DrupalCon Barcelona 2015

Fundamentals of Front-End Ops

Preston So

September 22, 2015

Welcome!

- **Preston So** (@prestonso) has designed websites since 2001 and built them in Drupal since 2007. He is Development Manager of Acquia Labs at Acquia and co-founder of the Southern Colorado User Group.

preston.so
drupal.org/u/prestonso
preston.so@acquia.com
pso@post.harvard.edu

What we'll cover

1. Why front-end ops?
2. Scaffolding and dependencies
3. Automation: Grunt and Gulp
4. Regressions and rendering
5. Tools and discussion

Why front-end ops?

1

Why ops?

Why front-end ops?

Front-end workflow

Why ops?

- Automation engenders consistency.
- Leverage iterative development.
- DevOps:

  Product delivery

  Quality testing

  Release management

Why front-end ops?

- We have many front-end tasks that would be better served with automation.
- In the past, we only needed to worry about a few HTML and CSS files, and perhaps a script or two.
- Today, we have many new front-end abstractions and frameworks that have their own dependencies.

Why front-end ops?

- Front-end ops (Alex Sexton) is a response to the proliferation of front-end tools and frameworks, as “more application logic is being deferred to the client side.”
- The key difference between traditional front-end development and front-end ops is the emphasis on **process**.

Front-end ops

- Sexton: There is now too much work in front-end ops for FEDs to do everything on their own.
- Having a focus on operations yields a more iterative workflow focused on improving process.
- Chris Ruppel: Front-end ops is “how to automate the process of breaking things.”

Traditional front-end workflow

- Before, the traditional front-end workflow was simple and had a single track.

- A little basic scaffolding (download Drupal)
- Add a few dependencies (jQuery)
- Edit, upload, *voilà*

Modern front-end workflow

- Today, we have too many points of potential failure along the path that front-end code takes.

  Scaffolding (perhaps many steps)

  Download libraries and frameworks (and manage all these dependencies)

  Watch Sass/Compass, CoffeeScript, Haml

  Lint JS/CSS for standards

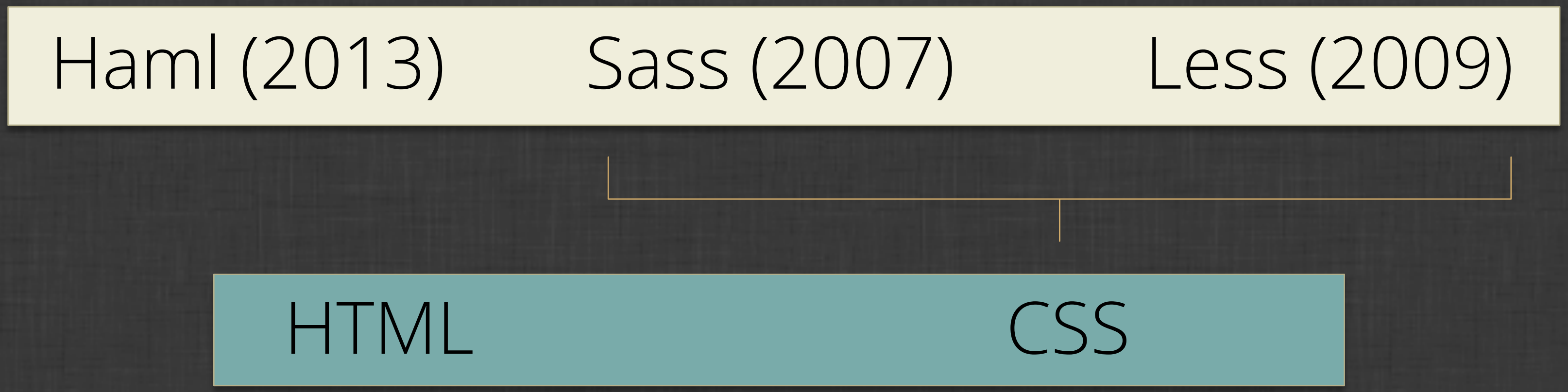
  Test suites and debugging tools

Modern front-end workflow

- Especially when we have very diverse needs:

  Unit tests
  Preprocessing
  Minification and concatenation
  Performance
  Display testing
  Builds and deployment

Abstractions



A front-end ops workflow

- More abstraction means more overhead.
- More team members means more potential for unaligned code.
- More front-end code means more risk of error.
- More users means more drags on performance.

A front-end ops workflow

- We need an easier way to scaffold, manage dependencies, and automate tasks.

A front-end ops workflow

- We need an easier way to scaffold, manage dependencies, and automate tasks.

- Introducing …

Yeoman (app scaffolding)
Bower (dependency management)
Grunt/Gulp (task automation)

A front-end ops workflow

- Addy Osmani: How does this new workflow help us?

  “Flexibility to customize your setup as much as you desire.” (Yeoman generators, bower.js, gulpfile.js)

  “Limit the time spent writing boilerplate.” (Yeoman)

  “Improve your productivity and delight during development.” (Bower, Grunt, Gulp)

Yeoman

- Yeoman explicitly recommends a workflow that involves Bower and Grunt or Gulp.
- You can install Yeoman generators and write your own, and generators exist for frameworks such as Angular.js and Bootstrap.
- Yeoman can also scaffold portions of apps for isolated testing or demonstration, especially for Angular.js.

Bower

- Bower helps you find your dependencies and updates them when you prefer.

- Bower provides one single command-line interface to manage all versioned and potentially obsolescent dependencies.

Grunt and Gulp

- Grunt and Gulp are task automators, which given some configuration in a Gruntfile or gulpfile.js, will run through selected piped tasks.
- The Grunt and Gulp communities are replete with plugins that provide many useful commands.
- Grunt and Gulp reduce the time for you to get your code to a deliverable state.

Roles and responsibilities

- Sexton proposes a new *front-end operations engineer* role that would oversee front-end ops responsibilities and argues that further specialization is imminent.
- I believe that the trend of front-end development is toward diversification rather than specialization: developers will increasingly need to know more and more of the stack.

Scaffolding and dependencies

2

Bower

Yeoman

Scaffolding a Drupal theme

Node.js

- First, install Node.js at nodejs.org.

```
$ node -v
v4.0.0
```

npm

- Check the version of npm, package manager.

```
$ node -v
v4.0.0
$ npm -v
3.3.3
```

npm

- If need be, update npm.

```
$ node –v
v0.12.4
$ npm -v
3.3.3
$ npm install npm -g
```

npm

- With npm, we can install what we need once.

```
$ npm install –g yo bower grunt-cli gulp
```

bower

- Bower takes care of the grunt work in finding, downloading, and tracking your dependencies.
- Bower leverages a manifest file, *bower.json*, which contains a list of your packages.

bower

- Install packages to bower_components/

```
# An already registered package.
$ bower install jquery

# GitHub shorthand or Git endpoint.
$ bower install drupal/drupal

# A path.
$ bower install http://my.com/package.js
```

bower

- Add to the manifest with *bower init*.

```
# Save packages to the manifest.
$ bower init

# Search Bower packages.
$ bower search

# List saved dependencies in project.
$ bower list
```

yo

- Now that our dependencies are sorted, let’s get our basic scaffolding.
- For this session we will be generating a Drupal theme in Yeoman, using this generator by **Ian Carrico**:

github.com/frontend-united/
generator-drupal-theme

yo

- Install a Yeoman generator.

```
$ npm install -g generator-drupal-theme
```

yo

- Scaffold a Drupal theme with initial files (*demo*).

```
$ mkdir barcelona2015 && cd barcelona2015
$ yo drupal-theme
```

package.json

- package.json declares our dev dependencies.

```
{
  "name": ”barcelona2015",
  "version": "0.0.0",
  "dependencies": {},
  "devDependencies": {
    "gulp": "^3.6.0",
    "gulp-jshint": "^1.5.1",
    "jshint-stylish": "^0.1.5",
```

package.json

- … continued.

```
    "compass-options": "^0.1.1",
       "browser-sync": "^0.7.4",
       "gulp-shell": "^0.2.4"
     },
     "scripts": {
       "postinstall": "find node_modules/ -
  name \"*.info\" -type f -delete"
     }
  }
```

Writing Yeoman generators

- We need a new directory with this package.json:

```
{
  "name": "generator-name",
  "version": "0.1.0”
  "description": "",
  "keywords": ["yeoman-generator"],
  "dependencies": {
    "yeoman-generator": "^0.17.3"
  }
}
```

Writing Yeoman generators

- Writing Yeoman generators is beyond the scope of this session, but Yeoman has an exceptional authoring resource:

yeoman.io/authoring/

Automation: Grunt and Gulp

3

Grunt

Gulp

Automating a Drupal theme

Grunt and Gulp

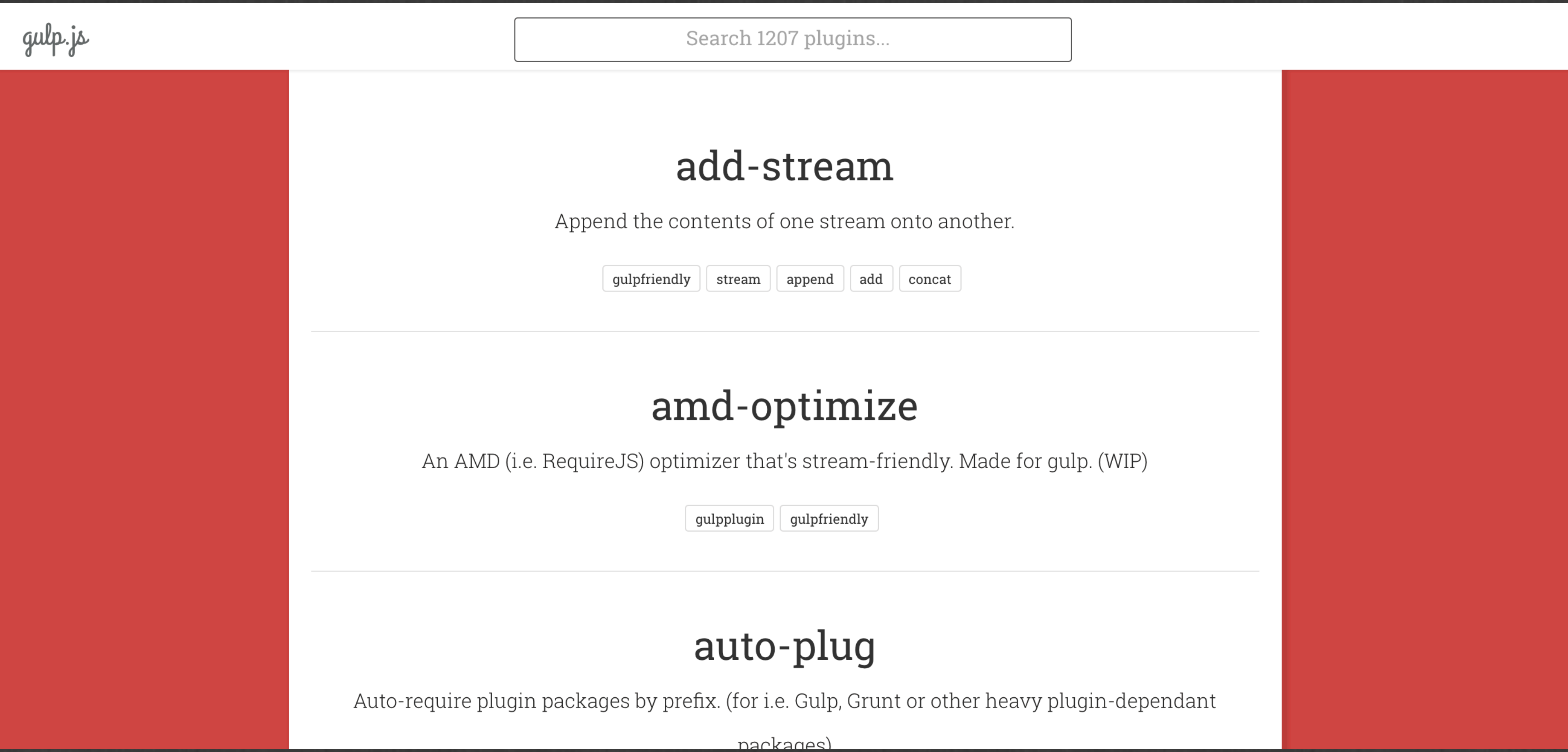
- There is no significant difference between Grunt and Gulp to the front-end developer.
- Their syntaxes and goals are slightly different; we’ll talk about both in this section.

gruntjs.com/plugins
gulpjs.com/plugins

Plugins

Plugin	Updated	Grunt Version	Downloads last 30 days
★ contrib-clean by Grunt Team Clean files and folders.	6 months ago	~0.4.0	**653323**
★ contrib-uglify by Grunt Team Minify files with UglifyJS.	about a month ago	~0.4.0	**647922**
★ contrib-copy by Grunt Team Copy files and folders.	4 months ago	~0.4.0	**596210**
★ contrib-concat by Grunt Team Concatenate files.	7 months ago	~0.4.0	**563112**
★ contrib-watch by Grunt Team Run predefined tasks whenever watched file patterns are added, changed or deleted.	7 months ago	~0.4.0	**498276**
★ contrib-jshint by Grunt Team Validate files with JSHint.	10 months ago	~0.4.0	**471029**
★ contrib-connect by Grunt Team Start a connect web server.	3 months ago	~0.4.0	**453994**

Plugins



add-stream

Append the contents of one stream onto another.

gulpfriendly stream append add concat

amd-optimize

An AMD (i.e. RequireJS) optimizer that's stream-friendly. Made for gulp. (WIP)

gulpplugin gulpfriendly

auto-plug

Auto-require plugin packages by prefix. (for i.e. Gulp, Grunt or other heavy plugin-dependant packages)

Plugins

- First, let’s install a Grunt plugin.

```
$ npm install grunt-contrib-uglify
  >> --save-dev

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

Grunt vs. Gulp

- With the *--save-dev* flag, Grunt and Gulp will both automatically add the plugin to package.json as a development dependency.
- Where Grunt and Gulp differ is in the files they require in the project and in their focus: Grunt is more configuration-focused, while Gulp is more task execution-focused.

Gruntfile.js

- Here’s the initial structure of our Gruntfile.js:

```
module.exports = function (grunt) {

  // Configure Grunt here.

};
```

Gruntfile.js

- Let’s get Grunt to read our package.json.

```
module.exports = function (grunt) {
  // Configure Grunt here.
  grunt.initConfig({
    pkg: grunt.file.readJSON(
>> ‘package.json’),
  });

};
```

Gruntfile.js

- We can also configure the plugin.

```
module.exports = function (grunt) {
  // Configure Grunt here.
  grunt.initConfig({
    uglify: {
      // Configure uglify here.
    }
  });
};
```

Gruntfile.js

- Load the task and register the task with Grunt.

```
module.exports = function (grunt) {
  // Tell Grunt that plugin will be used.
  grunt.loadNpmTasks(‘grunt-contrib-
>> uglify’);
  // Provide Grunt a task to register.
  grunt.registerTask(‘default’,
>> [‘uglify’]);
};
```

Gruntfile.js

- Within *grunt.initConfig*(), let’s configure.

```
uglify: {
  options: {
    // Plugin-specific configuration.
  },
  dist: {
    src: [‘src/**/*.js’],
    dest: ‘dist/<%= pkg.name %>.js’
  }
};
```

Gruntfile.js

- Let’s run grunt!

```
// With our task registered …
grunt.registerTask(‘default’,
>> [‘uglify’]);
};

# … we can run grunt.
$ grunt
```

Gruntfile.js

- Let’s run grunt!

```
// With our task registered …
grunt.registerTask(‘concat-uglify’,
>> [‘concat’, ‘uglify’]);
};

# … we can run grunt.
$ grunt concat-uglify
```

Useful Grunt plugins

- *grunt-responsive-images*
  Save multi-resolution images to a destination.

- *grunt-contrib-imageoptim*
  Optimize images for web use.

- *grunt-newer*
  Only execute tasks on modified files.

Useful Grunt plugins

- *grunt-uncss*
  Remove CSS not used across your project.

- *grunt-uncss* and *gulp-uncss* can also detect and remove styles injected into the page dynamically, leveraging PhantomJS.

gulpfile.js

- The syntax of gulpfile.js is slightly different.

```
var gulp = require(‘gulp’);

gulp.task(‘default’, function() {
  // Code for your task.
});

# Run gulp.
$ gulp
```

gulpfile.js

- First, we invoke *require*() on plugins.

```
var gulp = require('gulp');
var jshint = require('gulp-jshint');
var compass = require('gulp-compass');
var concat = require('gulp-concat');
var uglify = require('gulp-uglify');
```

gulpfile.js

- Then, we set paths.

```
var paths = {
  scripts: ‘js/**/*.js’,
  img: ‘img/**/*’
};
```

gulpfile.js

- Then, we configure and order tasks.

```
gulp.task(‘process-js’, function() {
  return gulp.src(paths.scripts)
    .pipe(concat(‘processed.js’))
    .pipe(gulp.dest(paths.js))
    .pipe(uglify())
    .pipe(gulp.dest(paths.js));
});
```

gulpfile.js

- Let’s run gulp!

```
$ gulp process-js
```

Grunt vs. Gulp

- It’s often useful to look at different codebases’ Gruntfile.js and gulpfile.js files to iterate on your front-end workflow. What works for other teams may not work for yours.
- In Grunt, tasks are preconfigured then registered.
- In Gulp, tasks are configured as they register.

Grunt vs. Gulp

- We can use Gulp to track changes in our theme files and automate what's tedious so we can focus on what's important. (*demo*)
- Whenever package.json (or bower.json) changes, run *npm install* or *bower install* to get the most up-to-date dependencies.

Regressions and rendering

4

Visual regressions

Testing rendering engines

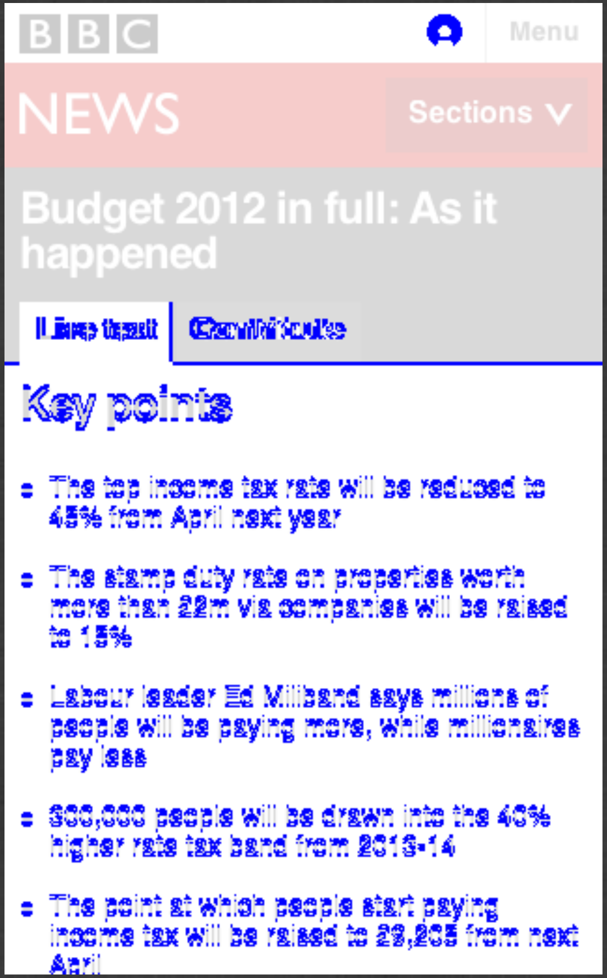
Testing devices

Visual regressions

- CSS is usually fast-moving and prone to more errors than other languages.
- Wraith leverages PhantomJS or SlimerJS to snap screenshots as visual diffs between two environments.

github.com/BBC-News/wraith

Visual regressions



BBC

Menu

NEWS

Sections

Budget 2012 in full: As it happened

Live text

Key points

- The top income tax rate will be reduced to 45% from April next year
- The stamp duty rate on properties worth more than £2m via companies will be raised to 15%
- Labour leader Ed Miliband says millions of people will be paying more, while millionaires pay less
- 300,000 people will be drawn into the 40% higher rate tax band from 2013-14
- The point at which people start paying income tax will be raised to £9,205 from next April

Wraith

- Install Wraith and set up for capturing.

```
# Install Wraith.
$ gem install wraith

# Create template JS and config YAML.
$ wraith setup
```

Wraith

- Install Wraith and set up for capturing.

```
# Start Wraith and capture using configs.
$ wraith capture configs/config.yml
```

Visual regressions

- Huxley (built by Facebook and Instagram, but currently unsupported) helps you by scrutinizing diffs in screenshots.

github.com/facebookarchive/huxley

Huxley

- Huxley “watches you browse, takes screenshots, [and] tells you when they change.”
- Huxley uses Huxleyfiles that allow you to configure the URLs to be tested.
- Huxley generates .huxley files that are records of your tests whose changes you can track.

Huxley

- Hit *Enter* to take a screenshot, and *q* to quit.

```
# Run Huxley.
$ huxley --record
```

PhantomCSS

- PhantomCSS automates visual regression testing for “web apps, live style guides, and responsive layouts.”

github.com/Huddle/PhantomCSS
tldr.huddle.com/blog/css-testing

Testing rendering engines

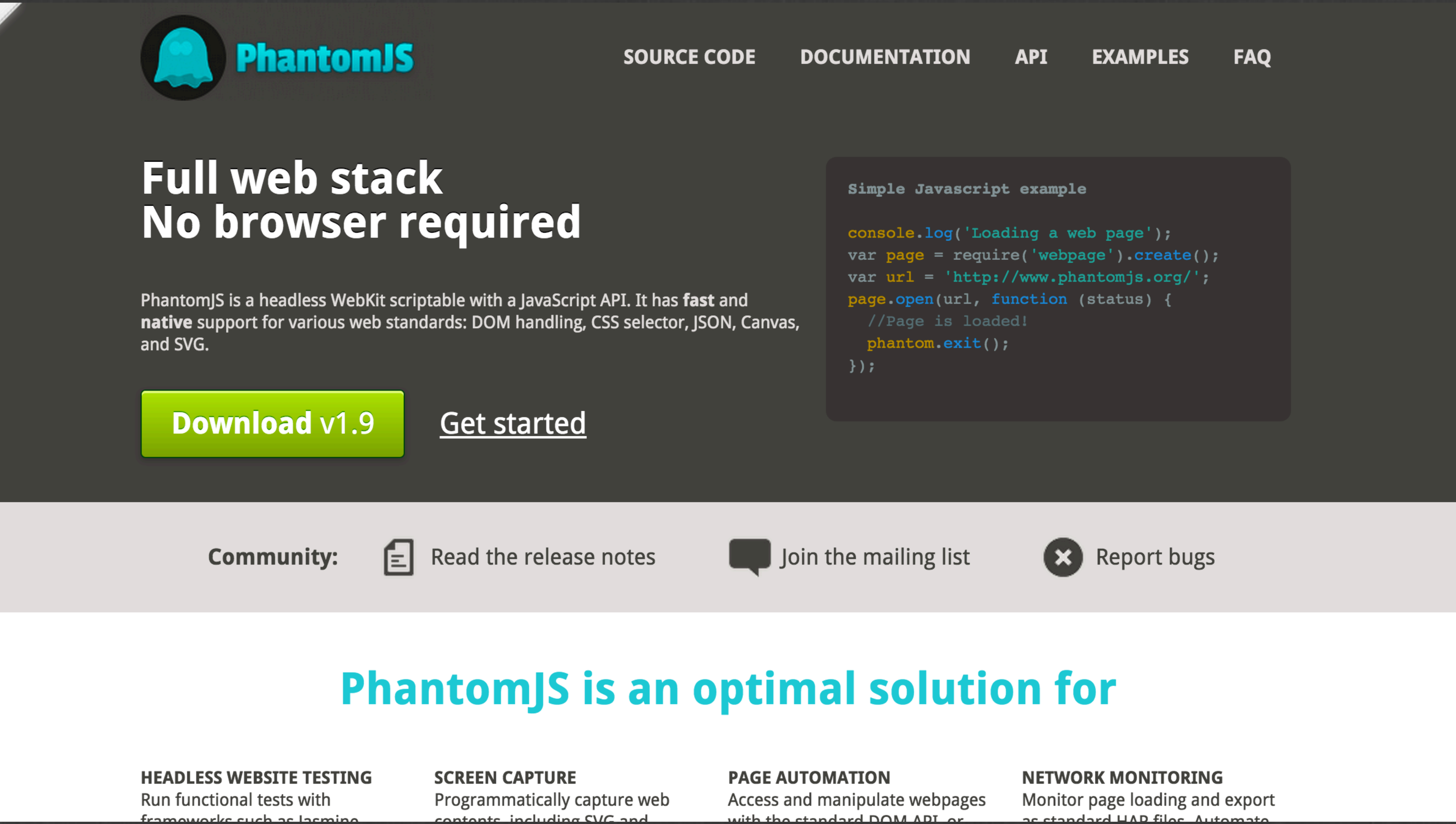
- A faster front-end workflow means we need a faster turnaround on tests.
- How can we test rendering engines more quickly and without waiting for pageloads on each browser?

Testing rendering engines

- We can use headless instances of rendering engines to render our pages without display.
- PhantomJS (Webkit) can be set to spit out visual pages when asked. It’s particularly useful for batch actions on web pages (screenshots, viewport changes, etc.).

phantomjs.org

Testing rendering engines



Full web stack
No browser required

PhantomJS is a headless WebKit scriptable with a JavaScript API. It has **fast** and **native** support for various web standards: DOM handling, CSS selector, JSON, Canvas, and SVG.

Simple Javascript example

```
console.log('Loading a web page');
var page = require('webpage').create();
var url = 'http://www.phantomjs.org/';
page.open(url, function (status) {
  //Page is loaded!
  phantom.exit();
});
```

PhantomJS is an optimal solution for

HEADLESS WEBSITE TESTING
Run functional tests with

SCREEN CAPTURE
Programmatically capture web

PAGE AUTOMATION
Access and manipulate webpages

NETWORK MONITORING
Monitor page loading and export

Testing rendering engines

- SlimerJS is the equivalent for Gecko.

  slimerjs.org

- CasperJS builds on top of PhantomJS or SlimerJS to provide a great deal of interaction, including form-filling, clicking links, logging, and scraping.

  casperjs.org

Testing devices

- GhostLab allows you to conduct synchronized testing on diverse types of devices.

  vanamco.com/ghostlab

- Other device simulators are available, such as Xcode’s iOS Simulator.

GhostLab

- Synchronize navigation across mobile and desktop and all devices.
- Synchronize taps, clicks, scrolls, and other user interactions.

Tools and discussion

5

Front-end and debugging tools

Chrome DevTools

Discussion

Chrome DevTools

- DevTools Remote Debugging allows for better mobile and tablet testing.
- Conduct audits of CSS to determine which CSS is unused—same story as *grunt-uncss*.
- DevTools Terminal gives you a shell in Chrome.

github.com/petethepig/devtools-terminal

Up for discussion

- What is the future of front-end development with the advent of front-end ops?
- How will development workflows change due to front-end ops?
- What will front-end workflows look like 1 year from now? 5 years from now?

Useful resources

- Intro to Front-End Ops (Chris Ruppel)
  http://rupl.github.io/frontend-ops
- Front-End Ops (Alex Sexton)
  http://www.smashingmagazine.com/2013/06/11/front-end-ops/

Useful resources

- Automating Workflow (Addy Osmani)
  https://speakerdeck.com/addyosmani/automating-front-end-workflow

- Grunt for People Who Think Things Like Grunt Are Weird and Hard (Chris Coyier)
  http://24ways.org/2013/grunt-is-not-weird-and-hard/

More front-end ops at Barcelona

- Visual Regression Testing

  Amitai Burstein; 13:00-14:00; 117 (Acquia)

What did you think?

- Evaluate this session at:

  barcelona2015.drupal.org/schedule

Thank you!

- **Preston So** (@prestonso) has designed websites since 2001 and built them in Drupal since 2007. He is Development Manager of Acquia Labs at Acquia and co-founder of the Southern Colorado User Group.

preston.so
drupal.org/u/prestonso
preston.so@acquia.com
pso@post.harvard.edu