# LEAN, MEAN & MAINTAINABLE THEMING

## DRUPAL THEMING BEST PRACTICES

PRESENTED BY @MIKEHERCHEL


[ a poem ]

# ABOUT ME

- Working with Drupal for 6 years
- Working on the web for 12 years
- Awesome family
- Passionate about usability & ux
- Passionate about Drupal
- #FLDC14 website designer, themer, & volunteer
- Love crappy beer

HECKLE ME AT @MIKEHERCHEL

# PART 1: LAISSEZ-FAIRE THEMING

… a deliberate abstention from direction or interference

# "LAISSEZ-FAIRE" THEMING APPROACH

1. Does it work?
2. Does it work well?
3. Can it be abstracted for reuse?
4. Make it maintainable.
5. Know *everything* that's going on.

HECKLE ME AT @MIKEHERCHEL


THE PETER GRIFFIN
APPROACH TO THEMING

# KNOW WHAT EXACTLY IS GOING ON

- If you want to use a base theme, invest the proper amount of time and be *the* expert on this theme. Don't half-ass it.
- Know the mixins
- Know the theory
- Know your text editor

# BUILDING FROM THE BOTTOM UP

- Start on a low level
- Understand everything that's happening
  - Sometimes this means that you might not be using the new hotness
  - If you use Bootstrap/Foundation understand the hell out of it!

# "BASE" VS. "STARTER" THEMES

- Both provide a great starting point with integrated tools
- Both dramatically simplify markup
- Both can enforce best practices
- Base themes can be upgraded
- Starter themes do not have children – you modify them
- Base themes can sometimes be tough to troubleshoot
- Base theme updates can sometimes break things

HECKLE ME AT @MIKEHERCHEL

# DITCHING DRUPAL'S CSS CRUFT

- Template pre-process functions to remove CSS (https://github.com/mherchel/bastard/blob/master/template.php)

```
21  function bastard_css_alter(&$css) {
22    /* Remove some default Drupal css */
23    $exclude = array(
24      'modules/aggregator/aggregator.css' => FALSE,
25      'modules/block/block.css' => FALSE,
26      'modules/book/book.css' => FALSE,
27      'modules/comment/comment.css' => FALSE,
28      'modules/dblog/dblog.css' => FALSE,
29      'modules/field/theme/field.css' => FALSE,
30      'modules/file/file.css' => FALSE,
31      'modules/filter/filter.css' => FALSE,
32      'modules/forum/forum.css' => FALSE,
33      'modules/help/help.css' => FALSE,
34      'modules/menu/menu.css' => FALSE,
35      'modules/node/node.css' => FALSE,
36      'modules/openid/openid.css' => FALSE,
37      'modules/poll/poll.css' => FALSE,
```

# DITCHING CSS CRUFT

- Magic module
  (https://drupal.org/project/magic)

# SEMANTIC MARKUP

- Good base-theme, or starter theme
- Fences module
- Semantic Views module
- Borealis Semantic Blocks module
- Block Class module

# MY LAISSEZ FAIRE THEME

- Bastard Starter Theme
  (https://github.com/mherchel/bastard)
  - Bare bones
  - Strips out unwanted css
  - Some base styles (tabs etc)
  - Integrated responsive primary menu
  - Sass, modernizr, live-reload, etc
  - Very lean & semantic

HECKLE ME AT @MIKEHERCHEL

# WHY WORKS FOR ME

- Copied favorite parts from various themes (thanks GPL!)
  - HTML5_base
  - Wundertheme
  - Omega
- Wrote much of it
- I *know* what's going on!

HECKLE ME AT @MIKEHERCHEL

# PART 2: SOME BEST PRACTICES, TIPS, & TRICKS

# FRONT END DEVELOPMENT MOVES FAST!

- Don't be afraid to stick with what you know
- Don't be afraid to try out new technologies
- The trick is finding the right balance
- One or two new technologies per project

# SASS TIP: PARTIAL STRUCTURE

- Base themes like Aurora, Zen, & Omega 4 will do this for you!
- Structure examples
  - Base structure: http://thesassway.com/beginner/how-to-structure-a-sass-project
  - More info: http://bramsmulders.com/how-i-improved-my-workflow-with-smacss-sass.html
  - WunderTheme: https://github.com/Krimson/wundertheme
  - Bastard: https://github.com/mherchel/bastard
  - Make sure it makes sense to you !

HECKLE ME AT @MIKEHERCHEL

# SAMPLE SASS PARTIAL STRUCTURE

```
abstractions
- _mixins.scss
base
- _animations.scss
- _fonts.scss
- _forms.scss
- _media.scss
- _tables.scss
- _typography.scss
components
- _aside.scss
- _branding.scss
(still under components)
- _comments.scss
- _footer.scss
- _messages.scss
- _navigation.scss
- _pager.scss
- _tabs.scss
variables
- _breakpoints.scss
- _colors.scss
no-query.scss
print.scss
styles.scss
```

HECKLE ME AT @MIKEHERCHEL

# SASS TIP: MAXIMUM SELECTOR NESTING

- Don't follow DOM structure with sass structure
- Keep your selectors no more than 3 deep!
- Definitely no more than 4!
- Looking at the compiled CSS helps you spot areas for improvement
- OOCSS (object oriented CSS) helps with this

HECKLE ME AT @MIKEHERCHEL

# SASS TIP: MIXINS & EXTENDS

- Both greatly help simplify your sass
- Extends compile by adding your selector to the extended selector
  - Can be tricky with media queries
- Mixins add properties to your CSS selector
- Lots of great mixins on webhttp://zerosixthree.se/8-sass-mixins-you-must-have-in-your-toolbox

HECKLE ME AT @MIKEHERCHEL

# FIND AND USE MIXINS LIKE THIS DANDY

```
@function calculateRem($size) {
  $remSize: $size / 16px;
  @return $remSize * 1rem;
}

@mixin font-size($size) {
  font-size: $size;
  font-size: calculateRem($size);
}
```

## Usage

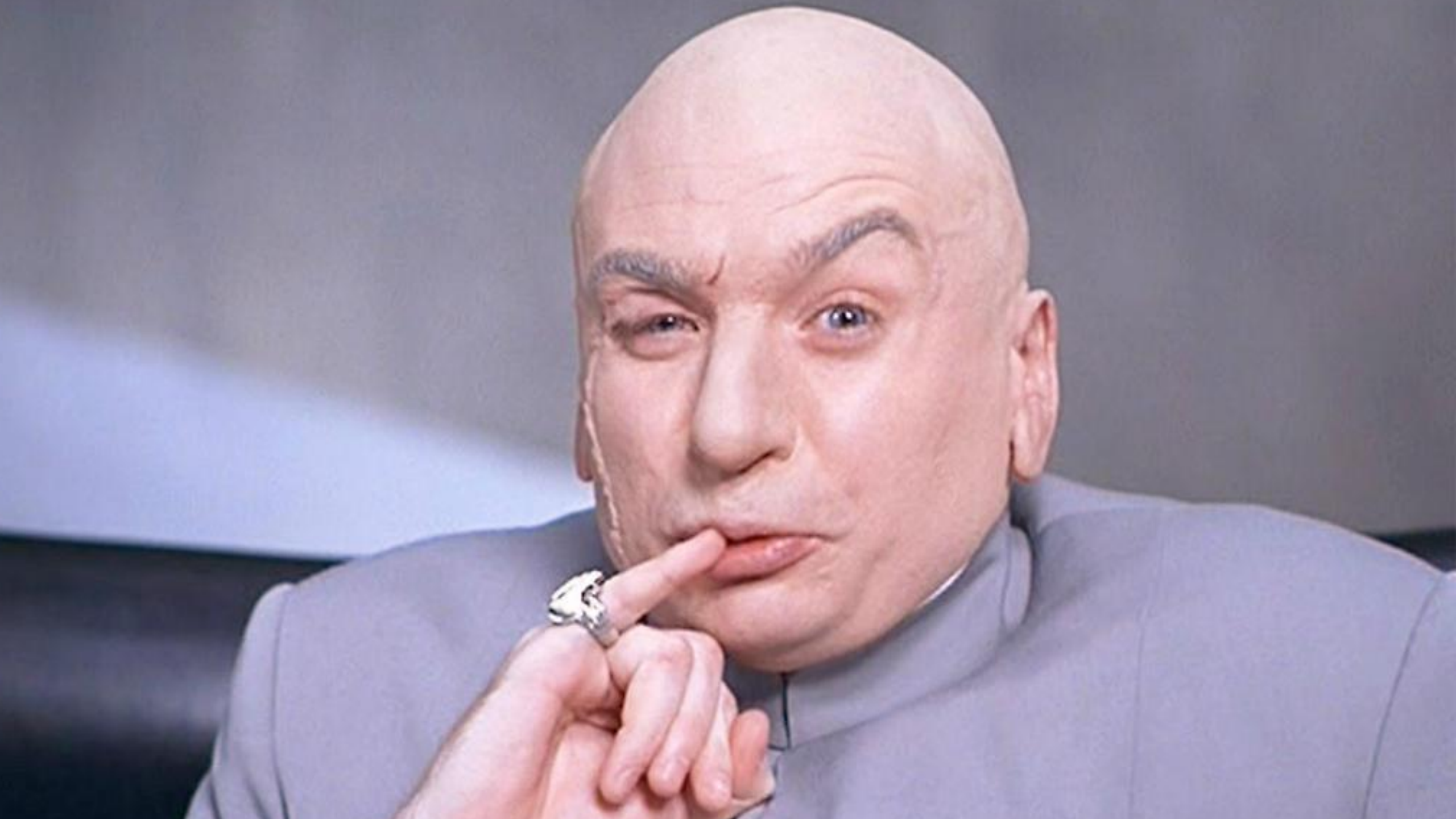
```
p {
  @include font-size(14px)
}
```

## Output

```
p {
  font-size: 14px; //Will be overridden if browser supports rem
  font-size: 0.8rem;
}
```

# SASS TIP: BREAKPOINTS

- Huge fan of the Breakpoint compass gem
  https://github.com/Team-Sass/breakpoint
- Enables developer to quickly and easily manage breakpoints and IE8 fallbacks
- Question: How many breakpoints?



# OBJECT ORIENTED CSS (OOCSS)

- In 2009 Nicole Sullivan first talked about OOCSS
- Consulted for Facebook
- Found over 6,498 color declarations
- 261 variations of “Facebook Blue”
- Something is wrong here

"THE PURPOSE OF OOCSS IS TO ENCOURAGE CODE REUSE AND, ULTIMATELY, FASTER AND MORE EFFICIENT STYLESHEETS THAT ARE EASIER TO ADD TO AND MAINTAIN."

http://coding.smashingmagazine.com/2011/12/12/an-introduction-to-object-oriented-css-oocss/

# OOCSS: CLASS NAMING

- Class names should communicate useful information to developers.
- It's helpful to understand what a specific class name is going to do when you read a DOM snippet

  Example: use `.article-list` instead of `.news`

http://nicolasgallagher.com/about-html-semantics-front-end-architecture/

# OOCSS: CONTENT INDEPENDENT

- Content-independent class names
- Content independent styling - Make it so when placed in different area looks good
  (location agnostic)

# OOCSS: BEM VS SMACSS

- SMACSS is more of a way to organize
- BEM is a great way to name your classes
- Why not use both
- And, sometimes none
    - … because its generally a lot easier to write CSS than to do php functions to add classes.

HECKLE ME AT @MIKEHERCHEL

# TIP: DEVELOP LOCALLY WITH REMOTE FILES

Have Drupal/Apache redirect files/* to live site

- State_file_proxy module
- Apache .htaccess rewrite rules
  - https://www.lullabot.com/blog/article/using-remote-image-files-when-you-develop-locally
  - Note: Add this code to the *beginning* of your .htaccess file

# TIP: MANAGING RUBY & GEM VERSIONS

- Two ruby managers
  - Ruby Version Manager (RVM) https://rvm.io
  - Ruby Environment (RBENV) https://github.com/sstephenson/rbenv
- Bundler (http://bundler.io) - Manage Ruby Gems

# TIP: RESPONSIVE MENU PATTERNS

- Create your own menu system (it's easy)
  - Works all the way back to IE8

```
.menu li ul { display: none; }
.menu li:hover ul { display: block; }
```

# TIP: RESPONSIVE MENU PATTERNS (CONT')

- To do a mobile menu, add some jQuery click events

```
$('.menu .nav-click').click(function(){
        $('.nav-click').toggleClass('nav-click-active');
        $(this).siblings('.menu li .menu').slideToggle();
})
```

- Or a toggle classes and show/hide in css
- The point here is if you write it, you understand it and can… *bend it to your will!!!*
- Code at https://github.com/mherchel/bastard/blob/master/js/scripts.js

**HECKLE ME AT @MIKEHERCHEL**

# TIP: GRID SYSTEMS

- Susy
- Singularity
- None (OMG!)
- Pick one, stick with it, know it inside and out

# TIP: REFRESH CSS AUTOMATICALLY

- Use LiveReload (http://livereload.com) to automatically refresh the css in your browser *without* a full page refresh!
- Makes in-browser development & design much more efficient
- Tips:
  - Add CSS directory to live-reload app
  - Installing the browser extension negates the need for the JS snippit

# MISCELLANEOUS TIPS

- Learn mobile-first development
  - Breakpoint is built to use it, and it simplifies your code
- Sweat the minor (visual) theming
  - Don't forget to theme Drupal's status messages
  - Use some transitions
- Don't sweat the extraneous wrapper div its going to be a PITA

# PART 3: TROUBLESHOOTING AND RESOURCES

# DEVELOPING FOR IE8

- Test early and test often
  - Make it easy for yourself to test
  - You'll get a sense of what works and what doesn't
  - Test menus
  - Test positioning
- Breakpoint gem can create no-query fallback
  - Load this using IE Conditionals

# PROGRESSIVE ENHANCEMENT VS GRACEFUL DEGRADATION

- Progressive Enhancement: Start at the least compatible browsers (IE8, Android 2.x) and work up
- Graceful Degradation: Start at the most compatible browsers and develop fallbacks for earlier
- Use combination of both
- User Modernizr to help (http://modernizr.com)

# BECOME A DEV-TOOLS POWER USER!

- Chrome dev tools
- FF dev tools
- IE dev tools (OMG!)
- Resources
  - http://devtoolsecrets.com
  - https://developers.google.com/chrome-developer-tools

HECKLE ME AT @MIKEHERCHEL

# FRONT END RESOURCES

- http://zerosixthree.se/8-sass-mixins-you-must-have-in-your-toolbox
- https://github.com/snugug/north * This is awesome
- http://nicolasgallagher.com/

# QUESTIONS?