Why Every Writer Needs a Designated Domain in Documentation

When you write technical documentation, examples, or tutorials, you often need to show a web address. It might be a link to a configuration file, a sample API endpoint, or a placeholder for a future service. The problem is that many real domains are already registered and serving content. Using them in a doc example can confuse readers, cause broken links, or even lead to legal issues. That is where the concept of a **domain in documentation** becomes essential.

A domain in documentation is a reserved, non-functional web address used solely as an example. It is not a live service. It does not host a website or an API. Its only job is to sit inside code blocks, configuration files, and explanatory text as a stand-in. The most well-known example is example.com, which the Internet Assigned Numbers Authority (IANA) set aside specifically for this purpose. But the idea goes beyond just one domain. It is a practice that every technical writer, developer, and trainer should understand and adopt.

Why Real Domains Cause Real Problems

Imagine writing a tutorial for a new authentication library. You need to show a callback URL. If you write https://myapp.com/auth/callback, a few things can go wrong. First, myapp.com might actually exist. Someone could have registered it. If a reader types that URL into a browser, they might land on a stranger's website, not a helpful page. Second, if the domain later gets taken over by a malicious actor, your documentation could become a vector for phishing. Third, the example becomes untestable. A reader trying to follow along cannot deploy code that points to a real domain they do not control.

These are not hypothetical concerns. There have been incidents where popular documentation used unregistered domains that later expired and were bought by advertisers or scammers. The result was that anyone following the old instructions would be redirected to unrelated or harmful sites. Using a dedicated domain in documentation prevents all of this. It provides a stable, neutral placeholder that will never change hands or serve unintended content.

How Reserved Domains Work

IANA and the Internet Engineering Task Force (IETF) have set aside several top-level domains and specific second-level domains for documentation and testing. The most common ones are:

- example.com, example.net, example.org, These are reserved by IANA for use in examples.

- .example, A top-level domain reserved specifically for documentation and testing.
- test, Often used in local or internal documentation, though it is not officially reserved at the global level.

When you use one of these reserved domains, you are following a standard that has been in place since the early days of the internet. The domain in documentation acts as a signal to the reader: this is not a real address. Do not try to visit it. It is a teaching tool, not a service. This clarity improves the reader's experience and protects the integrity of your documentation.

Subdomain and Path Considerations

It is not enough to just use example.com. You also need to think about the full URL. For instance, https://api.example.com/v1/users is a valid example. The subdomain api.example.com is not separately reserved, but because it is a subdomain of a reserved domain, it carries the same guarantee. No one can register it. Similarly, you can use any path or query string under example.com and know it will remain a safe placeholder.

One common mistake is to use a domain that looks reserved but is not. For example, mycompany.com is not reserved. Even if your company owns it, using it in public documentation can confuse outsiders and create maintenance burdens if the domain changes. Stick to the officially reserved set.

Practical Guidelines for Writers

Using a domain in documentation is simple, but it requires discipline. Here are a few rules of thumb that I have seen work well in real projects:

1. **Always use a reserved domain.** Before you type any web address into a doc, ask yourself if it is an example. If yes, use example.com or a subdomain of it. Do not use a domain you own, a domain you wish you owned, or a domain that sounds plausible but is unregistered.
2. **Be consistent across your entire documentation set.** If you use example.com in one chapter, do not switch to test.example in another without a good reason. Consistency reduces cognitive load for readers.
3. **Include a brief note in your style guide.** Tell contributors why a domain in documentation is important and which domains are acceptable. This prevents well-meaning but incorrect contributions.
4. **Test your examples.** Even though the domain itself will not resolve, you can still test the structure of the URL or the code logic. Make sure the example is syntactically correct and logically sound.

These guidelines apply whether you are writing a README file, a full reference manual, or a blog post about a technology. The principle is universal: keep examples safe, predictable, and clearly marked as examples.

Common Misconceptions

Some writers resist using a domain in documentation because they think it looks unprofessional or fake. They worry that readers will not take the example seriously if the domain is obviously a placeholder. In my experience, the opposite is true. Readers appreciate the clarity. When they see example.com, they immediately know the URL is not meant to be clicked. It signals that the writer has thought about the reader's experience. It also shows that the writer understands internet standards.

Another misconception is that using a reserved domain somehow limits the realism of the example. But you can add as much detail as you want after the domain. For example, https://example.com/very/long/path/with/realistic/parameters?user=123 is both realistic and safe. The domain itself is the only part that matters for reservation status.

The Bigger Picture

Adopting a domain in documentation is part of a larger mindset: writing for the reader's safety and understanding. It is similar to using placeholder text like your-api-key-here instead of an actual key, or using user@example.com instead of a real email address. These practices protect both the reader and the writer. They make documentation more portable, more durable, and more trustworthy.

If you are maintaining a large documentation set, consider auditing your existing examples. Search for any domain that is not in the reserved set. Replace those with example.com or another approved domain. This is a small change that can prevent big headaches later. It also brings your documentation into compliance with best practices that have been recommended for decades.

In short, the humble domain in documentation is a small but powerful tool. It sits quietly in your code samples and configuration files, doing nothing except being a placeholder. But that nothing is exactly what makes it so valuable. It guarantees that your examples remain examples, and that your readers stay safe. Next time you write a tutorial or a reference guide, take a moment to choose your domain carefully. It is a sign of professional craftsmanship.