PREFACE

This document was originally produced as part of an effort to standardize WYSIWYG implementations at Stanford University, refined and evolved for badcamp 2011, Drupalcon Denver, and was mildly updated for badcamp 2012.

Really exciting developments are underway for inline editing in Drupal 7 & 8 via the Spark project. Most likely, you are here to solve today's problems on your Drupal 6 or 7 site, so we'll focus on the nuts and bolts of making that work for you, your organization, or your client.

An distribution profile resides on github. The make file is smooth. There's a demo site up at wysiwyw.ideograph.biz that you can log into and play in.

Here are shortlinks to this document you can share:
http://bit.ly/ideograph-wysiwyg or http://bit.ly/d7wysiwyg

For errors and omissions in this document, please use google docs comments. For issues with my code, please create tickets on github.

The purpose of this document is to empower you. If you would prefer to hire me to fix your content editing experience, feel free to reach out.

Andrew Mallis
Principal, Ideograph
http://ideograph.biz

about.me/andrew_mallis

TABLE OF CONTENTS

Project Context
- How to use this document

Modules
- text-input related modules:
- File/image handling:
- SEO/paths:
- Features:

TEXT (INPUT) FORMATS
- 1. adjust the Filtered HTML filter
- 2. Create dedicated input formats for WYSIWYG use
  - 2.1 text editor
    - 2.1.1 ROLES
    - 2.1.2 FILTERS
  - 2.2 advanced text editor
    - 2.2.1 ROLES
  - 2.3 Configure
    - WYSIWYG FILTER
      - What we are trying to achieve, generally:
    - STYLE PROPERTIES
      - ADVANCED RULES
        - CLASS NAMES
        - RULES FOR ELEMENT IDs
        - RULES FOR URLs USED IN INLINE STYPES:
        - SPAM LINK DETERRENT SETTINGS
    - 3. Rearrange the order of your filters

WYSIWYG PROFILE
- Basic Setup
- Buttons & Plugins
  - CKEditor
  - TinyMCE (Drupal 6)
  - CLEANUP AND OUTPUT
  - CSS
    - Define a custom CSS (the easy way)
    - Define custom CSS (the better way)
    - Set some Custom styles
  - Setting body classes on the WYSIWYG iFrame

BETTER FORMATS

Image & File Workflows
- Overview
  - 1. Path and filename sanitization
    - Transliteration
    - Filefield Paths
    - Think about your File Structure
      - File structure within the filepath
      - Refactoring your existing file uploads
  - 2. Files (attachments)
  - 3. Images
    - A. IMAGE FIELD Configuration

B. INLINE IMAGE OPTIONS
INSERT MODULE
FILEFIELD SOURCES
IMCE
IMAGE RESIZE FILTER
Other useful UI modules
REFERENCES

Project Context

Configuring WYSIWYG editors and image handling is a laborious process, riddled with caveats.

The goal of this document is to provide baseline concepts that you can use to inform your WYSIWYG implementation and satisfy your particular use case.

Let's consider the following 3 roles to control permissions:

- contributor
- editor
- administrator

Your site's roles may differ, but likely can be understood in these terms. For example, you may not have a contributor role, but instead want to provide a basic wysiwyg editor experience to authenticated users. Editors and administrators might be one and the same, or you may have a more complex editorial workflow, in which case your deductive skills will prevail.

How to use this document

We're assuming you already have a WYSIWYG implementation on your site, and are well along, but are dissatisfied with some part or another of it. This document aims to help you jump to your pain point and fix it, or move methodically through each step to complete a new configuration. A distribution that exemplifies key concepts outlined here is also available for you to examine.

There are lots of components and configurations to keep track of. Let's go through them methodically...

Modules

Listed are the modules key to our wysiwyg strategy. Some are optional, depending on your workflows and conifigurations. Module dependencies (such as token, libraries) aren't listed here. All modules are for both Drupal 6 and 7, unless noted for 6.x only.

text-input related modules:

wysiwyg – the api that enables 3rd party editors

wysiwyg_filter – text format (input filter in D6) that gives us control over HTML output (think: output filter)

better_formats – define text/input formats per content type (mostly in D7 core now)

File/image handling:

filefield_sources – re-use existing files, upload via URL
insert – adds a button to insert images or file links into textfields from image/file fields
image_resize_filter – creates on-the-fly image-style/imagecache derivatives when inline images are resized
imce – configurable image/file uploader and browser that supports personal & shared directories and quotas
imce_wysiwyg – bridges imce and wysiwyg editors
imagecache [6.x] – create derivative images from a single file for presentation in different contexts
itweak_upload [6.x] – beautifies file upload forms; mostly rolled into D7.

SEO/paths:

pathauto – automatically generates path aliases according to configurable patterns
transliteration – rename files on upload to use only US-ASCII characters
pathologic – input filter that makes image links in content areas relative
filefield_paths – use node tokens in destination paths and filenames

Features:

features
ctools
strongarm
exportables [6.x] – 6.x-2.0-beta1 or later is required
input_formats [6.x] – export input formats in D6
diff – see how a feature is overridden

Here is the drush command you can run to download everything in Drupal 6:

```
$ drush dl wysiwyg wysiwyg_filter better_formats filefield_paths
filefield_sources pathauto transliteration itweak_upload insert
image_resize_filter imagecache imce imce_wysiwyg features ctools strongarm
exportables-6.x-2.0-beta1 input_formats diff
```

TEXT (INPUT) FORMATS

1. adjust the Filtered HTML filter

If you are using Drupal 6, Set FIltered HTML as your default format -- this will keep our forms on the site in line so we don't have to handle a million exceptions. We will use better formats to set default editors for individual content types. I like to use plain text in D7 as a default.

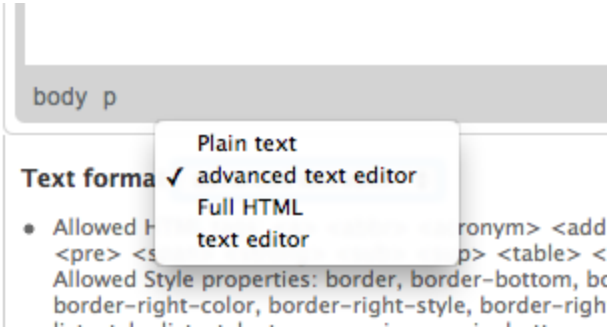
These are the default tags for the *Filtered HTML* filter:
`<a> <em> <strong> <cite> <code> <ul> <ol> <li> <dl> <dt> <dd>`

Which we can optionally modify to include some H tags (leaving out H1), if we like:
`<a> <em> <strong> <cite> <code> <ul> <ol> <li> <dl> <dt> <dd> <h2> <h3> <h4> <h5> <h6> <br>`

2. Create dedicated input formats for WYSIWYG use

We create new input formats for use with the WYSIWYG editor. Most if not all use cases require a more permissive use of the editor by higher-level administrators of the site, and a basic editor for contributors and/or (if you wish) anonymous users. We might want to let only admins add tables, or DIVs, or use IMCE, for example.

The 2 profiles we create appear to users as selectable options under textfields, and should be named descriptively. Let's call our advanced profile "advanced text editor" and our basic one simply "text editor". Avoid the temptation to call your format "WYSIWYG", as it will confuse contributors.



Let's start with the basic text editor configuration first, then extend that example to be more permissive. Where no details are provided, the default options work fine.

2.1 text editor

2.1.1 ROLES

We select our admin, contributor, and editor roles.

2.1.2 FILTERS

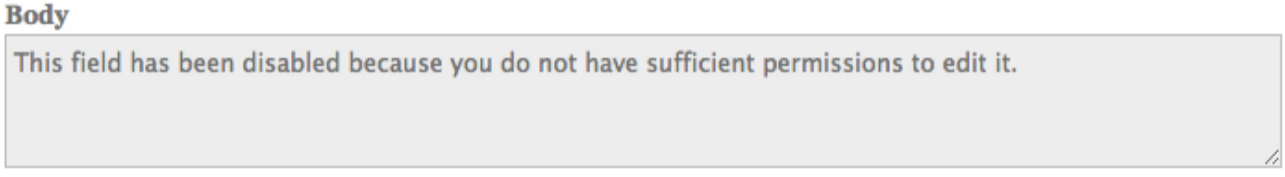
be sure to disable the Line break converter and the HTML filter
enable the HTML corrector and URL filter
enable WYSIWYG Filter

2.2 advanced text editor

2.2.1 ROLES

We select our admin and editor roles.

 *WARNING! Choices here may impact your workflows.*
Any content saved using this text format will only be editable by users with roles permissioned to edit this format. This means that if you go advanced, you lock out contributors from editing. My thinking is that advanced formatting should only be done by advanced users, but these users might not be aware of this repercussion, so there are equally good reasons to allow all content creators access to both formats.



2.3 Configure

WYSIWYG FILTER

The syntax used by this filter is not immediately intuitive. It's worth reading the specification page on the TInyMCE site.

What we are trying to achieve, generally:

- allow classes on all elements, but filter out all those we don't want later using advanced CSS rules
- limit HTML tags
- Convert some legacy tags like `<i>` to `<em>`
- allow or deny properties on elements selectively
- let only the advanced editor post tables and pre-formated content

These are the default settings the module provides you with:

```
a[!href|target<_blank|title],
div[align<center?justify?left?right],
p[align<center?justify?left?right],
br,span,em,strong,cite,code,blockquote,ul,ol,li,dl,dt,dd
```

Here is what we use instead for our simple **text_editor** input format:

```
@[class|style|title],
a[href|target<_blank|name|rel],
-div[align<center?justify?left?right],
#p[align<center?justify?left?right],
strong/b,em/i,u,
-ol[type|compact],-ul[type|compact],-li,-dl,-dt,-dd,
caption,-h2,-h3,-h4,-h5,-h6,hr[size|noshade],
br,cite,code,-blockquote,
img[src|width|height|align|hspace|vspace],
-span[align<center?justify?left?right]
```

And here is what we'll use for the more permissive **advanced_text_editor**:

```
@[class|style|title],
a[!href|target<_blank|name|rel|id],
-div[align<center?justify?left?right],
#p[align<center?justify?left?right],
strong/b,em/i,u,-sup,-sub,
-ol[type|compact],-ul[type|compact],-li,-dl,-dt,-dd,
caption,-code,-pre,address,-h1,-h2,-h3,-h4,-h5,-h6,hr[size|noshade],
```

```
br,cite,code,-blockquote,cite,abbr,acronym,legend,
img[src|width|height|align|hspace|vspace],
-table[border=0|cellspacing|cellpadding|width|frame|rules|height|align|summary
|bgcolor|background|bordercolor],
-tr[rowspan|width|height|align|valign|bgcolor|background|bordercolor],
tbody,thead,tfoot,#td[colspan|rowspan|width|height|align|valign|bgcolor|backgr
ound|bordercolor|scope],
#th[colspan|rowspan|width|height|align|valign|scope],
-span[align<center?justify?left?right]
```

If you really need to allow font selection, resizing, coloring, (yuck!) add:

```
-font[face|size|color]
```

however, classes are way to go.

If you absolutely need to allow IDs, replace the first line with

```
@[id|class|style|title]
```

If you want to disallow inline images, remove:

```
img[src|width|height|align|hspace|vspace]
```

STYLE PROPERTIES

There are lots of options in this section. Here is a middle-of-the-road option that we use for **advanced_text_editor**:

Color and background properties:

- [ ] color
- [ ] background
- [ ] background-color
- [ ] background-image
- [ ] background-repeat
- [ ] background-attachment
- [ ] background-position

Font properties:

- [ ] font
- [ ] font-family
- [ ] font-size
- [ ] font-size-adjust
- [ ] font-stretch
- [x] font-style
- [ ] font-variant
- [x] font-weight

Text properties:

- [x] text-align
- [x] text-decoration
- [ ] text-indent
- [ ] text-transform
- [ ] letter-spacing
- [ ] word-spacing
- [ ] white-space
- [ ] direction
- [ ] unicode-bidi

Box properties:

- [x] margin
- [x] margin-top
- [x] margin-right
- [x] margin-bottom
- [x] margin-left
- [x] padding
- [x] padding-top
- [x] padding-right
- [x] padding-bottom
- [x] padding-left

Border properties (1):

- [x] border
- [x] border-top
- [ ] border-right
- [x] border-bottom
- [ ] border-left
- [ ] border-width
- [ ] border-top-width
- [ ] border-right-width
- [ ] border-bottom-width
- [ ] border-left-width

Border properties (2):

- [x] border-color
- [ ] border-top-color
- [ ] border-right-color
- [ ] border-bottom-color
- [ ] border-left-color
- [x] border-style
- [ ] border-top-style
- [ ] border-right-style
- [ ] border-bottom-style
- [ ] border-left-style

Dimension properties:

- [ ] height
- [ ] line-height
- [ ] max-height
- [ ] max-width
- [ ] min-height
- [ ] min-width
- [ ] width

Positioning properties:

- [x] bottom
- [ ] clip
- [x] left
- [ ] overflow
- [x] right
- [x] top
- [ ] vertical-align
- [ ] z-index

Layout properties:

- [ ] clear
- [ ] display
- [ ] float
- [ ] position
- [ ] visibility

List properties:

- [x] list-style
- [ ] list-style-image
- [ ] list-style-position
- [x] list-style-type

Table properties:

- [ ] border-collapse
- [ ] border-spacing
- [ ] caption-side
- [ ] empty-cells
- [ ] table-layout

User interface properties:

- [ ] cursor
- [ ] outline
- [ ] outline-width
- [ ] outline-style
- [ ] outline-color
- [ ] zoom

For the basic **text_editor**, remove all “box properties” to be safe.

ADVANCED RULES

CLASS NAMES

You need to correlate the class names here with those used in the WYSIWYG profile or they'll get stripped out. These values won’t get presented to the user, rather, you’ll make human readable names for them, so think about what makes sense for you semantically and create a comma-separated list:

```
node-section,
node-subsection,
img-right,
img-left,
img-center,
container
```

If your editors are floating images, a “container” that creates a div with the clearfix (or clear-block) class can be useful in preventing images from overlapping the node links on a short post, or other nodes below when presented in lists.

There is no option to allow all class names, but you can do something funky-awkward like this to allow all lower-case class names if you really need to:

```
a*,b*,c*,d*,e*,f*,g*,h*,i*,j*,k*,l*,m*,n*,o*,p*,q*,r*,s*,t*,u*,v*,w*,x*,y*,z*
```

RULES FOR ELEMENT IDs

Really? You want your users to have ID level control over elements. Leave this blank.

Unless you plan on using named anchors in HTML 5. The name tag works in HTML 4, but is now depreciated in favor of id for anchor tags. In order to not be too liberal, ensure others’ id tags are stripped out, consider carving out a namespace prefix. First, we adjust better formats:
`a[!href|target<_blank|name|rel|id]`
Then add our prefix to the rules, followed by a wildcard, i.e.

```
ideograph*
```

RULES FOR URLs USED IN INLINE STYPES:

set this to /*
so we are only linking to background images on our domain.

SPAM LINK DETERRENT SETTINGS

Set a value of **Whiltelist** here, so that `rel=”nofollow”` is added to all but the domains we like.
Under the **Domains list** we our domains. Subdomains are automatically covered:

```
mydomain.com
myotherdomain.org
```

3. Rearrange the order of your filters

URL filter first, always

HTML Corrector next
Click "show row weights" and manually set the weight very light, in case you install new modules with input filters – they'll still run first

All macros and filters that take bracketed strings like [myformat] go next (i.e. GMap, Lightbox)

WYSIWYG filter must be last!! (it is our last line of defense, after all)
Click "show row weights" and manually set the weight to be as heavy as you can.

 *If using Pathologic, Image Resize Filter must be run AFTER the Pathologic filter too, since Pathologic must correct image path locations for Image Resize Filter to find the images.*

Also note that if you are using Media module to insert video assets in the body of your text, you may want to run it AFTER WYSIWYG filter, since it may be popping iframes in there.

WYSIWYG PROFILE

The WYSIWYG module provides clear instructions on how to install the various editors it supports at **http://yoursite.com/wysiwyg/admin/settings/wysiwyg**

This is also where you assign profiles to input formats.

Once the assignment is made, you can set up options for your WYSIWYG editor of choice.

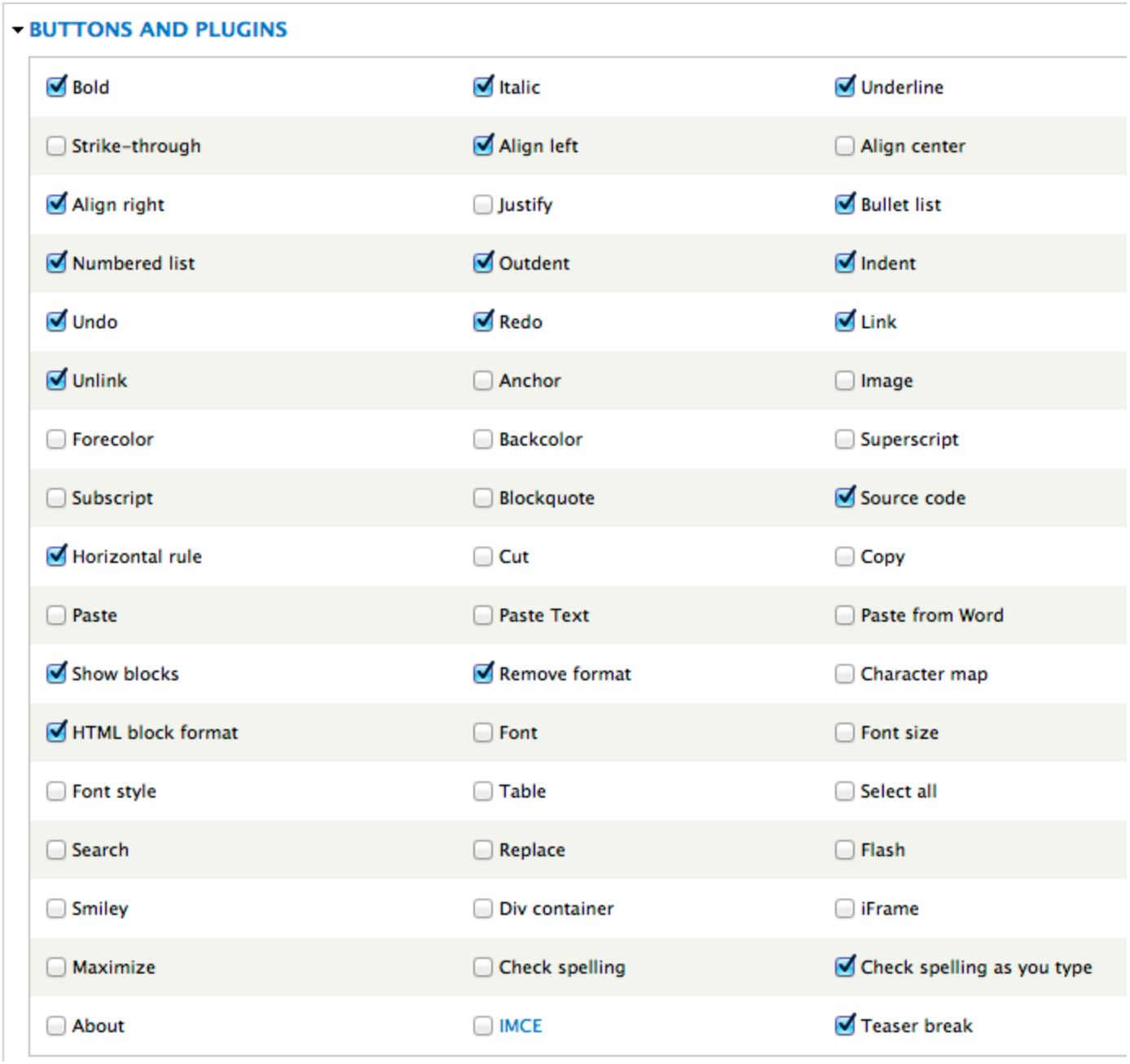
Beware – you cannot change a WYSIWYG editor assignment once a profile is created. You will need to delete it and reconfigure everything again. Don't worry, deleting only removes the associated editor selection, not the input format itself.

Basic Setup

Less is more – uncheck *Show enable/disable rich text* toggle link. Instead either check the "Source code" button. CKEditor kills TinyMCE in user experience on this point.

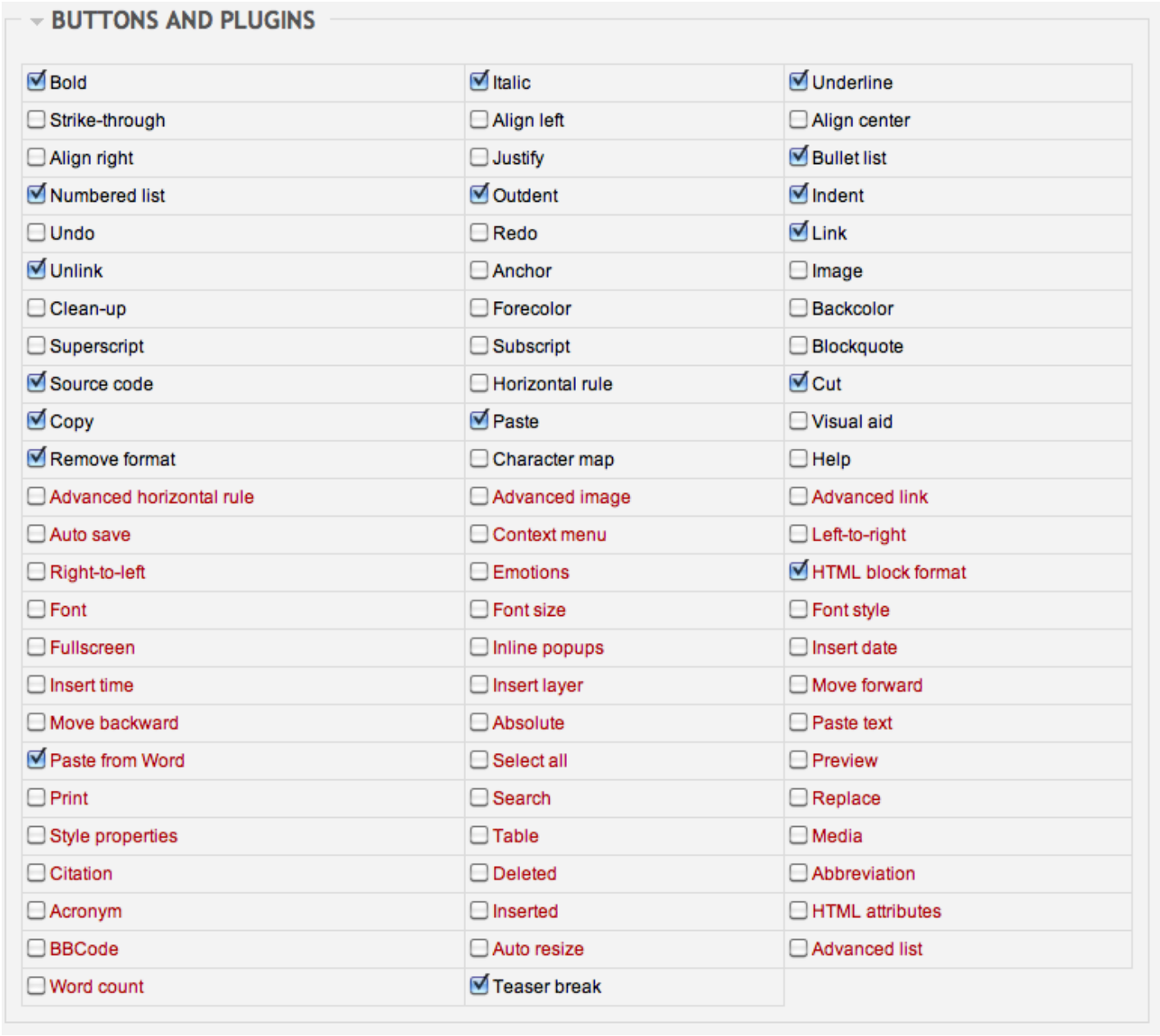
Buttons & Plugins

CKEditor



and we enable a couple more options for the **advanced editor**. Don't center your text. Use a class instead. It's classier. Increasingly, I've been unchecking HTML block format on the basic editor, and valuing instead font styles, which will wrap elements in block-level formats anyway.

TinyMCE (Drupal 6)



CLEANUP AND OUTPUT

- uncheck Preformatted
- optionally, uncheck Remove linebreaks
- Apply source formatting should be checked so flipping to code view is actually useful

CSS

You can re-arrange the order of the block formats so the dropdown is more pleasant. (Who needs

Address anyhow?) We'll be using the editor button for blockquotes, so no need in the dropdown. A webpage should only contain a single h1; we should never prompt for h1 tags in the body.

```
p,h2,h3,h4,h5,h6,pre
```

Define a custom CSS (the easy way)

Define your own CSS that rips from the wysiwyg default css, but adds some basic html styles and alignment from the theme styles. This way, what you SEE in the WYSIWYG will be more like what you actually GET on save.

use the full path:

```
path/to/theme/css/wysiwyg.css
```

or use the %t to find the current theme automatically:

```
%t/theme_name/css/wysiwyg.css
```

If you want to avoid raw cutting and pasting from your main stylesheet(s), and therefore minimize the risk of drifting configurations, you can split your html styles to re-purpose them inside of the wysiwyg.css using an @import statement to include it instead.

```
@import ../styles/html.css
```

However, this can cause issues with CSS aggregation.

Define custom CSS (the better way)

Better still, we define multiple sheets:

```
%t/mytheme/css/html-reset.css,/%t/mytheme/css/nodes.css,%t/mytheme/css/w
ysiwyg.css
```

If we are using an admin theme, we cannot use the supplied tokens, like the previous option, so we define the full path to the admin them.

```
/sites/all/themes/mytheme/css/html-reset.css,/sites/all/themes/mytheme/c
ss/nodes.css,/sites/all/themes/mytheme/css/wysiwyg.css
```

You can include stylesheets from any path here. If you have a sub-theme, you might need some CSS from the parent and some from the subtheme.

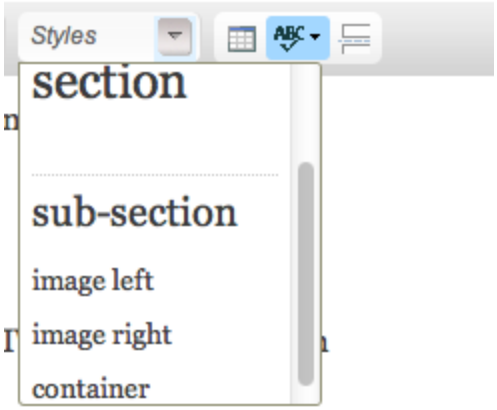
We try to include only what's required for HTML entities and layout of images inside the text areas. If you have one big CSS file, that's ok too.

Inside `wysiwyg.css` we can set some overrides. Namely, we need to re-define the body class font and

line-height so our font sizes are accurate; wysiwyg editors load a complete page in an iFrame.

Set some Custom styles

We can make sensible styles available to the user in a drop down with CKeditor:



Don't forget to define these in your theme's css, and make sure you are pointing to this stylesheet in your wysiwyg profile so the user sees immediate feedback.

You need to define the HTML entity that will wrap your selection using the **dot notation** using the format [name appearing to user]=[html.class] where html is the wrapper element that will be injected around the element you've selected:

```
title=h2.node-title
subtitle=h3.node-subtitle
section=h4.node-section
sub-section=h5.node-subsection
image left=div.img-left
image right=div.img-right
image center=div.img-center
container=div.clearfix
```

In some cases, like for the basic editor, you can consider not exposing block formats at all, and just running with your own custom styles. Users will tend to select "subtitle" regardless of the font-size. But, if they feel an h3 is looking small, they'll just bump up to an h2, which isn't the semantic markup that best optimized your content for screen readers and search engines.

Custom class name for CKEditor won't work without the patch here [http://drupal.org/node/746524#comment-4730732]. This patch is (finally, after 3 years of discussion) being committed to both 6.x-2.x-dev and 7.x-2.x-dev. The last stable tag was over a ago, so either apply the patch to 7.x-2.1 or 6.x-2.4, or roll with -dev [TODO: test and recommend].

Setting body classes on the WYSIWYG iFrame

This was how we reset the content area in a wysiwyg.css that is loaded only when the editor loads:

```
body.cke_show_borders,
body.mceContentBody {
  font-size: 1em; /* reset font size */

  /* just in case the theme sets a background on the body tag */
  background: none #FFF;

  margin: 10px; /* add a bit of spacing so we aren't right to the edge */
}
```

To get even more accurate previews, we should add a class of "node" to the WYSIWYG body tag so that we inherit the cascade of styles defined under this class, i.e. `.node h1 { font-size:2em; }`

If you make heavy use of wysiwyg text areas in blocks, you can consider adding a class of content. Test.

We'll need a custom module to modify the library defaults that contains the following function:

```
function mymodule_wysiwyg_editor_settings_alter(&$settings, $context) {
        if ($context['profile']->editor == 'ckeditor') {
          $settings['customConfig'] = base_path() . drupal_get_path('module',
      'my_custom_module') .'/ckeditor-config.js';
        }
        else if ($context['profile']->editor == 'tinymce') {
            $settings['body_class'] = 'node';
        }
      }
}
```

and in the .js we set:

```
CKEDITOR.editorConfig = function( config ) {
  config.bodyClass = 'node';
};
```

I have created a sandbox project on D.O. that does just this:
http://drupal.org/sandbox/Andrew_Mallis/1315584

inspiration: http://drupal.org/node/160657

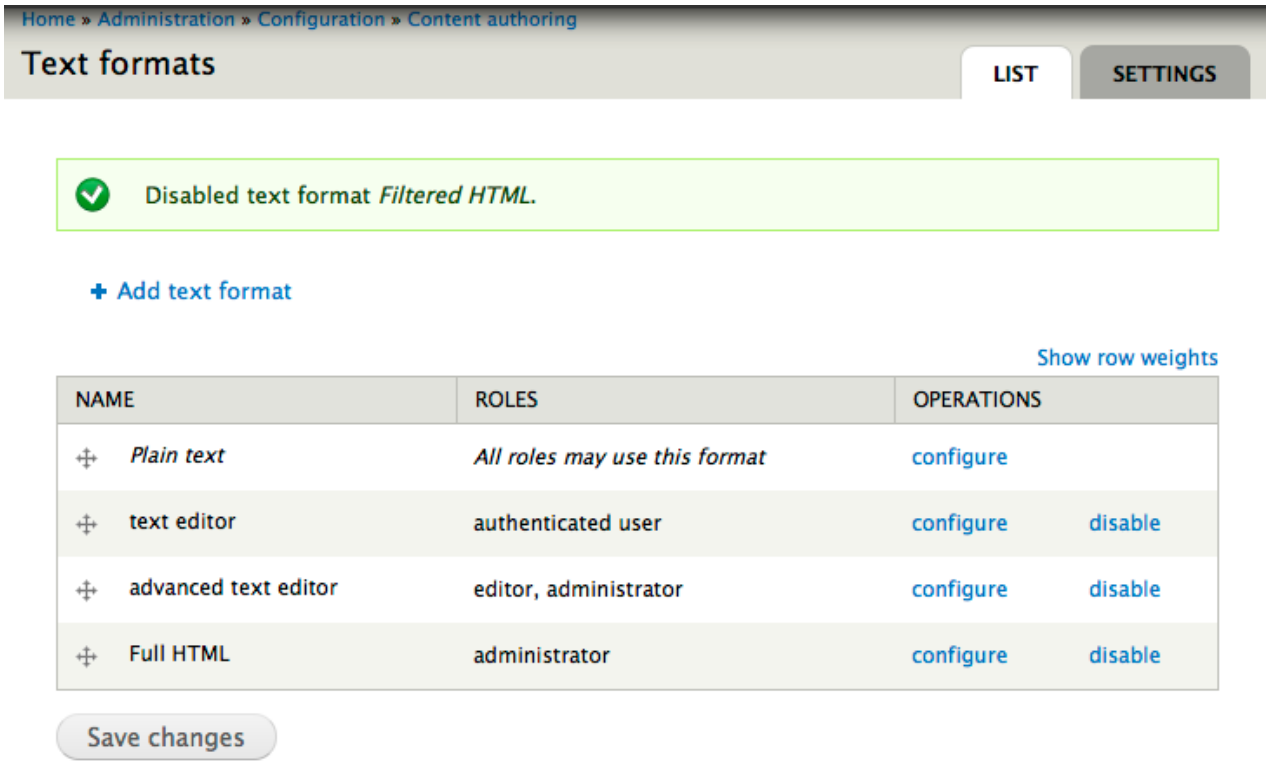
If you do not use git (you should), here are snapshots you can download the module directly:
http://drupalcode.org/sandbox/Andrew_Mallis/1315584.git/snapshot/refs/heads/7.x-1.x-dev.tar.gz
http://drupalcode.org/sandbox/Andrew_Mallis/1315584.git/snapshot/refs/heads/6.x-1.x-dev.tar.gz

https://drupal.org/node/1576296 explains how you can grab such snapshots.

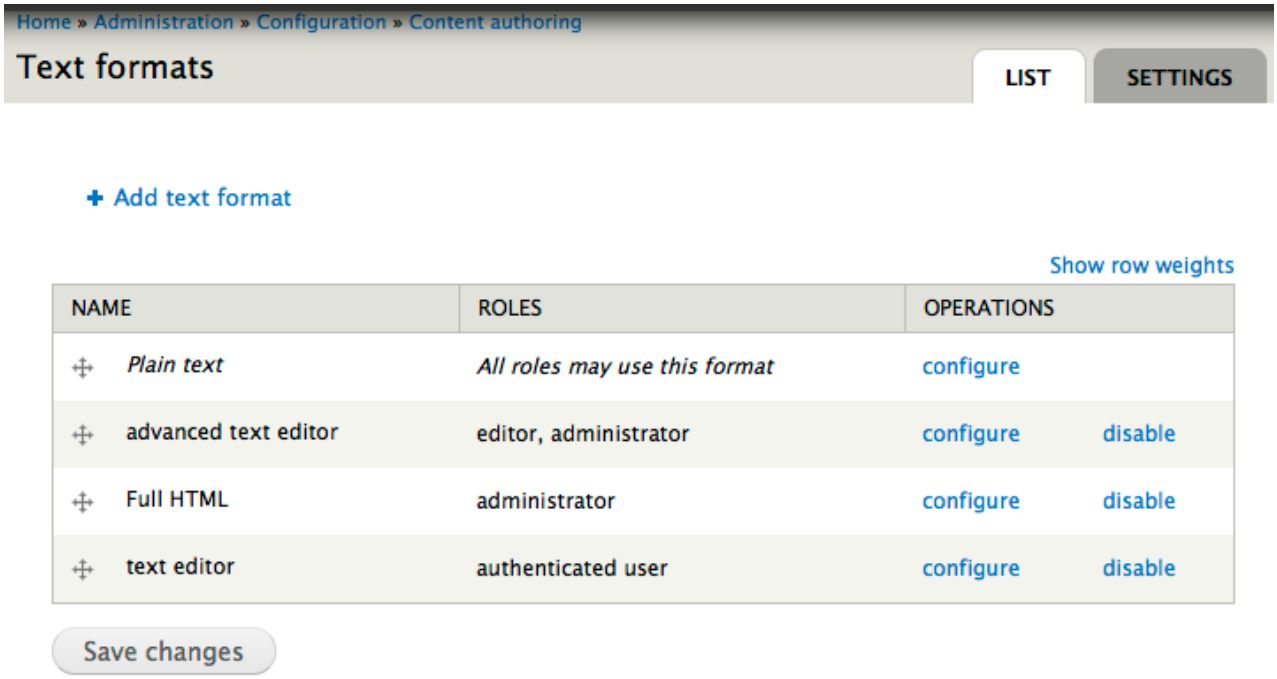
Because of how the drupal.org infastructure creates these packages, you will need to extract from this archive the directory which contains the module and rename it to ideograph_wysiwyg. Place it in a sensible location, like sites/all/modules/contrib/.

BETTER FORMATS

Default text formats are set globally in Drupal 7 core. The first text format listed that a user's role has access to will be the default one used on node comment forms.



Make sure to move authenticated user role to the bottom of the list or it will override our role-specific settings. [is this just a D6 thing?]



In D7, Better Formats also allow us to set THE preferred input format for each textfield on its edit form. Edit your content type, then the field (most usually the body) and set your format.

In D6, this setting is on the content type edit form.

Common use-case for different filters per content type: a Page content type uses the advanced text editor by default, a Blog: text editor, Notifications: Plain text.

Image & File Workflows

Overview

Existing sites have defined Users, Roles, Content Types, Image Styles/Imagecache presets, and Field implementations. Our goal here is to first create a baseline file and image handling strategy that makes sense in the context of a fresh Drupal installation, then to consider how to adapt our Field definitions and other configurations in existing implementations.

1. Path and filename sanitization

We want control over where uploaded media resides in our file system. We also want to make sure to replace illegal characters in filenames on upload. For this, we need:

Transliteration

http://drupal.org/project/transliteration
Translates Unicode non-Latin characters in filenames (and, if desired file paths) into US-ASCII characters to ensure php compatibility, and web-friendliness.

Filefield Paths

http://drupal.org/project/filefield_paths
Control where in the filesystem uploaded images go on a per field, per content type basis. For example, you may want all images uploaded on News content types to reside in files/images/news/ and files in files/attachments/. This usage is crucial if you intend on granting access via a file browser like IMCE, because if you grant users access to the base files directory, they'll also see your cached css, js, etc. A smart use of paths and permissions allow you to grant editors access to only files/images/news for example, but not files/images, if you like that sort of granularity.

Think about your File Structure

It's a good idea to create some standards around where we store images. NEVER put your images and files uploaded from fields directly in your files directory. Always use a subfolder.

There is no catch-all strategy here. I find it is generally meaningful to group imagefield uploads by content type.

If users are to browse to files via IMCE, we can limit access to certain directories by user role, or by user. This can be useful if you have a bunch of stock images for inclusion in content by editors, for example, or need provide access to PDF uploads to the marketing department.

File structure within the filepath

…generally, in one of 3 locations:

```
sites/[site_name]/files
sites/default/files
sites/all/files
```

Items highlighted at the end of the list are examples auto-generated by popular modules in both Drupal 6 and 7.

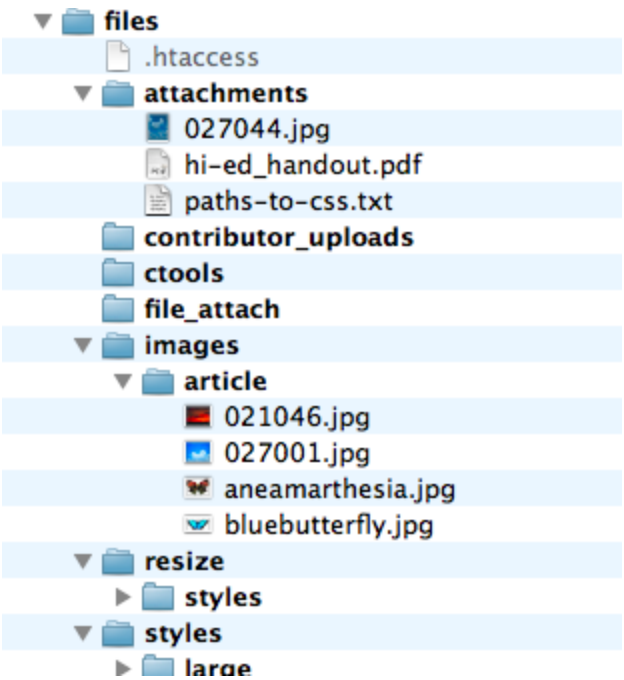
```
files
images
[type] (i.e. news, events, etc, by using the pattern images/[type])
      user_uploads
```

```
            user1
            user[uid]
      galleries
            [nid]
video
      thumbs
imagecache
imagefield_thumbs
imagefield_default_images
ctools
css
js
xmlsitemap
styles
color
adaptivetheme
feeds
```

Drupal 7 allows simultaneous private and public download paths. You should set as many utility directories to the private directory, outside your drupal root, as possible.

I always create multi-site installations and place my files directory inside the site. This results in absolute links to image files being perhaps more descriptive, i. e.

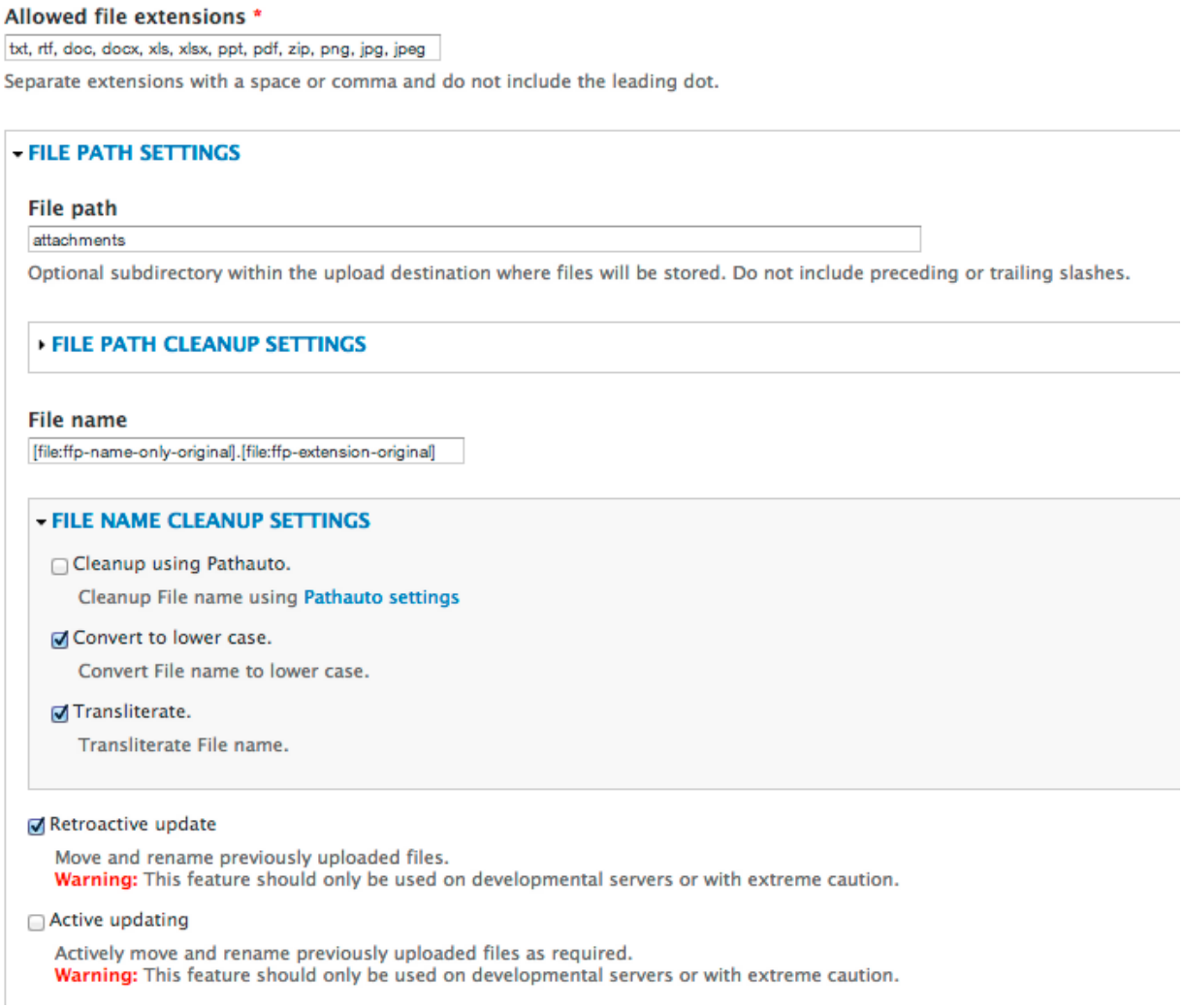
```
http://mysite.com/sites/mysite.com/files/images/news/myimage.jpg
```



Refactoring your existing file uploads

Assets a mess? It's not too late to organize. It is possible to refactor and move your images around

using the file field paths which can perform a retroactive updates. You'd edit your field settings, set a new upload path destination, and can also rename your files using transliteration and pathauto if you like. Don't try this on a LIVE site unless you've tested the effects first!



 *This (generally) will successfully migrate and link images you've uploaded through fielded entities, but can really mess up inline images in text fields, so test thoroughly.*

2. Files (attachments)

Files can be either attached to content or inserted inline. I like a unified approach that allows the user, on a single multi-value filefield, to list or not, and insert or not on each upload. We also open up the description field for custom renaming of these links when listed.

Permitted upload file extensions:

```
pdf, txt, rtf, doc, docx, xls, xlsx, ppt, zip, png, jpg, jpeg
```

of course you can add or remove to this list whatever file types you want to support, but I find the above covers 90% of use-cases and does not unnecessarily clutter the field's description on the node form.

3. Images

A. IMAGE FIELD Configuration

Image fields/imagefields allow control over the display of images attached to nodes in a precise, granular, and context-specific ways. Where uploaded files reside on upload can also be custom controlled with **filefield_paths** (media module lags behind on this). Image is a core field in D7.

There are often instances where we want to allow the user to insert images inline. You might not want to offer this on every content type however. There are a couple of strategies to take when implementing selective inline image uploads.

It's good idea to set the **Maximum image resolution** to something reasonable. The bigger the source files, the more your server will have to work to resize them. 1200 x1200 is a decent choice.

Don't bother with the **Maximum upload size** since you're already defining resolution.

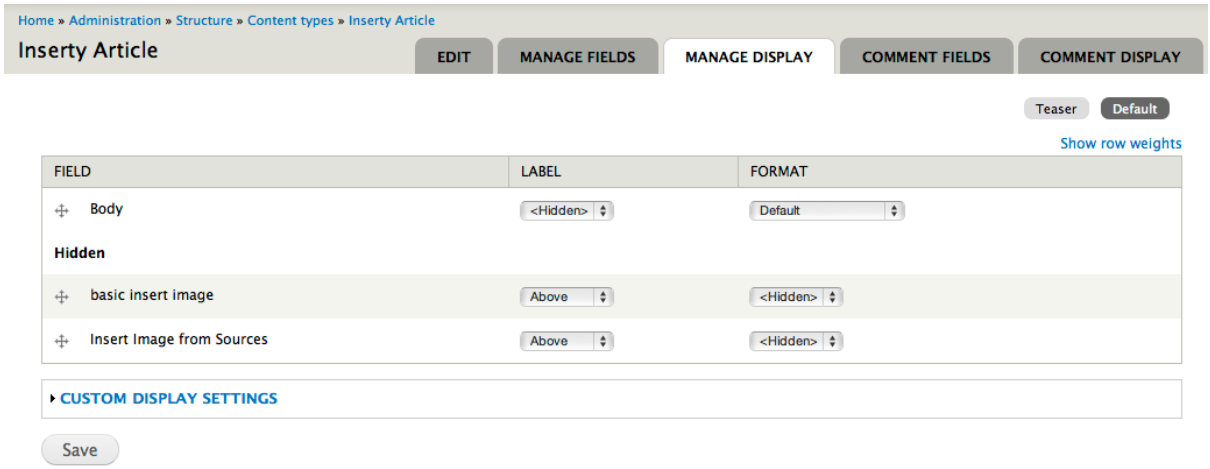
Always **Enable Alt fields** on all your image fields for accessibility. The **Title** attribute isn't necessary, and can be annoying to users on hover.

B. INLINE IMAGE OPTIONS

Let's look at a few different pieces and build a recipe. We will use the Insert module and, optionally, allow file browsing the filesystem with the Fieldfield Sources + IMCE modules to re-use images already on the server. Image Resize Filter will allow us to scale images.

INSERT MODULE

On the node's Manage Display tab we are going to set this field to <Hidden>. In Drupal 6, this is done on the node's Display Fields tab you should additionally check the "exclude" checkbox, as there is no need to load the fields themselves on the node view. The fields are simply a mechanism to inject <img> tags into our body field, and that's all that counts.



FILEFIELD SOURCES

This extends the filefield module and allows us to use IMCE to browse our files. The module also supports entering an external URLs, which'll download the resource to your site. In addition, you can set a directory in your filesystem as a kind of in bucket, if your workflow requires FTP uploading of a bunch of images ahead of time for.

IMCE

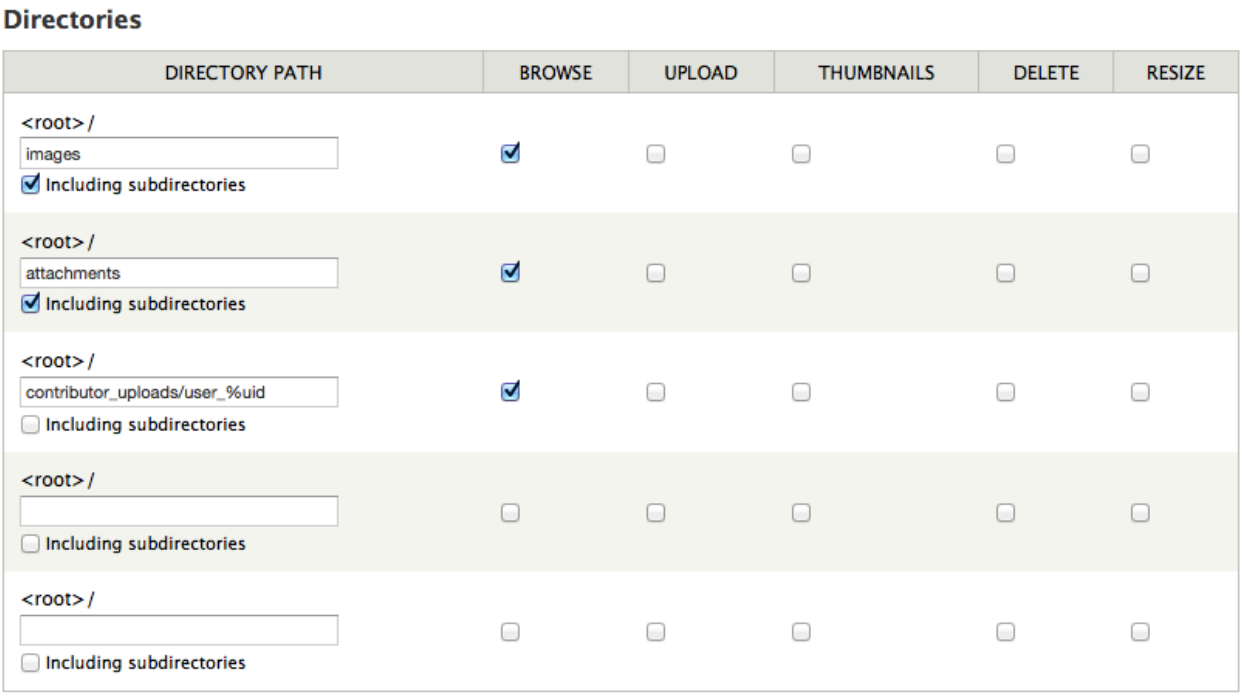
IMCE provides a file browser that allows us to see and preview images already on the server.

It offers granular permissions by role, controlling access on multiple directories, and governing who can upload or delete images from the filesystem.

I discourage using its upload features. It's better to use a field for that so we can transliterate files and upload them to the right place.

We are going to disable the thumbnail creation features because we will use imagecache (triggerd by the image resize filter module) to create smaller versions. This keeps our filesystem clean and lean and ensures contributors don't accidentally insert an image that is too small.

Really, you should un-check all IMCE options **except browse** and set paths to your images directory only. Now you see why we don't just dump images into the default sites/default/files directory and why a strategy for file management is important.

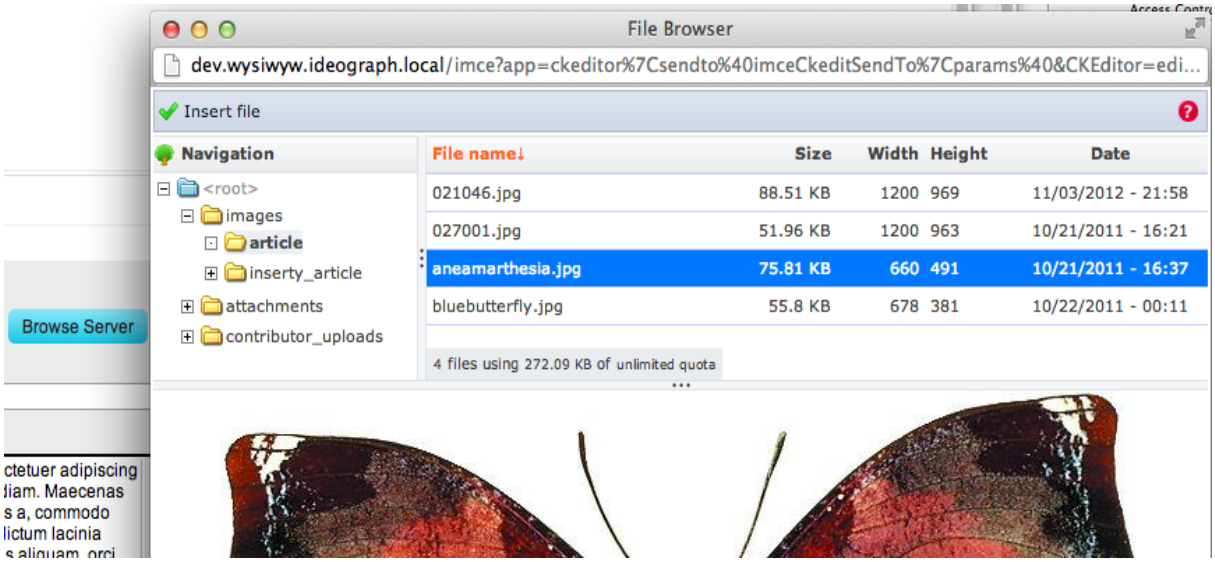


If you have a multi-user blog and want independent image libraries, you can set a profile up just for these users and set a directory path to something like: `contributor_uploads/`*`user%uid`*

You could then grant the editor role access to the contributor_uploads and it subdirectories.

N.B that `<root>/` here signifies the root to your public files directory, not the site root.

IMCE, integrates with filefield_sources, offering up a “File browser” option.



I also adds a browse server option to the image button, but the UI for the image button is horrendous. The filefield method is much better.

See Securing file permissions and ownership on drupal.org if you are having problems browsing the filesystem.

Also, If you are using the filefield sources module, and have set IMCE configuration profiles, you may need to set the field's "File browser mode" to "Full" or risk being denied.

IMAGE RESIZE FILTER

This module leverages the imagecache API to generate an version of your inserted image image on node save, so that it is rendered at 100% according to your image api settings.

Beware that this will scale up images at the size they are inserted at. So if you use insert module and insert a thumbnail, and drag the handles, your resulting image will be highly pixelated. Insert a size larger than you need, then scaling down.

This is an input format, so we need to check it on our text editor input format. We also need to make sure it runs BEFORE our WYSIWYG filter.

The resizing handles are broken in Chrome and Safari, but works fine in Firefox. https://drupal.org/node/839130 tracks this issue.

Other useful UI modules

Since we are dealing with user input, here are some other modules that might be useful to you:

WYSIWYG template
You can insert layouts directly in the editor. Example use cases: for static pages, you have a 2 column layout and want to use floated DIVs. You want to start the user off on the right foot by providing some dummy content that is appropriately formatted.

Image Javascript Crop
Create custom crops your images as an image style/imagecache derivative. Preserves the original image

Wysiwyg Fields
Builds a bridge between the file/image field and your wysiwyg by providing a button in the editor. Allows optional hiding of the field itself too, so you get the best of both worlds: tight file control, and a user-friendly editor with less form fields.

Wysiwyg Image Upload
D6 module alternative to the IMCE set-up outlined in this document.

CKEditor link
Easily create links to Drupal internal paths through CKEditor's Link button and dialog. Uses autocomplete and multilingual support.

Linkit
Linkit links to nodes, users, managed files, terms with its own autocomplete. Add-on for views pages and panel pages.

REFERENCES

TinyMCE valid element list which we use to create our WYSIWYG filter
http://tinymce.moxiecode.com/wiki.php/Configuration:valid_elements

Jen Lampton's tutorial
http://wysiwyg.jenlampton.com/

Video presentation of parts of this method in early days
http://www.chapterthree.com/presentations/intuitive-wysiwyg-editors-inline-image-handling

Karen Stevenson's article on building a wysiwyg feature in Drupal 6

http://www.lullabot.com/articles/wysiwyg-feature

Patching wysiwyg for features use in Drupal 6
http://drupal.org/node/624018#comment-2902814
https://github.com/sprice/da_wysiwyg

My presentation at Drupalcon Denver 2012
http://denver2012.drupal.org/program/sessions/what-you-see-not-always-what-you-get-it-can-be