FFI in PHP 7.4 and beyond

@chrisnharvey

Who am I?

- Chris Harvey
- I like using PHP for things PHP shouldn’t be used for

How did we get here?

- PHP was first released in 1995
- Was designed as a templating language to sit between C code and HTML
- The idea was that you'd write the backend code in C, and PHP would act as a way of exposing data in HTML

What is FFI?

- The Foreign Function Interface
- Allows you to run C code or any library that has a C compatible ABI inside PHP natively
- You can use libraries from C, C++, Rust and others inside your PHP code
- Available in PHP core since 7.4
- Also available as a pecl extension for some older versions

What is so good about that?

- There are tons of high-quality C, C++ and Rust libraries out there
- Can remove the need to use symfony/process, shell_exec or passthru to call an external binary and parse its response.
- Easier to test.
- Can provide a performance improvement in some cases.
- Can essentially provide PHP extensions via composer without the need to compile them as C extensions
- Can enable all kinds of crazy possibilities in PHP

What kinds of crazy?

- Machine Learning with Tensorflow
- Access to hardware
- Image processing
- Video transcoding
- Browser automation
- 2D/3D rendering
- Desktop applications

How it works

- Enabled by default in the CLI SAPI
- Also available in files loaded via OPcache preloading
- Can be enabled everywhere with ffi.enable=true in php.ini

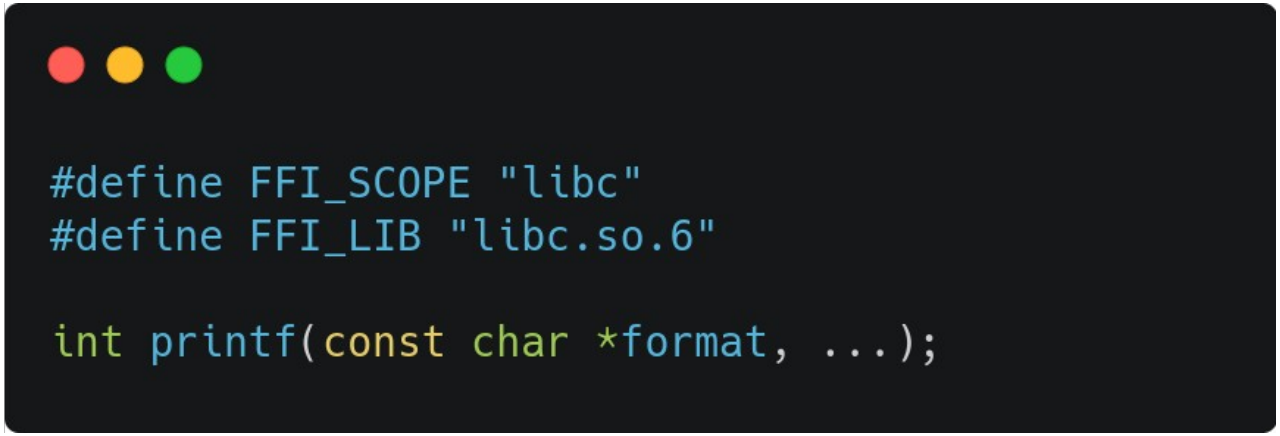
How it works

```php
<?php

$ffi = FFI::cdef(
    "int printf(const char *format, ...);",
    "libc.so.6");

$ffi->printf("%s from C!\n", "Hello");
```

How it works



```
#define FFI_SCOPE "libc"
#define FFI_LIB "libc.so.6"

int printf(const char *format, ...);
```

How it works

```php
<?php

$ffi = FFI::load('libc.h');

$ffi->printf("%s from C!\n", "Hello");
```

Demo

FFI is cool, but what about real-world applications?

- Tensorflow
- Audio/Video/Image manipulation
- HTML to PDF converter

Here be dragons

- PHP manages memory for you, but in C, you're on your own.
- PHP will try manage FFI memory for you, but this is not always possible.
- Pointers returned from C function calls will not be freed automatically.
- When you're done with a pointer, call FFI::free($pointer)
- C libraries will generally have functions to free resources (e.g. curl_close($ch)).
- Try to keep the FFI logic of your app/library self contained (don't return pointers to consumers of your library)

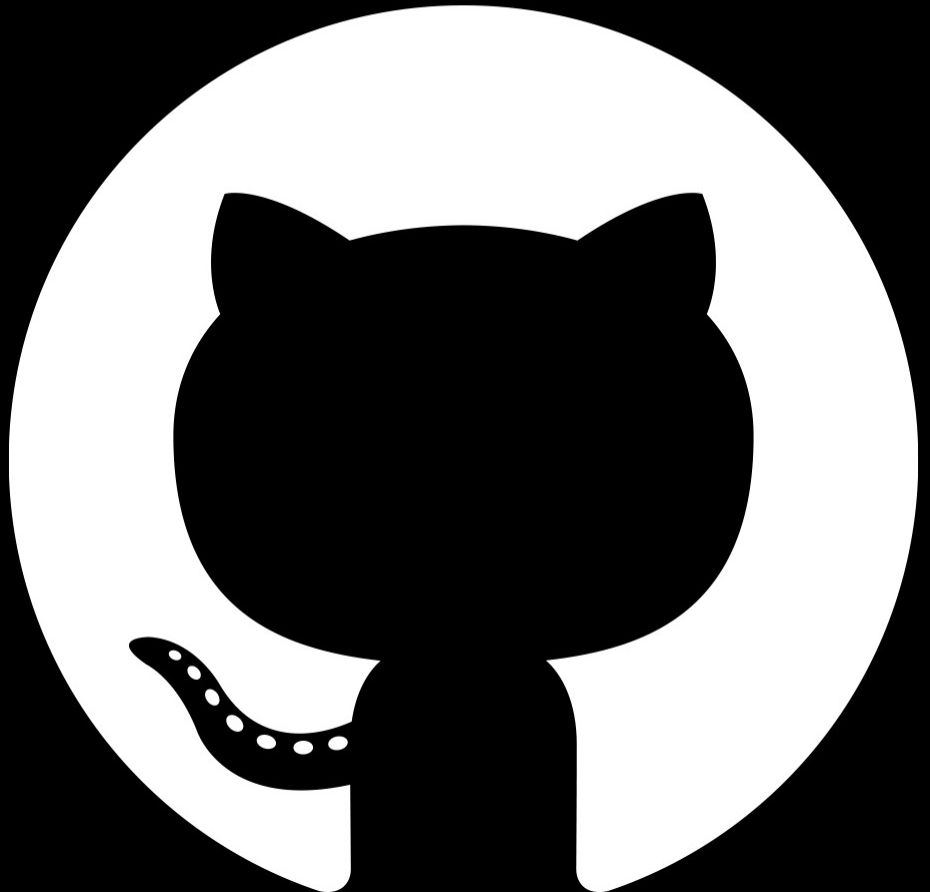
Demo

Considerations

- To use in web requests, use OPcache preloading
- You will need library binaries for each platform you intend to support (Linux, macOS, Windows)
- Be careful.

Give it a try

- chrisnharvey/pdfkit
- gabrielrcouto/awesome-php-ffi
- darkin1/PhpShooter



Thank you

Chris Harvey

@chrisnharvey

chrisnharvey.com