

Keeping your design system alive

@yaili

DIGICOM, Barcelos, November 2017

About me...

make us proud



**Barbican
Centre**

 Joyent[®]

Home > Warp Records Blog >

Project overview Services Instances Versions Project manifest Settings

Services

Define services
 Review and deploy

Project manifest ✕

The project manifest defines all the services of a project, as well as other details that may be added in time. Each project has a single active manifest that defines the expected state of the entire project. [Learn more](#)

To learn how a project manifest is written, try out by pasting in a Demo manifest for a WordPress website.

[Paste a Demo manifest](#)

Upload manifest

Add services from catalogue

Project manifest

```

1  ## Consul service:
2  ## This is the service scheduler that helps discovery and communcation between the seperate services
3  ## This service is added by default as it's integral to a Container Pilot Application

```

Components

Forms

Forms

Basic

Labels:

Form:

Checkbox input

Enabled expansion Disabled expansion

Disabled Disabled

Radio input

Enabled expansion Disabled expansion

Disabled Disabled

Select

Form with error

Create an account

Input validation

Success: Success

Warning: Warning

Error: Error

Input hint validation

Success

Checkbox input validation

Enabled expansion Success

Disabled expansion Warning

Disabled Error

Radio validation

Enabled expansion Success

Disabled expansion Warning

Disabled Error

Buttons

Buttons

Types: **Primary** **Secondary** **Small** **Quick actions** **Toolbar**

States: **Focus** **Hover** **Click/Tap** **Disabled**

Primary button

Secondary button

Small button

Quick actions button primary

Quick actions button secondary

Toolbar

Card

Card

Card Container

Card Container Shaded

Card Header Collapsed

Card Header

Card Single state

Card Alert

Card Metrics

Card Profile

Card Disabled

Card Header

Card Header - Selected

Card Header - Unmap

Card Header - Full

Navigation

Navigation

Header

Subnav

Breadcrumb

Steps

ubuntu®



Vanilla Framework

Vanilla is a simple extensible CSS framework, written in Sass, by the Ubuntu Web Team.

[See the docs](#)[Vanilla on GitHub](#)

Lightweight

Vanilla contains a responsive CSS grid, basic style for HTML elements and a selection of key useful patterns and utility classes that you can extend.



Composable

Vanilla is designed to be composable — you can include the whole framework to avail of all styles or you can use only what you need for your specific project.



Open source

Anyone can contribute to Vanilla, improve it and extend it. All the code is available on GitHub and is licensed under LGPLv3 by Canonical.

Quick start

The recommended method of including Vanilla Framework in your project is via NPM. You can then simply include the framework in your SCSS files and compile.

For other methods, please see the [advanced usage docs](#).

```
npm install --save-dev vanilla-framework
export SASS_PATH=`pwd`/node_modules:${SASS_PATH}

// Add to your main build scss file the following line
@import 'vanilla-framework/scss/build'
```

Ubuntu Advantage Storage

Ubuntu Advantage Storage from Canonical embeds the proven software-defined storage (SDS) technologies of Ceph, NexentaEdge, Swift and SwiftStack, into a 24x7-supported software solution with a unique metered pricing model.

[Request a demo](#)

The challenge

The pace of data creation is exploding. As structured and unstructured data fills disks and data retention policies lengthen, every organisation must find cost-effective ways to grow their storage infrastructure.

Traditionally enterprise storage needs have been met with appliance-like SAN (Storage Area Network) and NAS (Network Attached Storage) solutions. However, they are often expensive to purchase, expand and upgrade.

Designed from the ground up for petabyte-scale deployments, Ubuntu Advantage Storage now offers a proven software-defined storage (SDS) alternative at the lowest per gigabyte price.

[Read the Ubuntu Advantage Storage Factsheet](#)



Ubuntu Advantage Storage FAQs

Answers to your most frequently asked questions about Ubuntu Advantage Storage.

[FAQ](#)

Storage options

With Ubuntu Advantage Storage, you choose between four leading open source Software-defined storage technologies.

Both technologies are available within our fully-supported reference architectures, deployed on Ubuntu 16.04 LTS alongside our award-winning cloud tools.



Swift

The high performance S3-compatible object store developed and shipped with OpenStack, ideal for storing unstructured data.



Ceph

A converged storage framework that has been designed to present object, block, and file from a single distributed computer cluster.



NexentaEdge

Block and object storage offering inline de-duplication & compression, low latency block services, instant snapshots, and enterprise grade, end-to-end data integrity.



SwiftStack

An object storage system built on OpenStack with an innovative storage controller for management and NFS and SMB/CIFS file gateways.

Metered pricing

Ubuntu: creating the world's best open source software platform

Ubuntu is a platform that spans from the PC and IoT devices to the server and the cloud. It includes a comprehensive suite of enterprise-grade tools for development, configuration, management and service orchestration.

[Learn more on ubuntu.com](#)



Canonical and Ubuntu



Canonical defines Ubuntu's strategy and drives innovation with a team of over 100 dedicated designers, developers and project managers.



Since 2004, we have ensured that Ubuntu is released on time, twice every year.



Ubuntu is the leading OS in the cloud — OpenStack is built into Ubuntu Server and Ubuntu is the reference operating system for OpenStack.



With our partner network, we make Ubuntu available globally through retail channels and on most major public clouds.

In the cloud: driving enterprise agility

Ubuntu is at the forefront of large cloud infrastructure deployments, thanks to Canonical's experience in building clouds for our customers and our involvement in the OpenStack project as a founding member. Ubuntu is also optimised and certified for the most popular public clouds — so wherever you choose to run your applications and services, you can always use Ubuntu.



Canonical has also created several important tools to help customers build, manage and scale their clouds. For telcos and enterprises, Landscape helps administrators deploy and manage Ubuntu clouds cost-effectively. And whether you are using your own

“Ubuntu remains the most popular operating system for OpenStack deployments.”

Why design systems

“A design system offers a library of visual style, components, and other concerns documented and released by an individual, team or community as code and design tools so that adopting products can be more efficient and cohesive.”

—*Nathan Curtis*

“Almost always, a design system offers a library of visual style and components documented and released as reusable code for developers and/or tool(s) for designers. A system may also offer guidance on accessibility, page layout, and editorial and less often branding, data viz, UX patterns, and other tools.”

—Nathan Curtis

- DESIGN
 - Design Palette >
 - UI Colours >
 - Destination Next Colors >
 - Typography >
- ICONS
 - Destination >
 - Interests >
 - Interface >
 - Need to Know >
 - Weather >

Colour Palette

Find closest color

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	#dbe6ec
<code>\$darkgray</code>	<code>\$titlegray</code>	<code>\$bodygray</code>	<code>\$lightgray</code>	<code>\$subduedgray</code>

PRIMARY PALETTE

#142b44	#1d508d	#297cbb	#238ad6	#0fdebd
<code>\$darkblue</code>	<code>\$navblue</code>	<code>\$lpblue</code>	<code>\$linkblue</code>	<code>\$teal</code>
#16c98d	#feef6d	#ffc83f	#fa5e5b	#bf538d
<code>\$green</code>	<code>\$yellow</code>	<code>\$orange</code>	<code>\$red</code>	<code>\$plum</code>

SECONDARY PALETTE

#684e79	#ff708e	#47a899	#8abee5	#c7e6aa
<code>\$mauve</code>	<code>\$pink</code>	<code>\$darkcyan</code>	<code>\$softblue</code>	<code>\$softgreen</code>
#cfcbafe	#582c2b	#841e1b		
<code>\$beige</code>	<code>\$maroon</code>	<code>\$darkred</code>		

COLOR LIST

	#2c3643
--	---------



LIGHTNING DESIGN SYSTEM

Create the world's best enterprise app experiences.

GET STARTED

Current release: [Winter '18 \(SLDS 2.4.4\)](#) | [Archives](#)

[Guidelines](#)

[Accessibility](#)

[Components](#)

[Utilities](#)


[Design Tokens](#)

[Icons](#)

[Help & Support](#)




NEW



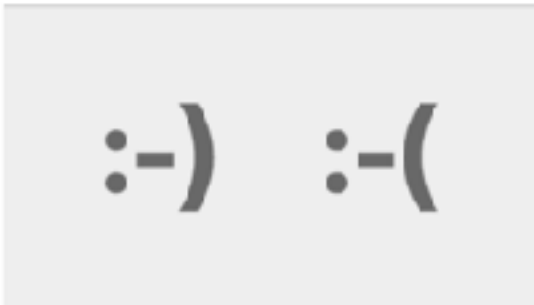
What is Design Research?
Learn how design research helps us create digital experiences that meet audience needs.

a day ago | [Articles](#)




What is GEL?

7 Apr 2016 | [Articles](#)



How to use Reactions in your content

a month ago | [How-tos](#)



Never Letting Wonder Go To Waste

17 Aug 2017 | [Articles](#)



Share Tools

22 Jun 2017 | [Design Patt...](#)



The ABCs of Motion

7 Apr 2016 | [Articles](#)



Grid

19 Feb 2016 | [Foundations](#)

GEL Design Principles

Drive Discovery

The journey's as important as the destination. And if your user veers off in a new, exciting direction along the way, all the better.

< >

MATERIAL DESIGN

Material Design is a unified system that combines theory, resources, and tools for crafting digital experiences.

[WATCH THE VIDEO](#)

GUIDELINES & RESOURCES

TOOLS

MATERIAL DESIGN

interaction, and motion under a consistent set of principles. With Material we believe product teams can realize their greatest design potential.

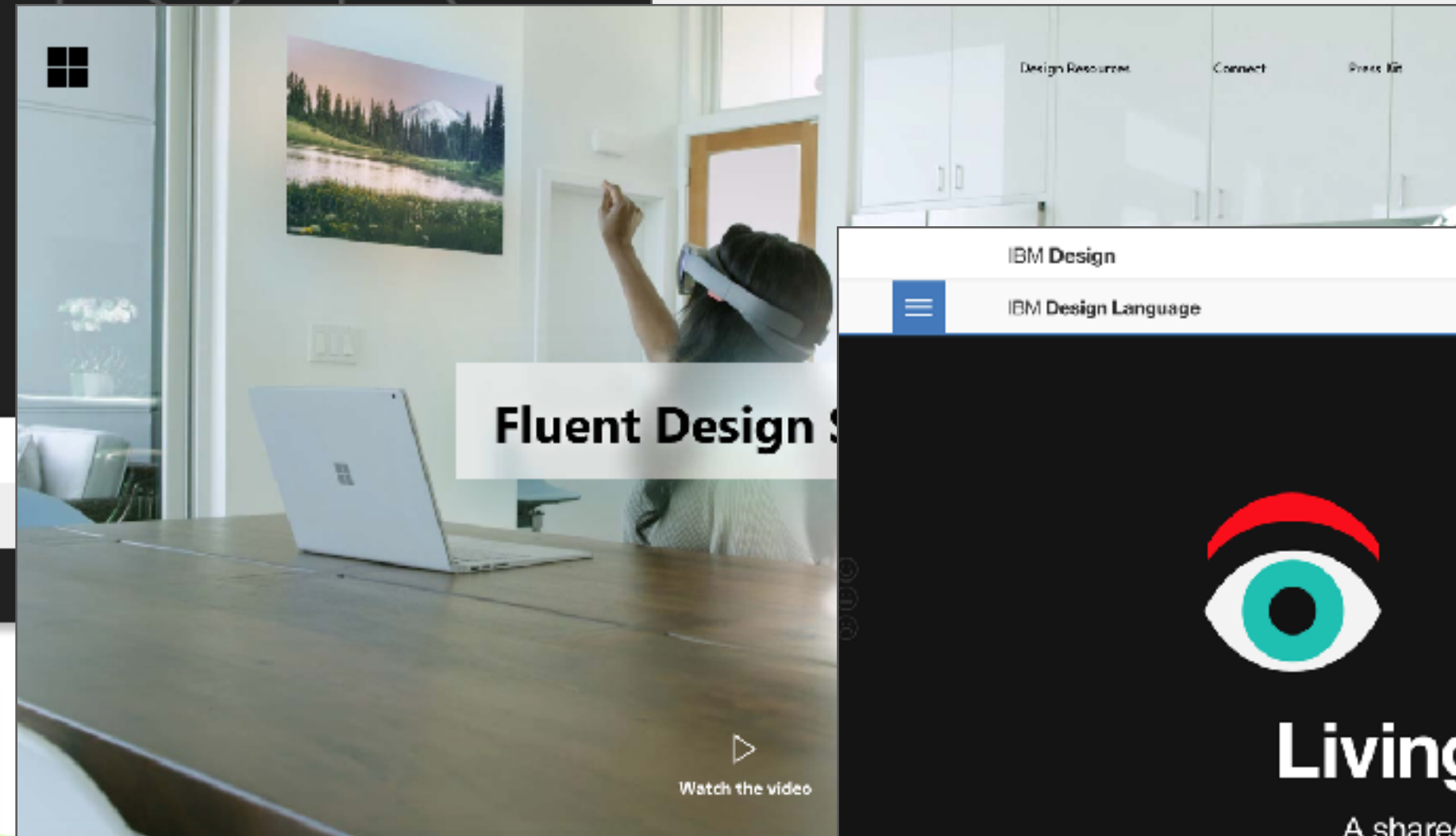
Our new site signals a more centralized approach to Material Design. We're thrilled to see what you make with it.

[READ THE ARTICLE](#)

DESIGN GUIDELINES

The Material Design guidelines are a living document of visual, interactive, and motion guidance.

LAST UPDATED SEPT 2017



Fluent Design

[Watch the video](#)

An eloquent design system for

Now's the time for bold, scalable, universal design. This is a time for sensory experiences. The world is at our fingertips – join Microsoft and develop apps using Fluent Design.

[GET STARTED](#)

IBM Design Studios Work Practices Careers Blog

IBM Design Language

Living Language

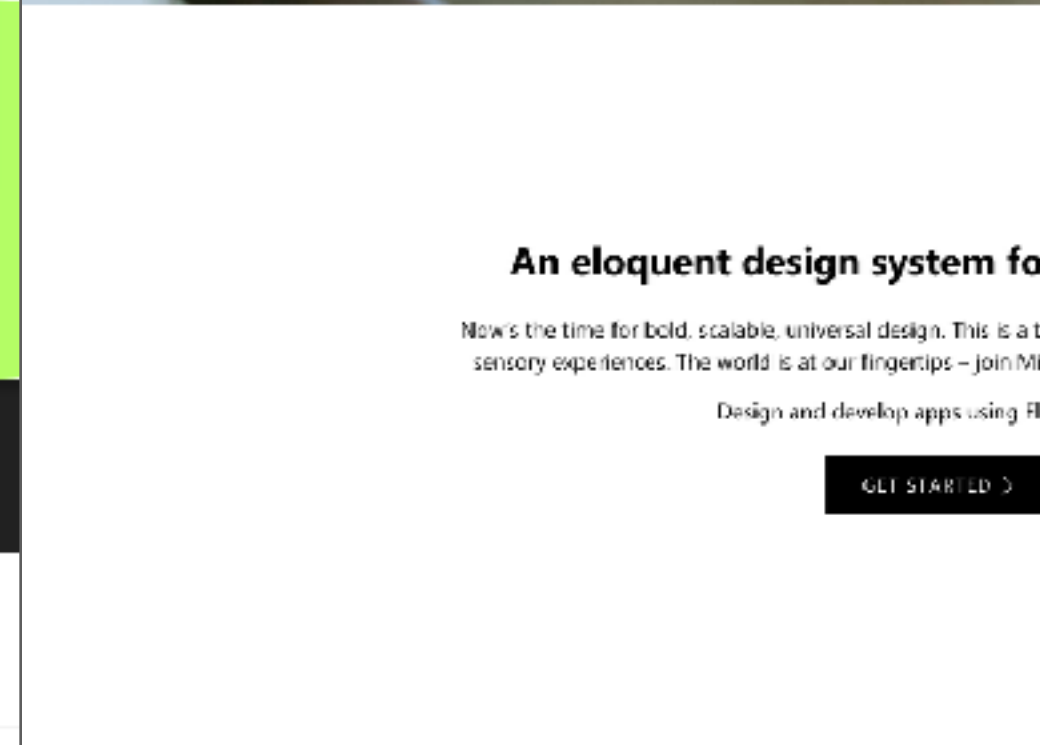
A shared vocabulary for design

[↓](#)

Start the conversation

Explore the guidelines to design for empowering experiences

[Get going](#)



Sensor Experi

Look sharp, feel vibrant





[Home](#)

Guidance

Government design principles

From: [Government Digital Service](#)
Published: 3 April 2012

The UK government's design principles and examples of how they've been used.

Contents

- [1. Start with user needs](#)
- [2. Do less](#)
- [3. Design with data](#)
- [4. Do the hard work to make it simple](#)
- [5. Iterate. Then iterate again](#)
- [6. This is for everyone](#)
- [7. Understand context](#)
- [8. Build digital services, not websites](#)
- [9. Be consistent, not uniform](#)
- [10. Make things open: it makes things better](#)

1. Start with user needs

Service design starts with identifying user needs. If you don't know what the user needs are, you won't build the right thing. Do research, analyse data, talk to users. Don't make assumptions. Have empathy for users, and remember that what they ask for isn't always what they need.

- [What we mean when we say 'service transformation'](#), by Mike Bracken
- [Most of government is mostly service design most of the time](#), by Matt Edgar
- [Vertical campfires: our user research walls](#), by Kate Towsey

2. Do less

Government should only do what only government can do. If we've found a way of doing something that works, we should make it reusable and shareable instead of reinventing the wheel every time. This means building platforms and registers others can build upon, providing resources (like APIs) that others can use, and linking to the work of others. We should concentrate on the irreducible core.

- [Building digital civic infrastructure from the ground up](#), by Mike Bracken
- [What we've learned about scaling agile](#), by Jamie Arnold



Supported browsers

We design GitHub to support the latest web browsers. We support the current versions of [Chrome](#), [Firefox](#), [Safari](#), [Microsoft Edge](#) and [Internet Explorer 11](#).

Firefox Extended Support Release

We do our best to support Firefox's latest [Extended Support Release](#) (ESR). Older versions of Firefox may disable some features on GitHub and require the latest version of Firefox.

Internet Explorer on Windows XP

Because Windows XP is not supported, you see an error message when you try to load GitHub in Internet Explorer. For more information, see "[Improving GitHub's SSL setup](#)."

Internet Explorer Outdated Browser Error

We only support IE running in "Standards Mode." If you see an error message about your outdated browser, turn off "Compatibility View."

Beta and developer builds

You may encounter unexpected bugs in beta and developer builds of our supported browsers. If you encounter a bug on GitHub in one of these unreleased builds, please verify that it also exists in the stable version of the same browser. If the bug only exists in the unstable version, consider reporting the bug to the browser developer.

Article versions

[GitHub.com](#)

[GitHub Enterprise 2.11](#)

[GitHub Enterprise 2.10](#)

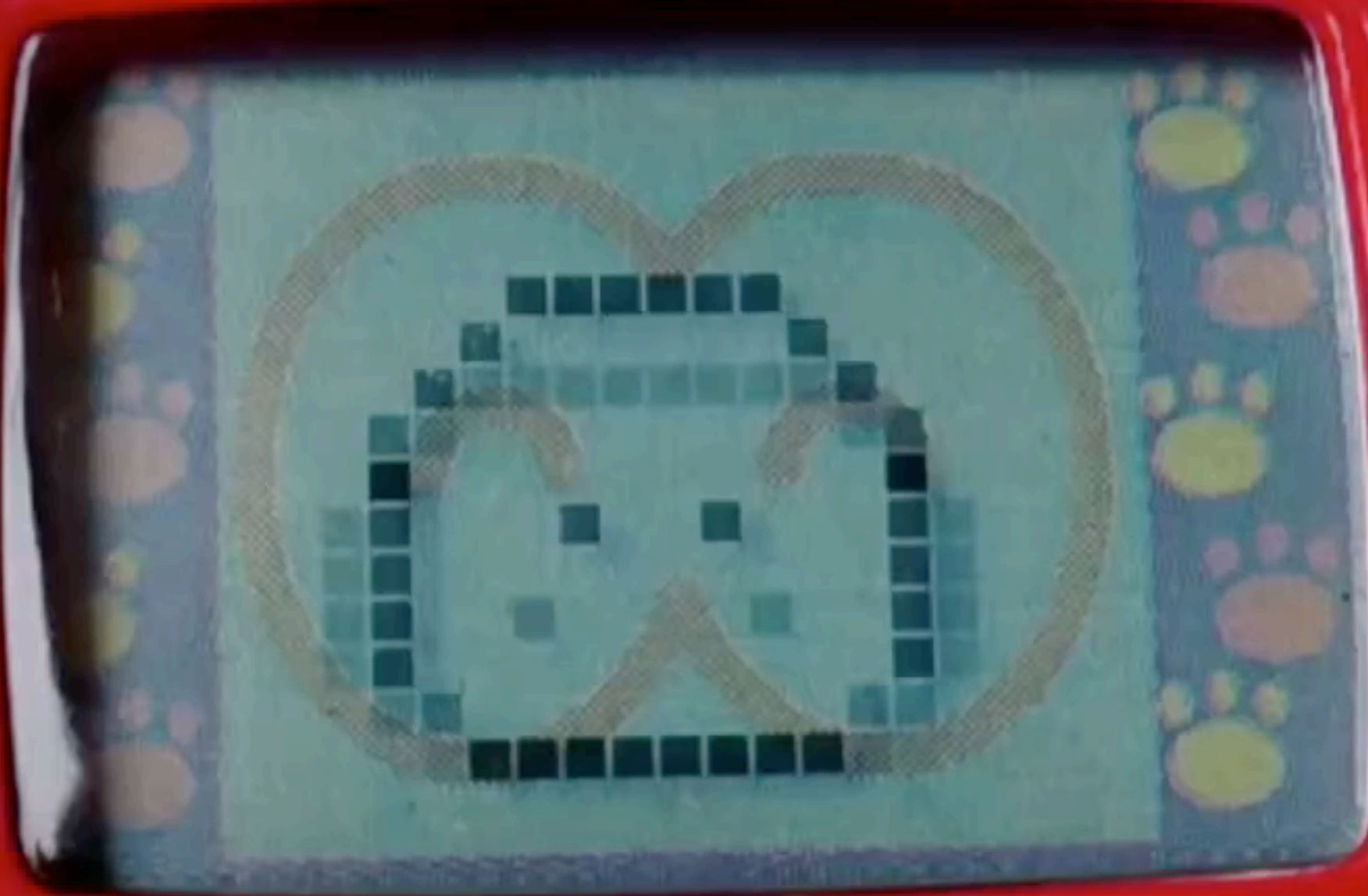
[GitHub Enterprise 2.9](#)

[GitHub Enterprise 2.8](#)

A quick note

“design system”

Rakuraku Dinokun



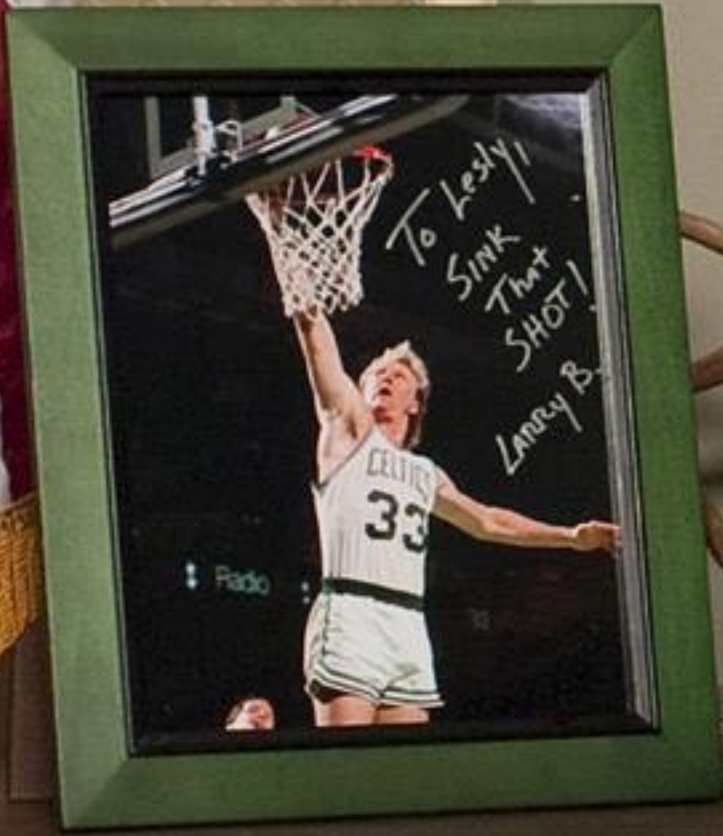
ESC

Enter

Close



1. Put someone in charge



AMERICA'S WOMEN
KEEPING FAITH
WITH JIMMY CARTER
LAWRENCE JOHNSON
GAIL COLLINS
JANE



“System enthusiasts must become entrepreneurs, pitching and selling ideas that get a possibly resistant organization to commit.”

—Nathan Curtis

2. Have a roadmap

Vanilla framework high level roadmap ☆ Private Team 🌐 Public

👁 Subscribed ⋮ Show Menu

Long-term goals ⋮

- Do a talk on Vanilla at an event
- Create Dashboard theme v1
- Improve content of docs.vanillaframework.io
- Improve docs.vanillaframework.io look and feel
- Develop Vanilla Web Components
- External accessibility audit
- Build a service API

Add a card...

Current quarter (Q2 2017) ⋮

- Publish Vanilla docs on docs.ubuntu.com
- Publish Docs theme docs
- Publish Brochure theme docs
- Create Docs theme v1
- Create Store theme v1

Add a card...

In progress ⋮

- Design front end of Vanilla website
- Release Vanilla website

Add a card...

Done ⋮

- Design a logo
- Create Brochure theme v1
- Improve release process ✅ 5/5
- Provide a starter theme to extend VF
- Investigate solution for visual regression testing over two repos
- Create social channels ☰ 1 📧 3/4
- Update Sketch library
- Create Brochure theme
- Create Docs theme
- Setup Sketch library
- Define accessibility standards
- Conduct an accessibility audit
- Clean up code to agreed code standards ☰
- Audit of code to ensure no invisible styles
- Finish interface inventory, inc. themes
- Set up Metalsmith processes
- Document workflow and process 👤 1
- Define browser support

Add a card...

Add a list...

3. Be open

- > AJ
- Yaili
- Rectangle 6
- Rectangle 6
- Rectangle 6
- Rectangle 6
- T Available instances In...
- T Description Public net...
- △ Polygon 3
- ▽ Polygon 3
- ▽ Polygon 3
- > Group
- Rectangle 4
- T Public
- T Joyent-SDC-Public
- Rectangle 4
- T Private
- T Joyent-SDC-Private
- Rectangle 4
- T Fabric
- T Private
- T default
- T See all available netw...
- T Choose a network

Sergio

Networks:

Joyent-SDC-Public ● NAT ● Fabric ● Public
 Public networks with IP addresses routable over the internet

Joyent-SDC-Private ● Fabric ● Public
 Networks with non-routable IP addresses, shared by all customers

 Gateway:
 Subnet:
 Resolvers:

default ● NAT ● Fabric ● Public

custom ● Fabric ● Public

 Gateway:
 Subnet:
 Resolvers:

BACKGROUND

#E5E5E5 100%

Show in exports

PIXEL PREVIEW

Pixel Preview

Pixel Grid

EXPORT

Click + to add an export setting



Show
and tells



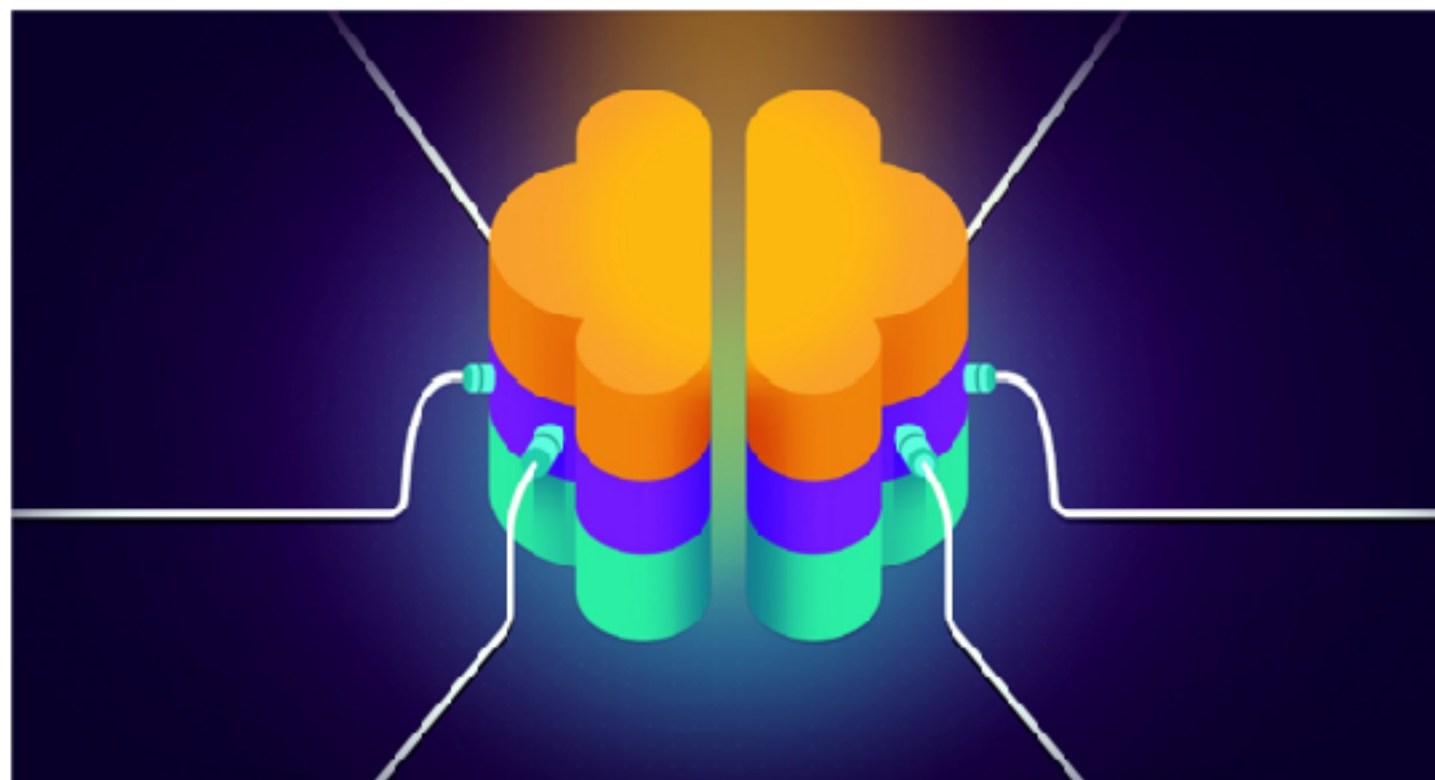


Make Us Proud

Thoughts and ideas on technology and design from the Make Us Proud team



Follow



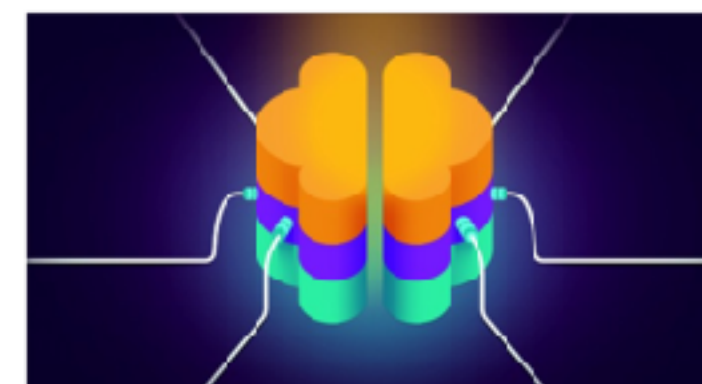
Joyent CoPilot: Bringing Application Awareness to Cloud Infrastructure. Part III.

Story behind building an experimental
application management platform

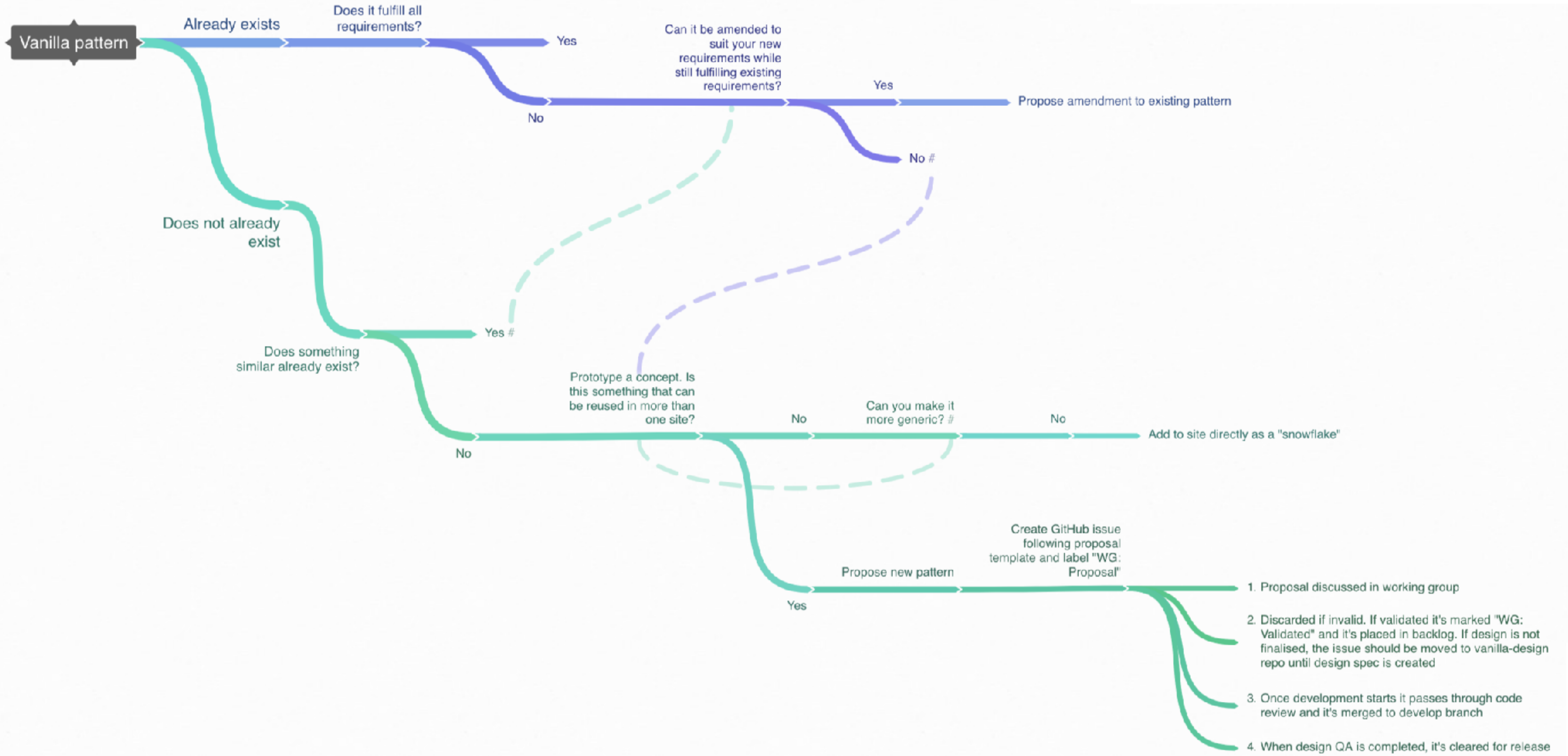


Antonas Deduchovas

Oct 10



4. Define contribution process



Vanilla pattern

Already exists

Does it fulfill all requirements?

Yes

Can it be amended to suit your new requirements while still fulfilling existing requirements?

Yes

Propose amendment to existing pattern

No

No #

Does not already exist

Does something similar already exist?

Yes #

Prototype a concept. Is this something that can be reused in more than one site?

No

Can you make it more generic? #

No

Add to site directly as a "snowflake"

No

Yes

Propose new pattern

Create GitHub issue following proposal template and label "WG: Proposal"

1. Proposal discussed in working group
2. Discarded if invalid. If validated it's marked "WG: Validated" and it's placed in backlog. If design is not finalised, the issue should be moved to vanilla-design repo until design spec is created
3. Once development starts it passes through code review and it's merged to develop branch
4. When design QA is completed, it's cleared for release



This repository Search

Pull requests Issues Marketplace Explore



yldio / joyent-portal

Unwatch 13

Unstar 19

Fork 2

Code

Issues 54

Pull requests 4

Boards

Reports

Projects 1

Wiki

Insights

Settings

UI Toolkit: making and proposing changes

Edit

New Page

Inayaili de León Persson edited this page 5 days ago · 1 revision

1. The source of truth for the UI toolkit components is the Joyent Figma team library.
Remember: Some components might have been updated in the Figma library but not yet moved into the codebase.
2. When creating new components, strive to be as modular as possible.
3. New components should have a design spec before they are added to the codebase
Remember: Do not add a new component directly to the codebase.
4. Ideally, more than one team member agrees to that the component should be added to the toolkit.
Remember: Not everything is a component. If a particular element is not yet proven to be useful in multiple places, it should live in the app where it was created.
5. When a new component is added to the codebase, it should be added with the appropriate level of documentation. At a minimum, documentation should include all React props with their type and description. Ideally, also include usage guidelines and good practices.
6. Variations of the same component should be displayed in different demos in the documentation.
7. New and existing components should be reviewed by the team, ideally on a demo link for easier testing (for example, deployed to <https://zeit.co/now>). Testing should include code, visual and interaction review.
8. Requests for new components should be submitted in a GitHub issue. Anyone can submit requests for new components.
Remember: Both design and engineering work should be outlined in GitHub issues.
9. Changes to, and bugs on, existing components should always be requested via, and

Pages 3

Clone this wiki locally

<https://github.com/yldio/jc>

Clone in Desktop

5. Look after the docs



GO TO SECTION

[Writing Goals and Principles](#)

[Voice and Tone](#)

[Writing About People](#)

[Grammar and Mechanics](#)

[Content Types](#)

[Web Elements](#)

[Writing Blog Posts](#)

[Writing Technical Content](#)

[Writing Legal Content](#)

[Writing Email Newsletters](#)

[Writing for Social Media](#)

[Writing for Accessibility](#)

[Writing for Translation](#)

GO TO SECTION

[Writing Goals and Principles](#)

[Voice and Tone](#)

[Writing About People](#)

[Grammar and Mechanics](#)

[Content Types](#)

Welcome to the MailChimp Content Style Guide

This style guide was created for MailChimp employees, but we hope it's helpful for other content and communications teams too.

If you work at MailChimp

This is our company style guide. It helps us write clear and consistent content across teams and channels. Please use it as a reference when you're writing for MailChimp.

This guide goes beyond basic grammar and style points. It's not traditional in format or content. We break a number of grammar rules for clarity, practicality, or preference.

We've divided the guide by topic based on the types of content we publish, so you can reference it as needed or browse in order. The entire guide is searchable, so you can go straight to the item you're looking for.

If you work at another organization

We invite you to use and adapt this style guide as you see fit. It's completely public and available under a Creative Commons Attribution-NonCommercial 4.0 International license. All we ask is that you credit MailChimp.

We welcome any feedback for improving the guide.

[MailChimp Content Style Guide on GitHub](#)

BETA This is new guidance. Complete our quick 5-question survey to [help us improve it](#).

[Service manual](#) > [Design](#) > Form structure

Design

[Give feedback about this page](#)

Form structure

Published by: [Design community](#)
Last updated: 10 months ago

Page contents:

- **Meeting the Digital Service Standard**
- Don't treat online forms like paper forms
- Know why you're asking every question
- Design for the most common scenarios first
- Start with one thing per page
- Structure your form to help users

Page contents:

- Meeting the Digital Service Standard
- **Don't treat online forms like paper forms**
- Know why you're asking every question
- Design for the most common scenarios first
- Start with one thing per page
- Structure your form to help users
- Discuss online forms

This guide explains how to structure online forms.

Meeting the Digital Service Standard

To pass [point 13 \(make the user experience consistent with GOV.UK\)](#) in your [service assessments](#), you must use GOV.UK design patterns and guidance.

Read the guide on [using, adapting and creating patterns](#) before you start designing or building anything.

Don't treat online forms like paper forms

Don't assume you should create a digital version of an existing paper form. Paper forms are a product of a pre-digital era and are subject to different constraints than digital services.

Consider the service as a whole.

Know why you're asking every question



A/B testing in 5 steps

- 1 Write a hypothesis
- 2 Get data about it
- 3 Create variants
- 4 Run the experiment
- 5 Analyze the results

© 2014

6. Make things easy to find

Getting Started

The Salesforce Lightning Design System includes the resources to create user interfaces consistent with the Salesforce Lightning principles, design language, and best practices. Rather than focusing on pixels, developers can focus on application logic, while designers can focus on user experience, interactions, and flows.

This site provides a range of resources for designers and developers, which includes:

- semantic and accessible component markup.
- cross-browser compatible CSS.
- icons, font, and design guidelines.

Developers

Familiarize yourself with the following:

1. Get an overview of our [Markup and Style](#) guidelines including the class naming conventions used in our CSS.
2. Review the Components; each component provides semantically correct and accessible markup and documentation.
3. Dive into the platform-specific getting started steps below.

Designers

Start with the following:

1. Explore the [Guidelines](#) to learn the Salesforce product design patterns and principles.
2. Review the Components section, to familiarize yourself with the existing components which you can incorporate into your designs.

KSS Introduction Documentation System Generate Styleguides [View the code on GitHub](#)

KSS Kyle Style Sheets

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

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

```
// A button suitable for giving a star to someone.
//
// :hover          - Subtle hover highlight.
// .star-given     - A highlight indicating you've already given a star.
// .star-given: hover - Subtle hover highlight on top of star-given styling.
// .disabled       - Dims the button to indicate it cannot be used.
//
// Styleguide 2.1.3.
```

Documentation for humans

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

Create atomic design systems with Pattern Lab

Documentation

Download

Pattern Lab helps you and user interfaces using atomic design

ATOMS • MOLECULES • ORGANISMS • TEMPLATES • PAGES • ALL

adactio / Pattern-Primer

10 commits 1 branch 0 releases 2 contributors

Generating styled markup from a folder of markup snippets.

STYLEGUIDEGUIDE

Style Guide Guide

A boilerplate for creating superb style guides

Getting started

Guidelines

Styles

Components

Utilities

Page templates

Downloads

Support

Contribute

Design System

App Gallery

Contributing

Fractal

Build. Document. Integrate.

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

Get started →

Clearleft.com Component Library

DOCUMENTATION

Overview

Styleguide

COMPONENTS

Grids

Patterns

Story Grid

DIGITAL STORES

A storytelling platform for exploring The Wellcome Trust's eclectic collection of medical and historical

CHANNEL 4 SCRAPBOOK

A new digital lifestyle experience developed with the product team at Channel

7. Plan maintenance

“A 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



This repository Search

Pull requests Issues Marketplace Explore



vanilla-framework / vanilla-framework

Watch 23

Unstar 145

Fork 27

Code

Issues 28

Pull requests 0

Boards

Reports

Projects 0

Wiki

Insights

Filters

label:"WG: Proposal" is:closed

Labels

Milestones

New issue

Clear current search query, filters, and sorts

0 Open	15 Closed	Author	Labels	Projects	Milestones	Assignee	Sort
	Navigation pattern needs to show current highlighted page and rollover state WG: Proposal						4
	#1269 by barrymcgee was closed on 15 Aug						
	Include CSS variables into our SCSS variables Type: Enhancement WG: Proposal						1
	#1242 by richmccartney was closed on 7 Aug						
	[Proposal] Improve re-usability of mixins and patterns. WG: Proposal						
	#1147 by richmccartney was closed on 10 Jul						
	create a compact pull-quote variation WG: Proposal						3
	#1108 by pmahnke was closed on 27 Jun						
	Build is-crossed state for list items Priority: Medium Type: Enhancement WG: Proposal						5
	#1101 by anthonydillon was closed on 26 Jun						
	Review vertical space between Headings and containers WG: Proposal						3
	#1077 by joanasa89 was closed on 26 Jun						
	[Proposal] Create a "Design" and "Development" label in GitHub WG: Proposal						7
	#1019 by yailli was closed on 20 Apr						
	Add p-strip--light Type: Question WG: Proposal						13
	#966 by anthonydillon was closed on 24 Mar						
	Do we need index.md in docs root? Type: Question WG: Proposal						
	#862 by yailli was closed on 27 Feb						
	Review equal height util across breakpoints WG: Proposal						1
	#771 by barrymcgee was closed on 27 Mar						
	[Proposal] Utility naming conventions and breakpoints WG: Proposal						5
	#745 by richmccartney was closed on 20 Mar						
	[Proposal] Minor grid improvements WG: Proposal						8
	#733 by yailli was closed on 27 Jan						
	[Proposal] Update notifications design based on new designs Status: Blocked Type: Enhancement						1
	WG: Proposal #641 by yailli was closed on 17 Sep 2016						
	[Proposal] Hide blockquote page from docs WG: Proposal						



Standards for table patterns #1355

New issue

Open nottrobin opened this issue 23 days ago · 1 comment

nottrobin commented 23 days ago Owner

Bartek created a new pattern for snapcraft-flask, `p-table-key-value`, which follows precedent from existing table patterns in Vanilla pretty well. However, I'm not too happy about this precedent:

- there is one actual pattern, `p-table-expanding`, and two modifiers for "p-table" - `p-table--sortable` and `p-table--mobile-card`
- all three patterns actually target `td` and `tr` elements directly

I thought it was policy not to target elements directly. And so these should ideally be targeting e.g. `.p-table-expanding-row` and `.p-table-expanding-cell` instead of `tr` and `td`. I think these patterns may be out-of-date because they were imported from themes with less strict standards, but I just wanted to confirm if my understanding of the ideal implementation is correct or not.

I don't think modifiers based on a `p-table` pattern should exist if the `p-table` pattern does not itself exist. And so e.g. `p-table--sortable` should really be `p-table-sortable`.

This brings up the issue that `p-table-expanding` in fact looks incredibly similar to `p-table--sortable`, even though the former is a pattern rather than a modifier. One could be forgiven for confusing patterns named like this with modifiers.

So I'd also like to suggest that when we create *patterns* with adjectives in the name, we consider using the more natural form of `p-{adjective}-{noun}` rather than `p-{noun}-{adjective}`, to clearly distinguish them from modifiers. Then `p-expanding-table` would be clearly of a different type than `p-table--sortable`. Thoughts?

Standards for table patterns has no dependencies

nottrobin added **Don't merge** **Duplicate** **WG: Proposal** and removed **Don't merge** **Duplicate** labels 23 days ago

anthonydillon commented 12 days ago Owner

Approved in the working group.

Areas of work:

- Audit the table pattern and come up with a naming convention
- Rename deprecate old pattern names
- Consider combining patterns if required

Pipeline
New issues

Assignees
No one assigned

Labels
WG: Validated

Projects
None yet

Milestone
No milestone

Estimate
No estimate yet

Releases
Not inside a Release

Epics
Not inside an Epic

Notifications
Subscribe

You're not receiving notifications from this thread.

2 participants
[Profile] [Profile]

Move Issue

8. Learn from others

Design Systems

Enter your email below to join Design Systems on Slack!

Enter your email

No spam. Ever.

Design Systems Weekly

A curated selection of links covering all things design systems, styleguides, pattern libraries and front-end architecture, and preventing entropy in the wild.

Website Style Guide Resources

Articles Books Podcasts Talks Tools Examples

Your email

No spam. Ever.

Issues

Style Guide Podcast

A small batch series of interviews on Style Guides, hosted by [Anna Debenham](#) and [Brad Frost](#).

Subscribe in iTunes or Get the feed

Season 2

Website Style Guide Resources

Articles Books Podcasts Talks Tools Examples

+ SPEC

Search...

About Sponsors

Home | Podcasts | Design Details

Design Details

A show about the people who design our favorite products.

Subscribe Follow Sponsor

221: Rate The Fire (feat. Chikezie Ejiasi)

November 1, 2017

Today we caught up with Chikezie Ejiasi, a designer currently working on Daydream at Google. In this episode we dig into building for VR, moving from software to hardware to virtual reality, design tooling, career navigation, and so much more.

Listen to this Episode Download

222: The Quintan

October 11, 2017

Today we caught up with Quintan, a designer currently working on Daydream at Google. In this episode we dig into building for VR, moving from software to hardware to virtual reality, design tooling, career navigation, and so much more.

Listen to this Episode Download

The five best design links, every day.

11/07

iPhone X Web Navigation Concept

Is there a better way to navigate on a website than the traditional hamburger menu button in the top left or right corner?

Concept UI Mobile medium.muzli

Designing a private Amazon shopping experience

Amazon Iacognito concept. Because sometimes you just want privacy while shopping on Amazon.

UI UX Concept undesign.cc

Making Spotify recommendations better

Getting a deeper understanding of why people like or dislike certain songs.

UX UX Content medium.com

1. Put someone in charge
2. Have a roadmap
3. Be open
4. Define contribution process
5. Look after documentation
6. Make things easy to find
7. Plan maintenance
8. Learn from others

@yaili

github.com/yaili/speaking

—

designers: bit.ly/mup-jobs

developers: bit.ly/yld-jobs

Thank you!