**Modulr**

Project Manager: Benjamin Chen ([bcfour@princeton.edu](mailto:bcfour@princeton.edu))  
Group Members

Michael Lee ([mmltwo@princeton.edu](mailto:mmltwo@princeton.edu))  
Antonio Juliano ([ajuliano@princeton.edu](mailto:ajuliano@princeton.edu))

**Overview**

We intend to build a Google Chrome extension that splits a webpage into various modules that the user can move around and customize. Many modern websites present their articles in a cluttered fashion, including extraneous information such as advertising and recommendations: This extension will give news-reading users the freedom to organize their information the way they like it, reducing annoyances and increasing readability.

The extension will allow users to customize the modules on a webpage, by changing its position, color, font, and/or orientation. A particular format can be saved for all articles from a given website, so that the extension automatically filters out the unwanted page elements. Additionally, for the users who do not want to customize so intensely, the extension will offer simple ways to view the wanted relevant elements (title, author, date, text, comments). Global settings regarding certain constants (i.e. font, text color), can be saved across viewing websites.

This extension will be programmed in Javascript with the jQuery library. Node.js and MySQL will be used for server side interactions. Similar products include Readability (<http://www.readability.com/>) , Adblock (<http://adblockplus.org/en/chrome>), and Kickass (<http://kickassapp.com/>), which all incorporate some degree of modularization.

**Functionality**

Scenario 1:  
An older user, not very tech-savvy, just wants to read articles off of his/her favorite website (i.e. NYTimes). He just wants to read the article, without the distractions of ads or extra information (i.e. pictures/captions, comments). Simply by using the ‘simplify’ function, our extension extracts the relevant text (title, author, date, article) and puts them on a single page (or more, depending on the length of the article).  
  
Scenario 2:  
A websurfer notices that a webpage has a horrible design with yellow text on a white background. He can use this extension to modularize the page and customize all text modules to have black text and his favorite font.   
  
Scenario 3:  
Suppose you happenchance upon a webpage full of pictures of your favorite actresses. Unfortunately, these pictures are improperly rotated, diminishing your viewing experience. With this extension, you can modularize the webpage and rotate all of the picture modules to the correct orientation.   
  
Scenario 4:  
A young web user wants to read articles off of the Daily Princetonian in a certain format -- get rid of non-relevant information and pictures, but keep the comments. He edits the page using the extension, saves the format (saving the site as dailyprincetonian.com) and is now able to browse all articles on the Daily Princetonian with this filter applied.  
  
Scenario 5:   
An older user has trouble reading articles with small text (including captions). He/she could use this extension to create a global rule to increase text size for the articles he reads.

**Design**

Every Google extension has an options page where a user can customize the extension to his or her own liking. This options page will allow the user to set a global template (e.g. all text modules comes in Arial font) or other logistics such as blacklisting pages that should not be modularized, modularizing only for specific domains, clearing template data, or clearing templates for specific pages.

During normal operation, the extension’s icon sits in the top right of the window next to the other extensions that the user may have. Clicking this icon either initiates the modularization process of the current webpage or reloads a saved template for that page. This modularization is a client side process (as opposed to calling a remote cgi script) that uses Javascript with the jQuery library. Clicking this icon again saves the current state of the webpage into a template, which is sent to a remote database running MySQL and Node.js. Because Chrome extensions have a maximum local storage size of 5 MB, the template data should be saved remotely rather than locally.

**Milestones**  
03/18-03/24 (Spring Break): Get “Hello World” chrome extension working. Get client-server database interaction working.  
03/25-03/31: Be able to divide webpage into modules.   
04/01-04/07: Be able to customize modules (orientation, font, color, etc.). Be able to translate modules into a template that can be read from and sent to a server.  
04/12: Project Prototype  
04/08-04/14: Implement the client-server interaction with a translatable template  
04/15-04/21: Establish standards for saving templates across specific domains, add additional functions (relevant information only ala Readability, etc)  
04/26: Alpha Test  
04/22-04/28: Should have working project; if we have time, add ability to send templates to friends, link others to your template.  
05/02 and 05/03: Beta Test (Last Class)  
05/08,05/09,05/10: demo days  
Dean’s Date: Final Submission

**Risks and Open Issues**

- Some of the members will have to learn about Javascript, MySQL, jQuery, Node.js from scratch.   
-We have to make sure that the database that stores template information is reliable.   
-If the data is stored remotely, the processing speed of the extension may be too slow.   
-Scaling the application may require porting to more scalable servers (maybe Amazon EC2)  
-There may be complications with getting on the Chrome Web Store

**Timeline**

<http://www.princeton.edu/~mmltwo/>