|  |  |
| --- | --- |
| **PERL BEST PRACTICES** |  |

A list of the complete set of 256 guidelines presented in *Perl Best Practices* by Damian Conway (2005, O’Reilly). The section heading under which each guideline appears is also provided in square brackets.

|  |
| --- |
| 1. **B.1. Chapter 2, Code Layout**  * Brace and parenthesize in K&R style. [Bracketing] * Separate your control keywords from the following opening bracket. [Keywords] * Don't separate subroutine or variable names from the following opening bracket. [Subroutines and Variables] * Don't use unnecessary parentheses for builtins and "honorary" builtins. [Builtins] * Separate complex keys or indices from their surrounding brackets. [Keys and Indices] * Use whitespace to help binary operators stand out from their operands. [Operators] * Place a semicolon after every statement. [Semicolons] * Place a comma after every value in a multiline list. [Commas] * Use 78-column lines. [Line Lengths] * Use four-column indentation levels. [Indentation] * Indent with spaces, not tabs. [Tabs] * Never place two statements on the same line. [Blocks] * Code in paragraphs. [Chunking] * Don't cuddle an else. [Elses] * Align corresponding items vertically. [Vertical Alignment] * Break long expressions before an operator. [Breaking Long Lines] * Factor out long expressions in the middle of statements. [Non-Terminal Expressions] * Always break a long expression at the operator of the lowest possible precedence. [Breaking by Precedence] * Break long assignments before the assignment operator. [Assignments] * Format cascaded ternary operators in columns. [Ternaries] * Parenthesize long lists. [Lists] * Enforce your chosen layout style mechanically. [Automated Layout] |

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |
| --- |
| 1. **B.2. Chapter 3, Naming Conventions**  * Use grammatical templates when forming identifiers. [Identifiers] * Name booleans after their associated test. [Booleans] * Mark variables that store references with a \_ref suffix. [Reference Variables] * Name arrays in the plural and hashes in the singular. [Arrays and Hashes] * Use underscores to separate words in multiword identifiers. [Underscores] * Distinguish different program components by case. [Capitalization] * Abbr idents by prefx. [Abbreviations] * Abbreviate only when the meaning remains unambiguous. [Ambiguous Abbreviations] * Avoid using inherently ambiguous words in names. [Ambiguous Names] * Prefix "for internal use only" subroutines with an underscore. [Utility Subroutines] |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.3. Chapter 4, Values and Expressions**  * Use interpolating string delimiters only for strings that actually interpolate. [String Delimiters] * Don't use "" or '' for an empty string. [Empty Strings] * Don't write one-character strings in visually ambiguous ways. [Single-Character Strings] * Use named character escapes instead of numeric escapes. [Escaped Characters] * Use named constants, but don't use constant. [Constants] * Don't pad decimal numbers with leading zeros. [Leading Zeros] * Use underscores to improve the readability of long numbers. [Long Numbers] * Lay out multiline strings over multiple lines. [Multiline Strings] * Use a heredoc when a multiline string exceeds two lines. [Here Documents] * Use a "theredoc" when a heredoc would compromise your indentation. [Heredoc Indentation] * Make every heredoc terminator a single uppercase identifier with a standard prefix. [Heredoc Terminators] * When introducing a heredoc, quote the terminator. [Heredoc Quoters] * Don't use barewords. [Barewords] * Reserve => for pairs. [Fat Commas] * Don't use commas to sequence statements. [Thin Commas] * Don't mix high- and low-precedence booleans. [Low-Precedence Operators] * Parenthesize every raw list. [Lists] * Use table-lookup to test for membership in lists of strings; use any( ) for membership of lists of anything else. [List Membership] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.4.** [**Chapter 5**](mk:@MSITStore:C:\Users\Martin\Documents\Languages\Perl\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-5.html#perlbp-CHP-5)**, Variables**  * Avoid using non-lexical variables. [Lexical Variables] * Don't use package variables in your own development. [Package Variables] * If you're forced to modify a package variable, localize it. [Localization] * Initialize any variable you localize. [Initialization] * use English for the less familiar punctuation variables. [Punctuation Variables] * If you're forced to modify a punctuation variable, localize it. [Localizing Punctuation Variables] * Don't use the regex match variables. [Match Variables] * Beware of any modification via $\_. [Dollar-Underscore] * Use negative indices when counting from the end of an array. [Array Indices] * Take advantage of hash and array slicing. [Slicing] * Use a tabular layout for slices. [Slice Layout] * Factor large key or index lists out of their slices. [Slice Factoring] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.5.** [**Chapter 6**](mk:@MSITStore:C:\Users\Martin\Documents\Languages\Perl\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-6.html#perlbp-CHP-6)**, Control Structures**  * Use block if, not postfix if. [If Blocks] * Reserve postfix if for flow-of-control statements. [Postfix Selectors] * Don't use postfix unless, for, while, or until. [Other Postfix Modifiers] * Don't use unless or until at all. [Negative Control Statements] * Avoid C-style for statements. [C-Style Loops] * Avoid subscripting arrays or hashes within loops. [Unnecessary Subscripting] * Never subscript more than once in a loop. [Necessary Subscripting] * Use named lexicals as explicit for loop iterators. [Iterator Variables] * Always declare a for loop iterator variable with my. [Non-Lexical Loop Iterators] * Use map instead of for when generating new lists from old. [List Generation] * Use grep and first instead of for when searching for values in a list. [List Selections] * Use for instead of map when transforming a list in place. [List Transformation] * Use a subroutine call to factor out complex list transformations. [Complex Mappings] * Never modify $\_ in a list function. [List Processing Side Effects] * Avoid cascading an if. [Multipart Selections] * Use table look-up in preference to cascaded equality tests. [Value Switches] * When producing a value, use tabular ternaries. [Tabular Ternaries] * Don't use do...while loops. [do-while Loops] * Reject as many iterations as possible, as early as possible. [Linear Coding] * Don't contort loop structures just to consolidate control. [Distributed Control] * Use for and redo instead of an irregularly counted while. [Redoing] * Label every loop that is exited explicitly, and use the label with every next, last, or redo. [Loop Labels] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.6. Chapter 7, Documentation**  * Distinguish user documentation from technical documentation. [Types of Documentation] * Create standard POD templates for modules and applications. [Boilerplates] * Extend and customize your standard POD templates. [Extended Boilerplates] * Put user documentation in source files. [Location] * Keep all user documentation in a single place within your source file. [Contiguity] * Place POD as close as possible to the end of the file. [Position] * Subdivide your technical documentation appropriately. [Technical Documentation] * Use block templates for major comments. [Comments] * Use full-line comments to explain the algorithm. [Algorithmic Documentation] * Use end-of-line comments to point out subtleties and oddities. [Elucidating Documentation] * Comment anything that has puzzled or tricked you. [Defensive Documentation] * Consider whether it's better to rewrite than to comment. [Indicative Documentation] * Use "invisible" POD sections for longer technical discussions. [Discursive Documentation] * Check the spelling, syntax, and sanity of your documentation. [Proofreading] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.7.** [**Chapter 8**](mk:@MSITStore:C:\Users\Martin\Documents\Languages\Perl\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-8.html#perlbp-CHP-8)**, Built-in Functions**  * Don't recompute sort keys inside a sort. [Sorting] * Use reverse to reverse a list. [Reversing Lists] * Use scalar reverse to reverse a scalar. [Reversing Scalars] * Use unpack to extract fixed-width fields. [Fixed-Width Data] * Use split to extract simple variable-width fields. [Separated Data] * Use Text::CSV\_XS to extract complex variable-width fields. [Variable-Width Data] * Avoid string eval. [String Evaluations] * Consider building your sorting routines with Sort::Maker. [Automating Sorts] * Use 4-arg substr instead of lvalue substr. [Substrings] * Make appropriate use of lvalue values. [Hash Values] * Use glob, not <...>. [Globbing] * Avoid a raw select for non-integer sleeps. [Sleeping] * Always use a block with a map and grep. [Mapping and Grepping] * Use the "non-builtin builtins". [Utilities] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.8.** [**Chapter 9**](mk:@MSITStore:C:\Users\Martin\Documents\Languages\Perl\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-9.html#perlbp-CHP-9)**, Subroutines**  * Call subroutines with parentheses but without a leading &. [Call Syntax] * Don't give subroutines the same names as built-in functions. [Homonyms] * Always unpack @\_ first. [Argument Lists] * Use a hash of named arguments for any subroutine that has more than three parameters. [Named Arguments] * Use definedness or existence to test for missing arguments. [Missing Arguments] * Resolve any default argument values as soon as @\_ is unpacked. [Default Argument Values] * Always return scalar in scalar returns. [Scalar Return Values] * Make list-returning subroutines return the "obvious" value in scalar context. [Contextual Return Values] * When there is no "obvious" scalar context return value, consider Contextual::Return instead. [Multi-Contextual Return Values] * Don't use subroutine prototypes. [Prototypes] * Always return via an explicit return. [Implicit Returns] * Use a bare return to return failure. [Returning Failure] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.9. Chapter 10, I/O**  * Don't use bareword filehandles. [Filehandles] * Use indirect filehandles. [Indirect Filehandles] * If you have to use a package filehandle, localize it first. [Localizing Filehandles] * Use either the IO::File module or the three-argument form of open. [Opening Cleanly] * Never open, close, or print to a file without checking the outcome. [Error Checking] * Close filehandles explicitly, and as soon as possible. [Cleanup] * Use while (<>), not for (<>). [Input Loops] * Prefer line-based I/O to slurping. [Line-Based Input] * Slurp a filehandle with a do block for purity. [Simple Slurping] * Slurp a stream with Perl6::Slurp for power and simplicity. [Power Slurping] * Avoid using \*STDIN, unless you really mean it. [Standard Input] * Always put filehandles in braces within any print statement. [Printing to Filehandles] * Always prompt for interactive input. [Simple Prompting] * Don't reinvent the standard test for interactivity. [Interactivity] * Use the IO::Prompt module for prompting. [Power Prompting] * Always convey the progress of long non-interactive operations within interactive applications. [Progress Indicators] * Consider using the Smart::Comments module to automate your progress indicators. [Automatic Progress Indicators] * Avoid a raw select when setting autoflushes. [Autoflushing] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.10.** [**Chapter 11**](mk:@MSITStore:C:\Users\Martin\Documents\Languages\Perl\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-11.html#perlbp-CHP-11)**, References**  * Wherever possible, dereference with arrows. [Dereferencing] * Where prefix dereferencing is unavoidable, put braces around the reference. [Braced References] * Never use symbolic references. [Symbolic References] * Use weaken to prevent circular data structures from leaking memory. [Cyclic References] | |

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |
| --- |
| 1. **B.11.** [**Chapter 12**](mk:@MSITStore:C:\Users\Martin\Documents\Languages\Perl\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-12.html#perlbp-CHP-12)**, Regular Expressions**  * Always use the /x flag. [Extended Formatting] * Always use the /m flag. [Line Boundaries] * Use \A and \z as string boundary anchors. [String Boundaries] * Use \z, not \Z, to indicate "end of string". [End of String] * Always use the /s flag. [Matching Anything] * Consider mandating the Regexp::Autoflags module. [Lazy Flags] * Use m{...} in preference to /.../ in multiline regexes. [Brace Delimiters] * Don't use any delimiters other than /.../ or m{...}. [Other Delimiters] * Prefer singular character classes to escaped metacharacters. [Metacharacters] * Prefer named characters to escaped metacharacters. [Named Characters] * Prefer properties to enumerated character classes. [Properties] * Consider matching arbitrary whitespace, rather than specific whitespace characters. [Whitespace] * Be specific when matching "as much as possible". [Unconstrained Repetitions] * Use capturing parentheses only when you intend to capture. [Capturing Parentheses] * Use the numeric capture variables only when you're sure that the preceding match succeeded. [Captured Values] * Always give captured substrings proper names. [Capture Variables] * Tokenize input using the /gc flag. [Piecewise Matching] * Build regular expressions from tables. [Tabular Regexes] * Build complex regular expressions from simpler pieces. [Constructing Regexes] * Consider using Regexp::Common instead of writing your own regexes. [Canned Regexes] * Always use character classes instead of single-character alternations. [Alternations] * Factor out common affixes from alternations. [Factoring Alternations] * Prevent useless backtracking. [Backtracking] * Prefer fixed-string eq comparisons to fixed-pattern regex matches. [String Comparisons] |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.12.** [**Chapter 13**](mk:@MSITStore:C:\Users\Martin\Documents\Languages\Perl\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-13.html#perlbp-CHP-13)**, Error Handling**  * Throw exceptions instead of returning special values or setting flags. [Exceptions] * Make failed builtins throw exceptions too. [Builtin Failures] * Make failures fatal in all contexts. [Contextual Failure] * Be careful when testing for failure of the system builtin. [Systemic Failure] * Throw exceptions on all failures, including recoverable ones. [Recoverable Failure] * Have exceptions report from the caller's location, not from the place where they were thrown. [Reporting Failure] * Compose error messages in the recipient's dialect. [Error Messages] * Document every error message in the recipient's dialect. [Documenting Errors] * Use exception objects whenever failure data needs to be conveyed to a handler. [OO Exceptions] * Use exception objects when error messages may change. [Volatile Error Messages] * Use exception objects when two or more exceptions are related. [Exception Hierarchies] * Catch exception objects in most-derived-first order. [Processing Exceptions] * Build exception classes automatically. [Exception Classes] * Unpack the exception variable in extended exception handlers. [Unpacking Exceptions] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.13. Chapter 14, Command-Line Processing**  * Enforce a single consistent command-line structure. [Command-Line Structure] * Adhere to a standard set of conventions in your command-line syntax. [Command-Line Conventions] * Standardize your meta-options. [Meta-options] * Allow the same filename to be specified for both input and output. [In-situ Arguments] * Standardize on a single approach to command-line processing. [Command-Line Processing] * Ensure that your interface, run-time messages, and documentation remain consistent. [Interface Consistency] * Factor out common command-line interface components into a shared module. [Interapplication Consistency] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.14.** [**Chapter 15**](mk:@MSITStore:C:\Users\Martin\Documents\Languages\Perl\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-15.html#perlbp-CHP-15)**, Objects**  * Make object orientation a choice, not a default. [Using OO] * Choose object orientation using appropriate criteria. [Criteria] * Don't use pseudohashes. [Pseudohashes] * Don't use restricted hashes. [Restricted Hashes] * Always use fully encapsulated objects. [Encapsulation] * Give every constructor the same standard name. [Constructors] * Don't let a constructor clone objects. [Cloning] * Always provide a destructor for every inside-out class. [Destructors] * When creating methods, follow the general guidelines for subroutines. [Methods] * Provide separate read and write accessors. [Accessors] * Don't use lvalue accessors. [Lvalue Accessors] * Don't use the indirect object syntax. [Indirect Objects] * Provide an optimal interface, rather than a minimal one. [Class Interfaces] * Overload only the isomorphic operators of algebraic classes. [Operator Overloading] * Always consider overloading the boolean, numeric, and string coercions of objects. [Coercions] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.15.** [**Chapter 16**](mk:@MSITStore:C:\Users\Martin\Documents\Languages\Perl\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-16.html#perlbp-CHP-16)**, Class Hierarchies**  * Don't manipulate the list of base classes directly. [Inheritance] * Use distributed encapsulated objects. [Objects] * Never use the one-argument form of bless. [Blessing Objects] * Pass constructor arguments as labeled values, using a hash reference. [Constructor Arguments] * Distinguish arguments for base classes by class name as well. [Base Class Initialization] * Separate your construction, initialization, and destruction processes. [Construction and Destruction] * Build the standard class infrastructure automatically. [Automating Class Hierarchies] * Use Class::Std to automate the deallocation of attribute data. [Attribute Demolition] * Have attributes initialized and verified automatically. [Attribute Building] * Specify coercions as :STRINGIFY, :NUMERIFY, and :BOOLIFY methods. [Coercions] * Use :CUMULATIVE methods instead of SUPER:: calls. [Cumulative Methods] * Don't use AUTOLOAD( ). [Autoloading] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.16.** [**Chapter 17**](mk:@MSITStore:C:\Users\Martin\Documents\Languages\Perl\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-17.html#perlbp-CHP-17)**, Modules**  * Design the module's interface first. [Interfaces] * Place original code inline. Place duplicated code in a subroutine. Place duplicated subroutines in a module. [Refactoring] * Use three-part version numbers. [Version Numbers] * Enforce your version requirements programmatically. [Version Requirements] * Export judiciously and, where possible, only by request. [Exporting] * Consider exporting declaratively. [Declarative Exporting] * Never make variables part of a module's interface. [Interface Variables] * Build new module frameworks automatically. [Creating Modules] * Use core modules wherever possible. [The Standard Library] * Use CPAN modules where feasible. [CPAN] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.17.** [**Chapter 18**](mk:@MSITStore:C:\Users\Martin\Documents\Languages\Perl\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-18.html#perlbp-CHP-18)**, Testing and Debugging**  * Write the test cases first. [Test Cases] * Standardize your tests with Test::Simple or Test::More. [Modular Testing] * Standardize your test suites with Test::Harness. [Test Suites] * Write test cases that fail. [Failure] * Test both the likely and the unlikely. [What to Test] * Add new test cases before you start debugging. [Debugging and Testing] * Always use strict. [Strictures] * Always turn on warnings explicitly. [Warnings] * Never assume that a warning-free compilation implies correctness. [Correctness] * Turn off strictures or warnings explicitly, selectively, and in the smallest possible scope. [Overriding Strictures] * Learn at least a subset of the perl debugger. [The Debugger] * Use serialized warnings when debugging "manually". [Manual Debugging] * Consider using "smart comments" when debugging, rather than warn statements. [Semi-Automatic Debugging] | |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 1. **B.18. [Chapter 19](mk:@MSITStore:C:\\Users\\Martin\\Documents\\Languages\\Perl\\O'Reilly%20-%20Perl%20Best%20Practices.chm::/0596001738/perlbp-CHP-19.html" \l "perlbp-CHP-19), Miscellanea**  * Use a revision control system. [Revision Control] * Integrate non-Perl code into your applications via the Inline:: modules. [Other Languages] * Keep your configuration language uncomplicated. [Configuration Files] * Don't use formats. [Formats] * Don't tie variables or filehandles. [Ties] * Don't be clever. [Cleverness] * If you must rely on cleverness, encapsulate it. [Encapsulated Cleverness] * Don't optimize codebenchmark it. [Benchmarking] * Don't optimize data structuresmeasure them. [Memory] * Look for opportunities to use caches. [Caching] * Automate your subroutine caching. [Memoization] * Benchmark any caching strategy you use. [Caching for Optimization] * Don't optimize applicationsprofile them. [Profiling] * Be careful to preserve semantics when refactoring syntax. [Enbugging] | |

|  |  |
| --- | --- |
|  |  |