OOCSS - Object Oriented CSS

Introduction

- What is OOCSS? Why OOCSS? Check out the video at http://wiki.github.com/stubbornella/oocss/
- We've taken these raw ideas, changed a few things, and added A LOT more

Outline

- Objects: Interface Objects in the HL Platform
- Components: Parts of Objects
- Views: Different ways to view an Object or Component
- Decorators: Decorative variations of an Object or Component
- Gotchas: Things to watch out for

Objects

- Objects are "things". They describe what something is, NOT how it looks.
- Objects can be platform objects (e.g. User, Group, Hive, Post, Comment)
- Objects can be UI objects (e.g. Layout, List, Dialog, Widget)
- All Objects begin with HL_ for namespacing (e.g. HL_List, HL_Widget)
- OOCSS uses CSS classes to style objects

```
<!-- Example Objects -->

<div class="HL_Widget">...</div>

<ul class="HL_List">...</ul>
```

- Objects can be extended using an underscore (e.g. HL_Widget => HL_Widget_Search => HL_Widget_Search_Advanced)
- Extended objects inherit styles from parent classes, as ancestor CSS classes are included in the markup

```
<div class="HL_Widget HL_Widget_Search
HL_Widget_Search_Advanced">...</div>

<ul class="HL_List HL_List_Object
```

- Generically, the word Object means something under the HL namespace (HL_*)
- When used explicity, Object refers specifically to platform objects (e.g. HL_Object_Post, HL_List_Object_Post).
- We know this is confusing. Sorry.

Components

- Objects can contain any number of Components, which are specified with a hyphen (e.g. HL_Object-Name)
- Components can only live in Objects (e.g. HL_Object-Name always appears in HL_Object)

```
<div class="HL_Object HL_Object_Post">
    <span class="HL_Object-Name">Aunt Martha's
Meatballs</span>
</div>
```

- Components can be combined into Meta Components (e.g. HL_Object-Name + HL_Object-Count = HL_Object-Title)

```
Component -->

<div class="HL_Object HL_Object_Post">
    <div class="HL_Object-Title">
        <span class="HL_Object-Name">Aunt Martha's
Meatballs</span>
        <span class="HL_Object-Count">5
Comments</span>
```

- Components can also have Sub Components (e.g. HL_Object-Field: HL_Object-Field-Name, HL_Object-Field-Value)
- What's the difference between a Sub Component and a Component in a Meta Component? Sub Components can't live outside their parent Component, while Components in a Meta Component can be freestanding or even exist in other Meta Components

```

<div class="HL_Object HL_Object_Post">
    <div class="HL_Object-Title">
        <span class="HL_Object-Name">Aunt Martha's
Meatballs</span>
        <span class="HL_Object-Count">5
Comments</span>
    </div>
    <div class="HL_Object-Fields">
        <div class="HL_Object-Field">
            <div class="HL_Object-Field-
Name">course</div>
            <div class="HL_Object-Field-
Value">dinner</div>
        </div>
        <div class="HL_Object-Field">
            <div class="HL_Object-Field-
Name">ingredients</div>
            <div class="HL_Object-Field-Value">ground
beef, onions</div>
        </div>
        <div class="HL_Object-Field">
```

- Components (and Sub Components) can be extended (HL_Object-Name => HL_Object-Name_Post)

```
<!-- Extended Components -->

<div class="HL_Object HL_Object_Post">
    <div class="HL_Object-Title">
        <span class="HL_Object-Name HL_Object-
Name_Post">Aunt Martha's Meatballs</span>
        <span class="HL_Object-Count HL_Object-
Count_Comment">5 Comments</span>
```

- Any Object can have Components, including extended Objects (e.g. HL_Widget_Content_Header-Title)

Views

- Objects can have Views, which define different ways you can view them
- Views are generally different combinations of components (e.g. HL_ObjectView_Title vs HL_ObjectView_Clip)

```
<!-- Adding a View to an Object -->

<div class="HL_Object HL_Object_Post
HL_ObjectView_Title">
    <div class="HL_Object-Title">
        <span class="HL_Object-Name HL_Object-
Name_Post">Aunt Martha's Meatballs</span>
        <span class="HL_Object-Count HL_Object-
```

- Views can be extended (e.g. HL_ObjectView_Cloud => HL_ObjectView_Cloud_XL)
- Like extended Objects, extended Views inherit styles from ancestors

```
<!-- Extended Views -->

<div class="HL_Object HL_Object_Post
HL_ObjectView_Cloud HL_ObjectView_Cloud_XL">
    <span class="HL_Object-Name HL_Object-
```

- Objects can only have 1 View per level (e.g. HL_ListView_Inline, HL_List_ObjectView_Cloud)

```
<!-- Multiple Views (1 per Level) -->

<div class="HL_List HL_ListView_Inline HL_List_Object
HL_List_ObjectView_Cloud HL_List_Object_Post">
    <div class="HL_Object HL_Object_Post
HL_ObjectView_Cloud HL_ObjectView_Cloud_XL">
        <span class="HL_Object-Name HL_Object-
Name_Post">Aunt Martha's Meatballs</span>
```

- Components can also have views, which follow all the same rules (e.g. HL_Object-FieldsView_Clip, HL_Object-FieldsView_Full)

```
<!-- Component View -->

<div class="HL_Object HL_Object_Post">
    ...
    <div class="HL_Object-Fields HL_Object-
FieldsView_Clip">
        <div class="HL_Object-Field">
            <div class="HL_Object-Field-
Name">course</div>
            <div class="HL_Object-Field-
Value">dinner</div>
        </div>
```

- Any Object can have Views, including extended Objects (e.g. HL_List_ObjectView_Title, HL_List_ObjectView_Clip)
- Top level Views are Views that can be used on any object or component (e.g. HL_View_IconAndText), while namespaced Views (e.g. HL_ObjectView_Clip) can only be used in their specific namespace (e.g. HL_Object_*).
- Not all Objects have Views. I.e. Views are usually only specified if the Object supports more than one.

Decorators

- Objects can have Decorators, which provide decorative variations of a given Object
- What's the difference between a View and a Decorator? Views are generally used

to provide different zoom levels of an Object, while Decorators provide variations w/in a View
- Decorators use the keyword 'Style' in OOCSS classes

```
<!-- Styles on a Site Message Widget -->

<div class="HL_Widget HL_Widget_Message
HL_Widget_Message_Site HL_Style_Success">
    <div class="HL_Widget-Body">Your request was
successful</span>
</div>

<div class="HL_Widget HL_Widget_Message
```

- Although Decorators are often decorative in nature, they shouldn't have explicit styling in their name (e.g. we use HL_Style_Error instead of HL_Style_Red)
- While Objects may only have one View per level, Objects may have as many Decorators as necessary
- Any Object can have Decorators, including extended Objects (e.g. HL_WidgetStyle_Admin)
- Many Decorators are conceptually top level Decorators. E.g. HL_ObjectStyle_Answer and HL_WidgetStyle_Answer would ideally use a top level Decorator (e.g. HL_Style_Answer) in a compound class selector (e.g. .HL_Object.HL_Style_Answer). However, IE6's lack of compound class support[2] often caused us to split common Decorators into different namespaces.

Gotchas

1. Be careful styling high level objects (e.g. HL_List) as it will have many downstream effects in extended classes
2. Don't use compound class selectors (e.g. .HL_Widget.HL_Style_Highlight) as IE6 does NOT support them. Using them at all, even for other browsers, can have unintended consequences in IE6 as it reads a compound class selector as simply a rule for the last class specified (e.g. .HL_Style_Highlight). Any classes you see with _DOT_ in the middle (e.g. HL_Object-Date_DOT_HL_View_Badge) are conceptually compound classes, but were morphed into one for IE6.
3. Beware of specificity

EXTRA: Attributes

- Object attributes provide name/value pairs in a single CSS class, and are specified with a double underscore
- Currently, they are only used for User bling based on UserType Id (e.g. HL_Object-Type_User__6, where 6 is the User's UserType Id)
- An attribute is similar to a component. While a component provides the name as a CSS class and the value in the markup, an attribute provides both in a CSS class. One advantage of an attribute is that b/c it is added to the Object itself, the entire Object can be styled based on an attribute value.

```
<!-- UserType Id in a Component -->

<div class="HL_Object HL_Object_User
HL_ObjectView_Summary">
    <span class="HL_Object-Type HL_Object-
Type_User">6</span>
</div>

<!-- UserType Id in an Attribute -->

<div class="HL_Object HL_Object_User
```