Tutorial

Before we start this tutorial, I want to make one thing clear. After you look at this, go and convert one of your ERB files to Haml. Just try it. Just take the file and start hitting delete. You don't have to keep the file if you don't like it, but after you're done with this tutorial, just try one file.

Haml feels odd for the first 20 minutes, but then after that **you will be faster**.

Getting Started

First, get Haml and install the gem (this tutorial assumes you're using Rails — it should apply to other frameworks and standalone Haml just as well, though). Haml is a drop-in replacement for ERB. That means any file in your `app/views` folder can be switched over to Haml by simply changing the extension of the file.

```
app/views/account/login.html.erb → app/views/account/login.html.haml
```

Now, when you view that page, instead of ERB getting its hands on the template, it's handled by Haml instead. You can mix up ERB and Haml and on the fly throughout your site.

How to Convert

Let's start off with some basic ERB that we want to convert.

ERB

```
<strong><%= item.title %></strong>
```

Haml

```
%strong= item.title
```

In Haml, we write a tag by using the percent sign and then the name of the tag. This works for `%strong`, `%div`, `%body`, `%html`, any tag you want. Then, after the name of the tag is `=`, which tells Haml to evaluate Ruby code to the right and then print out the return value as the contents of the tag. Unlike the ERB above, Haml will automatically detect newlines in the return value and format the tag properly.

Adding Attributes

Simple tags are all well and good, but what about adding attributes to tags?

HTML

```
<strong class="code" id="message">Hello, World!</strong>
```

Haml

```
%strong{:class => "code", :id => "message"} Hello, World!
```

The attributes are just a standard Ruby hash. The `class` attribute is "code", the `id` attribute is "message". Notice that in this example, we didn't use `=`, so "Hello, World!" is interpreted as a normal string, not Ruby code.

There is an easier way to define this tag in Haml, since `class` and `id` are such common attributes and since most designers (and developers) are familiar with CSS, we can use similar notation to describe this tag.

Haml

```
%strong.code#message Hello, World!
```

Not only that, but since `div` tags are so common, you can leave off the tag definition and have it default to `%div`.

Haml

```
.content Hello, World!
```

HTML

```
<div class='content'>Hello, World!</div>
```

Upping the Complexity

Now what about something a little more complicated?

ERB

```
<div class='item' id='item<%= item.id %>'>
  <%= item.body %>
</div>
```

Pretty basic. This might be part of a partial or something. Let's convert it to Haml.

Haml

```
.item{:id => "item#{item.id}"}= item.body
```

This stuff is fun! Now, nesting is taken care of in Haml via whitespace.

ERB

```
<div id='content'>
  <div class='left column'>
    <h2>Welcome to our site!</h2>
    <p><%= print_information %></p>
  </div>
  <div class="right column">
    <%= render :partial => "sidebar" %>
  </div>
</div>
```

Haml

```
#content
  .left.column
    %h2 Welcome to our site!
    %p= print_information
  .right.column
    = render :partial => "sidebar"
```

Look at that. Doesn't that just make you smile?

There is a lot more to learn, so I'd highly recommend checking out the reference. It's filled with awesome little tricks that we've added to Haml to make building sites even more fun.