CodingStyle

Coding Style

Perhaps this should be a table of contents, but for now, I am placing all info on this page (we're just getting warmed up).

Recent notes/actionable items:

- Look into Perl::Tidy and Perl::Critique for standards enforcement. We can either try to plug something into cvs checkin/out to "tidy" automagically, or we can say that a pass through one of these modules is a part of the clean-up process each developer goes through. That is, don't automate the process, but put onus on developers to run their code through the module, and modify based upon suggestions.

Brackets

TUSK follows a slight variant of 'K & R' style, ex:

```
if (x == 0) {
    print "zero\n";
}
```

if/[elsif | TUSK:else]

~~Still up for debate: do we "cuddle" the else as below, or give it its own line? Cuddlers don't want to waste the line, others think uncuddling improves readability. Damian Conway in *Perl Best Practices* suggests no cuddle. Perhaps we don't legislate one or the other, but endorse both methods and urge consistency.~~

Update: For consistency, moving forward, let's do the cuddled way for the new code or whenever we refactor the old code.

"Cuddled"

```
if (x == 0) {
    print "zero\n";
} else {
    print "Non-zero\n";
}
```

"Uncuddled"

```
if (x == 0) {
    print "zero\n";
}
else {
    print "Non-zero\n";
}
```

Tabs and Spaces

Use tabs to indicate blocks. (Please set your editor to have tab width as 4 characters)

```
if (1) {
\t-->doSomething();
\t-->if (1) {
\t-->\t-->doSomethingElse();
\t-->}
}
```

Use spaces to separate entities (var names, parens, etc)

```
simple variable declarations

Preferred:
my $var = value;

Incorrect:
my $var=value;

Preferred:
if (x == 3) {
}

Incorrect:
if(x==3){
}
```

Proposed: Use spaces for vertical variable alignment

Perhaps we don't legislate tabs vs. spaces, but again enforce consistency within a file

```
$m->comp('/tmpl/process:manipulate_rows',
----row_type-----=> 'Direct',
----user---------=> $user_id,
----object_name--=> 'TUSK::Case::Test',
----fields_data--=> $rowHash,
----default_data-=> {
--------'master_test_id'-=> $exam_id,
--------'battery_id'-----=> $battery_id
----},
----field_function-=> {
--------'test_id'---=> 'setPrimaryKeyID',
--------'test_name'-=> 'setTitle',
--------'sortorder'-=> 'setSortOrder',
----},
----display_data-=> \@subtest_data
);

$m->comp('/tmpl/process:manipulate_rows',
    row_type     => 'Direct',
    user         => $user_id,
    object_name  => 'TUSK::Case::Test',
    fields_data  => $rowHash,
    default_data => {
        'master_test_id' => $exam_id,
        'battery_id'     => $battery_id
    },
    field_function => {
        'test_id'   => 'setPrimaryKeyID',
        'test_name' => 'setTitle',
        'sortorder' => 'setSortOrder',
    },
    display_data => \@subtest_data
);
```

ToCamelCase or_not_to_camelcase

- CamelCaseNames
- Perl Classes
- Perl Method Names
- JS Method Names
- Underscores or runtogether (no camel case)
- Variable Names
- Debate

- CSS style declarations - style.css has underscores, camel, hyphens, runtogether. underscore is least used. if we adopt a gStyleName or pcsStyleName convention, i think camel case is easier to read, but this is obviously up to debate.
- Mason method names - code standard is underscore... preserve this standard for consistency

Parentheses

1. Use a space around parens when writing a control structure. Ex.: if_(1)_{ }
1. Always use parens, without a space, after a method name, even if no params passed. Ex.: someFx()

-- Main.DanielWalker - 25 Feb 2009