Litecloud

Programming Conventions

A) Variable Names

- Name the variable after what it is – as meaningful as it is.
- Use all lower case letters
- Use '_' as the word separator
- Affix input type prefix if variable will be used to store values from/to input elements and followed by the input type prefixes.
- Use the **Camel Case** convention when affixing both input and data type prefixes followed by the name of the variable.

Input Type Prefixes

- txt - Textbox
- htxt - Hidden Textbox
- cmb - Combobox
- rd - Radio Button
- chx - Checkbox
- btn - Button

Input Data Type Prefixes

- chr - Character/String
- flt - Float
- int - Integer

Example
cmbChrcollection_id
txtIntdisbursed_quantity
flttotal_amount
btnsave

B) Global Variables Names

- Global variables should be prepended with a 'gL'.
- Use the same rule as for variable names.

Example
gLcmbcollection_id
gLtxtdisbursed_quantity
gLtotalamount

C) Array Variable Names

- Array variables should be prepended with a 'aR'.
- Use the same rule as for variable names.

Example
aRcmbcollection_list
aRtxtitems
aRaccount_list

D) Database Result Array-Object Names

- Database result array-object name should be prepended with a 'dbR'.
- Use the same rule as for variable names.

Example
```
aRaccount_list['dbRaccount_list']
aRdisbursement['dbRdisbursement_list']
aRprint_query['dbRprint_query']
```

E) Class Names

- Name the class depends on the action to be performed to make clear what it does.
- Use the **Pascal Case** convention
- No underscores '_' to be used.

Example
```
Accounting()
BankProfile()
Billing()
```

F) Class Attribute Names

- Name the class attribute after what it is – as meaningful as it is.
- Use all lower case letters
- Use '_' as the word separator

Format

(a) Class attributes with less than four(4) collection-pairs(key,value).

Example

```
ClassName('account_id' => 'value1', 'first_name' => 'value2', 'last_name' => 'value3')
ClassName = {account_id : 'value1', first_name : 'value2', last_name : 'value3'}
```

(b) Class attributes with more than three(3) collection-pairs(key,value).

Example

```
ClassName(
    'account_id' => 'value1',
    'first_name' => 'value2',
    'last_name' => 'value3',
    'address' => 'value4'
)
```

```
ClassName = {
    account_id : 'value1',
    first_name : 'value2',
    last_name : 'value3'
    address : 'value4'
};
```

G) Method and Function Names

- Name the method/ function depends on the action to be performed to make clear what it does.
- Use the **Camel Case** convention
- No underscores '_' to be used.

Example
```
handleError()
deleteBankProfile()
userLogin()
```

H) HTML Input Element Names and IDs

- Input element's Name and ID should be named after what it is – as meaningful as it is.
- Affix input type prefix.
- Input element id should be the same as its name.
- Use all lower case letters
- Use '_' as the word separator

Example

```
<input type="text" id="txtcounter" name="txtcounter">
<?php echo form_input( array( 'name' => 'cmdservice_charge', 'id' => 'cmdservice_charge') ); ?>
<input type="button" name="btnsave_project" id="btnsave_project" value="S" title="Save"/>
```

Input Type Prefixes

- txt - Textbox
- htxt - Hidden Textbox
- cmb - Combobox
- rd - Radio Button
- chx - Checkbox
- btn - Button

I) HTML Element Class and IDs

- HTML element's ID should be named after what it is – as meaningful as it is.
- Use all lower case letters
- Use '_' as the word separator
- Affix cL for class or iD for id followed by the HTML element tag name, dash sign (-) and the desire name or id.

Example

```
<table class="cLtable-list">
<div class="cLdiv-account" id="iDdiv-account>
```

J) Braces Policy{}

- Place brace under and inline with keywords:

Example

```
if (condition)                    while (condition)
{                                 {
   …                         ...
}else if(condition)               }
{
   …                         ...
}
```

K) Indention Policy

- Indent as much as needed to help identify block of codes.
- Use at least a Tab instead of spaces.

Example

```
while (x == y)
{
        someThing();
        someThingElse();

        if (some_error)
        {
                doCorrect();
        }else if(condition)
        {
                continueAsUsual();
        }
}
```

L) Leaving Comments

- Always leave comments upon writing a code, whether it is a function, method or a block of code to justify its functionality and to be used for future purposes.

Guidelines:

✔ Each of the developer should leave a comment indicating the starting and ending block of codes to be written.

Example:

```
/* ---------- Start of Programmer's Name Code ---------- */


/* ---------- End of Programmer's Name Code ---------- */
```

✔ In every component or module to be developed, the starting and ending block of code of it should also be determined together with the name of it, the description and the date when it was started.

Example:

```
/* ---------- Start of Module/Component Name ----------
    Date Started: 02-01-14
    Module/Component: (Module/Component Name)
    Description: (State the description and the purpose of the Module/Component as a whole.)
    Functions: (Enumerate all functions and events that were involved.)
        a) firstFunctionName()
            Purpose: (State the purpose of the function in relation with the module/component)
            Process: (* State how does it work within the module/component.)

            Relative Function/s:(* If some functions or events need it, enumerate each function or event which it was called and
            Relative Event/s:       provide a  brief description why it needs to be called.)

           Called Function/s:   ( * Enumerate functions that were called and provide a brief description why those were needed
                                    to be called.)

    Events:
        a) $("#btnevent").click()
            Purpose: (State the purpose of the event in relation with the module/component)
            Process: (* State how does it work within the module/component.
            Function/s: ( * Enumerate functions that were called and provide a brief description why those were needed to be
                        called.)
            Other Event/s: (* Enumerate other events that were involved and provide a brief description how does it work.)

*/

        // codes.....................

/* ---------- End of Module/Component Name ----------  */
```

✔ Functions, methods and set of codes in every event to be written in order to complete a component or module should also be determined.

Example:

```
// +++++++ Start of functionName Function +++++++

/*  Created by: Programmer 1
    Date Created: 02-03-14

    Function Name: deleteRow
    Function Description: (State the description and the purpose of the function.)
    Parameters: 1. first_parameter     -    (State the description.)
                2. second_parameter   -    (State the description.)
    Return Value/s:

    Called function/s:(Enumerate other function/s that were involved and state its purpose.)
    Event/s:  (Enumerate events involve and state its purpose.)

   Modified by: Programmer 2
    Date Modified: 02-03-16
    Modification Description: (State the reason why it needs to be modified.)
    Additional Parameters: 1. first_parameter     -    (State the description.)
                           2. second_parameter   -    (State the description.)
    Return Value/s:
    Additional called function/s:(Enumerate additional called function/s that were involved and state its purpose.)
    Additional Event/s: (Enumerate additional events involved and state its purpose.)

*/

function deleteRow(row_id, row_decription, click_count)
{
      //code here.....
}

// +++++++ End of functionName Function +++++++
```

Example:

```
// +++++++ Start of adding new row Event +++++++

/*  Created by:
    Date Created:
    Event: $("#btnadd_new_row").click()
    Description: (State the description and the purpose of the event.)
    Function/s:(Enumerate functions that were involved and state its purpose.)
    Other Event/s:  (Enumerate other events involved and state its purpose.)

   Modified by:
   Date Modified:
   Modification Description: (State the reason why it needs to be modified.)
   Additional Function/s:(Enumerate additional function/s that were involved and state its purpose.)
   Additional Other Event/s: (Enumerate additional other event/s involved and state its purpose.)
*/

$("#btnadd_new_row").click(function(){
      //code here.....
});

// +++++++ End of adding new row Event +++++++
```

✔ Conditional Statements and Looping Statements purposes should be determined.

Example

```
// State the purpose of every set of conditional and looping statement before each of it.
while (x == y)  // Define every condition that needs to be consider beside it.
{
        some codes.....
}

// State the purpose of every set of conditional and looping statement before each of it.
if (some_error) // Define every condition that needs to be consider beside it.
{
        some codes.....
}else if(condition) // Define every condition that needs to be consider beside it.
{
        some codes.....
}
```

Comment Structure Example: Indentions and line spacing should be observed.

```
/* ---------- Start of Programmer's Name Code ---------- */

    /* ---------- Start of Module/Component Name ---------- */
        Date Started: 02-01-14
            Module/Component: (Module/Component Name)
            Description: (State the description and the purpose of the Module/Component as a whole.)
            Functions: (Enumerate all functions and events that were involved.)
                a) firstFunctionName()
                   Purpose: (State the purpose of the function in relation with the module/component)
                   Process: (* State how does it work within the module/component.)

                   Relative Function/s: (* If some functions or events need it, enumerate each function or event which it was called and
                   Relative Event/s:       provide a  brief description why it needs to be called.)

                  Called Function/s:   ( * Enumerate functions that were called and provide a brief description why those were needed
                                           to be called.)

            Events:
                    a) $("#btnevent").click()
                       Purpose: (State the purpose of the event in relation with the module/component)
                       Process: (* State how does it work within the module/component.
                       Function/s: ( * Enumerate functions that were called and provide a brief description why those were needed to be called.)
                      Other Event/s: (* Enumerate other events that were involved and provide a brief description how does it work.)
    */

        /*  +++++++ Start of functionName Function +++++++
            Created by: Programmer 1
            Date Created: 02-03-14

            Function Name: deleteRow
            Function Description: (State the description and the purpose of the function.)
            Parameters: 1. first_parameter     -    (State the description.)
                        2. second_parameter   -    (State the description.)
           Return Value/s:

           Called function/s:(Enumerate other function/s that were involved and state its purpose.)
           Event/s:  (Enumerate events involve and state its purpose.)

          Modified by: Programmer 2
           Date Modified: 02-03-16
           Modification Description: (State the reason why it needs to be modified.)
           Additional Parameters: 1. first_parameter     -    (State the description.)
                                  2. second_parameter   -    (State the description.)
           Return Value/s:
           Additional called function/s:(Enumerate additional called function/s that were involved and state its purpose.)
           Additional Event/s: (Enumerate additional events involved and state its purpose.)
        */
            function deleteRow(row_id, row_decription, click_count)
            {
                //code here.....
            }
        // +++++++ End of functionName Function +++++++

        // +++++++ Start of adding new row Event +++++++
            /*Created by:
            Date Created:
            Event: $("#btnadd_new_row").click()
            Description: (State the description and the purpose of the event.)
            Function/s:(Enumerate functions that were involved and state its purpose.)
            Other Event/s:  (Enumerate other events involved and state its purpose.)

            Modified by:
            Date Modified:
            Modification Description: (State the reason why it needs to be modified.)
            Additional Function/s:(Enumerate additional function/s that were involved and state its purpose.)
            Additional Other Event/s: (Enumerate additional other event/s involved and state its purpose.)
          */

          $("#btnadd_new_row").click(function(){
                //code here.....
          });
        // +++++++ End of adding new row Event +++++++

    /* ---------- End of Module/Component Name ----------  */

/* ---------- End of Programmer's Name Code ---------- */
```

M) Folders and Folder Names

- Name the folder according to its purpose.
- Use all lower case letters
- Use '_' as the word separator
- Different modules/components should be separated by a folder.
- Name the folder according to the module/component name.
- Affix the folder name prefixes followed by the folder names.

Folder Name Prefixes

- mdf - Model
- vwf - View
- clf - Controller
- jsf - JavaScript

Example

mdf_accounting
vwf_create_disbursement
clf_login
jsf_disbursement

N) File and Filenames

- Name the file according to its purpose.
- Use all lower case letters
- Use '_' as the word separator
- Affix filename prefix.
- Each module/component should have its own controller, model and JS file.

Filename Prefixes

- md - Model
- vw - View
- cl - Controller
- js - JavaScript

Example

md_accounting
vw_create_disbursement
cl_login
js_disbursement