Cheat Sheet for comprehensive CIW Perl Specialist

Perl Basics

- **Perl Interpreter**:

- `#!/usr/bin/perl` - Shebang line to specify Perl interpreter.
- `perl script.pl` - Run a Perl script.

- **Comments**:

- `# This is a single-line comment.`
- `=begin comment`

  Multi-line comment.

```
=end comment
```

- **Variables**:

  - **Scalar Variables**:

- `$variable_name` - Holds a single value.
- Example: `$name = "John";`

  - **Array Variables**:

- `@array_name` - Holds a list of values.
- Example: `@fruits = ("apple", "banana", "cherry");`

  - **Hash Variables**:

- `%hash_name` - Holds key-value pairs.
- Example: `%ages = ("John" => 25, "Jane" => 30);`

- **Operators**:

  - **Arithmetic**: `+`, `-`, `*`, `/`, `%`, `**` (exponentiation)

  - **Comparison**: `==`, `!=`, `<`, `>`, `<=`, `>=`

  - **Logical**: `&&`, `||`, `!`

  - **String**: `.` (concatenation), `x` (repetition)

Control Structures

- **Conditional Statements**:

- **if-else**:

```
if ($condition) {
    # code
} elsif ($another_condition) {
    # code
} else {
    # code
}
```

- **unless**:

```
unless ($condition) {
    # code
}
```

- **Loops**:

- **for**:

```
for ($i = 0; $i < 10; $i++) {
    # code
}
```

- **foreach**:

```
foreach $item (@array) {
    # code
}
```

- **while**:

```
while ($condition) {
    # code
}
```

- **until**:

```
until ($condition) {
    # code
}
```

Subroutines and Functions

- **Defining Subroutines**:

```
sub function_name {
    # code
    return $result;
}
```

- **Calling Subroutines**:

```
$result = function_name($arg1, $arg2);
```

- **Passing Arguments**:

- `@_` - Array containing all arguments passed to the subroutine.
- Example:

```
sub add {
    my ($a, $b) = @_;
    return $a + $b;
}
```

Regular Expressions

- **Basic Syntax**:

- `m/pattern/modifiers` - Match pattern.
- `s/pattern/replacement/modifiers` - Substitute pattern.

- **Common Metacharacters**:

- `.` - Any single character.
- `\d` - Any digit.
- `\w` - Any word character (alphanumeric + underscore).
- `\s` - Any whitespace character.
- `^` - Start of string.
- `$` - End of string.

- **Modifiers**:

- `i` - Case-insensitive.
- `g` - Global match.

- `m` - Multiline mode.

- **Examples**:

- `if ($string =~ m/\d{3}/) { ... }` - Match three digits.
- `$string =~ s/foo/bar/;` - Replace "foo" with "bar".

File Handling

- **Opening a File**:

```
open(my $fh, "<", "file.txt") or die "Cannot open file: $!";
```

- **Reading from a File**:

```
while (my $line = <$fh>) {
    chomp $line;
    # process $line
}
```

- **Writing to a File**:

```
open(my $fh, ">", "output.txt") or die "Cannot open file: $!";
print $fh "Hello, World!\n";
```

- **Closing a File**:

```
close($fh);
```

Error Handling

- **die**:

```
die "Error message" if $condition;
```

- **warn**:

```
warn "Warning message" if $condition;
```

- **eval**:

```
eval {
    # risky code
};
if ($@) {
    # handle error
}
```

Modules and CPAN

- **Using Modules**:

```
use ModuleName;
```

- **Installing Modules**:

```
cpan ModuleName
```

- **Creating Modules**:

```
package MyModule;
sub my_function {
    # code
}
1;
```

Advanced Features

- **References**:

  - **Scalar Reference**:

```
$ref = \$scalar;
```

  - **Array Reference**:

```
$ref = \@array;
```

  - **Hash Reference**:

```
$ref = \%hash;
```

- **Anonymous Arrays and Hashes**:

- **Anonymous Array**:

```
$array_ref = [1, 2, 3];
```

- **Anonymous Hash**:

```
$hash_ref = {key1 => "value1", key2 => "value2"};
```

- **Autovivification**:

- Automatically creates nested data structures.
- Example:

```
$hash_ref->{key1}{key2} = "value";
```

Debugging and Profiling

- **Perl Debugger**:

- `perl -d script.pl` - Start the debugger.
- Commands: `n` (next), `s` (step), `p` (print), `q` (quit).

- **Profiling**:

- `use Devel::NYTProf;` - Use NYTProf for profiling.
- `nytprofhtml` - Generate HTML reports.

Best Practices

- **Use strict and warnings**:

```
use strict;
use warnings;
```

- **Use `my` for variable declaration**:

```
my $variable = "value";
```

- **Indentation and Formatting**:

- Consistent indentation for readability.
- Use spaces around operators.

- **Documentation**:

- Use POD (Plain Old Documentation):

```
=head1 NAME
MyModule - Brief description.
=cut
```

Useful CPAN Modules

- **Data::Dumper**:

- `use Data::Dumper;`
- `print Dumper(\%hash);` - Dump data structures.

- **Storable**:

- `use Storable;`
- `store(\%hash, "file.dat");` - Serialize data structures.

- **DateTime**:

- `use DateTime;`
- `my $dt = DateTime->now;` - Handle dates and times.

Examples

- **Simple Script**:

```
#!/usr/bin/perl
use strict;
use warnings;

my $name = "World";
print "Hello, $name!\n";
```

- **File Processing**:

```
#!/usr/bin/perl
use strict;
use warnings;
```

```
open(my $fh, "<", "input.txt") or die "Cannot open file: $!";
while (my $line = <$fh>) {
    chomp $line;
    print "$line\n";
}
close($fh);
```

- **Regular Expression Example**:

```
#!/usr/bin/perl
use strict;
use warnings;

my $string = "The quick brown fox jumps over the lazy dog.";
if ($string =~ m/fox/) {
    print "Match found!\n";
}
```

This cheat sheet provides a comprehensive overview of essential Perl concepts, syntax, and best practices, ensuring you have a handy reference for your Perl programming tasks.

By Ahmed Baheeg Khorshid

ver 1.0