
Learning JavaScript Design Patterns

Addy Osmani

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Learning JavaScript Design Patterns

by Addy Osmani

Copyright © 2012 Addy Osmani. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://my.safaribooksonline.com>). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

Editor: Mary Treseler
Production Editor: Rachel Steely
Proofreader: Linley Dolby

Indexer: Fred Brown
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrators: Robert Romano and Rebecca Demarest

August 2012: First Edition.

Revision History for the First Edition:

2012-08-06	First release
2012-09-14	Second release
2013-03-01	Third release

See <http://oreilly.com/catalog/errata.csp?isbn=9781449331818> for release details.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. *Learning JavaScript Design Patterns*, the image of a cuckoo pheasant, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc., was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Learning JavaScript Design Patterns is released under a [Creative Commons Attribution-Noncommercial-No Derivative Works 3.0](#) unported license. It is available for purchase via [O'Reilly Media](#) but will remain available for both free online and as a physical (or eBook) purchase for readers wishing to support the project.

ISBN: 978-1-449-33181-8

[LSI]

1361833802

Table of Contents

Preface	ix
1. Introduction	1
2. What Is a Pattern?	3
We Already Use Patterns Every Day	4
3. “Pattern”-ity Testing, Proto-Patterns, and the Rule of Three	7
4. The Structure of a Design Pattern	9
5. Writing Design Patterns	11
6. Anti-Patterns	13
7. Categories of Design Patterns	15
Creational Design Patterns	15
Structural Design Patterns	16
Behavioral Design Patterns	16
8. Design Pattern Categorization	17
A Brief Note on Classes	17
9. JavaScript Design Patterns	21
The Constructor Pattern	22
Object Creation	22
Basic Constructors	24
Constructors with Prototypes	25
The Module Pattern	26
Object Literals	26

The Module Pattern	27
Module Pattern Variations	32
The Revealing Module Pattern	37
Advantages	38
Disadvantages	38
The Singleton Pattern	39
The Observer Pattern	43
Differences Between the Observer and Publish/Subscribe Pattern	47
Advantages	49
Disadvantages	50
Publish/Subscribe Implementations	50
The Mediator Pattern	60
Basic Implementation	61
Advanced Implementation	62
Example	67
Advantages and Disadvantages	69
Mediator Versus Observer	69
Mediator Versus Facade	69
The Prototype Pattern	70
The Command Pattern	73
The Facade Pattern	75
Notes on Abstraction	77
The Factory Pattern	78
When to Use the Factory Pattern	80
When Not to Use the Factory Pattern	81
Abstract Factories	81
The Mixin Pattern	82
Subclassing	82
Mixins	83
Advantages and Disadvantages	87
The Decorator Pattern	87
Pseudoclassical Decorators	91
Interfaces	91
Abstract Decorators	92
Decorators with jQuery	95
Advantages and Disadvantages	97
Flyweight	97
Using Flyweights	98
Flyweights and Sharing Data	98
Implementing Classical Flyweights	99
Converting Code to Use the Flyweight Pattern	102
A Basic Factory	104
Managing the Extrinsic States	105

10. JavaScript MV* Patterns	111
MVC	111
Smalltalk-80 MVC	111
MVC for JavaScript Developers	112
Models	113
Views	115
Controllers	117
Controllers in Another Library (Spine.js) Versus Backbone.js	118
What Does MVC Give Us?	120
Smalltalk-80 MVC in JavaScript	120
Delving Deeper	121
Summary	121
MVP	121
Models, Views, and Presenters	122
MVP or MVC?	123
MVC, MVP, and Backbone.js	123
MVVM	125
History	126
Model	126
View	127
ViewModel	129
Recap: The View and the ViewModel	131
Recap: The ViewModel and the Model	132
Pros and Cons	132
Advantages	132
Disadvantages	132
MVVM with Looser Data Bindings	133
MVC Versus MVP Versus MVVM	137
Backbone.js Versus KnockoutJS	138
11. Modern Modular JavaScript Design Patterns	139
A Note on Script Loaders	139
AMD	140
Getting Started with Modules	141
AMD Modules with Dojo	144
AMD Module Design Patterns (Dojo)	145
AMD Modules with jQuery	146
AMD Conclusions	149
CommonJS	149
Getting Started	149
Consuming Multiple Dependencies	150

Loaders and Frameworks that Support CommonJS	151
Is CommonJS Suitable for the Browser?	151
Related Reading	152
AMD and CommonJS: Competing, but Equally Valid Standards	152
UMD: AMD and CommonJS-Compatible Modules for Plug-ins	152
ES Harmony	157
Modules with Imports and Exports	158
Modules Loaded from Remote Sources	159
Module Loader API	159
CommonJS-like Modules for the Server	159
Classes with Constructors, Getters, and Setters	160
ES Harmony Conclusions	161
Related Reading	161
Conclusions	161
12. Design Patterns in jQuery	163
The Composite Pattern	163
The Adapter Pattern	164
The Facade Pattern	166
The Observer Pattern	168
The Iterator Pattern	171
Lazy Initialization	172
The Proxy Pattern	173
The Builder Pattern	175
13. jQuery Plug-in Design Patterns	177
Patterns	178
A Lightweight Start Pattern	179
Complete Widget Factory Pattern	181
Nested Namespacing Plug-in Pattern	183
Custom Events Plug-in Pattern (with the Widget Factory)	185
Prototypal Inheritance with the DOM-to-Object Bridge Pattern	186
jQuery UI Widget Factory Bridge Pattern	188
jQuery Mobile Widgets with the Widget Factory	190
RequireJS and the jQuery UI Widget Factory	193
Usage	196
Globally and Per-Call Overridable Options (Best Options Pattern)	196
A Highly Configurable and Mutable Plug-in Pattern	198
What Makes a Good Plug-in Beyond Patterns?	200
Quality	200
Code Style	200
Compatibility	200
Reliability	201

Performance	201
Documentation	201
Likelihood of maintenance	201
Conclusions	201
Namespacing Patterns	202
Namespacing Fundamentals	202
Single Global Variables	203
Prefix Namespacing	203
Object Literal Notation	203
Nested Namespacing	207
Immediately Invoked Function Expressions (IIFE)s	208
Namespace Injection	210
Advanced Namespacing Patterns	212
Automating Nested Namespacing	212
Dependency Declaration Pattern	214
Deep Object Extension	215
Recommendation	218
14. Conclusions	219
Appendix: References	221
Index	223