HARVEY NASH

The Power of Talent

INTRODUCTION TO GULP, PUG, SASS

Security Classification: **Internal**

AGENDA

- Gulp
- Pug
- Sass
- Assignment
- Q&A

Objectives

- Understand and use Frontend build tools
- Can work with template engine
- Can work with CSS preprocessor
- Familiar with Frontend workflow

NASH TECH
The Power to Invent

GULP

What is Gulp?

- Gulp – the streaming build system
  - A Node.js package that enables you to easily write tasks in JavaScript which help you to automate stuff you do often.
  - It can automate common repetitive tasks in your workflow and make you more productive, e.g. it can minify your CSS, compile Sass, lint JavaScript files, etc.
  - Code over configuration.
  - Using the power of node streams.

Installation

- Install Gulp globally

```
$ npm install --global gulp-cli
```

- Initialize your project directory

```
$ npm init
```

- Install gulp in your project devDependencies

```
$ npm install --save-dev gulp
```

Usage

- Create a gulpfile.js at the root of your project

```
var gulp = require(‘gulp’);

gulp.task(‘default’, function() {
  // place code for your default task here
});
```

- Run gulp

```
$ gulp
```

Gulp API

- gulp.task – Define a task
- gulp.src – Read files in
- gulp.dest – Write files out
- gulp.watch – Watch files for changes

```
gulp.task(‘styles’, function() {
  return gulp.src([‘app/styles/*.scss’])
    .pipe(sass())
    .pipe(autoprefixer(‘last 2 versions’))
    .pipe(gulp.dest('static/css'))
});
```

```
gulp.task(‘watch’, function() {
  gulp.watch(‘app/styles/**/*.scss’, [‘styles’]);
});
```

Gulp Plugins

- gulp-pug
- gulp-pug-lint
- gulp-sass
- gulp-scss-lint
- gulp-autoprefixer
- gulp-concat
- gulp-jshint
- gulp-uglify
- gulp-imagemin

Other Build Tools

- Grunt
  - The JavaScript Task Runner
  - http://gruntjs.com

PUG

What is Pug?

- Pug – template engine
  - A clean, whitespace-sensitive template language for writing HTML
  - High performance template engine heavily influenced by Haml
  - Implemented with JavaScript for node and browsers

Installation

- Install Pug globally

```
$ npm install --global pug-cli
```

Usage

- Compile

```
$ pug filename.pug
$ pug -P filename.pug
$ pug --watch -P filename.pug
```

Example

```
doctype html
html(lang=“en”)
  head
    title Pug
  body
    h1 Welcome to Pug!
    p A simple Pug example.
```

```
<!DOCTYPE html>
<html lang=“en”>
  <head>
    <title>Pug</title>
  </head>
  <body>
    <h1>Welcome to Pug!</h1>
    <p>A simple Pug example.</p>
  </body>
</html
```

Syntax

- Tags: By default, text at the start of a line (or after only white space) represents an html tag. Indented tags are nested.

```
ul
  li Item A
  li Item B
  li Item C



p: a(href=“#”) Item
```

```
<ul>
  <li>Item A</li>
  <li>Item B</li>
  <li>Item C</li>
</ul>


<p><a href="#">Item</a></p>
```

Syntax

- Text: Prefix the line with a | character.

```
| Plain text

p Text inside <em>paragraph</em>

p
  | Plain text
  | multiple line

p.
  Plain text
  blocks of text
```

```
Plain text

<p>Text inside <em>paragraph</em></p>
<p>
  Plain text
  multiple line
</p>
<p>
  Plain text
  blocks of text
</p>
```

Syntax

- Attributes: Tag attributes look similar to html, however their values are just regular JavaScript.

a(class=“link”, href=“#”) More	<a class=“link” href=“#”>More</a>
a.link(href=“#”) More	<a class=“link” href=“#”>More</a>
#container	<div id=“container”></div>

Syntax

- Code: Write inline JavaScript code in your templates.

- for (var x = 0; x < 3; x++) p x= #{x}	<p>x= 0</p> <p>x= 1</p> <p>x= 2</p>
p= ‘This code is <escaped>!’	<p>This code is &lt;escaped&gt;!</p>
p!= ‘This code is <strong>not</strong> escaped!’	<p>This code is <strong>not</strong> escaped! </p>

Syntax

- Comments: Single line comments look the same as JavaScript comments and must be placed on their own line.

// Just some paragraphs	<!-- Just some paragraphs-->
p foo	<p>foo</p>
p bar	<p>bar</p>
//- will not output within markup	
p foo	<p>foo</p>
p bar	<p>bar</p>

Syntax

- Conditionals: if/else statement

- var lang = ‘Pug’
if lang == ‘Pug’
  p Awesome
else
  p Not awesome
	<p>Awesome</p>

- var lang = ‘Jade’ unless lang == ‘Pug’ p Not awesome	<p>Not awesome</p>

Syntax

- Case: switch statement

```
- var friends = 1
case friends
  when 0: p You have no friends
  when 1: p You have a friend
  default: p You have #{friends} friends
```

```
<p>You have a friend</p>
```

Syntax

- Iteration: each/for and while

```
ul
  each val in [1, 2, 3]
    li= val



- var n = 0
ul
  while n < 3
    li= n++
```

```
<ul>
  <li>1</li>
  <li>2</li>
  <li>3</li>
</ul>

<ul>
  <li>0</li>
  <li>1</li>
  <li>2</li>
</ul>
```

Syntax

- Interpolation: #{}, #[]

- var title = ’Pug’	<h2>Pug</h2>
h2= title	<p>Hello Pug</p>
p Hello #{title}	
p Click #[a(href=”#”) me]	<p>Click <a href="#">me</a></p>

Syntax

- Mixins: create reusable blocks of code

```
mixin pet(name)
  li.pet= name

ul
  +pet(‘Cat')
  +pet(‘Dog')
  +pet(‘Pig')
```

```
<ul>
  <li class="pet">Cat</li>
  <li class="pet">Dog</li>
  <li class="pet">Pig</li>
</ul>
```

Syntax

- Includes: insert the contents of one file into another.

```
//- header.pug
p Header

// - site.pug
include header
p Content
p Footer
```

```
<p>Header</p>
<p>Content</p>
<p>Footer</p>
```

Syntax

- Template Inheritance: block and extends

```
//- layout.pug
h1 Welcome to Pug!
block content
  p Page
```

```
<h1>Welcome to Pug!</h1>
<p>Page</p>
```

```
//- page-a.pug
extends layout

block content
  p Page A
```

```
<h1>Welcome to Pug!</h1>
<p>Page A</p>
```

Syntax

- Template Inheritance: block and extends

```pug
//- page-b.pug
extends layout

block prepend content
  p Page B

//- page-c.pug
extends layout

block append content
  p Page C
```

```html
<h1>Welcome to Pug!</h1>
<p>Page B</p>
<p>Page</p>

<h1>Welcome to Pug!</h1>
<p>Page</p>
<p>Page C</p>
```

Other Template Engine

- Haml
  - Haml (HTML abstraction markup language) is based on one primary principle: markup should be beautiful.
  - http://haml.info

- Handlebars
  - Handlebars provides the power necessary to let you build semantic templates effectively with no frustration.
  - http://handlebarsjs.com

SASS

What is Sass?

- Sass – Syntactically Awesome Style Sheets
  - Is an extension of CSS, adding nested rules, variables, mixins, selector inheritance, and more.
  - Generates well formatted CSS and makes your stylesheets easier to organize and maintain.
  - Sass has two syntaxes:
    - SCSS: Sassy CSS, is fully CSS-compatible
    - SASS: is whitespace-sensitive and indentation-based
  - Frameworks: Compass, Bourbon, Susy

Installation

- Install Sass

```
$ gem install sass
```

Note: Before you start using Sass you will need to install Ruby

Usage

- Compile

```
$ sass input.scss output.css
$ sass --watch input.scss:output.css
$ sass –watch --style expanded input.scss:output.css
$ sass --watch app/styles:static/css
```

Syntax

- Comments: /* */ and //

```
/**
  * Multiple
  * lines.
  */
body {
  color: black;
}

// Inline
a {
  color: green;
}
```

```
/**
  * Multiple
  * lines.
  */
body {
  color: black;
}

a {
  color: green;
}
```

Syntax

- Variables: uses the **$** symbol to make something a variable.
  – Flag: !global, !default

```
$primary-color: #333;

body {
  color: $primary-color;
}
```

```
body {
  color: #333;
}
```

Syntax

- Nesting Rules: rules to be nested within one another.

```
nav {
  ul {
    list-style: none;
  }
  li {
    display: inline-block;
  }
  a {
    text-decoration: none;
  }
}
```

```
nav ul {
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  text-decoration: none;
}
```

Syntax

- Referencing Parent Selectors: &

```
a {
  font-weight: bold;
  text-decoration: none;
  &:hover {
    text-decoration: underline;
  }
  .mac & {
    font-weight: normal;
  }
}
```

```
a {
  font-weight: bold;
  text-decoration: none;
}
a:hover {
  text-decoration: underline;
}
.mac a {
  font-weight: normal;
}
```

Syntax

- Partials
  - A partial is simply a Sass file named with a leading underscore, for example _colors.scss
  - The underscore lets Sass know that the file is only a partial file and that it should not be generated into a CSS file.
  - Sass partials are used with the **@import** directive.

Syntax

- Import

```
// _reset.scss
body {
  margin: 0;
  padding: 0;
}

// base.scss
@import ‘reset’;
body {
  background-color: #efefef;
}
```

```
// base.css
body {
  margin: 0;
  padding: 0;
}

body {
  background-color: #efefef;
}
```

Syntax

- Mixins

```scss
@mixin size($width, $height: $width) {
  width: $width;
  height: $height;
}

.square-box {
  @include size(300px);
}
.rectangle-box {
  @include size(300px, 100px);
}
```

```css
.square-box {
  width: 300px;
  height: 300px;
}

.rectangle-box {
  width: 300px;
  height: 100px;
}
```

Syntax

- Functions

```
@function calc-percent($target, $container) {
  @return ($target / $container) * 100%;
}

.box {
  width: calc-percent(650px, 1000px);
}
```

```
.box {
  width: 65%;
}
```

Syntax

- Extend

```
.message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  @extend .message;
  border-color: green;
}
```

```
.message, .success {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  border-color: green;
}
```

Syntax

- Interpolation: #{}

```scss
$name: foo;
$attr: border;
p.#{$name} {
  #{$attr}-color: blue;
}

p {
  $font-size: 12px;
  $line-height: 30px;
  font: #{$font-size}/#{$line-height};
}
```

```css
p.foo {
  border-color: blue;
}

p {
  font: 12px/30px;
}
```

Syntax

- Operations: + - * / %

```
p {
  font: 10px/8px;
  $width: 1000px;
  width: $width/2;
  height: (500px/2);
  margin-left: 5px + 8px/2px;
}
```

```
p {
  font: 10px/8px;
  width: 500px;
  height: 250px;
  margin-left: 9px;
}
```

Syntax

- Conditional Statement: @if, @else

```
$type: monster;
p {
  @if $type == ocean {
    color: blue;
  } @else if $type == matador {
    color: red;
  } @else if $type == monster {
    color: green;
  } @else {
    color: black;
  }
}
```

```
p {
  color: green;
}
```

Syntax

- Loop: @for

```
@for $i from 1 through 3 {
  .item-#{$i} { width: 2em * $i; }
}

@for $i from 1 to 4 {
  .item-#{$i} { width: 2em * $i; }
}
```

```
.item-1 {
  width: 2em;
}
.item-2 {
  width: 4em;
}
.item-3 {
  width: 6em;
}
```

Syntax

- Loop: @each

```
$colors: (
    ‘strawberry‘: red,
    ‘orange‘: orange,
    ‘lemon‘: yellow
);

@each $fruit, $color in $colors {
    .#{$fruit} {
        color: $color;
    }
}
```

```
.strawberry {
  color: red;
}
.orange {
  color: orange;
}
.lemon {
  color: yellow;
}
```

Syntax

- Loop: @while

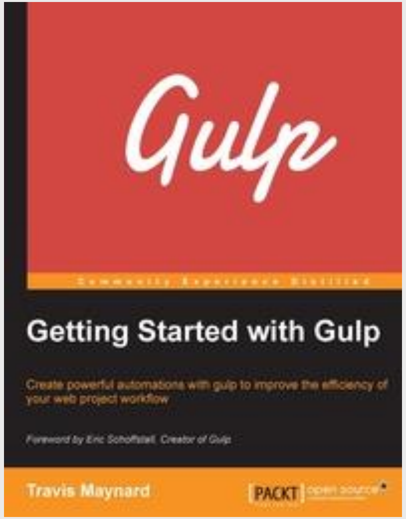
```
$i: 6;
@while $i > 0 {
  .item-#{$i} {
    width: 2em * $i;
  }
  $i: $i - 2;
}
```

```
.item-6 {
  width: 12em;
}
.item-4 {
  width: 8em;
}
.item-2 {
  width: 4em;
}
```

Other Preprocessors

- Less
  - Extends the CSS language, adding features that allow variables, mixins, functions
  - http://lesscss.org
- Stylus
  - Expressive, dynamic, robust CSS
  - http://stylus-lang.com

Reference



Community Experience Distilled

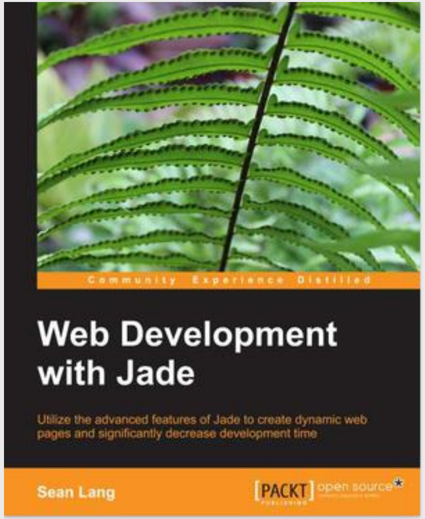
Getting Started with Gulp

Create powerful automations with gulp to improve the efficiency of your web project workflow

Foreword by Eric Schoffstall, Creator of Gulp

Travis Maynard

[PACKT] open source



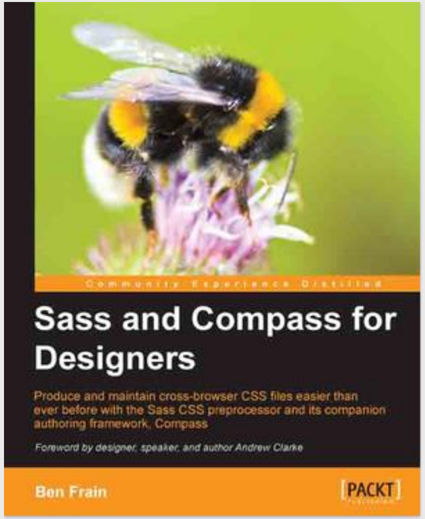
Community Experience Distilled

Web Development with Jade

Utilize the advanced features of Jade to create dynamic web pages and significantly decrease development time

Sean Lang

[PACKT] open source



Community Experience Distilled

Sass and Compass for Designers

Produce and maintain cross-browser CSS files easier than ever before with the Sass CSS preprocessor and its companion authoring framework, Compass

Foreword by designer, speaker, and author Andrew Clarke

Ben Frain

[PACKT]

Assignment

- Create a Gulp package that can automatically do some tasks below:
  - Compiles Pug to HTML
  - Compiles Sass to CSS
  - Check code quality for JS and CSS
  - Optimizes Images, CSS, JS for production
  - Reload the browser whenever a file is changed
- Create a simple Pug template that can do some tasks below:
  - Define master layout that can include header, footer, script
  - Create simple Homepage and About page extend from master layout
  - Create a mixin for creating menu 2 levels
  - Use Sass to style the template

Q&A

THANK YOU