SINGLE FILE RUBY PROGRAMS

CHRISTIAN BÄUERLEIN

Created: 2020-03-26 Thu 18:58

WELCOME!

THE LOST ART OF SINGLE FILE RUBY PROGRAMS

A loose collection of Ruby fun facts and examples to organize your code in a single file.

Let's have some fun with Ruby!

CODE & TESTS IN ONE FILE

RUBY MAGIC CONSTANTS

There is `~$0`.

> *Contains the name of the file containing the Ruby script being executed.*

Source: Pre-defined variables and constants

THE SOURCE FILE

```
def greet(name)
  "Hello #{name}!"
end

# this will only run if the script was the main
# not load'd or require'd
if __FILE__ == $0
  require "test/unit/assertions"
  include Test::Unit::Assertions

  assert_equal 'Hello Ruby', greet('Ruby'), "greet function shoul
end
```

WHEN CALLED DIRECTLY

```
$ ruby code_and_test.rb
 greet function should return 'Hello Ruby!'. (Test::Unit::Asserti
<"Hello Ruby"> expected but was
<"Hello Ruby!">.

diff:
- Hello Ruby
+ Hello Ruby!
?
```

WHEN REQUIRED

```
require './code_and_test.rb'

puts greet "Christian"
```

```
$ ruby code_and_test_usage.rb
Hello Christian!
```

YOU MAY KNOW THIS FROM PYTHON

```
def hello():
    print("Hello world, this script was called!")

if __name__ == '__main__':
    hello()
```

Source:

Python Define Unit Test Classes Together with Code

A WEBSERVER IN ONE FILE?

MY LIFE BEFORE RUBY

Before Ruby, my favorite web framework was CodeIgniter.

Security first: One index.html file in **EVERY** folder.

SINATRA 0.6

```
sudo gem install sinatra
```

```
require "rubygems"
require "sinatra"

get '/' do
 "Hello World"
end
```

Awesome!!!

```
ruby myapp.rb
```

TEMPLATES: UNCOOL WAY

"You can do this too but it's not as cool" - Sinatra Readme

```
template :index do
  '%div.title Hello World!'
end
```

TEMPLATES

As documented in the README.rdoc this was the cool way to do it.

```
get '/' do
  haml :index
end

use_in_file_templates!

__END__

@@ layout
X
= yield
X

@@ index
%div.title Hello world!!!!!
```

A LITTLE RUBY HISTORY

RUBY IS A BETTER PERL

Why the name 'Ruby'?

> *Influenced by Perl, Matz wanted to use a jewel name for his new language, so he named Ruby after a colleague's birthstone.*

Source: The Ruby Language FAQ

PERL'S LEGACY

Ruby took a lot of things from Pearl.

Today we will learn about:

- Keywords
- Command line flags

LET'S START WITH PERLDATA

Perl has two special literals:

- `__END__` - Indicates the logical end of the script before the actual end of file. Any following text is ignored.
- `__DATA__` - A filehandle that points to everything that comes after `__END__`.

Source: perldata - perldoc.perl.org

THE __END__ AND DATA KEYWORDS

Denotes the end of the regular source code section of a program file. Lines below `__END__` will not be executed.

Those lines will be available via the special filehandle `DATA`

Source: Class: Object

SIMPLE EXAMPLE

```
DATA.each_line do |line|
  puts line
end

__END__
Doom
Quake
Diablo
```

ERB TEMPLATE AND CODE IN ONE FILE

```
require 'erb'

time = Time.now
renderer = ERB.new(DATA.read)
puts renderer.result()

__END__
The current time is <%= time %>.
```

EXPLAINED: SINATRA STYLE MULTIPLE TEMPLATES IN FILE

```
get '/' do
  haml :index
end

use_in_file_templates!

__END__

@@ layout
X
= yield
X

@@ index
%div.title Hello world!!!!!
```

```
File.read(caller.first.split(":").first).split("__END__", 2).last
```

Source: Mixing code and data in Ruby

PSA: PHP CAN DO IT AS WELL

```
// open self
$fp = fopen(__FILE__, 'rb');
// seek file pointer to data
//__COMPILER_HALT_OFFSET__ will return
//the point after __halt_compiler();
fseek($fp, __COMPILER_HALT_OFFSET__);
// and output it
$unpacked = gzinflate(stream_get_contents($fp));
__halt_compiler();
//now here... all the binary gzdeflate already items!!!
```

Source: PHP: __halt_compiler - Manual Example: __halt_compiler(), make install files for PHP smaller

BUNDLER INLINE

Fun fact: You don't need a `Gemfile` to use bundler!

Useful for scripts in your `/utils` folder that you only use once a year.

Source: How to use Bundler in a single-file Ruby script

INLINE HTTPARTY

```
require 'bundler/inline'

gemfile do
  gem 'httparty'
end

puts HTTParty.get('https://www.boredapi.com/api/activity')
```

INLINE MINITEST

```
require 'bundler/inline'

gemfile do
  gem 'minitest', require: false
end

require 'minitest/autorun'

class MyTest < Minitest::Test
  def test_should_be_true
    assert_equal true, true
  end
end
```

ADVANCED EXAMPLE: DOWNLOAD ICAL TO ORG

Source: ical_to_org.rb

ADVANCED EXAMPLE: DOWNLOAD ICAL TO ORG

- Install Dependencies

Source: ical_to_org.rb

ADVANCED EXAMPLE: DOWNLOAD ICAL TO ORG

- Install Dependencies
- Do stuff (download calendar events)

Source: ical_to_org.rb

ADVANCED EXAMPLE: DOWNLOAD ICAL TO ORG

- Install Dependencies
- Do stuff (download calendar events)
- Render to ERb template

Source: ical_to_org.rb

BEGIN AND END KEYWORDS

Yes, this is taken from Perl as well.

DEFINITION

BEGIN defines a block that is run before any other code in the current file. It is typically used in one-liners with ruby -e.

Similarly END defines a block that is run after any other code.

Source: Documentation for Ruby 2.2.0

EXAMPLE

```
END { puts 3 }
BEGIN { puts 1 }
puts 2
```

```
ruby begin.rb
1
2
3
```

INTRODUCING LRUBY

Logging Ruby - The Ruby alias for the forgetful scripter Logging Ruby!

Only Feature: No more scrolling through your terminal... Logs the output of a script to the script itself!

LET'S TRY THIS OUT

```
cat log_results/hello_world.rb
```

```
ruby log_results/hello_world.rb
```

INTRODUCING: LRUBY

```
lruby log_results/hello_world.rb
```

```
cat log_results/hello_world.rb
```

HOW DOES IT WORK?

```
which lruby
```

Let's check out the source of LRuby log_results.rb

THE GARBAGE FLAG

Aaaaand back to Perl!

PERLRUN

```
perl -x
```

Leading garbage will be discarded until the first line that starts with #! and contains the string "perl".

Source: perlrun - perldoc.perl.org

BUT... WHY?

*Tells Perl that the program is embedded in a larger chunk of unrelated text, such as in a **mail message**.*

AND IN RUBY..

```
ruby -x
```

Tells Ruby that the script is embedded in a message. Leading garbage will be discarded until the first that starts with "#!" and contains string "ruby".

Source: Ruby Docs Command line Options

EXAMPLE

```
Hello dear friend,
this is a mail message. Please execute it with your ruby interpre

Thanks,
a random stranger
#! hahaha this is ruby now
puts "Hello World"
```

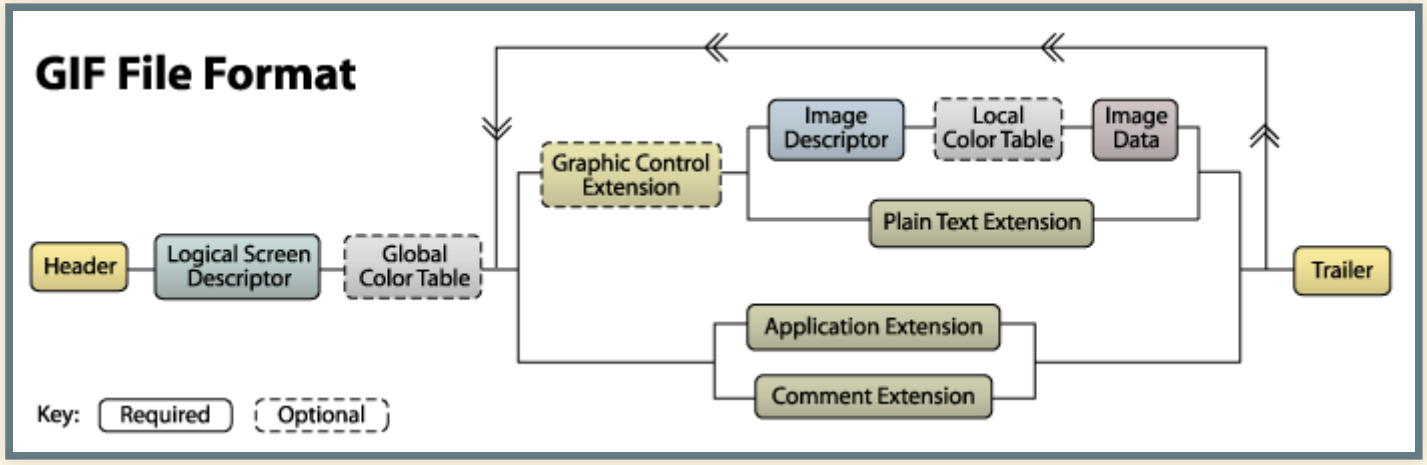
```
ruby -x email.eml
```

A SELF-ANIMATING GIF

This is not an animated gif, but a gif that animates itself.

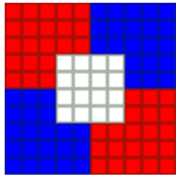
LET'S TALK ABOUT GIFS

A GIF FILE CONSISTS OF BLOCKS



EXAMPLE

Enlarged



(100x100)

Bytes

```
47 49 46 38 39 61 0A 00 0A 00 91 00 00 FF FF FF FF 00 00 00 00 FF 00 00
00 21 F9 04 00 00 00 00 00 2C 00 00 00 00 0A 00 0A 00 00 02 16 8C 2D 99
87 2A 1C DC 33 A0 02 75 EC 95 FA A8 DE 60 8C 04 91 4C 01 00 3B
```

TERMINATOR BYTE

The trailer block indicates when you've reached the end of the file. It is always a byte with a value of `3B`.

Source: What's In A GIF

SOOOO NOW WE KNOW THAT..

SOOOO NOW WE KNOW THAT..

- GIFs are nice

SOOOO NOW WE KNOW THAT..

- GIFs are nice
- GIFs always end with the same terminator byte

SOOOO NOW WE KNOW THAT..

- GIFs are nice
- GIFs always end with the same terminator byte
- Ruby is nice

SOOOO NOW WE KNOW THAT..

- GIFs are nice
- GIFs always end with the same terminator byte
- Ruby is nice
- Ruby can start with a defined start line

SOOOO NOW WE KNOW THAT..

- GIFs are nice
- GIFs always end with the same terminator byte
- Ruby is nice
- Ruby can start with a defined start line
- Nice.

A SELF-ANIMATING GIF

This is not an animated gif, but a gif that animates itself.

A SELF-ANIMATING GIF

This is not an animated gif, but a gif that animates itself.

- One file

A SELF-ANIMATING GIF

This is not an animated gif, but a gif that animates itself.

- One file
- Upper half is a GIF

A SELF-ANIMATING GIF

This is not an animated gif, but a gif that animates itself.

- One file
- Upper half is a GIF
- Lower half is Ruby code

A SELF-ANIMATING GIF

This is not an animated gif, but a gif that animates itself.

- One file
- Upper half is a GIF
- Lower half is Ruby code
- File rewrites itself!

A SELF-ANIMATING GIF

This is not an animated gif, but a gif that animates itself.

- One file
- Upper half is a GIF
- Lower half is Ruby code
- File rewrites itself!
- Profit!

SOURCE CODE

Let's check out the rbgif.gif source code together!

LIVE DEMO!

```
while 1; do cd ~/Dropbox/slides/single-file-ruby-programs/rbgif;
```

ONE MORE THING...

```
#!/bin/sh

echo This is bash
i=12
echo $i

perl - $i <<'__HERE__'
my $i = shift;
print "This is perl\n";
print ++$i . "\n";
__HERE__

echo This is bash again
echo $i
```

Source: perl script inside a shell script

PERLMONKS.ORG 2000 TESTIMONIALS

PERLMONKS.ORG 2000 TESTIMONIALS

- "Ain't that fun?" - dchetlin

PERLMONKS.ORG 2000 TESTIMONIALS

- "Ain't that fun?" - dchetlin
- "It's strange and terrible and I'm not sure how to get something out of the perl part, but this sort of works" - eg

PERLMONKS.ORG 2000 TESTIMONIALS

- "Ain't that fun?" - dchetlin
- "It's strange and terrible and I'm not sure how to get something out of the perl part, but this sort of works" - eg
- "This, on the other hand is just ... just .. well, I don't know. Not right. Not even wrong. It just is." - Blue

SUMMARY

RUBY IS FUN!

SINGLE FILE RUBY PROGRAMS

- Code & Tests
- Dependencies & Code
- Data & Code
- Code & Data
- Code & Output

Try it out for fun and profit!

THANKS!

Questions?

Christian Bäuerlein @fabrik42