No Frills Magento 2 Layout

Alan Storm

April 2017

IMPORTANT

Contents

Chapter 1

Introduction

The kindest thing I can say about writing this book is that it's been a challenge. The best way to introduce this book is to explain why it's been so challenging.

Writing the first edition of No Frills Magento Layout was no picnic. Even in 2011, the level of knowledge around Magento's HTML-generating domain-specific-language was scant. However, in the world of web development, there **was** a consensus on how we should be building web applications.

1. HTML delivered to the browser, possibly generated with server-side languages like PHP, ruby, python, java, etc.
2. CSS for layout and styles, with an occasional image or background-image if you wanted to get fancy
3. Javascript to add interactive elements to the page
4. AJAX requests if those interactive elements needed additional server resources/information

Even in 2018, these four principles remain a solid and proven way to build software that's delivered to users via a web browser.

Unfortunately, in the years since Magento's release and my writing the first edition of this book, **a lot** has changed in the world of web development.

The biggest change has been the rise of mobile and the stagnation of the mobile web. Chrome and Safari on Android and iOS based phones are wonders of software engineering, but the web browser remains a second class citizen in the battery constrained world of mobile devices. Where the aughties culminated with web-browsers recognized as the superior method of delivering cross platform software, the mobile computing world took a few steps backwards into device specific software and locked-in file formats. In this enviornment "the web" became a transport layer for data, which in turn led to the server side world becoming a specialized, UI-less island.

CSS has also seen its fair share of changes. Again, the aughties saw a wealth of clever CSS design and development techniques arise, often on a grassroots level. These techniques allowed web developers to tame the complicated world of cascading style sheets. In 2018, these techniques still exist, but they're locked into specific frameworks (Bootstrap) or preprocessing programs (LessCSS, Sass). When you consider a mobile web that has given up on a stable "CSS API" it's clear that writing your style sheets without one of these tools is a dicey proposition. By itself this might not be a problem, but these tools are often developed inside of agency culture, a culture that's more interested in short term results than they are in long term sustainability. This makes for complicated toolchains that change dramatically year-to-year.

All this bleeds over to the world of Javascript, where it's both the best of times and worst of times. The ubiquity of javascript in the browser has brought a **tremendous** amount of engineering resources to bear on the language, runtimes, and third party libraries. There's now a school of thought that says the **only** HTML you need to render on the server is the HTML that bootstraps your javascript runtime. However, despite (or because of?) all this attention, javascript based applications tend to be built-on hundreds of thinly sliced libraries with no dominant paradigm from application to application.

The PHP landscape has also seen its fair share of changes – the most important of which is Composer. While Composer modestly bills itself as a dependency manager, in reality it's taken off (and over) as PHP's de-facto package manager, autoloader bootstrap enviornment, and distribution channel for PHP code.

Another change has been PHP 7.0, 7.1, and 7.2's making a Zeno like march towards explicitly typed language semantics. PHP 7 lets developers familiar with java like semantics ply their wares like never before, and the tension between done-quick vs. done-correct has never been higher within the PHP community.

While some would chalk all this up as progress, it's balkanized the developer community. *The Right* way to develop a web application is much less clear. There's a tension as to how much attention we should pay to the web application vs. just building an API for mobile support.

Magento 2 jumped into a web development world that's dramatically changed, and it's not clear which direction (if any) Magento core is leaning with regards to to the chaos.

Magento 2

With Magento 2, Magento Inc.'s committed hard to PHP's trends. Composer's fully embraced, Magento 2's PHP semantics favor multi-level class based abstractions, and the XML configuration files have multiplied like guinea pigs. While all this does confer the Magento system with certain enterprise market

advantages, it's also created a system where getting started is a tad more complicated than dropping your PHP files into a `public_html` folder.

Magento's Layout XML files are a *domain specific language* that use PHP classes and templates to render HTML. In this way, Magento 2 is a traditional server side framework, similar to the version developed in the aughties.

However: Magento 2 has **also** revamped its API. Formerly a tool for data transport, Magento 2's new API puts REST services front and center, including browser based sessions and permissions. It's the sort of API you can call directly from javascript.

To support building javascript applications using this API, Magento 2 also features a RequireJS module system, a significant extension of Knockout.js for AJAX based template rendering, and a **new** domain specific language that enables pure javascript based UIs with **no server side rendered HTML**. In this way, Magento seems to be forward thinking.

However however: While Magento 2 has these newer systems in place, the Magento 2 application remains a *mix* of pure javascript UI, and the older server side rendered HTML *enhanced* by javascript.

Finally, in the realm of CSS, the Magento application ships with a heavily integrated LessCSS system and a grunt build enviornment. There's been community led efforts to enable both Sass and gulp for Magento 2 development workflows, but the core system itself has LessCSS as a hard dependency. This means developers targeting **all** Magento systems face some hard choices with regards to to their CSS build pipelines.

What We'll Cover

All of which is a preamble to saying: This book is not a top to bottom course in Magento's front end systems. Also, while you can get something out of this book if you're coming in fresh to Magento, you'll be best served if you already have an inkling of where Magento 2 came from. To help with this we've included a copy of the original No Frills Magento Layout with your purchase.

We'll cover using Magento's *layout handle XML files* to render HTML content at a module level. We'll also cover what *themes* are in Magento 2, how their handling of layout handle XML files has changed, and how you can use LessCSS via themes to style Magento pages. Finally, we'll take a brief survey of how Magento uses new front end technologies like RequireJS, Knockout.js, and LessCSS and show you how to get your front end files loaded into Magento's application context.

Conventions

There's a few last bits of intro business before we can get on with the book.

When you downloaded your copy of this book you also received a `Pulsestorm_Nofrillslayout` Magento module. You'll want to install this single module into your system, as many of the code samples included in this book start with this module as their base. If you're unsure how to install a stand alone module in Magento 2, read the "Installing the `Pulsestorm_Nofrillslayout` Module" appendix.

Once installed, you'll want to flip your system into `developer` mode. The simplest way to do this is to run the following command

```
1 $ php bin/magento deploy:mode:set developer
```

If you're not familiar with Magento's various modes, or not familiar with the command line, we have an appendix that covers each of those as well.

The appendixes are quick primers on topics that aren't *quite* related to Magento's layout system, but are still required knowledge for working with the system. We've broken them off into appendixes to avoid breaking flow in our tutorials.

Alright, let's get to it!

Chapter 2

Block Basics

The atomic unit of an HTML page in Magento 2 is a block. A block is a PHP object that renders a bit of HTML. To start with, we're going to use PHP to instantiate a block object, and then output a bit of HTML.

...

Chapter 3

Building Layouts via XML

In Chapter 1, we spent a lot of time using PHP code to create block objects. Towards the end, you may have noticed the PHP code involved was growing in both size and complexity. Depending on your comfort level and expertise with PHP, this was somewhere between a "mild annoyance" and an "occasion to go into the stairwell and cry".

...

Chapter 4

Layout Handles

So far in this book, we've been directly `echo`ing output from a controller action. While this works, and is a great way to learn, it's **not** how client programmers are meant to use Magento's system.

...

Chapter 5

Page Layouts in Magento 2

When it comes to the HTML page layout for a page, so far we've spent a lot of time talking about

1. Individual components of Magento's layout system and using those components to build up simple examples "from scratch"
2. Starting with a fully rendered Magento page and slightly altering it.

In this chapter, we're going to dive a bit deeper and take a look at how **Magento** creates its fully designed layouts from scratch. We'll do this by stripping a Magento MVC endpoint down to a naked, blank, HTML page.

...

Chapter 6

Magento 2 Theming

Theming systems occupy a strange space in web application frameworks. From a user facing perspective, it may seem obvious what themes are – they determine how a site looks and feels. However, on a technical level, a theming system is something that allows developers *limited* access to the system that outputs HTML and CSS for web pages. The question becomes **how** limited is that access.

Themes in Magento 2 are, technically speaking, pretty straight forward. A Magento theme allows theme developers to **replace** or **add to** assets added by Magento modules, or by their parent themes. These assets include

- Layout Handle XML files
- CSS/LessCSS files,
- Javascript source files
- Knockout.js html templates
- Email templates
- Additional requirejs-config.js configuration
- Additional translation strings

In this chapter we're going to create a new child theme, and then use that theme to replace/extend a few of Magento's stock layout handle xml files, phtml files, and LessCSS files.

...

Chapter 7

Advanced XML Loading

Compared to Magento 1, the code behind Magento 2's layout rendering is woefully complicated. There's no clear story to be told about *how* a Magento 2 page renders. There is, perhaps, a story to be told about what happens when a team of engineers sets out to refactor and enhancing a domain specific language they neither use nor fully understand. That is, however, a story for another day.

While Magento 2's layout system uses many of the same objects as Magento 1's layout system and still uses XML as its medium for a domain specific language, *how* Magento 2 uses these objects has changed dramatically. There are also new, similarly named objects that perform new tasks, heretofore unseen in the world of Magento layouts.

This preamble is to warn you that this section will be both difficult and rarely necessary for day-to-day work. Skip ahead to the front end chapters if you like.

...

Chapter 8

Front End Starter Kit – CSS

We've just spent 6 chapters winding our way through the complex world of generating HTML using Magento's domain specific layout XML language. While HTML generation is the layout system's primary purpose, HTML is only one third of a browser based page-or-application's appearance and behavior. The remainder of this book will be a crash course on Magento's systems for working with Cascading Style Sheets (CSS) and the javascript programming language.

...

Chapter 9

Front End Starter Kit – Javascript

It's hard to escape the fact that the modern web runs on javascript. This puts Magento in a tricky position. As a traditional PHP MVC framework, it's hard for Magento to take advantage of everything the modern javascript world has to offer. There's also parts of Magento 2 that are still running on Magento 1 code, and still require the grandparent of all javascript frameworks – PrototypeJS!

…

Chapter 10

Advanced Front End Topics

Now that you're grounded in the basics of Magento's CSS and Javascript systems, there's a few advanced topics to cover. First, we'll cover everything you can do in a layout handle XML file's `<head/>` section. Second, we'll show you how to add arbitrary code to your HTML page's head sections. Third, we'll explain how those Magento CSS and Javascript URLs are served during development mode, which should help you debug missing file problems.

...

Chapter 11

Registering Knockout.js Custom Scopes

To close out the book, we're going to take a deeper look at Magento's javascript based, front end view model system. Unfortunately, we don't have the time or space to cover every topic in the depth that we've covered Magento's layout XML. In other words, this chapter assumes some knowledge we haven't yet covered in this book. If you're stuck on a fundamental concept or specific problem remember that help is just a Stack Overflow question away.

...

Chapter 12

Appendix

Magento 2 Areas

Areas were a tricky concept to understand in Magento 1, and remain so in Magento 2. To understand areas, you need to consider **how** web applications are typically built. Consider

1. A URL
2. A web application that lives at that URL
3. A web application that has several distinct features living at sub-URLs
4. And those features each require a different set of assets and resources

...

PHP Autoloading

Autoloading is the PHP system that ensures when a developer attempts to instantiate an object from a class, that the correct class definition file is loaded.

...

The Magento Cache

The concept of a *cache* in programming is a relatively simple one. However, as time goes on, the *practice* of caching has become more and more complicated.

...

The Command Line

The "command line" is one of the oldest ways to run programs and communicate with your computer. Despite being a relic of another time, the command line remains popular for developers and other technical professionals who need programs that take clear inputs, and produce clear outputs.

...

Magento 2 Components

In Magento 1, the idea of modules, themes, libraries, and language packs all got a little blurry around the edges. Magento 2 attempts to make these distinctions a bit clearer. Specifically, while Magento 2 still has code modules, themes, code libraries, and language packs, there's a higher level idea of a Magento Component that sits above all of them.

...

Downloading URLs with curl

Every web page on the internet has a URL

```
1 http://example.com/hello.html
```

In modern times, URLs often resolve to data documents like XML or JSON files.

```
1 http://magento.example.com/foo.json
2 http://magento.example.com/foo.xml
```

Some of these may be actual documents on a server. Others may be generated on the fly by a software application (like Magento). One of the beauties of the internet is web browsers don't care how a document gets made – as long as the server speaks the *HyperText Transfer Protocol* (or HTTP), web browsers are happy to grab the document or data from the server.

As a human being, you use a program like Internet Explorer, Chrome, Firefox, Opera, Netscape, etc. to browse the web. These programs fetch the documents for you.

As a programmer, it's often useful to make an HTTP request **without** a web browser. Browsers have many layers of caching that happen automatically, some of which are hard to turn off. Web browsers also try to render documents as HTML by default, and sometimes you want to see the *raw data* returned by the server. Other times, you may be writing a software application that isn't a web browser, but still needs to use pages and data on the web. Regardless of your motivations, there are many libraries and programs for doing this. One of the most popular and longest standing is a program and library called `curl`.

The `curl` command line program is available for all popular operating systems, and many unpopular ones. It's based on a C library named `libcurl` that programmers embed in their own C based software applications. Chances are you use something that uses `libcurl` everyday.

...

Magento 2 Dependency Injection

There's no way around this: If you're going to work with Magento 2, you need to have a high level understanding of dependency injection in order to understand Magento 2's automatic constructor dependency injection system. This appendix is meant as a general introduction to dependency injection – if you don't understand everything at first, don't worry. We'll try to address specific, more complicated concerns in the text itself.

...

Front End Build System

Although we won't cover it in great detail, Magento uses a "modern" front end build system that's based on NodeJS and the `grunt` build tool. The build system is modern in the sense that it exists (vs. working with plain, non-preprocessed CSS files). However, some bleeding edge front end developers might find both

`grunt` and LessCSS beneath them. Regardless of your opinion, these are the systems stock Magento uses, so you'll need to be aware of them.

...

Installing the Pulsestorm_Nofrills Magento Module

This book includes a Magento code module (`Pulsestorm_Nofrills`) which you'll need to install.

...

Interfaces

Interfaces are a feature of the PHP programming language. They originally come from the java programming language, and fall under the broad category of "Object Oriented Programming". By and large a front end developer working with Magento doesn't need to understand interfaces, but if you want to have a better understanding of *why* certain features work a certain way understanding interfaces is important.

...

Magento Modes

Once you've installed Magento 2, there's three different *modes* you can run it in.

...

Unix Find

This book will occasionally use the venerable unix `find` command to look for files that match a particular pattern. This appendix will briefly cover `find`'s syntax in case you've never run across it before.

...

Want to read more? Get your full copy today at http://store.pulsestorm.net/

Viewing HTML Source

There will be a number of times in this book where we ask you to view the rendered HTML source of your page. For long time web developers, this is a simple, obvious task. However, I've noticed that there's a certain class of young javascript developers who are **only** aware of HTML tags as they relate to their UI abstraction (React, Knockout.js, etc.) of choice.

...

Want to read more? Get your full copy today at http://store.pulsestorm.net/