The 12 Modules of (ForgeBox) Christmas — Day 3 — str (https://www.ortussolutions.com/blog/the-12-modules-of-forgebox-christmas-day-3-str)


Eric Peterson

Dec 16, 2017

Modules don't have to be big. In fact, the [Unix philosophy](https://en.wikipedia.org/wiki/Unix_philosophy) says to do one thing and do it well. While modules on ForgeBox run the gamut of [doing one thing](https://www.forgebox.io/view/orm-reload-interceptor) and [full blown frameworks](https://www.forgebox.io/view/coldbox), today we will look at one of the utility libraries on ForgeBox — a string manipulation library called [`str`](https://www.forgebox.io/view/str).

str (https://www.forgebox.io/view/str)

Rather than walk you through the entire API, let me share with you some of the methods from this library that I think will be interesting to you.

`capitalize` & `capitalizeWords`

```
str.capitalize( "my post title" );
// "My post title"
```

Super simple function but annoying to write over and over again. Glad I don't have to write myself it again anymore.

```
str.capitalizeWords( "my post title" );
// "My Post Title"
```

I've always been jealous of the built in `ucwords` in PHP. Well, this method makes that jealousy obsolete.

`startsWith`

```
str.startsWith( "getDynamicMethod", "get" );
// true

str.startsWith( "setDynamicMethod", "get" );
// false
```

Even more simple than the last one, it's implemented in the source code as follows: `return left( word, len( substring ) ) == substring;`. But this is such an improvement in readability!

`slug`, `snake`, and `kebab`

```
str.slug( "My Post Title", "-" );
// "my-post-title"
```

`slug` is actually the method that powers a whole slew of case transformations in str such as `snake` and `kebab`.

`plural`

```
str.plural( "entity" );
// "entities"

str.plural( "tree", 1 );
// "tree"
```

I saved the best for last here. This will handle the edge cases around pluralizing nouns for you. You're welcome.

Benefits of a shared utility library

Before we end, let's address a common concern I've heard:

> I could write all those utilities myself! Why rely on a third party library for it?

Let me give you three reasons:

1. Well tested

   A shared utility library is likely to have a good test suite. This isn't to say that your utilities aren't tested. But by having the library used by more people in more situations, it is more battle-tested and ready for your *next* project.

2. One command installs and updates

   When you need your own hand-written utility, you go find the repo or find the code in another project and copy and paste it over. Conversely, you could spend a few seconds typing an install slug in to CommandBox and be done. Additionally, when there are updates to the library, whether new features or bug fixes, all of the places that use the shared library can be updated via CommandBox. The alternative? A frantic find and replace.

3. Familiarity between developers

This point will vary from project to project, but it's more likely that another developer is familiar with a shared utility library than your own hand written library. If you need proof of this, think of Underscore or Lodash in JavaScript. This familiarity can range from being familiar with the code to being familar with the concepts and names. Both are valuable.

Wrap Up

`str` is one of many utility libraries on ForgeBox. Next time you need a utility, first check on ForgeBox. You might find you don't need to write it yourself. You may find a library that could use a pull request from you. Or you may find an opportunity to share your own utlities with the rest of us.