DRY CSS

A DON'T-REPEAT-YOURSELF METHODOLOGY FOR CREATING EFFICIENT, UNIFIED AND SCALABLE STYLESHEETS

© Jeremy Clarke • http://jeremyclarke.org

Download these slides:
http://slideshare.net/jeremyclarke

WHO IS JEREMY CLARKE?

- Communications Studies at Concordia University.
- HTML+CSS since 2003
- Montreal WordPress Community/ WordCamp organizer.

Montréal WordPress Community

simian uprising by jeremy clarke

OVERVIEW

- An example of DRY CSS, Global Voices.
- DRY principle and problems with CSS.
- Preparsers (LESS and SASS) and their issues.
- DRY CSS theory, implementation and benefits.
- Appendix: OOCSS

DRY EXAMPLE: GLOBAL VOICES

GlobalVoices in Aymara, bahasa Indonesia, Deutsch, Español, Français, Italiano, Malagasy, Nederlands, Português, srpski, Svenska, polski, Ελληνικά, Македонски, русский, عربي, বাংলা, 한국어, 简体中文, 繁體中文, 日本語 Learn more »

English

GlobalVoices

The world is talking, are you listening?

About Get Involved Press Contact Donate Subscribe

Global Voices is an international community of bloggers who report on blogs and citizen media from around the world. Learn why »

Search Countries Topics Contributors

Photos Video

Search

Thank you for supporting us!

"I rely on Global Voices to help me make sense of the world and connect me to other communities and other perspectives."

Amy Selwyn, United States

Donate now »

Featured stories

Bahrain: #Hungry4BH Trends Worldwide

1 2 3

China, Koreas: Netizens Express Humanitarian Concern Over North Korean Refugees 26 February 2012

Liberia: "Kill the Gays" Bill Spreading 24 February 2012

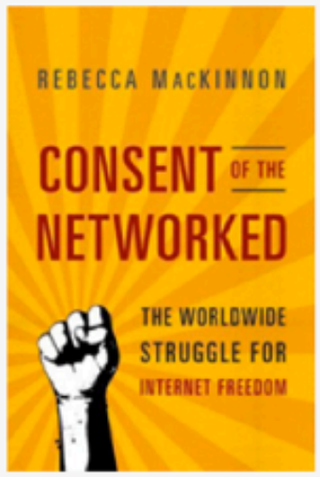
Brazil: The Deficient Prison Systems of the Americas 24 February 2012

Chile: Twitter Users Share Images of Aysén Protests 23 February 2012

Sahel: Islamic Fundamentalist Organisations Tighten Their Grip 22 February 2012

Special Coverage

New book from Global Voices co-founder Rebecca MacKinnon

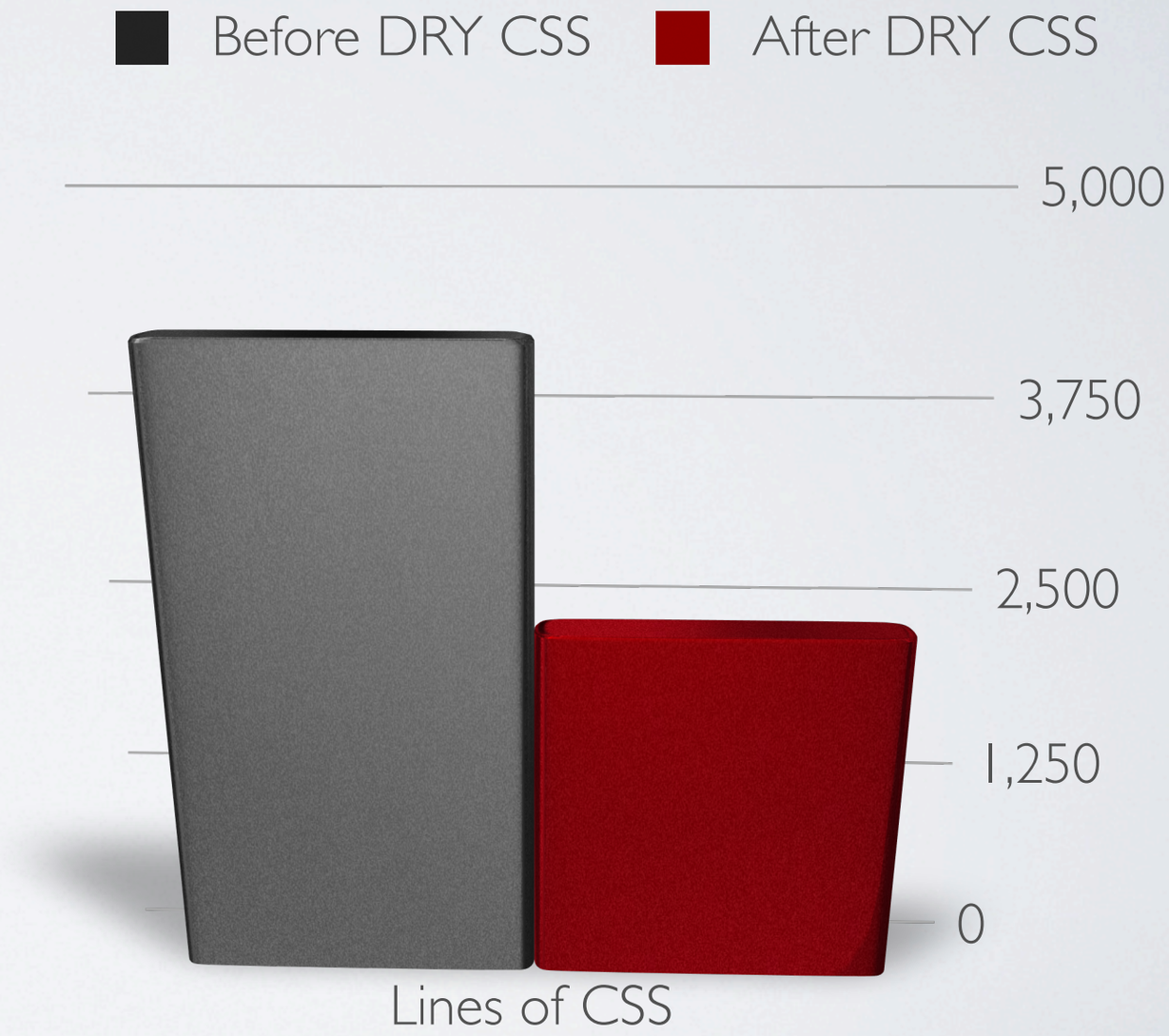


In Consent of the Networked, internet policy specialist Rebecca MacKinnon argues that the purpose of technology is to serve humanity, not the other way around. It's time to wake up and act before the reversal becomes permanent.

Buy your copy through this link and

DRY EXAMPLE: GLOBAL VOICES

- Applying DRY reduced 4200 lines to 2400.
- Standardized Shapes, text sizes, colors and modules.
- Nothing lost, just better organized.

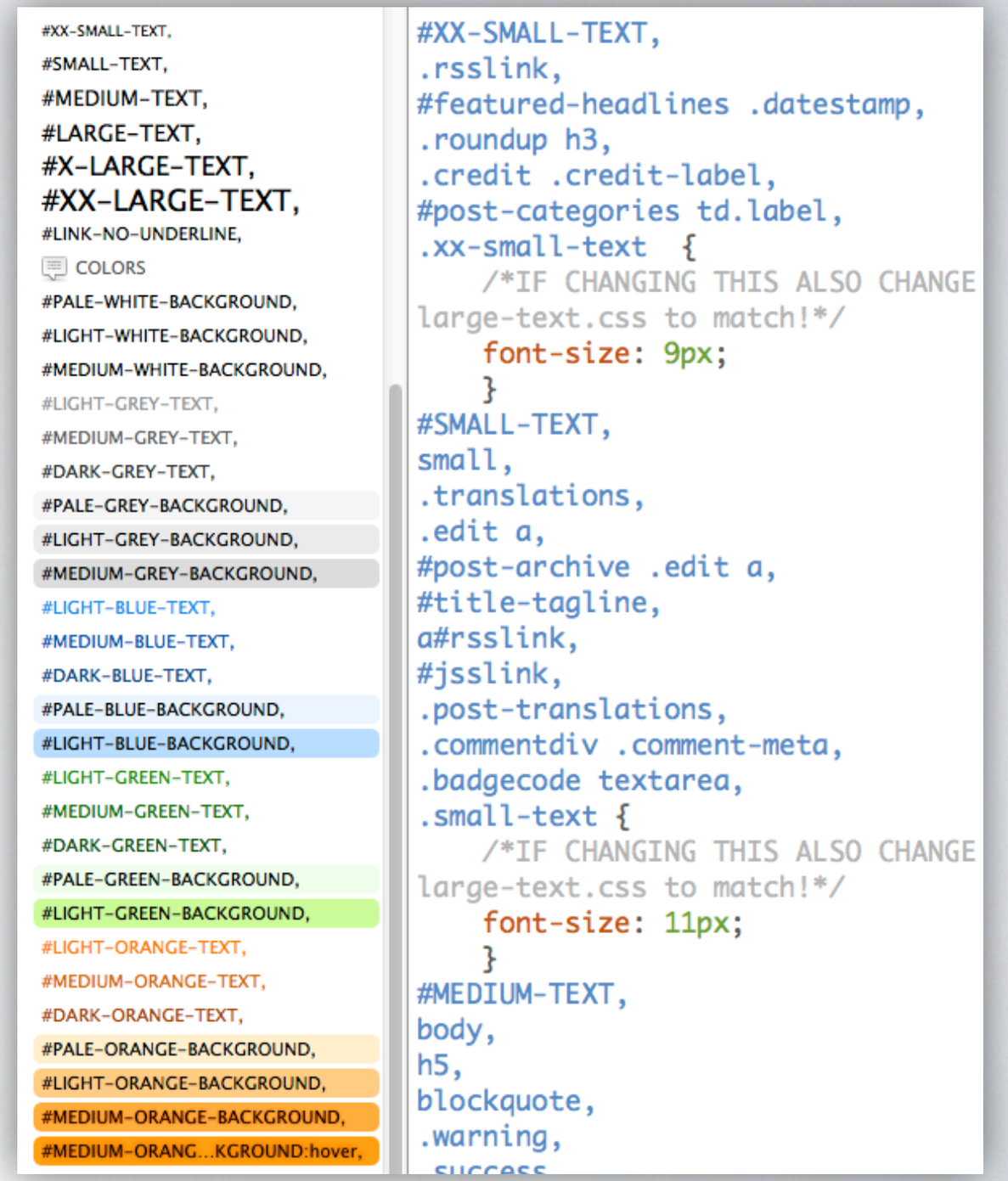


DRY EXAMPLE: GLOBAL VOICES

- Groups define shared properties.

- Groups have many selectors.

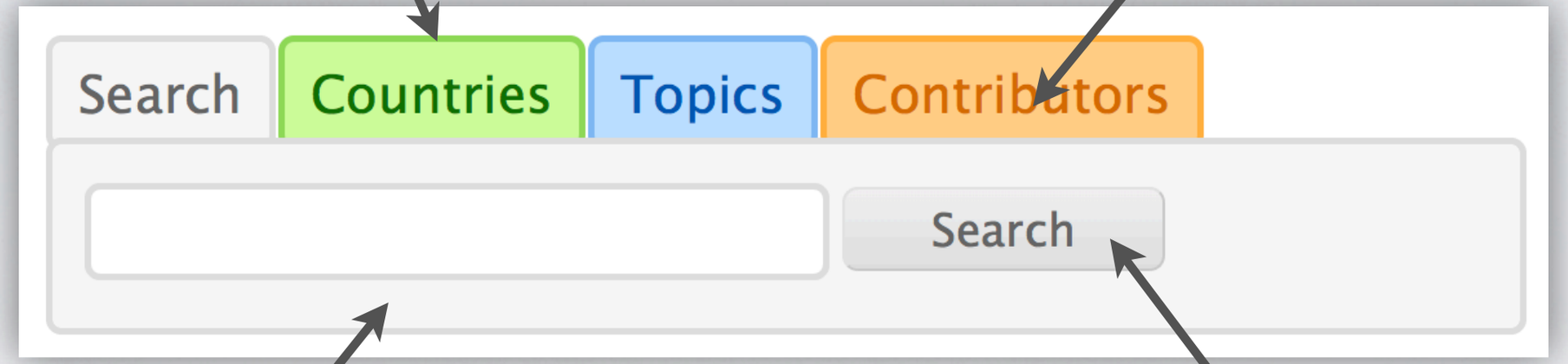
- Each property/value pair is defined only once.



DRY EXAMPLE: GLOBAL VOICES

#ROUNDED5-BORDER2

#MEDIUM-ORANGE-TEXT



#PALE-GREY-BACKGROUND

#GREY-BUTTON
#GREY-BUTTON-HOVER
#GREY-BUTTON-CLICK

DRY EXAMPLE: GLOBAL VOICES

- Elegantly organized IE hacks and Right-To-Left (RTL) styles into groups for the common fixes.

- Font-size groups re-used in optional Large-Font theme.

```
#ZOOM-FIX,
#disambiguation,
.translations,
.dategroup,
#post-archive,
.tabcontainer,
.archive-header,
.archive-meta,
.media-icons .media-type,
.recent-articles,
.feed-items .feed-item,
.sidebar .recent-comment,
.commentdiv,
.animated-form-trigger,
#single-tools,
#full-article,
.sharing-tools,
#respond,
#foot-container,
.sponsors-badges,
.user-list-avatars,
.clear {
    /*This kicks in has-layout in ie6-7
    Can potentially solve various problems
    related to containing boxes that lack
    substance (borders, floats etc)*/
    zoom: 1;
    }
```

2 PRINCIPLES OF GOOD CSS

- **Keep style separate from content.**
  - HTML should be structural. Tags, classes and IDs should refer to the content itself rather than how it looks.
- **Avoid specificity by harnessing the cascade.**
  - Selectors and definitions should be as global as possible for design consistency and to avoid specificity collisions.

DRY: DON'T REPEAT YOURSELF

- a.k.a "Single Source of Truth" principle.
- "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system." *
- DRY code means all uses of data change simultaneously rather than needing to be replicated.
- DRY is the fundamental reason to use of variables and functions in all programming.
- **Bugs love a wet mess.**

* http://en.wikipedia.org/wiki/Don't_repeat_yourself

PROBLEMS WITH KEEPING CSS DRY

- **No variables** - We can't re-use values across properties (I.e a color hex).

- **No mixins** (functions) - No way to re-use sets of property/value pairs across definitions.

- Result: Constant duplication of style definitions across different selectors.

BRUTE FORCE: CSS PREPARSERS

- Add a new language based on CSS that has the necessary tools and a new parsing layer to read it.
  - Variables to re-use values across definitions.
  - Mixins to include a class inside another class.
  - All kinds of other programmy things (functions, switch conditions)

* E.g. http://lesscss.org/ and http://sass-lang.com/

```
@color: #4D926F;

#header {
  color: @color;
}
h2 {
  color: @color;
}
```

Variables in LESS*

```
.rounded-corners (@radius: 5px) {
  border-radius: @radius;
  -webkit-border-radius: @radius;
  -moz-border-radius: @radius;
}

#header {
  .rounded-corners;
}
#footer {
  .rounded-corners(10px);
}
```

Mixins in LESS*

CSS PREPARSERS: ISSUES

- LESS and SASS do create a fantastic, DRY-capable CSS.
- **BUT** at the cost of destandardization and dependence on external libraries. **It isn't CSS.**
- They can still be misused to create wet, sloppy stylesheets. Organizational methodology remains vital.
- Question: How can we get the same effect with standard CSS?

PRINCIPLES OF DRY CSS (TL;DR)

- **Don't Repeat Yourself**. Never repeat a style/property definition if you can avoid it.

- **Group selectors with shared properties** rather than defining them separately.

CREATING DRY CSS

- Group selectors that share properties above the properties they share.

- Name the groups based on their role in the design.

- Use the name as an ID at the top of the list and a class at bottom.

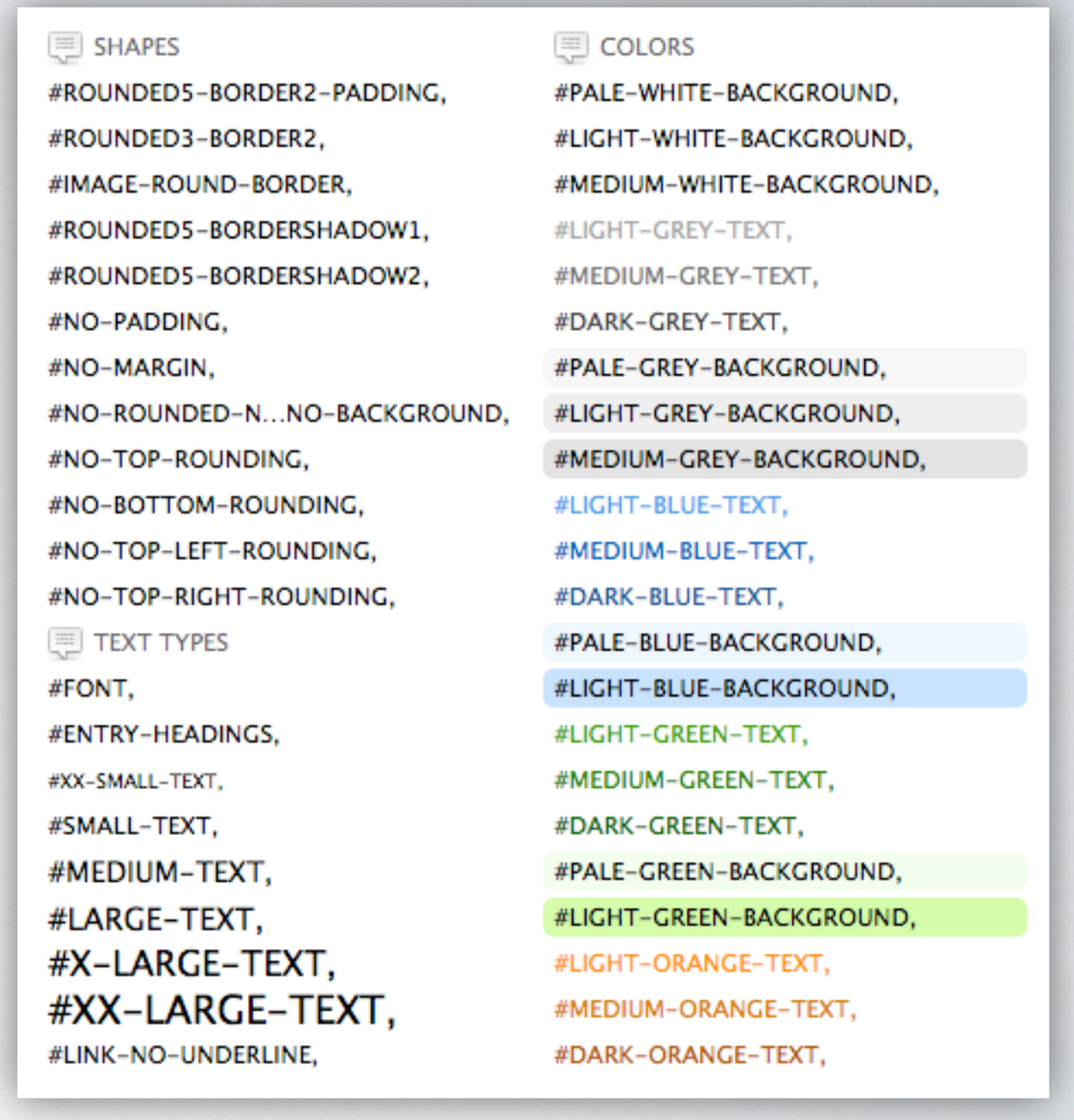
```
#LIGHT-WHITE-BACKGROUND,
.translation,
.entry .wp-caption,
#full-article .entry img,
.recent-comment .comment-text,
.roundup h3,
.post-header-sharing,
#post-categories td.label,
#post-archive .roundup h3,
.subscription-manager ol,
.light-white-background {
    background-color: #fff;
    border-color: #ccc;
    }
#MEDIUM-WHITE-BACKGROUND,
textarea:focus,
input:focus,
#author-email-form input:focus,
#respond p input:focus,
.wpcf7 input:focus,
.medium-white-background {
    background-color: #fff;
    border-color: #bbb;
    }
```

ORGANIZING DRY CSS

- Decide on group types and groups as necessary for your design, operational needs and/or other CSS methodologies.

- Recommended: Colors, Text, Shapes, Structures, Modules.

- At right: GV's DRY groups, visualized by CSSEdit for Mac*



* http://macrabbit.com/espresso/

KEEPING IT DRY

- Make individual selectors as rare and sparse as possible, only use them as exceptions.

- Always have an answer to "Why isn't this part of a group?"

- Don’t go crazy. Benefits scale even if some definitions remain moist.

```
.warning {
    margin: 10px 0 20px 0;
    text-align: center;
    }
.error {
    font-weight: bold;
    }
.edit a {
    padding: 1px 3px;
    }
.more,
.more-link,
a.more{
    display: block;
    text-align: right; /*RTL*/
    margin: 0;
    padding: 0;
    }
```

BENEFITS OF DRY CSS

- 2 simple rules to follow that ensure a solid organizational structure.

- Less CSS, shorter and smaller (kb) stylesheets.
- Promotes good design practice and uniform updates.
- Uses only philosophically and syntactically standard CSS.

BENEFITS:
PROMOTES CONCEPTUAL DESIGN

- Encourages you to think in terms of style patterns (groups), rather than individual objects.

- Naming of groups encourages rational organization of designs.

- Overall design tightens up, related objects become stylistically paired rather than simply matching.

- Listing all group members together encourages optimization and generalization of selectors because you see how interrelated/inheritable they are.

BENEFITS:
EDITS TO GROUPS AFFECT ALL MEMBERS

- Avoids need to find all related styles to match changes.
- Changes are less likely to be inconsistently applied.
- When testing changes the dev sees all group members change as well.
- Moving selectors between groups to restyle is fast and easy.

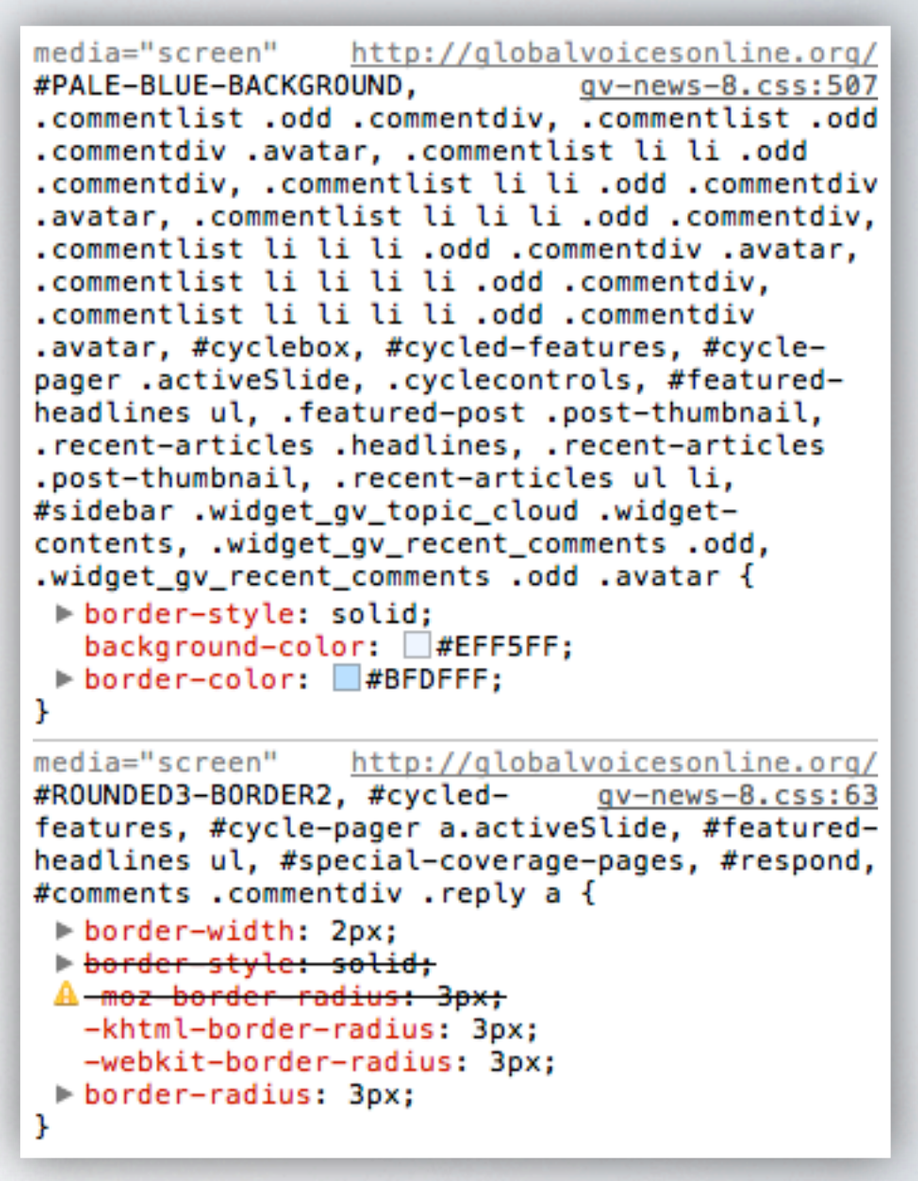
BENEFITS:
TAKES ADVANTAGE OF INSPECTORS

- Inspecting an object shows cascade of groups it participates in.

- Group "name" (starting ID, i.e. #SMALL-TEXT) is visible in inspector.

- Seeing all group members informs dev of related elements that should be considered.

- Live-edits expose the change in related items as well.



BENEFITS:
DOESN'T REQUIRE CHANGES TO HTML

- Uses whatever classes and IDs are already present.

- If HTML needs editing it's only to add more detailed semantic classes to use in your DRY groups.

- Useful when generated HTML is out of your control (WordPress core/plugins, 3rd party widgets)

BENEFITS: COMPLETELY STANDARD

- Simply an organizing princinple for normal CSS.
- Adheres to separation of style and content.
- Both backward and forward compatible, from IE6 hacks to prefixed CSS3 properties.
- CSS can be pasted anywhere and just work.
- Groups can be extracted and re-used elsewhere by removing group members and inserting new ones.

BENEFITS: INTEGRATES WITH OTHER METHODOLOGIES

- Compatible with most other CSS methodologies like grids, OOCSS* or SMACSS** because the way you organize your groups is up to you.

- Progressive enhancement and browser support is your choice. DRY only defines how you organize your selectors and properties.

* https://github.com/stubbornella/oocss/wiki ** http://smacss.com/book/

PERFORMANCE ISSUES?

- **TL;DR: Probably not.**

- "For most web sites, the possible performance gains from optimizing CSS selectors will be small, and are not worth the costs." ~Steve Sounders*

- Having many selectors does not inherently hurt performance.

- See Steve Sounders research** for optimizations that matter (and can be applied to DRY CSS).

* http://www.stevesouders.com/blog/2009/03/10/performance-impact-of-css-selectors/
** http://www.stevesouders.com/blog/2009/06/18/simplifying-css-selectors/

OOCSS: "OBJECT ORIENTED"

- "Separate structure and skin."
- "Separate container and content."
- Objects are re-usable content patterns.
- Skins are sets of visual decorations to be applied to objects.
- Implementation
  - Define re-usable classes like .media, .box and .red
  - Add them to HTML tags throughout your site to create a consistent and efficient design.

Sources
https://github.com/stubbornella/oocss/wiki/FAQ
http://slideshare.net/stubbornella/object-oriented-css

DRY CSS <3 OOCSS

- The fundamental principles and goals are the same:
  - Create logical style groups that define your site.
  - Apply them to elements rather than redefining css on many selectors.
  - Focus on consistency of objects across the site through direct style linkage.
- "Object-orientation" can be integrated into a DRY approach when choosing your groups and group types.

DRY CSS > OOCSS

- The difference:
  - DRY stacks selectors above a single definition in the CSS.
  - OOCSS applies generic classes to HTML tags throughout the document.
- OOCSS violates the separation of content and style by polluting HTML with meaningless style-classes.
- This is **unnecessary** because DRY CSS can achieve the same goal while maintaining separation.

DRY CSS > OOCSS (DETAILS)

- DRY offers an easy way to route around situations where OOCSS classes can't be added to HTML (out of your control, inconvenient, undesirable).

- Generic group class (i.e .small-text) for each DRY group can still be used in OOCSS way if needed (especially in transient *content* rather than templates).

- DRY allows group membership to be managed without editing HTML templates. Less files to edit, less people involved.

- DRY keeps style and content separate LIKE IT SHOULD BE.

DRY CSS REVIEW

Group re-usable properties

```
. {
    background-color: #fff;
    border-color: #ccc;
    }
. {
    background-color: #fff;
    border-color: #bbb;
    }
```

Name the groups logically

```
#LIGHT-WHITE-BACKGROUND,
.light-white-background {
    background-color: #fff;
    border-color: #ccc;
    }
#MEDIUM-WHITE-BACKGROUND,
.medium-white-background {
    background-color: #fff;
    border-color: #bbb;
    }
```

```
#LIGHT-RED-TEXT,
#MEDIUM-RED-TEXT,
#PALE-RED-BACKGROUND,
#LIGHT-RED-BACKGROUND,
BUTTONS
#GREY-BUTTON,
#GREY-BUTTON-HOVER,
#GREY-BUTTON-CLICK,
```

Add selectors to various groups

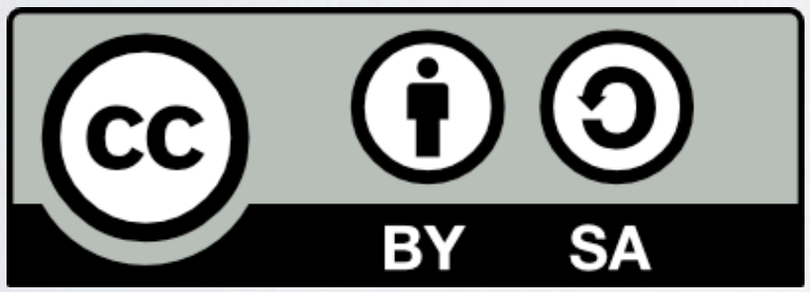
```
#LIGHT-WHITE-BACKGROUND,
.translation,
.entry .wp-caption,
#full-article .entry img,
.recent-comment .comment-text,
.roundup h3,
.post-header-sharing,
#post-categories td.label,
#post-archive .roundup h3,
.subscription-manager ol,
.light-white-background {
    background-color: #fff;
    border-color: #ccc;
    }
#MEDIUM-WHITE-BACKGROUND,
textarea:focus,
input:focus,
#author-email-form input:focus,
#respond p input:focus,
.wpcf7 input:focus,
.medium-white-background {
    background-color: #fff;
    border-color: #bbb;
    }
```

QUESTIONS?

DRY CSS

A DON'T-REPEAT-YOURSELF METHODOLOGY FOR CREATING EFFICIENT, UNIFIED AND SCALABLE STYLESHEETS

Jeremy Clarke • http://jeremyclarke.org



Creative Commons Share Alike
http://creativecommons.org/licenses/by-sa/3.0/

Download these slides:
http://slideshare.net/jeremyclarke