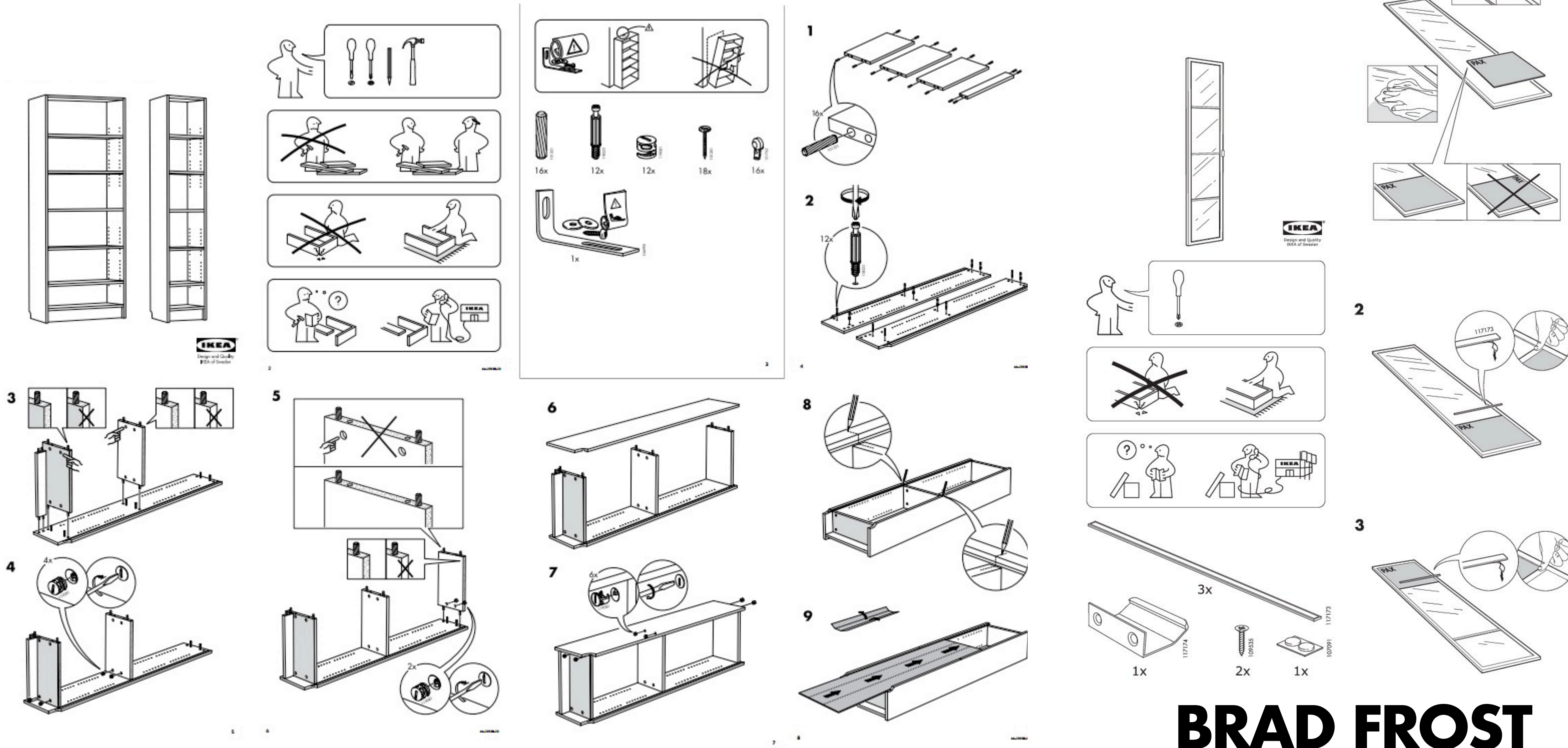


# THE TECHNICAL SIDE OF DESIGN SYSTEMS



**BRAD FROST**





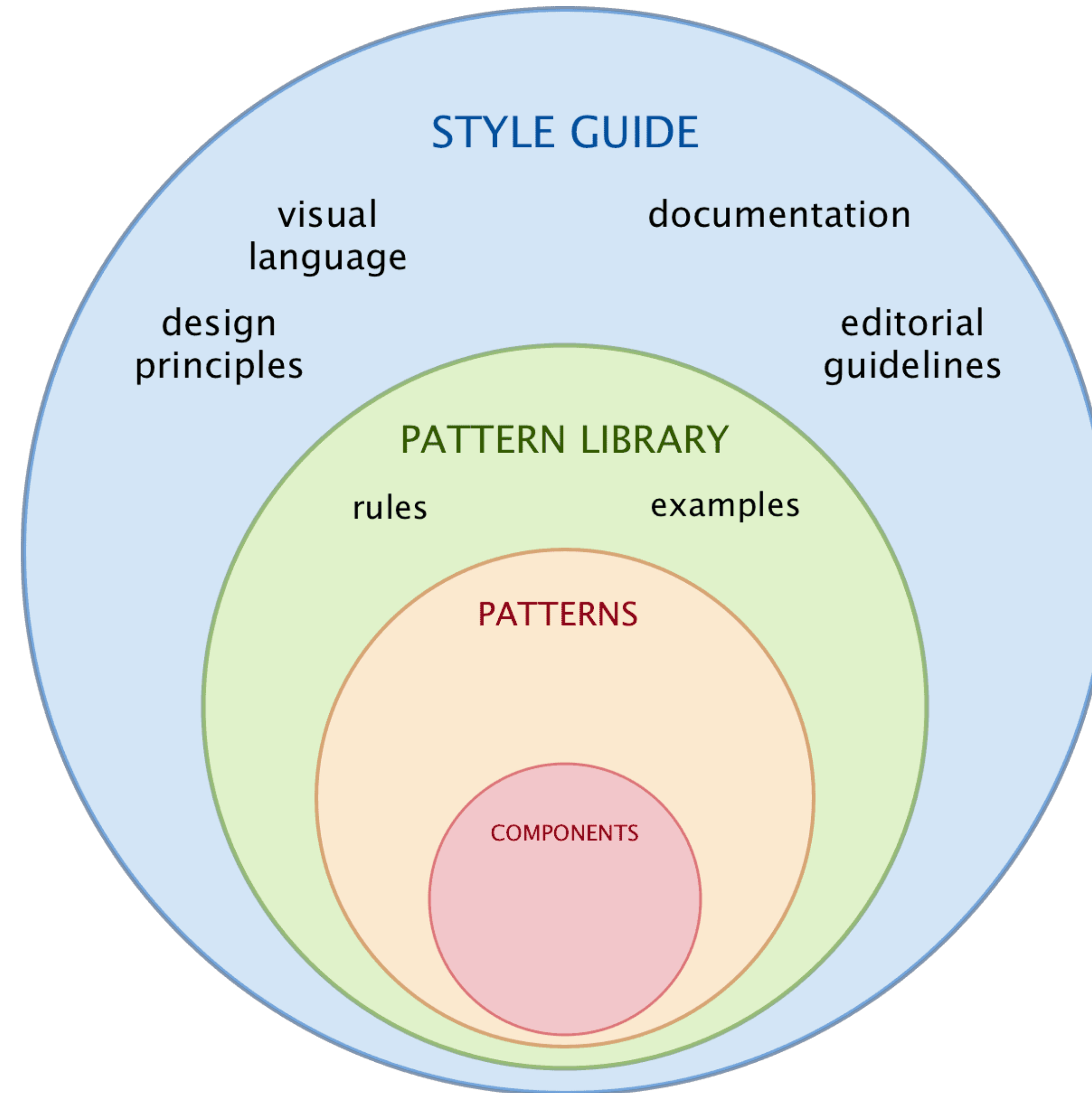


**"DESIGN SYSTEM" IS AN  
UNFORTUNATE NAME**



**A DESIGN SYSTEM IS THE STORY  
OF HOW YOUR ORGANIZATION  
DESIGNS AND BUILDS DIGITAL PRODUCTS**







Artboard

Option 1



Option 2



Option 3



Rotate

Flip

Radius

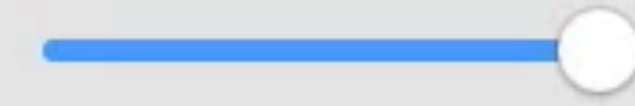


20

No Shared Style



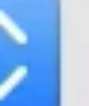
Opacity



100%

Blending

Normal



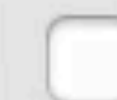
Fills



Normal



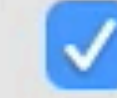
100%



Normal



100%



Normal



100%

Fill

Blending

Opacity

Borders



Shadows

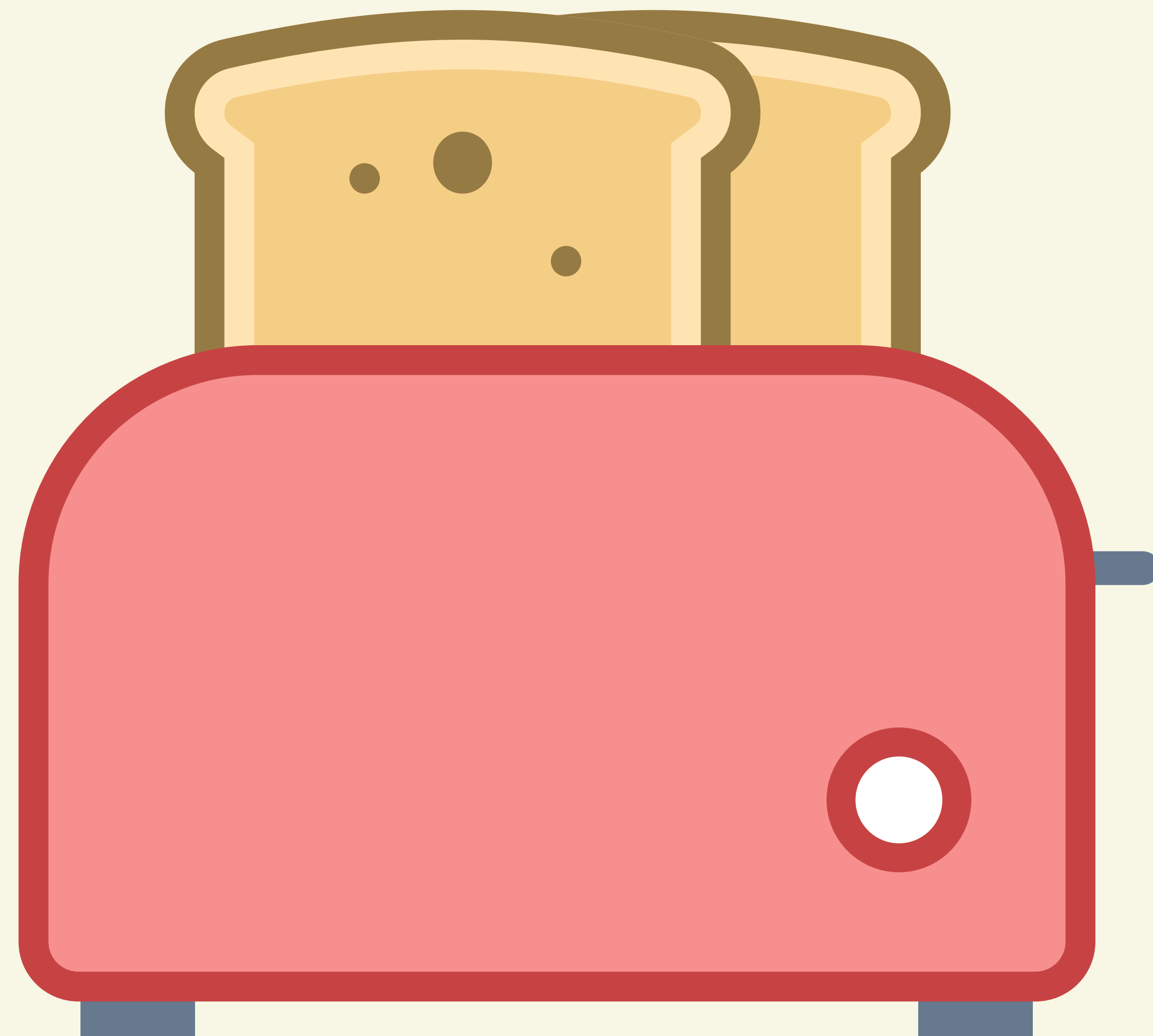




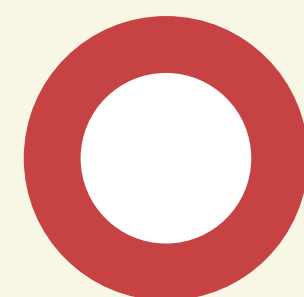
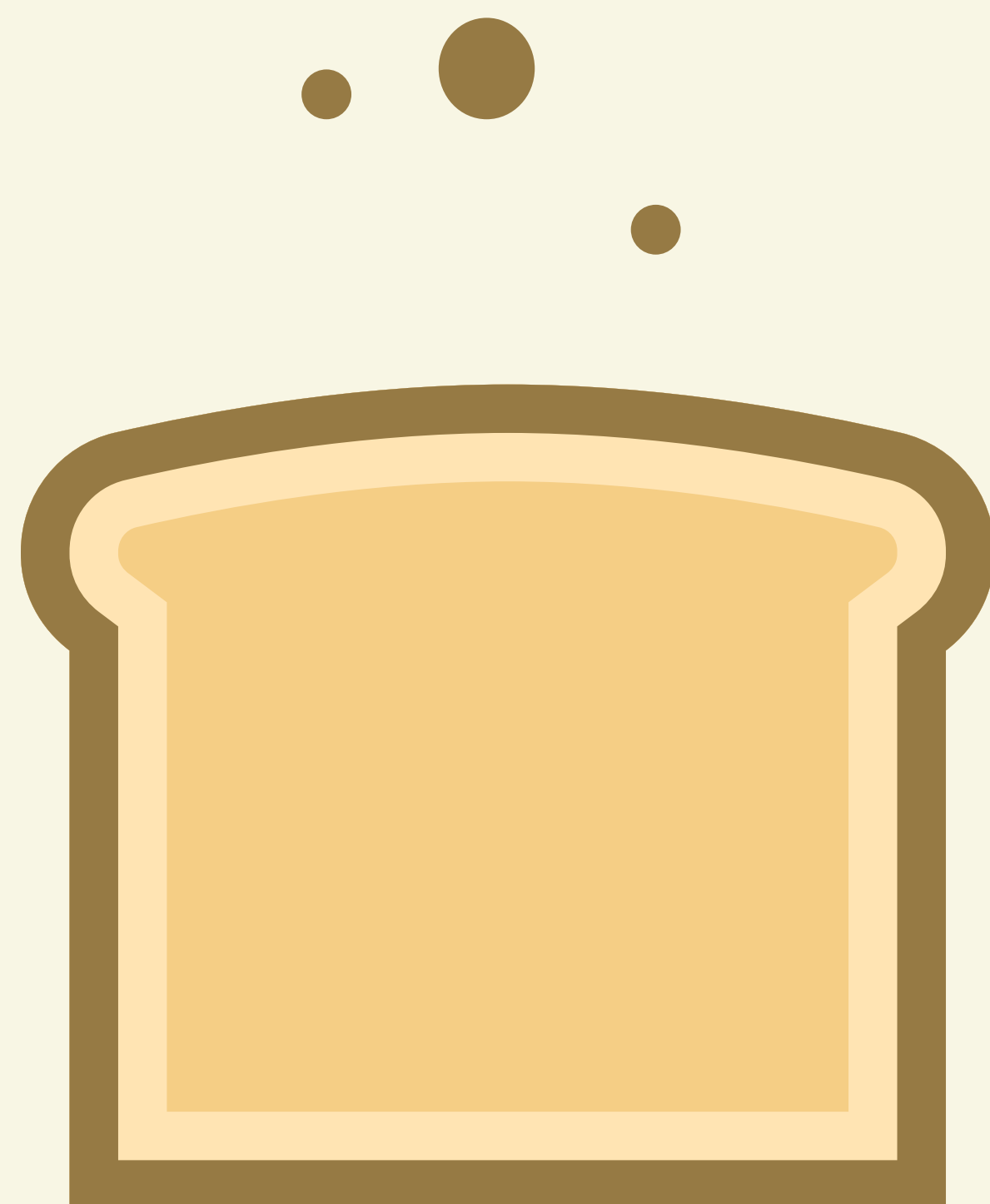


**!= design system**

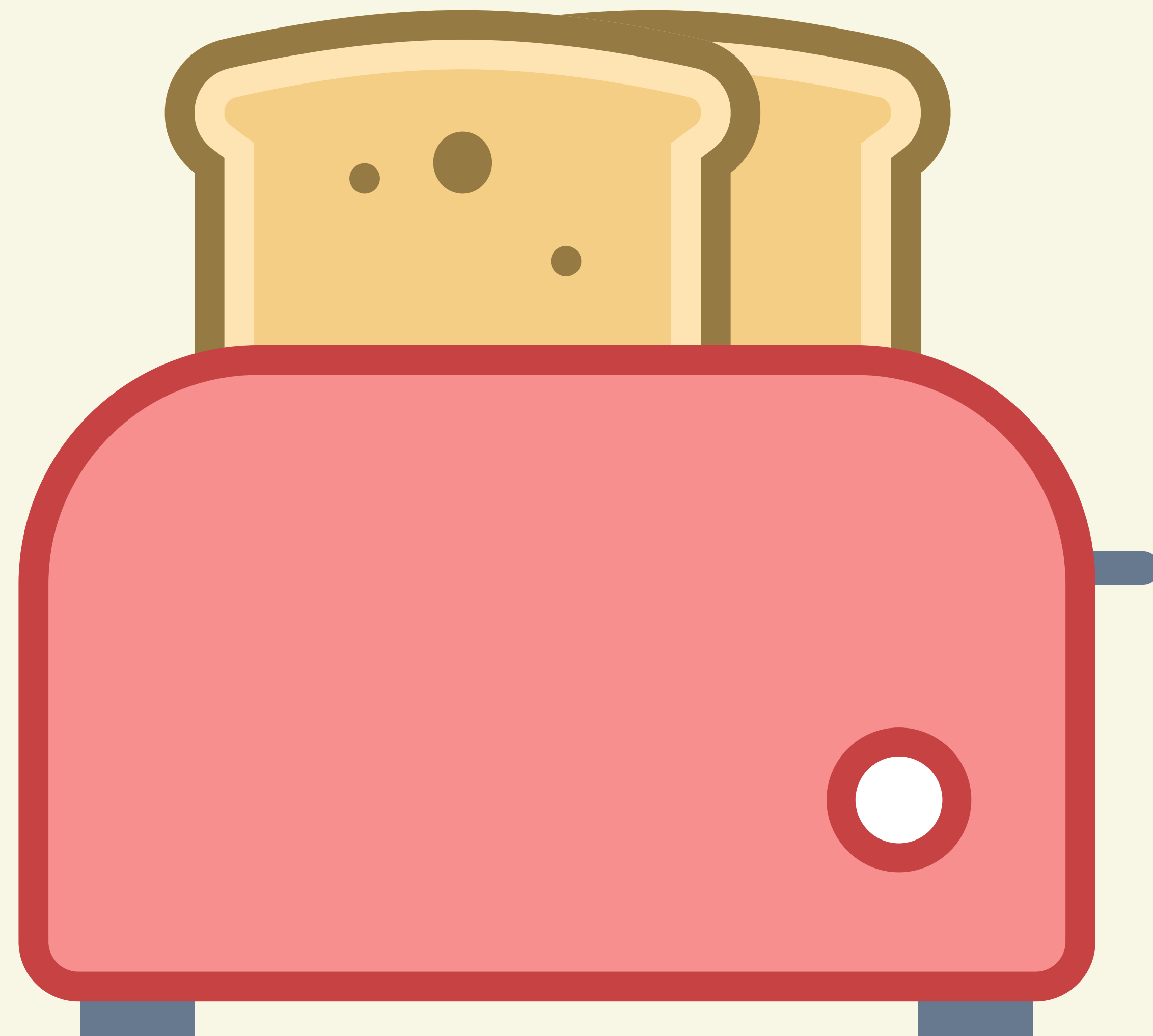
















comp

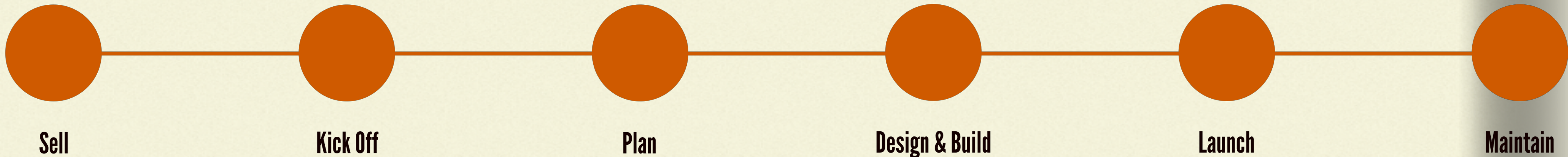


production

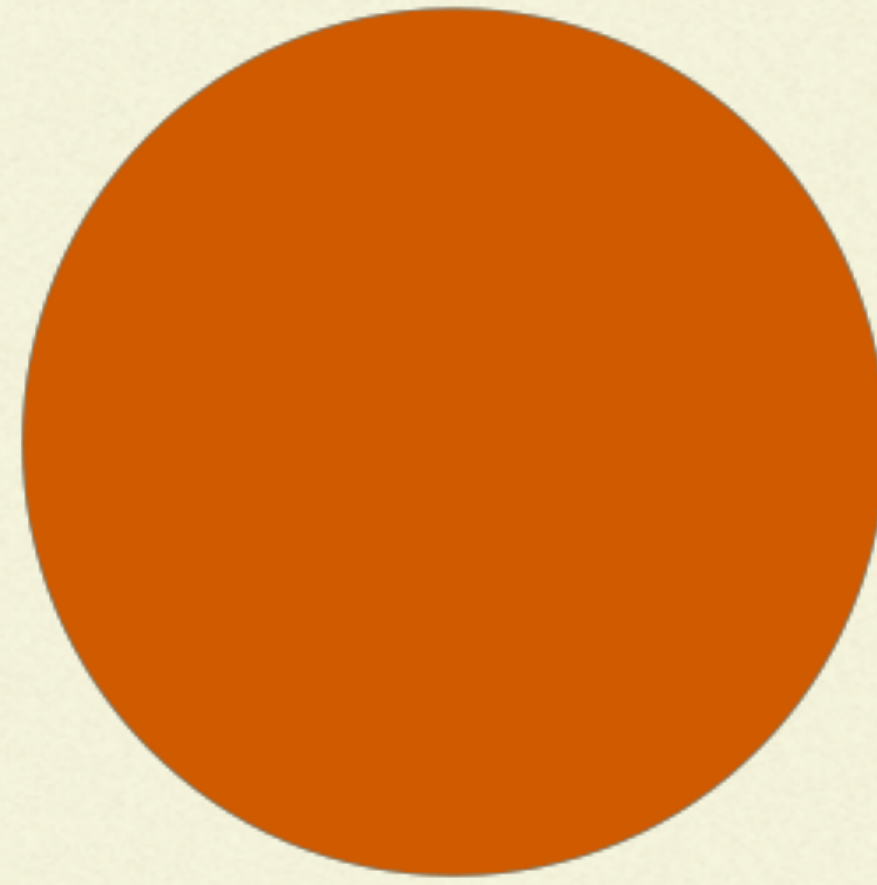


**THE HEART AND SOUL OF A DESIGN SYSTEM IS A CODE  
LIBRARY OF REUSABLE UI COMPONENTS THAT POWER  
REAL SOFTWARE APPLICATIONS**







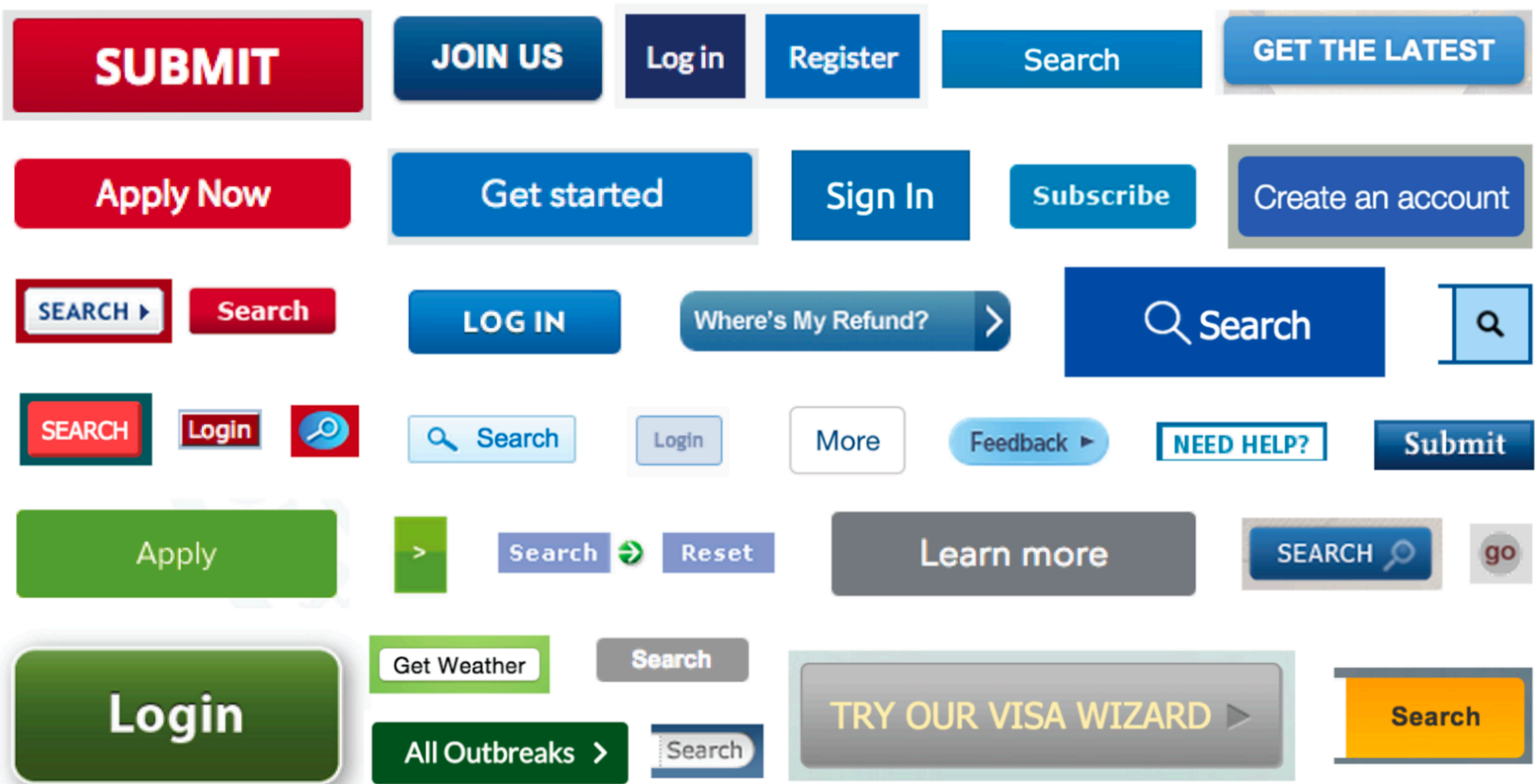


**Sell**



**Kick**







## JUST SOME OF THE INFINITE FLAVORS OF BUTTON MARKUP

<button>

<button class="btn">

<button class="button">

<button class="slds-button">

<button class="button-green">

<button class="green-button">

<button data-style="btn-green">

<button class="btn btn--green">

<button class="c-button c-button--primary">

<button class="f6 link dim ph3 pv2 mb2 dib">

<button class="bg-green hover:bg-green-dark">

<button style="background: green; color: white;">

Button



[Shop Now](#)

Submit



More +

 Search





Button



Button

Button



Button

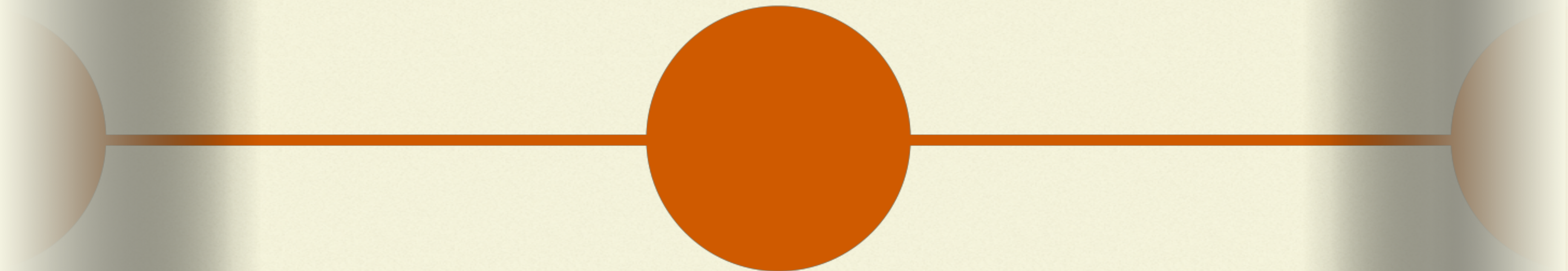
Button



# TECH BENEFITS OF DESIGN SYSTEMS

- **Reduce technical debt** - less frontend spaghetti code
- **Faster production** - less time coding common UI components and more time building real features and products
- **Higher-quality production** - bake in and enforce frontend best practices; teams can focus on iterating and improving rather than reinvention
- **Reduce QA efforts** - centralize certain QA tasks
- **Potentially adopt new technologies faster** - a design system can help make adding additional frameworks/technology more manageable
- **Useful reference** - an essential resource and hub for development best practices
- **Future-friendly foundation** - modify, extend, & improve upon over time





**Kick Off**



**WHAT'S YER TECH STACK?**



drupal

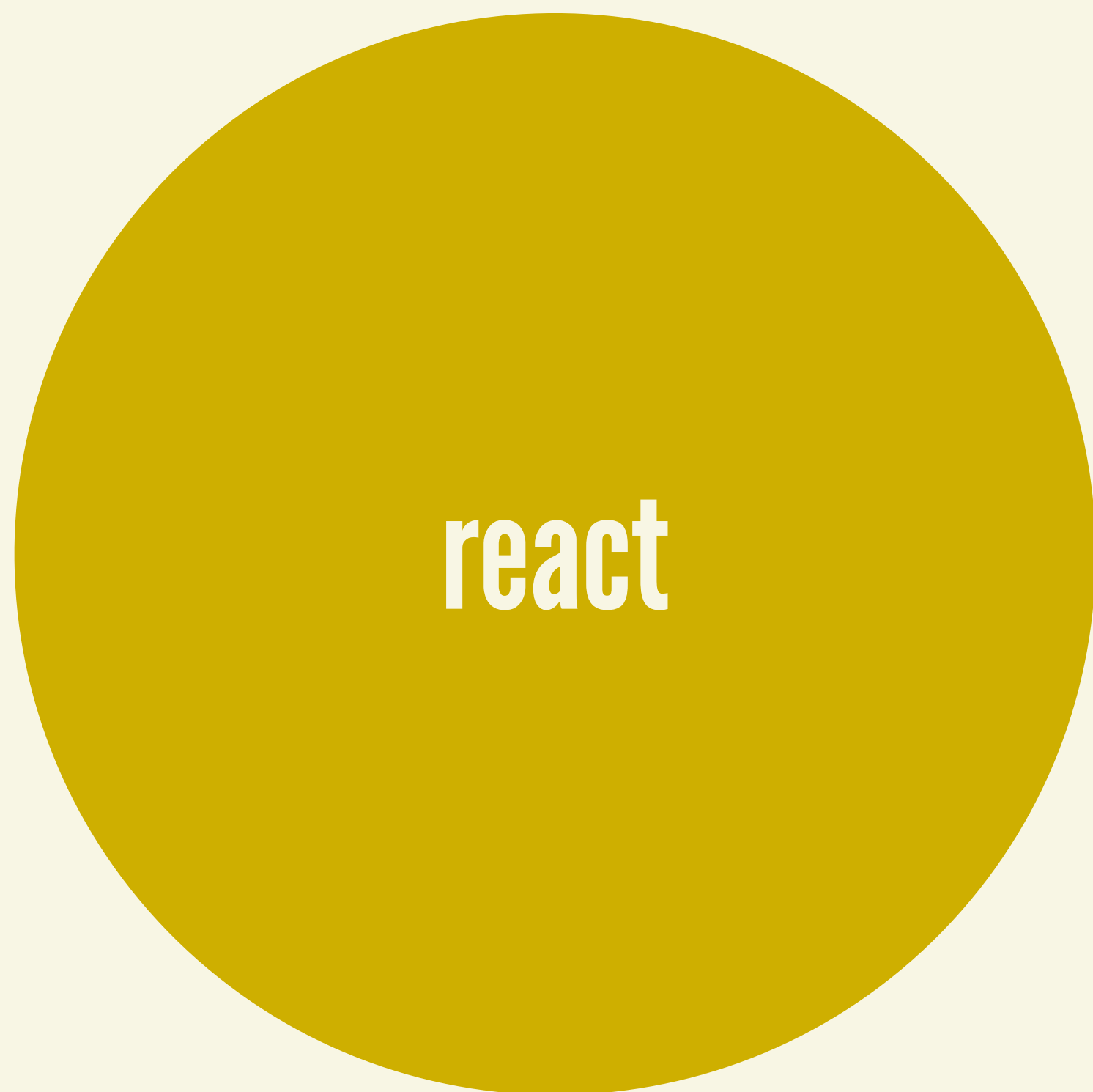


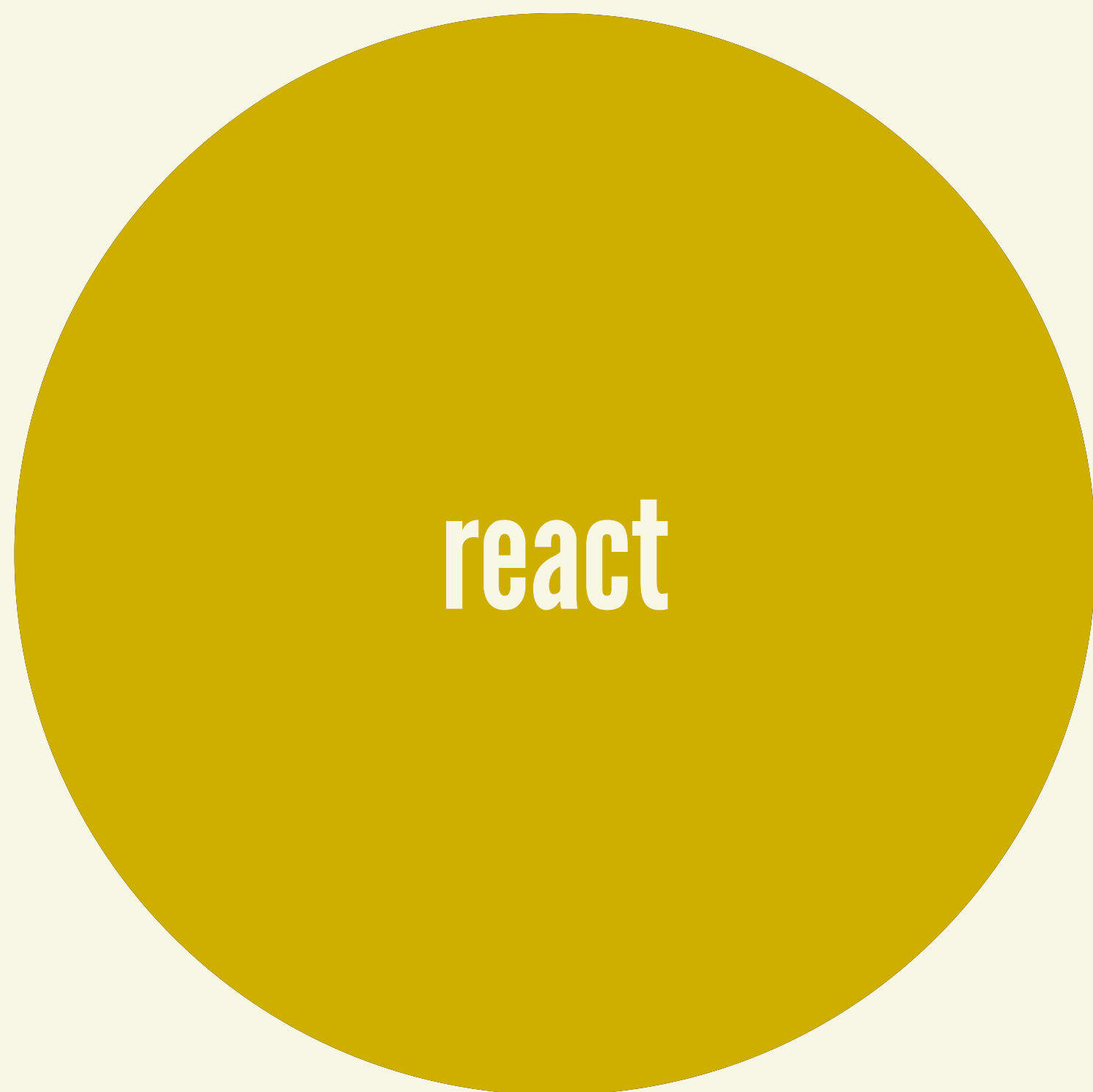
**angular**



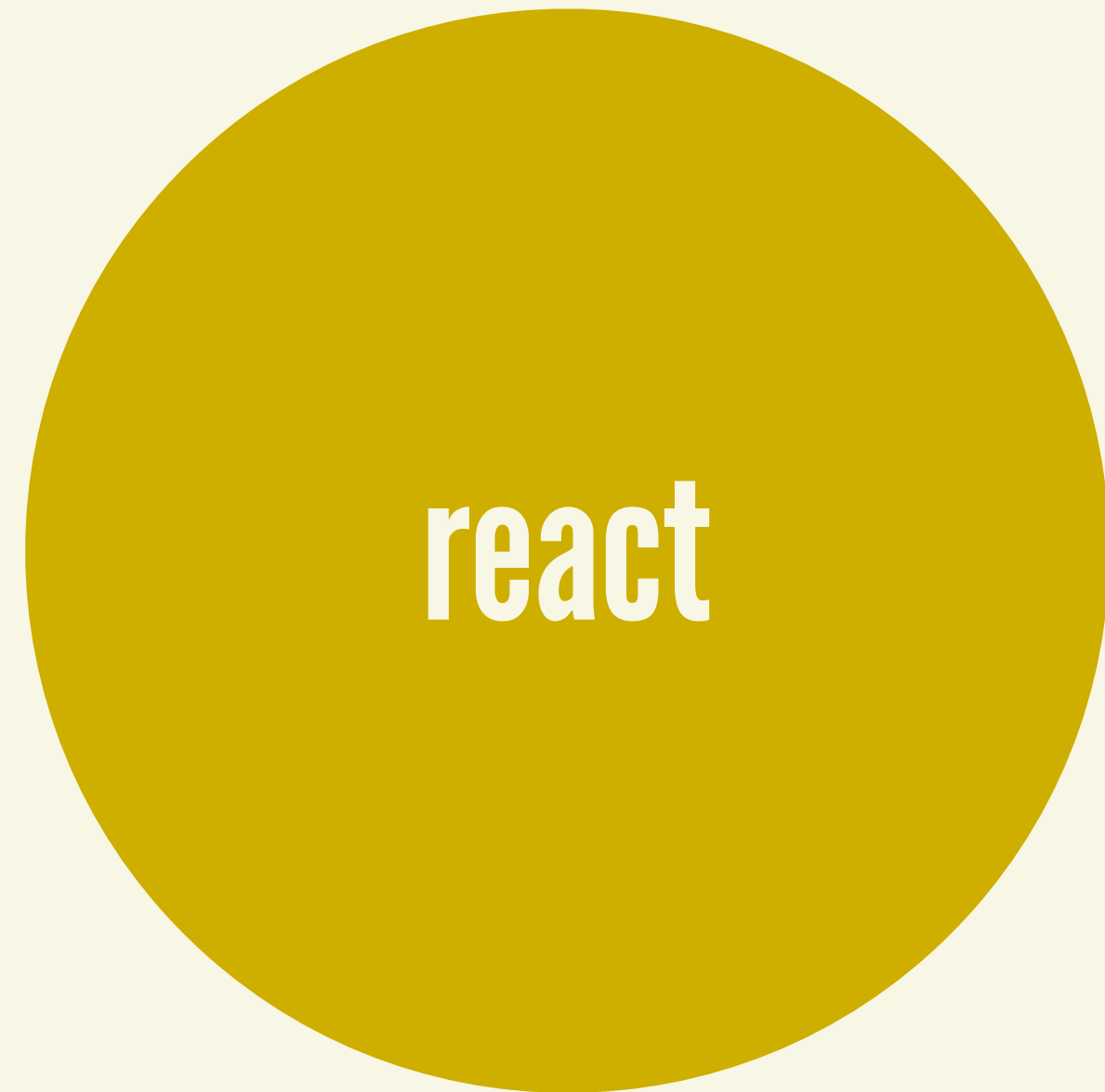
A large, solid orange circle is centered on a light cream-colored background. Inside the circle, the word "wordpress" is written in a white, lowercase, sans-serif font.

**wordpress**











**vue**

**drupal**

**react**

**wordpress**

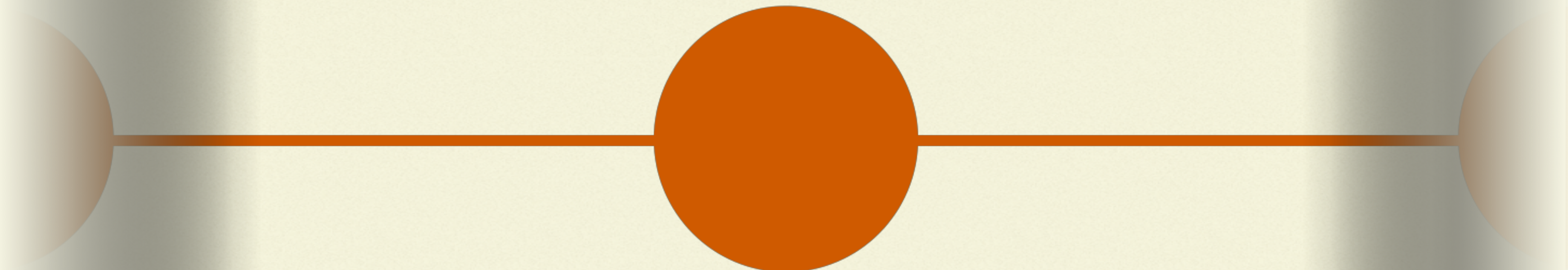




# ONE BRAND







**Plan**

**Diff**

**Design**





**HOW THE HELL ARE WE GOING TO DO ALL THAT?!**



**WE'RE NOT.**

**AT LEAST NOT YET.**









**products**



products



products





products



**products**

video editor

store lookup

inventory manager

look book

point of sale

company blog

partnership microsite

vendor info

fashion design

fashion merchandising

homepage

e-commerce

careers

chat

products



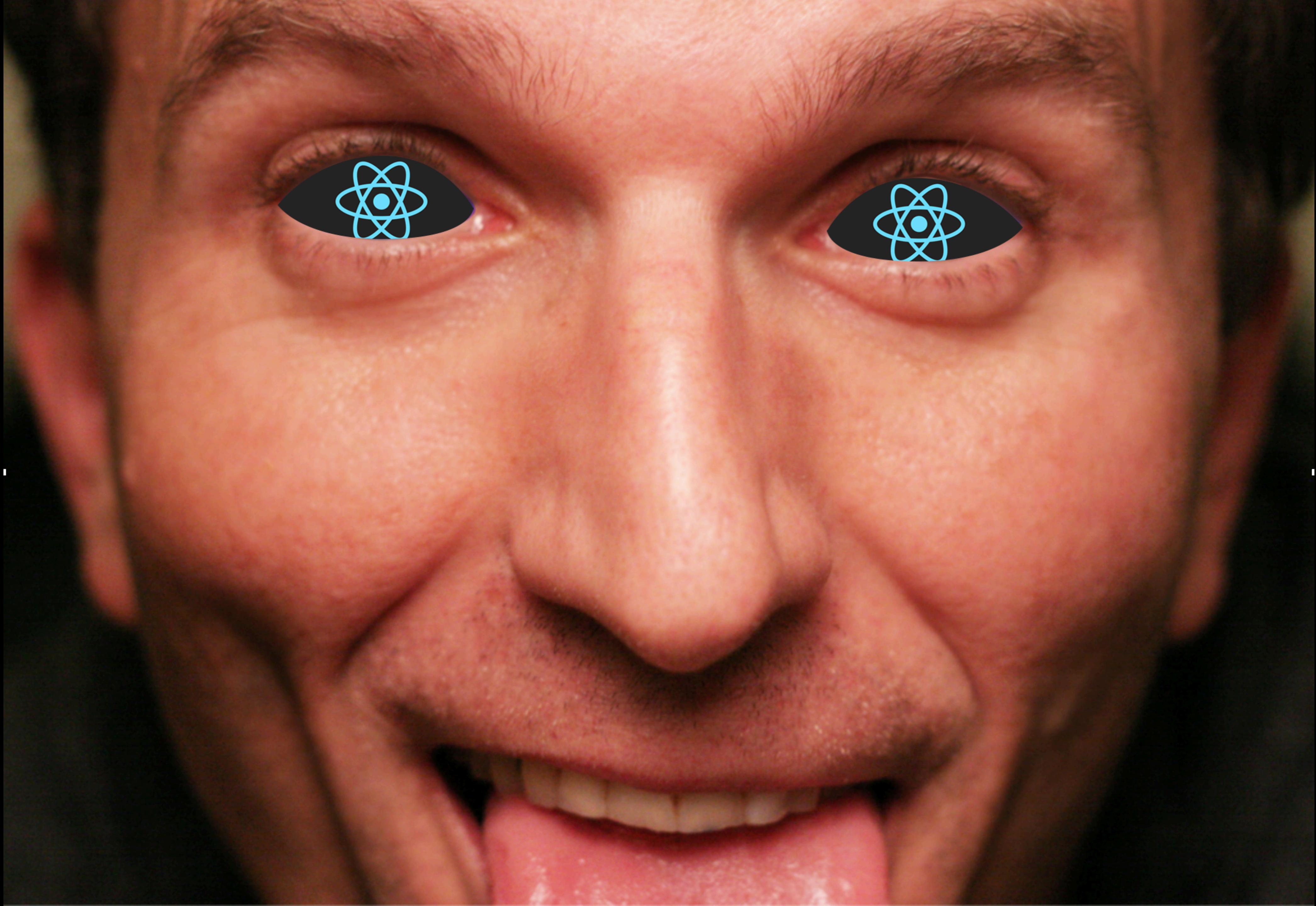
**products**





products





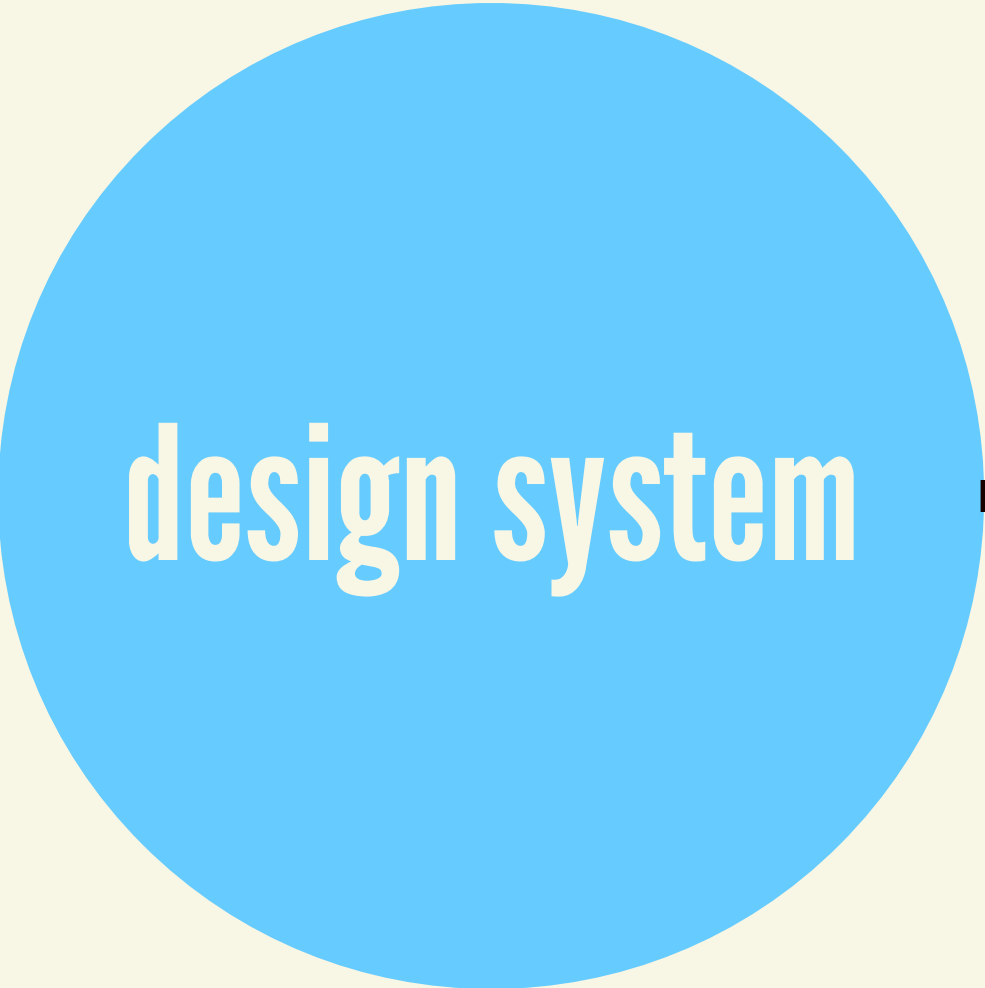




**products**



look book



design system



video editor



store lookup



inventory manager



point of sale



company blog



partnership microsite



vendor info



fashion design



fashion merchandising



homepage



e-commerce



careers



chat

pilot project dev

design system

products





# Design Systems: Pilots & Scorecards

April 4, 2017 at 12:05 A.M.

PILOTS ARE ONE OF the best ways to put your design system through its paces, especially before the design system even gets to a [v1](#). Like television pilots help test audience reactions to a series concept without investing significant resources to create the whole thing, application pilots are a good foundation for ensuring your design system's design and code are battle-tested.

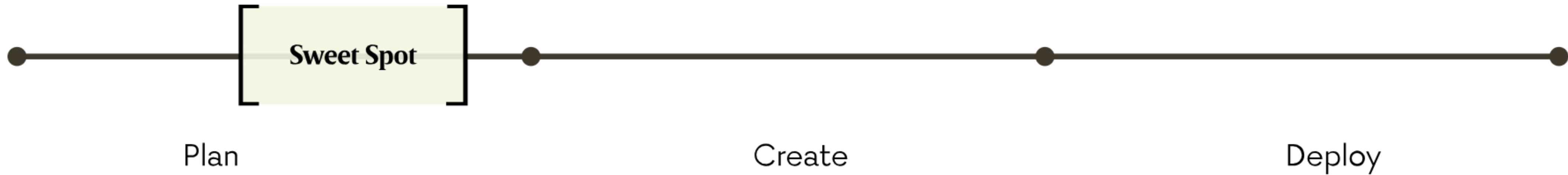
Here is how my teams and I identify great pilot candidates.

First, we want to know what kinds of digital products a design system should help our client to make. We'll ask them to tee up as many product presentations as they can muster. They'll usually do this by either

<http://danmall.me/articles/design-systems-pilots-scorecards/>

**Like television pilots help test audience reactions to a series concept without investing significant resources to create the whole thing, application pilots are a good foundation for ensuring your design system's design and code are battle-tested.**

**—Dan Mall**



# CRITERIA FOR CHOOSING PILOT PROJECTS

- **Potential for common components & patterns** - Does this pilot have many components and patterns that can be reused elsewhere?
- **Scope** - Is this work accomplishable in our pilot timeframe of [3–X weeks]?
- **Technical feasibility & independence** - How simple is the technical implementation? Is a large refactor or migration required?
- **Available champion** - Will someone working on this product be a good guinea pig, see it through, and then celebrate/evangelize (and even contribute back to) the design system?
- **Marketing potential** - Will this work excite others to use the design system?



**WHERE TO BUILD THE DESIGN SYSTEM?**

An iceberg floating in a body of water. The tip of the iceberg, which is visible above the water line, is colored orange and has a rough, rocky texture. The much larger part of the iceberg, which is submerged in the water, is a dark, mottled greyish-brown color with a similar rough texture. The water is a solid light blue color, and the sky above is a solid light yellow color. The text 'tech stack' is written in a bold, dark blue, sans-serif font across the middle of the submerged part of the iceberg.

**tech stack**

An iceberg floating in a body of water. The water is light blue and the sky is a pale yellow. The iceberg is split horizontally: the small tip above the water is orange and textured like ice, while the large submerged part is a darker, brownish-grey and also textured. The text 'ui code' is written in black on the orange tip, and 'tech stack' is written in dark blue on the submerged part.

**ui code**

**tech stack**



**ui code**



# a frontend workshop environment

Where do you build UI code? That sounds like a dumb question, but it's a deceptively important one. There's a few ways to do it:

1. **UI code is authored within your application environment.** If you're working on a WordPress or Drupal project, maybe you're writing HTML, CSS, and JavaScript in your theme folder. If you're building a UI in React or Vue.js, you're writing that presentational frontend code in

<http://bradfrost.com/blog/post/a-frontend-workshop-environment/>

**Create atomic design systems  
with Pattern Lab.**

**Download**

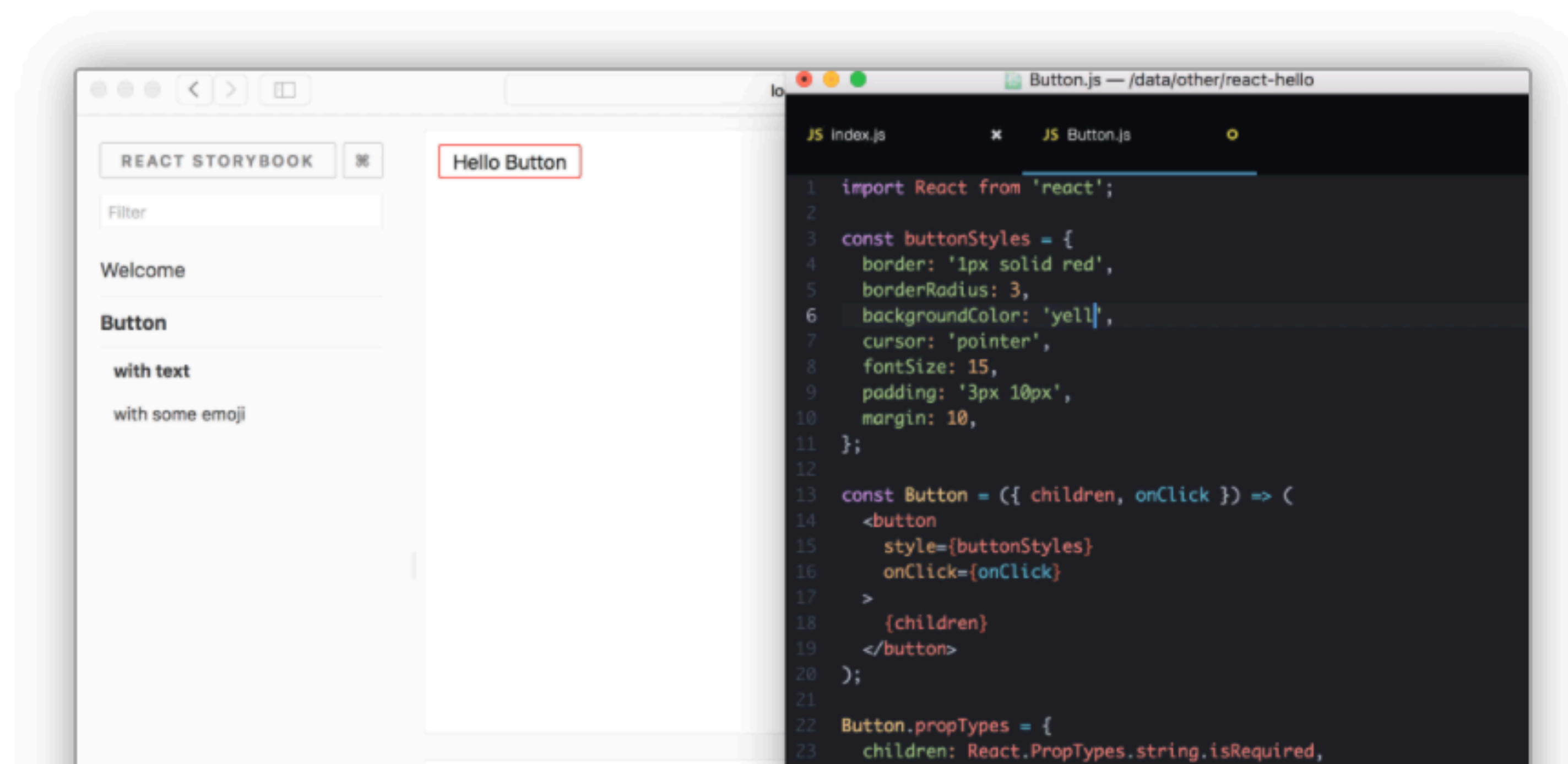
**View on Github**

**View Demo**

**<http://patternlab.io>**



The UI Development Environment  
You'll ❤️ to use



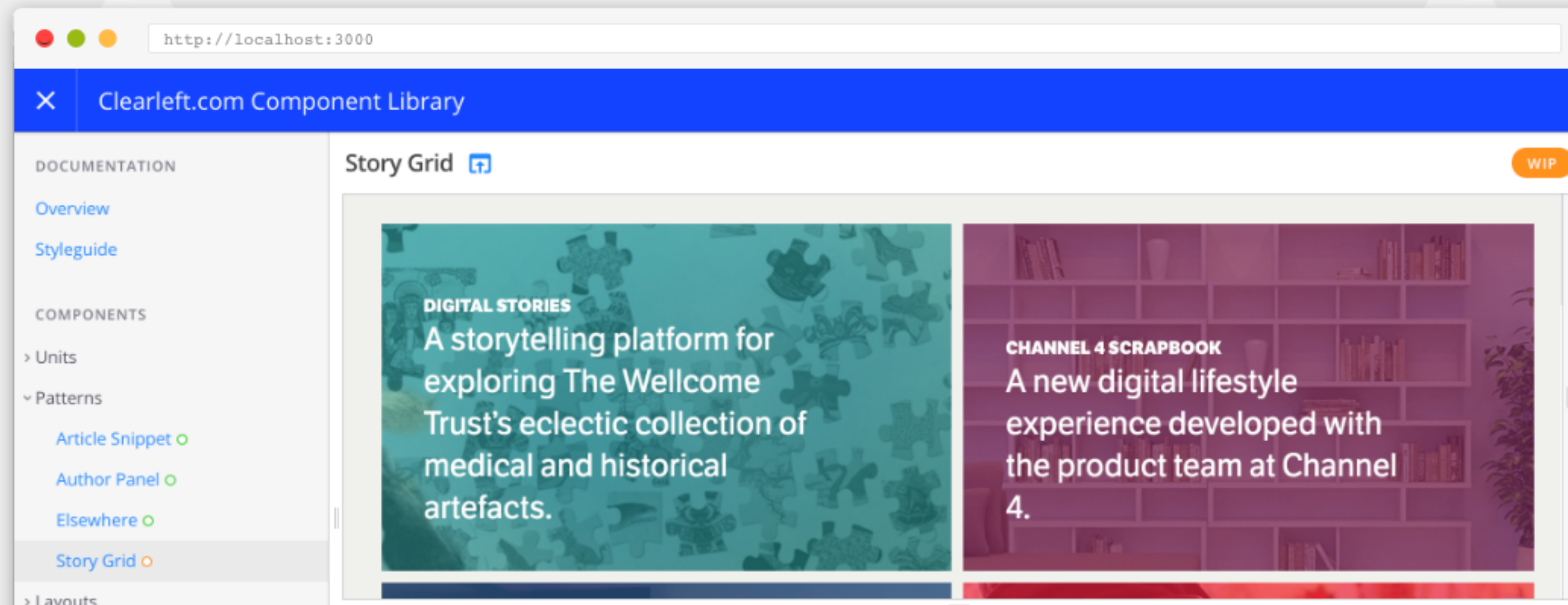
<https://storybook.js.org/>



# Build. Document. Integrate.

Powerful component libraries & styleguides that fit the way *you* work.

Get started →







# 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.

<https://vueds.com/>

Colors

▼

#ffffff

#cccccc

#999999

#666666

#333333

#000000

Fonts

▼

Primary font: "HelveticaNeue", "Helvetica", "Arial", sans-serif;  
*Primary font italic: "HelveticaNeue", "Helvetica", "Arial", sans-serif;*  
**Primary font bold: "HelveticaNeue", "Helvetica", "Arial", sans-serif;**  
Secondary font: Georgia, Times, "Times New Roman", serif;  
*Secondary font italic: Georgia, Times, "Times New Roman", serif;*  
**Secondary font bold; Georgia, Times, "Times New Roman", serif;**

Animations

▼

Fade: Duration: 0.3s Easing: ease-out (Hover to see effect)



Press "/" to search...

- Atoms
  - Design Tokens
  - Text
  - Forms
  - Icons
- Organisms
  - Interactive
    - Accordion
      - Default
      - Multiple Open
      - Accordion initially open
    - AccordionTabs
    - Carousel
    - Drawer
    - Modal
    - ResponsiveLinkListItem
    - ShowHide
    - Tabs
  - Buttons
  - Lists-And-Collections
  - Forms
    - DatepickerField
      - Default**
      - DatepickerFieldSingle



## Dates

Departure - Return







BENJAMIN ANDERSON #1234567

15,420 IDENTITY POINTS

37,543 TIER POINTS

12,457 TO PLATINUM

SEE ALL YOUR OFFERS >

ENJOY 6 EXCLUSIVE OFFERS ON US

ALL DATES

ALL OFFERS



MISCHIEF AHEAD

Valid Sep. 28, 2017 - Jan. 17, 2018



MIDNIGHT WAS MADE TO  
MISBEHAVE TESTING A VERY LONG  
OFFER HEADER



BEST SEATS, BEST SHOWS

Valid Nov. 1, 2017 - Nov. 30, 2017





BENJAMIN ANDERSON #1234567 ▾

15,420 IDENTITY POINTS

50,000 TIER POINTS

0 TO GOLD

SEE ALL YOUR OFFERS >

ENJOY 6 EXCLUSIVE OFFERS ON US

ALL DATES ▾

ALL OFFERS ▾



MISCHIEF AHEAD

Valid Sep. 28, 2017 - Jan. 17, 2018



MIDNIGHT WAS MADE TO  
MISBEHAVE TESTING A VERY LONG  
OFFER HEADER



BEST SEATS, BEST SHOWS

Valid Nov. 1, 2017 - Nov. 30, 2017





BENJAMIN ANDERSON #1234567 ▾

15,420 IDENTITY POINTS

50,000 TIER POINTS

0 TO STERLING

SEE ALL YOUR OFFERS >

# ENJOY 6 EXCLUSIVE OFFERS ON US

ALL DATES ▾

ALL OFFERS ▾



## MISCHIEF AHEAD

Valid Sep. 28, 2017 - Jan. 17, 2018



## MIDNIGHT WAS MADE TO MISBEHAVE TESTING A VERY LONG OFFER HEADER



## BEST SEATS, BEST SHOWS

Valid Nov. 1, 2017 - Nov. 30, 2017





BENJAMIN ANDERSON #1234567

15,420 IDENTITY POINTS

SEE ALL YOUR OFFERS >

ENJOY 6 EXCLUSIVE OFFERS ON US

ALL DATES

ALL OFFERS



MISCHIEF AHEAD

Valid Sep. 28, 2017 - Jan. 17, 2018



MIDNIGHT WAS MADE TO  
MISBEHAVE TESTING A VERY LONG  
OFFER HEADER



BEST SEATS, BEST SHOWS

Valid Nov. 1, 2017 - Nov. 30, 2017



# PACKAGES *and* OFFERS

Want more? Identity Membership and Rewards members enjoy exclusive, tailored offers.

SIGN IN

JOIN NOW



## MISCHIEF AHEAD

Valid Sep. 28, 2017 - Jan. 17, 2018



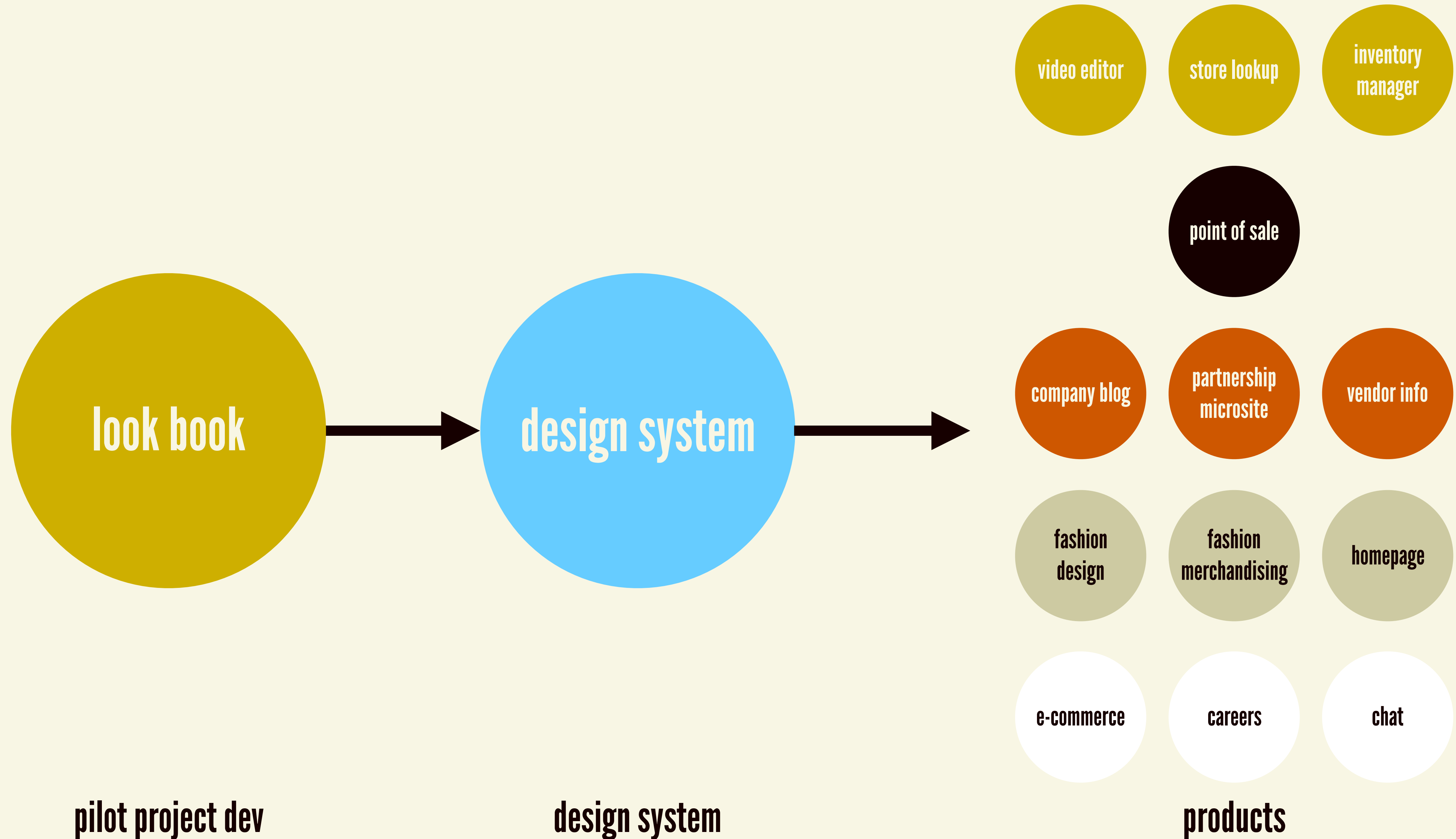
## MIDNIGHT WAS MADE TO MISBEHAVE TESTING A VERY LONG



## BEST SEATS, BEST SHOWS

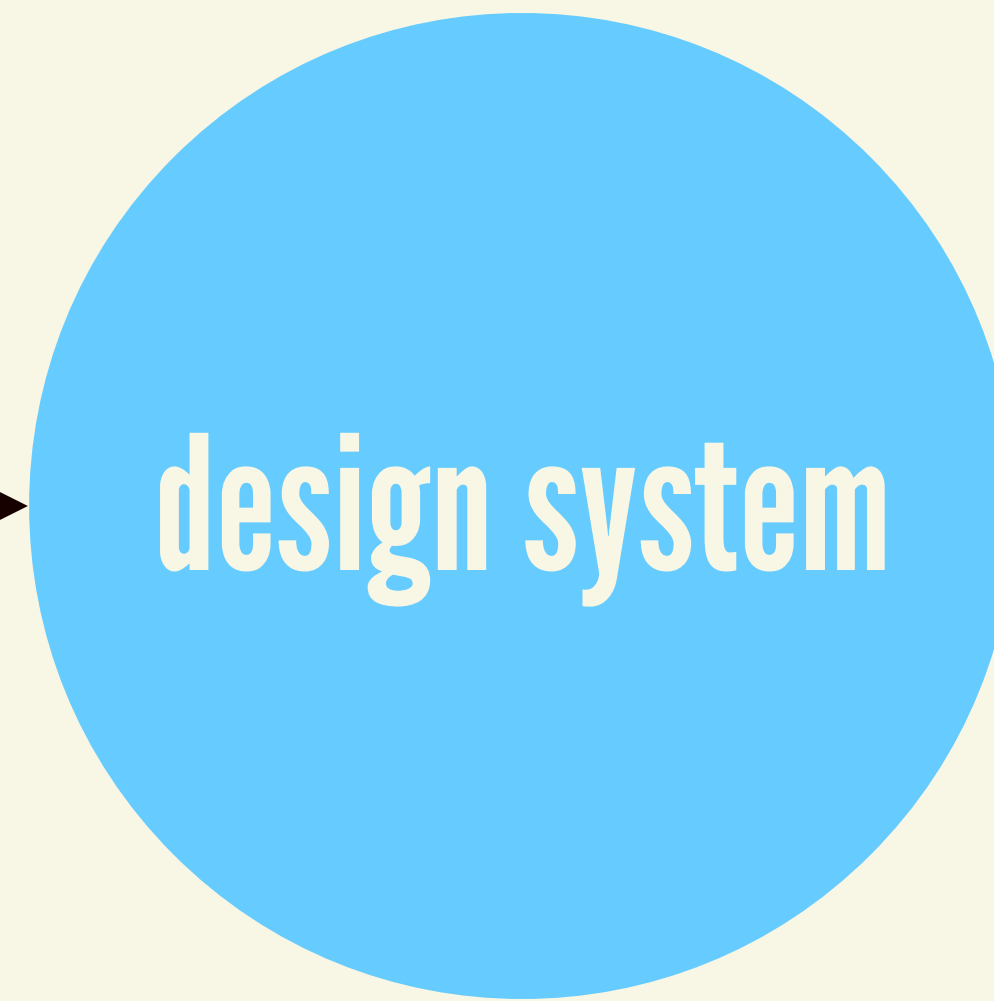
Valid Nov. 1, 2017 - Nov. 30, 2017



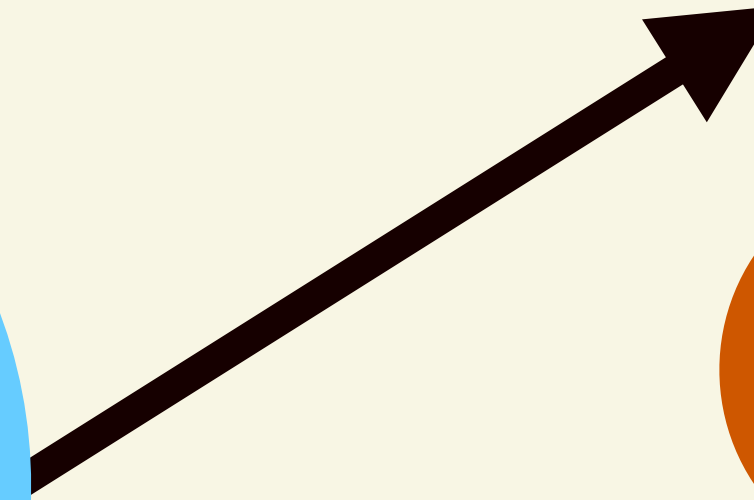




**pilot project & component dev**



**design system**



**products**

**HOW ARE WE GOING TO CODE THIS?**

## JUST SOME OF THE INFINITE FLAVORS OF BUTTON MARKUP

<button>

<button class="btn">

<button class="button">

<button class="slds-button">

<button class="button-green">

<button class="green-button">

<button data-style="btn-green">

<button class="btn btn--green">

<button class="c-button c-button--primary">

<button class="f6 link dim ph3 pv2 mb2 dib">

<button class="bg-green hover:bg-green-dark">

<button style="background: green; color: white;">



**A DESIGN SYSTEM'S CODEBASE NEEDS TO  
BE CONSISTENT AND COHESIVE**

# Frontend Guidelines Questionnaire

---

A one-page questionnaire to help your team establish effective frontend guidelines, so that you can write consistent & cohesive code together.

## HTML

---

### HTML Principles

- What are some general principles your team should follow when writing HTML? *(for example, authoring semantic HTML5 markup, accessibility, etc. See [these resources](#) for [inspiration](#))*

### HTML Tools

- Are you using an HTML preprocessor *(such as [HAML](#), [Jade](#), etc)?*
- Are you using a templating engine *(such as [Mustache](#), [Handlebars](#), etc)?*
- Does your backend architecture influence the frontend markup in any way (for example, WordPress will add `wp-paginate` to a class in your markup)? If so, can you highlight these conventions?

### HTML Style

- Spaces or Tabs?
  - What does HTML commenting look like?
-

**WHAT ARE SOME PRINCIPLES YOUR TEAM  
SHOULD FOLLOW WHEN WRITING HTML?  
CSS? JAVASCRIPT?**



**SMACSS, BEM OR OTHER METHODOLOGY?**

**WHAT CSS TOOLS (SASS, NORMALIZE, AUTOPREFIXER,  
ETC) WILL THE DESIGN SYSTEM USE?**

**WHAT JS TOOLS (FRAMEWORKS, LIBRARIES, TOOLING)  
WILL THE DESIGN SYSTEM USE?**

**SPACES OR TABS?**





KICKER

## This Is the Article Title

### HTML

```
1 <a class="c-media-block" href="#">
2
3   <div class="c-media-block__thumb">
4
5     <img class="c-media-block__img"
```

### CSS (SCSS)

```
9 /*-----*\
10    #MEDIA BLOCK
11  \*-----*/
12
13 /**
14  * 1) A media block is a collection of a linked thumbnail,
15  *    kicker, and headline displayed in a horizontal fashion
16  * 2) Vertically center the contents of the media block
17  */
18 .c-media-block {
19   display: flex;
20   align-items: center; /* 2 */
21   padding: 1rem;
22   text-decoration: none;
23 }
24
25 /**
26  * Thumbnail container
27  * 1) Can contain image, video, canvas, etc
28  */
29 .c-media-block__thumb {
30   margin-right: 1rem;
```

### JS

**GUIDELINES ARE GREAT AND ALL,  
BUT LET'S ENFORCE THEM!**

# ESLint

The pluggable linting utility for JavaScript and JSX

Get Started »

## Welcome

ESLint is an open source project originally created by [Nicholas C. Zakas](#) in June 2013. Its goal is to provide a pluggable linting utility for JavaScript.

## Latest News

- [Funding ESLint's Future: An Update](#) 1 May 2019

---

About

Rules

<https://eslint.org/>

# stylelint

npm v10.0.1 build passing build passing downloads 3M/month backers 4 sponsors 4

A mighty, modern linter that helps you avoid errors and enforce conventions in your styles.

## Features

---

It's mighty because it:

- understands the **latest CSS syntax** including custom properties and level 4 selectors
- extracts **embedded styles** from HTML, markdown and CSS-in-JS object & template literals
- parses **CSS-like syntaxes** like SCSS, Sass, Less and SugarSS
- has over **170 built-in rules** to catch errors, apply limits and enforce stylistic conventions
- supports **plugins** so you can create your own rules or make use of plugins written by the community
- automatically **fixes** some violations (*experimental feature*)
- is **well tested** with over 10000 unit tests
- supports **shareable configs** that you can extend or create your own of
- is **unopinionated** so you can tailor the linter to your exact needs
- has a **growing community** and is used by [Facebook](#), [GitHub](#) and [WordPress](#)

## Example output

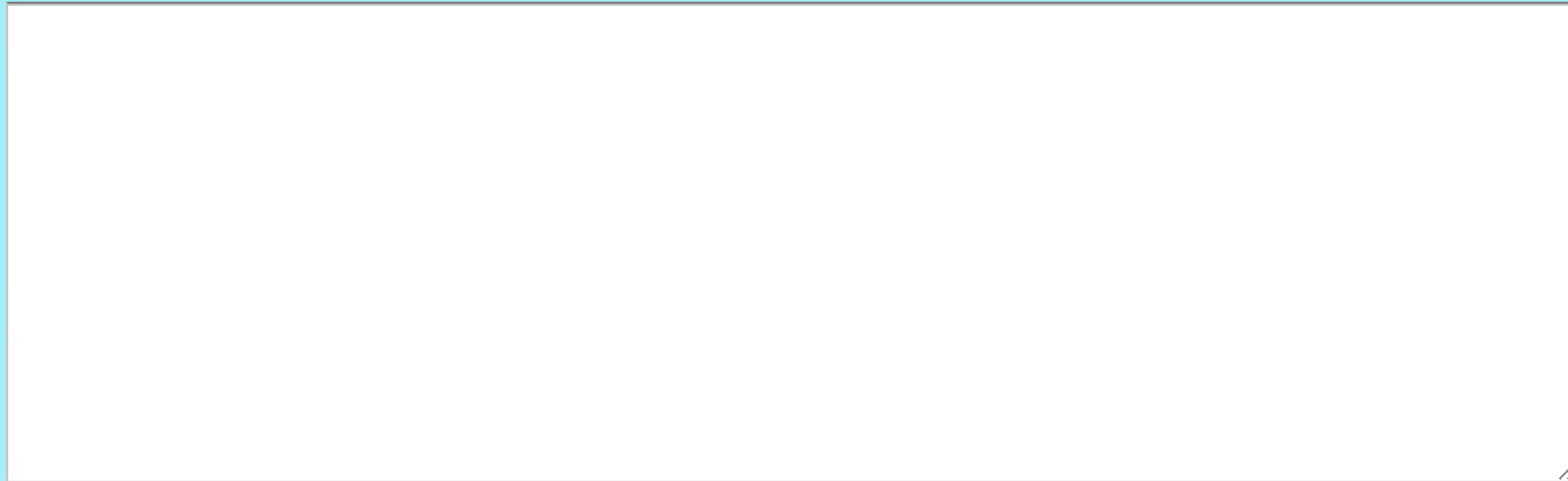
<https://stylelint.io/>



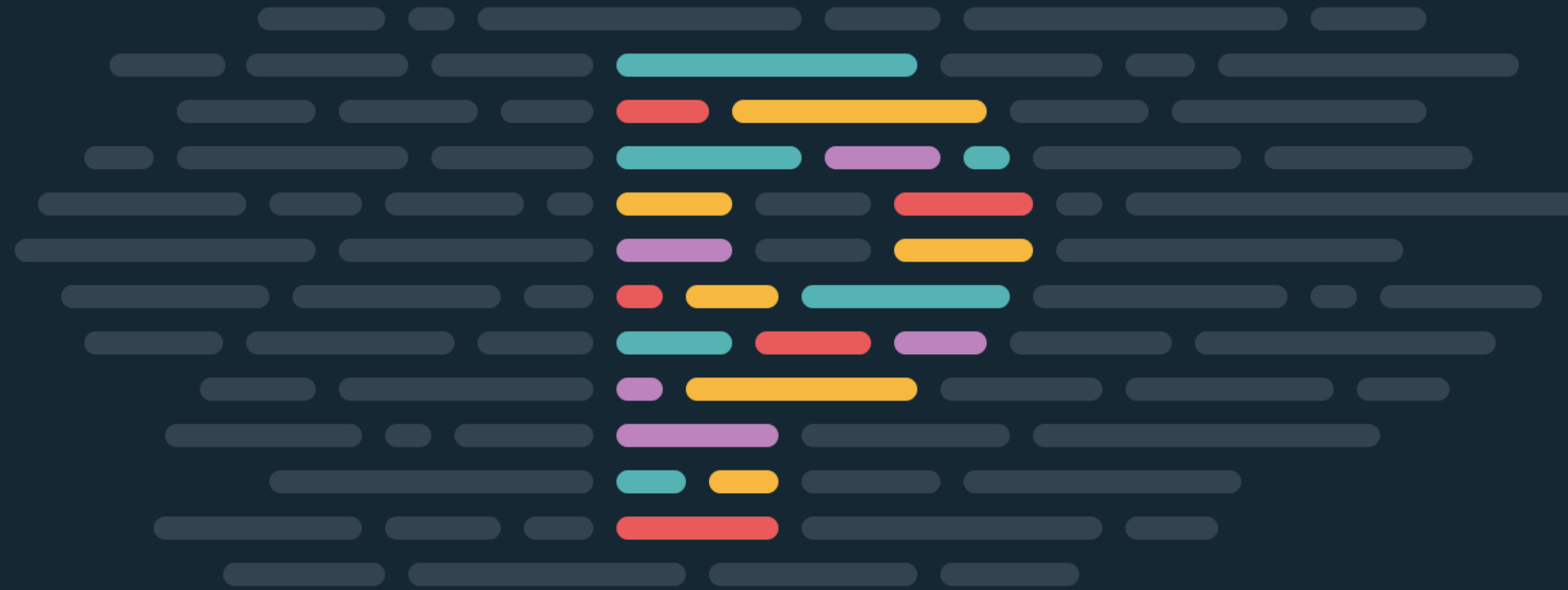
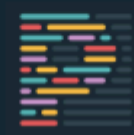
# CSS LINT

Will hurt your feelings\*  
(And help you code better)

Your CSS goes here. The more, the better. Linting works best when we see the big picture, so give us everything you've got.



<http://csslint.net/>



TRY IT OUT

GET STARTED

OPTIONS

## # What is Prettier?

\* An opinionated code formatter

## # Why?

\* You press save and code is formatted

<https://prettier.io/>



## EXPLORER

- DESIGN-SYSTEM
  - Autocomplete
    - Autocomplete.css
    - Autocomplete.js
    - Autocomplete.scss
    - Autocomplete.stories.js
    - index.js
  - Badge
  - Band
  - Boilerplate
  - Box
  - Breadcrumbs
    - Breadcrumbs.css
    - Breadcrumbs.js
    - Breadcrumbs.scss
    - Breadcrumbs.stories.js
    - index.js
  - Button
    - Button.css
    - Button.js
    - Button.scss
    - Button.stories.js
    - index.js
  - ButtonGroup
    - ButtonGroup.css
    - ButtonGroup.js

## JS Breadcrumbs.js

src › components › Breadcrumbs › JS Breadcrumbs.js › Breadcrumbs › render › items.map() callback

```
1  import React from "react";
2  import PropTypes from "prop-types";
3  import shortid from "shortid";
4  | import "../Breadcrumbs.css";
5  import Icon from "../Icon";
6
7  class Breadcrumbs extends React.Component {
8  |   constructor(props) {
9  |     super(props);
10 |     this.generateId = shortid.generate();
11 |   }
12
13  render() {
14 |   const { items, id,      ariaLabel,      ...other } = this.props;
15
16 |   return (
17 |     <nav aria-label={ariaLabel} id={id || this.generateId} className='c-breadcrumbs' {...other}>
18 |       <ul className='c-breadcrumbs__list'>
19 |         {items.map(function(item, index) {
20 |           return (
21 |             <li
22 |               className='c-breadcrumbs__item'
23 |               key={`c-breadcrumbs__item-${index}`}
24 |             >
25 |               <a
26 |                 className='c-breadcrumbs__link'
27 |                 href={item.href}
28 |               >
29 |                 {item.text}
30 |               </a>
31 |             </li>
32 |           );
33 |         })}
34 |       </ul>
35 |     </nav>
36 |   );
37 | }
38
39 Breadcrumbs.propTypes = {
40 |   items: PropTypes.arrayOf(PropTypes.shape({
41 |     href: PropTypes.string.isRequired,
42 |     text: PropTypes.string.isRequired,
43 |   })).isRequired,
44 |   id: PropTypes.string,
45 |   ariaLabel: PropTypes.string,
46 |   ...PropTypes.exact(Icon.propTypes)
47 | };
48
49 Breadcrumbs.defaultProps = {
50 |   id: null,
51 |   ariaLabel: null,
52 | };
53
54 export default Breadcrumbs;
```





# Make it hard to screw up driven development

Chris Coyier - Apr 2, 2019

<https://css-tricks.com/make-it-hard-to-screw-up-driven-development/>



# CSS ARCHITECTURE

```

const Button = styled.a`
  display: inline-block;
  border-radius: 3px;
  padding: 0.5rem 0;
  margin: 0.5rem 1rem;
  width: 11rem;
  background: transparent;
  color: white;
  border: 2px solid white;
  |
  ${props => props.primary && css`
    background: white;
    color: palevioletred;
  `}
  ,

```

```

import { css, cx } from 'emotion'

const color = 'white'

render(
  <div
    className={css`
      padding: 32px;
      background-color: hotpink;
      font-size: 24px;
      border-radius: 4px;
      &:hover {
        color: ${color};
      }
    `}
  >
    Hover to change color.
  </div>
)

```

## OL' TRUSTY, STURDY, PORTABLE, RELIABLE CSS

```
<link rel="stylesheet" href="cdn.com/design-system.1.0.css" />
```





## ONE TYPE OF CSS NAMING (THAT HAS WORKED PRETTY DANG WELL FOR ME OVER THE YEARS)

```
.cn-c-btn--secondary {  
  
}
```

# GLOBAL NAMESPACE

**.cn-**

## CLASS PREFIXES

- .c-** for UI components
- .l-** for layout-related styles
- .u-** for utility classes
- .is-** and **.has-** for state-based classes
- .js-** for JavaScript-specific classes

# BEM-STYLE SYNTAX

```
.cn-c-btn {} /* btn is Block */
```

```
.cn-c-btn__icon {} /* __icon is Element */
```

```
.cn-c-btn--secondary {} /* --secondary is Modifier */
```



# css architecture for design systems

We just created a design system for a huge organization and established a CSS architecture we're quite pleased with. It's one of the first times I've ever gotten to a project's finish line without wishing I'd done at least a few things differently. So I thought it would be great to share how we went about creating our system's CSS architecture.

To give a bit of a context for the project, we were tasked with creating a design system and style guide meant to serve the organization's thousands of developers, who employ a vast array of technologies to build their over 500 internal web applications.

~~The overwhelming majority of the organization's developers don't~~

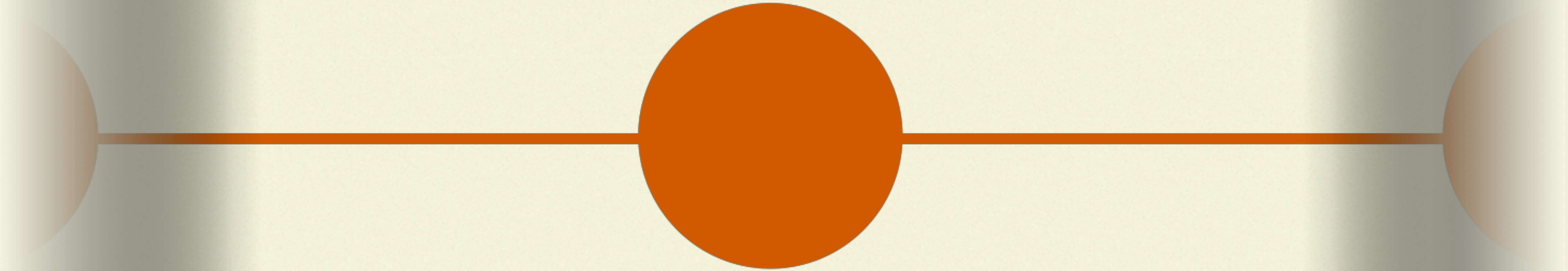
<http://bradfrost.com/blog/post/css-architecture-for-design-systems/>

```
css { guide:  
      lines; }
```

High-level advice and  
guidelines for writing sane,  
manageable, scalable CSS

<http://cssguidelin.es/>

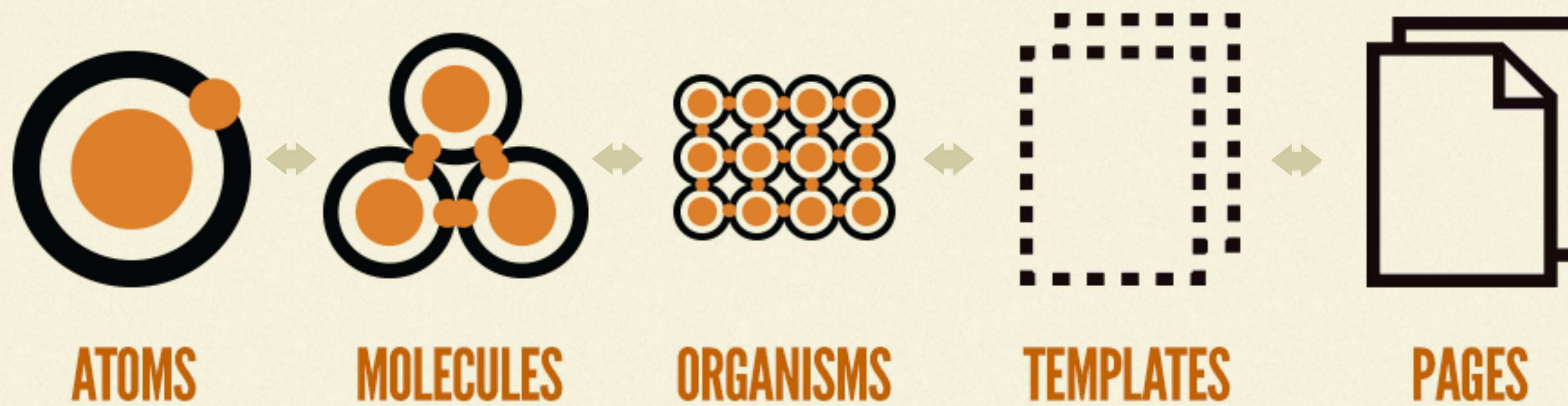




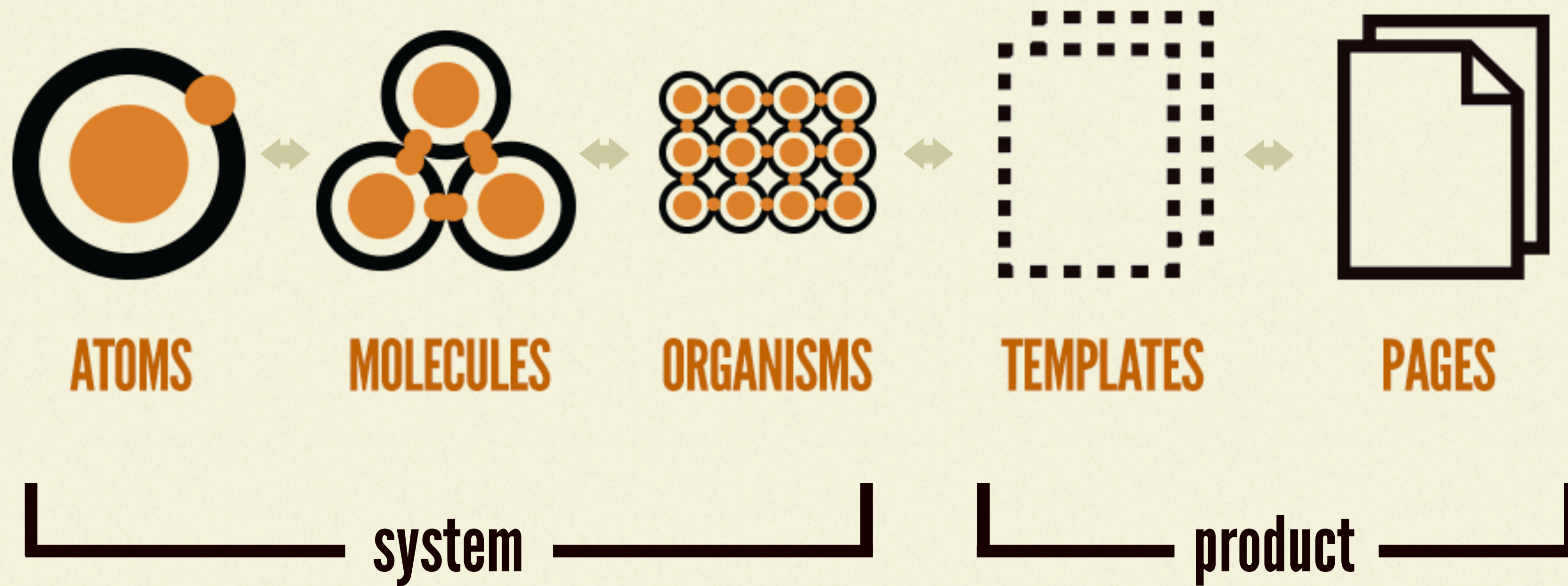
**Design & Build**

La











**The Design System informs our Product Design.  
Our Product Design informs the Design System.**  
**-Jina Anne**

**HERE'S THE PART WHERE I**

**~~PEE MY PANTS~~**

**PEE MY PANTS**

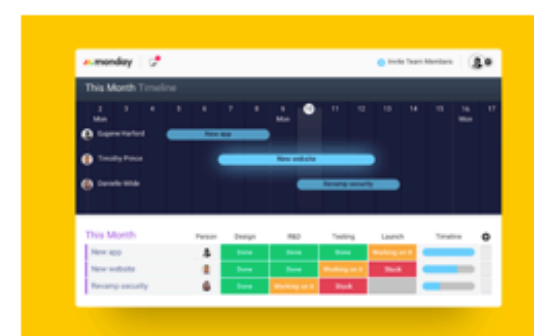
**LIVE CODE**



# Designer + Developer Workflow

August 8, 2018 at 10:31 P M

THE WAY DESIGNERS AND DEVELOPERS work together today is broken. It's too siloed and separate; “collaboration” is a fantasy that few enjoy.



The new generation of project management tools is here and it's visual.

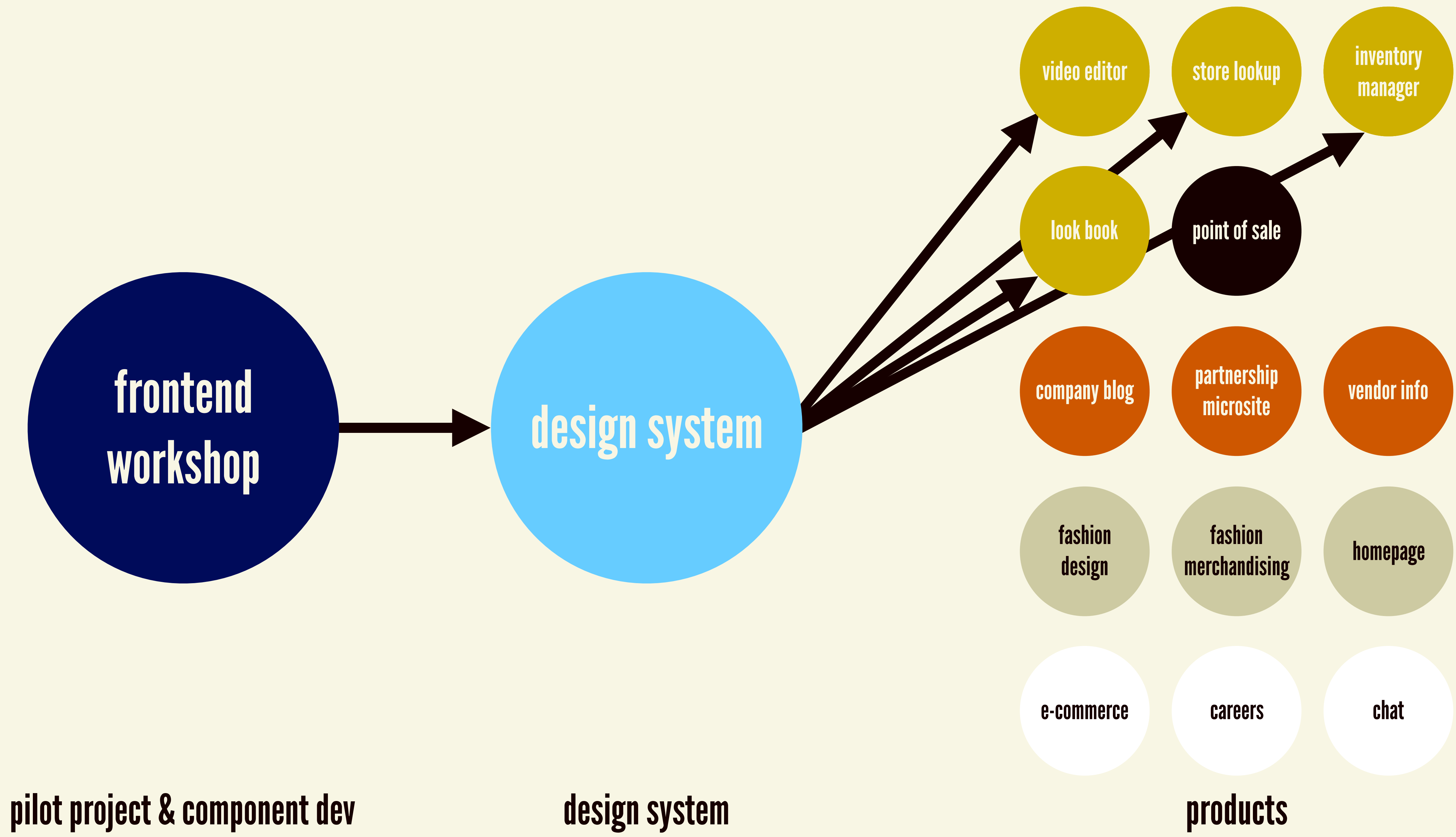
ads via Carbon

 **monday.com**

<https://danmall.me/articles/designer-developer-workflow/>



**LATHER. RINSE. REPEAT.**













**THE DESIGN SYSTEM CODE MUST BE  
INTUITIVE FOR DEVELOPERS USING IT**

# API DESIGN

▢ Molecules

▸ ▢ Advertisements

▸ ▢ Messaging

▸ ▢ Forms

▸ ▢ Layout and Containers

▸ ▢ Category

▸ ▢ Navigation

▼ ▢ Buttons

▼ ▢ Button

▢ Default

▢ Default w/ screen reader text

▢ Primary

▢ Primary button (link)

▢ Disabled

▢ Bare Icon

▢ Text Link Button

▢ **Text Link w/ Icon**

Text Link Button ▸

**JSX**

```
<Button
  iconName="triangle-right"
  iconPosition="after"
  text="Text Link Button"
  variant="link"
/>
```

Getting started	
Guidelines	>
Components	▼
Overview	
Blocks and cards	>
Buttons	▼
Overview	
Button	
Button group	
Form controls	>
Headers and footers	>
Icons	>
Interactive	>
Layout	>
Lists and collections	>
Media	>
Messaging	>
Navigation	>
Tables	>

Name	Type	Required	Description
<b>className</b>	string		CSS class names that can be appended to the component.
<b>disabled</b>	bool		Disables the field and prevents editing the contents
<b>fullWidth</b>	bool		Toggles button that fills the full width of its container
<b>hideText</b>	bool		Visually hide button text (but text is still accessible to assistive technology)
<b>href</b>	string		Link to URL. If href is present, the button will be rendered as an <a> element.
<b>iconName</b>	string		Name of SVG icon (i.e. caret-down, minus, warning)
<b>iconPosition</b>	oneOf: "before", "after"		Determines position of icon relative to button text. before places icon before button text after places icon after button text
<b>screenReaderText</b>	string		Visually hidden additional instruction text to help provide screen reader users additional context. For instance, "View details" might be the visible button text, but screenReaderText might add additional instructions such as "for confirmation number C1234567"



Submit

`variant="primary"`

Button

`variant="secondary"`

Button

`variant="link"`

Button

`size="large"`

Button

`size="small"`

Button

`fullWidth={true}`

# Component API Naming Conventions

## `items` and `item` for groups of things

- Components that get passed an array of items must use a prop called `items`
- Mapping over an array of items should use the singular `item`

## Events

- `handle[eventName]` should be used. `handleOnClick`, `handleOnFocus`, `handleOnBlur`, etc

## Variants

- `variant` should be used for primary *stylistic* variations of a component, such as (i.e. `<Card variant="bordered">` or `<Button variant="secondary">`). `variant` should be used if there is primarily one variable used to manipulate the component style.
- `theme` should be used consistently for stylistic variations that "invert" the color schemes (i.e. `theme="inverse"`) to work on a darker background.
- `size` should be used for adjusting size attributes (i.e. `<Button variant="secondary" size="small">` or `<Text passage`

## Variants

- `variant` should be used for primary *stylistic* variations of a component, such as (i.e. `<Card variant="bordered">` or `<Button variant="secondary">`). `variant` should be used if there is primarily one variable used to manipulate the component style.
- `theme` should be used consistently for stylistic *variations* that "invert" the color schemes (i.e. `theme="inverse"`) to work on a darker background.
- `size` should be used for adjusting size attributes (i.e. `<Button variant="secondary" size="small">` or `<TextPassage size="large">`). Default to `small` and `large`, with `"medium"` being the default.
- `behavior` should be used for specific behavioral variations of a pattern, such as `<Table behavior="stacked">` or `<Alert behavior="dismissable">`
- `align` should be used for aligning content, and should include `left`, `center`, `right` if needed.

## Text, Labels, Titles

- Default to `text`, such as `<Label text="First Name">` or `<Breadcrumb href="#" text="Home">`.
- For headings, default to `title`, such as `<PageHeader title="My Trips">` or `<Card title="This is my title">`

**OPEN VS LOCKED DOWN?**



An aerial photograph of a surfer riding a large, powerful wave. The water is a deep blue-green color, and a vibrant rainbow is visible in the spray of the wave. The surfer is positioned near the base of the wave, and the overall scene is dynamic and scenic.

# Hero Title

This is the hero description

16:9

**1280x720**



## HERO COMPONENT

```
<Hero  
  imgSrc="https://placeholder.com/1200x650"  
  imgAlt="Hero Alt Txt"  
  title="Hero Title"  
  description="This is the hero description"  
>
```



An aerial photograph of a surfer riding a massive, curling wave. The water is a deep, vibrant blue-green, and the wave's crest is white with foam. In the upper right corner, a faint rainbow is visible in the mist created by the breaking wave. The surfer is a small figure in the lower right, wearing a bright yellow-green wetsuit.

# Hero Title

This is the hero description

Call to action

16:9

1280x720



## ADD CALL TO ACTION TO HERO COMPONENT

```
<Hero  
  imgSrc="https://placeholder.com/1200x650"  
  imgAlt="Hero Alt Txt"  
  title="Hero Title"  
  description="This is the hero description"  
  cta="Call to action"  
>
```



An aerial photograph of a surfer riding a large, powerful wave. The water is a deep blue-green color, and the wave's crest is breaking into white foam. A faint rainbow is visible in the background, adding a vibrant touch to the scene. The surfer is positioned in the lower right quadrant of the frame, providing a sense of scale to the massive wave.

# Hero Title

This is the hero description

Primary call to action

Secondary call to action

16:9

1280x720



## ADD ANOTHER CALL TO ACTION TO HERO COMPONENT

```
<Hero
```

```
  imgSrc="https://placeholder.com/1200x650"
```

```
  imgAlt="Hero Alt Txt"
```

```
  title="Hero Title"
```

```
  description="This is the hero description"
```

```
  cta="Primary call to action"
```

```
  secondaryCta="Secondary call to action"
```

```
/>
```



An aerial photograph of a surfer riding a large, powerful wave. The water is a deep blue-green color, and the wave's crest is breaking into white foam. The surfer is positioned near the base of the wave, leaving a white wake behind them. The overall scene is dynamic and captures the raw power of the ocean.

# Hero Title

This is the hero description

*Sign up*

Action

16:9

1280x720



## MAKING THE HERO MORE COMPOSABLE

```
return (  
  <div className="c-hero">  
    <img className="c-hero__img" src={imgSrc} alt={imgAlt} />  
    <div className="c-hero__body">  
      <h2 className="c-hero__title">{title}</h2>  
      <p className="c-hero__description">{description}</p>  
      <div className="c-hero__children">  
        {children}  
      </div>  
    </div>  
  </div>  
</div>  
>);
```

# MAKE THE HERO MORE COMPOSABLE

```
return (
  <div className="c-hero">
    <img className="c-hero__img" src={imgSrc} alt={imgAlt} />
    <div className="c-hero__body">
      <h2 className="c-hero__title">{title}</h2>
      <p className="c-hero__description">{description}</p>
      <div className="c-hero__children">
        {children}
      </div>
    </div>
  </div>
);
```

**put stuff here**



# Hero Title

This is the hero description

**box to put stuff**

16:9

**1280x720**



## HERO COMPONENT WITH CHILDREN

```
<Hero>  
  <InlineForm />  
</Hero>
```

## HERO COMPONENT WITH CHILDREN

```
<Hero>  
  <Button />  
</Hero>
```

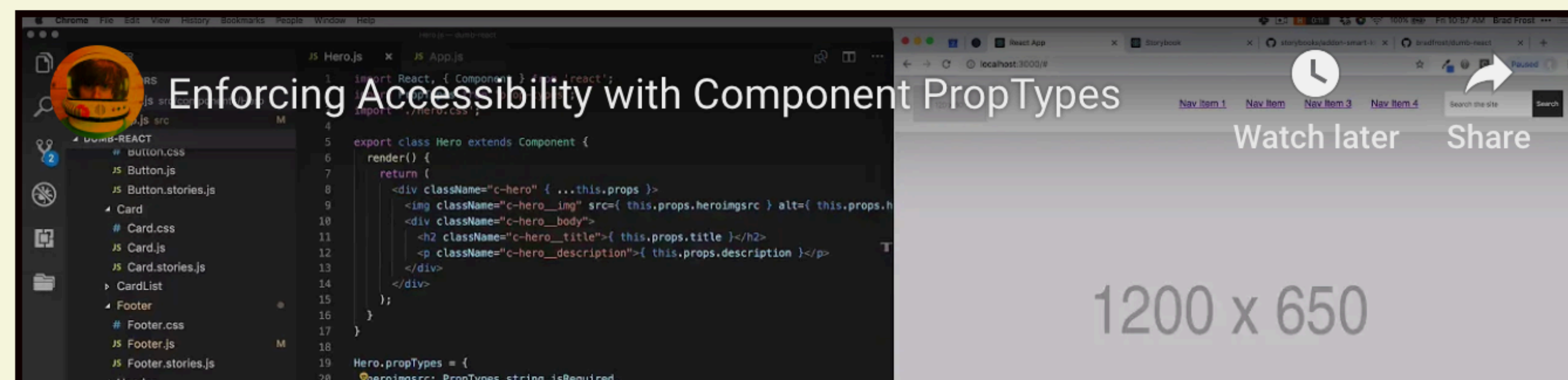
## HERO COMPONENT WITH CHILDREN

```
<Hero>  
  <WhateverYouWant />  
</Hero>
```



**BAKED-IN BEST PRACTICES**

# enforcing accessibility best practices with component proptypes



<http://bradfrost.com/blog/post/enforcing-accessibility-best-practices-with-component-proptypes/>

## VALIDATING WITH PROTOTYPES

```
Hero.propTypes = {  
  imgSrc: PropTypes.string.isRequired,  
  imgAlt: PropTypes.string.isRequired,  
  title: PropTypes.string.isRequired,  
  description: PropTypes.string  
}
```





Elements

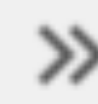
Console

Audits

Sources

React

Network



1



top



Filter

Default levels



1 hidden



✖ ▶ Warning: Failed prop type: The prop `heroimgalt` is marked as required in [index.js:2178](#)  
`Hero`, but its value is `undefined`.  
in Hero (at App.js:18)  
in App (at [src/index.js:6](#))



# enforcing accessibility best practices with automatically- generated ids

One of the best things about design systems is **you can create**  
**components that have design, development, accessibility, responsive,**

<http://bradfrost.com/blog/post/enforcing-accessibility-best-practices-with-automatically-generated-ids/>

## A LABEL NEEDS A `FOR` ATTRIBUTE

```
<label for="first-name">First Name</label>  
<input type="text" id="first-name" />
```



**Choose a password**

Your password must be at least 8 characters long

## ADDING FIELD INSTRUCTIONS WITHOUT ARIA-DESCRIBEDBY

```
<label for="password">Choose a password</label>  
<input type="password" id="password" />  
<p>Your password must be at least 8 characters  
long</p>
```

**Choose a password**

Your password must be at least 8 characters long



## ADDING ARIA-DESCRIBEDBY TO A FIELD

```
<label for="password">Choose a password</label>
<input type="password"
      id="password"
      aria-describedby="instructions"
/>
<p id="instructions">Your password must be at
least 8 characters long</p>
```

# AUTOMATICALLY GENERATING IDS

```
import shortid from "shortid";

class TextField extends React.Component {

  componentDidMount() {
    this.generateId=shortid.generate();
    this.generateAriaId=shortid.generate();
  }

  render() {
    const { id, label, ariaDescribedBy, instructions, ..other} = this.props;
    return (
      <div className="c-text-field">
        <label htmlFor={id || this.generateId}>{label}</label>
        <input type="text" id={id || this.generateId}
          aria-describedby={ariaDescribedBy || this.generateAriaId} />
        <p id={ariaDescribedBy || this.generateAriaId}>{ instructions }</p>
      </div>
    )
  }
}
```



## IMPLEMENTING A TEXT FIELD COMPONENT

```
<TextField
  label="Last Name"
  id="last-name"
  ariaDescribedBy="last-name-instructions"
  instructions="Please provide your family name"
/>
```

## IMPLEMENTING A TEXT FIELD COMPONENT WITHOUT `ID` ATTRIBUTES

```
<TextField  
  label="Last Name"  
  instructions="Please provide your family name"  
>
```



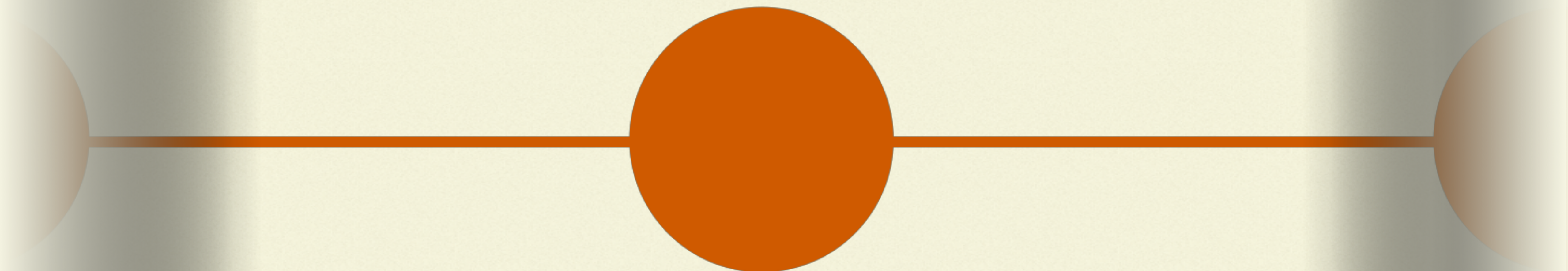
## AUTOMATICALLY ASSIGNING UNIQUE IDS IF THEY AREN'T EXPLICITLY DEFINED

```
<label for="qPbdivDEiH">Last Name</label>
<input type="text"
      id="qPbdivDEiH"
      aria-describedby="0LvdkMfqA"
/>
<p id="0LvdkMfqA">Please provide your family
name</p>
```

**YOU CAN'T SCREW IT UP!\***

\*You can probably still screw it up if you try really hard.





**Build**

**Launch**

**Maintain**



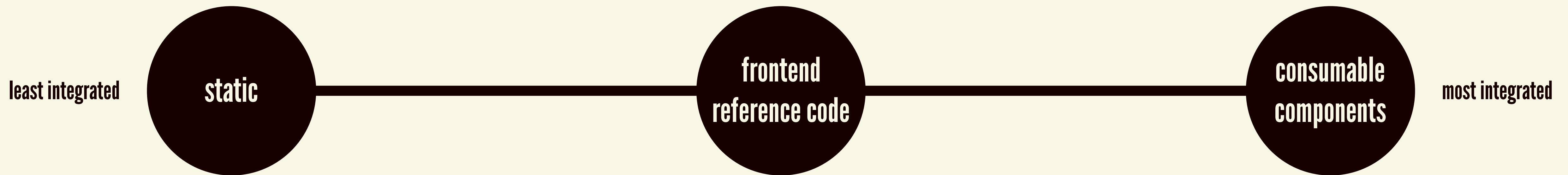
**WHAT DOES IT MEAN TO LAUNCH A  
DESIGN SYSTEM?**



**A design system isn't a project with an end, it's the origin story of a living and evolving product that'll serve other products.**

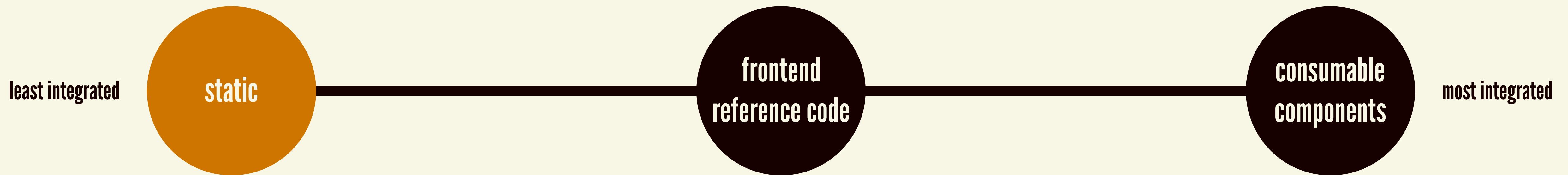
**-Nathan Curtis**

# SPECTRUM OF INTEGRATION





# SPECTRUM OF INTEGRATION







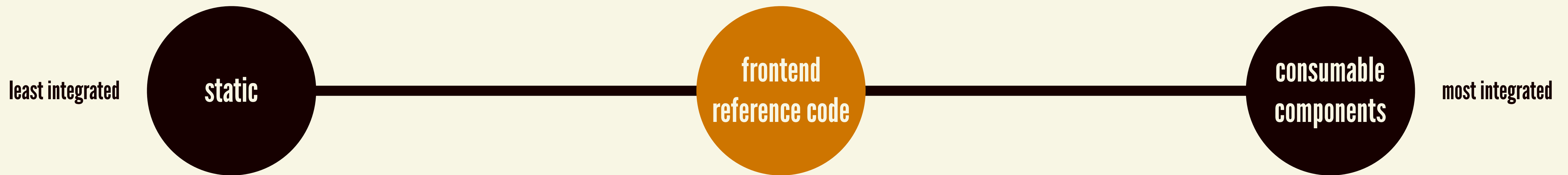
comp



production



# SPECTRUM OF INTEGRATION

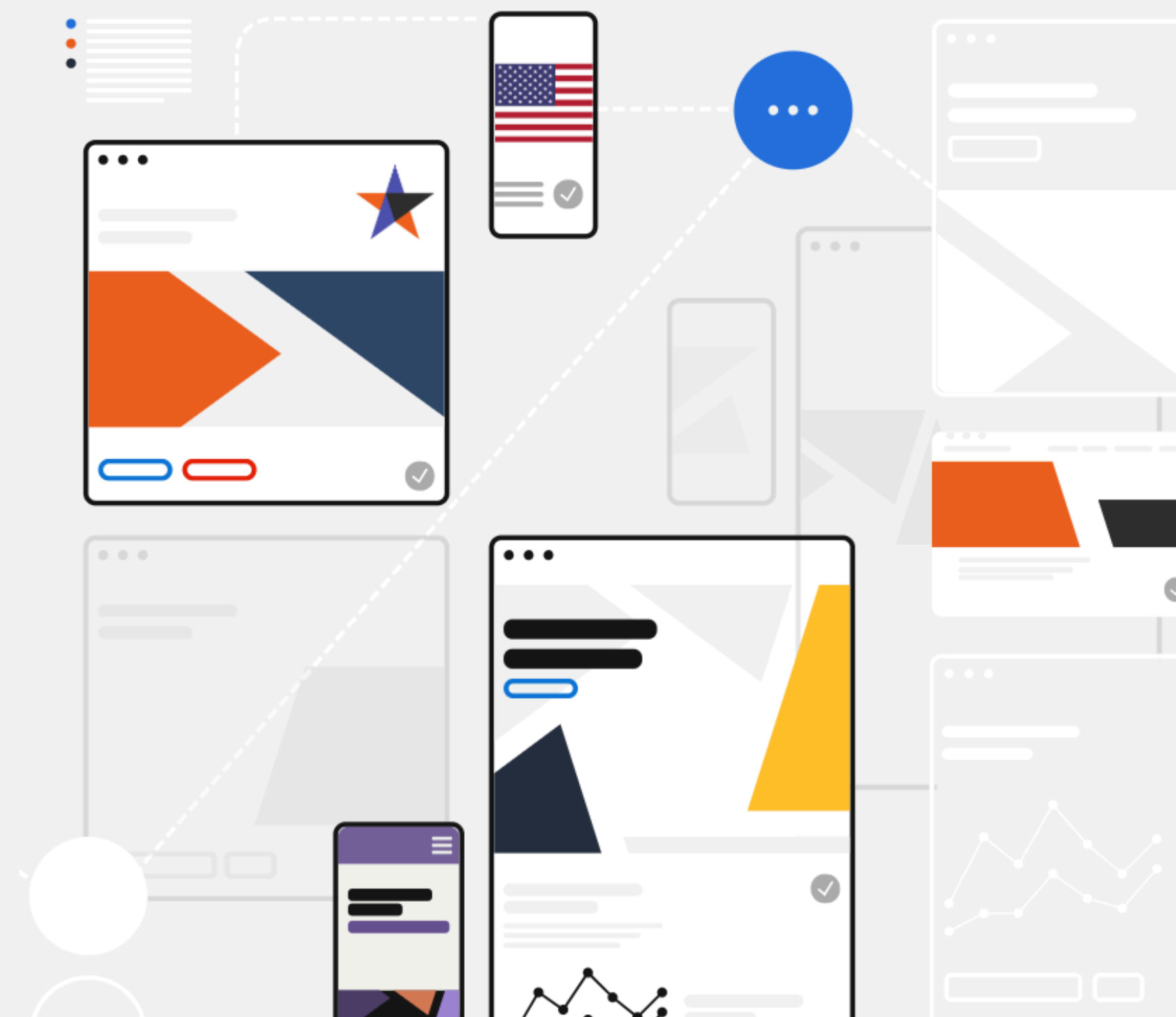


# A design system for the federal government.

We make it easier to build accessible, mobile-friendly government websites for the American public.

Learn about USWDS 2

Migrate to v2.0.1





**30,000 FREAKING WEBSITES**

## ▲ USWDS-2.0.1

▸ CSS

▸ fonts

▸ img

▸ js

▸ SCSS

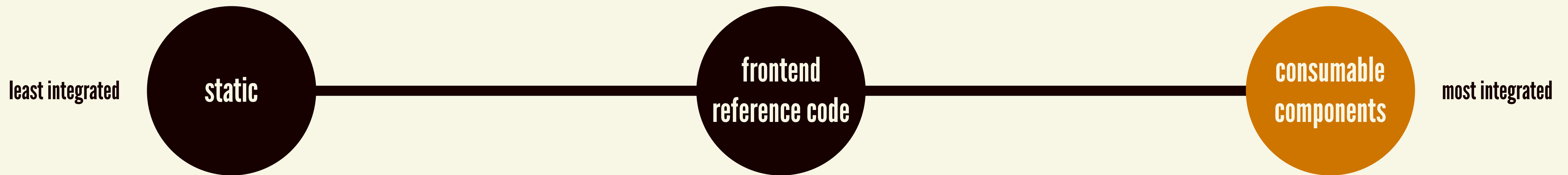
↓ CONTRIBUTING.md

↓ LICENSE.md

ⓘ README.md



# SPECTRUM OF INTEGRATION



Getting started ▾

Guidelines ▾

Components ▲

Overview

Component status

Accordion

Breadcrumb

Button

Checkbox

Code snippet

Content switcher

Data table

Date picker

Dropdown

File uploader

Form

# Accordion

Section 1 title	▾
Section 2 title	▾
Section 3 title	▾
Section 4 title	▾

Vanilla JS   [React ↗](#)   [Angular ↗](#)   [Vue ↗](#)   [CodePen ↗](#)

Code:



```
<!-- Copyright IBM Corp. 2016, 2018 This source code is licensed under the Apache-2.0 license found in t

<ul data-accordion class="bx--accordion">
  <li data-accordion-item class="bx--accordion__item">
    <button class="bx--accordion__heading" aria-expanded="false" aria-controls="pane1">
      <svg focusable="false" preserveAspectRatio="xMidYMid meet" style="will-change: transform;" xml
```

<https://www.carbondesignsystem.com/components/accordion/code>



# DEPLOYING A DESIGN SYSTEM

npm is the package manager for javascript

Popular libraries

lodash

request

chalk

react

express

commander

moment

debug

async

prop-types

bluebird

react-dom

Discover packages

IoT

mobile

front end

backend

robotics

css

testing

cli

documentation

math

coverage

frameworks

By the numbers

Packages

976,060

Downloads · Last Week

9,545,122,212

Downloads · Last Month


41,629,569,299

FAST, RELIABLE, AND SECURE

DEPENDENCY MANAGEMENT.


GET STARTED

INSTALL YARN

 Star 35,639

Stable: [v1.15.2](#) • Release Candidate [v1.16.0](#)

Node: [^4.8.0](#) || [^5.7.0](#) || [^6.2.2](#) || [>=8.0.0](#)



Dependency Manager for PHP

Latest: v1.8.5

Getting Started


Download

Documentation

Browse Packages

Issues

GitHub



NEWSGEMSGUIDESCONTRIBUTE  
SIGN IN


Find, install, and publish  
RubyGems.


Advanced Search →

34,524,882,624

DOWNLOADS & COUNTING

Install RubyGems

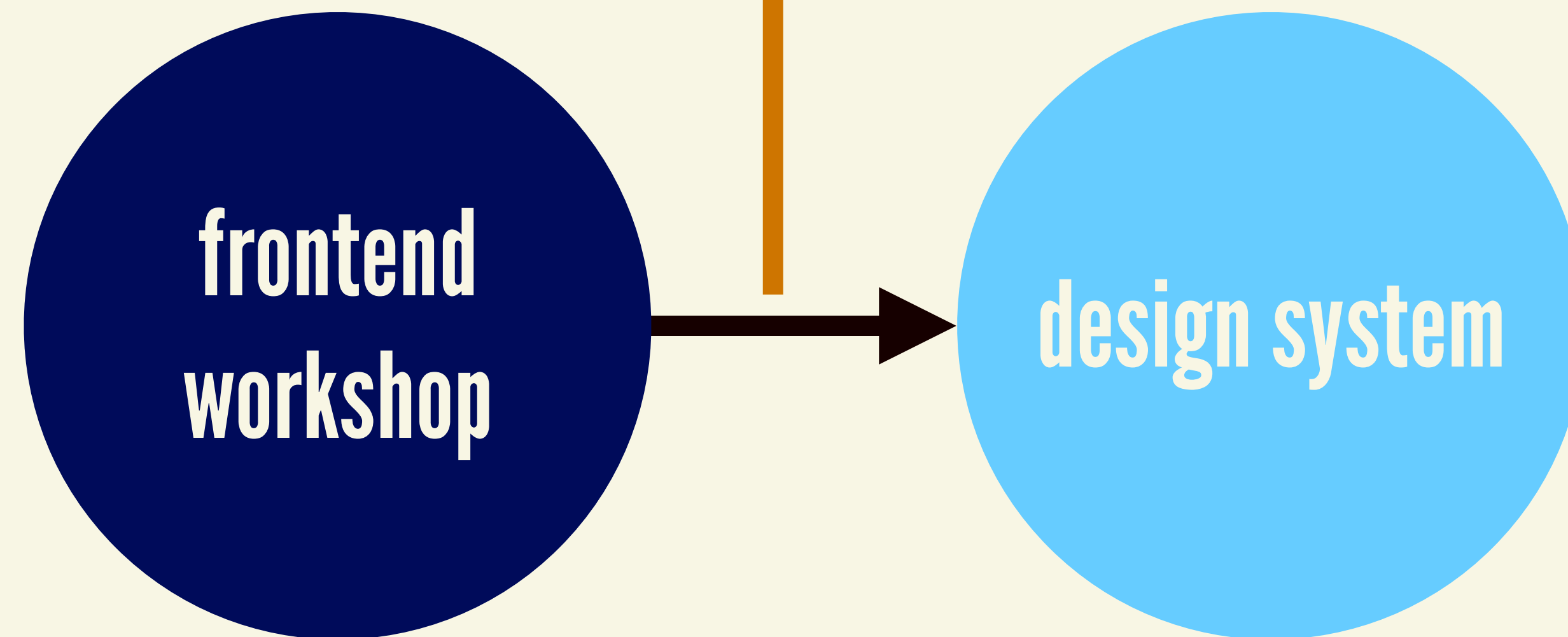
 STATUS

 UPTIME

RubyGems.org is the Ruby community's gem hosting service. Instantly publish your gems and then install them. Use the API to find out more

STATUS

UPTIME



**pilot project & component dev**

**design system**



**products**



## ▲ dist

- ▶ components

- ▶ CSS

- ▶ fonts

- ▶ icons

**}** componentDocumentation.json

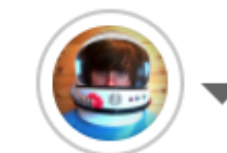
**}** package.json

**i** README.md



Search packages

Search



npm's 2019 JavaScript ecosystem survey analysis is now available! [Get your copy here »](#)

## carbon-components

10.2.0 • [Public](#) • Published 6 days ago

[Readme](#)

4 Dependencies

34 Dependents

616 Versions

## carbon-components

license [Apache-2.0](#) build [unknown](#) PRs [welcome](#)

The Carbon Design System is a series of individual styles and components, that when combined make beautiful, intuitive designs. These designs are systemic and logical, as they all follow the same universal principles.

The component library gives developers a collection of re-usable HTML and SCSS partials for building their products.

install

```
> npm i carbon-components
```

↓ weekly downloads

33,676



version

10.2.0

license

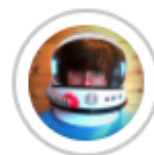
Apache-2.0

<https://www.npmjs.com/package/carbon-components>



Search packages

Search



npm's 2019 JavaScript ecosystem survey analysis is now available! [Get your copy here »](#)

@salesforce-ux/design-system

2.8.3 • Public • Published 3 months ago

Readme

0 Dependencies

13 Dependents

58 Versions

# Lightning Design System

Version: 2.8.3

## Salesforce Lightning Design System

Welcome to the [Salesforce Lightning Design System](#) brought to you by [Salesforce UX](#).

install

```
> npm i @salesforce-ux/design-system
```

weekly downloads

9,291

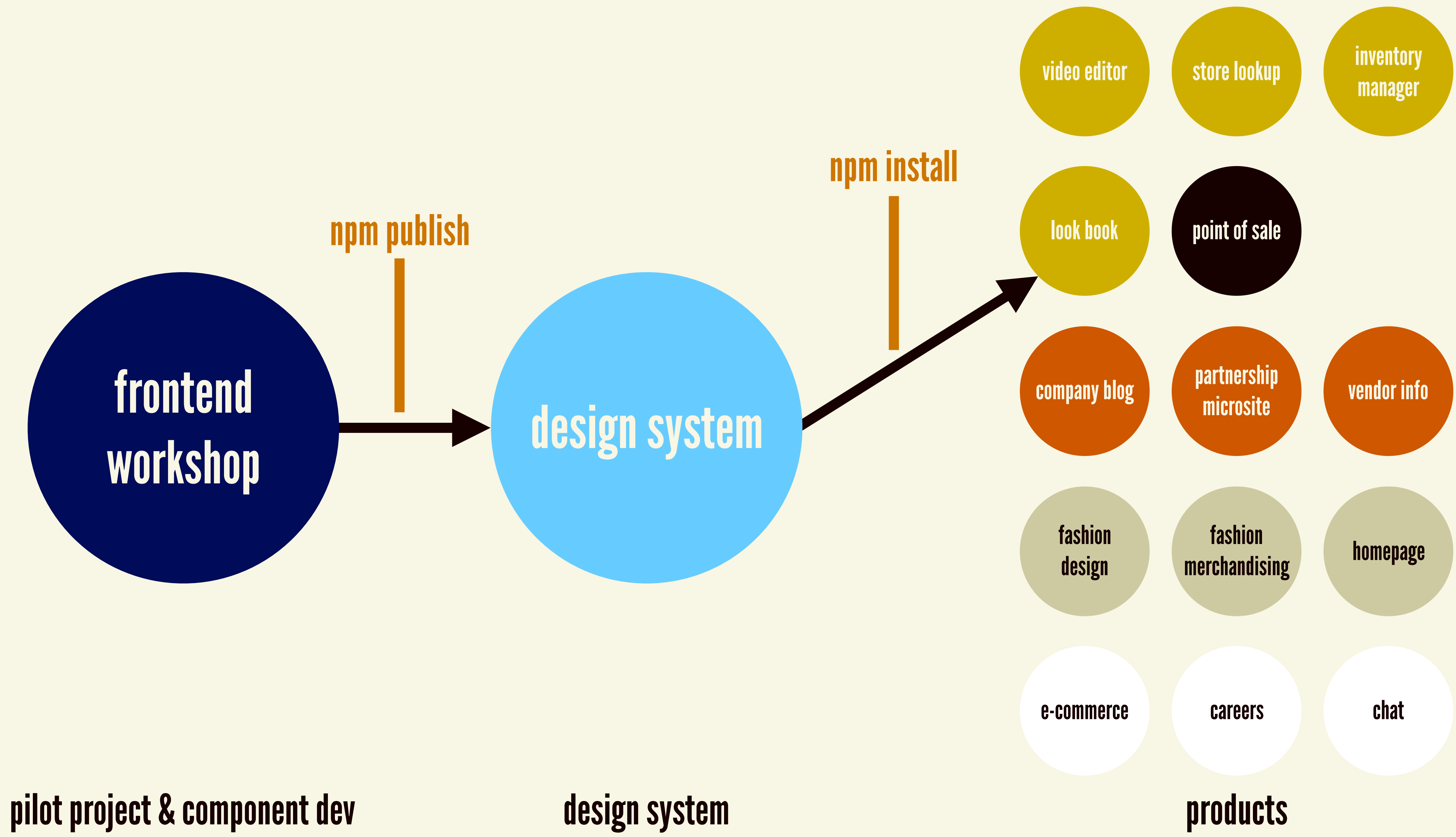
license

version

2.8.3

SEE LICENSE IN R...





## INSTALLING YOUR DESIGN SYSTEM

```
npm install your-design-system
```

## IMPORTING A DESIGN SYSTEM COMPONENT

```
import Button from your-design-system/Button
```



## IMPLEMENTING A DESIGN SYSTEM COMPONENT

```
<Button text="Book Room" variant="primary" />
```





**Maintain**



**BIG MENTAL SHIFT ALERT:**

**FROM "LET'S BUILD A WEBSITE" TO "LET'S  
MAINTAIN A PRODUCT WHICH OTHER PRODUCTS  
USE AS A DEPENDENCY."**



# VERSIONING

# Semantic Versioning 2.0.0

## Summary

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards-compatible manner, and
3. PATCH version when you make backwards-compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

## Introduction

In the world of software management there exists a dreaded place called “dependency hell.” The bigger your system grows and the more packages you integrate into your software, the more likely you are to find yourself, one day, in this pit of despair.

In systems with many dependencies, releasing new package versions can quickly become a nightmare. If the dependency specifications are too tight, you are in danger of version lock (the inability to upgrade a package without having to release new versions of every dependent package). If dependencies are specified too loosely, you will inevitably be bitten by version promiscuity (assuming compatibility with more future versions than is reasonable). Dependency hell is where you are when version lock and/or version promiscuity prevent you from

<https://semver.org/>

**v0.1.0**



# v0.1.0

patch

Fix bugs or optimize existing features without breaking changes

v0.1.0



minor

Add new features or deprecate features without breaking changes

# v0.1.0

major

Breaking changes, such as box model styles, color revisions, public API, markup, icons, fonts



**v0.30.0**

**v0.30.1**

**v0.30.2**



**v0.31.0**

v1.0.0



**A 1.0.0 designation comes with commitment.  
Freewheeling days of unstable early foundations  
are behind you.**

**—Nathan Curtis**

<https://medium.com/eightshapes-llc/versioning-design-systems-48cceb5ace4d>



**v1.1.0**

## UPDATING TO THE LATEST VERSION OF THE DESIGN SYSTEM

```
npm update your-design-system
```

[Release History](#)[Report a Bug](#)[Request a New Part](#)[▼ Development](#)[Overview](#)[AVR Testing](#)[Building a Component](#)[Coding Standards](#)[Javascript in MDS](#)[MDSWC Base API](#)[Publishing Documentation](#)[Running Local Environment](#)[Unit Testing](#)[Versioning & Breaking Changes](#) >[► Contributions](#)[► Adoption](#)[Terms of Use](#)

# Versioning & Breaking Changes

MDS uses SemVer semantic versioning. The three types of versions are:

- **Major (X.y.z)**  
Major versions contain breaking changes (defined below).
- **Minor (x.Y.z)**  
Minor versions add new features or deprecate existing features without breaking changes.
- **Patch (x.y.Z)**  
Patch versions fix defects or optimize existing features without breaking changes.

---

## Major

[Major](#)[Style](#)[Markup](#)[Web Components](#)[Constants](#)[Icons](#)[Fonts](#)

---

[Minor](#)[Style](#)[Markup](#)[Web Components](#)[Icons](#)

---

[Patch](#)[Web Components](#)





Nathan Curtis [Follow](#)

Founder of UX firm @eightshapes. Speaker. Writer. Fan of Arsenal, Hokies. Cyclist & runner. Father & husband. VT & @uchicago grad.

Aug 28 · 5 min read

# Releasing Design Systems

Delivering Interconnected Outputs to Adopters Over Time



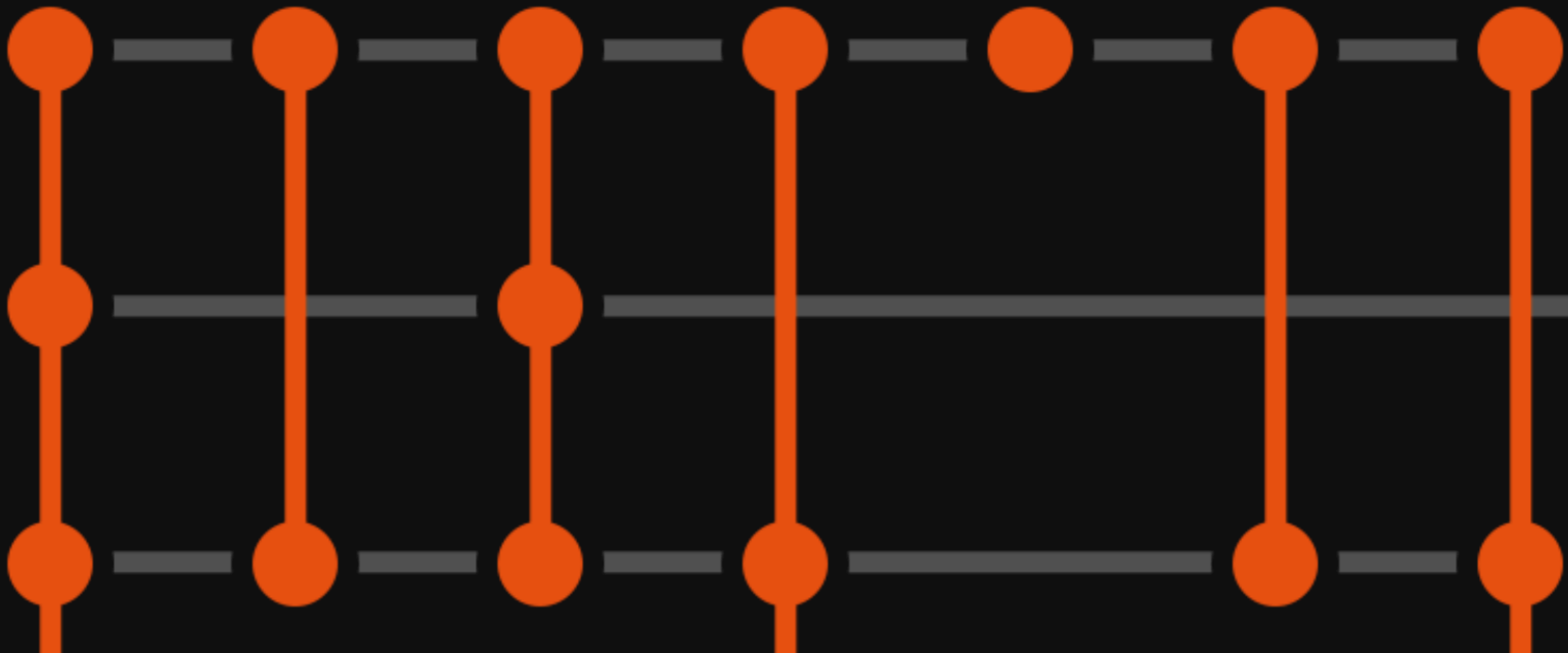
**Doc  
Site**



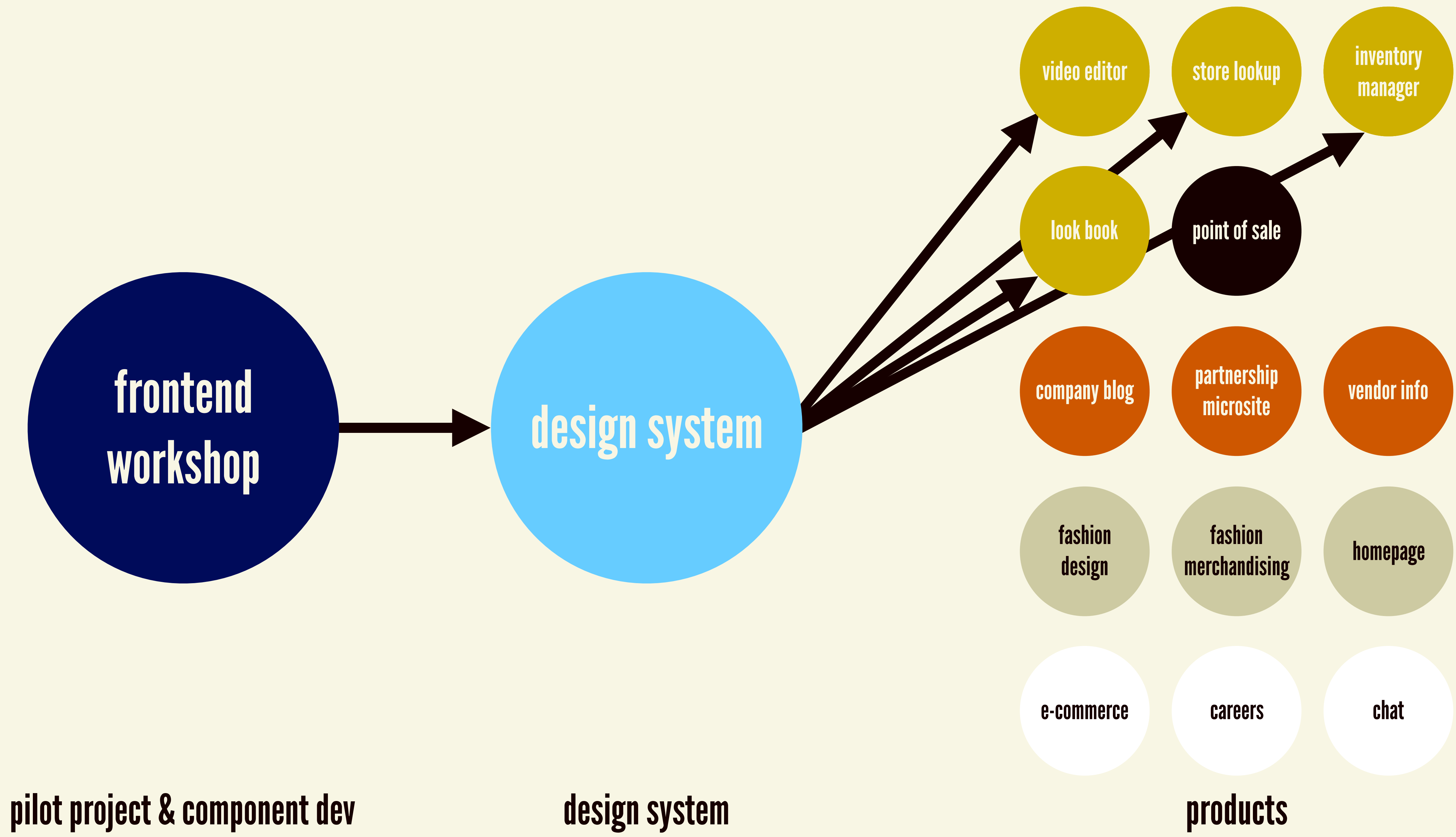
**Design  
Toolkit**

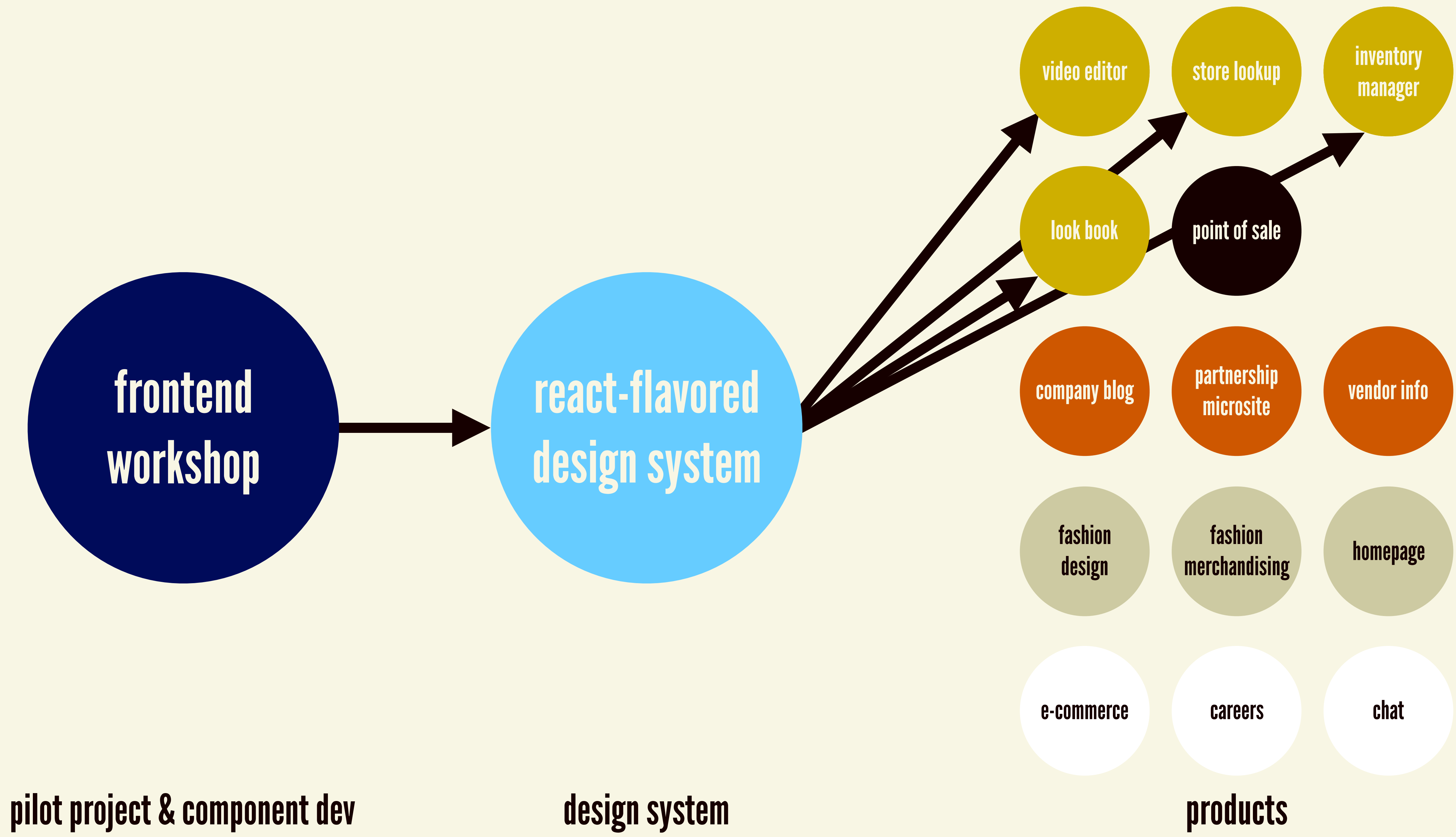


**UI Code  
Components**

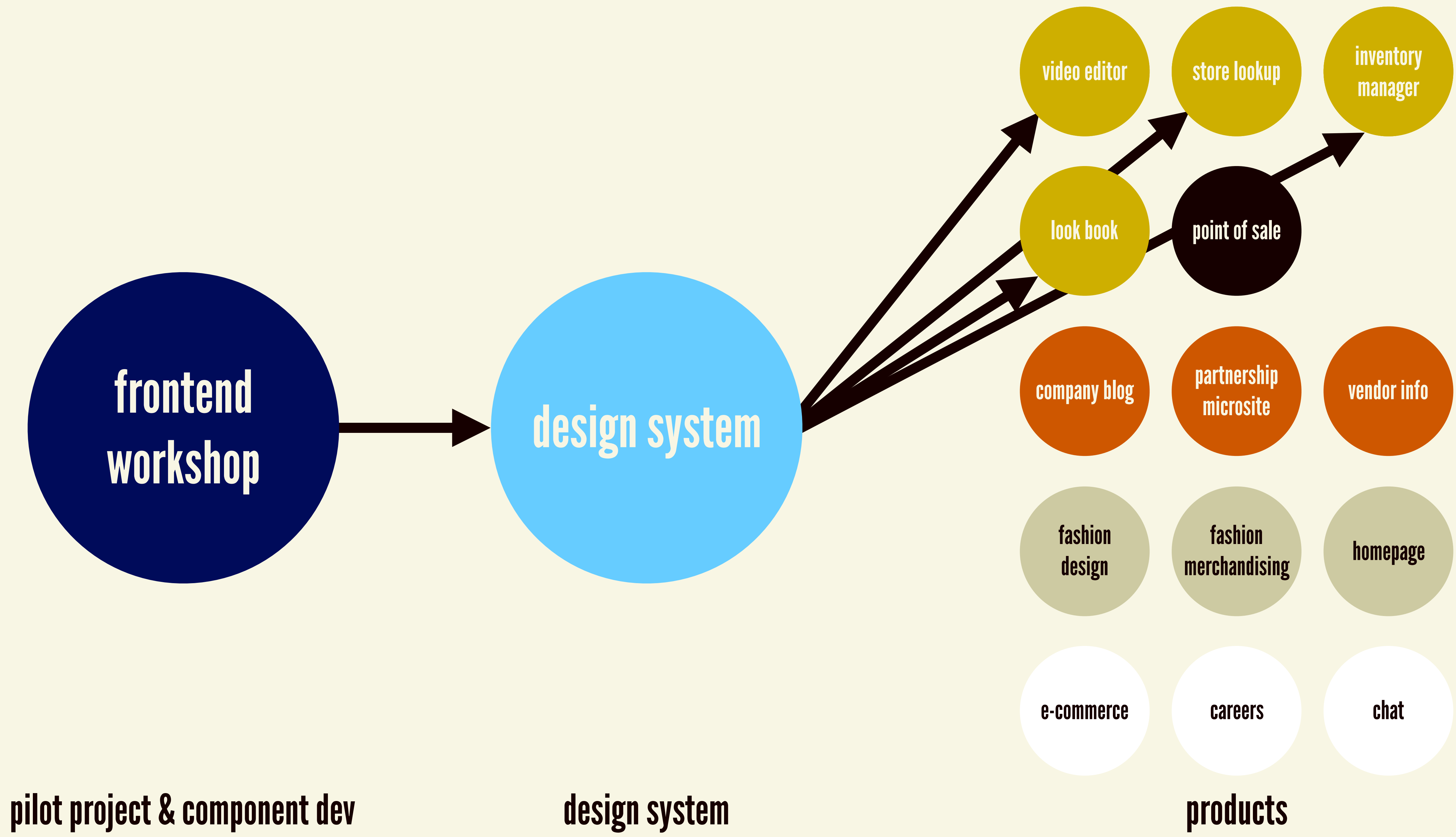


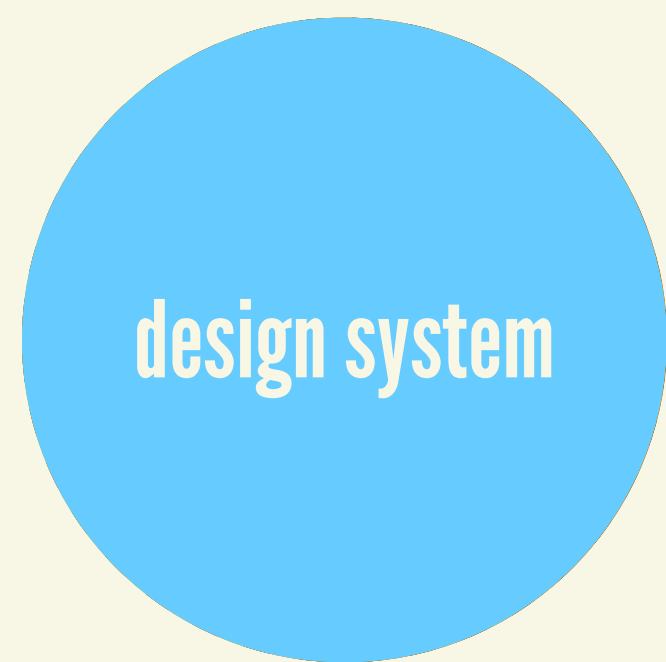
<https://medium.com/eightshapes-llc/releasing-design-systems-57fca91a23f6>









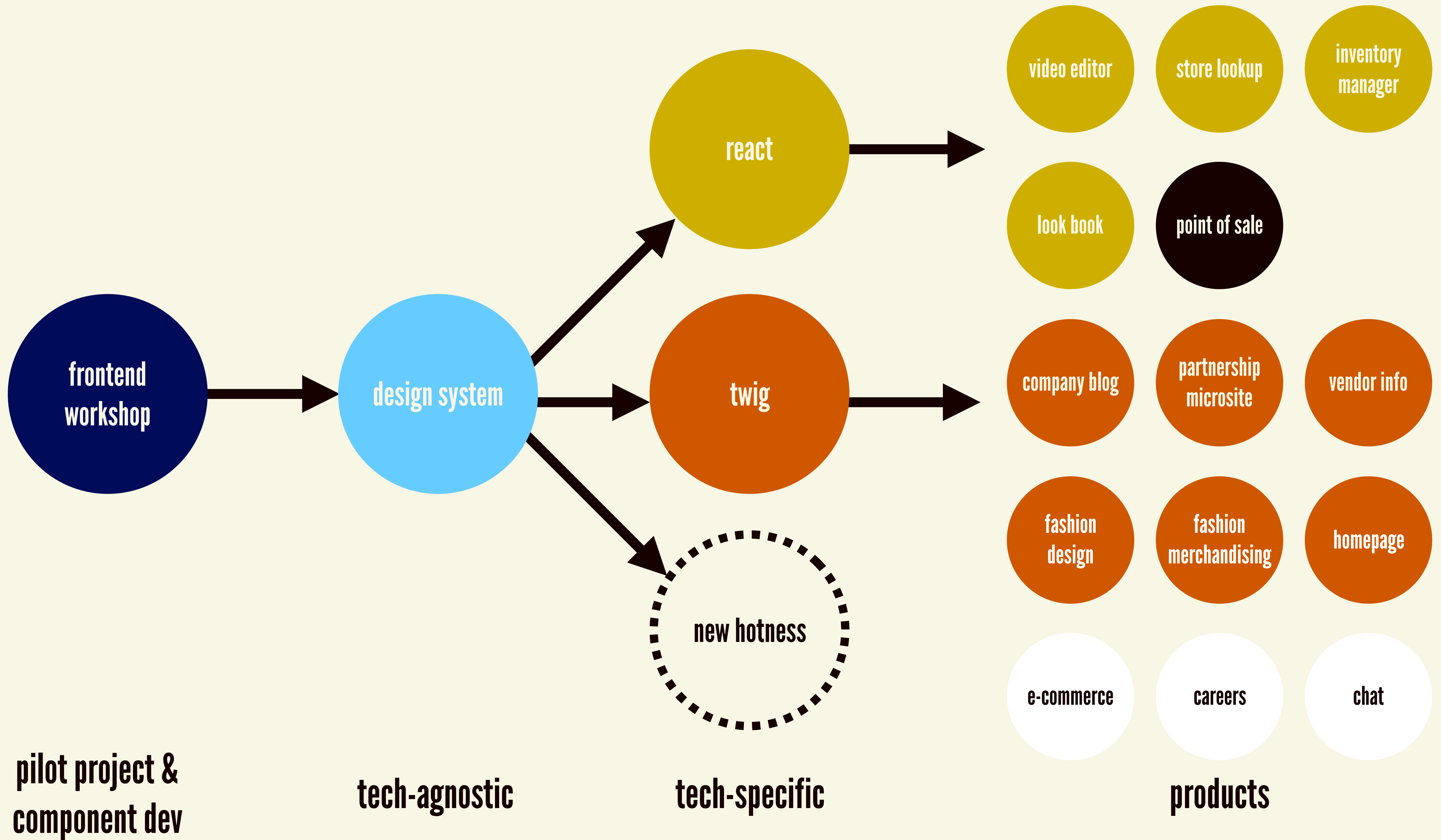


**pilot project &  
component dev**

**tech-agnostic**

**tech-specific**

**products**



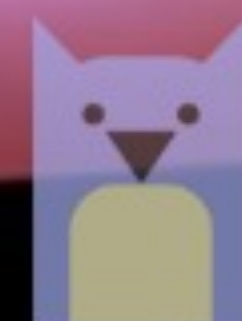


**What happens when the new hotness  
isn't the new hotness anymore?**

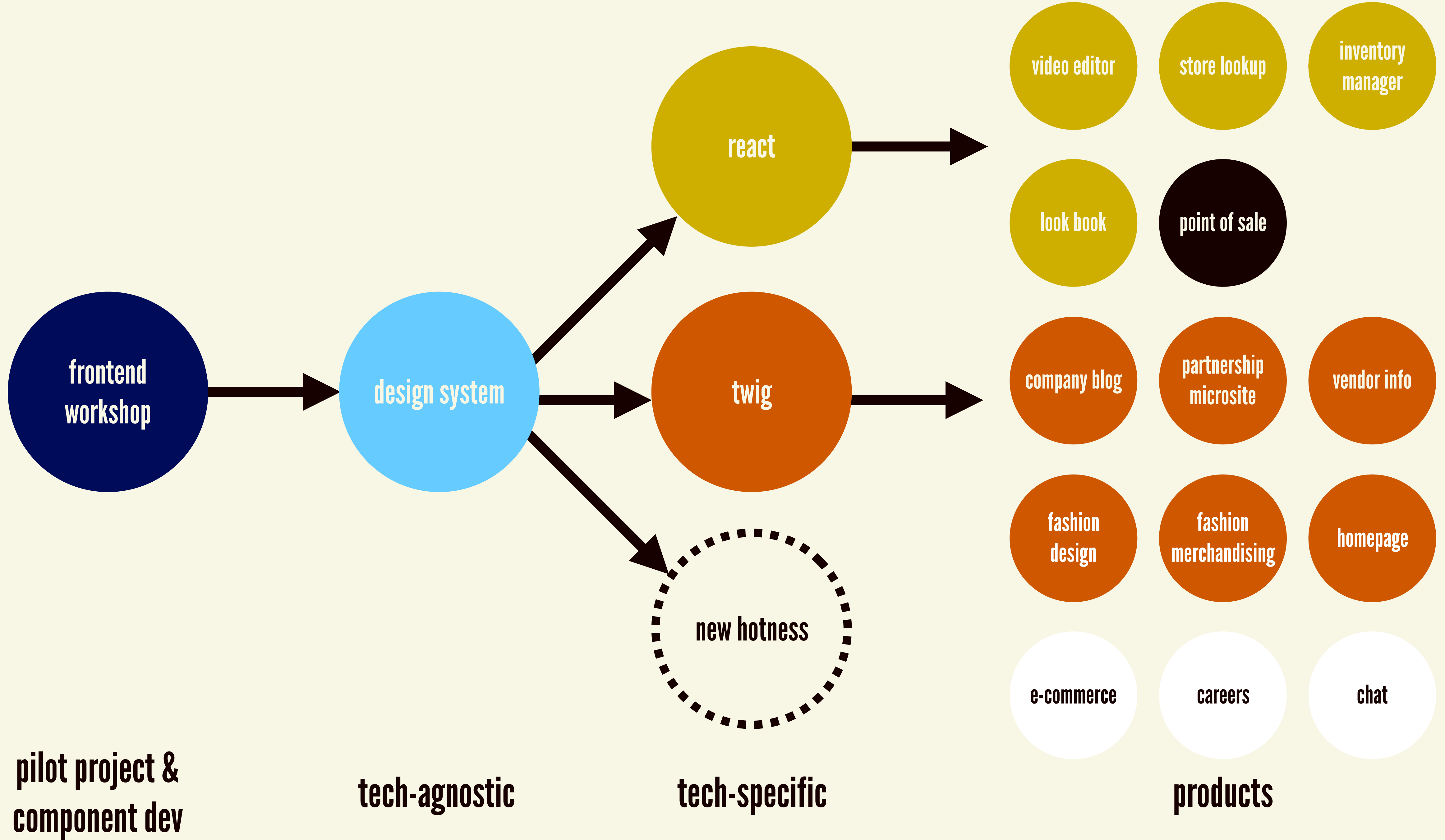
**A new foe has appeared!**

**CHALLENGER APPROACHING**

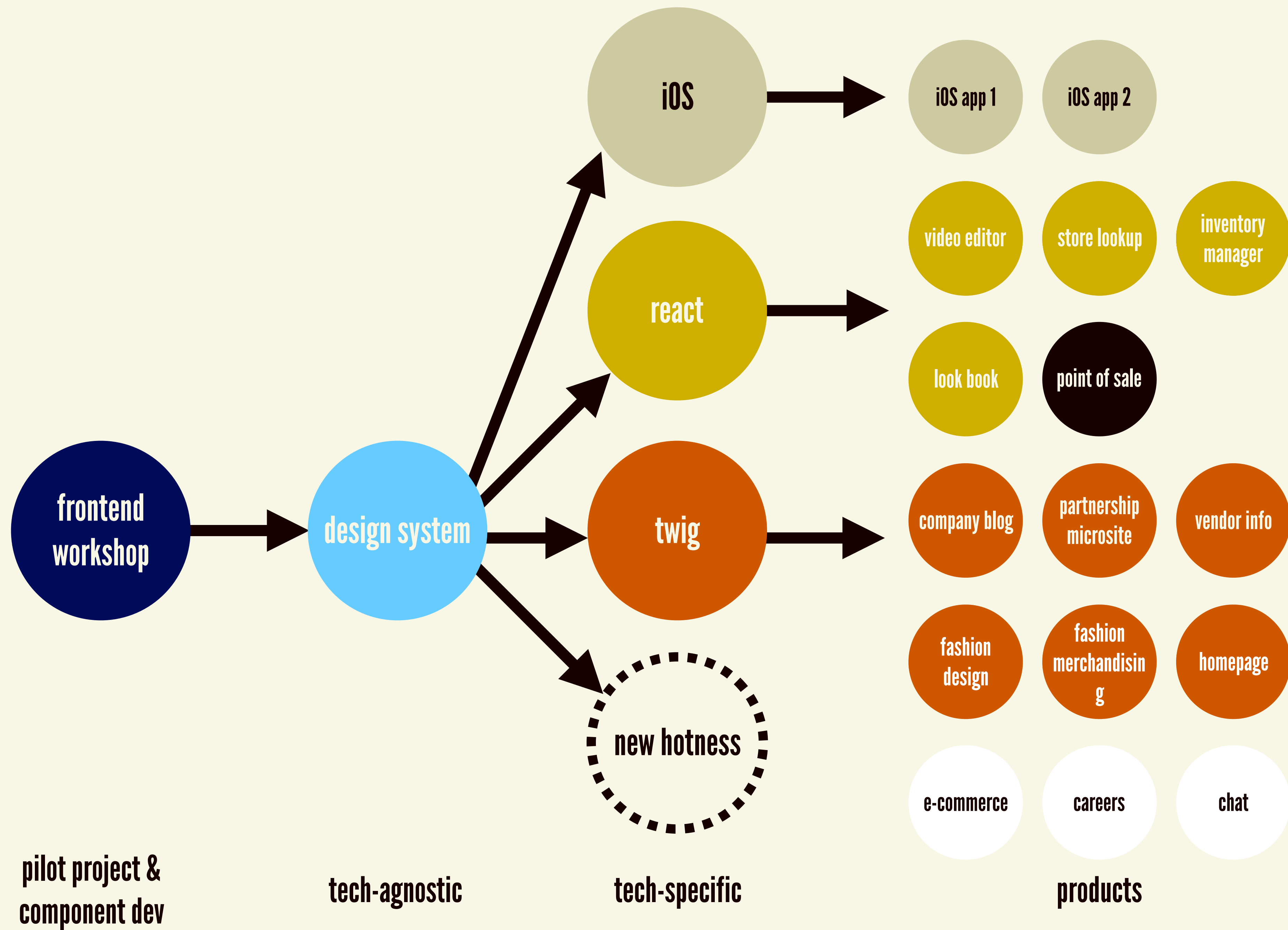
ios



**KAPWING**







# DESIGN TOKENS



# Design Tokens

Getting Started

Platforms ▶

Guidelines ▶

Components ▶

Design Tokens

Icons

Downloads

Articles

FAQ

Feedback

Format:

Lightning ▾

Design tokens are the visual design atoms of the design system – specifically, they are named entities that store visual design attributes. We use them in place of hard-coded values (such as hex values for color or pixel values for spacing) in order to maintain a scalable and consistent visual system for UI development.

Using Lightning Components? Read the Developer Guide on [Styling with Design Tokens](#).

## Background Color

Use these tokens for background colors only. Do not use these for border colors or text colors.

TOKEN

EXAMPLE

t(colorBackground)

rgb(244, 246, 249)  
#f4f6f9

### CATEGORIES

Background Color

Text Color

Border Color

Font

Font Size

Opacity

Line Height

Spacing

Radius

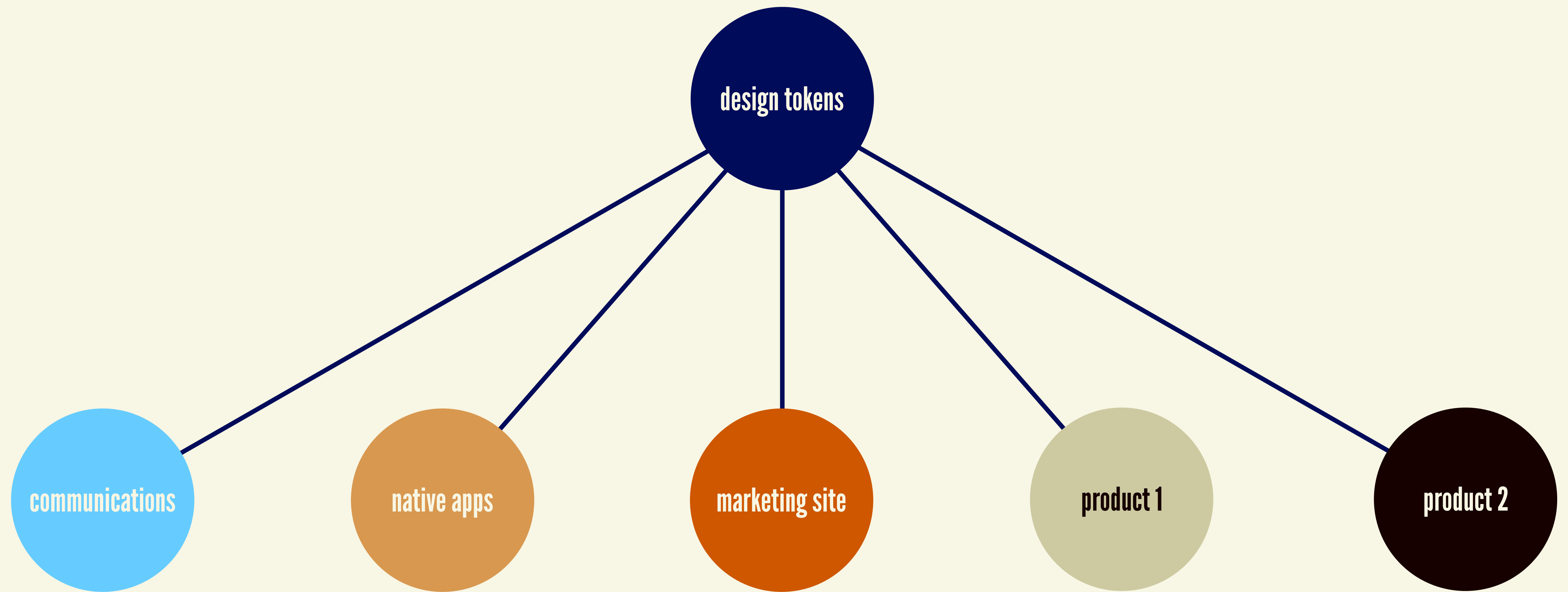
Sizing

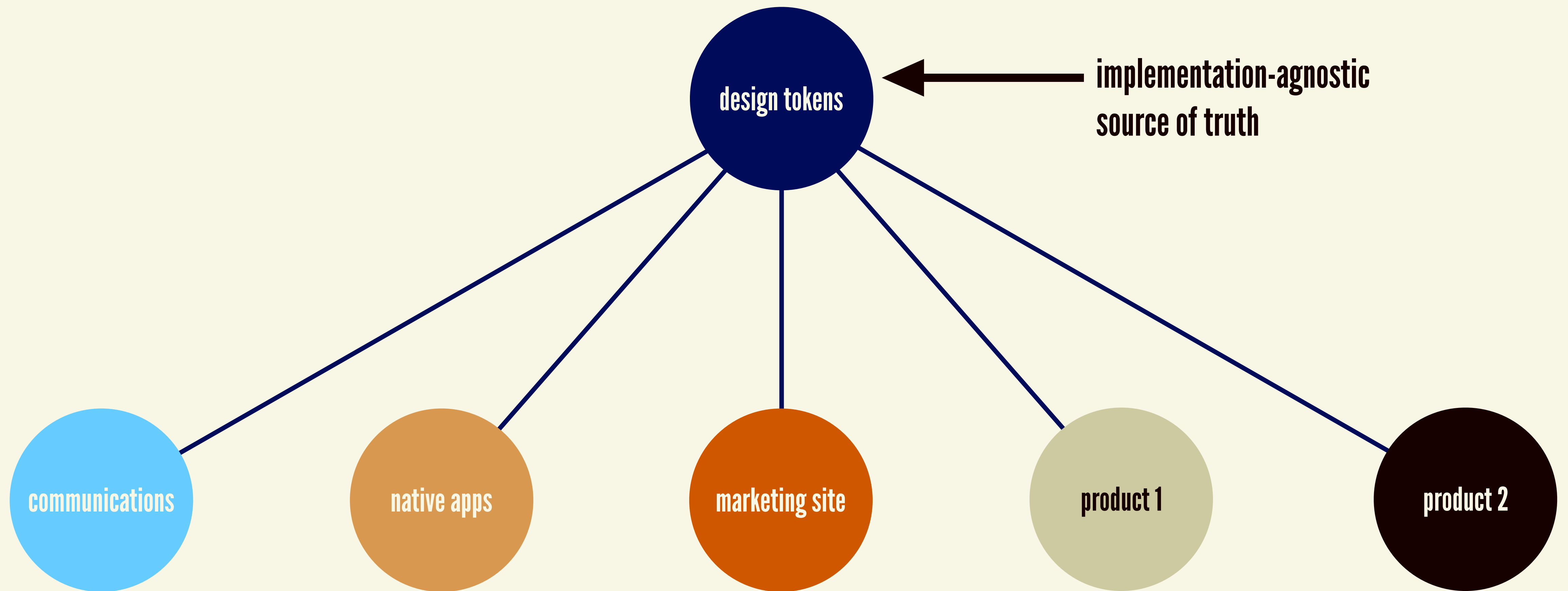
Shadow

<https://www.lightningdesignsystem.com/design-tokens/>











# DESIGN TOKEN PROPERTIES

- **Background colors**
- **Type colors**
- **Font families**
- **Font sizes**
- **Font weights**
- **Line Heights**
- **Border colors**
- **Border thicknesses**
- **Border radii**
- **Animation speeds**
- **Media queries**
- **Margins**
- **Padding**



salesforce-ux / theo

Watch 26 Star 463 Fork 33

- Code
- Issues 4
- Pull requests 1
- Projects 1
- Pulse
- Graphs

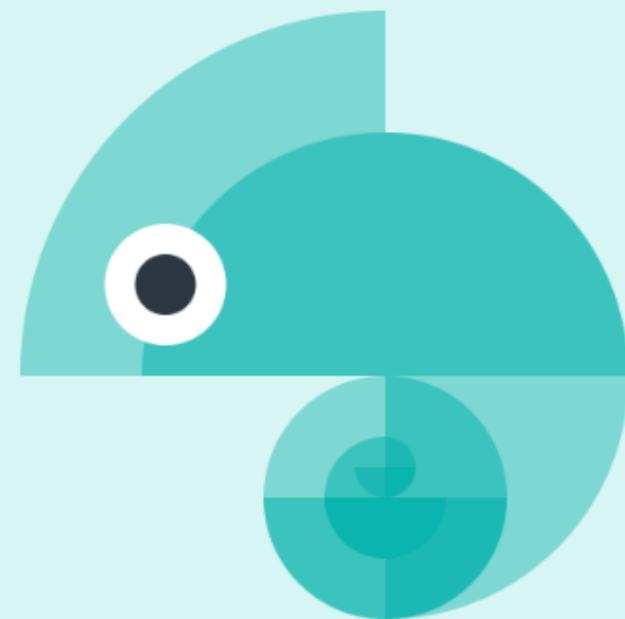
A set of Gulp plugins for transforming and formatting Design Tokens

- design-systems
- design-tokens

66 commits 3 branches 22 releases 3 contributors BSD-3-Clause

Branch: master New pull request Create new file Upload files Find file Clone or download

kaelig committed on GitHub Merge pull request #103 from salesforce-ux/greenkeeper/jest-20.0.0 ... Latest commit 7a790ce 3 hours ago		
assets	Add README.md	3 months ago
lib	Merge pull request #96 from didoo/add_ios_format	2 months ago
.editorconfig	Initial commit	3 months ago
.gitignore	Adjust ignores	3 months ago
.npmignore	Include the changelog in the package	3 months ago
.travis.yml	Initial commit	3 months ago
CHANGELOG.md	Add changelog.md	3 months ago
CONTRIBUTING.md	Update CONTRIBUTING.md	3 months ago
LICENSE.txt	Update license headers	2 months ago



# Style Dictionary

*Style once, use everywhere.*

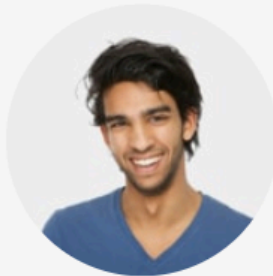
**Style Dictionary** is a build system that allows you to define styles once, in a way for any platform or language to consume. A single place to create and edit your styles, and a single command exports these rules to all the places you need them - iOS, Android, CSS, JS, HTML, sketch files, style documentation, or anything you can think of. It is available as a CLI through npm, but can also be used like any normal node module if you want to extend its functionality.

GitHub

Get Started

<https://amzn.github.io/style-dictionary>






Admin

SCOTT MATTHEWS

# DSM.

## Design System Manager

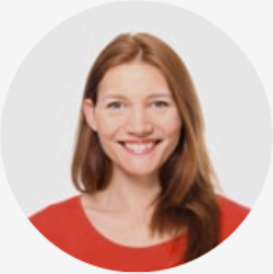
Design. Maintain. Evolve. Together.





SIGN IN

WATCH VIDEO



Editor

ALICA LEHMANN

Aa

Typeface



- Colors
- Text Styles
- Icons
- Components
- Logos
- Fonts

# ThreadAhead

...

Add library description



Colors



Text Styles



Icons



Components



Logos



Fonts

- Colors
  - Brand Colors
  - Neutral Colors
  - Utility Colors
- Text Styles
- Icons
- Components
- Logos
- Fonts

# Colors

Description

## Brand Colors

Description



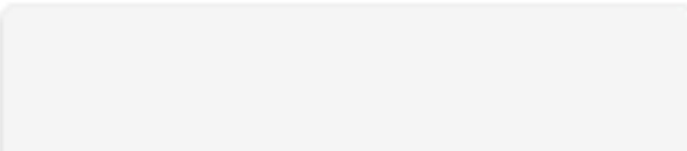
brand-blue



brand-gray

## Neutral Colors

Description





- Colors
- Text Styles
- Icons
- Components
- Logos
- Logos
  - Logos
- Fonts

# Logos

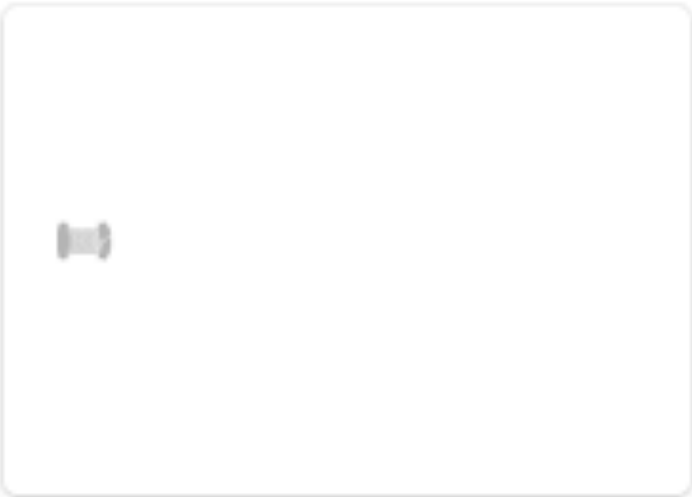
Description

# Logos

Description



threadahead



threadahead-reversed

[Colors](#)[Text Styles](#)[Icons](#)[Components](#)[Logos](#)[Fonts](#)

# Text Styles

Description

**THE QUICK BROWN FOX JU...**

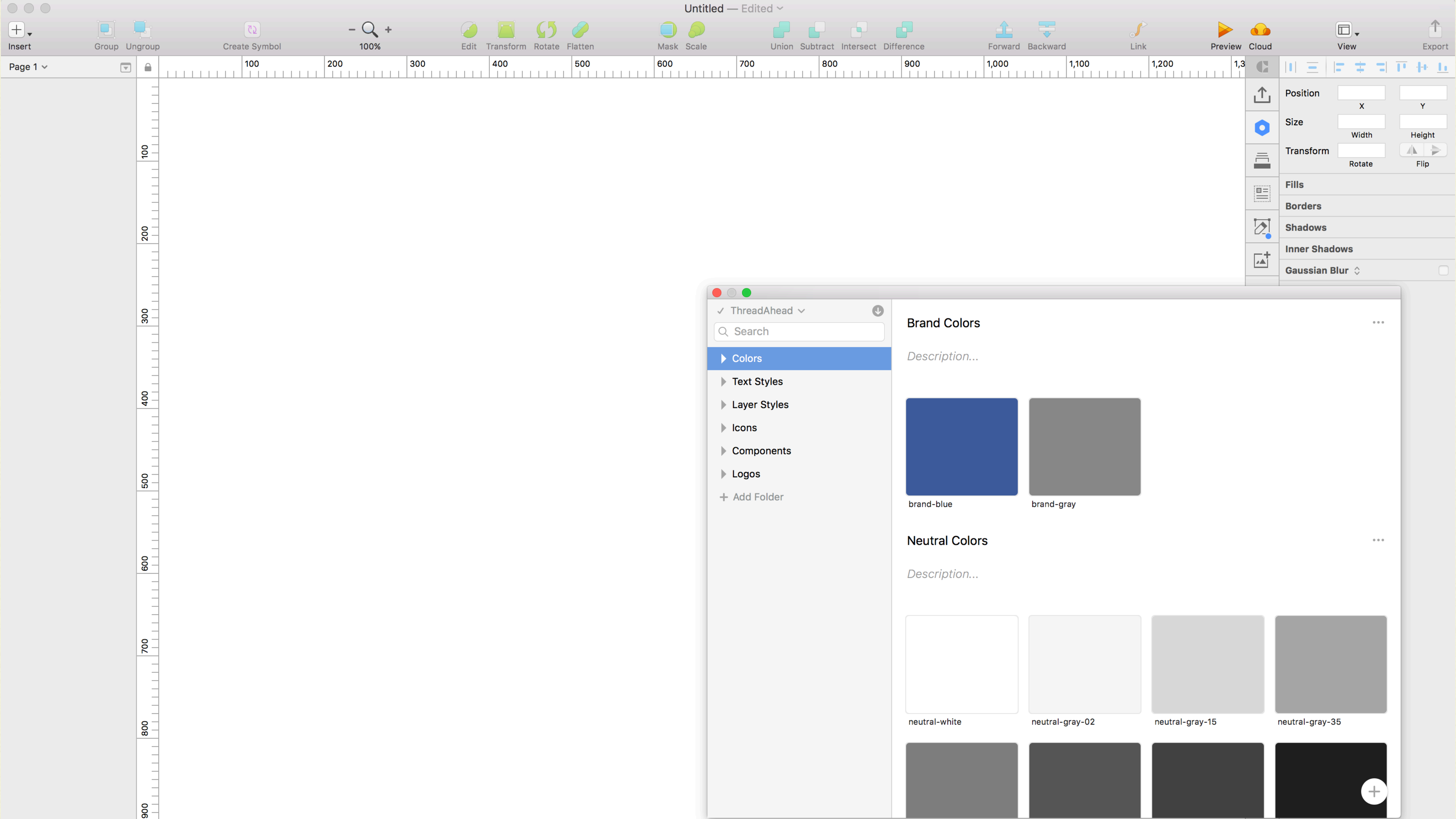
Primary Heading

**THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.**

Heading 2 Style

**THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.**

Heading 3 Style





# Sass

Overview

CSS

Sass

Less

Stylus

XML

JSON

YAML

Android

iOS

## Styles

[download](#)

[https://projects.invisionapp.com/dsm-export/brad-frost-web/thread-ahead/\\_style-params](https://projects.invisionapp.com/dsm-export/brad-frost-web/thread-ahead/_style-params)

COPY

```
/*
  Colors:
*/

/* Brand Colors */
$color-brand-blue: #3f5a9d;
$color-brand-gray: #888888;

/* Neutral Colors */
$color-neutral-white: #ffffff;
$color-neutral-gray-02: #f7f9f9;
$color-neutral-gray-15: #d9d9d9;
$color-neutral-gray-35: #a5a5a5;
$color-neutral-gray-50: #808080;
$color-neutral-gray-65: #595959;
$color-neutral-gray-73: #444444;
$color-neutral-gray-87: #222222;
$color-neutral-black: #000000;
$color-neutral-dim-50: rgba(0, 0, 0, 0.5);
$color-neutral-dim-70: #4a4a4a;

/* Utility Colors */
$color-utility-neutral: #0192d0;
$color-utility-neutral-subtle: #d3f2ff;
$color-utility-negative: #b12a0b;
$color-utility-negative-subtle: #fdded8;
$color-utility-caution: #a59b15;
$color-utility-caution-subtle: #fffecf;
```

# JSON

Overview

CSS

Sass

Less

Stylus

XML

JSON

YAML

Android

iOS

## Styles

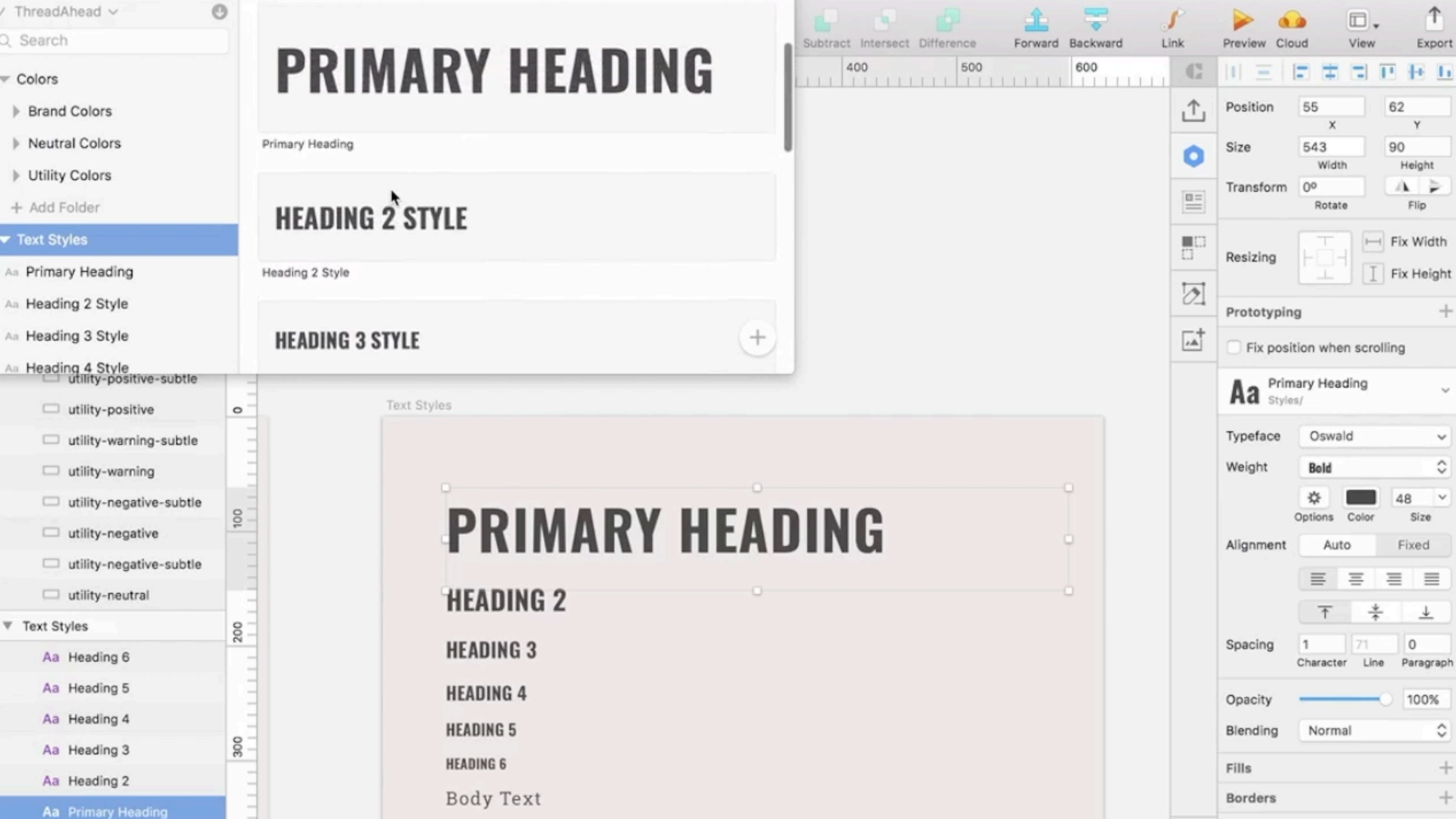
[download](#)

<https://projects.invisionapp.com/dsm-export/brad-frost-web/thread-ahead/style-data.js>

COPY

## Preview

```
{
  "list": {
    "name": "ThreadAhead",
    "organization": "brad-frost-web",
    "colors": [
      {
        "name": "Brand Colors",
        "colors": [
          {
            "name": "brand-blue",
            "value": "#3f5a9d"
          },
          {
            "name": "brand-gray",
            "value": "#888888"
          }
        ]
      },
      {
        "name": "Neutral Colors",
        "colors": [
          {
            "name": "neutral-white",
            "value": "#ffffff"
          }
        ]
      }
    ]
  }
}
```





# JSON

Overview

CSS

Sass

Less

Stylus

XML

JSON

YAML

Android

iOS

## Styles

[download](#)

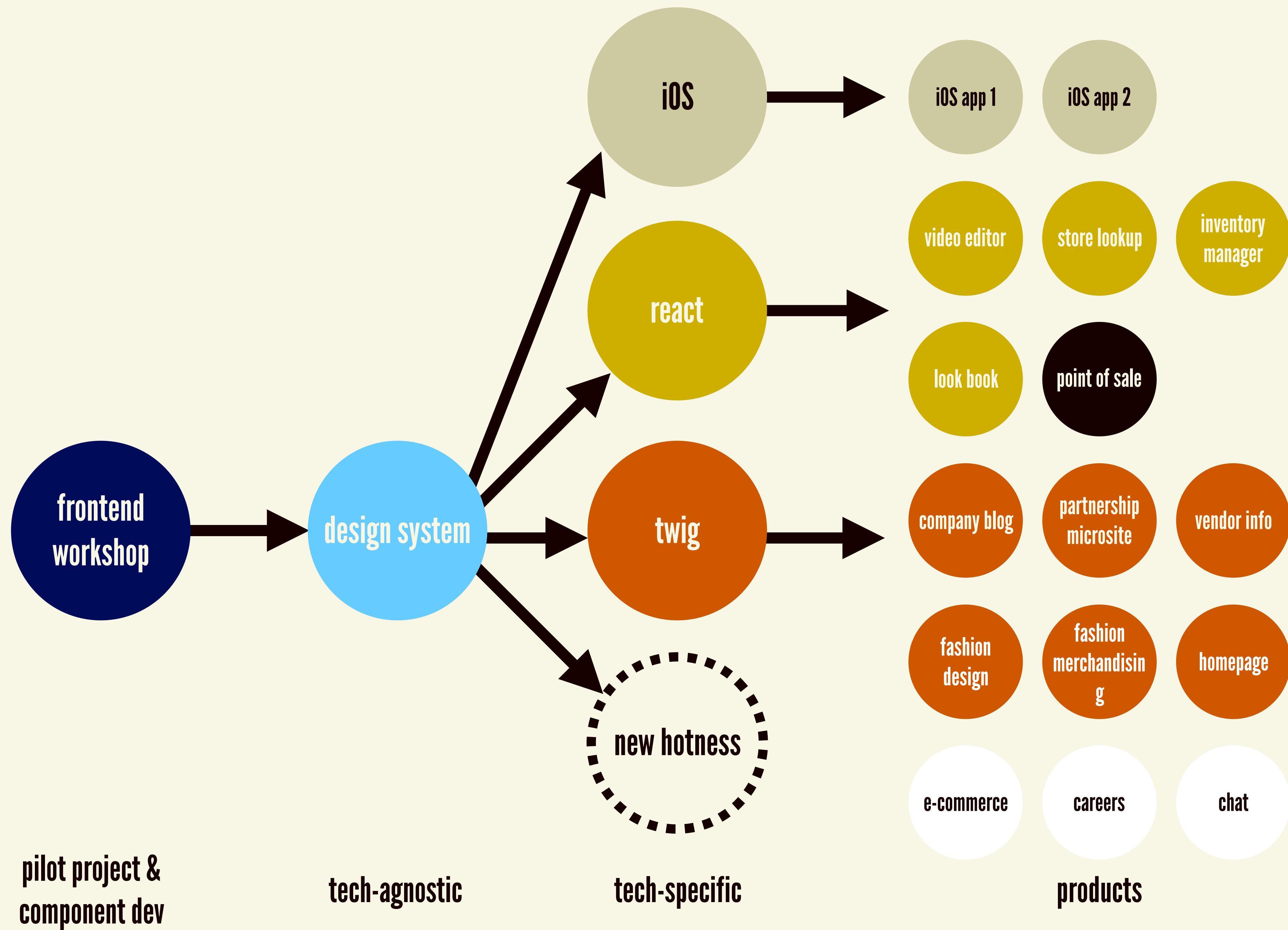
`https://projects.invisionapp.com/dsm-export/brad-frost-web/thread-ahead/style-data.js`

COPY

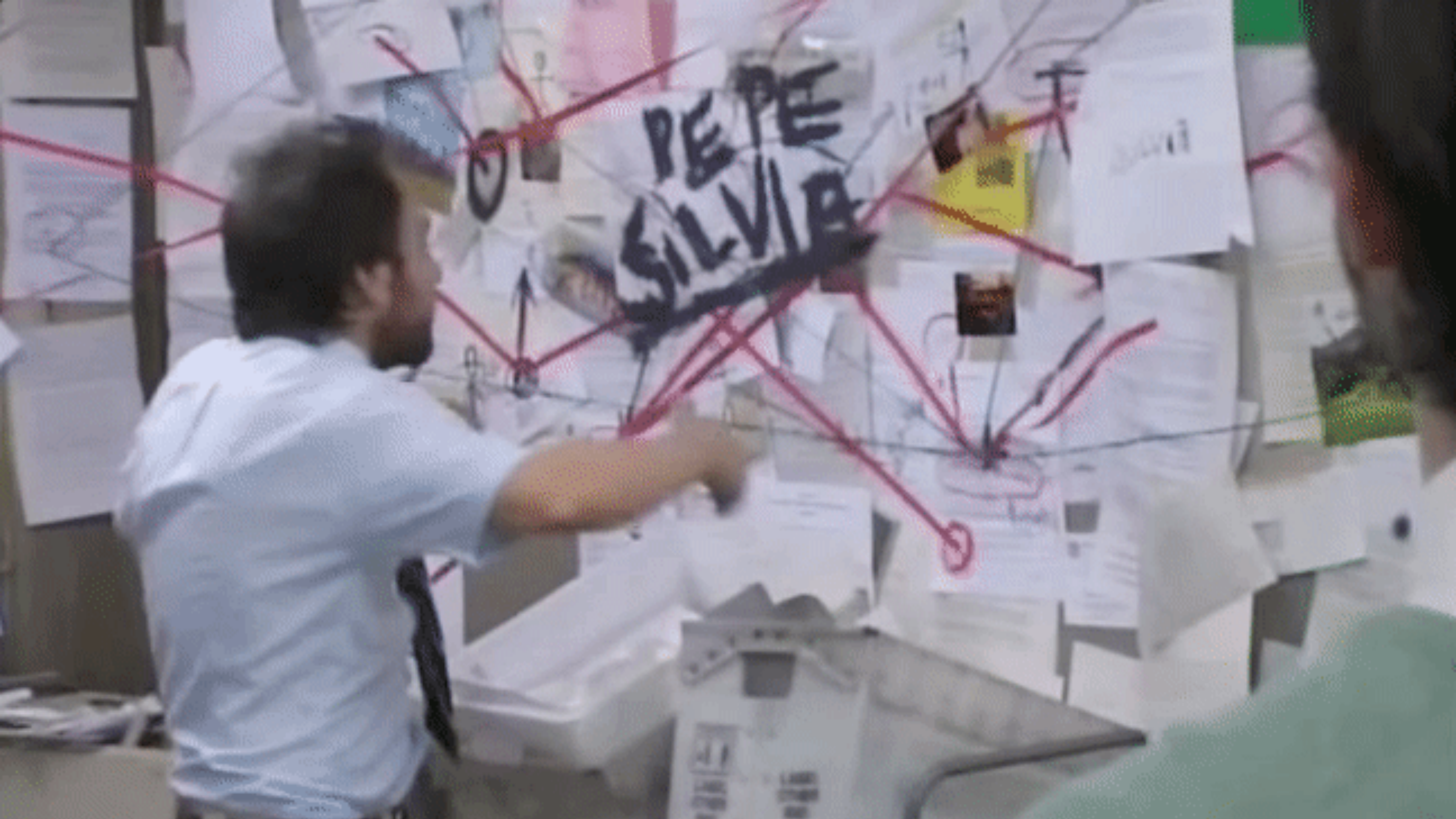
## Preview

```
{
  "list": {
    "name": "ThreadAhead",
    "organization": "brad-frost-web",
    "colors": [
      {
        "name": "Brand Colors",
        "colors": [
          {
            "name": "brand-blue",
            "value": "#3f5a9d"
          },
          {
            "name": "brand-gray",
            "value": "#888888"
          }
        ]
      },
      {
        "name": "Neutral Colors",
        "colors": [
          {
            "name": "neutral-white",
            "value": "#ffffff"
          }
        ]
      }
    ]
  }
}
```

**THIS IS HARD.**









# TAKEAWAYS

- ◎ **Your design system must live in the technologies your products use**
- ◎ **Look at your product roadmaps for design system pilot project opportunities**
- ◎ **Establish code conventions and use tooling & process to enforce them**
- ◎ **Build your design system and pilot project UI screens in a frontend workshop environment**
- ◎ **Bake best practices into reusable components & make them as rigid or flexible as you need them to be**
- ◎ **Use semantic versioning to manage ongoing design system product work**
- ◎ **Use design tokens to feed common design properties into different platforms**





**BABY STEPS**





# THANKS!

*brad@bradfrost.com | bradfrost.com*