OddBird Web Consulting

Custom Apps, Design Systems, Refactors - since 2008
OddBird siblings (Jonny, Carl, and Miriam) in 1994
Style with Ease

With the Lightning Design System you can build custom applications with a look and feel that is consistent with Salesforce core features – without reverse engineering our styles! Simply download our platform-agnostic CSS framework and get started today.

Design with Expertise

Utilize our detailed guidelines to confidently design excellent apps that fit right into the Salesforce ecosystem. With the Design System, you get access to all of the Salesforce organization's Interaction Design patterns and that
Colour Palette

This is the primary and secondary colour palettes as described in the PSD styleguide and in `colour_palette.sass`.

**GRAY PALETTE**

- #2c3643
- #3b444f
- #67747c
- #99a9b3
- #d0e9ec

**PRIMARY PALETTE**

- #142b44
- #1d508d
- #297cbb
- #38ad66
- #6fdebd
- #16c98d
- #feef6d
- #ffe33f
- #fa5e5b
- #bf538d
- #15c98d
- #feef6d
- #ffe33f
- #fa5e5b
- #bf538d
- #15c98d
- #feef6d
- #ffe33f
- #fa5e5b
- #bf538d

**SECONDARY PALETTE**

- #684e79
- #7f08e
- #47a899
- #8abee5
- #c7e6aa
- #a5c9de
- #582c2b
- #841e1b
- #5e342b
- #3e079c
- #6a519f
- #832f6c
- #722f66
- #6a519f
- #832f6c
- #722f66
- #6a519f
- #832f6c
- #722f66
Grid System

Our grid system is composed of 8 flexible columns with a gutter between columns of 30px. We apply border-box so that the border and padding is included in the width of the grid columns.

**Grid sizes**

Size 1 of 1

Size 1 of 2

Size 1 of 3
<table>
<thead>
<tr>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>rhythm, gutter</td>
<td>1.4rem</td>
</tr>
<tr>
<td>gutter-plus</td>
<td>2.1rem</td>
</tr>
<tr>
<td>double-gutter</td>
<td>2.8rem</td>
</tr>
<tr>
<td>flex-gutter</td>
<td><code>calc(0.7rem + 2.5vw)</code></td>
</tr>
<tr>
<td>spacer</td>
<td><code>calc(4.2rem + 2.5vw)</code></td>
</tr>
<tr>
<td>gutter-minus</td>
<td>1.05rem</td>
</tr>
<tr>
<td>shim</td>
<td>0.7rem</td>
</tr>
<tr>
<td>half-shim</td>
<td>0.35rem</td>
</tr>
<tr>
<td>quarter-shim</td>
<td>0.175rem</td>
</tr>
</tbody>
</table>
Graphics Standards Manual
1970
(Typical)
Steel Enamed Sign Details (Type "D" Hanging Sign)

Elevation

Section

Detail of Bolt Slots
at top of Pans

Detail of Headrail

---

Sign length varies—see sign schedule

---

Note: Where headroom is insufficient for use of hangers, mount headrail directly to ceiling.
Nerd
Nerd
Design Systems

Style Guides + Tool Kits + Component Libraries + ???
Single Source of Truth
Potentially **Cross-Application**

even cross-platform...
Introduction

Get started with Bootstrap, the world's most popular framework for building responsive, mobile-first sites, with BootstrapCDN and a template starter page.

Quick start

Looking to quickly add Bootstrap to your project? Use BootstrapCDN, provided for free by the folks at StackPath. Using a package manager or need to download the source files? Head to the downloads page.

CSS

Copy-paste the stylesheet `<link>` into your `<head>` before all other stylesheets to load our CSS.

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-gQy9Xl2tdxaKHIq/CJmBxXUfXTHZu5O9w1ixF7nj3RF0xK5lAOsAtvPzj46+tCJ..."
```

JS

Many of our components require the use of JavaScript to function. Specifically, they require jQuery, Popper.js, and our own JavaScript plugins. Place the following `<script>`s near the end of your pages, right before the closing `<body>` tag, to enable them. jQuery must come first, then Popper.js, and then our JavaScript plugins.

We use jQuery's slim build, but the full version is also supported.

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/Xr+ friendly to humans. Get reading for an example page template.

```

Curious which components explicitly require jQuery, our JS, and Popper.js? Click the show components link below. If you're at all unsure about the general page structure, keep reading for an example page template.

Your Own Personal Bootstrap
Voice and Tone

One way we write empowering content is by being aware of our voice and our tone. This section explains the difference between voice and tone, and lays out the elements of each as they apply to MailChimp. To learn more about how we adapt our tone for different situations, check out our Voice and Tone guide.

What’s the difference between voice and tone? Think of it this way: You have the same voice all the time, but your tone changes. You might use one tone when you’re out to dinner with your closest friends, and a different tone when you’re in a meeting with your boss.

Your tone also changes depending on the emotional state of the person you’re addressing. You wouldn’t want to use the same tone of voice with someone who’s scared or upset as you would with someone who’s laughing.

The same is true for MailChimp. Our voice doesn’t change much from day to day, but our tone changes all the time.

Voice

MailChimp's voice is human. It's familiar, friendly, and straightforward. Our priority is explaining our products and helping our users get their work done so they can get on with their lives. We want to educate people without patronizing or confusing them.

One way to think of our voice is to compare what it is to what it isn’t.
Key Principles

**TIMING**
A "grid of time" creates consistent rhythm. Animation is defined as multiples of a base grid.

**ELEVATION**
Utilizing the Z-axis to support spatial organization aids in creating a visual hierarchy we refer to as "atmosphere".

**EFFECTS**
Attributes are animated over time to create a vocabulary of effects.

Type of Animation
Design Systems = Integration

Design & Code & Process & Tools & ... & ??
Icons provide visual context and enhance usability. Looking for the icons? Go to Resources › Icons.

**Base**

```html
<span class="slds-icon-container slds-icon-standard-account" title="description of icon when needed">
<svg aria-hidden="true" class="slds-icon">
  <use xlink:href="/assets/icons/standard-sprite/svg/symbols.svg#account"></use>
</svg>
</span>
```

- For **action** icons, add the `slds-icon-container--circle` class to the container `.slds-icon-container`, which changes the background shape to a circle.
- **Doctype** icons have no background color.
- **Standard** and **Custom** icons have a rounded square shape and use a class on the container for the background color.
- **Utility** icons have no background color.

**Icon Colors**
You are not **MailForce** or **InstaFace**

(unless you are)
Similar Problems

Unique Constraints
(Team Size)
(Team Structure)
(Internal vs. Consulting)
(Web vs. OS-Specific)
(Is it 1970?)
Web Consulting
Designers + Developers + ???
“How do you sell it to clients (or the boss)?
(We Don’t)
“Since OddBird thinks about handoff from the beginning, maintenance has been super easy. For example, 100% unit test coverage was a given. I never had to ask for it.

—Sara Taillon, CTO at ORCAS
“If it doesn’t have tests, it’s legacy code”
—Carl Meyer
“If it’s not documented, it doesn’t exist

—Me
“If it’s not accessible, it doesn’t exist

—Also Me
Accessibility is a Basic Human Right

UN Convention on the Rights of Persons with Disabilities
Perceivable, Operable, Understandable, Robust
Accessibility is **For Everyone**

long-term, temporary, situational, or robotic...
The Web is...

Accessible by Design
The Web... Has a Mission
“Web for all.
Web on everything.

WORLD WIDE WEB

The WorldWideWeb (W3) is a wide-area hypermedia[1] information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary[2] of the project, Mailing lists[3], Policy[4], November’s W3 news[5], Frequently Asked Questions[6].

What’s out there?[7] Pointers to the world’s online information, subjects[8], W3 servers[9], etc.

Help[10] on the browser you are using

Software A list of W3 project components and their current state. (e.g. Line Mode[12], X11 Viola[13], NeXTStep[14], Servers[15], Tools[16], Mail robot[17], Library[18])

Technical[19] Details of protocols, forms, program internals etc
The Web is...

Contextual by Design
The Web is...

Resilient by Design
The Web is...

Customizable by Design
The Web Model

Is **User-Controlled** By Design
HTML/CSS are **Declarative**

Too Many **Variables**
“The fact we can control a paper page is really a limitation of that medium.”

Accessible = User Friendly
Use **Built-In Tools**

<button>
  Do a thing.
</button>
Fakes Are **Not Robust**

```html
<div onclick="DoThing();" onkeypress="DoThing();"
tabindex="0"
role="button">
  Do a thing.
</div>
```
Provide Accessible Patterns
1. Why Design Systems?

2. Pattern Audit

3. Start Small

4. Define The API

5. Integrated Agile Process
Existing Apps: What Do We Have?
### 24 Unique Colors

<table>
<thead>
<tr>
<th>Color Code</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>#fff</td>
<td>White</td>
</tr>
<tr>
<td>#999</td>
<td>Gray</td>
</tr>
<tr>
<td>#525252</td>
<td>Gray</td>
</tr>
<tr>
<td>#0092db</td>
<td>Purple</td>
</tr>
<tr>
<td>#e96900</td>
<td>Orange</td>
</tr>
<tr>
<td>#ae81ff</td>
<td>Purple</td>
</tr>
<tr>
<td>#2973b7</td>
<td>Gray</td>
</tr>
<tr>
<td>#42b983</td>
<td>Gray</td>
</tr>
<tr>
<td>#a6e22e</td>
<td>Green</td>
</tr>
<tr>
<td>#b33b3b</td>
<td>Gray</td>
</tr>
<tr>
<td>#7f8c8d</td>
<td>Gray</td>
</tr>
<tr>
<td>#34495e</td>
<td>Gray</td>
</tr>
<tr>
<td>#aaa</td>
<td>Gray</td>
</tr>
<tr>
<td>#666</td>
<td>Gray</td>
</tr>
<tr>
<td>#1c90f3</td>
<td>Purple</td>
</tr>
<tr>
<td>#2c3e50</td>
<td>Purple</td>
</tr>
<tr>
<td>#ccc</td>
<td>Gray</td>
</tr>
<tr>
<td>#3a5169</td>
<td>Gray</td>
</tr>
<tr>
<td>#000</td>
<td>Black</td>
</tr>
<tr>
<td>#2c815b</td>
<td>Purple</td>
</tr>
<tr>
<td>#333</td>
<td>Gray</td>
</tr>
<tr>
<td>#0064e1</td>
<td>Purple</td>
</tr>
<tr>
<td>inherit</td>
<td>Phi</td>
</tr>
<tr>
<td>#555</td>
<td>Gray</td>
</tr>
</tbody>
</table>

### 8 Unique Background Colors

<table>
<thead>
<tr>
<th>Background Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>White</td>
</tr>
<tr>
<td>Gray</td>
</tr>
<tr>
<td>Green</td>
</tr>
<tr>
<td>Purple</td>
</tr>
<tr>
<td>Gray</td>
</tr>
<tr>
<td>Purple</td>
</tr>
<tr>
<td>Gray</td>
</tr>
<tr>
<td>Gray</td>
</tr>
</tbody>
</table>

*color stats from the VueConf website*
Font Size 2.4em
Font Size 2em
Font Size 1.5em
Font Size 1.333333333em
Font Size 1.2em
Font Size 1.05em
Font Size inherit
Font Size 16px
Font Size 1em
Font Size 15px
Font Size 0.9em
Font Size .9em
Font Size 0.9rem
Font Size 14px
Font Size 0.85em

font-size stats from the VueConf website
Take Screenshots

Organize into Groups
All Apps:

What Do We Need?
1. **Why** Design Systems?
2. Pattern **Audit**

3. Start **Small**

4. Define The **API**

5. **Integrated** Agile Process
Up-To-Date > Comprehensive

Start small, \textit{get it right}, and expand...
Design Tokens

Abstract: broad reach, low specificity
# Color Tokens

## Color Previews

<table>
<thead>
<tr>
<th>text-default</th>
<th>text-warning</th>
<th>text-error</th>
</tr>
</thead>
<tbody>
<tr>
<td>#3e3e3c</td>
<td>#ff75d</td>
<td>#23934</td>
</tr>
<tr>
<td>rgb(62, 62, 60)</td>
<td>rgb(255, 183, 93)</td>
<td>rgb(194, 57, 52)</td>
</tr>
<tr>
<td>hsl(60, 2%, 24%)</td>
<td>hsl(33, 100%, 68%)</td>
<td>hsl(2, 58%, 48%)</td>
</tr>
</tbody>
</table>
Inverted Triangle by Harry Roberts
Cascading Style Sheets
Invented to create re-usable patterns
.warning { color: red; }
Sass Variables

$color-text-default: rgb(62, 62, 60);
$color-text-warning: rgb(255, 183, 93);
$color-text-error: rgb(194, 57, 52);

.example {
    background: $color-text-error;
}
Not Meaningfully Organized
Encourages One-Offs
Difficult to Automate
YAML & Theo

props:
  text_default:
    value: "rgb(62, 62, 60)"
  text_warning:
    value: "rgb(255, 183, 93)"
  text_error:
    value: "rgb(194, 57, 52)"

global:
  type: "color"
  category: "brand-colors"
aliases:
  vermilion:
    value: "7, 83%"

props:
  color_vermilion:
    value: "hsla({!vermilion}, 53%, 1)"
  color_vermilion_dark:
    value: "hsla({!vermilion}, 43%, 1)"
Easier to **Export** from YAML

Not A **Style Language**
Sass Maps

```
$text-colors: (
  'default': rgb(62, 62, 60),
  'warning': rgb(255, 183, 93),
  'error': rgb(194, 57, 52),
);

.example {
  background: color('error');
}
```
The Map Problem

```
$colors: {
    'brand-blue': hsl(195, 85%, 35%),
    'gray': desaturate(map-get($colors, 'brand-blue'), 80%),
};
```

[ERROR] Undefined variable: "$colors".
Map Self-Reference

```
$colors: (  
  '_brand-pink': hsl(330, 85%, 68%),
  'escher': '#_brand-pink',
  'godel': '#escher',
  'bach': '#godel',
  'kevin bacon': '#bach' ('lighten': 20%),
);
```
color('kevin bacon')
Code is Communication
```css
code {
    background: #d4e0e4;
    border: #92b1bc;
    color: #13323c;
}
```
code {
    background: mix($brand, #fff, 80%);
    border: mix($brand, #fff, 50%);
    color: mix($brand, #000, 50%);
}
Code Patterns Add Meaning

code {
    background: color('callout');
    border: color('border');
    color: contrast('callout');
}
Meaningful Relationships!

Encourages Patterns

Functional Adjustments
Requires Additional Tooling
Even More Layers

Global Settings » Theme Defaults » Component Details
$brand-colors: (  
  '_brand-blue': hsl(195, 85%, 35%),  
  '_brand-pink': hsl(330, 85%, 48%),  
);  

$theme-colors: (  
  'text': ...,  
  'border': ...,  
  // ...  
);  

$button-colors: (  
  'button-text': ...,  
  'button-border': ...,  
  // ...  
);
CSS Custom Properties

:root {
  --color-text-default: rgb(62, 62, 60);
  --color-text-warning: rgb(255, 183, 93);
  --color-text-error: rgb(194, 57, 52);
}

e.example {
  background: var(--color-text-error);
}
👍 Browser Native
👍 Live Themes
👍 Cascading is Awesome
Not Meaningfully Organized
Encourages One-Offs
Difficult to Automate(?)
Worth Some Effort?
Custom Property Color Palettes

Initial Light/Dark Based on User-OS (Firefox/Safari). Each theme comes with different levels of customization. Sass is only used to simplify the range-selector prefixes, and JS is only used to set custom properties. This is what a link looks like.

data-colors='invert' and now a link looks like this.

prime

accent
Know The **Trade-Offs**

adjust to your needs...
1. **Why** Design Systems?
2. Pattern **Audit**
3. Start **Small**
4. Define The **API**
5. **Integrated** Agile Process
Icons provide visual context and enhance usability.

Looking for the icons? Go to Resources > Icons.

---

**Base**

```html
<svg aria-hidden="true" class="slds-icon">
  <use xlink:href="/assets/icons/standard-sprite/svg/symbols.svg#account"></use>
</svg>

<span class="slds-assistive-text">Description of icon</span>
```

- For **action** icons, add the `.slds-icon_container--circle` class to the container (.slds-icon_container), which changes the background shape to a circle.
- **Doctype** icons have no background color.
- **Standard** and **Custom** icons have a rounded square shape and use a class on the container for the background color.
- **Utility** icons have no background color.

---

**Icon Colors**

---

**VARIANTS & STATES**

- **Base**
- **Standard**
- **Utility**
- **Action**
- **Doctype**
- **Custom**

---

**DOCUMENTATION**
SLDS Docs ⇒ HTML & CSS

Copy & Paste
Related Classes ⇒ **Data Attributes**

icon **color** and **size** classes
data-slds-icon-color="<option>"
.slds-icon-text-default | .slds-icon-text-warning | .slds-icon-text-error
HTML Template Logic is Great

pre-processors for your markup!
Example: Vue

<icon :image="isSuccess ? 'checkmark' : 'x'">
Components Provide Meaning

<my-icon name="gear" />
SLDS React ⇒ Component Library

Install the package on npm...
✅ type

JS

CSS

HTML

✅ purpose

button

modal

nav

\(\_\_\{(ツ\)_\/\_\)
1. **Why** Design Systems?

2. Pattern **Audit**

3. Start **Small**

4. Define The **API**

5. **Integrated Agile Process**
“Less Theory, More Practice”

Less Mockups, More Code
Ugly Helps Communicate

Isolate variables to focus on what’s important
Iterating Like Mad
Organizational Prototyping - Iteration Structures - Aligned

3:30 SWITCH - An option with:
Expanded Time Experiences
Ways to Think About Thirds
Copy - Variations of a Theme
Something Else
X Another Scenario
Yay Scenario
Iterate on Features/Epics

sketch » wire-frame » html markup » back-end » design
Test, Test, Test

Tight Feedback Loop
“Move Fast & Fix Things”
Everyone is responsible & collaborating
Make Patterns & Documentation

The **Lazy** Option
Automate from Structured Code
OddBird's **Herman** is based on **SassDoc** syntax
// SLDS Colors

// -----------

/// These are the colors we stole from SLDS system,
/// in order to create a demo.
/// @group color
/// @colors

$slds-colors: ( 
    'text-default': rgb(62, 62, 60),
    'text-warning': rgb(255, 183, 93),
    'text-error': rgb(194, 57, 52),
);
These are the colors we stole from SLDS system, in order to create a demo.

**text-default**
- #3e3e3c
- rgb(62, 62, 60)
- hsl(60, 2%, 24%)

**text-warning**
- #ff7500
- rgb(255, 183, 93)
- hsl(33, 100%, 68%)

**text-error**
- #c23934
- rgb(194, 57, 52)
- hsl(2, 58%, 48%)
### @sizes

#### SIZE PREVIEWS

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>rhythm, gutter</td>
<td>1.4rem</td>
</tr>
<tr>
<td>gutter-plus</td>
<td>2.1rem</td>
</tr>
<tr>
<td>double-gutter</td>
<td>2.8rem</td>
</tr>
<tr>
<td>flex-gutter</td>
<td>calc(0.7rem + 2.5vw)</td>
</tr>
<tr>
<td>spacer</td>
<td>calc(4.2rem + 2.5vw)</td>
</tr>
<tr>
<td>gutter-minus</td>
<td>1.05rem</td>
</tr>
<tr>
<td>shim</td>
<td>0.7rem</td>
</tr>
<tr>
<td>half-shim</td>
<td>0.35rem</td>
</tr>
<tr>
<td>quarter-shim</td>
<td>0.175rem</td>
</tr>
</tbody>
</table>
```latex
/// \texttt{@sizes \{ruler\}}

<table>
<thead>
<tr>
<th>SIZE PREVIEWS</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>rhythm, gutter</td>
<td>1.4rem</td>
</tr>
<tr>
<td>gutter-plus</td>
<td>2.1rem</td>
</tr>
<tr>
<td>double-gutter</td>
<td>2.8rem</td>
</tr>
<tr>
<td>flex-gutter</td>
<td>calc(0.7rem + 2.5vw)</td>
</tr>
<tr>
<td>spacer</td>
<td>calc(4.2rem + 2.5vw)</td>
</tr>
<tr>
<td>gutter-minus</td>
<td>1.05rem</td>
</tr>
<tr>
<td>shim</td>
<td>0.7rem</td>
</tr>
<tr>
<td>half-shim</td>
<td>0.35rem</td>
</tr>
<tr>
<td>quarter-shim</td>
<td>0.175rem</td>
</tr>
</tbody>
</table>
```
### @ratios

**RATIO: line-height (1.4)**

<table>
<thead>
<tr>
<th>Value</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>0.36</td>
</tr>
<tr>
<td>-2</td>
<td>0.51</td>
</tr>
<tr>
<td>-1</td>
<td>0.71</td>
</tr>
<tr>
<td>— base</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.4</td>
</tr>
<tr>
<td>2</td>
<td>1.96</td>
</tr>
<tr>
<td>3</td>
<td>2.74</td>
</tr>
</tbody>
</table>
/// @icons templates/icons/
Herman Components

/// Navigation items, with inactive and active states.
/// @group nav
/// @example njk
/// { % import 'nav.macros.njk' as nav %}
/// {{ nav.bar(active="dashboard", items=[...]) }}

.nav-bar {
  /* ... */
}

### EXAMPLE

<table>
<thead>
<tr>
<th>njk</th>
<th>html – compiled</th>
</tr>
</thead>
<tbody>
<tr>
<td>{% import 'nav.macros.njk' as nav %}</td>
<td>&lt;div class=&quot;nav-bar&quot;&gt;</td>
</tr>
<tr>
<td>{{ nav.bar(active='Dashboard', items=[</td>
<td>&lt;a href=&quot;/&quot; class=&quot;nav-item nav-active&quot;&gt;Dashboard&lt;/a&gt;</td>
</tr>
<tr>
<td>{name: 'Dashboard', component: 'Dashboard', href: '/'},</td>
<td>&lt;a href=&quot;/posts&quot; class=&quot;nav-item&quot;&gt;Posts&lt;/a&gt;</td>
</tr>
<tr>
<td>{name: 'Posts', component: 'Posts', href: '/posts'},</td>
<td>&lt;a href=&quot;/users&quot; class=&quot;nav-item&quot;&gt;Users&lt;/a&gt;</td>
</tr>
<tr>
<td>{name: 'Users', component: 'Users', href: '/users'},</td>
<td>&lt;a href=&quot;/settings&quot; class=&quot;nav-item&quot;&gt;Settings&lt;/a&gt;</td>
</tr>
<tr>
<td>{name: 'Settings', component: 'Settings', href: '/'}</td>
<td>&lt;/div&gt;</td>
</tr>
<tr>
<td>}}</td>
<td></td>
</tr>
</tbody>
</table>
Sass-First Breaks Down...
Only Nunjucks For Now...
VueDS (with Theo)

An open source tool for building Design Systems with Vue.js

Vue Design System provides you and your team a set of organized tools, patterns & practices. It works as the foundation for your application development.

Download (v1.3.1)  Docs
VueDS Components

<docs>
  jsx
  <nav-bar active="Dashboard" :navItems="[...]"/>

</docs>
<table>
<thead>
<tr>
<th>Prop name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>nav</td>
<td>The HTML element name used for the nav bar.</td>
</tr>
<tr>
<td>active</td>
<td>string</td>
<td>Required</td>
<td>State which tab is active when initiated (using name of the component).</td>
</tr>
<tr>
<td>navItems</td>
<td>array</td>
<td>Required</td>
<td>Menu items to be displayed on the nav bar.</td>
</tr>
</tbody>
</table>

```html
<nv-bar active="Dashboard" :nvItems="[

{name: 'Dashboard', component: 'Dashboard', href: '/'},
{name: 'Posts', component: 'Posts', href: '/posts'},
{name: 'Users', component: 'Users', href: '/users'},
{name: 'Settings', component: 'Settings', href: '/settings'}
]"/>
```
I Want Both :) 
please...
Design System as Artifact

Code is a Single Source of Truth
1. **Why** Design Systems?
2. **Pattern** Audit
3. Start **Small**
4. Define The **API**
5. **Integrated** Agile Process
Up-To-Date > Comprehensive
Start small, make it easy, and expand...
Meaningful & Structured Code
Readable by Humans & Machines
Inline Documentation

Helps the human factors...
Agile & Integrated Process

Everyone shares a single-source...
DRAGON!!!
Stay in touch...

Name

Email Address

Message

send it!
© 2019 OddBird | We provide custom web design and development services – from new products to refactors, design systems, and more – with a commitment to human-centered software and scalable, accessible, performant code.

Sign up for our developer newsletter

Subscribe