Learnathon Learning Tips & Tricks

As you head into these days of focused upskilling, we have seven pieces of advice to help you make the most out of your learning time. Read on to unlock your learning potential!

✔Create learning objectives that are **specific**, **authentic**, **actionable**, and **demonstrable**.

If you've chosen to learn more about GitHub Copilot, you won't get too far in any one direction if your learning objective is simply:

> *Learn more about GitHub Copilot.*

There's nothing *specific*, *authentic*, *actionable* or *demonstrable* about that objective.

Instead, try something like this:

> *Effectively and efficiently add new functionality to an existing codebase using Github Copilot.*

This learning objective describes what you should be able to do as a result of your learning (hence, they start with a verb), and its specificity will help guide the content you seek for your learning.

This work is licensed under Attribution-ShareAlike 4.0 International

It's also specific in the sense that it describes a practical, hands-on project you can build in satisfaction of the objective. This won't always be the case with a learning objective, nor is it always necessary.

Let's look at a more robust example. Say you're an experienced Python developer, but need to upskill around Python's **`asyncio`** module. Here are a set of learning objectives you might create around that larger goal that don't inherently describe a project or activity:

- *Explain the core concepts of asynchronous programming, such as event loops, coroutines, and non-blocking I/O, including why they are essential for efficient and responsive applications.*
- *Write Python code using async and await keywords to create asynchronous functions and coroutines.*
- *Implement and comprehend the difference between* **`asyncio.sleep`**, *awaitable objects, and* **`asyncio.gather`** *to manage concurrency.*

If you want to get really serious about your learning objectives, check out Bloom's Taxonomy. But one word of caution here: Don't get too lost in the weeds by trying to create perfect learning objectives. It's a skill that has to be developed, and we'd much rather you spend your time *upskilling* than becoming ninjas at … writing learning objectives 😆

✔️ Be **ambitious** in your objectives, but don't be **inflexible**.

You'll want to make sure that your learning objectives are challenging and ambitious enough that you don't get bored, or achieve them after just a couple hours of work. Better to not quite reach your learning objectives, than to achieve them too quickly.

And truly, don't denigrate yourself if you don't achieve every single one of those objectives in two days. Learning unfolds in unpredictable ways, and it's okay to adjust and adapt your learning path as you go.

✔️Be clear about your **motivations** for pursuing the particular learning objectives you've chosen.

There is so much research that emphasizes the importance of understanding why you're learning something. Seriously: Just open up Google Scholar, type in *learning* and *motivation*, and browse the titles. This research indicates that knowing your motivations for learning is crucial for staying committed and overcoming the inevitable challenges that come with the learning process.

So before jumping into learning, review the motivations behind why you're pursuing each of your learning objectives. Are your motivations strong enough? Are they valid? Are they intrinsic? Write down your motivations and keep them close during the **Learnathon** so that when things get tough, or you get bored, or you start to ask yourself *Why am I even doing this?*, you have a ready reminder of why you are even doing this 😀

✔️Don't just **consume content**. (And please don't consume 8 hours of content in a single day 🙀)

We've heard so many people say *I just don't learn from watching four hours straight of videos*.

Well, yeah. Nobody does.

Please don't sit down and watch four hours straight of videos. Watch seven minutes and then take a note, or stand up and have a think about what you just watched, or open up your IDE and play a little bit. To learn new knowledge, you have to give things time to sink in, and to learn new skills, you have to practice. And in the space of software development and engineering, you have to *build things*.

✔️ **Articulate** (frequently) what you're learning.

There's a lot of research that shows that we remember things better when we *produce*, whether that production occurs through written or spoken language and code. It's also a common refrain that we really *learn* something when we have to teach it.

So, focus on your production skills as you move through your learning. Record your newfound knowledge through creating personal (or – even better – *shareable*) documentation, note cards, code snippets – whatever works for you. Practice articulating your knowledge every couple hours through sharing in your group Slack channel what has been most interesting or impactful to you. Bonus points if you share via video!

✔️ Engage with your **learning community**.

Learning is a social endeavor, just like writing code and building software is a social endeavor. And learning as part of a *learning community* offers numerous benefits that can enhance the overall learning experience.

For example, in a learning community, you have a built-in support system. Your peers can provide encouragement, answer questions, and help you stay motivated.

Learning communities also often consist of individuals from various backgrounds who have differing levels of experience. This diversity can bring different viewpoints to discussions and problem-solving, enriching your understanding of the subject matter.

Building relationships within a learning community can lead to valuable connections with your colleagues. You may work with people who you've never had a chance to work with before, which can open up the possibility of future collaboration and mentorship.

Being part of a learning community can also foster a sense of accountability. When others are aware of your learning goals, you may be more likely to stick to them.

Finally, receiving constructive criticism on your own work and reviewing the work of others can help you improve your newfound skills and knowledge.

✔️Don't **multitask**. This is a time for **focused upskilling**.

Listen. Everyone is here for it: your leaders, your teammates. We're all in agreement that for these several days, we're just taking some time to do focused learning together.

So, set aside your sprint work, pause your Slack notifications, ignore your on-call rotation messages (Just kidding! Don't do that...talk to your leader!), and just let the company pay you to learn for a few days, okay?

Authorship: This Toolkit was created by the Developer Success Lab, a team of scientists & software developers that conducts empirical research that aims to both share developers' experiences, and provide models for human-centered software development. The research behind this Toolkit is from **The New Developer** project.

Our public-facing work on developer experience thrives when people share and support it!

Help to support our work by sharing our research, insights, and toolkits, available at **DevSuccessLab.com**.

Have thanks or impact to share? Give us a shout-out on social media, or drop us a line at **flow-research@pluralsight.com**