Vanilla Documentation

Everything that you need to know about Vanilla

Copyright

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 the 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 Flavored LaTex** documents. This document itself is written in **Vanilla Flavored LaTex**

Why Vanilla?

LaTex is a wonderful typesetting system but its reach is limited because of its complexity, verbose syntax and steep learning curve. So **Vanilla Flavored LaTex** is directed towards solving those above mentioned problems. It derives most of its ideas from **Markdown**, a small but powerful markup language.

Open Source Project

The authors of this project have decided to open source the source code of the project. The code is hosted on github. The code is released under MIT License. Please report any bugs to the issues tab on our github homepage. If you have any new ideas or suggestions, please leave them on the issues tab. We will greatly appreciate your feedback.

Technical Details

Vanilla first started out as a separate markup language with a backward compatiblity to Latex. But it became apparent that there will be not enough interest in creating large amounts of libraries and productivity tools for Vanilla alone. So the authors departed from the idea and made it a preprocessor that incorporated a lot of features from their original markup language idea.

Any LaTex document is a valid **Vanilla Flavored LaTex** document but no **Vanilla Flavored LaTex** document is a valid LaTex document without being processed through the Vanilla.

The original compiler prototype was built using Matlab and then Python for more reachability. The preprocessor grew out of that compiler and was rewritten from scratch in Ruby and is shipped as a commandline utility.

Vanilla builds on top of **PDFLatex**. So it is very important that you have any of the major tex distributions installed on your computer. If you are on windows I recommend MikTex and on mac, I recommend MacTex.

Current State

The current and stable vanilla release is version 1.0. The release provides a minimal set of base features which might be retained in most of the future releases. Significant amounts of features have been left out either due to bugs or incomplete implementation or not enough testing. They will be released in the

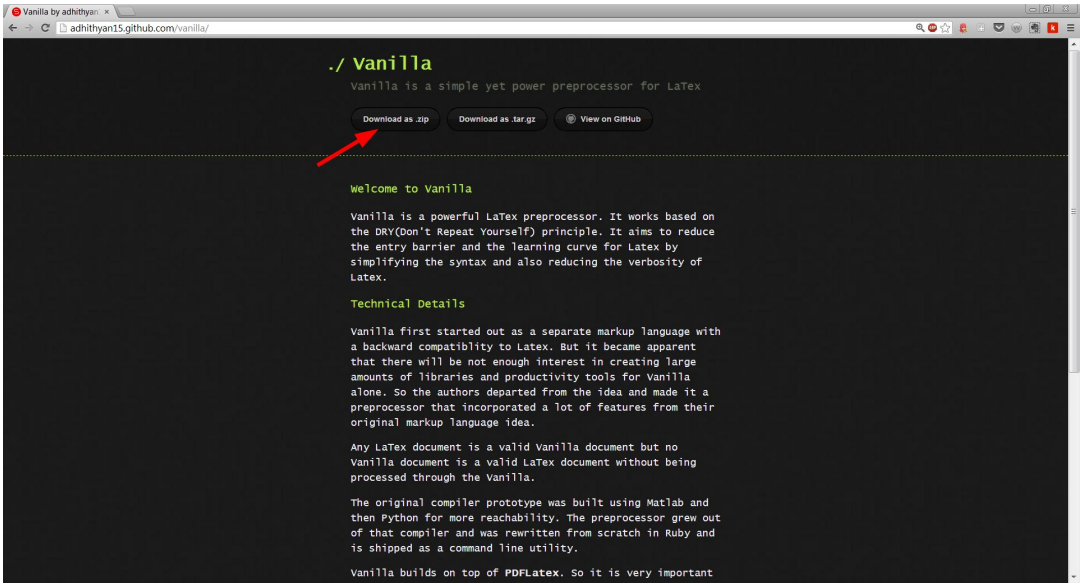
upcoming releases. One of the main drawbacks of Vanilla is that it doesn't ship with an **error catcher and notifier**. So Vanilla will simply ignore the errors and try to compile the error free parts of the document.

The other major disadvantage is that the preprocessor is not available in **Mac or other Unix based platforms** as a bash utility. A full fledged bash/terminal utility will be available in version 2.0.

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.



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

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

   - **Vanilla.exe**: This is the primary compiler that will help you compile **Vanilla Flavored LaTex** to **Pure LaTex**. This might be of no use to you if you are running linux or mac.
   - **Vanilla.rb**: This file contains the source code of the compiler. If you are in mac or linux, you can use this file to compile your **Vanilla Flavored LaTex** to LaTex documents.
   - **Readme.md**: Since the project is hosted in Github, Github added a readme file markuped in **Markdown**.
   - **Documentation.pdf**: This is the same document that you are reading right now.
   - **Documentation.tex**: This is the .tex version of documentation written using **Vanilla Flavored LaTex**. I highly recommend you to read through this document so that you can get example usages of **Vanilla Flavored LaTex**.

4. For windows users, please follow this tutorial to add the Vanilla.exe to your path so that it can be accessed from your commandline prompt. Mac and other Unix based OS users can use the Vanilla.rb to process your **Vanilla Flavored LaTex** documents because a bash/terminal utlity is not yet ready.

5. You are all set. You can compile your **Vanilla Flavored LaTex** documents by calling **"vanilla -c somefile.tex"**. You have to set the directory in which your **Vanilla Flavored LaTex** document is located as your **current directory** for the above command to work.

   Please see this tutorial to understand what the above phrase is describing.

Vanilla compiler can work with LaTex Editors. But configuration seems to be a problem. Vanilla is an unconventional preprocessor in which it doesn't have a seperate extension. After the preprocessing stage, Vanilla needs the authority to overwrite the original file and many of the LaTex editors block that kind of action from happening. So a solution is under development. Until such time, you can use Notepad, Notepad++, Wordpad or Sublime Text.

Features of Vanilla

Vanilla Flavored LaTex offers a lot of features to enhance people's productivity with LaTex. Let us see what those features below

- **Multiline Comments**: In **Vanilla Flavored LaTex** you can comment out multiple lines of code using matlab style comments. Let us see a quick demo of this feature

```
%You can comment out multiple lines using matlab style comments %{.....}%

%{Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris placerat tristique purus,
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 **Vanilla Flavored LaTex** 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 **Vanilla Flavored LaTex**, 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** %must be declared in the preamble

%Usage

Welcome to @(vfl) %should output Welcome to \textbf{Vanilla Flavored LaTex}
```

  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**: Formatted Text Blocks were inspired by Less CSS' parametric mixins. Formatted Text Blocks can include any kind of LaTex or **Vanilla Flavored LaTex** formatting. **Vanilla Flavored LaTex** forces the formatted text blocks to be declared inside preamble to increase readability and formatted text blocks are immutable. Mutable formatted text blocks are under development and they will be released in version 2.0.

```
%Declaration

#(welcome_message)[@(name),@(message)] = [Welcome **@(name)**,@(message)] %should be
%declared in the preamble

%Usage
```

```
#(welcome_message)[Adhithya Rajasekaran,Hello World] %This should output
%Welcome \textbf{Adhithya Rajasekaran}, Hello World
```

Formatted Text Blocks can span multiple lines. Parameterless formatted text blocks can be used as a constants spanning multiple lines.

```
%Declaration

#(disclaimer_statement)[] = [**WARNING**: Computer viruses can be transmitted via email.
The recipient should check this email and any attachments for the presence of viruses.
The company accepts no liability for any damage caused by any virus transmitted by this email.
E-mail transmission cannot be guaranteed to be secure or error-free as information could be
intercepted, corrupted, lost, destroyed, arrive late or incomplete, or contain viruses.
The sender therefore does not accept liability for any errors or omissions in the
contents of this message, which arise as a result of e-mail transmission.]
%must be declared in the preamble

%Usage

#(disclaimer_statement)[] %anywhere in the body
```

- **Formulas**: Formulas were inspired by Homebrew's formulas. **Vanilla Flavored LaTex** is shipped with very few formulas and the number is slowly increasing. The formulas that are shipped with **Vanilla Flavored LaTex** 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 **Vanilla Flavored LaTex**

  1. **Matrix Formula**: Matrices are very hard to construct in LaTex even with **AMSMATH** package. So **Vanilla Flavored LaTex** ships with a $(matrix)[] formula that allows 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 = $$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

     ```
     %using AMSMATH package

     A = $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

     %using Vanilla formula $(matrix)[]

     A = $(matrix)[1 2 3;4 5 6;7 8 9]
     ```

     Vanilla Flavored LaTex compiled the $(matrix)[] formula into AMSMATH's **bmatrix** environment.

  2. **Determinant Formula**: Determinants are very similar to matrices except they use a straight line to enclose the contents of the matrix. **Vanilla Flavored LaTex** ships with the $(det)[] formula that takes care of determinants. Let us see a quick demo of this feature

     I want to create a determinant

$$B = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$$

```
%using AMSMATH package

B = $\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$

%using Vanilla formula $(matrix)[]

B = $(det)[1 2 3;4 5 6;7 8 9]
```

Vanilla Flavored LaTex compiled the $(det)[] formula into AMSMATH's **vmatrix** environment.

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
   $(cap)[this is a wonderful day] will yield This Is A Wonderful Day

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 an example

  ![5*(45*(60/2))+3] compiles into **6753**

- **Inline Formatting**: Formatting is made easy by **Vanilla Flavored LaTex**. There are several formatting options provided by **Vanilla Flavored LaTex**. We will look at each one of them individually

  1. **Boldface**: You can boldface any text using **some text** and it will output **some text**.
  2. **Italicize**: You can italicize any text using ///some text/// and it will output *some text*.
  3. **Underline**: You can underline any text using - - -some text- - - and it will output some text.

- **Tex Packs**: Tex Packs allow users to abstract away their most used LaTex packages into a file with any name and an extension .texpack. You can then import these Tex Packs into your **Vanilla Flavored LaTex** documents by using $(importpack)[package location] in your preamble. Tex Packs are designed to encourage reusability and seperating the structure of the document from the content of the document. The LaTex packages that this document uses have been defined in a texpack called documentation.texpack. Please take a look into that to get an idea of how to write a .texpack.

  There is no special syntax for defining a texpack. You can copy and paste all the usepackage statements into a texpack and import it using the importpack function.

  Vanilla Flavored LaTex allows beginners start typesetting documents without worrying about packages. If a document doesn't have any importpack function calls or any usepackage statements, then the preprocessor will automatically add some of the most used packages to the document. The following are the packages provided out of the box by **Vanilla Flavored LaTex**.

  1. Amsmath,Amssymb,Amsthm
  2. Geometry with A4paper and 1.0in margin
  3. Hyperref
  4. Xcolor
  5. Verbatim

6. Booktabs
7. Multicol
8. Graphicx
9. Siunitx
10. Cleveref

- **URLs and Hyperlinks**: **Vanilla Flavored LaTex** provides support for two kinds of URLs out of the box. **Vanilla Flavored LaTex** calls them bare urls and descriptive urls. Let us take a look into both of them.

    1. **Bare URLs**: With **Vanilla Flavored LaTex**, you can just type in your URL in the document and the preprocessor will automatically find it and convert it into a LaTex hyperlink.Let us see an example

       http://github.com/adhithyan15 will become `http://github.com/adhithyan15`
    2. **Descriptive URLs**: Descriptive URLs allow a word or a phrase to be hyperlinked to an URL.**Vanilla Flavored LaTex** comes with a simple syntax to write descriptive URLs easily. The syntax is ***[word or phrase =>URL]**.Let us see an example

       *[blog =>http://adhithyaraja.wordpress.com] will become blog.

- **Double Quotes**: In **Vanilla Flavored LaTex**, you can use " " to surround some text with double quotes. For example, "hello" will produce "hello".

- **Undocumented Features**: There are couple of features which were included in the compiler just for the sake of writing this documentation. They are the fenced code block and inline code block. It is not possible to demonstrate their features here because they determine whether a piece of code gets compiled by the preprocessor. You can look inside the source of this document to know how they are used.