Majestic Ruby

Working on PDF Generation in Ruby via Prawn has completely warped my mind. To support the huge featureset we have amassed, we need to track an incredible amount of state, and implement all levels of abstraction down from the little DSL that was used to generate this handout, to a binary file format that has a 1300+ page ISO specification describing how it should work.

I have never worked on such a hard project in my life, both from a design and implementation perspective. As we struggle to finally be able to release a 1.0 product, I am responsible for coming up with fixes to the design flaws I baked into the system over two years ago when I couldn't even dare to guess what sort of architecture we'd need.

Here's what I've come up with so far, we'll pick a few to demonstrate based on what topics are most interesting to the class.

Design Guidelines

- Non-local state should only be directly manipulated in constructor methods or constructor-like methods.
- Eliminate as much private functionality as possible by building lower level objects that serve as a developer API.
- Modules should not ever directly manipulate non-local state (such as instance variables).
- DSL Implementions should proxy to or delegate to an underlying object, not just use a raw instance_eval() on some object.
- Small functional components should be defined via the Module "extend self" idiom rather than using a class definition.
- Class definitions should consist of nothing more than constructors and accessors, relying on mixins for all behavior.
- Use lazy evaluation wherever possible.