**Overriding CSS from modules and themes** <https://drupal.org/node/263967>

**Adding style sheets** <https://drupal.org/node/171209>

**Adding styles through API functions** <https://drupal.org/node/225868>

CSS overview

Most modern web pages use external style sheets to control the presentation of a page. In a traditional static HTML page, a pointer to a style sheet must be manually placed within the HTML code (usually within the page header).

Here's an example:

<link rel="stylesheet" type="text/css" href="/mytheme.css" />

This code simply tells the browser where to find one of the style sheets (mytheme.css) that control the appearance of the page.

To the browser, a page from a Drupal site might look exactly the same: the HTML header has the same kind of pointers to external style sheets. The key difference is that behind the scene, those pointers are added to the HTML automatically. Some styles might come from the theme itself and others might be supplied by various Drupal modules (to provide default styling for the module output).

## Adding stylesheets in Drupal

You can add new style sheets to a theme or module, and you can override a default style sheet from Drupal core, contributed modules, or even another theme. There are two main ways to do this.

You can include the stylesheet using the .info file of a theme or module. This only works when the CSS file is bundled as part of that theme or module. It will cause the stylesheet to be included on every page. Here is an example (from the Zen theme):

stylesheets[all][] = system.base.css

Alternatively, you can inject CSS using the drupal\_add\_css() function, which will allow you to limit the pages where CSS appears, or to use an external stylesheet (for instance, one hosted on a CDN). To do this, call it from an appropriate function in a .module file or in your theme's template.php file.

Here is an example that adds a google font:

function mytheme\_preprocess\_html(&$variables) {

drupal\_add\_css('http://fonts.googleapis.com/css?family=News+Cycle', array('type' => 'external'));

}

# Overriding style sheets from modules and base themes

You can override the style sheet provided by Drupal core, contributed modules or even other themes.

Most modules provide defaults for the presentation of its output. This includes the markup itself and an associated style sheet. (see the explanation on the[*overriding behavior*](https://www.drupal.org/node/173880#theming-overrides)for the markup.) These default styles can be overridden by making changes within your theme.

## Overriding core and contributed module style sheets

To override a core or contributed module style sheet, it must be specified in your theme's .info file. Drupal overrides are cascading and your theme's stylesheets override any that were previously declared. For example, system-menus.css is located at "modules/system/system-menus.css". If you place a file with the same name in your theme's folder and add the following entry to the .info file, the original system-menus.css file will be ignored and your version will be loaded in its place.

stylesheets[all][] = system-menus.css

A few notes:

* Overriding core CSS files will prevent the default "style.css" file from loading. Remember to explicitly define any[*defaults*](https://www.drupal.org/node/171206)when needed. In Drupal 7, style.css does not load unless it is specified in the .info file.
* The themes override must have a matching media type of the original style.
* URLs within the replacement style sheet may need to be corrected. Check the file for any '*url()*' properties or '*@import*' rules to make sure they are pointing to the right resource.
* The order of style sheets listed in the head of the page will change. This may cause some cascading rules to change with it.
* Some core and module style sheets are loaded conditionally. Overriding through .info files will force the file to always be used.
* If only minor changes are required, consider using CSS selectors to override the styles instead of overriding the whole file.
* In Drupal 7, if you would like to override some css files please use*[hook\_css\_alter](http://api.drupal.org/api/function/hook_css_alter/7)*in template.php (see example in seven theme).

Remember to [clear your cache](https://www.drupal.org/node/337176) after making this change!

## Overriding or omitting a base theme style sheet

The following applies to [sub-themes](https://www.drupal.org/node/225125). To prevent a style sheet from a base theme from being carried over to a sub-theme, you can redefine the style sheet inside the .info file. This works the same way as overriding module or core style sheets.

The base theme and the sub-theme must have the same entry:  
  
stylesheets[all][] = masterStyle.css

If the file exists inside the sub-theme, it will be used. Remove the file in the sub-theme to prevent the file from loading altogether.

# Adding style sheets from .info files

This page explains how to add a style sheet using the [.info file](https://www.drupal.org/node/171205) of a theme. To add a style sheet programmatically, see the [API functions](https://www.drupal.org/node/225868) page. Styling themes purely through CSS is possible with the information provided here.

Notes:

* When working with style sheets, make sure that CSS Optimization is disabled.CSS Optimization aggregates all of the style sheets for a site in order to improve performance. When CSS Optimization is enabled, no changes to your style sheets will be reflected on your site until the aggregated styles are cleared.You can enable CSS Optimization again when you're done modifying your style sheets. For Drupal 7 it is located in "Administer > Configuration > Development > Performance".
* The .info file is cached. Adding or removing any styles will not be detected until the cache is cleared and the revised .info is read.(Do not confuse this with the[*theme registry*](https://www.drupal.org/node/173880#theme-registry).) You must[*clear the cache*](http://drupal.org/node/337176)to see the changes.

**Adding style sheets:**

In Drupal 6, by default, a "style.css" file will be used from your theme when no other styles are defined inside the .info file. In Drupal 7, the style.css file will be used only if it is specified in the .info file. Adding other styles is as simple as defining a new 'stylesheets' key with its [media property](http://www.w3.org/TR/CSS21/media.html) and the name of the style sheet. Keep in mind that defining custom styles will prevent the default "style.css" from loading. Remember to explicitly define the default style sheet if your theme uses it.

; Add a style sheet for all media

stylesheets[all][] = theStyle.css

; Add a style sheet for screen and projection media

stylesheets[screen, projection][] = theScreenProjectionStyle.css

; Add a style sheet for print media

stylesheets[print][] = thePrintStyle.css

; Add a style sheet with media query

stylesheets[screen and (max-width: 600px)][] = theStyle600.css

A few notes:

* Note the empty square brackets between the*[media]*and*= styleName.css*. These are always empty and denote that each stylesheet is appended to an array, as in php.
* The order in which the styles are listed in the .info file will reflect the order it is displayed on head of the page.
* The style sheets can be placed in sub-directories, i.e.,*stylesheets[all][] = stylesheets/styleName.css*. Useful for organizing style sheets.
* However, it is recommended that sub-directories be kept to one level, i.e.,*stylesheets[all][] = css/foo/styleName.css*may cause a problem with some templates. Safer is*stylesheets[all][] = css/styleName.css*or*stylesheets[all][] = foo/styleName.css*.
* The word "style" can not be used in the name, like "myown\_style.css". It will conflict with the default style.css. Anyhow, myownstyle.css will work.

**Adding external stylesheets**

To use a stylesheet external to your theme, such as one hosted on a CDN, you cannot use the themes .info file. Instead you can add this in template.php. In Drupal 7 do this as follows:

function mytheme\_preprocess\_html(&$variables) {

drupal\_add\_css('http://fonts.googleapis.com/css?family=News+Cycle', array('type' => 'external'));

}

# Adding CSS to form or page with attachments

On the [previous page](https://www.drupal.org/docs/7/theming/working-with-css/adding-style-sheets-from-info-files) we learned how to add css using the [.info](https://drupal.org/node/171205) file and the [drupal\_add\_css()](https://api.drupal.org/api/drupal/includes!common.inc/function/drupal_add_css/7" \o "drupal_add_css) function. You will recall that the major advantage to drupal\_add\_css() is that you can include CSS only on specific pages to reduce your page load. Unfortunately, if you have a lot of conditional rules for CSS inclusion, this can lead to complex logic in your template.php or in a hook\_init() implementation. Thankfully, there's an easier way!

## Adding css using #attached attribute

Drupal 7 introduced render arrays to improve the separation of content and markup. One of the elements of a render array is the [#attached](https://api.drupal.org/api/drupal/developer%21topics%21forms_api_reference.html/7#attached) attribute. This attribute allows you to declaratively state what CSS needs to be included whenever that array is rendered into HTML.

### Adding CSS to a cached block

If you add css using [drupal\_add\_css()](https://api.drupal.org/api/drupal/includes!common.inc/function/drupal_add_css/7" \o "drupal_add_css) inside [hook\_block\_view()](https://api.drupal.org/api/drupal/modules!block!block.api.php/function/hook_block_view/7" \o "hook_block_view), and then enable block cache, you will discover that the css will be included when the block is first rendered, but not if the block is cached. This is because the drupal\_add\_css() call only happens when the hook\_block\_view() function is called. Using #attached guarantees that the needed resource will be included as part of the rendering process even if the block is being pulled from cache. Under the hood, this happens via a call to drupal\_process\_attached(), which in turn calls drupal\_add\_css() for you.

**Moving from drupal\_add\_css() to #attached**

Initially, the code below will work, but if you enable block caching, or if you have defined your block to be cached in [hook\_block\_info()](https://api.drupal.org/api/drupal/modules%21block%21block.api.php/function/hook_block_info/7), the my.css will fail to be included.

function foo\_block\_view($delta = '') {

if ($delta === 'foo\_delta') {

drupal\_add\_css('path/to/my.css');

$block = array(

'subject' => t('Foo title'),

'content' => t('Lorem ipsum text.'),

);

return $block;

}

}

Changing the code to be cache-friendly is as simple as removing the call to drupal\_add\_css() and adding the #attached property with the resource included in the css property of the array you set as its value.

function foo\_block\_view($delta = '') {

if ($delta === 'foo\_delta') {

$block = array(

'subject' => t('Foo title'),

'content' => array(

'#markup' => '<p>' . t('Lorem ipsum text.') . '</p>',

'#attached' => array(

'css' => array(

'path/to/my.css'

),

),

),

);

return $block;

}

}

You can add internal, external and inline css using this attribute.

**We can also add css to the page in a similar way.**

$content = array();

$content['#attached']['css'] = array(

'path/to/my.css' => array(

'group' => CSS\_AGGREGATE\_THEME,

'weight' => 999

),

);

return $content;

# Adding styles through the API

Adding [styles through the .info file](https://www.drupal.org/node/171209) should be sufficient for most themes. Since the .info file is static, style sheets cannot be added dynamically. Depending on how the theme handles style sheets, it may not matter altogether. When in doubt, use the .info file.

There are two API functions for working with style sheets, [drupal\_add\_css](http://api.drupal.org/api/function/drupal_add_css/6) and [drupal\_get\_css](http://api.drupal.org/api/function/drupal_get_css/6). Here is an example to dynamically add styles sheets.

Change the "template" prefix to the name of your theme.

function template\_preprocess\_page(&$variables) {

$front\_style = path\_to\_theme() .'/front-page.css';

$path\_style = path\_to\_theme() .'/path-'. arg(0) .'.css';

if (file\_exists($front\_style) && $variables['is\_front']) {

$include\_style = $front\_style;

}

elseif (file\_exists($path\_style)) {

$include\_style = $path\_style;

}

if (isset($include\_style)) {

drupal\_add\_css($include\_style, 'theme', 'all', FALSE);

$variables['styles'] = drupal\_get\_css();

}

}

The above example would include the style sheet "front-page.css" on the front page or many others based on the internal path. For example, <http://example.com/admin> would pickup on "path-admin.css".

A few notes:

* Depending on where and when the style is added, drupal\_get\_css may need to be called in order to include the added styles. They are initially retrieved in*[template\_preprocess\_page](http://api.drupal.org/api/function/template_preprocess_page/6)*. See[*Preprocessors and variables*](https://www.drupal.org/node/223430)for details on the order of the preprocessors.
* There is a parameter in drupal\_add\_css to aggregate the added file. Consider disabling it like the above example when the inclusion of the style sheet is very dynamic, since files added to the larger aggregate will force a new aggregated CSS file to be recreated. In effect, it can slow down the retrieval of the page and consume more bandwidth.

### Where To Add Code

This code can be added in the template.php file in your theme directory.  
There may already be a function there called "phptemplate\_preprocess\_page".  
Just include it within the body of your XXX\_\_preprocess\_page function.

 You just have to use the same variable name that is passed in by reference within your function. For example, if your function signature looks like ...  
function template\_preprocess\_page(&$vars)  
then you should use  
$vars   
inside the function.

# Live\_css with .less support

Theming in Drupal has come a long way and the tools now available not only make theming easy but you can style your site very quickly.

This guide will outline the use of the live\_css ( **<http://drupal.org/project/live_css>** ) module and how to implement .less support ( [**http://lesscss.org/**](http://lesscss.org/) ) with the Omega theme ( [**http://drupal.org/project/omega**](http://drupal.org/project/omega) )

## Live\_css:

This module, once installed, adds a tab to the right side of your monitor. When you click on the tab a css development section opens with a drop down list of all the css files available to use. You can then select the one that you need to use and start playing with your theme. The major advantage of this module is that the results of your styling occur as you are typing. This allows very quick theming and with extreme ease you can try a bunch of different style options for elements.  
The configuration page (**admin/config/development/live\_css** ) allows you to select just the style sheets for your theme or all stylesheets as well as changing the development section color styles (which is fun).

CSS caching will need to be disabled to view these changes, which is fairly standard for theming development and you will need to have write permissions to the .css file you want the changes to effect (which you kinda need anyway if you are theming).

## .less:

live\_css comes with .less support so there is no need to add any other modules or tweak any other configuration. You just need to add a .less file to your site. I am outlining the procedure using the Omega theme because Omega it awesome and I highly recommend this theme.

You need to create a subtheme. I want to point out the Drush command here and only the Drush command because of its simplicity. If you do not use Drush - use it! These are the steps I use to create a subtheme using Drush and Omega.

1- drush dl omega omega\_tools

2- drush en -y omega\_tools

3- drush omega-subtheme SUBTHEME-NAME-HERE

Done. Now I have more time to theme.

But I want to use .less. Why? Because it not only makes the css more structured and a lot easier to use for yourself and other themers who might use it but, just in the writing style that you need to use your code becomes a lot smaller, simpler and practical.

So basically I just add a mystyle.less file in the css folder of the subtheme ( the same place where the global.css file is ) but this just adds the file, it is still not being called to the subtheme. In order to do that, we need to inform the theme through the .info file that this file exists and we want to use it.

Open the .info file of your subtheme. YOUR-SUBTHEME-NAME.info and look for:

css[global.css][name] = 'Your custom global styles'

css[global.css][description] = 'This file holds all the globally active custom CSS of your theme.'

css[global.css][options][weight] = '10'

This is how Omega adds the file to the user interface; copy and paste that code and change it with your new .less file.

css[global.less][name] = '.less subtheme file'

css[global.less][description] = 'New .less file for less integration with my subtheme'

css[global.less][options][weight] = '10'

Change global to the name of your .less file and **FLUSH ALL CACHES**

Now your .less file is recognized by Omega. There is just one more thing to do. Because of the very easy to use Omega interface for configuration of the sections and styles for the site, the .less file (or any other .css file) will not be editable straight away. You first need to select that the subtheme will be using the .less file in the theme configuration settings.

admin/appearance/settings/(THEME NAME GOES HERE )

Then select "toggle styles" in the vertical tabs and granted you have flushed all caches your new .less file will be in the options to enable. Check the checkbox and save. BAM! You now have .less integration with the live\_css development environment.

Why is .less important? If you have worked with css3 you will know that to create these new effects the css can start to look very cumbersome. Here is an example creating a gradient with rounded corners and a box shadow.

.element{

background: #1e5799; /\* Old browsers \*/

background: -moz-linear-gradient(top, #1e5799 0%, #7db9e8 100%); /\* FF3.6+ \*/

background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#1e5799), color-stop(100%,#7db9e8)); /\* Chrome,Safari4+ \*/

background: -webkit-linear-gradient(top, #1e5799 0%,#7db9e8 100%); /\* Chrome10+,Safari5.1+ \*/

background: -o-linear-gradient(top, #1e5799 0%,#7db9e8 100%); /\* Opera 11.10+ \*/

background: -ms-linear-gradient(top, #1e5799 0%,#7db9e8 100%); /\* IE10+ \*/

background: linear-gradient(top, #1e5799 0%,#7db9e8 100%); /\* W3C \*/

filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#1e5799', endColorstr='#7db9e8',GradientType=0 ); /\* IE6-9

\*/

border:1px solid

-moz-border-radius-topleft: 5px;

-moz-border-radius-topright: 5px;

-moz-border-radius-bottomright: 33px;

-moz-border-radius-bottomleft: px;

-webkit-border-radius: 5px 5px 33px px;

border-radius: 5px 5px 33px px;

-webkit-box-shadow: 2px 2px 2px 2px #ccc;

-moz-box-shadow: 2px 2px 2px 2px #ccc;

box-shadow: 2px 2px 2px 2px #ccc;

}

Now this is a cool bit of code and can make you site start to look very clean and who does not want to start using the css3 code? But the code above is just for the css3 and we haven't even begun to have to float:left or margin:0 0 0 10px; to make the design just right. So why would you not want to make that bit of code into this?

.element{

.l-gradient;

.l-border-radius;

.l-shadow;

border:1px solid @b-color;

}

Not only it is a lot more compact but by declaring all your css styles at the top of the style sheet.

.l-gradient{

background: #1e5799; /\* Old browsers \*/

background: -moz-linear-gradient(top, #1e5799 0%, #7db9e8 100%); /\* FF3.6+ \*/

background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#1e5799), color-stop(100%,#7db9e8)); /\* Chrome,Safari4+ \*/

background: -webkit-linear-gradient(top, #1e5799 0%,#7db9e8 100%); /\* Chrome10+,Safari5.1+ \*/

background: -o-linear-gradient(top, #1e5799 0%,#7db9e8 100%); /\* Opera 11.10+ \*/

background: -ms-linear-gradient(top, #1e5799 0%,#7db9e8 100%); /\* IE10+ \*/

background: linear-gradient(top, #1e5799 0%,#7db9e8 100%); /\* W3C \*/

filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#1e5799', endColorstr='#7db9e8',GradientType=0 ); /\* IE6-9

}

You can now use it throughout your theme by just calling .l-gradient. You can setup all your colors and simply call them by using @color.

See docs here -><http://lesscss.org/#docs>.

I see a lot of people use sass which is another option instead of .less but I prefer .less and so have outlined here. If you would like to add sass to this document, please just click the edit button.

# Sass techniques and tools

[Sass](http://sass-lang.com/) is a meta-language on top of CSS that’s used to describe the style of a document cleanly and structurally, with more power than flat CSS allows.

Sass provides a simpler, more elegant syntax for CSS and implements various features that are useful for creating manageable stylesheets -- such as variables, functions, nested rules and more. See the [Sass documentation](http://sass-lang.com/) for details.

**Compass** is an open-source Sass framework. It includes common design patterns such as setting vertical rhythm and converting between pixels and ems, as well as tools like automatic browser-prefixing of CSS3 rules. Compass is an extension to Sass, but is not required

Compass is often used with Sass -- and suggested below -- because it can automatically compile your Sass using the compass watch command. For more information, see the [Compass documentation](http://compass-style.org/).

Here is an article on [Sass CSS preprocessor](http://noeticforce.com/sass-css-preprocessor-framework-getting-started-guide) that talks about everything you need to get started with Sass.

Given below are quick instructions on Sass installation -

### Sass installation

#### Linux

1. Sass is written in Ruby. Install Ruby, if you don't already have it installed:

sudo apt-get install ruby1.9.1

2. Install Sass:

sudo gem install sass

3. Install Compass (optional):

sudo gem install compass

#### Windows

1. Sass is written in Ruby. [Install Ruby](http://rubyinstaller.org/downloads/), if you don't already have it installed.

2. Install Sass:

gem install sass

3. Install Compass (optional):

gem install compass

#### Mac OS X

1. Sass is written in Ruby. Install Ruby, if you don't already have it installed:

sudo apt-get install ruby1.9.1

2. Install Sass:

sudo gem install sass

3. Install Compass (optional):

sudo gem install compass

### Creating a project with compass

Go to the directory where you would like to create your project. For a custom theme, this will be sites/all/themes:

cd path/to/where-you-want-your-project

Create a folder with your project name and install your project inside of it:

compass create project-name

**(Optional)** You can modify, relocate or change the name of css and javascript directory by editing config.fb file.

#### Automatically watching and compiling changes with compass

Change directory to be inside your project folder:

cd path/to/project

Use Compass to automatically compile changed Sass files:

compass watch  
or (shortcut)  
compass watch path/to/project

Alternatively, to manually compile:

compass compile [options]

For specific Compass options and more commands see [Compass command line documentation](http://compass-style.org/help/tutorials/command-line/) and [production CSS with Compass documentation](http://compass-style.org/help/tutorials/production-css/).

### Some useful examples of Sass

#### Variables

Use variables to store something like a color or width and then use it throughout your stylesheets. Sass also supports basic math operations with variables.

### .SCSS syntax

$blue: #3bbfce;

$width: 100px;

.main{

color: $blue;

width: $width;

}

.footer {

width: $width / 2;

}

### .CSS output

.main{

color: #3bbfce;

width: 100px;

}

.footer {

width: 50px;

}

#### Nesting in Sass

Sass avoids repetition by nesting selectors within one another. The same thing works with properties.

### .SCSS syntax

#navigation #main-menu ul {

background: #e5e5e5;

border: 1px solid #e2e2e2;

li {

padding: 5px 10px 5px 10px;

&:hover {

background: #ccc;

a {

color: #fff;

}

}

a {

text-decoration: none;

text-shadow: 1px 2px rgb(211, 200, 200);

}

}

}

### .CSS output

#main-menu ul {

background: #e5e5e5;

border: 1px solid #e2e2e2;

}

#navigation #main-menu ul li {

padding: 5px 10px 5px 10px;

}

#navigation #main-menu ul li:hover {

background: #ccc;

}

#navigation #main-menu ul li:hover a {

color: #fff;

}

#navigation #main-menu ul li a {

text-decoration: none;

text-shadow: 1px 2px #d3c8c8;

}

#### Mixins in Sass

Even more useful than variables, mixins allow you to re-use whole chunks of CSS, properties or selectors. You can even give them arguments.

### .SCSS syntax

@mixin left($dist) {

float: left;

margin-left: $dist;

}

#data {

@include left(10px);

}

### .CSS output

#data {

float: left;

margin-left: 10px;

}

# Standard Drupal core styles and classes

Drupal core takes a modular approach to CSS classes for standard page elements. A number of classes occur throughout a Drupal site. This list is meant as a quick crib sheet for remembering which classes mean what and occur where.

Note: themes you download may alter these classes, and add further ones.

### Page elements

.menu

All menu trees get this class, such as the navigation menu.

.block

All blocks. See <http://drupal.org/node/104319> for more on styling blocks.

.links

Lists of links, including Primary and Secondary links in the page header, and also node links and taxonomy terms (see below).

.nowrap (D7)

The role of this class is to prevent text wrapping.

.element-hidden (D7) or .hidden (D8)

The role of this class is to hide elements from all users (both visually and screen-readers). This class should be used for elements which should not be immediately displayed to any user. An example would be a collapsible fieldset that will be expanded with a click from a user. The effect of this class can be toggled with the jQuery show() and hide() functions.

.element-invisible (D7) or .visually-hidden (D8)

The role of this class is to hide elements visually, but keep them available for screen-readers. This class should be used for information required for screen-reader users to understand and use the site, where visual display is undesirable. Information provided in this manner should be kept concise, to avoid unnecessary burden on the user. An example would be the title of a menu.

.element-invisible.element-focusable (D7) or .visually-hidden.focusable (D8)

The element-focusable (D7) or focusable (D8) class extends the element-invisible (D7) or visually-hidden (D8) class to allow the element to be focusable when navigated to via the keyboard. An example would be the "Skip to content" and "Skip to menu" links at the top of most themes.

.container-inline (D7)

Inline items inside the parent element having specified class.

.invisible (D8)

The role of this class is to hide elements from all users but maintain the visual layout (i.e.: a blank space where the element would be if visible).

### Node elements

.node

A wrapper div around all of a node, including its title.

.node-title

The title of the node.

.content

The body of the node. This will include additions other modules make, such as uploaded files or CCK fields.

.links

Applied to any UL that is a list of links, including Primary and Secondary links in the page header, and also node links and taxonomy terms (see below). Node links however get the .links class on their enclosing DIV.

.terms

Taxonomy terms, which also get .links and .inline.

.inline

This is a system class for styling UL items into a horizontal line.

.feed-icon

RSS feed icons, usually at the foot of the page content area.

### Core Block IDs

See [Core Block CSS IDs](http://drupal.org/node/778884) documentation.

# Using .clearfix

Last [updated](https://www.drupal.org/node/778998/discuss) on

19 September 2016

Drupal 6's “clear-block” class was a Drupalism for functionality which is better known by the CSS Community as “clearfix.” Furthermore, using the “block” term was confusing as it is not dependent on Drupal’s block system. The clear-block class has been renamed to clearfix in Drupal 7.

7.x and 8.x

<div class="clearfix">

Here's what Drupal provides by default from system.base.css:

/\*\*

\* Markup free clearing.

\*

\* @see http://perishablepress.com/press/2009/12/06/new-clearfix-hack

\*/

.clearfix:after {

content: ".";

display: block;

height: 0;

clear: both;

visibility: hidden;

}

/\* IE6 \*/

\* html .clearfix {

height: 1%;

}

/\* IE7 \*/

\*:first-child + html .clearfix {

min-height: 1%;

}

# Supporting "right to left" (RTL) languages

Adding support for RTL (Right to Left) languages involves overriding the lateral styles through cascades and naming the file based on the style sheet it is paired to. The inclusion of the RTL style sheet is automated. The inclusion of the file depends on the language settings set for the site.

For example, in the core theme Garland, "style.css" is the main style sheet. It also includes "style-rtl.css" for right to left languages like Arabic, Hebrew or Persian. The inclusion of the two styles always loads with the main style first and the RTL style second. This allows cascading of all the rulesets within the two files without having to worry about specificity in the selectors used in the RTL style.

There is a coding standard to keep the rules organized. Rules that are dependent on the lateral positioning or dimensions should be commented with /\* LTR \*/indicating that the property is specific to a left to right layout. This includes floats, margins, padding, etc. Inline text should flow automatically as long as the theme sets the language direction of the document through the "page.tpl.php" template.

Example base style:

ul.primary-links {

margin-top: 1em;

padding: 0 1em 0 0; /\* LTR \*/

float: left; /\* LTR \*/

position: relative;

}

Corresponding RTL style:

ul.primary-links {

padding: 0 0 0 1em;

float: right;

}

While working with the main CSS file, this makes it easier to spot where changes may be needed in the RTL style.

Note that if your theme[*overrides a module style*](https://www.drupal.org/node/171209#styles-override-module), the associated RTL style will be omitted unless it is present in your theme.