Drupal Site Building & Introduction to Module Development Training

(a) About Drupal (9:30 – 11:20)
   i. Drupal Community
      A. Brief history
      B. Core vs. contrib
      C. Excellent resources for the times when you get stuck
      D. The benefits of getting involved!
   ii. How Drupal works at a high level
      A. The LAMP stack
      B. How Drupal utilizes that stack
   iii. Meet Drupal
      A. Getting a development environment setup
      B. Posting a piece of content

(b) Break (11:20 – 11:40)

(c) Drupal's vocabulary and where to find the items (11:40 – 1:30)
   i. Node
      A. Node pages vs. module generated pages
      B. Workflow settings
      C. Creating/editing nodes
   ii. Taxonomy
      A. Categorizing content
      B. Usage best practices
   iii. Blocks
      A. Understanding what blocks are and what they are for
      B. Placing a block on the page
   iv. Menus
      A. More than links: What Drupal's menu system does
      B. A basic overview of path aliasing
      C. How path auto can be used to improve your site architecture

(d) Lunch (1:30 – 2:00)

(e) Content Construction Kit (CCK) (2:00 – 3:50)
   i. When to use CCK rather than taxonomy (AND WHEN NOT TO)
   ii. Creating new content types
   iii. Adding a custom image field to a content type
   iv. Using ImageAPI and Imagecache to resize the images
   v. Configuring the display of those fields

(f) Break (3:50 – 4:10)

(g) Views (4:10 – 6:00)
   i. Creating dynamic listings of content for pages and blocks using fields and filters
   ii. Creating a highly configurable image gallery with views
   iii. Using Views relationships to relate elements in your lists to other content on your site
   iv. Using arguments to create dynamic path based filters
The Drupal Community

Drupal Community Sites

1. [http://drupal.org](http://drupal.org) (or D.O) – generally where modules can be found and development discussion happens

2. [http://drupal.org/forum](http://drupal.org/forum) – a good, albeit very slow, resource for past issues and their resolutions

3. [http://drupal.org/handbooks](http://drupal.org/handbooks) – an excellent resource for examples and descriptions of how to use Drupal core, contributed modules and APIs

4. [http://groups.drupal.org](http://groups.drupal.org) (or G.D.O) – where people can subscribe to specific topics and working groups. This is where much of the conversation takes place and new initiatives are born.

5. [http://drupal.org/project/issues](http://drupal.org/project/issues) – issue queues are rich with information about bugs, fixes, future feature requests and general discussions about the direction of a given project

6. [http://planet.drupal.org](http://planet.drupal.org) – an aggregator for the RSS feeds for most Drupal related blogs and websites

7. [http://drupal.org/mailing-lists](http://drupal.org/mailing-lists) – an excellent resource for support and development discussions

Drupal Internet Relay Chat (IRC)

The Drupal Internet Relay Chat (IRC) rooms are active 24 hours each day with discussions about the project and providing support to anyone who asks nicely! Drupal's IRC is hosted on irc.freenode.net. For a full list please consult [http://drupal.org/irc](http://drupal.org/irc)

1. #drupal-support – answering questions like “how do I...” or “I installed this module, now what?”

2. #drupal – discussions about development, code and the project as a whole. If your question starts with <?php this is the place

3. #drupal-themes – Theming related discussions
How Drupal Works

The Drupal Stack

LAMP (Linux, Apache, MySQL, PHP)

Linux
Open source operating system:  http://www.ubuntu.com
Mac or Windows also work.

Apache
Open source web server:  http://www.apache.org
MySQL

Open source database:  http://www.mysql.com

Drupal also supports PostgreSQL, and support for many more is coming.

The database stores all of the site's content as well as the configuration of the site.

PHPMyAdmin is a tool used to view and edit the database directly.

PHP

Open source programming language:  http://www.php.net

Drupal

Open source code and mark-up files written in PHP, css, js, and HTML.

Drupal has its own API, a PHP “dialect”, see http://api.drupal.org.

The Drupal codebase consists mainly of “modules” and “themes” each of which can be either “core”, “contrib”, or “custom.”

Module

A Drupal module is a directory of files which give a site a certain functionality. Modules are built to work together so that sites can be assembled by adding modules like lego blocks.

Theme

A Drupal theme is a directory of files which give a site its look. Most sites use a single theme. A theme is essentially a visual
“skin” over the site functionality. Themes generally contain most of the css for the site.

Core

Drupal core is the set of files which come “out of the box” when you download Drupal. This code is extensively peer-reviewed and is generally the most reliable and bug-free. It contains enough modules to create some basic sites, such as blog or forum sites, and has a few basic themes. It also contains the core Drupal API.

Drupal core has major and minor releases, and is currently on version 6.12 (major version 6). Major releases bring drastic code changes, while minor releases are only for bug fixes. You can only use modules and themes which are of the same major version as your core code.

Contrib

Drupal developers and themers also contribute modules and themes for everyone to use. This code varies in quality, but much of it is extremely useful. There are thousands of contributed modules. Contributed themes are usually less useful, as most sites include branding in their theme and cannot be contributed.

See http://drupal.org/project/modules and http://drupal.org/project/themes

Acquia, a Drupal company, keeps a set of contributed modules and themes in a collection (including core Drupal) called “Acquia Drupal”. It includes many of the most useful contributed modules and a few nice themes.

Custom

A custom module or theme is one which is created for your site and is not shared with the community. Custom modules are written by Drupal developers, while custom themes are created
by Drupal themers.

Site Architecting / Drupal Vocabulary

Regions

Regions are areas in which blocks, menus, or node content can appear. Regions are declared in the theme .info file. An easy way to see what regions are available is to go to the blocks administration page at /admin/build/blocks. The areas listed in the block table are the available regions. The height and width of the regions can be edited with CSS. The placement of regions can be edited in the page.tpl.php file.

Below are the regions for the 'Acquia Marina' theme.

Content
Node

A piece of content containing a title and some text or media. E.g., a blog post, or an 'About Us' page. Nodes have a numeric node ID and can be found at /node/ID. See /admin/content/node.
Content type

A type of node. E.g., blog post, article, event, page, video, job listing, forum topic. Each Drupal site defines its own content types depending on its content requirements. Content types often differ structurally by having different fields in their input forms, and by being displayed differently. See /admin/content/types.

Teaser

A short version of a node. A blog site, for example, may show a list of teasers and require you to click 'read more' to see an entire node. You can adjust the teaser length at /admin/content/node-settings.

Comment

A comment made on a node. You can allow comments on some content types and not others. See /admin/content/comment.

User

A person (or bot!) who has used the site. If the user has logged in they will have at least a username, email address, and password. Users also have a numeric user ID and have an account page at /user/ID. See /admin/user/user.

Role

A type of user. Users can be anonymous, meaning they have not logged in, authenticated, meaning they have logged in, and can also have additional roles as defined for a site. Some examples are 'administrator', 'editor', 'blogger', and 'staff'. See /admin/user/roles.

Permission

A per-role ability. Authenticated users may be given the 'create
comments' permission and an administrator role may be given permission to 'delete comments.' See /admin/user/permissions.

**Super user**

The first user account on a site (usually the person who built the site) has a user ID of 1 and is called the super user. This account is granted every possible site permission and therefore has no need for roles. See /user/1.

**TAXONOMY**

**Taxonomy**

Classification of nodes into different categories. See /admin/content/taxonomy.

**Vocabulary**

A section of the site's taxonomy system. An example of a vocabulary would be “Topic”. A vocabulary may optionally be hierarchical, or allow free-tagging. Vocabularies are assigned to one or more content types. An event content type may use an “Event Type” vocabulary, and an image content type may use an “Image Gallery” vocabulary.

**Term**

A vocabulary contains multiple terms. On a blog, terms are often called tags. In a “News Type” vocabulary, example terms would be “press release”, “media coverage”, and “shameless self-promotion.” Each term has a term ID, and a corresponding listing page at /taxonomy/term/ID.

**BLOCK**

**Block**

A section of content which is not the central page content nor a part of the site's template. Blocks are often found in sidebars and
may contain a menu, a login form, a list, text, an image etc. Blocks can be defined by *modules*, or can be created at /admin/build/block.

**Region**

A section of the page which can contain *blocks*. “Left Sidebar” and “Footer” are examples of regions. The block administration page (/admin/build/block) shows the available regions. The choice of regions is defined by the *theme*.

**MENU**

**Menu**

A group of links (menu items) used to navigate a site. Most sites use a “primary links” menu at the top or left of the page. Drupal also has a “navigation” menu which includes all the administration links. See /admin/build/menu.

**Menu Item**

A link in a menu, which contains a *path* and a title. The menus are arranged hierarchically as a tree of menu items. Menu items can either come from a module or be added by an administrator. Menu items which come from modules have access control, so that only the items which a user can access are displayed.

**Path**

The URL of a Drupal page. For example, node/3 or admin/build/path.

**Path Alias**

An alternative path for a Drupal page. A Drupal page must have one true path and any number of path aliases. For example, 'node/3' may have an alias of 'about-us' to give users a more friendly URL. *Pathauto* is a contributed module which creates path aliases for content automatically, and is used on
most Drupal sites.

For more Drupal vocabulary:

Drupal general concepts:  http://drupal.org/node/19828
Drupal terminology:  http://drupal.org/node/937
Developers Toolkit

Develop Locally

• “You don't fix a car while it is on the road.”
• Test work locally before making changes to live site.
• Will save you time and effort and headache.

Workflow

1) Backup database using a backup script, phpmyadmin or backup and migrate module from server to local machine.

2) Save file system from server to local using ftp or svn.

3) Test configuration or code changes locally.

4) Backup files and database on server. Make changes to server which were tested locally. *

* Ideally this should be performed using some kind of an update script in a module's .install file. Having a module specifically for deployment is common and can help to keep your files separate and clean.
Setup a Local Environment:

MAMP: http://www.mamp.info for mac

XAMPP: http://www.apachefriends.org/en/xampp-windows.html or

WAMP for win. Demo: http://www.lullabot.com/node/275/play

Backup and Manipulate Database:

PHPMYADMIN: http://www.phpmyadmin.net

BACKUP AND MIGRATE MODULE: http://drupal.org/project/backup_migrate

Firebug

• Helpful for dissecting and testing HTML and debugging javascript

• http://getfirebug.com/
“It doesn't matter which one you use just learn it well” - Justin

CSS and function hints can be really helpful

Searching through file systems is really useful for Drupal development

### Tools

**TEXTMATE:** [http://macromates.com](http://macromates.com) for mac

**NOTEPAD++:** [http://notepad-plus.sourceforge.net](http://notepad-plus.sourceforge.net) for win

**VIM:** [http://vim.org](http://vim.org) for mac/linux

**Eclipse:** [http://www.eclipse.org/](http://www.eclipse.org/) cross platform
**Subversion**

- Don't be that person. Have a plan to back your database and filesystem.
- SVN holds all versions of files in a project.
- Allows you to make local changes that are saved.
- Revert to previous changes even if you've done something dumb like deleted the wrong file.

**Workflow**

1. Download the subversion repository to your local computer.
2. Make your changes in the file system (ie add a module, edit CSS or other code).
3. Commit your changes to the subversion repository.
4. Update the file system on the server!

** Please note: This is an oversimplified workflow and ideally your SVN workflow would include branches and tags for more granular version control. For more information on this please consult the free SVN book. [http://svnbook.red-bean.com/](http://svnbook.red-bean.com/) **
Content Types And Fields Overview

Content Types

Based on the handbook page:  [http://drupal.org/node/21947](http://drupal.org/node/21947)

A single web site could contain many types of content, such as informational pages, news items, polls, blog posts, real estate listings, etc. In Drupal, each item of content is called a node, and each node belongs to a single content type, which defines various default settings for nodes of that type, such as whether the node is published automatically and whether comments are permitted. (Note that in previous versions of Drupal, content types were known as node types.)

When you first install Drupal with the default installation profile, you will have two content types defined: "Page" and "Story". When you enable other core and contributed modules (by visiting Administer >> Site Building >> Modules), you will find that you have other content types available; you can also create your own content types.

In addition to these basic types, custom content types can also be created by going to Administer > Content > Content types > Add content type. You might do this as a way to organize your content -- for instance, you might have "Article" and "Image" as two simple content types on your site, rather than just using "Story" for both.

Custom Fields With the Content Construction Kit (CCK)

Custom fields can be added to a content type by using the Content Construction Kit (CCK) contributed module. Custom fields are used to store additional information beyond the Drupal defaults (title, body, authoring information, time created/updated, and publishing status); for instance, on a real estate site, a real estate listing content type might have fields for the type of property, land area, etc. Additional information on the CCK module is available from the Content Construction Kit Handbook.
Setting up Content Types and CCK

On the modules page (Site building → Modules) enable all the CCK modules except for FileField Meta which is unavailable. Save the configuration.

Setting up a New Content Type

1. Start by going to Administer → Content management → Content types → Add content type.

2. Give your new content type the “Name” that you would like displayed to users, such as “Services”. Give it a machine-readable “Type” (which may only contain lowercase letters, numbers, and underscores), such as “services”. Next, give the content type a “Description”. The description should explain to content creators what this type is or where it is displayed on the site.

3. Under “Submission form settings” you have the option to change the Title and Body labels. Generally you will not need to change these unless you want the label to be more specific (for example changing the Title to “Service Name”).

4. Under “Workflow settings” you will see the default publishing options. These set the default settings but can be changed by users with appropriate permissions (“administer nodes”) on a per node basis when the actual content is created or edited. You will usually want your nodes to be published by default but may not want nodes of this new type to be automatically promoted to the front page. Of course this depends on how you want your site to function.
5. Under “Comment settings” you will find a long list of options. If you do not want people to add comments to this content type you can simply click “Disabled”. If you do want comments you can usually go with the default “Read/Write” options. Depending on your preferences, you may wish to delve into the more advanced options related to how comments are displayed.

6. Now save the content type and you will be directed to the list of content types.
Adding Custom Fields Using the Content Construction Kit (CCK)

1. Start by going to Administer → Content management → Content types

2. To add fields to the new content type, navigate to admin click “manage fields” for the content type you wish to add fields to, or use the manage fields tab if you are currently editing the content type.

3. You will see that the Title, Menu Settings, and Body fields are already added by default.

4. To add a new field first give it a Label (such as “Contact”) and specify the machine readable Field name in the box next to “field_”. The field name may contain only lowercase letters, numbers, and underscores (such as “contact”).

5. Now you have several options for the Type of field. In this case we are using a “User reference” which allows us to connect this node with a user in the system.

6. Depending on what Type you select you will have different Operations options. In this case for the user reference we will use autocomplete field.

7. Once you have selected your options, save the field.
8. Next you will be taken to the settings page for your new field. Field settings usually include the option require the field be filled out and an option to set the number of values that can be entered. In the case of a user reference we also need to select the user roles that can be referenced.

![Global settings](image)

9. Once you have set all the options, save the field settings.

<table>
<thead>
<tr>
<th>Label</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Node module form.</td>
</tr>
<tr>
<td>Contact *</td>
<td>field_contact</td>
</tr>
<tr>
<td>Menu settings</td>
<td>Menu module form.</td>
</tr>
<tr>
<td>Body</td>
<td>Node module form.</td>
</tr>
</tbody>
</table>

10. You will be taken back to the manage fields page. Now we can change the order of fields which will be the order they are displayed on the form to a user adding content and the order they are displayed when the node is viewed later. Drag the fields into the new order then save the changes.

11. Add any additional fields that are needed.

12. Now you can determine how the fields are displayed by
clicking the Display fields tab. Here you can select the style for the label (Above, Inline, or Hidden) and how to show in the Teaser of Full Node.

13. Save your changes.
Views Overview

Based on the project page:  http://drupal.org/project/views

The Views module provides a flexible method for Drupal site designers to create lists and tables and control how of they are presented. Traditionally, Drupal has hard-coded most of this, particularly in how taxonomy and tracker lists are formatted. Using views arbitrary lists of almost any content on a Drupal site can be created by listing, filtering and sorting any fields that hold your content.

Under the Hood

This tool is essentially a smart query builder with a powerful user interface that, given enough information, can build the proper query, execute it, and display the results. Views builds queries for your database (and soon your web services) and intelligently displays and formats the results. It is capable of building filters based on dynamic input (arguments) and joining one piece of content with another if they share a piece of information (relationships).

You need views if:

a) You like the default front page view, but you find you want to sort it differently.

b) You like the default taxonomy/term view, but you find you want to sort it differently; for example, alphabetically.

c) You use /tracker, but you want to restrict it to posts of a certain type.

d) You like the idea of the 'article' module, but it doesn't display articles the way you like.

e) You want a way to display a block with the 5 most recent posts of some particular type.

f) You want to provide 'unread forum posts'.
g) You want a monthly archive similar to the typical Movable Type/Wordpress archives that displays a link to the in the form of "Month, YYYY (X)" where X is the number of posts that month, and displays them in a block. The links lead to a simple list of posts for that month.

For additional help download and enable the Advanced Help module (http://drupal.org/project/advanced_help) then navigate to [yoursite]/admin/advanced_help/views
Creating Lists of Content Using Views

1. Start by going to Administer → Build → Views
2. Click the “Add” tab
3. Enter a “View Name” that may contain only letters, numbers and underscores (e.g. “service_stories”). Next, provide a “View Description” which is the name for the view that will be displayed on the administrative pages (e.g. “Service Stories”). Now enter the “View Type” which is the type of content that this view will display (e.g. “Node”). Click “Next”.

4. This will take you to the page that where all of our actual configuration will take place.
5. To add fields to be displayed in our view, the click on the “+” button in the “Fields” box. A dialogue will appear below the main views control panel listing all of the available fields on your site. You can filter this list of fields by selecting “Node” (or another relevant item). Check the box by the field(s) you would like to add (e.g. “Node Title”) and click “Add”.

![Page: Add fields](image)

6. This will take you to a page where you can configure the way this field will be displayed. In the case of a Node Title you may set the label, decide whether to link the field to its node, etc. Generally you will want to select “Link this file to its node” and click “Update default display”.

![Configuration page](image)
7. At the bottom of the page you will now see a live preview section that shows you the content that your view is listing and the way it is being formatted.

8. To display this list differently, click on “Style” in the “Basic Settings” box. Below the views control panel you will see a configuration panel with a list of options, you may select a different style (e.g. “Table”) and click “Update Default Display”.

9. Many view styles will next present you with a configuration panel that will further customize the style. Make your selections and click “Update Default Display”. The updated display will be presented below the views control panel.

10. To display this list on its own page select “Page” in the far left column and click “Add Display”.

11. Next we must specify the URL or “Path” where we will find navigate to our View. Click on “Path”. A path dialogue will show up at the bottom of the screen.

Drupal Site Building Training
12. To add a Filter click on the “+” in the “Filters” box. A dialogue box will appear below the views control panel which can be sorted be filtered by the drop-down box at the top of the page. Select a category to filter by (e.g. “Node”). Next, select the field on which you would like to apply filters to the content (e.g. “Type”).

13. At this point a dialogue box will appear below the views control panel where you can configure this field. To select only content of the “story” type check “Story” and then click “Update Default Display”.

14. Click “Save” at the bottom of the views control panel. Congratulations! You have created a list of stories on your Drupal site!
Creating Views Relationships

Views Relationships allow you to join an additional piece of content with the content gathered by a view. For example, imagine you are creating a list of stories and those stories have a “Related Service” node reference field. If that “Service” node then has some content that we would like to display we can add that to the view of stories. The setup for this view is described below.

1. To add a Views Relationship to a view click the “+” in the “Relationships” box. A dialogue will appear below the views control panel where we can select the field that we would like to use to connect to another item inside views. In order to add a relationship filter the list by the relevant category using the drop-down box. Next select the field you would like to use (e.g. to use a “node reference” field called “Related Service” filter by “Content” then check the field). Click “Add”.

2. A configuration dialogue box will appear below the views control panel. Select a label and if you would like to list only content where this relationship exists check “Require this relationship”. Next click “Update”. 
3. Now you may add fields that exist on this related entity and display them along side the fields of the listed content. (See #5 of the previous example)

4. When you configure that field, you will notice that you may now specify a relationship from which to add this field. Select the relevant relationship and configure the field before clicking “Update”.

5. At this point your view should display the field pulled from both of both entities and additional relationships can easily be added! Don't forget to click “Save” to save your changes!
Filtering Views Dynamically Using Arguments

Views is capable of making lists that are filtered by arguments on the fly. These arguments can be appended to the end of the url (my_view_path/my_argument), generated by php code or provided several other ways. Below is a simple example of creating a “my stuff” page.

1. Create a view as described in the previous section with a new unique path and a without any filters.

2. Click the “+” in the “Arguments” box. A dialogue box will display below the views control panel. Filter the arguments by “Node” and then scroll to the bottom and check “Node: User posted or commented”. Click “Add”.

![Screenshot of Argument Selection Dialogue Box]
3. In the configuration dialogue box that will appear below the views control panel, find the “Action to take if argument is not present” section and select “Provide Default Argument”.

4. In the “Provide default argument options” section that appears below, select “User id from logged in user”. At the bottom of this panel click “Update”.

5. In the live preview you should now see a list of content that consists exclusively of items that you have created or commented on.

6. Do not forget to click “Save” so that you do not lose your changes.
Day Two – Introduction to Drupal Module Development

Day two will be a hands on introduction to Drupal's API. This section will explain what to do when no module is a good fit for your particular problem or use case. Students will follow along on their local environment as we begin building an example module that integrates with the most commonly used components of Drupal's API.

The goal of day two will be to establish a working knowledge of how Drupal modules integrate with Drupal's API and how to implement the necessary functions to do so.

1. Preparing for Drupal Development (9:30 – 11:20)
   (a) Getting setup with Drupal development tools
      i. A brief discussion of development best practices
         A. Working on a local environment
         B. The importance of version control
      ii. Devel module
      iii. Admin menu
   (b) How Drupal allows for extensibility throughout the system
   (c) The anatomy of a module
      i. .info file
      ii. .install file
      iii. .module file

2. Break (11:20 – 11:40)

3. The database abstraction layer (11:20 – 11:40)
   (a) Creating an installer for your module
      i. Understanding Schema API
      ii. Defining your tables
      iii. Installing/removing your module via hook_install/hook_uninstall
      iv. Updating your tables when your schema has to change
   (b) How to use Drupal's database functions
      i. db_query
      ii. db_fetch_result
      iii. db_fetch_array / db_fetch_object
      iv. drupal_write_record

4. Lunch (1:30 – 2:00)

5. The hook system (2:00 – 3:50)
   (a) How modules can leverage the hook system
   (b) Creating an implementation of hook_nodeapi
      i. Adding our data to the node object
      ii. Displaying our data on the node

6. Break (3:50 – 4:10)

7. Understanding the theming layer (4:10 – 6:00)
   (a) The anatomy of the php template system
      i. Understanding the theme registry
      ii. The basics of inheritance and overrides
      iii. The basics of template nesting and inheritance
   (b) How to properly generate themeable html inside your module
      i. Registering and writing a theme function
         A. What a good theme function contains
      ii. Registering and creating a template (.tpl.php) file for your module
      iii. Using preprocess functions to prepare your template variables
         A. Watching out for gotchas
The Drupal Database Abstraction Layer References

2. The [api.drupal.org](http://api.drupal.org) [http://api.drupal.org/api/group/database/6](http://api.drupal.org/api/group/database/6)
3. Pro Drupal Development: Page 95

The Hook System

1. Drupal developer handbook entry relevant to hooks and their implementation: [http://drupal.org/node/231276](http://drupal.org/node/231276)
2. [api.drupal.org](http://api.drupal.org) list of hooks and brief explanation of their use: [http://api.drupal.org/api/group/hooks](http://api.drupal.org/api/group/hooks)
3. Pro Drupal Development: Page 4

Understanding the Theming Layer

Drupal Theming Basics

Core Themes

Garland, Bluemarine, Pushbutton, Chameleon (in Drupal 6)
Useful only for quick sites that require no design, or as administrative themes
Garland integrates with Drupal's color picker for very quick site customization

Contributed Themes

Generally useful for small projects or intranet sites with no design budget, or as a starter theme
Quality varies widely

Starter themes

A contributed theme created only as a starting point for the creation of a custom theme. Zen theme ([drupal.org/project/zen](http://drupal.org/project/zen)) is the most used starter theme.
See a Drupal starter theme comparison at [http://adaptivethemes.com/starter-theme-comparison.html](http://adaptivethemes.com/starter-theme-comparison.html)

Subthemes

A theme which inherits some of its components from another theme. In
core, Minelli is a subtheme of Garland.

**Fixed-width vs Fluid layouts**
Themes generally have either a fixed-width or a fluid layout. Compare huffingtonpost.com (fixed-with) with drupal.org (fluid).

**Anatomy of a Theme**
See [http://drupal.org/node/171194](http://drupal.org/node/171194)

- .info file: defines the theme name, its regions, stylesheets, and optional parent theme (for subthemes). This file is the only required file of a theme.
- .css files: stylesheets defining the appearance of the theme
- .jpg, .gif, or .png image files: background images used by the stylesheets, the site logo, or the theme's screenshot
- .js files: javascript files used to create dynamic front-end elements. Drupal includes the jQuery javascript library and best practice is to use only jQuery javascript.
- .tpl.php files: template files for defining the structure of the HTML markup. These override default Drupal templates. You can create templates for the general page, node content, block content, and can override other templates found in your core and contrib modules.
- template.php file: holds additional logic required by your theme. It can define variables for use in template files (by using preprocess functions) and override themable functions.
- theme-settings.php: some themes contain this file for defining additional settings for you to use at /admin/build/themes/settings.

**Starting a Zen Subtheme**
See [http://drupal.org/node/226507](http://drupal.org/node/226507)
or use the zenophile module (drupal.org/project/zenophile)
CSS Best Practices

Less is more
Work your way up and in:
  Layout: Start with the back layer, working toward the front
  Elements: Start with the most general styles, working toward the specific
Continually refactor to minimize redundancy
Theme as though the speed and ease of making changes in the future is more important than getting done faster now
Do not use selectors more specific than required
Theme for IE last, putting any unavoidable IE hacks in a separate file
Consider accessibility (blind users), performance (minimizing large graphics), and variable screen sizes, browsers and available technology (javascript, Flash)
Follow Drupal's CSS coding standards: http://drupal.org/node/302199

Theming trouble-shooting and tips

After adding a new template file, you must rebuild Drupal's theme registry for it to take effect. One way to do this is to visit /admin/build/modules.

Do not edit CSS while CSS aggregation is turned on (this condenses all your CSS files into one cached file and is a great performance-enhancer for live sites.) Turn it off at /admin/settings/performance.

Internet Explorer 6 has a limit of 32 CSS files, which Drupal sites often exceed. If your site looks FUBAR in IE6 but improves with CSS aggregation turned on, you’ve hit the limit. In this case, turn CSS aggregation on to identify the true IE6 bugs, then turn it off to make adjustments.

Specific template files require the presence of a generic template file to take effect. For example, if you add a node-blog.tpl.php file but do not define a node.tpl.php file in your theme, the node-blog template will be ignored.

Use the W3C's validator for your HTML and CSS during and after theme creation (http://validator.w3.org/) to ensure that your problems are not due to broken markup.

Use excellent spacing and indentation in your template files to avoid unmatched HTML tags.

Master CSS through books and online resources (w3schools.com).
Master Firebug.
Push back on theming demands that aren't really worth the trouble.

Further Resources

Top Notch Themes guide: [http://www.topnotchthemes.com/handbook](http://www.topnotchthemes.com/handbook)
CSS book: [http://tinyurl.com/l5as5y](http://tinyurl.com/l5as5y)
IRC: [http://drupal.org/irc](http://drupal.org/irc) #drupal-themes on freenode
Drupal group: [http://groups.drupal.org/theme-development](http://groups.drupal.org/theme-development)
Drupal group: [http://groups.drupal.org/design-drupal](http://groups.drupal.org/design-drupal)

How to properly generate themeable html inside your module

1. Examples and an explanation of how to use Drupal's theme system to generate themable HTML: [http://api.drupal.org/api/group/themeable/6](http://api.drupal.org/api/group/themeable/6)
Day Three – Introduction to Drupal Module Development

Day three will dive further into Drupal’s API as we further develop the example module begun on day two. This will be the most technical day and will provide students with hands on experience modifying the behavior of other modules as well as how to integrating with more of Drupal’s advanced functionality such as exposing custom data to the Views and implementing access control.

The goal of day three will be to build on the working knowledge of the API established on day two and will teach students how to perform more advanced tasks while watching out for common Drupal “gotchas”.

1. The Menu system (9:30 – 11:20)
   (a) Creating an implementation of hook_menu
      i. Registering our URL
      ii. Specifying an access control callback and argument
      iii. Specifying a callback function and passing it arguments
      iv. Types of menu items (how to add tabs to a page, create a new page, etc.)

2. Break (11:20 – 11:40)

3. Creating a settings form (11:40 – 1:30)
   (a) Modifying existing forms
      i. Creating an implementation of hook_form_alter
         A. Adding a field to the node form
      ii. Adding a second submit handler to a form
   (b) Defining the form
   (c) Rendering the form using drupal_get_form
   (d) Specifying submit function(s)
   (e) Alternately using system_settings_form to build the form and store the values

4. Lunch (1:30 – 2:00)

5. Adding views integration (2:00 – 3:50)
   (a) Registering your module with views (hook_views)
   (b) Creating an implementation of hook_views_data to expose your custom fields to views
   (c) Specifying handlers for your fields
   (d) Creating and setting custom handlers for your fields

6. Break (3:50 – 4:10)

7. Scaling Drupal (4:10 – 6:00)
   (a) Considerations when building your module
      i. Using Drupal’s caching system
   (b) Using Memcache
      What a scalable stack looks like
Menu System

1. Drupal handbook entry on the menu system: [http://drupal.org/node/102338](http://drupal.org/node/102338)
2. api.drupal.org section on the menu system: [http://api.drupal.org/api/group/menu/6](http://api.drupal.org/api/group/menu/6)
3. Pro Drupal Development: Page 59

Form API (FAPI)

1. Drupal handbook entry on the Form API: [http://drupal.org/node/204270](http://drupal.org/node/204270)
3. api.drupal.org section on the Form API: [http://api.drupal.org/api/group/form_api/6](http://api.drupal.org/api/group/form_api/6)
5. Pro Drupal Development: Page 221

Scaling Drupal

3. Pro Drupal Development: Optimizing Drupal: Page 527 - 545