CM0133 Internet Computing

Optimizing and Testing Code

Why do we want to optimize code ?

- Make code execute **faster**.
- Make code use **less memory**.
- Make code **readable, reusable and maintainable** for others.
- Make web pages **visible**.
- Make web pages **user friendly.**
- **Secure** web pages.

Tools

- Code Optimizations
- Profiling
- Web Traffic Analysis Tools
- Guidelines for commenting code
- Web Accessibility
- Testing, testing, testing ...

Optimization

- Software optimization doesn't begin where coding ends – It is ongoing process that starts at design stage and continues all the way through development.
- But "premature optimization is the root of all evil" - Donald Knuth
  - Early optimizations do not necessarily result in a better overall system performance
  - Optimizations can cause badly readable and/or badly maintainable code.

Javascript – Just in Time Compilation

- Javascript was initially a purley interpreted language.
- Nowadays Javascript Engines use a Just in Time (JIT) Compiler
  - Firefox v 3.5 uses **Greasemonkey.**
  - Google's Chrome 2 browser has an JIT engine called **V8.**
  - Apple's Safari has **Nitro** as a JIT compiler.
- JIT compilers translate a script into an efficient executable call whenever the script is called for the first time, but not before that, i.e. just in time.
- Some javascript code forces the Script Engine to switch back into the interpreting mode, e.g. eval().

Javascript

- Define scripts externally and refer to the character encoding:

  <script src=”my_script.js” charset=”utf-8” type=”text/javascript”>

- Minify code e.g. with Firefox “Page Speed” deleting unecessary spaces and line breaks. The resulting code is 20 – 30 % shorter and increases the load time of the script
- Shorten variable and function names http://dean.edwards.name/packer/ - another 25 %

Minification vs Obfuscation

- Reduce the amount of source code to reduce download time.
- Minification deletes whitespace and comments.
- Obfuscation also changes the names of things.
- Obfuscation can introduce bugs.
- Never use tools that cause bugs if you can avoid it.

http://www.crockford.com/javascript/jsmin.html

EXAMPLE: http://www.stunnix.com/prod/jo/

Javascript – Avoid Depth First Search

```
function foo() {
    localVar = document.getElementById('foo');
    return localVar;
}
```

```
function foo() {
    var localVar = document.getElementById('foo');
    return localVar;
}
```

Avoid unnecessary symbol table lookups

```
function doSomething() {
    var a,b,c;
    // …
    a = “A”;
    b = 2;
    c = “zeh”
    // …
}
```

```
function doSomething() {
    var a = “A”;
    var b = 2;
    var c = “zeh”
    // …
}
```

It is better to declare AND assign a value because it does not require an unnecessary lookup in the symbol table.

PHP Profiling

- An important function of code profilers are timing functions:
  - How much time does the execution of a certain code sequence take?
  - How often is the code sequence executed?
  - How much does the code sequence occupy the overall runtime of the script.

```
function getTime() {

   $timer = explode( ' ', microtime() );

   $timer = $timer[1] + $timer[0];

   return $timer;

}
```

http://www.webdesign.org/web-programming/php/benchmark-and-optimize-php-script-speed.14875.html

Profiling Tools

- Profiler may show time elapsed in each function and its descendants
  - number of calls , call-graph (some)
- Profilers use either instrumentation or sampling to identify performance issues
- DEMO – Firebug Page Speed

PHP - Profiling

```
$start = getTime();

    $aUsers = $wms->getUsers( 0, $showType, true,
($_REQUEST['page']*$perPage), $perPage );

$end = getTime();

echo  '<strong>getUsers/getUsersByAbc</strong>: '.round($end - $start,4).'
seconds<br />';
```

http://www.webdesign.org/web-programming/php/benchmark-and-optimize-php-script-speed.14875.html#ixzz0ltiFeEs0

Some PHP Best Practices

- Use static methods when possible, this is 4 times faster
- Echo is faster as print
- Use , instead of . to concatenate a string
- Unset vars to clear memory (especially when using arrays)
- Use full paths in includes and requires, so the server doesn't have to resolve the paths for you

Some PHP Best Practices

- Use strncasecmp, strpbrk and stripos in stead of regex
- It's better to use a select statement then multiple if statements with multiple else statements
- Close database connections when you don't need them anymore
- Incrementing a global var is 2 times slower as a local var

Read more:
http://www.webdesign.org/web-programming/php/benchmark-and-optimize-php-script-speed.14875.html#ixzz0ltj0JUnM

http://phplens.com/lens/php-book/optimizing-debugging-php.php

Mem Usage - Example

```
echo "Stage 1: Mem usage is: ", memory_get_usage(),"\n";
```

Stage 1: Mem usage is: 37712

```
  $arr = array();
  for ($i = 0; $i < 1000000; ++$i) {
    $arr[] = rand();
  }
```

http://www.tuxradar.com/practicalphp/18/1/11

Mem Usage - Example

```
echo "Stage 2: Mem usage is: ", memory_get_usage(), "\n";
```

Stage 2: Mem usage is: 60232136

```
 $foo = 1;

 $bar = 2;


 echo "Stage 3: Mem usage is: ", memory_get_usage(), "\n";
```

Stage 3: Mem usage is: 60232248

http://www.tuxradar.com/practicalphp/18/1/11

Explaination

- Before the script has done anything, PHP is already using 37KB of RAM. This is where the parsed script and other basic components live - there's nothing we can do about that.
- In Stage 2, we have allocated 1,000,000 numbers into the array $arr, using up 57.5MB of RAM (**60232136 / 1024 (Bytes) / 1024 (KiloBytes) = 57.44 MB**).
- By stage 3 we've also got the $foo and $bar variables set to integers, so there's a nominal increase in memory usage. So far, so good.

http://www.tuxradar.com/practicalphp/18/1/11

Mem Usage - Example

```
$foo = $arr;

$bar = $arr;



echo "Stage 4: Mem usage is: ", memory_get_usage(), "\n";
```

```
Stage 4: Mem usage is: 60232248
```

```
$arr = array();

echo "Stage 5: Mem usage is: ", memory_get_usage(), "\n";
```

```
Stage 5: Mem usage is: 60232288
```

http://www.tuxradar.com/practicalphp/18/1/11

Explaination II

- In stage 5 $arr is set to be an empty array, and yet the memory usage barely moves. It goes up a little because a new array structure is allocated empty for $arr, but it's basically negligible.

Mem Usage - Example

```
$bar[] = "hello, world";

echo "Stage 6: Mem usage is: ", memory_get_usage(), "\n";
```

Stage 6: Mem usage is: 104426704

```
$foo = array();

echo "Stage 7: Mem usage is: ", memory_get_usage(), "\n";
```

Stage 7: Mem usage is: 60242672

http://www.tuxradar.com/practicalphp/18/1/11

Explanation III

- In stage 6 we've added an array element to the $bar array, so PHP performs the copy-on-write operation - $bar takes a full copy of the array it was previously pointing to, then adds the new element. At this point, $foo and $bar are point to two different arrays, and $arr is pointing to an empty array.
- In stage 7, $foo is also set to be an empty array, and suddenly there's a huge drop in the amount of memory used as $foo's array gets cleaned up. Note, however, that even though the $bar array is no longer referenced in the rest of the script, it is not garbage collected: PHP holds it in memory all the way until the script finishes.

Lessons learned

- If you want a global scope variable to release its memory, use the unset() function or set it to a different value. Otherwise, PHP will keep it floating around just in case.
- Copy-on-write is your friend, and means that for all intents and purposes arrays are copied by reference in the same way that objects are. The only difference is that if you change the array subsequently, a deep copy is performed.
- The minute you unset() or re-assign a variable, PHP frees its memory. Freeing memory - particularly large amounts - isn't free in terms of processor time, which means that if you want your script to execute as fast as possible at the expense of RAM, you should avoid garbage collection on large variables while it's running, then let PHP do it en masse at the end of the script.

Avoid Memory Leaks

```
function stopWatch() {
    var t0 = new Date();

    function elapsed(s) {
        return ((new Date()).getTime() - t0.getTime()) / s ;
    }
    return elapsed;
}


var t = stopWatch();
```

This requires to learn about the memory management of the used programming language.

Javascript and PHP use a garbage collector.

Summary – Code Optimization

- Identify bottlenecks already in the design stage of your software but be careful not to optimize to early in the development process.
- Learn how to use your tools
  - Compilation
  - Memory Management
  - Programming Language Specificities
- Consider the **Time-Memory Tradeoff** in the light of your application.
  - Memory use can be reduced at the cost of slower program execution (or, vice versa, the computation time can be reduced at the cost of increased memory use)

Visibility

- Create useful high-quality material that is of interest to users.
- Design your website for the blind and deaf, not for spiders or search bots.
  - Search bots can't see visuals or hear sound files.
  - Make your titles, anchor text, and ALT tags descriptive and relevant.
- Present information in more than one way.
  - People have different needs and preferences.
    - cheat sheets (quick reference or short summary)
    - online tutorials
    - problem sets and exercises
    - Quizzes
    - feature time line
    - printable files

http://www.googleguide.com/contentTalk.html

Visibility II

- Design names of pages to reflect what's on the page.
  - Google considers the text in the URL when indexing the page.
    page_6.html -> select_terms.html
    page_12.html -> google_works.html
    page_13.html -> results_page.html
- Include words on your web pages that users are likely to specify in a query when searching for your content.
- Design your site logically.
- Include site maps.
- Link to each page that you want accessible from a search engine.
  e.g.
    links from one page to the next and previous pages
    a table of contents
    a navigation bar
    topic links at the beginning of each part
    summaries
    links to relevant material both from your website and outside sources

http://www.googleguide.com/contentTalk.html

Visibility III

- Strive to keep your pages short and about at most a few topics.
  - A user is more likely to find what she seeks on a short page.
- Sparingly use dynamic content, e.g., JavaScript, Flash, DHTML, etc.
  - Search engine spiders are able to index plain text and html more easily than flashy pages.
  - Flashy pages are more likely to be left out of Google's index and search results.
- Correct misspellings.
  - Users are more likely to search for the correct spelling.

http://www.googleguide.com/contentTalk.html

Visibility IV

- Seek feedback and use it to improve your site.
  - Users and web logs are great sources for feedback.
  - Respond to email quickly
  - Acknowledge those who contribute ideas
- Learn from your logs.
  - Try to figure out how and why users are coming to your site.
  - If you suspect that users may seek information that isn't on your site, consider adding it.

http://www.googleguide.com/contentTalk.html

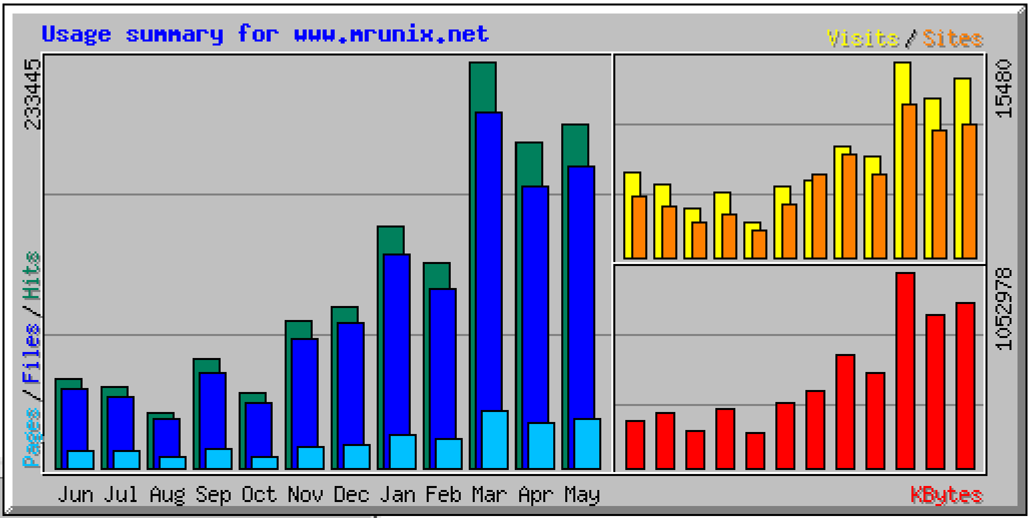
Visibility V

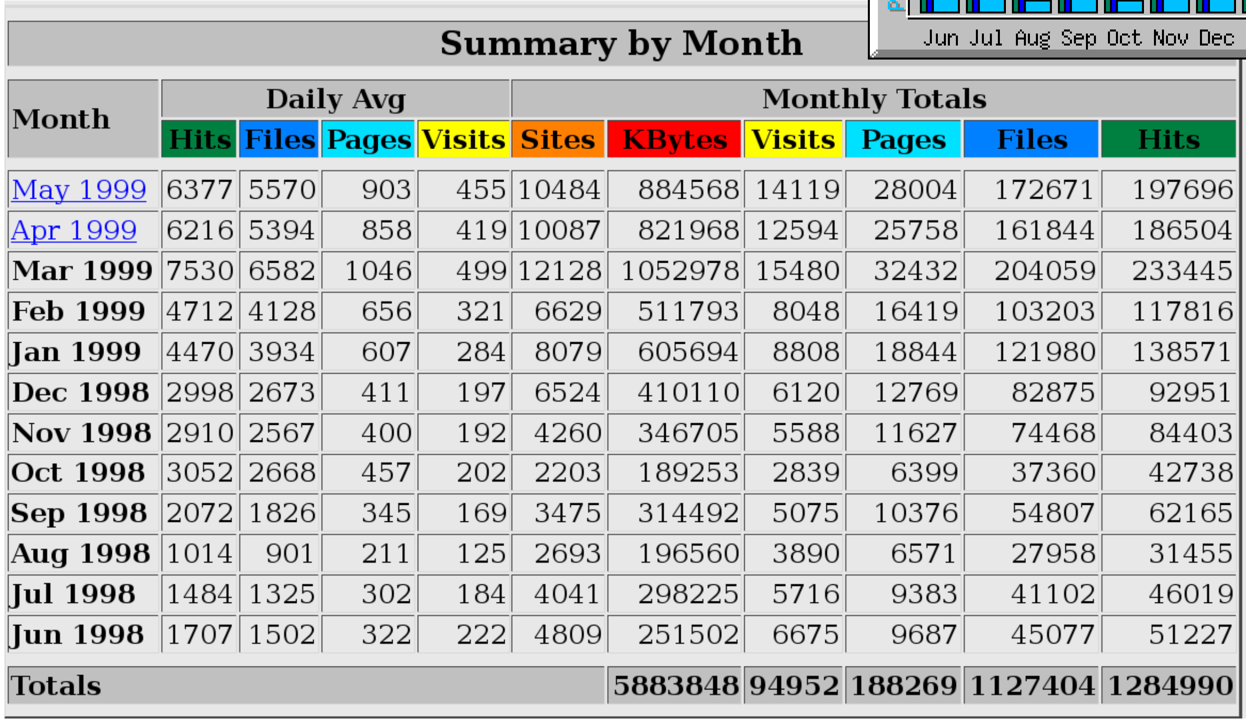
- Submit your site to various web directories and reference sites.
- Publicize your site to everyone with whom you communicate.
- Provide a Rich Site Summary (RSS) aka Really Simple Syndication
  - Make it easy for other sites to distribute your headlines and content.
  - Your RSS feed will be indexed by popular Blog search engines.
- Ask other high quality websites to link to your website.
- Keep your website up.
- Translate your website into foreign languages.

http://www.googleguide.com/contentTalk.html

Web Traffic Analysis

http://www.mrunix.net/webalizer/





Summary by Month

Month	Daily Avg				Monthly Totals					
	Hits	Files	Pages	Visits	Sites	KBytes	Visits	Pages	Files	Hits
May 1999	6377	5570	903	455	10484	884568	14119	28004	172671	197696
Apr 1999	6216	5394	858	419	10087	821968	12594	25758	161844	186504
Mar 1999	7530	6582	1046	499	12128	1052978	15480	32432	204059	233445
Feb 1999	4712	4128	656	321	6629	511793	8048	16419	103203	117816
Jan 1999	4470	3934	607	284	8079	605694	8808	18844	121980	138571
Dec 1998	2998	2673	411	197	6524	410110	6120	12769	82875	92951
Nov 1998	2910	2567	400	192	4260	346705	5588	11627	74468	84403
Oct 1998	3052	2668	457	202	2203	189253	2839	6399	37360	42738
Sep 1998	2072	1826	345	169	3475	314492	5075	10376	54807	62165
Aug 1998	1014	901	211	125	2693	196560	3890	6571	27958	31455
Jul 1998	1484	1325	302	184	4041	298225	5716	9383	41102	46019
Jun 1998	1707	1502	322	222	4809	251502	6675	9687	45077	51227
Totals						**5883848**	**94952**	**188269**	**1127404**	**1284990**

Web Traffic Analysis II

Google Analytics is an example for an online web analysis tool:

You need to sign up for a Google account

1) Inject javascript code into the web pages you want to monitor
2) Google monitors your pages and produces reports and statistics for you based on hourly / daily snapshots.
3) DEMO

Comments

Why do we write comments ?

Comments

- Inappropriate Information
  - Change histories, authorship of code, etc. should be kept in a source code control system, rather than in the comments of your code
- Obsolete Comment
  - Old, irrelevant or incorrect comments are obsolete that should be upgraded or deleted. Best not to write a comment that will become obsolete.
- Redundant Comment
  - i++; // increment i

Comments II

- Poorly Written Comment
  - A comment worth writing is worth writing well.
  - Take time
  - Choose words carefully
  - Use correct grammar and punctuation
  - Don't state the obvious
  - Be brief
- Commented-Out Code
  - DELETE IT and use a source code control system !

Functions

- **Too many arguments**
  - Functions should have a small number of arguments. More than three is questionable already.
- **Output Arguments**
  - Output arguments are counterintuitive. Readers expect arguments to be inputs, not outputs. If your function must change the state of something, have it change the state of the object it is called on.
- **Flag Arguments**
  - Boolean arguments declare that the function does more than one thing. They are confusing and should be eliminated
- **Dead Function**
  - Methods that are never called should be discarded. Keeping dead code around is wasteful. Use a source code control system – CVS, RCS, http://unfuddle.com/

Web Accessibility ?

"The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect."

-- Tim Berners-Lee, W3C Director and inventor of the World Wide Web

(Web Accessibility initiative, 2000)

Web Accessibility - Definition

- Making web pages accessible to all potential users
  - Those with *AND* without disabilities (usability)
  - Using assistive technologies – not just standard web browsers
- Usability and accessibility are intertwined – good accessibility is part of every good design for usability

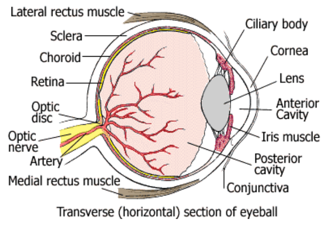
The following categories (which require attention for disabled users) have been identified:

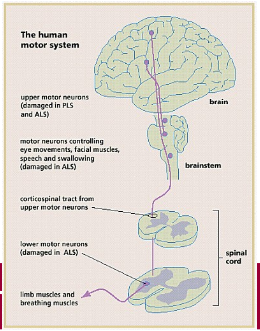
Visual

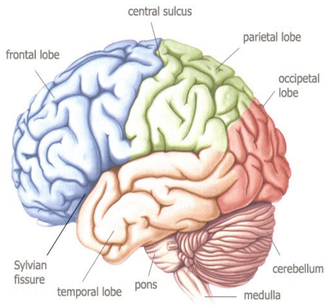
Motor/
Mobility

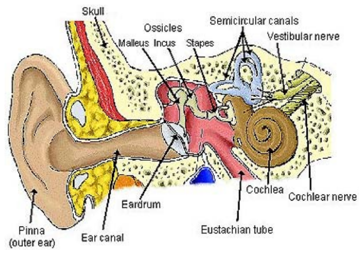
**Cognitive/
Intellectual**

Auditory




The human motor system
upper motor neurons (damaged in PLS and ALS)
brain
motor neurons controlling eye movements, facial muscles, speech and swallowing (damaged in ALS)
brainstem
corticospinal tract from upper motor neurons
lower motor neurons (damaged in ALS)
spinal cord
limb muscles and breathing muscles





The Disability Discrimination Act (DDA)

“The Disability Discrimination Act makes it unlawful for a service provider to discriminate against a disabled person by refusing to provide any service which it provides to members of the public

“**From 1st October 1999** a service provider has to take reasonable steps to change a practice which makes it unreasonably difficult for disabled people to make use of its services.”

“What services are affected by the Disability Discrimination Act? An airline company provides a flight reservation and booking service to the public on its website. This is a provision of a service and is subject to the act.”

“For people with visual impairments, the range of auxiliary aids or services which it might be reasonable to provide to ensure that services are accessible might include ... **accessible websites**.”

“For people with hearing disabilities, the range of auxiliary aids or services which it might be reasonable to provide to ensure that services are accessible might include ... **accessible websites**.

Implications of the DDA

You can be **sued** (as a company) for not catering for disabled users.

The Royal National Institute of Blind People (RNIB) is actively involved in checking web sites

- In Australia the Sydney Olympics was sued by a blind man who was unable to navigate their web site

Of 1000 websites, over 80% did not cater fully for disabled people (DRC -**The Disability Rights Commission** - formal study)

To Comply, follow the W3C's guidelines!!!

PAS 78

A guide to good practice in commissioning accessible websites

- PAS 78 is a **Publicly Available Specification** published on March 8, 2006 by the British Standards Institution (BSI) in collaboration with the Disability Rights Commission (DRC).
- It provides guidance to organisations in how to go about commissioning an accessible website from a design agency.
- It describes what is expected from websites to comply with the UK Disability Discrimination Act (DDA), making websites accessible to and usable by disabled people.
- The principal audience are businesses within the UK, but it is a relevant document for charity and volunteer organisations, as well as local and central government.
- Its also a useful document for web design agencies and web developers as a guide to what is expected of them.

Assistive Technologies

The following assistive technologies are in common use:

Speech Recognition
Audible / Visual

Screen Magnification
Visual

Screen Reader
Visual

Keyboard Overlays
motor control

Translation Software

Web Content Accessibility Guidelines v1

1. Provide equivalent **alternatives** to **auditory** and **visual** content.
2. **Don't rely** on **color** alone.
3. Use markup and **style sheets** and do so properly.
4. **Clarify natural** language usage
5. Create **tables** that **transform** gracefully.
6. Ensure that pages featuring **new technologies transform** gracefully.
7. Ensure **user control** of time-sensitive content changes.
8. Ensure **direct accessibility** of embedded user **interfaces**.
9. Design for **device-independence**.
10. Use **interim** solutions.
11. Use **W3C technologies** and **guidelines**.
12. Provide **context** and **orientation** information.
13. Provide **clear navigation** mechanisms.
14. Ensure that **documents** are **clear** and **simple**.

http://www.w3.org/TR/WAI-WEBCONTENT/

Web Content Accessibility Guidelines v2

Principle 1: Content must be perceivable.

1.1 Provide **text alternatives** for all **non-text** content
1.2 Provide synchronized **alternatives** for **multimedia**
1.3 Ensure that **information** and **structure** can be **separated** from **presentation**
1.4 Make it easy to **distinguish foreground** information from its **background**

Principle 2: Interface components in the content must be operable

2.1 Make **all functionality operable** via a **keyboard** interface
2.2 Allow **users** to **control time limits** on their reading or interaction
2.3 Allow users to **avoid content** that could **cause seizures** due to **photosensitivity**
2.4 Provide **mechanisms** to help users **find content**, orient themselves within it, and navigate
2.5 Help **users avoid mistakes** and make it **easy** to **correct mistakes** that do occur

http://www.w3.org/TR/WCAG20/guidelines.html#perceivable

Web Content Accessibility Guidelines v2

Principle 3: Content and controls must be understandable

- 3.1 Make **text** content **readable** and **understandable**.
- 3.2 Make the **placement** and **functionality** of **content predictable**.

Principle 4: Content should be robust enough to work with current and future user agents (including assistive technologies)

- 4.1 **Support compatibility** with **current** and **future** user agents (including assistive technologies)
- 4.2 **Ensure** that **content** is **accessible** or **provide** an **accessible alternative**

Different Levels Of Conformance

- Priority 1
  - **Mandatory** requirements.
- Priority 2
  - Should satisfy – remove significant barriers to accessing web documents.
- Priority 3
  - May satisfy – improve access.

Web Unit – A collection of information, consisting of one or more resources, intended to be rendered together, and identified by a single Uniform Resource Identifier (such as URLs)

Various Validators to test Accessibility

- http://validator.w3.org/
- http://colorvisiontesting.com/what%20colorblind%20people%20see.htm
- http://www.labnol.org/internet/design/completely-test-website-errors-html-standards/2673/

  http://colorfilter.wickline.org/
- http://www.softwareqatest.com/qatweb1.html#FREE

Literature

- http://progtuts.info/55/php-optimization-tips/
- Common tips for programming and commenting code
- http://www.devtopics.com/13-tips-to-comment-your-code/
- http://www.cprogramming.com/tutorial/comments.html