

~~RTFM~~

Write A Better FM



Rich Bowen - rbowen@apache.org



- PHP's docs are pretty much the best available in Open Source today
- I'm (sort of) on the PHP docs team, by the kindness and generosity of certain people.

Why are the PHP docs awesome?

- Organized docs team outside of the dev team
- <https://edit.php.net/>
- Smart decisions about formats
- Welcoming of contributions and criticisms

Home global.ent funcindex.xml manual.xml.in constants.xml

Main menu

Number of failures to meet 'strict standards' (1065)

All files

Search:

- LICENSE
- manual.xml
- manual.xml.in
- quickref.txt
- README
- test.txt
- TODO
- version.xml
- en
 - appendices
 - chapters
 - chapters1
 - chmonly
 - docweb
 - faq
 - features
 - install
 - internals2
 - language
 - reference
 - amqp
 - apache
 - functions
 - book.xml
 - constants.xml
 - entities.functions.xml
 - ini.xml
 - reference.xml
 - setup.xml
 - versions.xml
 - apc
 - apd

File: en/reference/apache/constants.xml

Tools

1 <?xml version="1.0" encoding="utf-8"?>
 2 <!-- \$Revision: 288721 \$ -->
 3
 4 <appendix xml:id="apache.constants" xmlns="http://docbook.org/ns/docbook" xmlns:x
 5 &reftitle.constants;
 6 &no.constants;
 7 </appendix>
 8
 9 <!-- Keep this comment at the end of the file
 10 Local variables:
 11 mode: sgml
 12 sgml-omittag:t
 13 sgml-shorttag:t
 14 sgml-minimize-attributes:nil
 15 sgml-always-quote-attributes:t
 16 sgml-indent-step:1
 17 sgml-indent-data:t
 18 indent-tabs-mode:nil
 19 sgml-parent-document:nil
 20 sgml-default-dtd-file:"~/ .phpdoc/manual.ced"
 21 sgml-exposed-tags:nil
 22 sgml-local-catalogs:nil
 23 sgml-local-ecat-files:nil
 24 End:
 25 vim600: syn=xml fen fdm=syntax fdl=2 si
 26 vim: et tw=78 syn=sgml
 27 vi: ts=1 sw=1
 28 -->
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39

Main menu ▾

Number of failures to meet 'strict standards' (1065) +

All files -

Search:

- Repository
 - doc-base
 - en
 - appendices
 - chapters
 - chapters1
 - chmonly
 - docweb
 - faq
 - features
 - install
 - internals2
 - language
 - context
 - control-structures
 - figures
 - oop5
 - predefined
 - types
 - wrappers
 - basic-syntax.xml
 - constants.xml
 - context.xml
 - control-structures.xml
 - exceptions.xml
 - expressions.xml
 - functions.xml
 - namespaces.xml
 - oop5.xml
 - operators.xml
 - references.xml
 - types.xml
 - variables.xml

Home global.ent funcindex.xml manual.xml.in constants.xml phpbook.dtd namespaces.xml

File: en/language/namespaces.xml

Tools

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- $Revision: 305081 $ -->
3 <chapter xml:id="language.namespaces" xmlns="http://docbook.org/ns/docbook"
4   version="1.1">
5   <title>Namespaces</title>
6
7   <sect1 xml:id="language.namespaces.rationale">
8     <title>Namespaces overview</title>
9     <simpara>
10      What are namespaces? In the broadest definition namespaces are a way of encaps
11      items. This can be seen as an abstract concept in many places. For example, i
12      operating system directories serve to group related files, and act as a namesp
13      the files within them. As a concrete example, the file <literal>foo.txt</liter
14      exist in both directory <literal>/home/greg</literal> and in <literal>/home/ot
15      but two copies of <literal>foo.txt</literal> cannot co-exist in the same direc
16      addition, to access the <literal>foo.txt</literal> file outside of the
17      <literal>/home/greg</literal> directory, we must prepend the directory name to
18      name using the directory separator to get <literal>/home/greg/foo.txt</literal>
19      same principle extends to namespaces in the programming world.
20    </simpara>
21    <simpara>
22      In the PHP world, namespaces are designed to solve two problems that authors
23      of libraries and applications encounter when creating re-usable code elements
24      such as classes or functions:
25    </simpara>
26    <para>
27      <orderedlist>
28        <listitem>
29          <simpara>
30            Name collisions between code you create, and
31            internal PHP classes/functions/constants or third-party classes/functions/c
32          </simpara>
33        </listitem>
34        <listitem>
35          <simpara>
36            Ability to alias (or shorten) Extra_Long_Names designed to alleviate the fi
37            improving readability of source code.
38          </simpara>
39        </listitem>
40      </orderedlist>
41    </para>

```


- These are my OPINIONS, and, as such, are probably wrong in many projects/products/communities.
- Mostly the result of ten years on the Apache HTTP Server documentation project



That's Kathy Sierra
She's amazing

- Documentation in Open Source is terrible
- Nobody wants to do it
- This is simply a reality of Open Source, and we have to put up with it

- Lots of people want to write docs (some of them can even write a coherent sentence)
- We make it way too hard for them to participate
- So they flock to third-party forums, where they share misconceptions and worst-practice "solutions."

Have you noticed?

The more likely a "support" channel is to say "RTFM", the worse the documentation is likely to be.

Smart Questions: Eric Raymond

RTFM and STFW: How To Tell You've Seriously Screwed Up

There is an ancient and hallowed tradition: if you get a reply that reads “RTFM”, the person who sent it thinks you should have Read The F*cking Manual. He or she is almost certainly right. Go read it.



RTFM has a younger relative: if you get a reply that reads “STFW”, the person who sent it thinks you should have Searched The F*cking Web. He or she is almost certainly right. Go search it. (The milder version of this is when you are told “Google is your friend!”)



Eric Is Wrong

**It is not a "hallowed
tradition".**

**It's bad manners, and
it's juvenile.**

It's time to grow up.

CC Jumer, Flickr

OmniTI

The RTFM attitude is indicative of arrogance and impatience, whereas truly great documentation is the result of patience and humility

(Yes, they should read the docs. No, you should not be rude.)

The docs are never comprehensive

All documentation is inadequate
for my specific weirdo situation

The question:

- Should we care?
- What should we do about it?

Or, stated differently

- What should the documentation cover?
- How should we go about getting there?

Most projects never ask these questions, which is why their documentation is so awful.

- What should the documentation cover?
- How should we go about getting there?

Who is your audience?

- Usually (at least) two answers to this:
 - Developers (core product - API docs)
 - Customizers (how do I make it do X?)
 - End-users (Just make the darned thing work?)

Audience => Concerns

- Just make it work
- Security
- Performance
- Maintenance

- These are all valid questions
- Just make it work
 - Security
 - Performance
 - Maintenance

- Core developers (API docs)
- Server admins (install, configure, security)
- Programmers (Reference manual)
- Hobbyists (I just wanna ...)
- Maintainers (This product is written in php and ...)

Apache HTTP Server: Audiences

- Developers (core)
- Developers (Third-party modules)
- Server admins (install, configure, secure)
- Developers (stuff on top of Apache)
- Website developers (HTML, Javascript, CSS)
- Users of third-party products that live on top of Apache

Drupal: Audiences

- Developers (core)
- Developers (extensions)
- Theme developers
- System admins
- End users
- All of these folks also need the PHP docs and the Apache docs, but may come to us with their seemingly off-topic questions

Documentation

[Docs Home](#)[API](#)[Recently Updated](#)

Getting Started with Drupal

Understanding Drupal

Learn about Drupal concepts, technology stack, terminology, and resources.

Installation Guide

Install Drupal and its contributed modules and themes. Run multiple sites from one installation. Migrate from other content management systems and address platform issues.

Administration Guide

Manage users and content, perform backups and upgrades, secure your site, tweak performance, etc. *Audience: System and site administrators*

Structure Guide

Work with content types, blocks, menus, views, panels, taxonomy, multilingual content, user profiles, and navigation. *Audience: information architects*

Site Building Guide

Add functionality and features such as e-commerce, forums, media, search, geographic data, dates, workflow, messaging, forms, social networking, etc. *Audience: site builders, developers and business architects*

Theming Guide

Customize the interface using templates, CSS, etc. Override the output from core or contributed modules. *Audience: designers, usability and accessibility professionals, interface experts*

Help Us Maintain

The Drupal.org online documentation is written by the Drupal community in operation with the Drupal community. If you are logged in, you can

Edit most Documentation pages. Click the "Edit" link at the top of the page.

Add new pages by using the "Add new link" link at the bottom of the page.

Use the [Documentation API](#) to propose major changes.

Drupal's online documentation was created in 2011 by the individual contributors and is licensed under the [Creative Commons License](#), and the PHP code is distributed under the [Public License](#).

Writing Your Own Code

Developing for Drupal

Work with the API, JavaScript, and databases. Learn the Drupal coding standards. *Audience: developers*

API Reference

Tutorials

Drupal Cookbook

Follow a walkthrough of a typical Drupal setup.

Tutorials

Follow step-by-step instructions for a number of common Drupal tasks.

Joomla! Official Documentation

Administrators

Installation	Installation Manual, Security Measures, Technical Requirements, Upgrade Instructions
Security	Information on how to secure your Joomla Installation
Beginners Guide	New to Joomla? This guide will answer your first questions, and give you a general understanding on how Joomla works
User and Access Management	Manage your users and their access Levels.
Article Management	Tutorials, and answers to your FAQs on Article Management
Category Management	Tutorials, and answers to your FAQs on Category Management
Menu Management	Tutorials, and answers to your FAQs on Menu Management
Module Management	Tutorials and answers to your FAQs on Module Management
Component Management	Tutorials and answers to your FAQs on Component Managment
Template Management	Tutorials and answers to your FAQs on Template Management
Plugin Management	Tutorials and answers to your FAQs on Plugin Management
Global Configuration Management	Tutorials and answers to your FAQs on Global Configuration Management

Extension Developers

Joomla Framework API Documentation	API Reference for the Joomla Framework + Tutorials
Templates	Tutorials, and answers to your FAQs on Template Development
Components	Tutorials, and answers to your FAQs on Component Development
Modules	Tutorials, and answers to your FAQs on Module Development
Plugins	Tutorials, and answers to your FAQs on Plugin Development
Languages	Tutorials and answers to your FAQs on Language Support for your extensions

Not so good



Getting started with Word 2010

[Start now](#)

1 2 3



More to explore

[What's new in Word 2010](#)
[Create your first document](#)



Learning resources

[Find the help you need](#)
[Word 2007 help](#)
[All Office training](#)

Top issues

1. [How do I adjust the spaces between lines and paragraphs?](#)
2. [Spell Checker does not recognize misspelled words in Word 2007](#)
3. [How to open and save Word 2007, Excel 2007, and PowerPoint 2007 files in earlier versions of Office programs](#)
4. [Can I view or print a file from the Backstage view?](#)

[More solutions: Word Solution Center](#)

Word 2010

- Activating Word
- Charts
- Creating documents
- File management
- File migration
- Formatting
- Getting help
- Getting started with Word
- Headers, footers, and page numbers
- Installing
- Page breaks and section breaks
- Page setup
- Pictures and Clip Art
- Saving and printing
- Security and privacy
- SmartArt graphics
- Tables
- Tables of contents and other references
- Tracking changes and comments

[All Categories](#)

Word 2007

Word 2003



Discover
Office Web Apps >

advertisement

Microsoft® Pinpoint™

Find the



MySQL 5.5 Reference Manual

Copyright © 1997, 2011, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

[Preface and Notes »](#)

Section Navigation [\[Toggle\]](#)

- [MySQL 5.5 Reference Manual](#)
 - [Preface and Notes](#)
 - [1 General Information](#)
 - [2 Installing and Upgrading MySQL](#)
 - [3 Tutorial](#)
 - [4 MySQL Programs](#)
 - [5 MySQL Server Administration](#)
 - [6 Backup and Recovery](#)
 - [7 Optimization](#)
 - [8 Language Structure](#)
 - [9 Internationalization and Localization](#)
 - [10 Data Types](#)
 - [11 Functions and Operators](#)
 - [12 SQL Statement Syntax](#)
 - [13 Storage Engines](#)
 - [14 High Availability and Scalability](#)
 - [15 MySQL Enterprise Monitor](#)
 - [16 MySQL Workbench](#)
 - [17 Replication](#)
 - [18 Partitioning](#)
 - [19 Stored Programs and Views](#)
 - [20 INFORMATION_SCHEMA Tables](#)
 - [21 MySQL Performance Schema](#)
 - [22 Connectors and APIs](#)
 - [23 Extending MySQL](#)
 - [A Licenses for Third-Party Components](#)
 - [B MySQL 5.5 Frequently Asked Questions](#)
 - [C Errors, Error Codes, and Common Problems](#)

Will it ever end?

Git User's Manual (for version 1.5.3 or newer)

Table of Contents

1. Repositories and Branches

[How to get a git repository](#)

[How to check out a different version of a project](#)

[Understanding History: Commits](#)

[Understanding history: commits, parents, and reachability](#)

[Understanding history: History diagrams](#)

[Understanding history: What is a branch?](#)

[Manipulating branches](#)

[Examining an old version without creating a new branch](#)

[Examining branches from a remote repository](#)

[Naming branches, tags, and other references](#)

[Updating a repository with git fetch](#)

[Fetching branches from other repositories](#)

2. Exploring git history

[How to use bisect to find a regression](#)

[Naming commits](#)

[Creating tags](#)

[Browsing revisions](#)

[Generating diffs](#)

[Viewing old file versions](#)

[Examples](#)

[Counting the number of commits on a branch](#)

[Check whether two branches point at the same history](#)

[Find first tagged version including a given fix](#)

[Showing commits unique to a given branch](#)

[Creating a changelog and tarball for a software release](#)

[Finding commits referencing a file with given content](#)

3. Developing with git

[Telling git your name](#)

[Creating a new repository](#)

[How to make a commit](#)

[Creating good commit messages](#)

[Ignoring files](#)

[How to merge](#)

[Resolving a merge](#)

[Getting conflict-resolution help during a merge](#)

[Undoing a merge](#)

[Fast-forward merges](#)

[Fixing mistakes](#)

[Fixing a mistake with a new commit](#)

[Fixing a mistake by rewriting history](#)

[Checking out an old version of a file](#)

Will it ever end?

- [Fixing a mistake by rewriting history](#)
- [Checking out an old version of a file](#)
- [Temporarily setting aside work in progress](#)

- [Ensuring good performance](#)

- [Ensuring reliability](#)

- [Checking the repository for corruption](#)

- [Recovering lost changes](#)

[4. Sharing development with others](#)

- [Getting updates with git pull](#)

- [Submitting patches to a project](#)

- [Importing patches to a project](#)

- [Public git repositories](#)

- [Setting up a public repository](#)

- [Exporting a git repository via the git protocol](#)

- [Exporting a git repository via http](#)

- [Pushing changes to a public repository](#)

- [What to do when a push fails](#)

- [Setting up a shared repository](#)

- [Allowing web browsing of a repository](#)

- [Examples](#)

- [Maintaining topic branches for a Linux subsystem maintainer](#)

[5. Rewriting history and maintaining patch series](#)

- [Creating the perfect patch series](#)

- [Keeping a patch series up to date using git rebase](#)

- [Rewriting a single commit](#)

- [Reordering or selecting from a patch series](#)

- [Other tools](#)

- [Problems with rewriting history](#)

- [Why bisecting merge commits can be harder than bisecting linear history](#)

[6. Advanced branch management](#)

- [Fetching individual branches](#)

- [git fetch and fast-forwards](#)

- [Forcing git fetch to do non-fast-forward updates](#)

- [Configuring remote-tracking branches](#)

[7. Git concepts](#)

- [The Object Database](#)

- [Commit Object](#)

- [Tree Object](#)

- [Blob Object](#)

- [Trust](#)

- [Tag Object](#)

- [How git stores objects efficiently: pack files](#)

- [Dangling objects](#)

- [Recovering from repository corruption](#)

- [The index](#)

[8. Submodules](#)

- [Pitfalls with submodules](#)

Will it ever end?

- [The index](#)
- 8. [Submodules](#)
 - [Pitfalls with submodules](#)
- 9. [Low-level git operations](#)
 - [Object access and manipulation](#)
 - [The Workflow](#)
 - [working directory → index](#)
 - [index → object database](#)
 - [object database → index](#)
 - [index → working directory](#)
 - [Tying it all together](#)
 - [Examining the data](#)
 - [Merging multiple trees](#)
 - [Merging multiple trees, continued](#)
- 10. [Hacking git](#)
 - [Object storage format](#)
 - [A birds-eye view of Git's source code](#)
- 11. [Git Glossary](#)
- A. [Git Quick Reference](#)
 - [Creating a new repository](#)
 - [Managing branches](#)
 - [Exploring history](#)
 - [Making changes](#)
 - [Merging](#)
 - [Sharing your changes](#)
 - [Repository maintenance](#)
- B. [Notes and todo list for this manual](#)

Git is a fast distributed revision control system.

This manual is designed to be readable by someone with basic UNIX command-line skills, but no previous knowledge of git.

[Chapter 1, Repositories and Branches](#) and [Chapter 2, Exploring git history](#) explain how to fetch and study a project using git—of a software project, search for regressions, and so on.

People needing to do actual development will also want to read [Chapter 3, Developing with git](#) and [Chapter 4, Sharing development](#).

Further chapters cover more specialized topics.

Comprehensive reference documentation is available through the man pages, or [git-help\(1\)](#) command. For example, for the cor

```
$ man git-clone
```

or:

```
$ git help clone
```

Yes, it's all one page

git help clone

With the latter, you can use the manual viewer of your choice; see [git-help\(1\)](#) for more information.

See also [Appendix A, Git Quick Reference](#) for a brief overview of git commands, without any explanation.

Finally, see [Appendix B, Notes and todo list for this manual](#) for ways that you can help make this manual more complete.

Chapter 1. Repositories and Branches

Table of Contents

[How to get a git repository](#)

[How to check out a different version of a project](#)

[Understanding History: Commits](#)

[Understanding history: commits, parents, and reachability](#)

[Understanding history: History diagrams](#)

[Understanding history: What is a branch?](#)

[Manipulating branches](#)

[Examining an old version without creating a new branch](#)

[Examining branches from a remote repository](#)

[Naming branches, tags, and other references](#)

[Updating a repository with git fetch](#)

[Fetching branches from other repositories](#)

Chapter 1. Repositories and Branches

How to get a git repository

It will be useful to have a git repository to experiment with as you read this manual.

The best way to get one is by using the [git-clone\(1\)](#) command to download a copy of an existing repository. If you don't already

```
# git itself (approx. 10MB download):
$ git clone git://git.kernel.org/pub/scm/git/git.git
# the Linux kernel (approx. 150MB download):
$ git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git
```

The initial clone may be time-consuming for a large project, but you will only need to clone once.

The clone command creates a new directory named after the project ("git" or "linux-2.6" in the examples above). After you cd files, called the [working tree](#), together with a special top-level directory named ".git", which contains all the information about t

How to check out a different version of a project

Git is best thought of as a tool for storing the history of a collection of files. It stores the history as a compressed collection of version is called a [commit](#).

Those snapshots aren't necessarily all arranged in a single line from oldest to newest; instead, work may simultaneously proc

So what?

- You need to answer the kinds of questions they're likely to have
- You need to respect each audience, while not ignoring any of them
- There is, of course, HUGE overlap
- But also there are areas that are only of interest to one audience or the other



CC alexkess, Flickr

Remember also that
your audience are not
all white american
males

Inside jokes and
cultural references are
unprofessional ...
usually

strpos — Find position of first occurrence of a string

Description

[Report a bug](#)

```
int strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )
```

Returns the numeric position of the first occurrence of *needle* in the *haystack* string. Unlike the [strrpos\(\)](#) before PHP 5, this function can take a full string as the *needle* parameter and the entire string will be used.

- My favorite function doc, anywhere
- Immediately obvious
- But how does it work for folks that aren't as proficient in English?

Lovely Plumage

Functions can also be called using keyword arguments of the form `keyword = value`. For instance, the following function:

```
def parrot(voltage, state='a stiff', action='vroom', type='Norwegian Bl
    print "-- This parrot wouldn't", action,
    print "if you put", voltage, "volts through it."
    print "-- Lovely plumage, the", type
    print "-- It's", state, "!"
```

could be called in any of the following ways:

```
parrot(1000)
parrot(action = 'VOOOOOM', voltage = 1000000)
parrot('a thousand', state = 'pushing up the daisies')
parrot('a million', 'bereft of life', 'jump')
```

but the following calls would all be invalid:

```
parrot()           # required argument missing
parrot(voltage=5.0, 'dead') # non-keyword argument following keyword
parrot(110, voltage=220)   # duplicate value for argument
parrot(actor='John Cleese') # unknown keyword
```

In general, an argument list must have any positional arguments followed by any keyword arguments, where the keywords must be chosen from the

Codex

Main Page

Welcome to the WordPress Codex. Here you can find and contribute to the WordPress documentation. The first section – [Getting Started](#) – should be enough if you want to dive in to publishing posts – more detailed topics and other FAQs are listed further down the page.

Getting Started with WordPress »

- [New To WordPress – Where to Start](#)
- [Installation](#)
- [Installation Help](#)
- [Upgrading WordPress](#)
- [WordPress in Your Language](#)
- [Posting in WordPress](#)

So ...



CC
herman.seoane,
Flickr

- Once you think you know who your audience is ...

What questions are they asking?

- When your product is young, you answer the questions that you expect to be asked
- As it matures, you should listen to what's actually being asked

What questions are they asking?

Most documentation never progresses past this point



- When your product is young, you answer the questions that you expect to be asked
- As it matures, you should listen to what's actually being asked

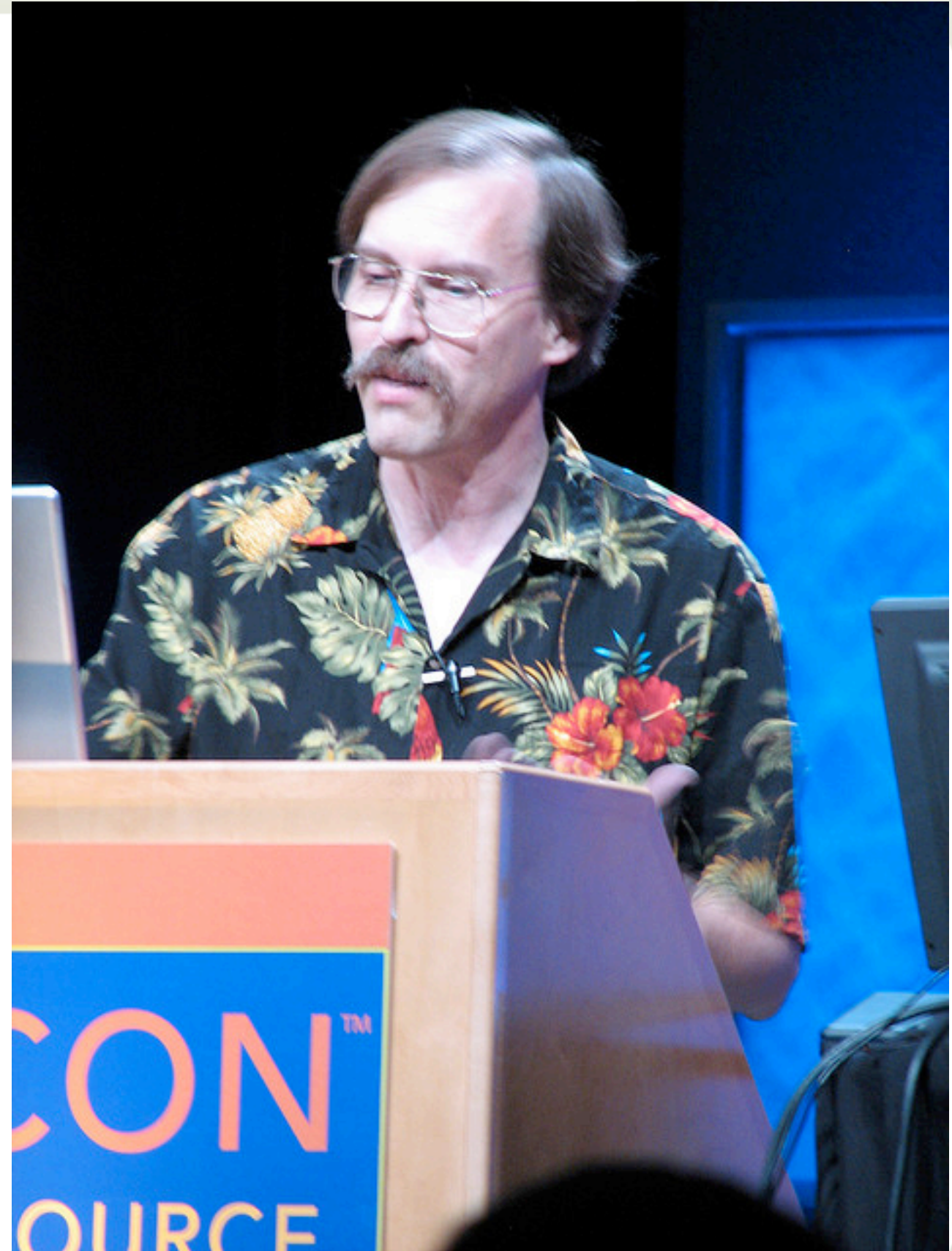
How docs are born

- Documentation tends to have one of two beginnings
 - Proactive: What do we think people will ask?
 - Reactive: What are people asking?

What are the questions?

- You have to actually listen
- You have to go where they're asking
 - Mailing lists
 - IRC
 - Third-party forums
- Their questions matter to them. Be respectful

- Laziness
- Impatience
- Hubris



- Answer the question thoroughly, once
- Save your answer
- Next time, give them a URL
- Better to do something well, once, than do to it poorly, again and again

- Impatience with the question comes across as disrespect for the questioner
- Arrogance and disrespect are at the heart of the RTFM mindset
- If you cannot be patient, please let someone else answer the question

Love is patient, love is kind ... it
does not keep a record of offenses.
(1 Corinthians 13 - The Bible)

- You don't know everything
- The documentation isn't perfect yet
- Remember that once you, too, were completely ignorant

FALSE

- There's no such thing as the wrong question: False - but it's your job to guide them to the right question, and its answer
- There's no such thing as a stupid question (but there are a lot of inquisitive idiots)



How do we answer them?

- Reference manual
- HowTos
- FAQs
- Examples

- Comprehensive and exhaustive
- Correct
- Consistent format
- Best practice
- Lots of examples
- All examples must be tested

- Obvious, right?
- You'd think

[Perl functions A-Z](#) | [Perl functions by category](#) | [The 'perlfunc' manpage](#)

- **sprintf FORMAT, LIST**

Returns a string formatted by the usual `printf` conventions of the C library function `sprintf`. See below for more details and see `sprintf(3)` or `printf(3)` on your system for an explanation of the general principles.

For example:

- What the heck are `sprintf(3)` and `printf(3)`?
- I came here for the general principles

- However ... "Comprehensive" needs to be clearly defined
- Do the PHP docs need to cover the history of computing?
- Do the Apache docs need to cover HTML?

AddAltByEncoding Directive

Description: Alternate text to display for a file instead of an icon selected by MIME-encoding

Syntax: `AddAltByEncoding string MIME-encoding [MIME-encoding] ...`

Context: server config, virtual host, directory, .htaccess

Override: Indexes

Status: Base

Module: mod_autoindex

`AddAltByEncoding` provides the alternate text to display for a file, instead of an icon, for [FancyIndexing](#). *MIME-encoding* is a valid content-encoding, such as x-compress. If *String* contains any whitespace, you have to enclose it in quotes (" or '). This alternate text is displayed if the client is image-incapable, has image loading disabled, or fails to retrieve the icon.

Consistent Format

preg_match

(PHP 4, PHP 5)

preg_match — Perform a regular expression match

Description

[Report a bug](#)

```
int preg_match ( string $pattern , string $subject [, array &$matches [,  
int $flags = 0 [, int $offset = 0 ]]] )
```

Searches *subject* for a match to the regular expression given in *pattern*.

Parameters

[Report a bug](#)

pattern

- Beginners often just want it to work
- The **best** answer is often more complicated than the **good enough** answer
- Doing it right now saves time and tears later

Third-party "documentation"


- Usually focused on "good enough"
- Thus usually insecure, inefficient, or fragile



Almost right ...

[AskApache](#)[Hacking](#)[Online Tools](#)

PHP to handle HTTP Status Codes for ErrorDocument

 protection. If you are reading this article, you already know enough about the benefits of making sure your site can handle HTTP Protocol Errors. This is a nice single php file with no dependencies or requirements, will work on anything. Optimized for minimizing bandwidth and resource-hogging connections from bots and spambots.

```
<?php
ob_start();
@set_time_limit(5);
@ini_set('memory_limit', '64M');
@ini_set('display_errors', 'Off');
error_reporting(0);
```

View ["PHP to handle HTTP Status Codes for ErrorDocument"...](#) –Nov 18, 2010

AskApache Password Protection 4.7 Update in 2 Weeks



I am now about 1 week away from publishing the much-anticipated 4.7 update to the AskApache Password Protection WordPress plugin. It's an upgrade I've been working on for almost 2 years (off and on)! I have been using the new version for quite some time now, and have made a lot of improvements to it, and finally I decided enough users have suffered with the old version. I am very excited for this release, it fixes all known bugs in the older versions, and brings some heavy-duty improvements to all facets of this plugin.. not to mention way better security modules (Lots more COOKIE use) based on code I use with clients.

View ["AskApache Password Protection 4.7 Update in 2 Weeks"...](#) –Nov 09, 2010

Examples

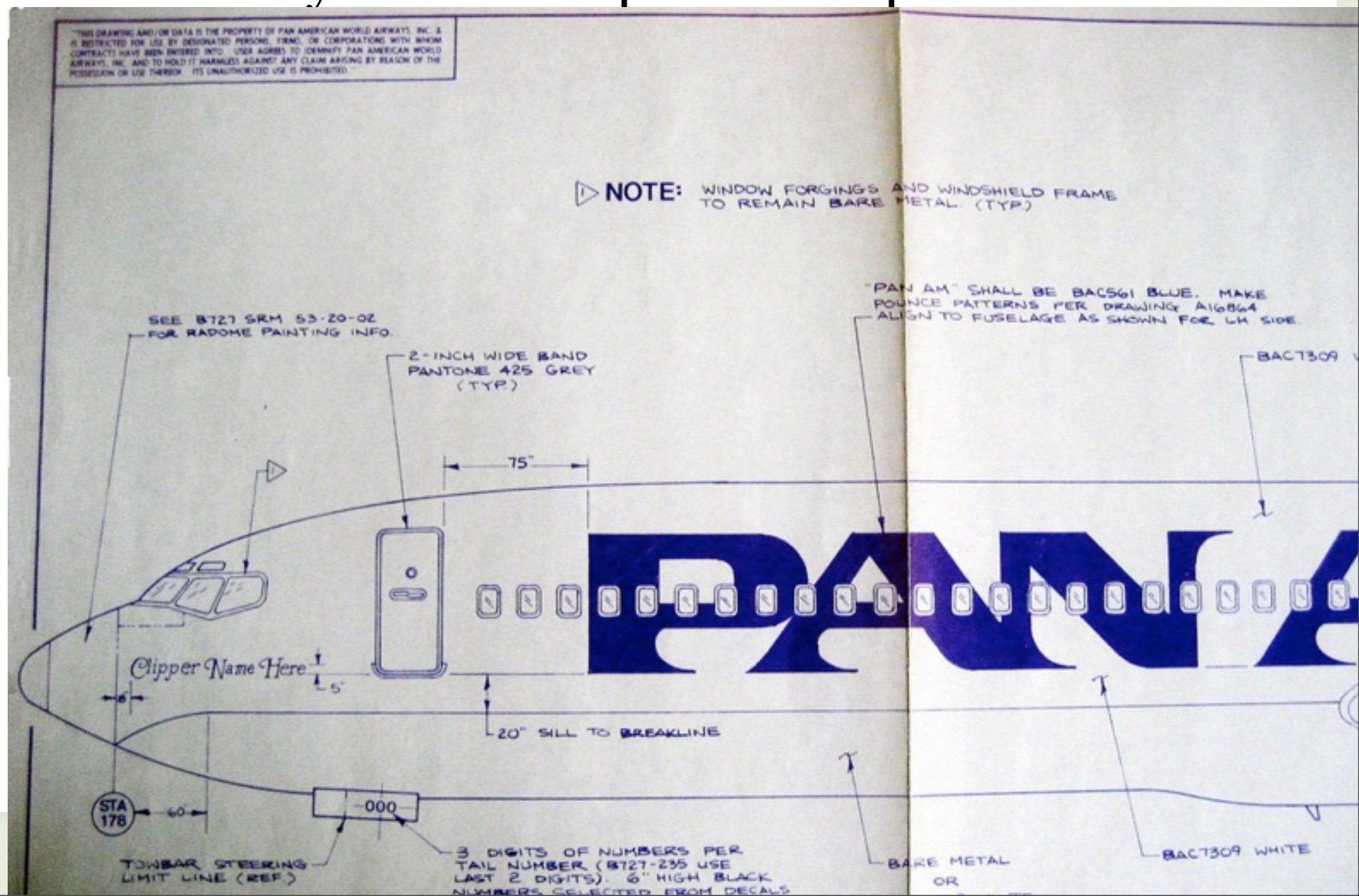
Configure handler based on final extension only

```
<FilesMatch \.cgi$>  
    SetHandler cgi-script  
</FilesMatch>
```

- Simple
- Copious
- Tested
- Consistent use of a fictitious site/project/implementation

Examples: Simple

- Simplest example that illustrates the concept
- Explained in exhaustive detail
- Perhaps followed by more complex examples



Examples: Copious

- More examples are always better than fewer
- ... if they are useful examples
- ... and are explained adequately
- O'Reilly's Cookbooks are consistently best-sellers

Reverse Dynamic Mirror

Description:

...

Solution:

```
RewriteEngine on
RewriteCond    /mirror/of/remotesite/$1          -U
RewriteRule    ^http://www\.remotesite\.com/(.*)$ /mirror/of/remotesite/$1
```

Example from the Apache 1.3
mod_rewrite documentation

Examples: Tested

- Few things spread faster than incorrect examples
- Test every example. Even ones that seem trivial
- Incorrect examples lead to many, many hours of lost productivity

- The examples are in the documentation, and automatically become tests
- It's practically magic
- Does PHP have something like this?

- Construct an imaginary project/company/site/whatever
- Consistently refer to it in the documentation
- **example.org** or **Acme Widgets, Inc**, for example
- Changing the hero of your story in the middle confuses the reader

Examples: Useful

```
<h1>El Jefe's Tours</h1>
<form action="path to setECURL" method="post">
<input id="submitBtn" type="submit"
    value="Pay with PayPal" />
<input type="hidden" name="success_url"
    value="path to successURL" />
<input type="hidden" name="cancel_url"
    value="path to cancelURL">
</form>
<!-- End custom merchant code -->
<!-- Example courtesy of PayPal documentation -->
```


Examples: Useful

```
<h1>El Jefe's Tours</h1>
```

```
<form action="path to setECURL" method="post">
```

```
<input id="submitBtn" type="submit"
```

```
  value="Pay with PayPal"/>
```

```
<input type="hidden" name="success_url"
```

```
  value="path to successURL"/>
```

```
<input type="hidden" name="cancel_url"
```

```
  value="path to cancelURL">
```


```
</form>
```

```
<!-- End custom merchant code -->
```

It took two hours to find the wrong value, and a further hour to find the right value ... which still didn't work.

- Complete - cover even the trivial steps.
- Never say "easy", "trivial", "simple", or "of course". Your reader is there because it's not.
- Test it. Repeatedly. On multiple systems.
- Let your inexperienced co-workers read it.

- Complete - cover even the trivial steps.



This is a delicate balance - between
deciding what's in scope, and not
leaving them to figure out everything
on their own

Silly (Apache mod_rewrite docs)

Given Rule	Resulting Substitution
-----	-----
<code>^localpath(*) otherpath\$1</code>	<code>/somepath/otherpath/pathinfo</code>
<code>^localpath(*) otherpath\$1 [R]</code>	<code>http://thishost/somepath/otherpath/pathinfo</code> via external redirection
<code>^localpath(*) otherpath\$1 [P]</code>	not supported, because silly!
-----	-----
<code>^localpath(*) /otherpath\$1</code>	<code>/otherpath/pathinfo</code>
<code>^localpath(*) /otherpath\$1 [R]</code>	<code>http://thishost/otherpath/pathinfo</code> via external redirection
<code>^localpath(*) /otherpath\$1 [P]</code>	not supported, because silly!
-----	-----
<code>^localpath(*) http://thishost/otherpath\$1</code>	<code>/otherpath/pathinfo</code>
<code>^localpath(*) http://thishost/otherpath\$1 [R]</code>	<code>http://thishost/otherpath/pathinfo</code> via external redirection
<code>^localpath(*) http://thishost/otherpath\$1 [P]</code>	not supported, because silly!
-----	-----
<code>^localpath(*) http://otherhost/otherpath\$1</code>	<code>http://otherhost/otherpath/pathinfo</code> via external redirection
<code>^localpath(*) http://otherhost/otherpath\$1 [R]</code>	<code>http://otherhost/otherpath/pathinfo</code> via external redirection (the [R] flag is redundant)
<code>^localpath(*) http://otherhost/otherpath\$1 [P]</code>	<code>http://otherhost/otherpath/pathinfo</code> via internal proxy

- FAQs are often an admission that the documentation is insufficient
- Should be a call for improving the docs, or even a scratch-pad for the new docs
- Most FAQs should be answered with "here's where that's covered in the docs."

Documentation format

- The choice of a documentation format can be quite divisive
- Choosing wrong can lead to many problems
- Of course, there is no right choice, either

Format considerations

- Easy to edit
- Multiple output formats
- Translation-friendly
- Text (ie, non-binary format)
- Revision control

Options include

- Docbook or other XML
- Wikis
- POD (In certain cultures)
- JavaDoc (or phpdoc, or rubydoc)
- If other projects in the same space have a consistent format, don't try to be clever

Searchable

- Excellent documentation without a decent search isn't worth anything
- There is no excuse for not having a good search. Google will do it for you for free.



CC Stéfan, Flickr

Encouraging participation

- Make it easy for people to complain
- Take their complaints seriously
- Don't get offended when they tell you the docs suck
- Do something about it

- PHP does this really well

```
afterwards
```

```
- run locale-gen from the  
shell
```




add a note

... but

- They're pretty much alone in this

Everyone else:

- Create an account
- Get a checkout
- Subscribe to this list
- Create a ticket in Bugzilla
- Email a patch
- Follow up on *that* list



I just wanted to
tell you that you
misspelled
"peony"

Welcome contributions



- Make it obvious how to submit comments, improvements, errata
- Don't ignore them once they're submitted
- Be quick to offer commit bits to repeat customers

Going to the source

- The developers (often) don't like writing docs
- When they realize you do, they'll be willing to answer your questions

Also, the source

- Learn to read the source code
- It will save you many tears in the long run
- However, don't require programming knowledge to participate in the documentation

```
static void *merge_core_dir_configs(apr_pool_t *a, void *basev,
{
    core_dir_config *base = (core_dir_config *)basev;
    core_dir_config *new = (core_dir_config *)newv;
    core_dir_config *conf;
    int i;

    /* Create this conf by duplicating the base, replacing elements
     * (or creating copies for merging) where new-> values exist
     */
    conf = (core_dir_config *)apr_pmemdup(a, base, sizeof(core_dir_config));

    conf->d = new->d;
    conf->d_is_fnmatch = new->d_is_fnmatch;
    conf->d_components = new->d_components;
    conf->r = new->r;
    conf->condition = new->condition;

    if (new->opts & OPT_UNSET) {
        /* there was no explicit setting of new->opts, so we merge
         * preserve the invariant (opts_add & opts_remove) == 0
         */
        conf->opts_add = (conf->opts_add & ~new->opts_remove) |
            new->opts_add;
        conf->opts_remove = (conf->opts_remove & ~new->opts_add) |
            new->opts_remove;
        conf->opts = (conf->opts & ~conf->opts_remove) | conf->opts_add;

        /* If Includes was enabled with exec in the base config
         * and was enabled without exec in the new config, then disable
         * it in the merged config.
         */
    }
}
```


Error messages

- Error messages are documentation, too

```
else if (!(status & LP_PERRORP)) { if (last !=  
LP_PERRORP) { last = LP_PERRORP; printk  
(KERN_INFO "lp%d on fire\n", minor); } }
```

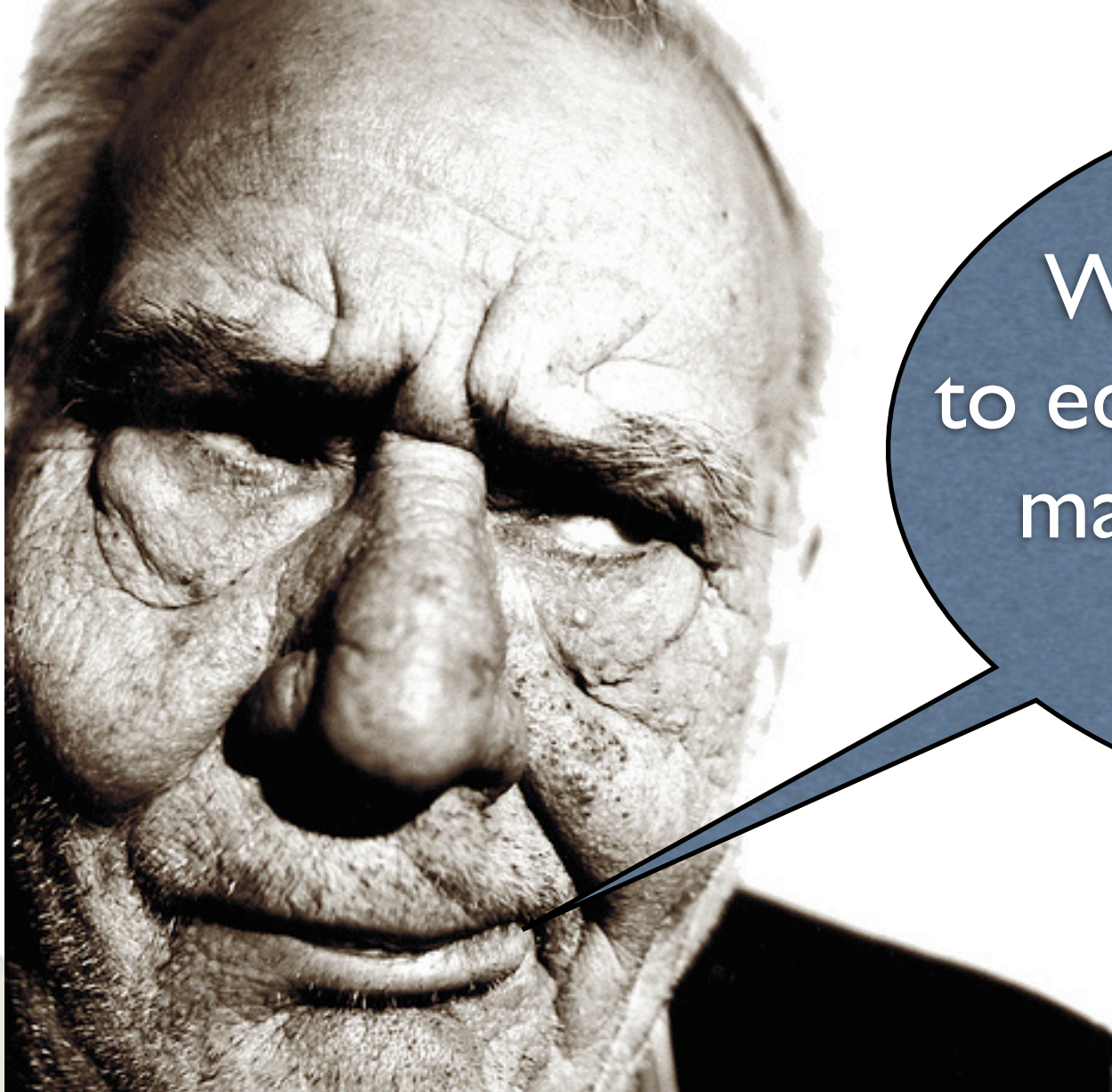
Linux printer driver, circa 2.2.1

How much to say

- What should the documentation **not** cover?
- You must decide what things are outside of the scope
- Once you cross the line, you'll end up writing books, and down that road lies madness

Work for it

We have a tendency to want to make them work for the information, either in a mistaken notion that this will make them remember it, or merely because *we* had to work for it, so they should too



Well, in my day, we had to edit the inodes with a bar magnet! Go figure it out yourself, punk!

Harness the whining



cc cindy47452 flickr

- Whiners -> Contributors: HARD
- Potential contributors -> Whiners: EASY



Your
documentation sucks

Shut up. You're ugly and your mother
dresses you funny.

Result: Your documentation still sucks, and
you've alienated someone who wanted, albeit
misguidedly, to help.



Your
documentation sucks

How do you think we could improve it?

Result: Your documentation might get better, and, even if it doesn't, you've told one person that you care what your customers think.

Questions?

- rbowen@php.net
- <http://people.apache.org/~rbowen> <-- Slides here
- <http://betterfm.org/>
- https://doc.php.net/php/missing_examples.php
(You can help)
- <http://joind.in/3437>