How custom is too custom?

Tips for coding (and when not too)

Brock Fanning

- brockfanning on drupal.org
- Drupal developer - government, content marketing, academia
- Party pooper (sort of)
- Question answerer

What is custom code?

Any Drupal code that does not have a presence on drupal.org. For example:

- PHP-driven blocks or pages
- alter hooks (hook_form_alter, hook_entity_view_alter, hook_query_alter)
- preprocess hooks (hook_preprocess_page, hook_preprocess_node)
- modules or javascript for unique site-specific functionality
- team-specific frameworks or helpers
- general-purpose modules that you haven’t put on drupal.org yet
- radical departures from the Drupal norms

What is contrib code?

Any Drupal code that has a presence on drupal.org

- modules, themes, install profiles
- patches

Choose your own adventure

You walk into a room, and are given a feature request to implement on your Drupal site.

Do you...

1. Open up your text editor and custom code it?

2. Open up your browser and look for a contrib solution?

Why not custom?

1. Cost of training and documentation, difficulty of hand-offs
2. Lack of outside support, responsibility of maintenance
3. Risk of re-inventing the wheel

What happens when custom solutions get contributed?

Why not contrib?

1. Inflexible - unusual or particular requirements (or environment)
2. Overkill - too complex, too many features, too hard on the server
3. Insufficient - not enough features
4. Incompatible environment

Some contrib solutions overcome these. And for cases where they do not, you can always change contrib (more about that later).

So contrib is preferred, but… fun?

How to have fun without writing custom code:

- Get involved in contrib (and core too)
  - Remember, Drupal is open source, all the code is there. If something isn't working, dig deeper.
- Embrace patches
  - Write them, use them, love them

You will be writing "custom" code, but instead of the code just being for your client, it will be for the whole Drupal community.

But be sure to use a Drush make file

Maintain all of your core and contrib code using Drush. Example:

```
---
api: 2
core: "7.44"
projects:
  media:
    download:
      type: git
      branch: 89287f76df1ff24409b3ec474c59218929385733
    patch:
      - https://www.drupal.org/files/issues/only_one_view_mode-2308451-17.patch
      - https://www.drupal.org/files/issues/wysiwyg_alt_and_title-2416701-30.patch
      - https://www.drupal.org/files/issues/term_reference_fields-2062365-13.patch
      - https://www.drupal.org/files/issues/media_module_token-2497529-3.patch
      - https://www.drupal.org/files/issues/incorrect-logic-display-value-2545738-1.patch
```

More about patching and contributing

- Patching is like acceptable hacking
- Getting your patch committed
  - Make sure the issue is well described (problem/motivation, solution, changes, etc)
  - After posting the patch, change the status to “Needs review”
  - Don’t be impatient
- But really, patches don’t need to be committed to be used
  - Well, not so much for D8
- Caution: Patches need to make the project better, not just different

Analyzing contrib projects

You need to get really good at this.

Red flags:

- Breaks site
- Impacts performance
- Not supported
- Has show-stopping bugs
- No stable release
- Similar projects on drupal.og

Analysis tips:

- Go to project page
- Look at usage statistics
- Look at date of last commit
- Browse the issue queue
- Look at available releases
- Try it out
- Benchmark it on your site
- Look at the code

Case Study: Contrib is overkill, or is it?

Site: large (hundreds of thousands of nodes)

Request: Add an og:image meta tag to all nodes of a single type

Contrib solution: Metatag module

Custom solution: drupal_add_html_head() inside hook_node_view_alter()

Analysis: Metatag project, gauge need for additional meta tags, difficulty of custom solution, installation/configuration of contrib solution, benchmarks

Twist: What is the need for additional meta tags increases?

Case study: Obscure contrib recipe wins

Request: A system for automatically importing data into Drupal, and displaying it in arbitrary ways. (Eg: phone directory). Needs to scale (many instances of this) and allow for dynamic display (sorting and filtering) of data

Obvious contrib solution: Node types, fields, and Feeds. But, at scale, that gets out of control.

Custom solution: Parse feed sources, save serialized arrays in database, customize each instance with its own template file which outputs the data uniquely.

Not-so-obvious contrib solution: Data, Feeds, Feeds Data, and Views

Analysis: Vet Data and Feeds Data (sandbox), try out recipe (patch)

Case study: Patch contrib to meet specs

Request: A text-sizing widget to increase/decrease font size on the page. Must be able to be placed using Panels.

The catch: No cookies allowed.

Contrib solution: Text Resize

Problem: It uses cookies, and is incompatible with Panels

Solution: Write 2 patches and use them

Case study: Patch contrib to fix bugs

Request: On a site that already uses the Media module, the client requests image captions and credits in the WYSIWYG.

Problem: The Media module has a multitude of show-stopping bugs. But, the client does not want a drastic change in workflow.

Solution: Dive in and help the Media module. 15+ patches later, your solution works and the community benefits.

Custom + Contrib combinations

Good way to minimize custom code: use contrib but alter it. In other words, use hooks provided by contrib, or general-use hooks like hook_form_alter().

- Look in the *.api.php files of modules to see what is possible
  - If this file is missing, look in the module for calls to drupal_alter() or module_invoke_all()
  - If still nothing, identify what you need and write a patch.
- Be future-proof though: there is the danger of future contrib updates breaking the custom code
  - Don’t assume the data structures, render arrays, etc, will always be the same
  - Use exceptions to make breakages very clear (eg, instead of empty())
  - Be cautious about using internal contrib functions

Maintaining your custom code

- Over-comment the code
- Keep it in its own folder (sites/all/modules/custom/)
- How to handle Features modules?
- Tool for producing HTML docs?
- Anything else?

Giving estimates

Problem: We (developers) are optimists, people-pleasers, and puzzle-enthusiasts

Solutions:

- Be realistic about time estimates
- Mention the level of customization needed
- Offer contrib alternatives (even if they don’t meet the requirements)
- Educate managers about the costs of customization

Modules that imitate custom code

- Context
- Conditional Fields
- Data
- Views
- Entity Construction Kit
- Display Suite
- Field Group
- Page Manager / Panels
- Token
- Rules

Summing up

- Custom code is fun. But it has a cost.
- Contributing to drupal.org is fun too.
- Get good at judging contrib projects.
- Think about the next developer. (which could be you)
- Don’t be afraid to hack.
- Drupal core/contrib is not a monolithic black box of software, it’s just code.

Thanks for listening!

Questions/comments?

You can also reach me at brockfanning@gmail.com