

# DRUPAL 6 LOCALIZATION CHEAT SHEET

v2.0, 2011 January - By Gábor Hojtsy - Check for new versions at <http://hojtsy.hu/>

## Basic localization

To translate a single piece of text:

```
t('Home');
```

## Replacement values

Strings can contain dynamic values, use an array of replacements:

```
t('@name user',  
array('@name' => $name));
```

MARK	PROCESSING
@count	Escaped with <code>check_plain()</code> .
%size	Escaped & formatted with theme ('placeholder').
!html	Insecure! No escaping performed. Use with care.

## Translating to specific language

A specific language can be set:

```
t('Home', array(), 'de');
```

## Plural translation

If a plural version of the string is needed (replacements and specific language can be used additionally):

```
format_plural($count,  
'1 comment',  
'@count comments');
```

## Others using localization

All accept optional language codes.

```
format_date(...);  
format_interval(...);  
format_size(...);
```

## Right to left display support

For a `modulename.css` file in your codebase, add a `modulename-rtl.css` file with overrides. Mark overridden elements with `/* LTR */` for easier recognition. Drupal will add the file automatically to the page.

## Text also translated from

1. Module or theme names, descriptions and package names in `.info` files.
2. Literal permission names from `hook_perm()` implementations in standard array syntax.
3. Literal title and description values from `hook_menu()` and `hook_menu_alter()` implementations in standard array syntax. Title callback and title arguments allow you to modify callback on the title, not on the description.
4. `watchdog()` log type and message values. Also supports replacement strings. Does not support plurals.

## Javascript translation API

Basics of the PHP API are mirrored, replacements are supported, but different language is not:

```
Drupal.t('Home');  
Drupal.t('@name user',  
{ '@name': name });  
Drupal.formatPlural(  
count, '1 comment',  
'@count comments');
```

## Translation in installer

Only basic translation supported, no plural, neither specific language.

```
st('Home');
```

For code that is run both runtime and in installer:

```
$t = get_t();  
$t('Home');
```

```
modulename.css:
```

```
.figure {  
  color: blue;  
  float: right; /* LTR */  
  margin-left: 1em; /* LTR */  
}
```

## Common mistakes

1. Do not attempt to translate dynamic data if at all possible. Avoid `t($value)`. No escaping of the translated string is performed, so could also be a security risk!
2. Do not reuse a string already run through `t()` for `watchdog()` and `drupal_set_message()`. The former disallows `t()`, the second requires calling `t()`, so just use the respective APIs directly.
3. Do not use `t()` in a global context, such as `define('TEXT', t(...))`. The locale system is not yet initialized at that point.

## Verify you are doing it right

1. Use coder module with potx module to receive accurate code review reports of mistakes in use of the API.
2. Use `l10n_client` and browse through pages defined by your module to see strings are properly translatable.
3. If your project is on `drupal.org`, look for parse warnings on `localize.drupal.org` under your module's page.

## Text tips

1. Use consistent terminology, especially with Drupal core.
2. Minimize text (such as in form labels and descriptions), be to the point but clear enough to understand.

```
modulename-rtl.css:
```

```
.figure {  
  float: left;  
  margin-left: 0;  
  margin-right: 1em;  
}
```