@(vfl) = **Vanilla Flavored LaTex**
@($\text{github}_u rl$) = $http://github.com/adhithyan15/vanilla$
@(tmaker) = **TexMaker**
@(ftb) = formatted text blocks
(cftb)[] = [$(capitalize)[formattedtextblocks]$]
$(importpack)[documentation]$

Vanilla Documentation

Adhithya Rajasekaran

What is Vanilla?

Vanilla is a powerful LaTex preprocessor. It works based on the DRY(Don't Repeat Yourself) principle. It aims to reduce the entry barrier and learning curve for Latex by simplifying the syntax and also reducing the verbosity of Latex. In this documentation you will learn how to write Vanilla flavoured Latex documents. This document itself is written in Vanilla flavoured Latex.

Let us start off with why I created Vanilla.

Why Vanilla?

I was trying to teach my mom, who is a high school math teacher, to use Latex. I was unsuccessful in that endeavour because my mom found Latex syntax very verbose and she hated the steep learning curve of Latex. I started to wonder if there are any ways to reduce the learning curve of Latex and also reduce the verbosity of Latex. I stumbled across **Markdown**, a small markup language that compiled into HTML. The syntax of Markdown was very intuitive and short. So I decided to implement a lot of ideas from Markdown into Latex and that's how Vanilla was born.

Technical Details

Vanilla first started out as a separate markup language with backward compatiblity to Latex. But I dropped that idea and made it as an extension of Latex so that people can keep using their editors. Vanilla compiler prototype was written in Python. Then the code was translated into Ruby and was released as a command line utility.

Getting Vanilla

It is very easy to get Vanilla. Visit www.adhithyan15.github.com/vanilla/ and follow the steps below.

1. Click on "Download Zip" to download both the source of the compiler and also the compiled version of the compiler.

   [scale=0.5]VanillaGithub

2. Extract the downloaded zip file using Winzip or any other archive extractor. If you don't have an archive extractor, I recommend *[7-zip =¿ www.7zip.org]

3. Once you extract it, you will find the following files inside the folder you extracted

Features of Vanilla

@(vfl) offers a lot of features to enhance people's productivity with LaTex. Let us see what those features below

- **Multiline Comments**: In @(vfl) you can comment out multiple lines of code using matlab style comments. Let us see a quick demo of this feature

  non tempus mi ultrices in. Praesent lobortis erat et lorem commodo mollis. Pellentesque eu euismod nulla.Ut vitae consequat est. Quisque adipiscing rhoncus tellus, eu fermentum augue tincidunt ut. Proin a dui dignissim massa auctor iaculis dictum a purus. Suspendisse porta felis at velit placerat varius.

  Vanilla compiler converts @(vfl) multiline comments into several single line LaTex comments. Vanilla compiler doesn't touch the LaTex comments. So they are still valid.

- **Constants**: Constants are very similar to variables but they are immutable. In @(vfl), you can declare constants through the following sytax @($constant\_name$) = $value$. Note: LaTex has offers mutable variables. We are working on implementing mutable variables in Vanilla. It will be released in version 2.0.

  @(vfl) = **Vanilla Flavored LaTex**

  Welcome to @(vfl)

  Vanilla compiler doesn't compile the Vanilla constants into LaTex variables. Instead, it converts them into plain text so that they can be used in a variety of purposes. Since the constants are rendered into plain text all the you can pass in a variable to most of the features listed below.

- **Formatted Text Blocks**: (cftb)[] were inspired by Less CSS' parametric mixins. (cftb)[] can include any kind of LaTex or @(vfl) formatting. @(vfl) forces the @(ftb) to be declared inside preamble to increase readability and @(ftb) are immutable. Mutable @(ftb) are under development and they will be released in version 2.0.

  (welcome$_m essage$)[@($name$), @($message$)] = [$Welcome$**@($name$)**, @($message$)]

(welcome$_m essage$)[$AdhithyaRajasekaran, HelloWorld$]

(cftb)[] can span multiple lines. Parameterless @(ftb) can be used as a constants spanning multiple lines.

(disclaimer$_s tatement$)[] = [**$WARNING$** : $Computervirusescanbetransmittedviaemail.Therecipients$
$mailtransmissioncannotbeguaranteedtobesecureorerror-freeasinformationcouldbeintercepted, corrupt$
$mailtransmission.$]

(disclaimer$_s tatement$)[]

- **Formulas**: Formulas were inspired by *[Homebrew's =¿ http://www.mxcl.github.com/homebrew/] formulas. @(vfl) is shipped with very few formulas and the number is slowly increasing. The formulas that are shipped with @(vfl) are written for tasks that are difficult to do with LaTex. All the formulas will accept accept a variable as a parameter. Let us take a look into the formulas that ship with @(vfl)

    1. **Matrix Formula**: Matrices are very hard to construct in LaTex even with **AMSMATH** package. So @(vfl) ships with a -for the easy creation of matrices. This formula will create a matrix with square brackets around it.Let us do an example
       I want to create a matrix

       A = $(matrix)[123; 456; 789]$

A = 123

456

789

A = $(matrix)[123; 456; 789]$

@(vfl) compiled the -

2. **Determinant Formula**: Determinants are very similar to matrices except they use a straight line to enclose the contents of the matrix. @(vfl) ships with the -
   I want to create a determinant

   B = $(det)[123; 456; 789]$

B = 123

456

789

B = $(det)[123; 456; 789]$

@(vfl) compiled the -

3. **Capitalize Formula**: Capitalize is a fun formula that capitalizes the first letter of every word in a string that is passed to it. Let us see a demo of this feature
   $(capitalize)[thisisawonderfulday]willyieldThisIsAWonderfulDay$

   As you can see that the number of formulas are very limited. Version 2.0 will allow users to create their own formulas using Ruby.

- **Inline Calculations**: Inline calculations allow you to perform simple yet powerful calculations with in the document. You can have to nest your calculations inside #[] to perform the calculations. Let us see some examples

    1. **-