Unplash

Managing Your Time as a Programming College Student

julia johnson Follow

Jan 2 · 9 min read

> *"Time is what we want most, but what we use worst."*
>
> *— William Penn*

It's interesting how the more successful people become, the less they value money and the more they value **time**.

In college, time is something that is paid a great deal of attention to. Whether its time until an assignment is due, or time until exam season begins, every student has an internal clock relentlessly turning, reminding them of the next thing they have to complete.

Programming students deal with this to an even greater degree. Often times classwork alone is not enough to land a coveted internship, or their courses don't teach them the modern frameworks or languages they need to know.

This causes students to spend their free time outside of classes building their own projects, or studying languages online, creating an even greater time crunch in their daily lives, but this, like many other aspects of life, can be managed with a little bit of work.

Here's how:

1. Tomorrow's Morning Starts Today

> *So get rest, get organized, and **get your sh*t together**.*

Photo by Eddi Aguirre on Unsplash

No matter what time you plan on waking up, plan accordingly the night before.

Spend a few minutes putting together your bags, printing any notes you may need, and reviewing your to-do list to make sure you completed everything for tomorrow.

How my nightly routine normally goes:

- **2 minutes** to review my email and double check my task lists
- **2 minutes** to prep one task that is absolutely vital to complete the next day. I'll often leave the tabs on my computer open that relate to this project, so I don't waste time the next day setting it all up.
- **1 minute** to put my work and laptop away

& that's pretty much it! It takes only 5 minutes to ensure that I'm not panicking the next morning trying to remember what I need to get done, putting my things together, or stressing that I forgot about an important assignment.

Small efforts like these improve your peace of mind and create a better sense of control over your work.

2. Find Your Own Way of Working

> *"Successful people wake up at 4 am"*

> *"Waking up early is the only way to be productive."*

> ***Yeah, no thanks. I'm good.***

I'm sick of these phrases being tossed around like proven facts or used as ways for natural early birds to brag about their "productiveness"

Everyones different, and everyone has their own internal clock and body rhythms.

Maybe 4 am is the best time for you to work, or maybe you're the most focused at midnight. In my experience, neither is superior or applicable to everyone. Find the time that **works for you** and plan accordingly.

I personally prefer to work at night and find that my best work is done in the late afternoon through the evening. Because of this, I schedule my classes later in the day so I don't have to wake up early and throw my sleep schedule off.

> *If 6 am or earlier isn't when you feel most productive, why are you forcing yourself to wake up then?*
>
> *Take a moment to think, am I doing this to be productive or to simply show off how productive I am?*

If that time in the morning would be better spent with you sleeping, don't feel guilty for using it that way.

On the other hand, forcing yourself to stay awake as late as possible won't help you in the long run either.

3. Make Learning Your First Priority in the Morning

> *"Continuous learning is the minimum requirement for success in any field"*
>
> *-Brian Tracy*

Photo by Nikita Kachanovsky on Unsplash

I recommend starting your day off with at least 30 minutes of an online course or studying through your newest programming language.

This helps you prioritize knowledge in your routine, which is an extremely vital part of becoming a talented young programmer.

If you put this off until the end of the day, chances are you could get wrapped up in an assignment or project, and never get a chance to learn what you wanted to. Starting it off first thing ensures that you'll always have time for it, and this gets you in the routine of learning something new every day.

What you spend this time doing will be completely unique to you, but here's how it works for me:

> *Pick 1 or 2 realistic thing(s) I can learn in the time I have available*

This can be a youtube video on a topic in a new language you're interested in, or finishing two lectures of the Udemy course you're currently taking.

I normally start my day with 20 minutes of my Udemy lectures and then 10 minutes of review and practice. If I have more time, I'll get more (40 or 50 minutes) of my lecture done and spend more time on my review.

This time setup is not for someone who has massive amounts of free time to code daily, its for people who are balancing coding & school and want to find the best way to incorporate coding into their daily lives.

4. Learn, Create, Repeat

> *There's **no magic spell** that makes watching lectures or reading someone else's code give you the ability to program perfectly. (but that would be awesome)*

Many young coders get frustrated and discouraged when they follow lectures and get stuck and feel the course didn't properly teach them how to handle specific problems. My guess is that most of these students didn't spend a proper amount of time implementing their own code based on the principles they just learned.

> *No matter how many lectures you watch or tutorials you complete, copying code or learning from a professor will only take you so far.*

And after an entire course of this, you'll find you're no better off than you were when the course began.

This is why it is so **incredibly important** to spend time creating projects, no matter how small, that allow you to practice and write your own code on the things you just learned to concrete your understanding.

I personally spend time creating at least 3 or 4 simple projects during a course completely on my own, and then at the end of the course, I do one larger project that reestablishes my comprehension on the language or framework as a whole.

I can't push the importance of this enough, I completed a course a few years ago in CSS and realized that I wasn't truly comfortable with CSS until after I *had* to use it for a project a few weeks later. Proving again, that constant implementation of a language helps you struggle through problems, and **nothing makes you learn a coding language faster than grappling with problems yourself.**

This may not even seem like a time management solution, but I truly realized how much time I had wasted sitting back and watching a lecturer, when I really should have just **started building something.** Don't waste more time than you have to listening, when you could already be building something awesome. **You've learned more than you think, go for it.**

5. Learn to organize coding tasks together, based on the degree of focus they require

Transitioning between multiple different tasks or implementations for your site or project is one of the many difficulties of coding time management.

I often find myself super focused and working on the backend code for a site, when I realize I need to stop and style a table or find the perfect

image to fit the site. This forces me to stop what I'm currently concentrated on and totally shift my focus to a styling conflict or googling the right image.

This is a **major** time waster and is where your (and even my) lack of organization is coming back to bite you.

The constant interruptions from deep logic based coding are probably affecting your productivity more than you realize, luckily planning ahead and creating stages to your development will help you a lot.

Say you were creating a web page that displayed a user's purchases for the past month. Normally, you'd start by adding a <table> then connecting its output to an external database, and add the CSS as you work through it.

On a small scale project this isn't a big deal, but when working with larger ones the stop and go and back and forth of adding front and back-end code can make the process less seamless and takes a lot more time than it should.

> *This is how this process should go:*

1. Draw or create an extremely simple mockup of what you're creating so you can visualize what needs to be done before you've actually started doing it. This is an extremely crucial step that really should only take a couple of minutes.
2. Next, collect the images and set up any external connections so they're ready to be connected and added to the site. If you don't currently have any images or data, create fake filler images or data so that your page won't display an empty table or white space.
3. Write the blank starter code (in this case a table) and leave it with its basic styling.
4. Now make it work, make sure the database is connecting properly and your backend code is acting exactly as you want it to.
5. Finally, move onto styling, which should be much easier now that everything is working and created.

This order saves much time from going back and forth and will give you a clear understanding of what needs to get done right now.

6. When to Struggle and When to Ask for Help

> *"But did you Google it first?"—Every Coder Since the Beginning of Time*

You're coding, it's going awesome, your project looks great, and now you're stuck. Happens to all of us, but it can leave new coders frustrated and forcing them to abandon projects. Getting stuck can be a huge time waster, which is why I've included these tips in this post.

So how do you know when you should struggle through a project and when you shouldn't?

To start, I'm going to give you three ways to try and discover a solution to your problem. These are for web developers mainly, as these are little tricks I've picked up while I've worked.

> ***1. Echo Echo Echo***

When I was picking up PHP & SQL echo was my best friend. Often in backend coding, a huge difficulty is understanding where your project is going wrong. If you aren't using echo or some sort of display output line of code, you're basically coding blind, and are going to have extreme difficulties finding the solution.

There are so many times when echoing out the result of my for loop or any equivalent line of code saved me from complete frustration, and in turn, saved me a lot of time.

Here's how to output your code in a few languages:

```
Javascript:

console.log(variable here);
```

```
PHP:
```

```
echo $variable;
```

Python:

```
print(variable)
```

> ***2. Add Comments***

Adding comments is another great way to see where your code is failing and is a quick solution to many problems. I know many people often use this as a way to remove broken code and add a new possible solution without deleting it.

If you're using VSCode a quick way to add comments is to hold CMD + K + C on mac and CTRL + K + C on windows. This will add a comment to the line you're currently on or whatever is currently highlighted and has saved me a ton of time.

> ***3. Just Google it***

Doesn't require much explanation, if you're stuck and think there might be an easy solution, just look it up. Spend some time actually reading through online resources and comparing their problems to yours and see if you're making any of the mistakes another person is.

Chances are you'll find exactly what you're looking for, as whatever you're struggling with has probably be struggled with before (and will be again). So before you really fret, spend some time doing the research.

. . .

So you've output the code, you've googled it, and nothing seems to make sense and no one else seems to have this problem. If possible, move away from that problem for at least 25+ minutes. Transition to something else, or simply take a break.

There have been handfuls of times a perplexing coding problem seemed impossible until I came back to it later and the easiest possible

answer jumped out to me and made sense. This feeling is *amazing* and is one of the reasons I love coding.

> *Nothing will make you feel more like a real programmer than struggling and getting frustrated, but still solving things on your own.*

If you come back and still don't understand what's going on, which has happened to me as well, repeat the process of testing outputs, review your code to make sure there aren't simple mistakes, and then and **only then** reach out to someone for help.

. . .

I hope these tips were helpful to you!

Give it a clap (or a few more) if you enjoyed it!

— Julia Johnson (@juliacodes) juliacodes.com