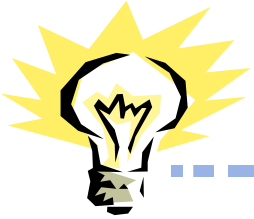


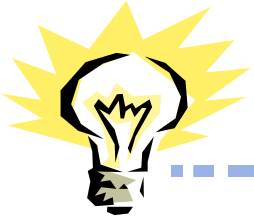
Help and Documentation

It's in the manual



Agenda

- Guidelines
- Types of doc/help
- Presentation issues
- Doc organization



Customer Support

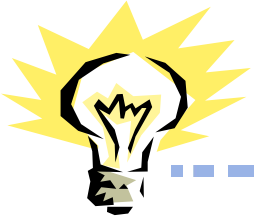
DILBERT / SCOTT ADAMS, scottadams@aol.com



Secure Access Login

Username:

Password:



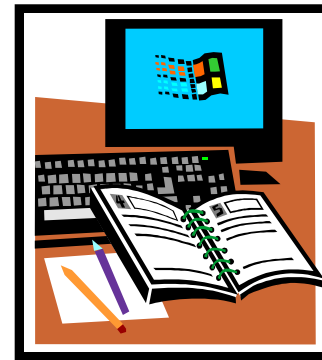
User Support

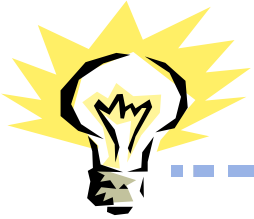
➤ Help

- ❖ Problem-oriented and specific

➤ Documentation

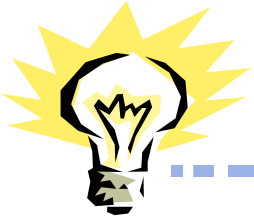
- ❖ System-oriented and general





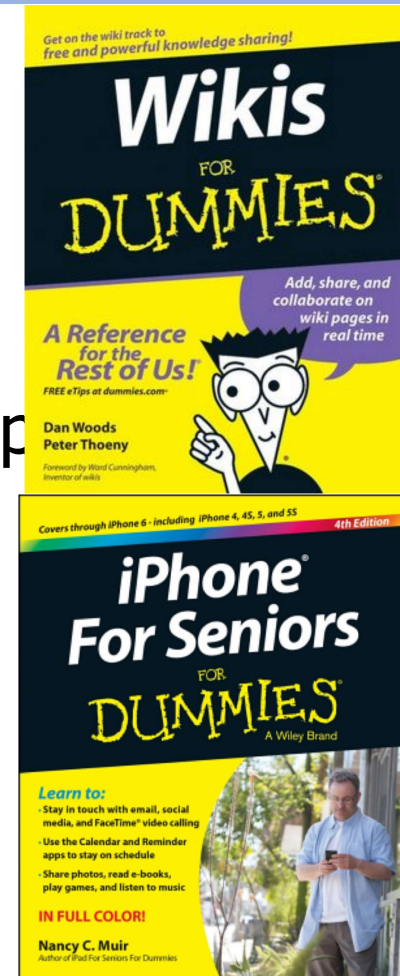
Help & Documentation

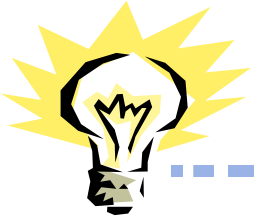
- Essential
 - ❖ BUT never a replacement for bad design
- Simple system
 - ❖ Walk up and use it
 - ❖ Name some
- Most systems with rich features (even well-designed ones) require Help systems



Documentation

- Users don't read manuals
 - ❖ Boring, no goal
 - ❖ Just dive in and start working
- Often use docs in panic mode, when user needs immediate help
 - ❖ Manuals probably locked away
 - ❖ Points to need for on-line help
 - ❖ Need search capability
- Sometimes want quick ref
 - ❖ phone feature card





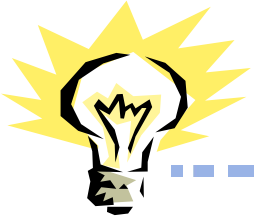
User Support Requirements

➤ Availability

- ❖ Any time the user is operating the system

➤ Accuracy & Completeness

- ❖ Accurate (tricky with changing versions)
- ❖ Cover all aspects of application



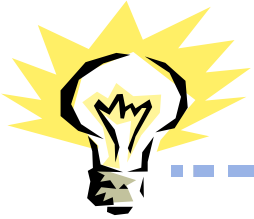
User Support Requirements

➤ Consistency

- ❖ Across different sections
- ❖ Between on-line and paper documentation
- ❖ In terms of terminology, content and style

➤ Robustness

- ❖ Predictable and free of errors



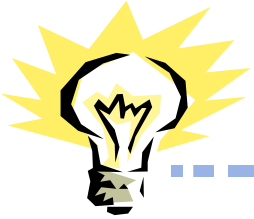
User Support Requirements

➤ Flexibility

- ❖ Appropriate for novices through experts
 - ...maybe have expandable sections of details

➤ Unobtrusiveness

- ❖ Shouldn't distract from or interfere with normal work flow

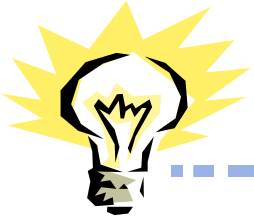


Types of Doc/Help

➤ 1. Tutorial

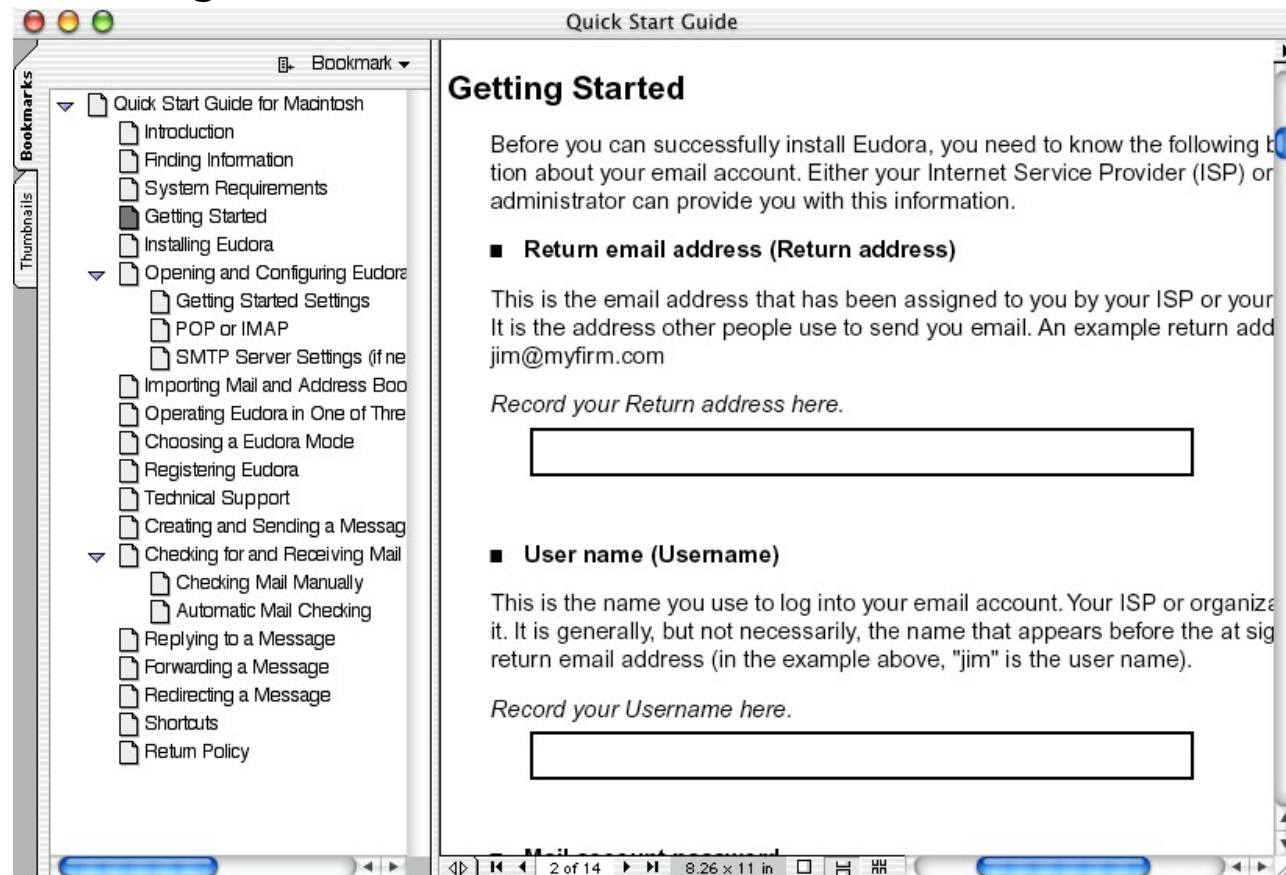
- ❖ For start-up
- ❖ Gets user going
- ❖ Convey conceptual model
- ❖ Communicate essential items
- ❖ Sometimes see on-line tour or demo

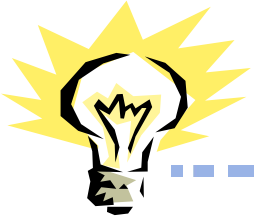




Types of Doc/Help

Quick start guide as a tutorial

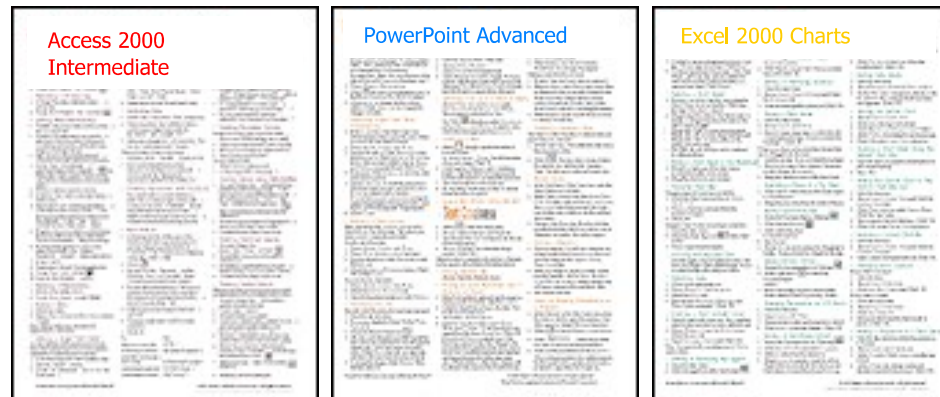


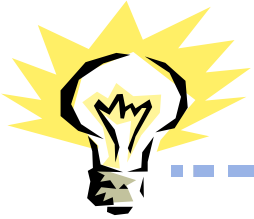


Types of Doc/Help

➤ 2. Quick reference/review

- ❖ Reminder or short reference
- ❖ Often for syntax
- ❖ Can be recall aid for expert
- ❖ Can allow novice to see what's available

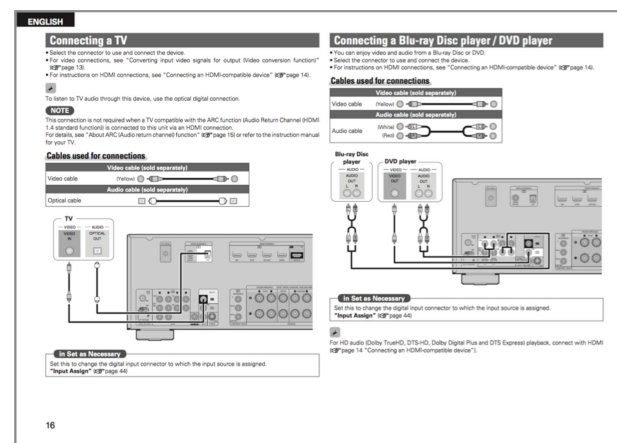
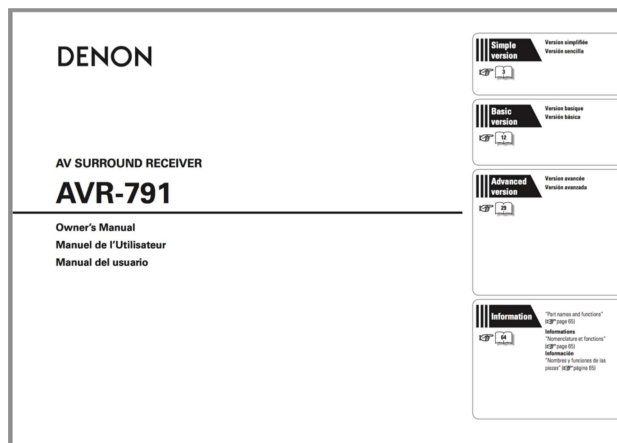


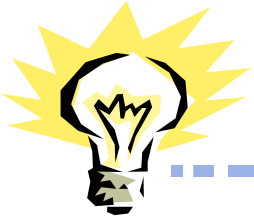


Types of Doc/Help

➤ 3. Reference Manual (Full explanation)

- ❖ Detailed command descriptions
- ❖ Usually for experts
- ❖ Unix on-line manual pages, for example





Types of Doc/Help

Combined
Quick Reference
and full
Reference
Manual

- command
- purpose
- syntax
- example
- links to details

Address: <http://www.php.net/manual/en/function.strftime.php> go

php downloads | documentation | faq | getting help | mailing lists | reporting bugs | php.net sites | links

search for in the

lookup:

PHP Manual

^Date/Time

- checkdate
- date
- getdate
- gettimeofday
- gmdate
- gmmktime
- gmstrftime
- localtime
- microtime
- mktime
- strtotime
- time

«[strftime](#)

view the [printer friendly version](#) or the [printer friendly version with notes](#) or change language to [Bra](#)

strtotime

(PHP 3>= 3.0.12, PHP 4)

strtotime -- Parse about any English textual datetime description into a UNIX timestamp

Description

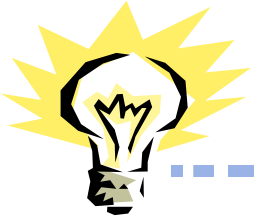
int **strtotime** (string time [, int now])

The function expects to be given a string containing an English date format and will try to Upon failure, -1 is returned.

Because **strtotime()** behaves according to GNU date syntax, have a look at the GNU m

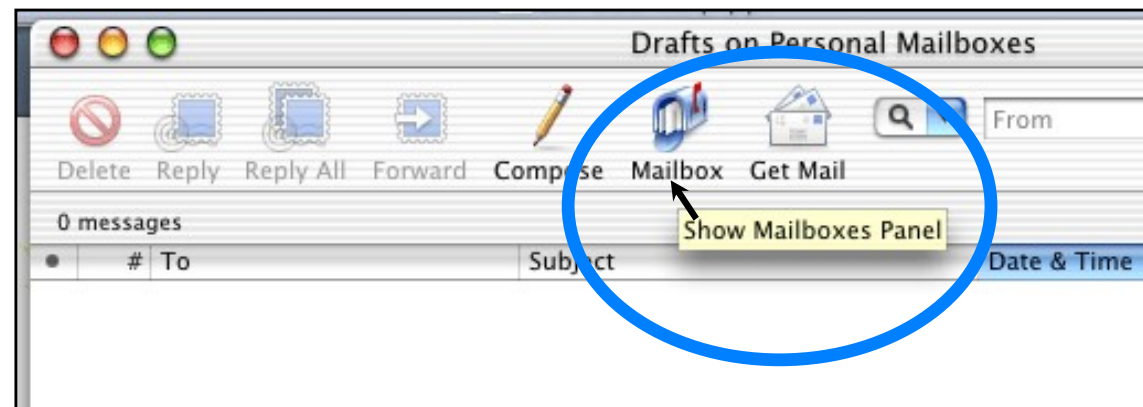
Example 1. strtotime() examples

```
echo strtotime ("now"), "\n";
echo strtotime ("10 September 2000"), "\n";
echo strtotime ("+1 day"), "\n";
echo strtotime ("+1 week"), "\n";
echo strtotime ("+1 week 2 days 4 hours 2 seconds"), "\n";
echo strtotime ("next Thursday"), "\n";
echo strtotime ("last Monday"), "\n";
```



Types of Doc/Help

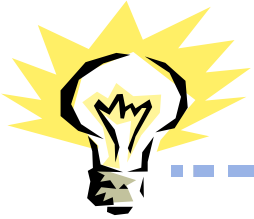
- 4. Context-sensitive (task-specific) help
 - ❖ System provides help on current situation
 - ❖ Balloon help, ToolTips
 - ❖ Other examples?





e.g. Photoshop





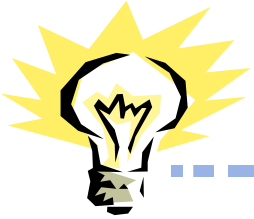
User Support Approaches

➤ Command assistance

- ❖ Specific details on particular command
 - e.g. UNIX `%> man ls`
- ❖ Good if user knows what s/he wants
 - not always the case!

➤ Command prompts

- ❖ Message when user commits an error
- ❖ Menus and icons fall under this category to a degree



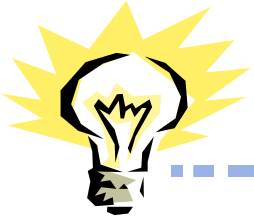
User Support Approaches

➤ Context-sensitive help

- ❖ Information pertinent to a particular situation or interface item

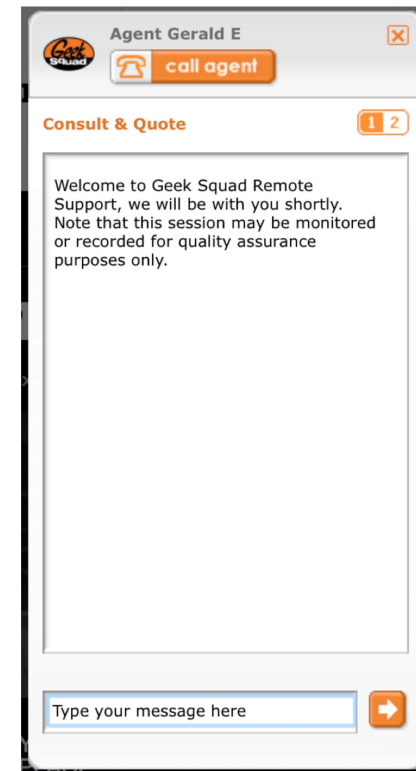
➤ On-line tutorials

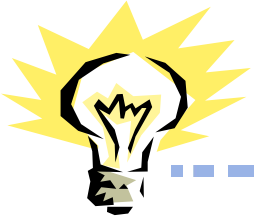
- ❖ Work through simple examples, provide a feel for application



User Support Approaches

- On-line documentation
 - ❖ How much like paper doc?
 - ❖ Electronic can emphasize hypertext, indexing, and searching
- Live help – phone or online chat
 - ❖ Can often see your screen, or even take control of your computer
 - ❖ Need to be online and logged in

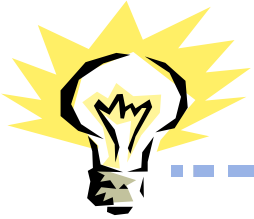




(Display) Medium

- Paper versus monitor?
- People are 15-30% slower reading and comprehending text from a display as compared to paper
- Generational effects

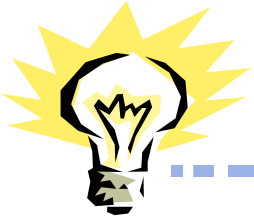




Monitor

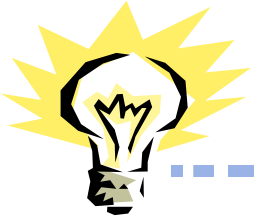
➤ Causes for slow-down

- ❖ Poor fonts (monospace, bad kerning, spacing)
- ❖ Low contrast of letters & background
- ❖ Emitted vs. reflected light (curved tube)
- ❖ Small display -> page turning
- ❖ Distance, placement of monitor
- ❖ Layout and formatting problems
- ❖ Reduced hand and body motion




Presentation Issues

- Integrate with system, don't "add on"
- 1. How is help requested?
 - ❖ Command, button, function, separate app.
 - ❖ Advantages, disadvantages?
- 2. How is help displayed?
 - ❖ Separate window, whole screen?
 - ❖ On top of application, pop-up box?
 - ❖ Command line, button, light bulb...?



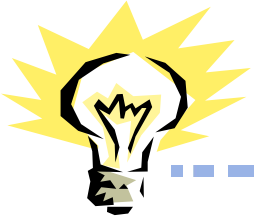
Presentation Issues

➤ 3. Effective presentation of help

- ❖ Design it like any other part of UI
 - language, terminology, jargon, etc.
- ❖ Use active voice
 - “To close a window, place the mouse cursor over the red circle at the upper left corner () and click the mouse button.”

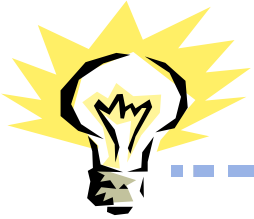
➤ 4. Implementation issues

- ❖ Fast response time is important
- ❖ How is help stored? File, database, ...?



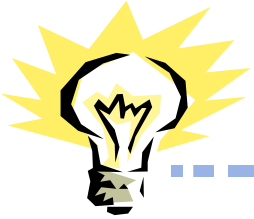
Adaptive Help

- Tailor help level and style to the *particular user*
- Usually requires a system to maintain a *user model*



User Model

- Creating & maintaining a user model
 - ❖ 1. Quantification - Numeric levels of use
 - ❖ 2. Stereotype
 - Novice, intermediate, expert
 - Utilize command use and errors to categorize
 - ❖ 3. Overlay model
 - Build expert user profile with optimal behavior
 - Compare to what user is currently doing



Adaptive Help Issues

➤ Initiative & control

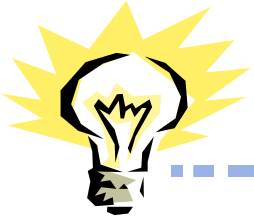
- ❖ Does user feel that control was taken away?
- ❖ “You’re not performing efficiently in this task”

➤ Use

- ❖ Is all this work actually useful?

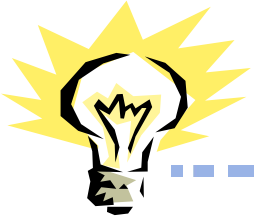
➤ Scope

- ❖ To what aspect of system does it apply?



Doc Organization

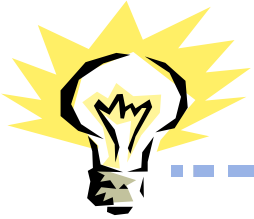
- State educational objectives
- Present concepts in logical sequence, increasing order of difficulty
- Avoid forward references
- Have plenty of examples, complete sample sessions



Doc Organization

- Each concept section:
 - ❖ Explain reason for concept
 - ❖ Describe concept in task-domain terms
 - ❖ Show computer-related semantic concepts
 - ❖ Offer syntax
- Table of contents and index are important
- Keep reading level simple
 - ❖ People liked 5th grade text best

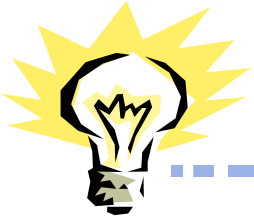
Roemer & Chapanis, CHI '82



Improving Doc

- Run through think-aloud sessions
- Use on-line example tutorials
- Try to predict common states & problems
- Anticipate errors
- Develop manuals early and pilot test
- Iteratively refine

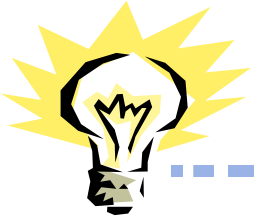
Sound familiar??



Human Characteristics

➤ Don't anthropomorphize

- ✗ “The computer will calculate an answer after you respond”
 - Gives user inaccurate impression
- ✓ “You can get the solution by pressing F1”
 - Better to put user in control



Terminology

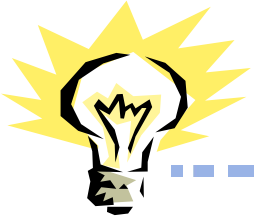
➤ Avoid

- ❖ know, think, understand, have memory
- ❖ ask, tell, speak to, communicate with

➤ Better

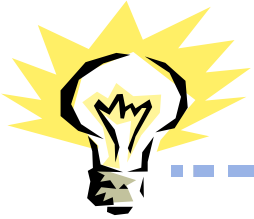
- ❖ process, print, compute, sort, store, search, retrieve
- ❖ use, direct, operate, program, control

But is this the whole story??
Is this always the case??



Help Levels

- 1. Designer model
 - ❖ System designer has model of typical user and builds interface with this in mind
- 2. Adaptable help
 - ❖ User can edit their own model, for example, .profile on UNIX
- 3. Adaptive help
 - ❖ System maintains a user model and can change it on the fly



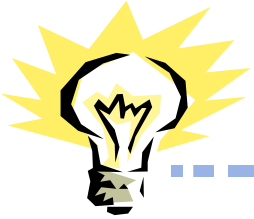
Recommendations

➤ OK

- ❖ All details of each command
- ❖ BNF or formal notation
- ❖ Terse, technical prose

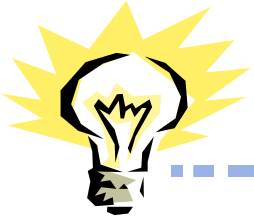
➤ Better

- ❖ Subsets of concepts
- ❖ Lots of examples
- ❖ Readable explanations with a minimum of technical terms

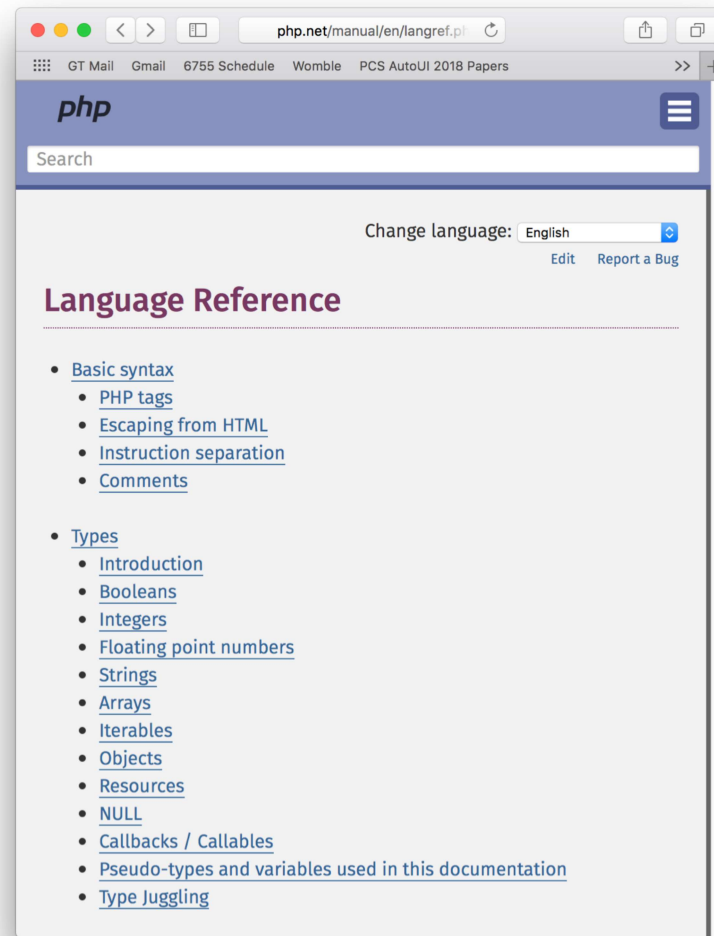


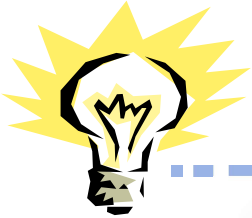
Studies

- Studies have taken documentation and improved it
 - ❖ People did perform better with the improved documentation
- -> Effort here is worthwhile





PHP Language Reference





PHP Language Reference

php.net/manual/en/language.ba:  

GT Mail Gmail 6755 Schedule Womble PCS AutoUI 2018 Papers Arris Gateway - Home ID-Book-offline >> +

php Downloads Documentation Get Involved Help Search

PHP Manual > Language Reference > Basic syntax « Basic syntax Escaping from HTML »

Change language: English [Edit](#) [Report a Bug](#)

PHP tags

When PHP parses a file, it looks for opening and closing tags, which are `<?php` and `>` which tell PHP to start and stop interpreting the code between them. Parsing in this manner allows PHP to be embedded in all sorts of different documents, as everything outside of a pair of opening and closing tags is ignored by the PHP parser.

PHP also allows for short open tag `<?` (which is discouraged since it is only available if enabled using the [short_open_tag](#) `php.ini` configuration file directive, or if PHP was configured with the `--enable-short-tags` option).

If a file is pure PHP code, it is preferable to omit the PHP closing tag at the end of the file. This prevents accidental whitespace or new lines being added after the PHP closing tag, which may cause unwanted effects because PHP will start output buffering when there is no intention from the programmer to send any output at that point in the script.

```
<?php
echo "Hello world";

// ... more code

echo "Last statement";

// the script ends here with no PHP closing tag
```

Basic syntax

- » [PHP tags](#)
- [Escaping from HTML](#)
- [Instruction separation](#)
- [Comments](#)

```
<?php
echo "Hello world";

// ... more code

echo "Last statement";

// the script ends here with no PHP closing tag
```

Changelog

Version	Description
7.0.0	The ASP tags <code><%, %></code> , <code><%=</code> , and the script tag <code><script language="php"></code> are removed from PHP.
5.4.0	The tag <code><?=></code> is always available regardless of the <code>short_open_tag</code> ini setting.

User Contributed Notes 5 notes [add a note](#)

▲ 61 ▼ crazytonyi at gmail dot com 2 years ago

Regarding earlier note by @purkrt :

> I would like to stress out that the opening tag is `<?php[whitespace]`, not just `<?php`

This is absolutely correct, but the wording may confuse some developers less familiar with the extent of the term `[whitespace]`.

Whitespace, in this context, would be any character that generated vertical or horizontal space, including tabs (`\t`), newlines (`\n`), and carriage returns (`\r`), as well as a space character (`\s`). So reusing purkrt's example: