Introducing Jade

Looking inside the views folder for the example project, you see a number of files with the .jade extension. Express makes use of template engines to compile views to HTML. By default, Express uses Jade as the template engine.

Template Engines Generate HTML

Template engines are used in most web frameworks to generate HTML and are typically used within a views folder. They allow developers to output data from an application to HTML.
Typical features include variables and loops. Template engines can also be known as template processors or filters.

Two popular template engines are Smarty (PHP) and ERB (Ruby).

HTML and Jade Comparison

```
<div class="wrapper">
<h1>My holiday in Prague</h1>
<p>I had a great holiday in Prague where I met some great people.</p>
<img src="images/photo.jpg" alt="Me on holiday!" />
</div>
```

Jade

```
.wrapper
h1 My holiday in Prague
p I had a great holiday in Prague where I met some great people
img(src='images/photo.jpg', alt='Me on holiday')
```

Notice several things:

- Jade is much more terse than HTML.
- Jade uses indentation to declare the hierarchy of an HTML document.
- You do not have to use tags in Jade. The <> characters are automatically added when the template is compiled.
- You do not have to close HTML tags in Jade. When Jade generates the HTML, it will close the tags for you.

When the page is generated, Jade compiles the template to HTML. the output is exactly the same as vanilla HTML. So, why bother with a template engine? The answer is that

it allows applications to output data dynamically to HTML. Examples of where you may use a template engine include

- To display a list of blog posts stored in a database
- To create a single template for displaying many different blog posts
- To change the `<title></title>` element of a page based on a variable
- To create header and footer includes that can be reused across templates

Jade Takes Its Inspiration from Haml

Jade takes much of its inspiration from Haml (HTML Abstraction Markup Layer). Haml is a popular template engine used in the Ruby Community.

Defining Page Structure with Jade

Jade defines page structure by indentation. If a line is indented under another one, it is assumed to be a child of the previous line. If you are new to template languages, this can take a little getting used to. This should become clear with examples, however.

```
Html
```

compiles to

```
<html></html>
```

You can use any HTML tag here (`body` , `section` , `p` , and so on).

To add an id to a tag, append a `#` symbol and then the name of your id. Note that spaces are not allowed.

```
section#wrapper
```

compiles to

```
<section id="wrapper"></section>
```

You can add a class by appending a dot followed by the name of your class.

```
p.highlight
```

compiles to

```
<p class="highlight"></p>
```

If you need a class and an id, Jade supports chaining.

```
section#wrapper.class-name
```

compiles to

```
<section id="wrapper" class="class-name"></section>
```

Jade also supports more than one class on a tag.

```
p.first.section.third.fourth
```

compiles to

```
<p class="first second third fourth">
```

To create HTML structure, indentation is used:

```
p
span
```

compiles to

```
<p><span></span></p>
```

To add text within a tag, simply add it after the tag declaration.

```
h1 Very important heading
```
compiles to
```
<h1>Very important heading<h1>
```

Jade also supports large bodies of text by using the pipe delineator:
```
p
| Text can be over
| many lines
| after a pipe symbol
```

compiles to

```
<p>Text can be over many lines after a pipe symbol<p>
```

Follow these steps to create a page structure in Jade:

1. Open your terminal and generate a skeleton Express site by running the following command:
express jade_structure
2. Enter the directory that you created and install the required dependencies:
cd jade_structure && npm install
3. Start the application, change into the jade_structure folder, and then run
node app.js
4. Open your web browser of choice and browse to http://127.0.0.1:3000. You see a basic website served from Express.
5. Edit the file index.jade file found at views/index.jade. You see a bare bones structure for the page:
h1= title
p Welcome to #{title}
6. Add the following code to the page:
section#wrapper
h2 Basic Structure
section
p
span This is a span within a p within a div!
7. Reload the page and check the HTML on the page by viewing the source. Although the HTML will be compressed, you should see the following HTML:
```
<section id="wrapper">
<section>
<p>
<span>This is a span within a p within a div!</span>
</p>
</section>
</section>
```

Outputting Data with Jade

While building page structure with Jade is good, the real power of a template language comes from manipulating data and outputting it to HTML.

Jade uses two special characters to decide how it should interpret code. The first, the minus sign (-), is used to tell Jade that the code that follows should be executed.

The second is the equals sign (=). This tells the interpreter that the code should be evaluated, escaped, and then outputted.

This can be a little confusing at first, but examples will make it clear.

Variables In this example, the code is executed and no output is to be returned. Jade just sets the variable
of `foo` to be `bar`:

```
- var foo = bar;
```

With the variable set, it can be used later on:

```
p I want to learn to use variables. Foo is #{foo}!
```

The special syntax of `#{ variable }` tells Jade to replace the variable with the string value. This compiles to

```
<p>I want to learn how to use variables. Foo is bar!<p>
```

Follow these steps to use variables with Jade:

1. Open your terminal and generate a skeleton Express site by running the following command:
express jade_variables
2. Enter the directory that you created and install the required dependencies:
cd jade_variables && npm install
3. Start the application, change into the jade_variables folder, and then run
node app.js
4. Open your web browser of choice and browse to http://127.0.0.1:3000. You see a basic website served from Express.
5. Edit the file index.jade file found at views/index.jade and add the following code:
- var name = "Your Name"
h1 Hello #{name}!
6. Reload the page in your browser. Check the HTML on the page by viewing the source. You should see the following HTML:
<h1>Hello Your Name</h1>

Loops

Loops allow you to iterate over arrays and objects. If you are creating anything other than a basic brochure site, you will find yourself using loops a lot.

This is commonly known as iteration,meaning that you iterate over an array or object and do the same thing over and over again.Perhaps because this is such a common pattern, Jade makes using the minus sign optional. Admittedly, it is confusing that you use a minus sign to specify a variable but not a loop.

In these examples, the minus sign is included before each loop for clarity, although it is optional.

```
- users = ['Sally', 'Joseph ', 'Michael', 'Sanjay']
- each user in users
p= user
```

compiles to

```
<p>Sally</p>
<p>Joseph</p>
<p>Michael</p>
<p>Sanjay</p>
```

If you prefer to use the `for` keyword, this can also be written as

```
- for user in users
p= user
```

It is also possible to iterate over objects.

```
- obj = { first_name: 'George', surname: 'Ornbo'}
- each val, key in obj
li #{key}: #{val}
```

compiles to

```
<li>first_name: George</li>
<li>surname: Ornbo</li>
```

Follow these steps to use loops with Jade:

1. Open your terminal and generate a skeleton Express site by running the following command:

```
express jade_loops
```

2. Enter the directory that you created and install the required dependencies:

```
cd jade_loops && npm install
```

3. Start the application, change into the jade_loops folder, and then run

```
node app.js
```

4. Open your web browser of choice and browse to http://127.0.0.1:3000. You see a basic website served from Express.

5. Edit the file index.jade file found at views/index.jade and add the following code:

```
- beatles = ['John', 'Paul', 'Ringo', 'George']
ul
each beatle in beatles
li #{beatle}
```

6. Reload the page in your browser. Check the HTML on the page by viewing the source. You should see the following HTML:

```
<ul>
<li>John</li>

<li>Paul</li>
```

```
<li>Ringo</li>
<li>George</li>
</ul>
```

Conditions

If you have had exposure to any programming language, you will be familiar with conditions. In plain English, we can describe conditional flow as “If something is true, do something; otherwise, do something else.” In Jade, conditions look like this:

```
- awake = false
- if (awake)
p You are awake! Make coffee!
- else
p You are sleeping
```

Note again that the minus sign (-) in front of the `if` and `else` keywords is optional.

Follow these steps to use conditions in Jade:

1. Open your terminal and generate a skeleton Express site by running the following command:
```
express jade_conditions
```
2. Enter the directory that you created and install the required dependencies:
```
cd jade_conditions && npm install
```
3. Start the application, change into the jade_conditions folder, and then run
```
node app.js
```
4. Open your web browser of choice and browse to http://127.0.0.1:3000. You see a basic website served from Express.
5. Edit the file index.jade file found at views/index.jade and add the following code:
```
- raining = true
- if (raining)
p It is raining. Take an umbrella!
- else
p No rain. Take the bike!
```

Inline JavaScript

It is possible to execute inline JavaScript in Jade templates. To do this, declare a script block and then add your JavaScript within it:
```
script
alert('You can execute inline JavaScript through Jade')
```

Follow these steps to execute inline JavaScript in Jade:
1. Open your terminal and generate a skeleton Express site by running the following command:
```
express jade_inline_javascript
```
2. Enter the directory that you created and install the required dependencies:
```
cd jade_inline_javascript && npm install
```
3. Start the application, change into the jade_inline_javascript folder, and then run
```
node app.js
```
4. Open your web browser of choice and browse to http://127.0.0.1:3000. You see a basic website served from Express.
5. Edit the file index.jade file found at views/index.jade and add the following code:

```
script
alert('Inline JavaScript in Jade')
```

6. Reload the page in your browser. You should see a JavaScript alert.

Includes

Most websites have parts of the page that appear on every page of the site. Example of these include

- Headers
- Footers
- Sidebars

Includes Make It Easier to Maintain a Website

Includes move common parts of a website into single files. This means that when a client asks for an extra item to be added to a header, you only have to change a single file.
Jade supports includes with the `include` keyword and then the name of the template that you want to include.

```
html
body
include includes/header
```

This includes code from the views/includes/header.jade file.

Follow these steps to use includes in Jade:

1. Open your terminal and generate a skeleton Express site by running the following command:
```
express jade_includes
```
2. Enter the directory that you created and install the required dependencies:
```
cd jade_includes && npm install
```
3. Start the application, change into the jade_includes folder, and then run
```
node app.js
```
4. Open your web browser of choice and browse to http://127.0.0.1:3000. You see a basic website served from Express.

5. Create a new folder at views/includes.
6. Within views/includes, add a file called footer.jade.
7. Add some content into the footer.jade file:
```
p This is my footer. Get me. I have a footer on my website.
```
8. Include your footer file in views/index.jade:
```
h1 Jade Includes Example
include includes/footer
```
9. Reload the page in your browser. You should see your footer included on the page.

Mixins

Mixins are a feature of Jade that not many other template engines have. If you find yourself repeating the same blocks of code over and over, using mixins is a good way to keep your code maintainable and clean. Think of mixins as includes for your code. An example of where you may want to use a mixin is outputting data in a loop. To define a mixin, use the `mixin` keyword:

```
mixin users(users)
ul
each user in users
li= user
```

When the mixin is defined, you can use and reuse it in your templates:

```
- users = ['Krist', 'Kurt', 'Dave']
mixin users(users)
```

Follow these steps to use mixins in Jade:

1. Open your terminal and generate a skeleton Express site by running the following command:
```
express_jade_mixins
```
2. Start the application, change into the jade_mixins folder, and then run
```
node app.js
```
3. Open your web browser of choice and browse to http://127.0.0.1:3000. You see a basic website served from Express.

4. E dit the file index.jade file found at views/index.jade and add the following code:
```
mixin users(users)
ul
each user in users
li= user
- users = ['Tanya', 'Jose', 'Kim']
mixin users(users)
- more_users = ['Mark', 'Elena', 'Dave', 'Pete', 'Keiron']
mixin users(more_users)
```
5. Reload the page in your browser. You should see two lists of users generated from a single mixin.