Front-end style guides are a modular collection of all the elements in your product’s user interface, together with code snippets for developers to copy and paste as needed to implement those elements. They include common UI components like buttons, form-input elements, navigation menus, modal overlays, and icons.

– Lean UX
ZOMBIE PATTERN LIBRARIES
“How do you keep living, knowing that’s what the world is like?”
RETHINKING THE DESIGN PROCESS
BUILD SYSTEMS NOT PAGES

-Andy Clarke

http://stuffandnonsense.co.uk/blog/about/walls_come_tumbling_down_presentation_slides_and_transcript/
I just joined the [redacted] team and found out about the UX site. I’m crying over here loving it. I tried to get [redacted] and other clients to see the same value. This coupled with the user stories makes the process much easier (less painful, more productive, expected, happier, etc.).

Nice job,

[redacted]
BUILD A TINY BOOTSTRAP

– Mark Otto & Dave Rupert
DESIGNING WITH PATTERNS
DESIGNING WITH PATTERNS

DESIGN

SLICE
DESIGNING WITH PATTERNS

DESIGN

SLICE

ASSEMBLE
DESIGN TOOLS
WHERE THE DESIGN PROCESS BREAKS DOWN
DESIGNERS

DESIGN

DEVELOPERS

DEVELOP

TEST
TRADITIONAL HANDOFF IS BROKEN
DESIGNING WITH PATTERNS

DESIGN  PATTERN LIBRARY  ASSEMBLE
DETAILS PANEL

Example

PIN Settings

**PIN Number** Required

Ex: 328491239

Min. Length 4; Max. 20

**PIN Generation**

Generate

**PIN append length**

Append

Code

```html
<ul class="nav nav-tabs call-details-tabs" style="width:300px">
  <li class="active"><a href="#pin-settings" data-toggle="tab">PIN</a></li>
  <li><a href="#call-settings" data-toggle="tab">Call Settings</a></li>
</ul>

<div class="tab-content" style="width:400px">
  <div role="tabpanel" class="tab-pane active" id="pin-settings">
    <div class="call-settings-call-form">
      <h4 class="settings-headline">
        <a data-toggle="collapse" href="#inmate-pin-details" role="button">
          ADD TITLE
          <div class="pull-right">
            <span class="toggle-hide"></span>
            <span class="toggle-show"></span>
          </div>
        </a>
      </h4>
      <!-- More code here -->
    </div>
  </div>
</div>
```
ELIMINATE WASTE
THE EVOLVING DESIGNER
"I can shift the bulk of my time and energy to looking at the bigger picture. I can think more widely about our users’ journey through our product and give attention to more than baseline functionality."

– Katey Basye (Salesforce UX)

https://medium.com/salesforce-ux/how-the-salesforce-design-system-helped-me-onboard-as-a-new-product-designer-92b7d5f42237#6brzoe1fu
WHERE DOES THE PATTERN LIBRARY LIVE IN YOUR PROCESS?
**STYLE GUIDE**

**MATURITY MODEL**

- **Governed**
  The pattern library process is built in to your organization

- **Automatic**
  The pattern library is a part of your app build process

- **Manual**
  The pattern library has code, but must be manually updated

- **Static**
  A one-time PDF of your brand guidelines

- **Inconsistent**
  The absence of a design system

This is the “chasm” that is hard for organizations to cross

[https://medium.com/@marcelosomers/a-maturity-model-for-design-systems-93fff522c3ba](https://medium.com/@marcelosomers/a-maturity-model-for-design-systems-93fff522c3ba)
UNLESS IT’S PART OF YOUR BUILD, YOUR STYLEGUIDE IS JUST MORE DOCUMENTATION TO MAINTAIN

Static sites go all Hollywood by Phil Hawksworth
Published September 22, 2015 in Programming

The popularity of building web sites with static site generators is on the rise. Their reduced complexity, easier compliance, cheaper hosting, and other benefits are getting people's attention, but they do have limits.

This talk will explore how we can break through some of those limits with the use of a new breed of
Chasing the Holy Grail
Strategies For Distributing Your Pattern Library and Keeping It in Sync

In our work with Pattern Libraries, we strive to achieve a level of maturity where the Pattern Library documentation is automatically in sync with the app(s) that it powers. Generally, this means sharing the CSS code so developers implementing patterns can just copy and paste snippets of HTML so it “just works.”
GETTING STARTED (TODAY)
1. TAKE AN INVENTORY
Option List Group

Seen On:
- My Account » Account List
- Send Money » Select Payment Option
The Component Cut-Up Workshop
Kickoff a Design Library Effort by Engaging the Whole Team
WHAT TO DOCUMENT

1. BASE STYLES
2. COMPONENTS
3. PAGE TEMPLATES
Front-End Style-Guides: Definition, Requirements, Component Checklist

by PAGE LAUBHEIMER on March 27, 2016
Topics: Agile  Visual Design

Summary: Front-end style-guides help efficient design and testing, and enforce UI consistency. We present 8 style-guide requirements and 25 common components.

What Is A Front-End Style Guide?
Front-end style guides have become an increasingly commonplace deliverable in UX practice. As described by Jeff Gothelf and Josh Seiden in their book Lean UX, they originated in Agile or Lean environments.

Definition: Front-end style guides are a modular collection of all the elements in your product’s user interface, together with code snippets for developers to copy and paste as needed to implement those elements. They include common UI components like buttons, form-input elements, navigation menus, modal overlays, and icons.

A front-end style guide is both a deliverable created by the UX team (in concert with the engineering team, typically) and a tool used by the entire team for maintaining consistent, nimble product design in a modular format.
Front-end style guides are distinct from design pattern libraries, which are a long-standing tool used by UX practitioners to define broad design ideas, rather than specific implementation details. Unfortunately, some proponents of front-end style guides refer to them as pattern libraries, which has caused some confusion in the UX community.

https://www.nngroup.com/articles/front-end-style-guides/
2 STANDARDIZE
Tablet
10

Page group

Core - page - header

Core - page - content

Core - page - actions
3 DOCUMENT
Create a button with a **button** or a `a` element to retain the native click function. Use a disabled attribute when a button can’t be clicked.

**ACCESSIBILITY**

If an icon button doesn’t include a label, use a span with `.slds-assistive-text` to describe the icon for screen readers.

**Base**  
DEV READY

The base `.slds-button` looks like a plain text link. It removes all the styling of the native button. It’s typically used to trigger a modal or display a “like” link. All button variations are built by adding another class to `.slds-button`.

```
<button class="slds-button">Button</button>
```
Create a button with a `button` or an `a` element to retain the native click function. Use a disabled attribute when a button can’t be clicked.

**ACCESSIBILITY**

If an icon button doesn’t include a label, use a `span` with `.slds-assistive-text` to describe the icon for screen readers.

**Base**  
`DEV READY`

The base `.slds-button` looks like a plain text link. It removes all the styling of the native button. It’s typically used to trigger a modal or display a “like” link. All button variations are built by adding another class to `.slds-button`.

```html
<button class="slds-button">Button</button>
```
Create a button with a `<button>` or an `a` element to retain the native click function. Use a disabled attribute when a button can’t be clicked.
Create a button with a **button** or an **a** element to retain the native click function. Use a disabled attribute when a button can’t be clicked.

**Neutral**

Add the `.slds-button--neutral` class to create a neutral button, which has a white background and gray border.

---

**LARGE**

- Button neutral
- Anchor Button Neutral
- Disabled
- Button Small
Create a button with a `button` or an `a` element to retain the native click function. Use a disabled attribute when a button can't be clicked.

**ACCESSIBILITY**

If an icon button doesn't include a label, use a `span` with `.slds-assistive-text` to describe the icon for screen readers.

**Base**  
DEV READY

The base `.slds-button` looks like a plain text link. It removes all the styling of the native button. It's typically used to trigger a modal or display a “like” link. All button variations are built by adding another class to `.slds-button`.

```
<Button class="slds-button">Button</Button>
```
ACCESSIBILITY

If an icon button doesn't include a label, use a span with `.slds-assistive-text` to describe the icon for screen readers.
Create a button with a `button` or an element to retain the native click function. Use a class attribute when a button can’t be clicked.

**ACCESSIBILITY**

If an icon button doesn’t include a label, use a span with `.slds-assistive-text` to describe the icon for screen readers.

**Base**

The base `.slds-button` looks like a plain text link. It removes all the styling of the native button. It’s typically used to trigger a modal or display a “like” link. All button variations are built by adding another class to `.slds-button`.

```html
<button class="slds-button">Button</button>
```
BASIC DOCUMENTATION

1. NAME
2. DESCRIPTION
3. EXAMPLE
3. CODE SNIPPET
CENTRALIZE YOUR CSS
4. DEFINE CSS STANDARDS

5. REFACTOR TO PERFECTION
START FUNCTIONAL

Basscss v8.0.1
Low-Level CSS Toolkit 2.21 KB

Getting Started
Type Grid

Lightning Fast Manual
Code with Confidence
Using clear, humanized naming conventions, Basscss makes quick to internalize and easy to read. Speed up development time with more readable code.

Designed for Design
Basscss strikes a balance between compression and flexibility to allow for rapid prototyping and iterative changes when designing interfaces.

Tachyons v4.0.0-beta.16

Tachyons was built for designing.
Create fast loading, highly readable, and 100% responsive interfaces with as little css as possible.

Modules can be altered, extended, or changed to meet your design needs. You shouldn't need to write css everytime you want to build a new UI component. By learning the composable building blocks in tachyons, you can quickly start to build out interfaces while writing little to no css.

Getting Started
Copy the line of code below and paste it in the head of the html file(s) you want to include tachyons in.

```html
<link rel="stylesheet" href="https://npmcdn.com/tachyons@4.0.0-beta.12/css/tachyons.min.css" />
```

or install via npm
```
npm install --save-dev tachyons@4.0.0-beta.12
```

or grab all the source files and build locally
```
git clone git@github.com:tachyons-css/tachyons.git
```

BREAK UP YOUR COMPONENTS

https://medium.com/@ahmedelgabri/global-scope-namespacing-css-681bda44c43e#.5i17q1gwv
NAMESPACE
THE CSS

https://medium.com/@ahmedelgabri/global-scope-namespacing-css-681bda44c43e#.5i17q1gwv
FIND IN PROJECT IS YOUR FRIEND
REGEX SEARCH FOR CLASSES

class\s*?=\s*?"\".*?table

will find all instances where class="" contains table
DON’T FORGET JAVASCRIPT!
GOVERN YOUR LIBRARY
What a CSS Code Review Might Look Like

Many programming languages go through a code review before deployment. Whether it's a quick once-over, in-depth peer review, or complete unit testing, code reviews help us release code into the wild with confidence.

I started to imagine what a CSS code review might look like. CSS can be written in a number of ways, and the best way is often subjective to the project. I'm definitely not trying to get dogmatic with a post like this, but instead lay the foundation for what could be a starting point for getting the most out of CSS before it is released.

Why should CSS be reviewed at all?

It's fair to wonder why we would want to review CSS in the first place. A review is yet
Team Models for Scaling a Design System
Evolving Past Overlords to Centralize or Federate Design Decision-Making Across Platforms

https://medium.com/eightshapes-llc/team-models-for-scaling-a-design-system
SOLITARY
CENTRALIZED
FEDERATED
Effective organizational design is more than drawing boxes and arrows, it flows from a thorough understanding of strategic objectives, takes into consideration both the formal and informal elements of the organizational units involved in the redesign and weighs the potential impact on the rest of the organization and its ability to fulfill strategic goals.

– Competing by Design, Nadler and Tushman
Effective organizational design is more than drawing boxes and arrows, it **flows from a thorough understanding of strategic objectives**, takes into consideration both the formal and informal elements of the organizational units involved in the redesign and weighs the potential impact on the rest of the organization and its ability to fulfill strategic goals.

– Competing by Design, Nadler and Tushman
Effective organizational design is more than drawing boxes and arrows, it flows from a thorough understanding of strategic objectives, takes into consideration both the formal and informal elements of the organizational units involved in the redesign and weighs the potential impact on the rest of the organization and its ability to fulfill strategic goals.

– Competing by Design, Nadler and Tushman
Effective organizational design is more than drawing boxes and arrows, it flows from a thorough understanding of strategic objectives, takes into consideration both the formal and informal elements of the organizational units involved in the redesign and weighs the potential impact on the rest of the organization and its ability to fulfill strategic goals.

– Competing by Design, Nadler and Tushman
1. WHY ARE WE DOING THIS?
2. HOW WILL IT FIT IN OUR ORGANIZATION?
3. WHAT INVESTMENT ARE WE WILLING TO MAKE?
4. WHO’S DOING THE WORK?
OPEN SOURCE CULTURE
GREAT DOCUMENTATION
MAKE IT EASY TO CONTRIBUTE
FIND EASY MISTAKES
EMBRACE NON-DEVELOPERS
MAKE IT EASY TO COMMUNICATE
OPEN & TRANSPARENT
ROADMAP
1 CSS DOCUMENTATION
Knyle Style Sheets

Documentation for any flavor of CSS that you’ll love to write. Human readable, machine parseable, and easy to remember.

Works great with CSS, SCSS, LESS, and much more.

// A button suitable for giving a star to someone.
//
// :hover
// .star-given
// .star-given:hover
// .disabled
//
// Styleguide 2.1.3.
a.button.star{
    ...
    &.star-given{
        ...
    }
    &.disabled{
        ...
    }
}

Documentation for humans

Works with SCSS, LESS, and more

Automatically generate styleguides

Documentation is all about communication. Between people, not computers. So why should your documentation format cater to computers?

KSS is designed to work with every flavor of CSS out there — preprocessor or not. Choose whatever works for you.

Create example HTML for your CSS and automatically generate variations of each element.
2 STATIC SITE GENERATION
3 INTEGRATED
Source
Living Style Guide Platform

The most advanced tool for documenting, testing and managing
UI components achieving productive team work

Quick Start
THERE HAS TO BE A BETTER WAY
GET STARTED WITH PATTERN PACK
What are pattern libraries?

A pattern library is a collection of user interface design patterns. It breaks down the overall design into element that solve common design problems and can be reused consistently.

A pattern library defines not only how an element of an application looks, but also provides simple examples of how each piece should be implemented. This allows designers to give examples of how each element looks and feels, while providing developers with tools to bring the design to life.

Atomic design principles

The atomic design principles outlined by Brad Frost have served as a significant influence to this pattern library. The concept of breaking down a design into its components (atoms, molecules, templates & pages) lends itself very well to the goals of a pattern library and leveraged heavily throughout.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atom</td>
<td>A component of your design that will not be broken down into smaller pieces. (button, text input, header, etc.)</td>
</tr>
</tbody>
</table>

https://github.com/patternpack/patternpack-example-library
$ npm init
$ git init
$ npm install grunt patternpack --save-dev
module.exports = function(grunt) {
  grunt.initConfig({
    patternpack: {
      run: {},
      build: {},
      release: {}
    }
  });

  grunt.loadNpmTasks('patternpack');

  grunt.registerTask('default', ['patternpack:run']);
}
$ grunt patternpack:run
Welcome to PatternPack
CREATE YOUR FIRST PATTERN
button {
  padding: 5px 20px;
  border: 0;
  border-radius: 5px;
  color: white;
}

.button--primary {
  background-color: #000;
}

.button--secondary {
  background-color: #ccc;
}

.button--small {
  padding: 2px 20px;
  height: 30px;
}

.button--large {
  height: 40px;
}
button {
  padding: 5px 20px;
  border: 0;
  border-radius: 5px;
  color: white;
}

.button--primary {
  background-color: #000;
}

.button--secondary {
  background-color: #ccc;
}

.button--small {
  padding: 2px 20px;
  height: 30px;
}

.button--large {
  height: 40px;
}
button {
  padding: 5px 20px;
  border: 0;
  border-radius: 5px;
  color: white;
}

.button--primary {
  background-color: #000;
}

.button--secondary {
  background-color: #CCC;
}

.button--small {
  padding: 2px 20px;
  height: 30px;
}

.button--large {
  height: 40px;
}
Title: Buttons

## Buttons

TODO: Describe what buttons are and how they are used in the application.

### Examples

```html
<div class="library_example">
  <button type="primary" class="button--primary">Primary</button>
  <button type="secondary" class="button--secondary">Secondary</button>
  <button type="primary" class="button--primary button--large">Primary Large</button>
  <button type="secondary" class="button--secondary button--large">Secondary Large</button>
  <button type="primary" class="button--primary button--small">Primary Small</button>
  <button type="secondary" class="button--secondary button--small">Secondary Small</button>
</div>
```

### Code

```html
  <button type="primary" class="button--primary">Primary</button>
  <button type="secondary" class="button--secondary">Secondary</button>
  <button type="primary" class="button--primary button--large">Primary Large</button>
  <button type="secondary" class="button--secondary button--large">Secondary Large</button>
  <button type="primary" class="button--primary button--small">Primary Small</button>
  <button type="secondary" class="button--secondary button--small">Secondary Small</button>
```
title: Buttons

## Buttons

TODO: Describe what buttons are and how they are used in the application.

```html
<br />
<button type="button" class="button--primary button--large">Primary Large</button>
<br />
<button type="button" class="button--secondary button--large">Secondary Large</button>
<br />
<button type="button" class="button--primary button--small">Primary Small</button>
<br />
<button type="button" class="button--secondary button--small">Secondary Small</button>
</div>

```
### Examples

```html
<div class="library_example">
  <button type="button" class="button--primary">Primary</button><br />
  <button type="button" class="button--secondary">Secondary</button><br />
  <button type="button" class="button--primary button--large">Primary Large</button>
  <button type="button" class="button--secondary button--large">Secondary Large</button><br />
  <button type="button" class="button--primary button--small">Primary Small</button>
  <button type="button" class="button--secondary button--small">Secondary Small</button>
</div>
```
```html
  <button type="button" class="button--primary">Primary</button>
  <button type="button" class="button--secondary">Secondary</button>
  <button type="button" class="button--primary button--large">Primary Large</button>
  <button type="button" class="button--secondary button--large">Secondary Large</button>
  <button type="button" class="button--primary button--small">Primary Small</button>
  <button type="button" class="button--secondary button--small">Secondary Small</button>
```
title: Buttons

## Buttons

TODO: Describe what buttons are and how they are used in the application.

### Examples

```
<div class="library_example">
  <button type="button" class="button--primary">Primary</button>
  <button type="button" class="button--secondary">Secondary</button>
  <button type="button" class="button--primary button--large">Primary Large</button>
  <button type="button" class="button--secondary button--large">Secondary Large</button>
  <button type="button" class="button--primary button--small">Primary Small</button>
  <button type="button" class="button--secondary button--small">Secondary Small</button>
</div>
```

### Code

```
```

```
Buttons

TODO: Describe what buttons are and how they are used in the application.

Examples

- Primary
- Secondary
- Primary Large
- Secondary Large
- Primary Small
- Secondary Small

Code

```html
<button type="button" class="button--primary">Primary</button>
<button type="button" class="button--secondary">Secondary</button>
```

UNLEASH YOUR CREATION
USE SEMANTIC VERSIONING FOR YOUR DESIGN SYSTEM
1.0.5

major release

minor release

patch release
Customize to your needs

1.0.5

- major release
- minor release
- patch release
$ grunt patternpack:release
$ git push --follow-tags
1. Build your static site
2. Increment your version
3. Create a new commit
4. Tag the commit
INTEGRATE INTO YOUR APPLICATION
Chasing the Holy Grail
Strategies For Distributing Your Pattern Library and Keeping It in Sync

In our work with Pattern Libraries, we strive to achieve a level of maturity where the Pattern Library documentation is automatically in sync with the app(s) that it powers. Generally, this means sharing the CSS code so developers implementing patterns can just copy and paste snippets of HTML so it “just works.”
SUPPORT FOR
NPM &
BO ther
$ npm install my-awesome-pattern-library
$ npm install https://user:pass@github.com/user/my-awesome-library.git
ANATOMY OF A PACKAGE.JSON DEPENDENCY

"devDependencies": {
  "my-awesome-pattern-library": "https://user:pass@bitbucket.org/jondoe/my-awesome-pattern-library.git#1.0.0"
}
ANATOMY OF A PACKAGE.JSON DEPENDENCY

"devDependencies": {
  "my-awesome-pattern-library": "https://user:pass@bitbucket.org/jondoe/my-awesome-pattern-library.git#1.0.0"
}

USERNAME & PASSWORD
ANATOMY OF A PACKAGE.JSON DEPENDENCY

"devDependencies": {
  "my-awesome-pattern-library": "https://user:pass@bitbucket.org/jondoe/my-awesome-pattern-library.git#1.0.0"
}

PATH TO GIT REPOSITORY
ANATOMY OF A PACKAGE.JSON DEPENDENCY

"devDependencies": {
  "my-awesome-pattern-library": "https://user:pass@bitbucket.org/jondoe/my-awesome-pattern-library.git#1.0.0"
}

VERSION
COMMITHASH
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Example Application</title>
  <link href="/node_modules/patternpack-example-library/dist/pattern-library/assets/css/patterns.css"
</head>
<body>
</body>
</html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example Application</title>
</head>
<body>
    <link rel="stylesheet" href="/node_modules/patternpack-example-library/dist/pattern-library/assets/css/patterns.css"/>
</body>
</html>
<link href="/node_modules/patternpack-example-library/dist/pattern-library/assets/css/patterns.css"
Build and Document Your Interface. Then Share the Code.

Get Started

http://patternpack.org/
THIS IS MORE THAN JUST A TECH PROBLEM
MODULARIZE
THANK YOU!

Marcelo Somers
@marcelosomers

Slides
http://j.mp/zombie-library-rwdsummit

PatternPack
http://patternpack.org/