**What to cover/research and create outline on topics**

* What it is used for and what the purpose is and history/timeline-Nikhil
* Core principles and examples of them
* benefits/negatives - Thomas
* how to code it/ Demo
* How it compares to similar languages-Zhichen

***What it is used for and purpose and history/timeline-Nikhil***

HISTORY

* Created by Hampton Catlin in 2006
* Natalie Weizenbaum was the main developer of Ruby’s implementation of HAML until 2012 and made HAML usable by doing a variety of things such as fixing bugs
* Version 2.2.0 : July 2009
* Version 3.0.0 :May 2010

PURPOSE

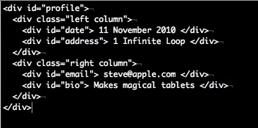
HAML stands for HTML abstraction markup language.

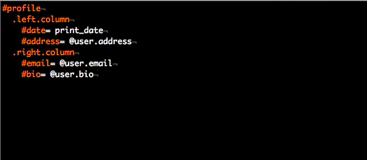
HAML purpose is to a way to cleanly and simply describe the HTML without inline code

It is used as a replacement for other inline systems such as ASP and ERB

HAML is a description of HTML so it avoids the need to code the HTML into the template

Before and after:





Simplified and cleaner version

The code has

-meaning

-does not repeat itself

-concise

-markup much quicker and does not have to be in HTML format

-easier to read

ERB HAML



Principles

* Markup should be beautiful
* Markup should be dry

<https://haml.info/>

<https://github.com/haml/haml>

[https://haml.info/docs/yardoc/file.REFERENCE.html#using\_ham](https://haml.info/docs/yardoc/file.REFERENCE.html#using_haml)

<https://www.sitepoint.com/an-introduction-to-haml/>

<https://coderwall.com/p/aivizg/introduction-to-haml--2>

Graded on:  
Presentation Organization

Presentation Content

Presentation Visuals

Presentation Delivery

Handouts/Materials

Presentation outline/structure:

HAML SUMMARY

Haml is a markup language that’s used to cleanly and simply describe the HTML of any web document without the use of inline code. Haml functions as a replacement for inline page templating systems such as PHP, ASP, and ERB, the templating language used in most Ruby on Rails applications. However, Haml avoids the need for explicitly coding HTML into the template, because it itself is a description of the HTML, with some code to generate dynamic content.

HAML CORE PRINCIPLES

## **Core Principles**

Haml’s development as a language is based on adherence to several core principles. These are:

1. **Markup Should be Beautiful**Markup should not be used **merely** as a tool to get browsers to render a page how the author wants it rendered. The rendering isn’t the only thing people have to see; they have to see, modify, and understand the markup as well. Thus, the markup should be just as user-friendly and pleasant as the rendered result.
2. **Markup Should be DRY**HTML involves major repetition. Every element is named twice: once before its content and once after. ERB adds even more repetition and unnecessary characters. HAML avoids all of this by relying on indentation, not text, to determine where elements and blocks of code begin and end. Not only does this result in smaller templates, it makes the code much cleaner to look at.
3. **Markup Should be Well-Indented**One of the major problems with ERB is that not only does it not encourage well-indented code, it actively makes it challenging, or even impossible, to write. This leads to confusing, unreadable HTML, when it should be well indented and reflecting the underlying structure of the document. Haml automatically properly formats all tags to that it does.
4. **HTML Structure Should be Clear**XML and HTML are formats built upon the idea of a structured document. That structure is reflected in their markup, and it should likewise be reflected in meta-markup such as Haml. Because Haml’s logic is based on indentation of child elements, this structure is naturally preserved, making the document much easier and more logical for mere humans to read.

HOW to get haml?  
  
For us, just run the command

gem install haml

In the terminal.

LIST of editors that have haml support:

Various text editors have different levels of support for Haml. Here’s a list of all known packages — if you create or learn of a more updated version, please email haml@googlegroups.com.

Haml’s syntax is updated from time to time, meaning some of these modes might not be as current as possible. If anyone wants to take on the task of updating an old mode, that would be greatly appreciated.

* [**Coda/SubEthaEdit**](http://github.com/gf3/haml.mode) by [Gianni Chiappetta](http://www.gf3.ca/).
* [**Eclipse**](http://haml.googlegroups.com/web/net.lucky_dip.hamleditor_site_0.5.4.zip) by [Brad Wilson](http://lucky-dip.net/). This is a snapshot of the site that used to host the plugin, so it may be a bit confusing to get it working. The [EclipseColorer plugin](http://colorer.sourceforge.net/eclipsecolorer/) by [Igor Russkih](http://cail.nm.ru/) also has support for coloring Haml and Sass.
* [**Emacs**](http://github.com/nex3/haml-mode/tree/master/haml-mode.el) by [Natalie Weizenbaum](http://nex-3.com/). Probably the most up-to-date mode, since it’s maintained by the former maintainer of Haml.
* [**GEdit/Gnome**](http://blog.ardekantur.com/2008/03/haml-syntax-highlighting-for-gedit/) by [Ardenkatur](http://ardekantur.com/). There’s also an [older package](http://groups.google.com/group/haml/msg/b5100d80d9182c71) by [Ivan Kerin](http://www.facebook.com/ivan.kn).
* [**JEdit**](http://blog.wolfman.com/articles/2006/12/01/a-haml-edit-mode-for-jedit) by [Jim Morris](http://blog.wolfman.com/).
* [**Kate/KDE**](http://haml.googlegroups.com/web/haml_highlighting_for_kde.tar.gz) by [Cies Breijs](http://ciesbreijs.blogspot.com/).
* [**Komodo**](http://community.activestate.com/files/haml_bundle-0.1.1_0.zip) (author unknown).
* [**NetBeans**](http://mediacast.sun.com/users/tnorbye/media/org-netbeans-modules-haml.nbm/details) (author unknown).
* [**RadRails**](http://www.aptana.com/products/radrails): built-in.
* [**RubyMine**](http://www.jetbrains.com/ruby): built-in.
* [**TextMate/E Text Editor**](http://github.com/handcrafted/handcrafted-haml-textmate-bundle) by [Adam Stacoviak](http://www.adamstacoviak.com/). There’s also an [official](http://svn.textmate.org/trunk/Bundles/Ruby%20Haml.tmbundle) but out-of-date bundle by [Kevin Ballard](http://kevin.sb.org/), and many [GitHub forks](http://github.com/douglasjarquin/ruby-haml-tmbundle/network) of that bundle.
* [**Vim**](http://github.com/tpope/vim-haml) by [Tim Pope](http://www.stopwritingramblingcommitmessages.com/). There’s also an [official](http://www.vim.org/scripts/script.php?script_id=1773) but out-of-date mode by [Dmitry Ilyashevich](http://github.com/dmitry-ilyashevich).
* [**Sublime Text 2**](https://github.com/phuibonhoa/handcrafted-haml-textmate-bundle) by [Adam Stacoviak](http://www.adamstacoviak.com/) and adapted by [Philippe Huibonhoa](http://github.com/phuibonhoa). It’s available via the [Sublime Package Control](http://wbond.net/sublime_packages/package_control) plugin as well.
* [**Atom**](https://github.com/HarlemSquirrel/language-haml) by [Kevin McCormack](https://github.com/HarlemSquirrel).
* [**Visual Studio Code**](https://github.com/vayan/vscode-haml-ruby) by [Kevin McCormack](https://github.com/HarlemSquirrel) and adapted by [Yann Vaillant](https://github.com/vayan).

To run Haml from the command line, just use

haml input.haml output.html

For help, do haml --help

File extension: xxx.html.haml

RUBY HAML extension in vscode is helpful

**What to cover/research**

* Principles and history/timeline-Nikhil
* What it is used for and what the purpose is
* benefits/negatives - Thomas
* how to code it/ Demo
* How it compares to similar languages-Zhichen

VERY GOOD LINK

https://haml-coffee-online.herokuapp.com/

What to present/required materials

* A clear statement of learning objectives (ie a mini-syllabus for your presentation). This statement does *not* need to be part of your presentation, but *must* be part of your submission materials.
* Audio-visual support (eg slides, videos, …).
* Handouts (eg example programs, cheatsheets,…).
* Interactive elements (eg "Your turn" problems, demos,…).
* Questions to assess student understanding that could be used on a final exam. You should include several different problems that cover different levels of Bloom's taxonomy.

ADVANTAGES

* easy to read and visually expresses your DOM hierarchy
* code is shorter as closing tags are eliminated
* speeds up development time
* no closing tags, simpler syntax, structure indentation
* proper indentation forces user to write cleaner html
* easier to debug
* quick to learn
* easy to install

DISADVANTAGES

The main disadvantage of using HAML over HTML is simply that HTML is just about universal among web developers, whereas the HAML community is still a relatively small community. This would undoubtedly make finding developers to work on your project in the future a more daunting task.

<https://blog.knoldus.com/why-haml-is-future-of-web-and-html-all-about-haml/>

<https://uharston.medium.com/5-reasons-you-should-be-using-haml-instead-of-erb-4d93766bd6b3>

<https://www.reddit.com/r/rails/comments/6t6sti/benefits_of_haml/>