave you with a rich jumble.

What sort of things should you track? In the Facts journal, log equations and explanations, questions and answers, terms and definitions, and any simple diagrams that help illustrate key concepts. In the Design journal, doodles and sketches, ideas and proposal drafts, storyboards and outlines, snippets of code and quick schematics, and the beginnings of any project ideas. Of course, what you ultimately end up journaling should be what you determine to be helpful. The same goes for how you journal it. Teachers, professors, employers, and those hard-to-please, and often hard-to-explain-to, grant lenders all may have their own requirements for notes or papers you turn over to them, but your journal should be in your words and how you feel most comfortable writing it. Just keep in mind that with whatever you are writing and recording, you should be comfortable reading it, thus meaning it should be done in a way that is not only easy to write but also useful.

Your journal is your reference and, sometimes, your lifeline. It's a quick reference and can be your most valuable tool.

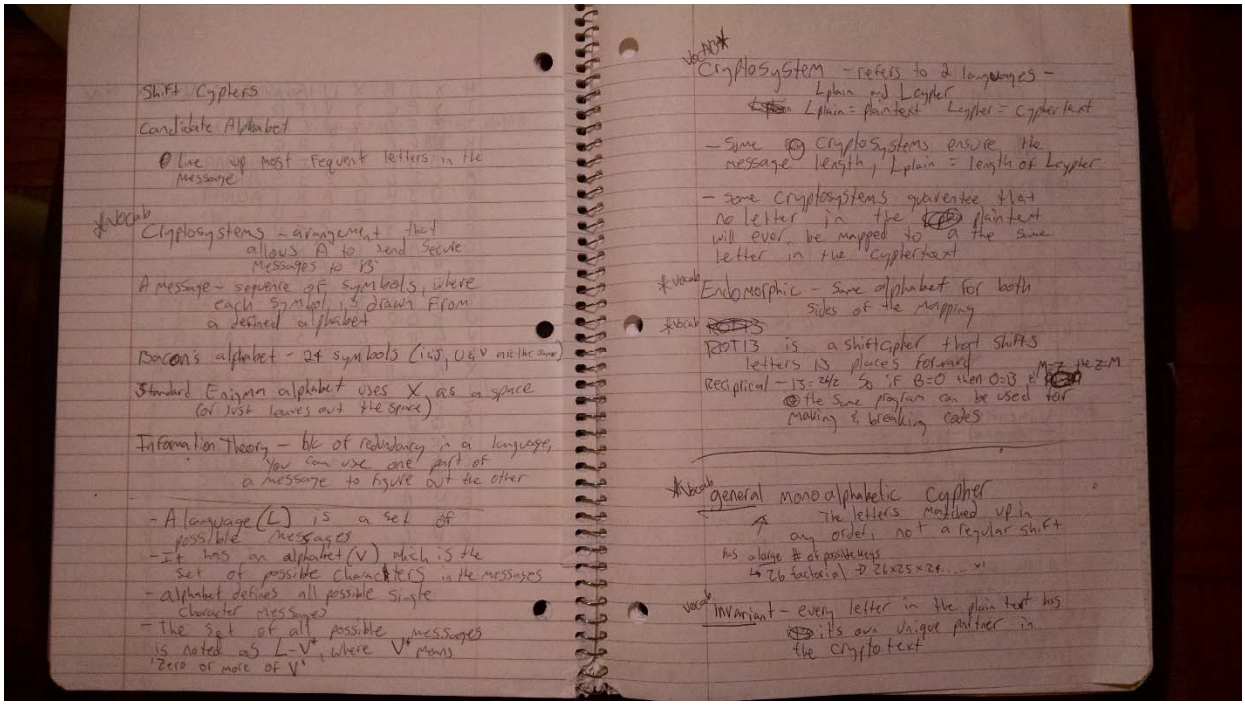
Facts Journal

Equations	*Voltage = Current X Resistance (V = I•R)* *1 inch = 2.54 centimeters*
Explanations	*The <head> tag stores meta data for the page* *Resistance is the friction encountered by the Current travelling along the circuit*
Questions and Answers	*How does Encryption work?* *Encryption is the process of taking plain data, like the password you input, and using an algorithm (cipher) to change it into a code undecipherable by a reader (unless they have the key)*
Terms and Definitions	*Server: Computer storage that lends out space to various data inputs, usually in the form of "renting" out space to Clients*
Simple Diagrams	

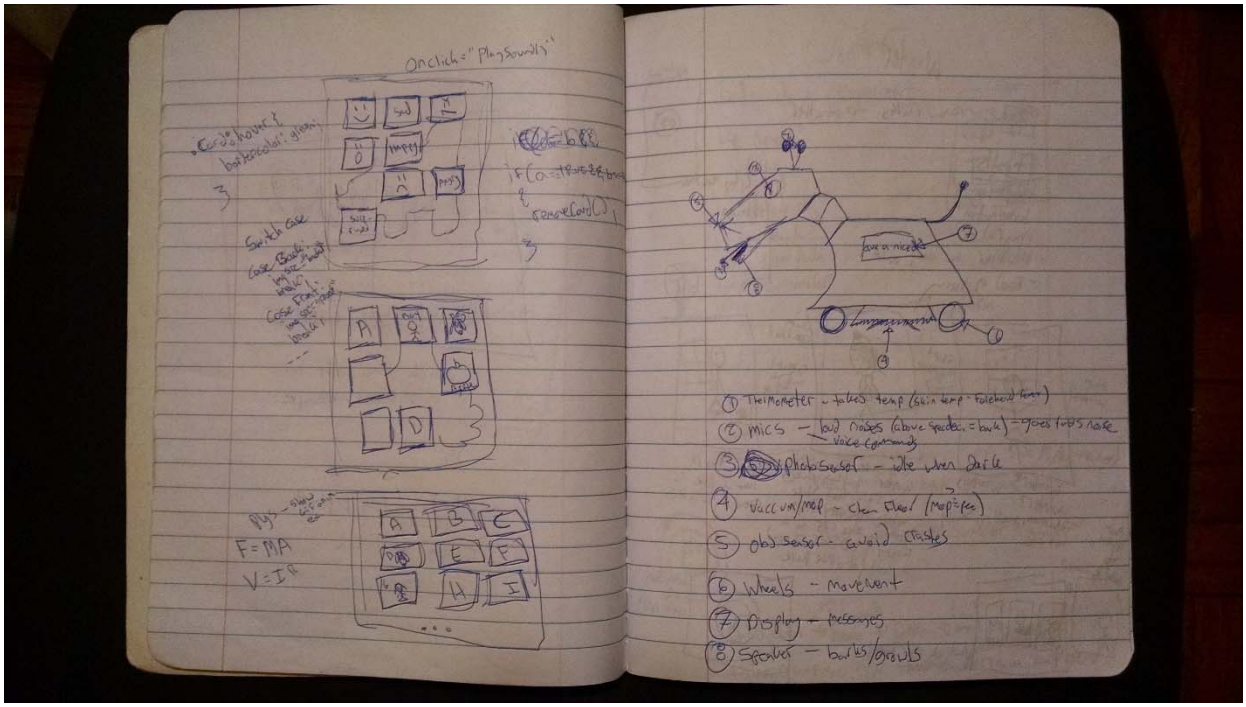
Design Journal

Doodles and Sketches	*Draw out your ideas, thoughts, caricatures, plans, anything that comes to mind whether you're listening to a lecture or singing in the shower*
Ideas and Proposal Drafts	*Map out in as much detail as you desire (usually enough to remember what you were thinking) any ideas that you think up - even if you don't plan to start on it for awhile* *I find it easier to start writing drafts of ideas/proposals by hand, it makes it more visual*
Storyboards and Outlines	*Draw your plans out - this is how you explain visually what your idea/project/program will do* *The more detail you include, the less thinking and work you'll have to do later - and it makes it easier to decide what will work*
Snippets of Code and Quick Schematics	*Sketching out brief blocks of code will help you recall later how you are planning for something to function, meaning you'll better remember how it is supposed to work* *Schematics are like blueprints for your hardware, whatever you write or draw now will help you out a lot later*
Beginnings of Project Ideas	*Brief notes on what you plan to do and how you plan to do it* *Remember - the more you jot down as you think of it, the easier it will be to figure out a plan later and execute that plan during your building phase*

Real Examples



Facts Journal



Design Journal

Quick Tips on Learning

How do you learn best? Some people are great note takers, others record the teacher's lectures and replay them over and over. Some folks can only learn by doing, known as *Project Based Learning*. There are people who just need a one-on-one explanation in order to "get-it". People will listen to audio lectures or watch videos or read books and articles in order to learn. All of these methods can be broken into three main learner categories: Auditory Learners, Visual Learners, and Kinesthetic Tactile Learners.

Auditory Learners, like the name suggests, learn by listening. They can attend a lecture or listen to an answer and walk away with a decent understanding of everything they need to know.

At the other end, Visual Learners learn by seeing, or watching. Videos and presentations, such as the teacher performing a classroom experiment, work best for them.

Kinesthetic Tactile Learners learn best by *doing*. They need that hands-on experience; they learn with their hands and fingers, by digging into the material in the literal sense.

An individual is not confined to one learning type, though many people tend to do better with one type of learning than others. The more you learn, the better you will be able to see a pattern in how you learn most effectively. Try to cater all of your acquisitions of knowledge towards the way in which you learn most effectively, even if it means you have to build models of concepts or ask someone to explain it again.

This goes back to knowing how to Use Your Resources. If you feel most comfortable learning by listening, ask your teacher to explain the concept to you, or listen to a video. If reading helps you understand, find an explanation that provides you with the information and details you need. If you need to get your hands dirty and *physically* learn, then try doodling or drawing a cartoon to illustrate the concept, or making a video, or even explaining it to someone else.

The most important thing to do is to *want to learn*. If you don't want to learn, you probably won't. If you want to learn but you feel the subject just isn't interesting, then ask

yourself why you need to learn it. If you can't figure out why, perhaps your teacher can tell you, or, check out the world around you - can you see where the subject might come up?

The next most important key to learning is to take an *active approach* to learning. This requires participating. Ask questions, give answers, listen to the answers your peers give and give something back. Even by arguing we are engaging, participating, and this, too, helps us to learn. You learn how to ride a bicycle by riding the bicycle, not by reading about it. You'll recall the details of a bicycle crash better if you are either the one to crash or are a witness to the crash, rather than if you only hear about it on the news.

The third tip is *repetition*. They say it takes 30 days to make or break a habit. The more you do something, the faster you'll do it the next time, and the more routine it becomes, and the better you know it. Think about learning to swim, ski, play piano, play basketball, or even walk. You had lessons, you had practice, you did drills, you fell down, you did it over and over and over again. This is where note taking comes in handy. Why take notes? Take notes merely because by writing down what you hear, you are engaging in a form of repetition (as well as actively learning).

Try reciting something eight times while facing North, then eight more times while facing East, then another eight South, and another West. Repeat it at school, in the car, in the kitchen, in the shower. If you can do it in one form or place, you might remember it; if you can remember it in two forms or locations, you'll probably remember it; if you can remember it in three or more forms or environments, you'll definitely remember it. Repeat concepts and equations to yourself in the shower, repeat them to your friends. Write out definitions, formulas, equations, and blocks of code multiple times. Whatever you need to commit to memory, just keep doing it. Soon it will become second nature.

Quick Look at Quick Tips

Three Main Learning Types	*Auditory, Visual, Kinesthetic Tactile*
Desire to Learn	Get interested in the subject, find a real-world application, try to expand your understanding, find out why you should be interested
Active Learning	*Get your hands dirty* *Ask questions* *Give answers* *Take notes and doodle pictures* *Discuss, Listen, Respond, Argue*
Repetition	*Take notes* *Repeat (summarize and then repeat)* *Practice over and over* *Repeat, or relay, concepts to someone else (a friend, a classmate, a sibling, the kid you babysit and want to impress)* *Write and rewrite key concepts, equations, definitions, and formulas* ***Just keep using what you're learning***

HTML

Hyper-Text Markup Language is the language the Internet speaks. It quite different from most other programming languages you will encounter. When it comes to programming, HTML is only a beginning. With *Content Management Systems*, like Joomla and Wordpress, **anyone** can build a website, but by knowing and understanding *what* makes a website and *how* to make a website (including data manipulation and SEO), you'll be **someone**.

Though webpages are often written using other languages, such as PHP or other *server-side scripting languages*, HTML is the basic language used for writing webpages. Ultimately, almost everything comes out as HTML in the end. HTML is what is called a *markup language*, meaning it's very similar to the way we talk or what we are used to reading, but it is significantly different in style and manner than plain English. In this light, HTML is written using HTML elements called "tags", these are sort of like labels, or tab dividers, that you might use to organize items or papers. Tags are written using angle brackets, such as <html> or <body> or <script>, to distinguish them in the code from the actual text, the plain English, that will appear on the webpage. While there are a ton of different tags that you will eventually pick up the more you use HTML, there is only a handful that you will use in almost every page.

For the following section, it is highly recommended that you re-type every line of code I provide in the examples. I suggest making changes one at a time, saving, and then refreshing your browser page to see them. This helps you see what each individual part does. If something suddenly doesn't work right, double check the code and then, if needed, back track until it works again and move forward slowly. Keep in mind that, in a lot of programming projects, for every hour of coding there are about three to four hours of *debugging* (i.e. trying to figure out why the heck the code isn't working correctly).

The first thing you need to do is to set up your text editor. There are tons of them. Find one you like, do a search and read reviews, or talk to other people and see what they prefer using. I most often use Notepad++, though it is, as of now, only available for Windows. Sublime Text is another very excellent text editor and is available cross-

platform. There are still many others but either of these should suffice for your purposes. (With some basic coding, you could also build your own text editor using a *general-purpose programming language*, such as C++ or C#, but this is not for now.) Of course, for those who would prefer to do it old school, there is always Notepad on Windows (TextEdit on Mac OS), where you just need to type *.html* at the end of your document title when you save it. There is also the Cloud, where you could use something like the Cloud9 *IDE* (IDE is an *Integrated Development Environment*) to write and preview your code. You can find out more about that with a quick search.

Once you choose and download your text editor of choice, take a moment to orient yourself with its interface - I find learning the keyboard commands (like Ctrl+S - Command+S on a Mac - to save) very helpful, since I'll be using the keyboard a lot anyway.

> (On that note, it might be of help for you to know one of my favorite keyboard commands: Alt+Tab (Command+Tab on Mac), which enables you to easily switch between open programs on your computer without having to leave the keyboard in search of a mouse. Take a moment and try it. Hold down the Alt (Command) key and the tab, then let go of just the tab to see all of the windows you have open.

Once you're done enjoying the thrill of Alt+Tab, get back to your text editor and open up your favorite browser. Now, whenever you make significant changes to your code, save and Alt+Tab over to your browser window, press Ctrl+R (Command+R) to refresh the page and display your changes.

The first thing you need to know when coding is that **everything is case-sensitive** (okay, so not *everything*, but close to it so you may as well treat it like it is all case-sensitive). This means, if something is written <class="MyParagraph"> then every other time you use the term "MyParagraph" it has to be typed out as "MyParagraph", not "Myparagraph" or "myparagraph". This is one of the most frequent sources of bugs in the code. *I've spent hours debugging just to find out that I had a "myParagraph" when I should've had a "MyParagraph".*

Now, let's get onto the HTML:

<!DOCTYPE> is always the first thing you put in your HTML code, right at the very top. This declaration is used to tell whoever (or whatever, in the case of the browser that processes your code) reads it what language the following code is written in. It's like asking someone to translate the text of a foreign language and letting them know which language it's written in - if it's written in Spanish, the translator will know to access his Spanish dictionary, etc.

In HTML5, and for your purposes, you will write the <!DOCTYPE> declaration as follows:

```
<!DOCTYPE html>
```

This is typically followed by the <html> tag, which essentially says that you are now starting to write the HTML code. Again, this is a markup language, meaning everything is very well organized and documented. You'll always know what section you're looking at because of these tags, these section markers.

The <head> tag usually comes next and opens up the space where you will store your metadata. Metadata is the stuff the browsers and search engines read but that you won't see on the actual webpage when it's displayed in the browser. Things like links to external code files and information about the author of the code and keywords about the site are some of the things stored inside the <head> tag.

Aside from opening tags, like <head>, there are closing tags, such as </head>, which you need for every opening tag (with the exception of the <!DOCTYPE> declaration). Aside from forgetting to use the correct case, another common bug in HTML coding is forgetting to close a tag.

So now our code should look something like this:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
</html>
```

The last section tag we'll add is the <body> tag. (There are other section tags, but these three, <html>, <head>, and <body>, are the ones you'll use on every HTML page you create.) The <body> is what contains the content of your webpage that you actually see displayed on the browser when you view the website, unlike the metadata in the <head> section. The <body> follows the <head> section and is also contained within the <html>.

```
<html>
    <head>
    </head>
    <body>
    </body>
</html>
```

Now, the fun stuff.

First, **save your document**. You should create a new folder and save all of your documents for this specific website together in this folder so the files can speak to each other. Once you have your folder, you can name this webpage file something like "MyFirstWebpage.html", just make absolutely sure that you are saving it as a file type of HTML (in Notepad++, this requires selecting the dropdown for *Save as type:* and selecting the "Hyper Text Markup Language" option), you should see the .html extension added automatically. The default for most text editors is to save files as a .txt (called a *text file*), which will not process as an HTML document. Try opening a .txt file in your browser.

Now that you have your HTML file, you open it in your browser. To do this in Notepad++ you will go to "Run" on the menu bar and select your browser. You can also locate your saved webpage in the folder you created and saved it in, right-click it and open it with your browser (because it's an HTML document, it may automatically open in your browser when you double click it).

When you open it in your browser it should be completely blank, since we haven't added any content yet, only the sections for the content. It's like having a filing cabinet

with labeled folders but no papers inside them.

Back to the code. Let's add content and make something appear on the webpage in your browser.

In the <head> section, we're going to add <title> tags. The <title> is the container for the words that appear on the tab at the top of your browser page. When you have ten tabs open in your browser, it's these words from the <title> that you can see in the tabs line the top of the browser, signaling that you have those webpages open. Let's give our page the title of "Home".

```
<title>Home</title>
```

Note that the <title> also has a closing </title> tag and that the text "Home" is contained within that.

Go ahead and Alt+Tab (Cmd+Tab) back to your browser and Ctrl+R (Cmd+R) on your webpage. The main part of the page should still be blank but the tab at the top should now say *Home*.

Onto the currently-very-boring, whitespace that makes up your webpage.

Inside the <body> tag, since that is where you put the content that shows up on the page (this is an example of repetition, which we went over in Quick Tips on Learning), add the <h1> tag. The <h1> through <h6> tags are pre-set header tags. These display text that have their font-size and font-weight (thickness) attributes already set up and defined in the global HTML dictionary that every browser uses to process HTML (we'll cover styling your text - adding color and changing font sizes - later). Header tags are used the same way chapter and section headings are used. Inside the <h1> tag put the heading for your page. In this case, I'll use "Welcome to My Website" as my heading for this homepage.

```
<h1>Welcome to My Website</h1>
```

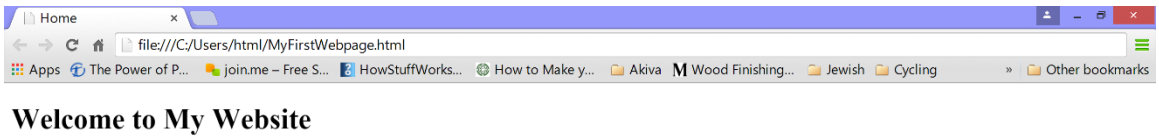
Note that we again contain the displayed text in between the opening and closing tags. Our code should now look like this:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Home</title>
    </head>
    <body>
        <h1>Welcome to My Website</h1>
    </body>
</html>
```

From now on, to save on space, I am going to leave out <!DOCTYPE> and <html>, just remember that they still remain there in your code.

Save your file. (You should save every 2-3 minutes, or after a significant change - you can use the keyboard shortcut of Ctrl+S (Cmd+S) to make saving flow along smoothly with your typing. **If you don't remember to save before you refresh your webpage, you will not see any of the changes**. You don't want to spend an hour of debugging and erasing code you've just written while trying to figure out why your page didn't display correctly when the only issue was that you didn't remember to save.

Alt+Tab back to your browser and refresh the page. You should now see something like the following:



Note that you can see the file path for my HTML document in the navigation bar at the top, where the URL (web address) goes. I created this file, "MyFirstWebpage.html", in the folder I created for this project, which I called "html", which is located in my "Users" folder on my C drive. This is called saving locally, since it is not posted on the World Wide Web, and, therefore, cannot be accessed from any other computer.

If you don't see what is displayed in my example, go back and double check the code to make sure it matches. Also, double check that you saved the file as an ***.html*** file. Below is a quick checklist for debugging:

Debugging Checklist

File type	*Is it a .html file?*
Case-sensitive	Are all of my capital letters capitalized and my lowercase letters lowercase?
Spelling	Did I misspell anything? Did I write <titel> instead of <title>?
Tags	Did I close all of the tags I opened? Did I open all of the tags I closed?

Writing code and developing programs, applications, apps, and anything else can

be very frustrating when it's not running correctly, or at all, and you can't seem to find any problems with the code. Sometimes, walking away and taking a break for a few hours, or for a shower or a bike ride, and then coming back to later will help you find errors. Other times, asking someone else to look over it or reading it out loud will help you to catch where the problem is. Many text editors and browsers can help, too, if you run the debugger, since that will show you what line of code is causing the problem. Sometimes I've found, after hours of debugging and a sore foot from kicking my desk in lieu of kicking my computer, that the error stemmed from something simple like writing *<body>* when I should've written *</body>* or *<H1>* instead of *<h1>*, etc. So don't despair! Until we have computers coding themselves, there will always be room for human errors.

(At this point, it's worth noting that one of my favorite error codes is the EEBKAC error, which stands for *Error Exists Between Keyboard And Chair* – as it often does.)

Moving onwards with some more HTML, let's expand our current webpage, which only contains a tab title and some header text so far, to include some simple paragraph text and a link.

So far, we are here (remember, I am leaving out the <!DOCTYPE> and <html> tags now):

```
<head>
    <title>Home</title>
</head>
<body>
    <h1>Welcome to My Website</h1>
</body>
```

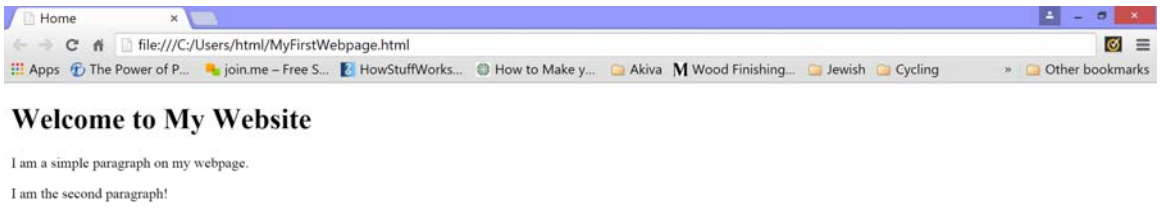
Underneath the header, let's add a line of text using the tag. Like the <h1> and other header tags, the tag has preset style settings that give the text a default size and color. We'll stick with the defaults for the moment. Go ahead and add an open * * and a closing * * to your code:

```
<head>
```

```
    <title>Home</title>
</head>
<body>
    <h1>Welcome to My Website</h1>
    <p>I am a simple paragraph on my webpage.</p>
    <p>I am the second paragraph!</p>
</body>
```

Now, save your file again and Alt+Tab back to your browser page. Refresh the page and view the results.

Now, your webpage should look like this:



Let's take one of those new <p> lines that we just wrote and add some Inline Styles. (Quick disclaimer: I don't recommend using inline styles except on special occasions, but it is a good way to see quickly and easily how a different color/font/size/etc. would look without having to edit the stylesheet, which we'll get into shortly.)

Here is our second <p> tag:

```
<p>I am the second paragraph!</p>
```

Let's modify it by adding some extra code inside the opening <*p*> tag:

I am the second paragraph!

The *style="…"* allows us to add some CSS code into our line of HTML code. For *font-size:28px,* the *font-size* property controls the size of the font, the *28px* is the value in pixels (you need to say what unit you're using). The ; semi-colon marks the end of your declaration (like the period at the end of a sentence), allowing us to move onto the font *color* property with its value of *green*.

Let's save and refresh again:

So far, we've covered a lot, but before we take a break, let's add one more new element to this webpage: the Link. We've discussed <h1> and tags, now we're going to add the <*a*> tag.

The <*a*> tag is used to create a hyperlink and is almost always accompanied by the *href* attribute, which is where you place the value, which, in the case of hyperlinks, is the link:

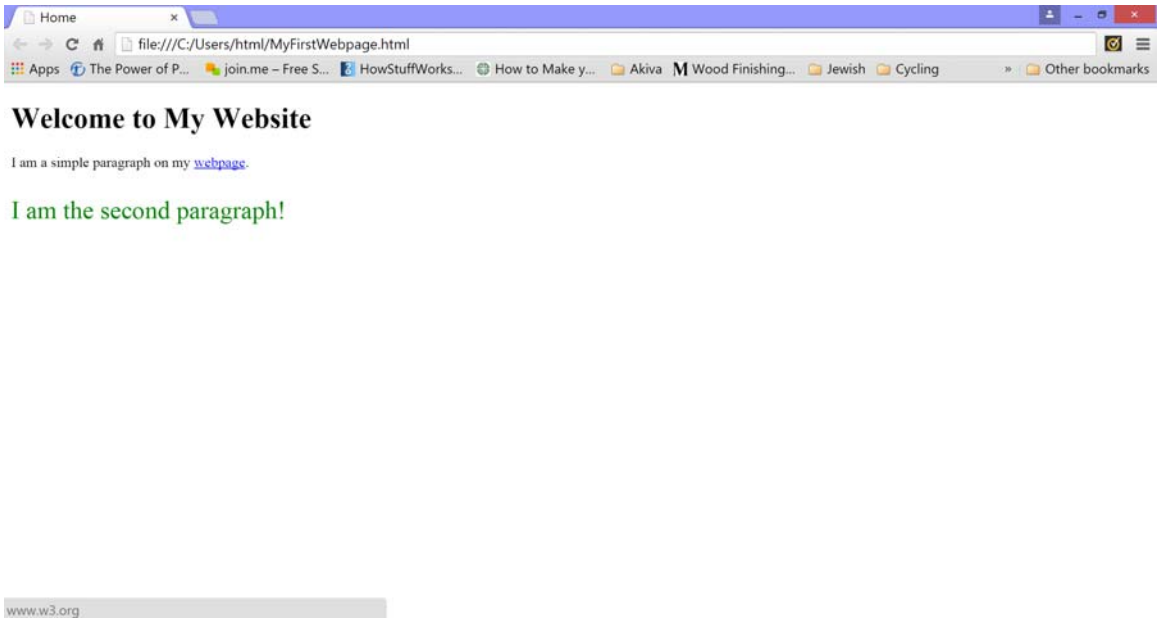
<*a href="somewebsiteURL"*>

Let's add a hyperlink in our first paragraph to the W3C:

I am a simple paragraph on my webpage.

Don't forget the closing tag for the link! Save, Alt+Tab, and refresh!

Now, try clicking your link:



If you hover over the link, you'll see its destination in the bottom left of the browser window. Try modifying the code for link to this:

```
<p>I am a simple paragraph on my
 <a href="http://www.w3.org/" target="_blank">webpage.</a></p>
```

We just added *target="_blank"* inside the opening <a> tag. (I inserted a line-break here because the whole line doesn't fit across the page; you could actually have a dozen blank lines in the middle of your code and the browser will ignore all of them – as long as it's not in the middle of a word.) Save and refresh to see what happens now when you click it.

There you go! In a matter of pages, you've:

1. created a webpage
2. created a header
3. created some simple paragraph text
4. learned and used a bit of CSS to style your text
5. added a link to your page

and

6. displayed your webpage in a browser

Congratulations! Now you have learned the basics for **all** HTML webpages. Of course, you're not done yet… Just like we discussed earlier, repetition is one of the best ways to make sure you learn something, so start repeating — type more webpages!

If you've just read through this whole chapter in one sitting, get up and take a break. Go for a run, jump up and down, climb something, get your blood flowing again. After you've rejuvenated yourself, practice some of what you just learned. Play around with it. Experiment. Make mistakes and fix them! Like it says in the Quick Tips for Learning, *Get Your Hands Dirty*.

Quick Review – HTML

Remember: repetition is key! Practice something over and over and over and over, wait until you can't bear to do it another time, then do it five more times. After that, you should have it committed to ***Long Term Memory***.

1. What is the ML in HTML and what does that mean?
2. In HTML code, how do you write the text that appears in those tabs at the top of the browser?
3. Name three common "bugs" that could occur in someone's code.
4. What are the four main tags found in almost every HTML page?
5. With few exceptions, what do you ***always*** need when you open a <tag>?
6. With few exceptions, what do you ***always*** need when you open a <tag>? (Repetition)
7. Write out **on paper** a basic HTML webpage that has a tab that says "Welcome", a header that says "My Website", and a paragraph that says "Hello, World" where the word *World* hyperlinks to a worldly website.
8. Write out **on paper** a basic HTML webpage that has a tab that says your surname, a header that says your full name, a sub-header (non-h1 header) that says the city you were born in, a sub-sub-header (non-h2 header) that says the year you were born in, and a paragraph with a description of the place you grew up in, and a second paragraph that describes you.

Feel free to take this quiz and review this chapter as many times as you'd like! I do, however, recommend continuing, since we'll end up using a lot of these concepts in the coming chapters anyway (repetition from every-day-usage).

Now that you've got HTML under control, you definitely can't stop there. It's time to work on your style — onwards to CSS!