

Magento 2 CSS Preprocessing

Less wanted.





Hrvoje Jurisic

Magento Frontend Developer - Inchoo

twitter: @hrvojejurisic



A photograph of a long, straight asphalt road stretching into the distance. The road is marked with a solid yellow center line. It is flanked by fields of crops, likely corn, on both sides. The sky above is a clear blue with some wispy white clouds. In the center of the image, the words "MAGENTO 1" and "CSS" are overlaid in large, bold, orange capital letters.

MAGENTO 1
CSS

Magento 1

page.xml:

```
<default translate="label" module="page"
    <block type="page/html" ..... template="page/3columns.phtml">
        <block type="page/html_head" name="head" as="head">
            <action method="addCss"><stylesheet>css/styles.css</stylesheet></action>
        </block>
    </block>
</default>
```



MAGENTO 2 CSS



Magento 2

default_head_blocks.xml:

```
<page>
  <head>
    <css src="css/styles-m.css" />
    <css src="css/styles-l.css" media="screen and (min-width: 768px)" />
    <css src="css/print.css" media="print" />
  </head>
</page>
```



Luma Theme

CSS file:

`/pub/static/frontend/Magento/luma/en_US/css/styles-l.css`



pub/static/frontend/Magento/luma/en_US/css/styles-l.css



app/design/frontend/Magento/luma/web/css/styles-l.css



app/design/frontend/Magento/luma/web/css/styles-l.less

LESS COMPILATION



A man with dark hair and a beard, wearing a grey t-shirt, is shown from the chest up. He is looking upwards and to his right with a thoughtful expression, his hand resting against his chin. A white thought bubble originates from his mouth and contains text.

But then...
I could have my own
decoupled preprocessing
workflow, and just include the
end result css files via XML,
just like we did on
Magento 1??

A photograph of a long, straight asphalt road stretching into the distance. The road is marked with a solid yellow center line. It is flanked by fields of crops, likely corn, on both sides. The sky above is a clear blue with some wispy white clouds. In the center of the image, the text "Yes, you can!" is written in a large, bold, orange sans-serif font.

Yes, you can!

A photograph of a long, straight asphalt road stretching into the distance under a vast, cloudy sky. The road is flanked by fields on both sides.

Yes, you can!

But what about:

1. Extensions ?
2. Maintainability ?
3. Certification ?

LESS Preprocessing



LESS



- Preprocessor
- Extending CSS
- Introduces:
 - Variables
 - Mixins
 - Functions
 - Nesting



LESS variables

```
@heading-color: #00FF00;
```



LESS variables

```
@heading-color: #00FF00;
```

Less file:

```
h1 {  
    color: @heading-color;  
}
```



LESS variables

```
@heading-color: #00FF00;
```

Less file:

```
h1 {  
    color: @heading-color;  
}
```

CSS:

```
h1 {  
    color: #00FF00;  
}
```



LESS variables

```
@heading-color: #00FF00;
```

New Module Less file:

```
.my-heading {  
    color: @heading-color;  
}
```

CSS:

```
.my-heading {  
    color: #00FF00;  
}
```



Why Preprocessing? Why LESS?

- Variables allow simple module and theme “communication”
- Modules and themes can easily override defaults
- At the time of decision-making, LESS was the only preprocessor with stable PHP compiler



LESS Preprocessing

1. Client Side
2. Server Side



Settings:

Stores > Configuration > Advanced > Developer > Frontend development workflow

Client side LESS compilation

- Compilation inside the browser
- Uses native less.js compiler



Frontend development workflow - for admin too :)

You saved the configuration.

GENERAL

CATALOG

CUSTOMERS

SALES

SERVICES

ADVANCED

Admin

System

Advanced

Front-end development workflow

Workflow type: Client side less compilation [GLOBAL]
Not available in production mode

Developer Client Restrictions

Debug

Template Settings

Translate Inline

JavaScript Settings

CSS Settings

Image Processing Settings

Static Files Settings

Grid Settings



Server side LESS compilation

- Default workflow
- Only option available in ***Production Mode***
- Uses built-in PHP compiler



Server side LESS compilation - how?

1. System searches for included files in pub/static/frontend/vendor/theme/locale
2. If found - return files
3. If not found - searches for them in active theme folder
4. If found - creates symlinks in pub/static/frontend/vendor/theme/locale
5. Not found - searches for same file names with .less extension
6. LESS Compile
 - a. Reads .less files content and resolves **@magento_import** and default LESS **@import** directives
 - b. Resolves all paths in LESS files to relative paths using fallback mechanism
 - c. All files resolved by LESS preprocessor are copied to var/view_preprocessed/
 - d. All source files are passed to PHP LESS compiler
 - e. Resulting .css files are published to pub/static/frontend/vendor/theme folder



@magento_import

LESS **@import** imports content of specified single file into current file.

@magento_import - Magento specific import allows including multiple files with the same name.

Unprocessed:

```
//@magento_import 'source/_module.less';
```

Processed:

```
@import '../Magento_AdvancedCheckout/css/source/_module.less';
@import '../Magento_Bundle/css/source/_module.less';
@import '../Magento_Catalog/css/source/_module.less';
@import '../Magento_CatalogEvent/css/source/_module.less';
@import '../Magento_CatalogSearch/css/source/_module.less';
```





`pub/static/frontend/vendor/theme/web/css
var/view_preprocessed`

That's a lot of cleaning!

Automated Preprocessing with Grunt



What is Grunt?

- Javascript task runner
- Automation of repetitive tasks
 - Watch
 - Compile
 - Clean
 - LiveReload



Working with Grunt

Prerequisites:

1. Install node.js
2. Install Grunt globally

```
npm install -g grunt-cli
```

3. Install Grunt in your Magento 2 project

```
npm install
```

```
npm update
```

4. Add theme to Grunt configuration in
dev/tools/grunt/configs/themes.js
5. Set up LiveReload





//@magento_import?

Magento Theme
Fallback??

bin/magento dev:source-theme:deploy

1. Resolve all file fallback paths
2. Create symlinks to source (.less) files
3. Expand all “@magento_import” to import single files
4. Publish all files in a tree to pub/static folder



Magento 2 comes with built-in grunt tasks

- **grunt clean:<theme>**

Removes static files from pub and var folders

- **grunt exec:<theme>**

*Creates whole tree with symlinks to the source .less files in
pub/static/frontend/vendor/theme/web*

- **grunt less:<theme>**

Compiles .css files using symlinks published in pub folder

- **grunt watch**

*Watches for changes in source .less files, compiles .css and injects new styles in
browser without page refresh (via Livereload)*



Stop... Rebuild... Watch...

- If you change root source files (*styles-l.less*):
- If you add, delete or rename imported .less files:
 1. Stop the Watcher
 2. Grunt exec
 3. Grunt watch



A photograph of a paved path winding through a dense forest. The path is grey asphalt with some brown dirt and fallen leaves. The forest is filled with tall, thin trees, mostly coniferous, with green needles. Some deciduous trees with green leaves are also visible. The lighting suggests it might be early morning or late afternoon, with sunlight filtering through the canopy.

Alternatives

Gulp Cake Broccoli

Npm

...



Build... Watch...

1. bin/magento dev:source-theme:deploy
2. Watch changes in pub/static/... and write resulting css to that folder

Community:

<https://github.com/poddubny/magento2-gulp>

<https://github.com/SnowdogApps/magento2-frontools>





Sass

Developers Paradise
2016 / Opatija / Croatia



You can add custom preprocessor

Documentation:

http://devdocs.magento.com/guides/v2.0/frontend-dev-guide/css-topics/custom_preprocess.html

Sample Scss Module

<https://github.com/magento/magento2-samples/tree/master/module-sample-scss>

- Proof of concept
- Ignores @import directives

Blank Theme on Sass

<https://github.com/SnowdogApps/magento2-theme-blank-sass>





**Sass
what's the point?**

Developers Paradise
2016 / Opatija / Croatia



Thank You!



DevelopersParadise
2016 / Opatija / Croatia