

Programming Guidelines for *The Empowered Non-Profit*

Thank you for your interest in *The Empowered Non-Profit*. By following these guidelines, the code for ENP will have a consistent look and flow, and will assist us in rapid code development, reduced bug rates, and quicker bug-fix time.

Use Variable Prefixes

Within the ENP the following variable prefixes are used:

<i>Type</i>	<i>Prefix</i>	<i>Example</i>
Boolean	b	\$bUseOutputHeaders
Integer (all types)	l (lower case "L")	\$lLoopCnt
Float	sng	\$sngMyRatio
String	str	\$strUserFirstName
Currency	cur	\$curDonationAmnt
Variant (type unknown)	v	\$vTypeTBD
Date/Time	dte	\$dteToday
Object (class object)	obj	\$objLabelClass

A few exceptions:

- SQL strings are often prefixed with \$sql
- The mySQL row array is usually simply called \$row
- The mySQL row count is usually called \$numrows

Functions that return values should also conform to the naming conventions.

Functions and Subroutines

Programmers of the ENP must know how to use functions and subroutines! Believe it or not, I've worked with "programmers" who didn't understand the concept of functions and would create massive files with horribly nested conditions and huge blocks of duplicated code. Think modular. Try to keep function length reasonable (say under a hundred lines of code).

Error Checking

A great way to avoid deeply buried software bugs is to have incremental error checking whenever possible. It is so much easier to fix a bug at the source, rather than three thousand lines of code downstream. Could this be called “Crash early, crash often”?

- always check for errors resulting from mySQL calls
- always make provisions for mySQL EOF conditions
- use the “default” clause in switch statements to catch unexpected situations (don't allow them to simply fall through”

Constants

Rather than plugging in numerical constants, use the php “define” statement.

Defense Against the Dark Arts

Sadly, there are enough sociopaths in our world that we must take precautions against them in our coding techniques. Please use the following:

- *never, ever* place an unencrypted key ID anywhere that is visible to the user. Always encrypt key IDs first. If a user with nefarious intentions has access to keyIDs, he can manipulate these keyIDs and view information outside his chapter. This is why you see the routine calls to strEncryptID and IDDecryptID throughout the code.
- Remember that “hidden” form values and hidden only in the sense that they don't appear on the screen. They are easily viewable by the user from the HTML source. Don't put secret stuff there.

Comments

Comment your code! We prefer a comment block at the top of each file and after each function definition. Make sure that code and comments are easily distinguishable.

When coding javascript, we sometimes are conflicted because comments increase the download size of the HTML. One way to avoid this is to pop into php interpretation mode in your code, add comments, then pop back out. This way, the comments are in your source code, but are stripped out at the server side and do not effect the download file size.

SQL

The strings that communication between the ENP source code and the mySQL database engine can be made more readable by following these guidelines:

- start each clause on a new line
- all SQL keywords, function names, etc, are capitalized
- use parentheses to clearly define the search criteria. Add parentheses for clarity even if not strictly required by the syntax. Remember, the computer that is parsing the parentheses is executing several billion instructions a second; the programmer trying to debug the SQL statement is running significantly slower!
- Avoid pulling in an entire row when only a few fields are required. Identify the needed fields in the SELECT clause.

Miscellaneous

- Don't use tabs in your source text
- Use proper indenting to accurately reflect the structured nature of your code. We use three spaces per level.

Conclusions

Thanks again for your interest in *The Empowered Non-Profit!*

As the developers of WordPress (<http://wordpress.org/>) say, “Code is poetry”. Think of your code as your Opus Magnus, your legacy, a thing of beauty, and your gift to future generations.

If you have suggestions or enhancements related to these coding suggestions, please email us at empower@enonprofit.net