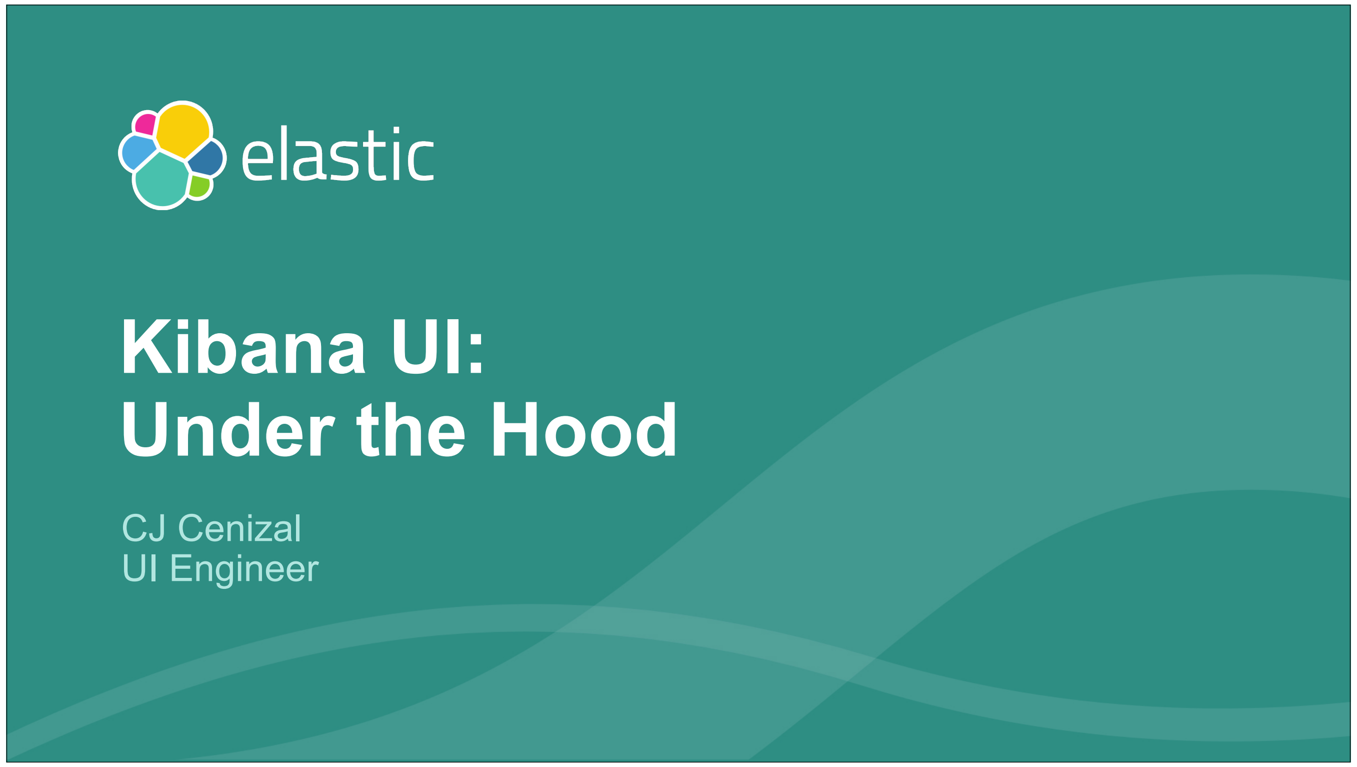
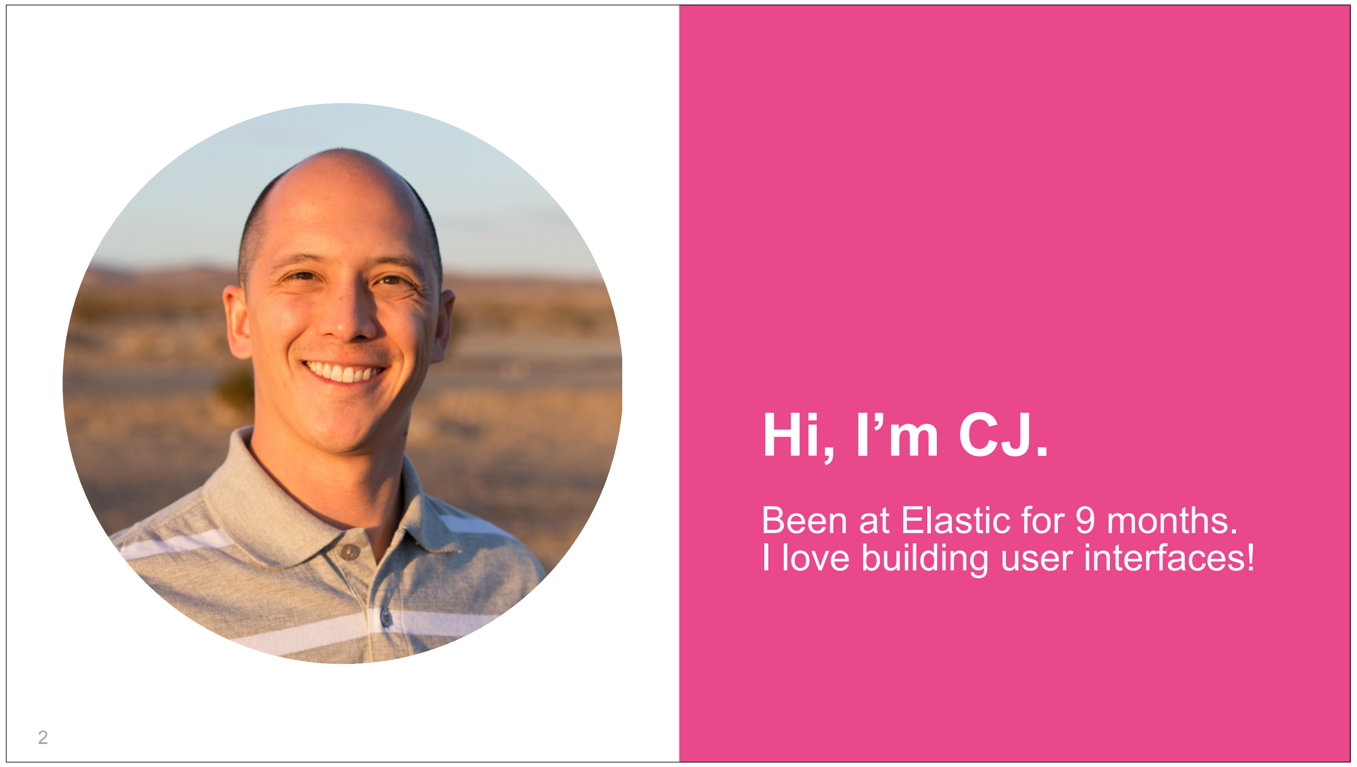


elastic

Kibana UI: Under the Hood

CJ Cenizal
UI Engineer

Hey everyone, I’m CJ. This talk is about the UI development process on the Kibana team.



Hi, I’m CJ.

Been at Elastic for 9 months.
I love building user interfaces!

Let me start out by sharing a little bit about myself. Been with Elastic for 9 months. I’m a UI Engineer on the Kibana team. I come from a design background. I studied Design in college.

After graduating I started building websites in Flash and when Steve Jobs killed Flash I moved on to JS web app development. I quickly realized that I love building user interfaces and making usable software. Now I’m on the Kibana team and my focus is on making our user experience better.

How’s the sausage made?

Let’s talk about the process, and how this affects you.

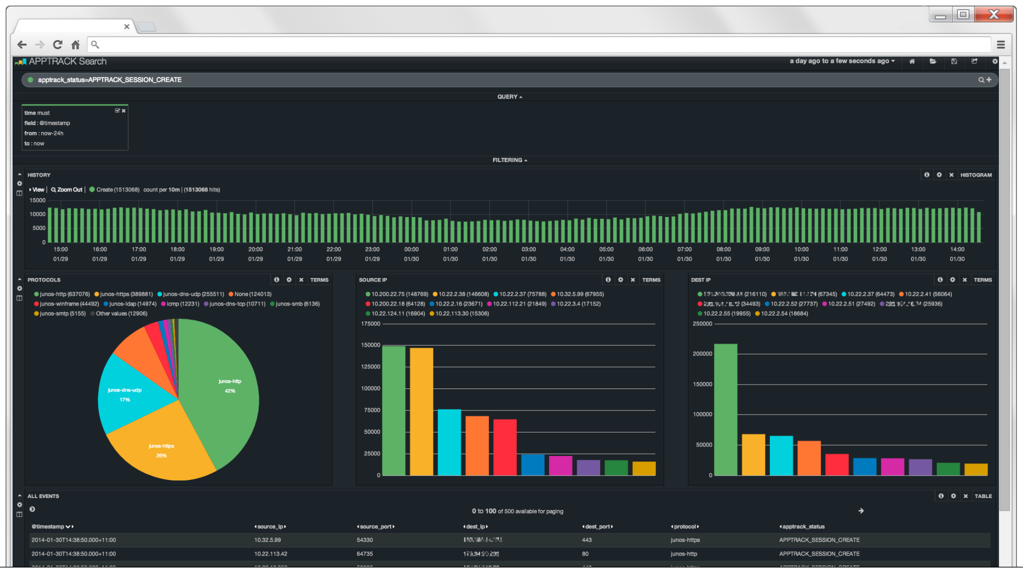
Most of our meetups involve showing off our tech, talking about how you can use it as a consumer. This talk is a little more geared towards people who write Kibana code, either as plugin developers or open source contributors. It’s more of a tour behind the scenes, so even if you don’t fall into one of these categories, you might find it interesting to learn how we work on Kibana and where the product is headed from a UI and UX standpoint.

I have a feeling this presentation will go by quickly so please keep track of any questions you have and ask them at the end, and I can go back and dive into detail on any points you’re curious about.

Kibana 1 & 2

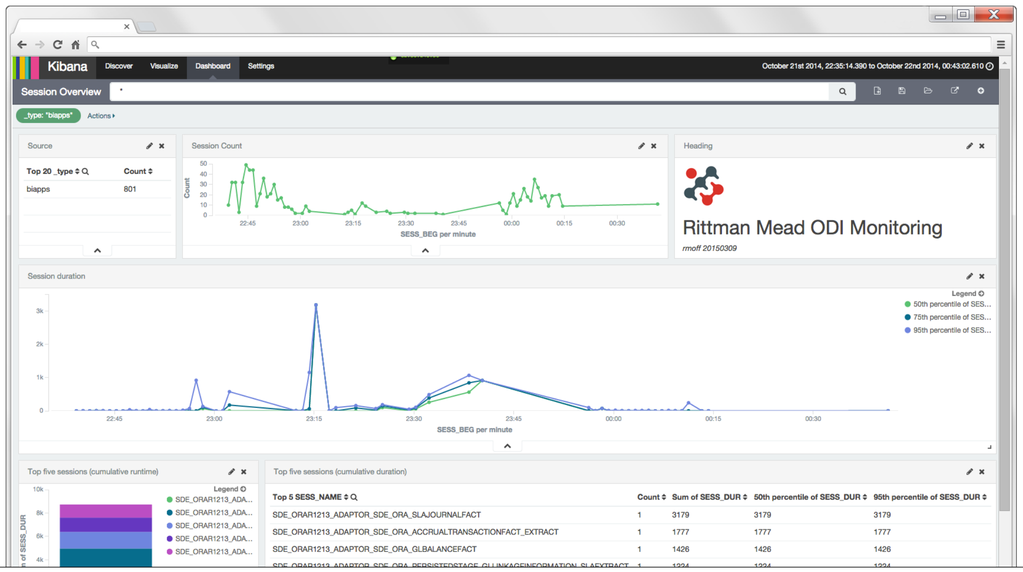
I couldn’t find any screenshots of Kibana 1 or 2.

Kibana 3



Kibana 3 was entirely focused on Dashboards. The simplicity of the product allowed the UI to also be very simple. Very minimal Kibana chrome around the content.

Kibana 4


Kibana
Discover
Visualize
Dashboard
Settings
October 21st 2014, 22:35:14.390 to October 22nd 2014, 00:43:02.610
Session Overview
_type: "biapps"
Actions
Session Count
Heading
Session duration
SESS_BEG per minute
Legend
50th percentile of SES...
75th percentile of SES...
95th percentile of SES...
Top five sessions (cumulative runtime)

Source

Top 20 _type	Count
biapps	801

Rittman Mead ODI Monitoring

rmoff 20150309

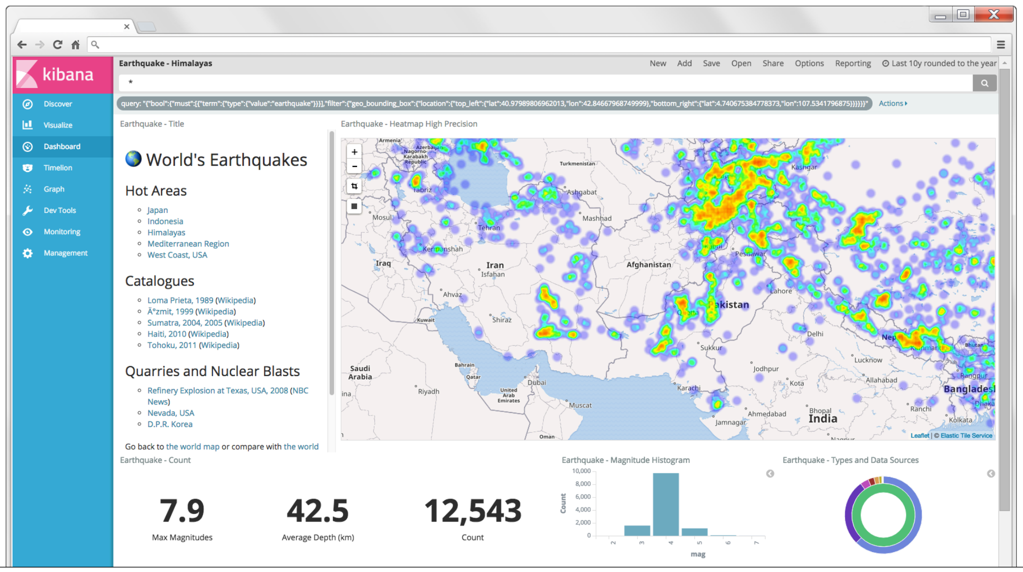
Top five sessions (cumulative duration)

Top 5 SESS_NAME	Count	Sum of SESS_DUR	50th percentile of SESS_DUR	95th percentile of SESS_DUR
SDE_ORAR1213_ADAPTOR_SDE_ORA_SLAJOURNALFACT	1	3179	3179	3179
SDE_ORAR1213_ADAPTOR_SDE_ORA_ACCRUALTRANSACTIONFACT_EXTRACT	1	1777	1777	1777
SDE_ORAR1213_ADAPTOR_SDE_ORA_GLBALANCEFACT	1	1426	1426	1426
SDE_ORAR1213_ADAPTOR_SDE_ORA_PERSISTEDSTAGE_GLLINKAGEINFORMATION_SLAEXTRACT	1	1224	1224	1224

elastic

With Kibana 4, we split this functionality apart into three apps: Discover, Visualize, and Dashboard. This explosive growth in functionality called for a new chrome which could surface a new navigation scheme. We added navigation buttons at the top of the screen to let people navigate to the new apps.

Kibana 5



*

query: "{"bool":{"must":[{"term":{"type":{"value":"earthquake"}}}],"filter":{"geo_bounding_box":{"location":{"top_left":{"lat":40.97989806962013,"lon":42.84667968749999},"bottom_right":{"lat":4.740675384778373,"lon":107.5341796875}}}}}}"

🌍 World's Earthquakes

Hot Areas

- Japan
- Indonesia
- Himalayas
- Mediterranean Region
- West Coast, USA

Catalogues

- Loma Prieta, 1989 (Wikipedia)
- Ä°zmit, 1999 (Wikipedia)
- Sumatra, 2004, 2005 (Wikipedia)
- Haiti, 2010 (Wikipedia)
- Tohoku, 2011 (Wikipedia)

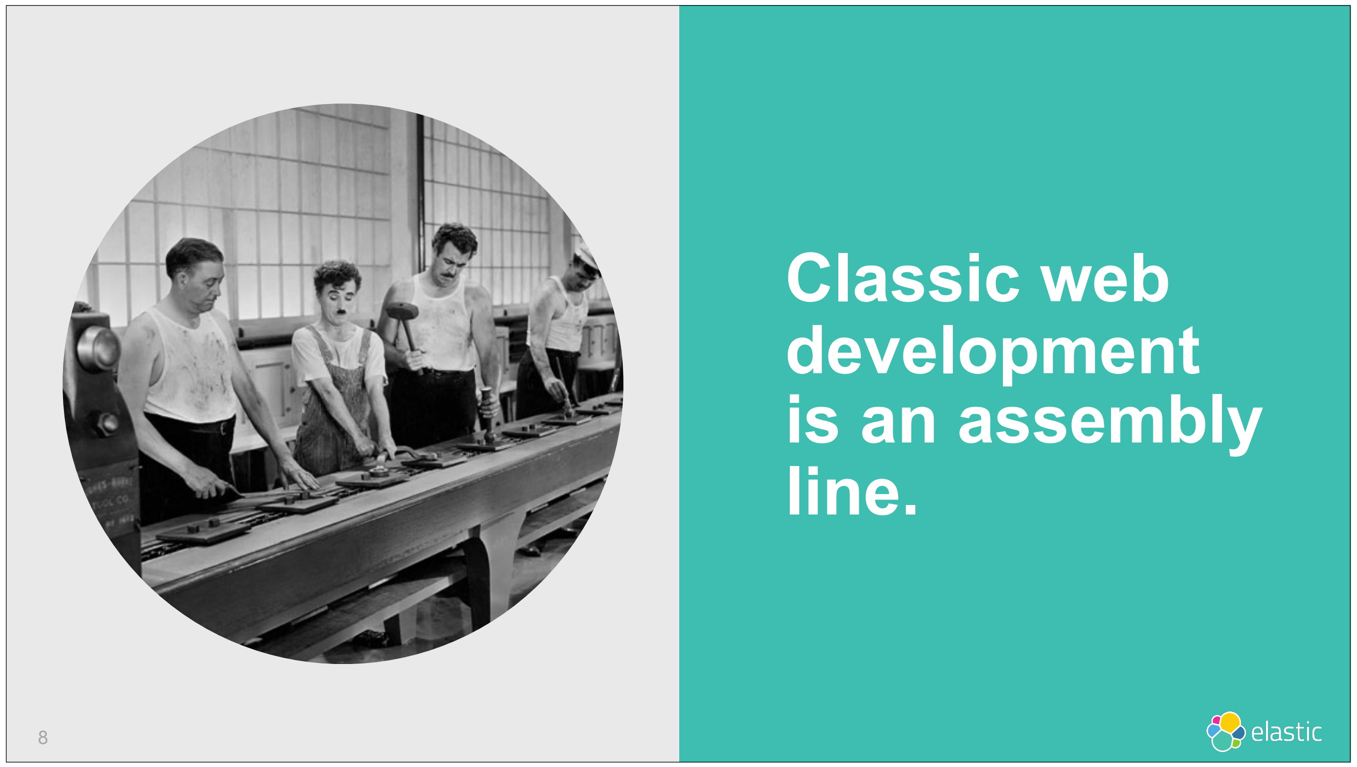
Quarries and Nuclear Blasts

- Refinery Explosion at Texas, USA, 2008 (NBC News)
- Nevada, USA
- D.P.R. Korea

Go back to the world map or compare with the world

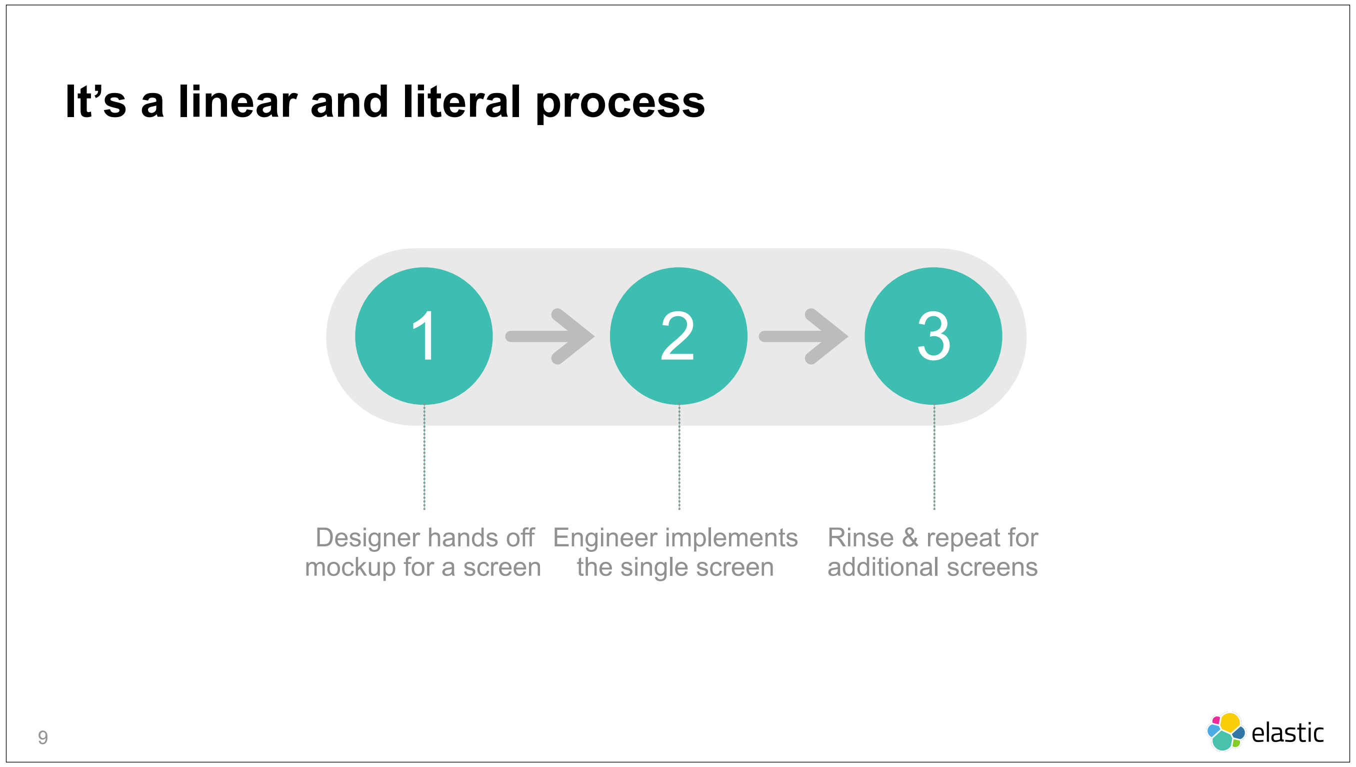
7.9	42.5	12,543
Max Magnitudes	Average Depth (km)	Count

We continued in this vein and added a ton of new apps to Kibana and we also introduced X-Pack. There wasn't enough space to keep adding these navigation buttons at the top of the screen, so we re-thought the chrome and moved the primary navigation into a side bar on the left side of the screen.

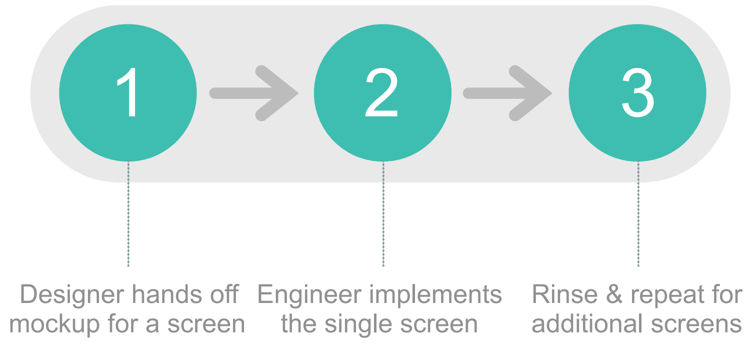


Classic web development is an assembly line.

So that’s how Kibana has changed over time. Now I want to step away from Kibana for a second and talk about a classic problem with web development.

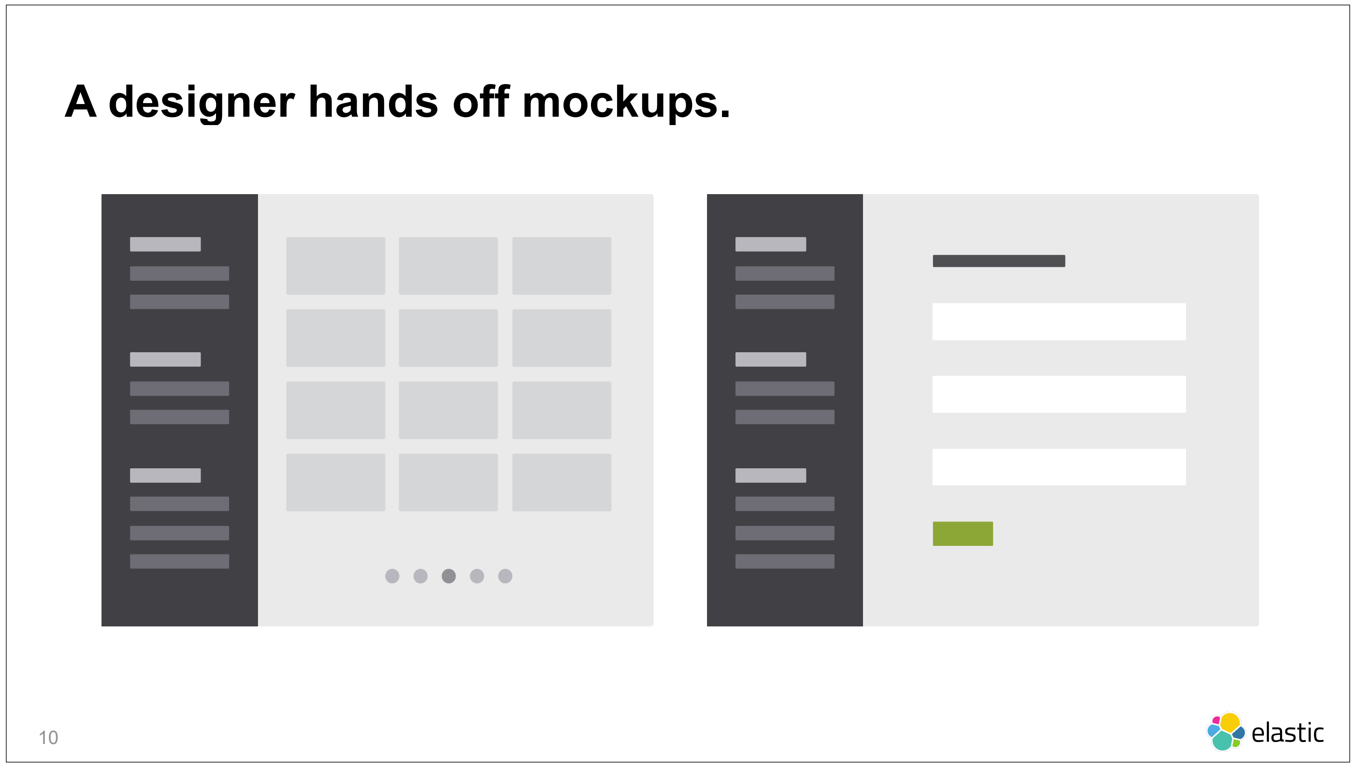


It’s a linear and literal process


1
2
3
Designer hands off mockup for a screen
Engineer implements the single screen
Rinse & repeat for additional screens

elastic

Typical web design and development is a very linear process. A designer designs a series of mockups and hands them off to the developers. The developers implement the mockups, page by page.



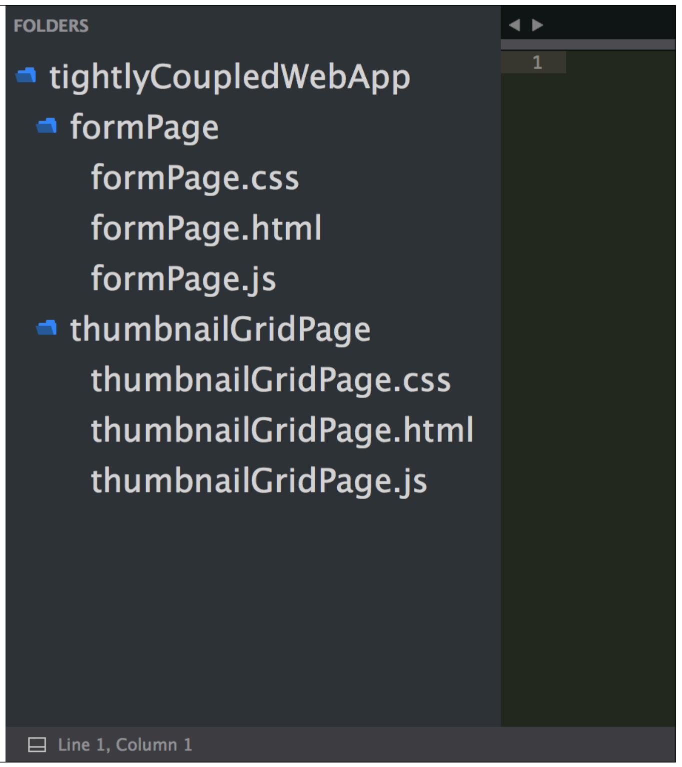
A designer hands off mockups.

elastic

Here’s what a designer might hand off. This is a mockup of Kibana 6. Just kidding. But this is what you get right? Side bar, a grid of thumbnails. Another mockup. Side bar, a form. And what happens next? The developers code it up. They get these mockups and build what they see, page by page.

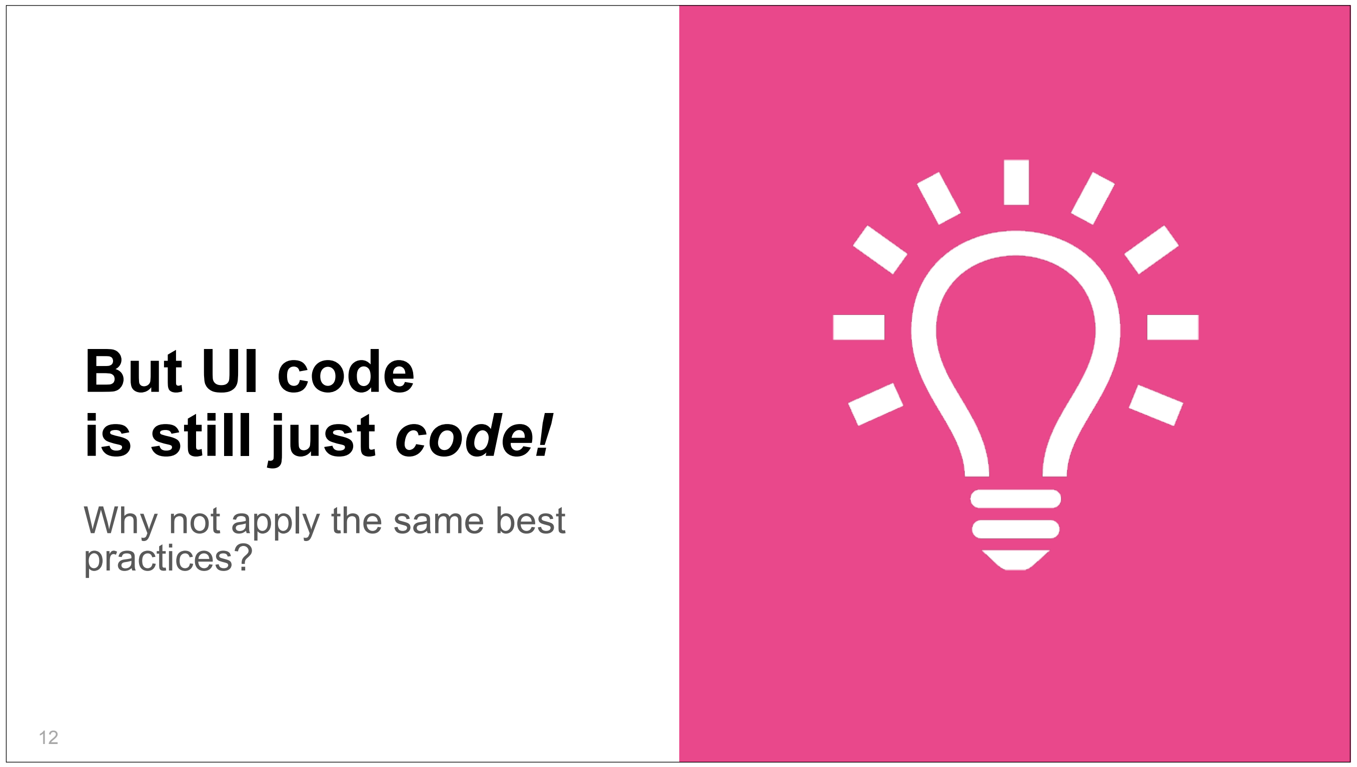
And a developer implements them, literally.

The code becomes tightly-coupled with the mockup.



```
tightlyCoupledWebApp
  formPage
    formPage.css
    formPage.html
    formPage.js
  thumbnailGridPage
    thumbnailGridPage.css
    thumbnailGridPage.html
    thumbnailGridPage.js
```

You end up with code that corresponds clearly to the individual mockups that the designer created. But when you have code that corresponds very concretely to a designer’s mockups, the code becomes very tightly-coupled with those designs. The tight coupling creates a brittle relationship. The code isn’t written to anticipate change; it’s written as if the designs are static. So when the design changes, you end up needing to completely rewrite a lot of existing UI code.

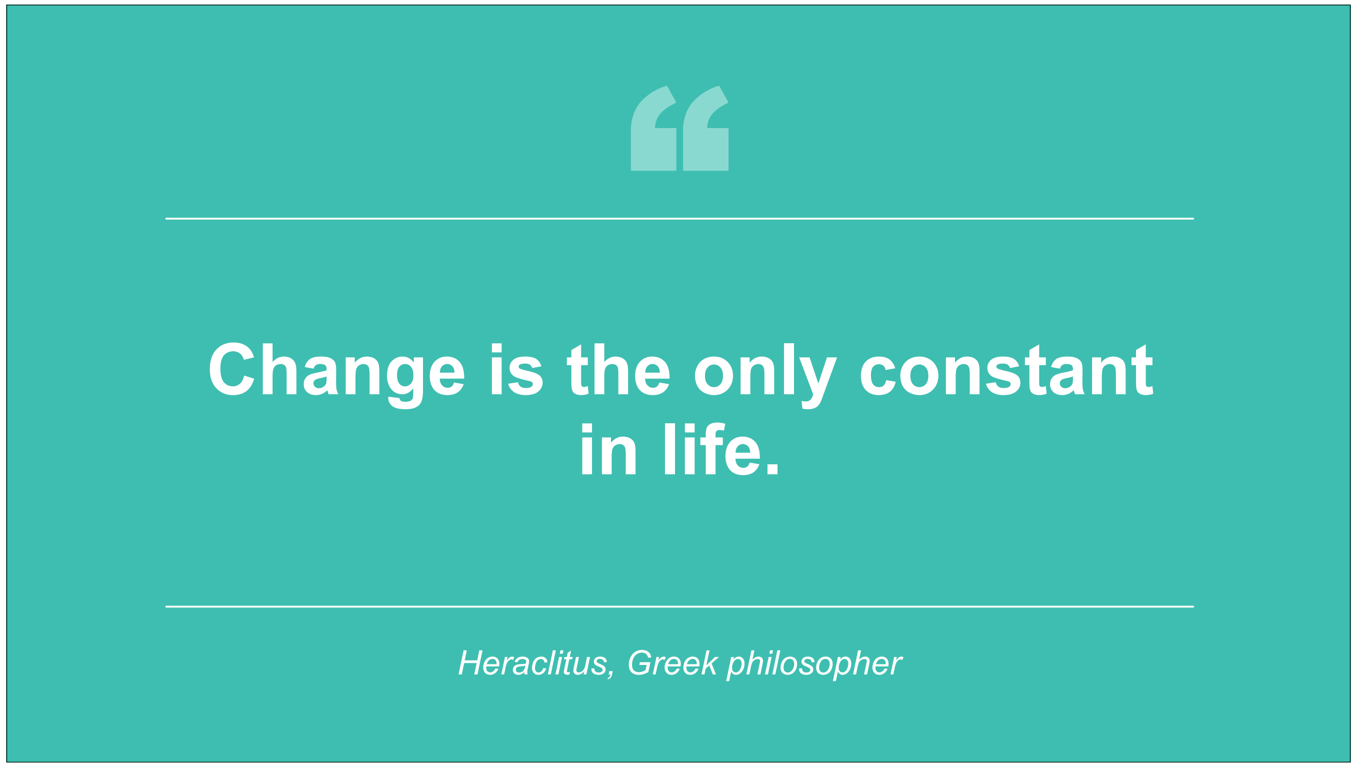


But UI code is still just *code!*

Why not apply the same best practices?

This problem isn’t unique to UI code though. It’s the same thing for all code. When you write code that’s tightly-coupled, you end up with a spaghetti codebase. It’s difficult to reason about, difficult to follow, difficult to change. Unclear logic flow, brittle relationships.

So how do we avoid this? With modular code. By creating useful abstractions. By designing ergonomic interfaces. We can write UI code the same way.



“

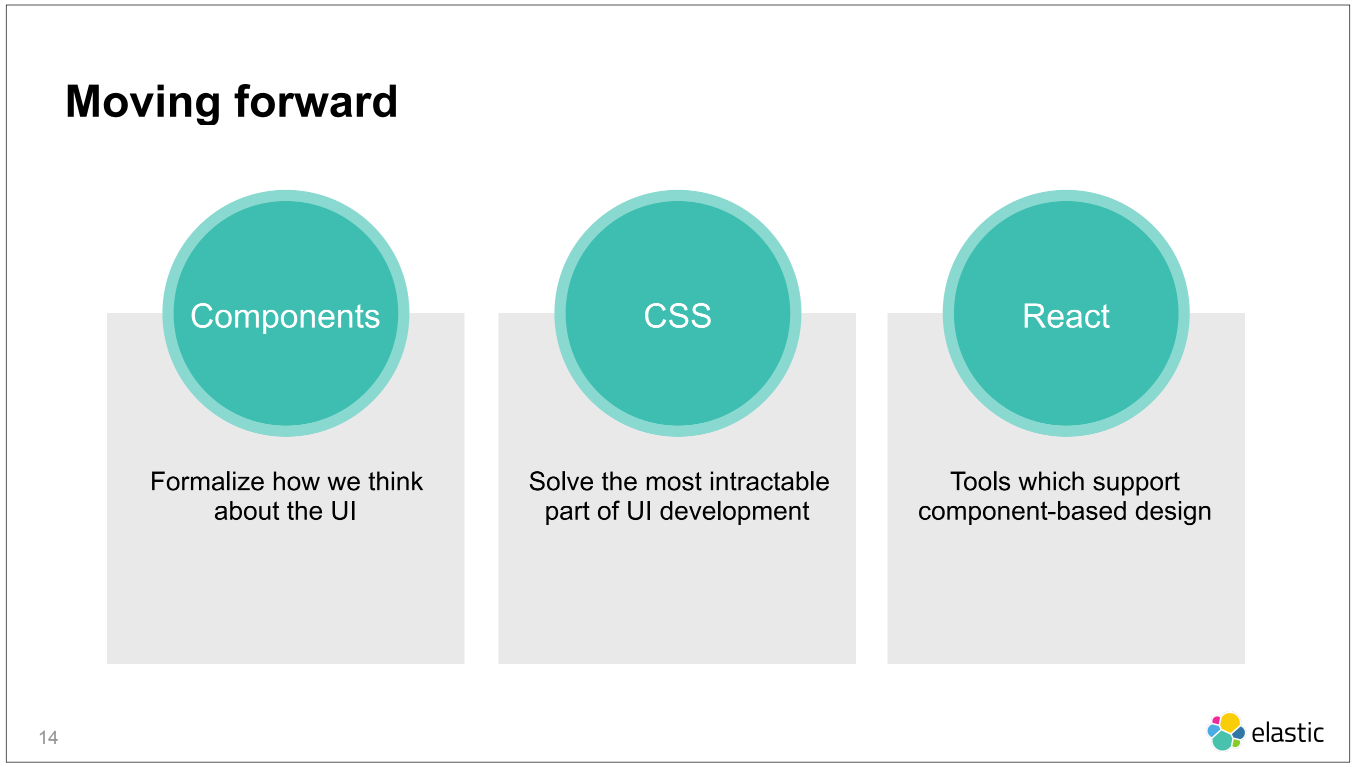
Change is the only constant in life.

Heraclitus, Greek philosopher

Kibana 5 codebase has some technical debt. A lot of the code was originally written in a way that was tightly-coupled to the UI design. And so it’s been difficult to change.

Kibana is a massive app, and you can see how much it’s grown over the past 3 versions. Looking ahead, we know we can expect it to continue to grow. We’ll need to add more apps, the chrome may change significantly, and plugins need significant support (mix and match UI without being tightly coupled to business logic, e.g. time picker). People also want customizability / skinnability.

So it’s important we build it in a way that scales. We need to reduce technical debt. We need to anticipate growth. We need to anticipate change.



Moving forward

Components

Formalize how we think about the UI

CSS

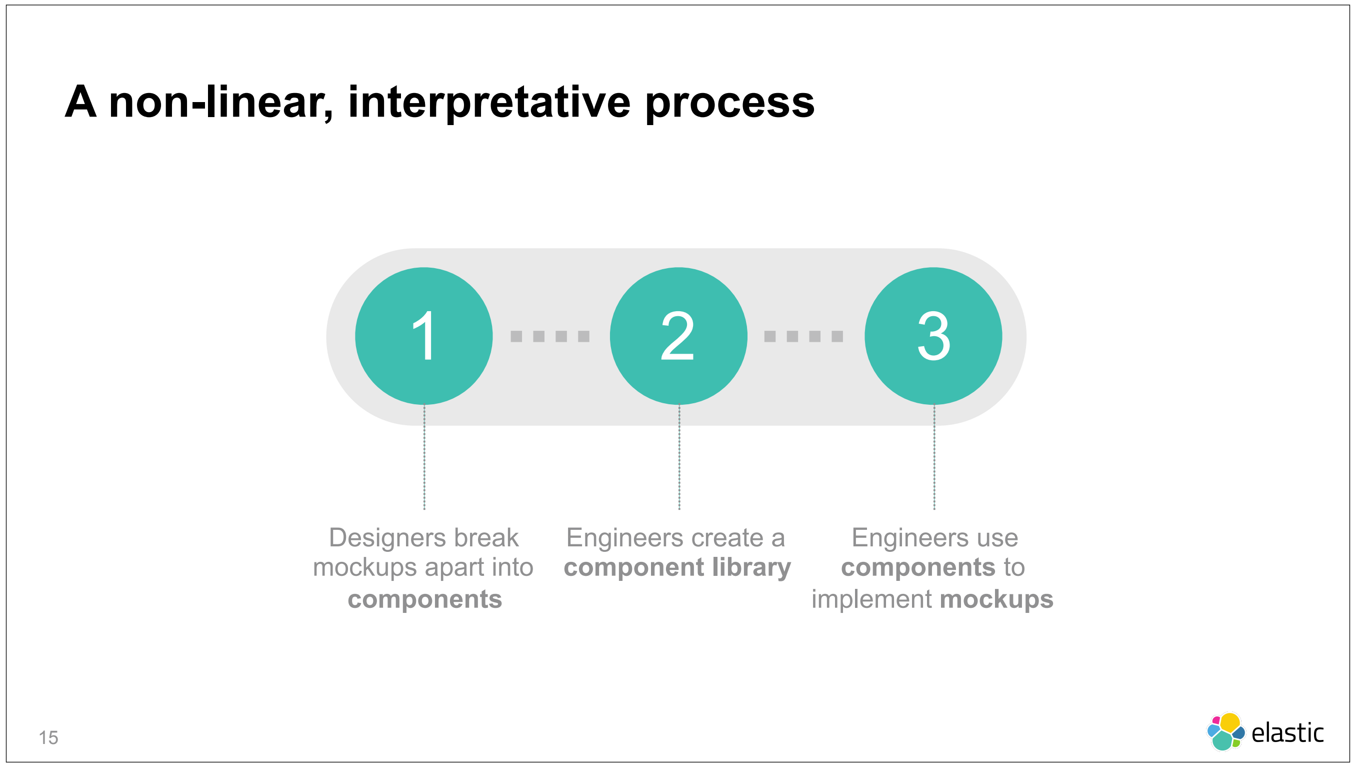
Solve the most intractable part of UI development

React

Tools which support component-based design

elastic

Need a source of truth from which the entire UX derives, so we can make changes in one place and have them ripple throughout the system. We're formalizing the way we think about UI design in terms of components. What's the worst part of building a UI? Writing CSS. We're writing better CSS that's simple and reliable. We're also migrating from Angular to React and starting to build React components.

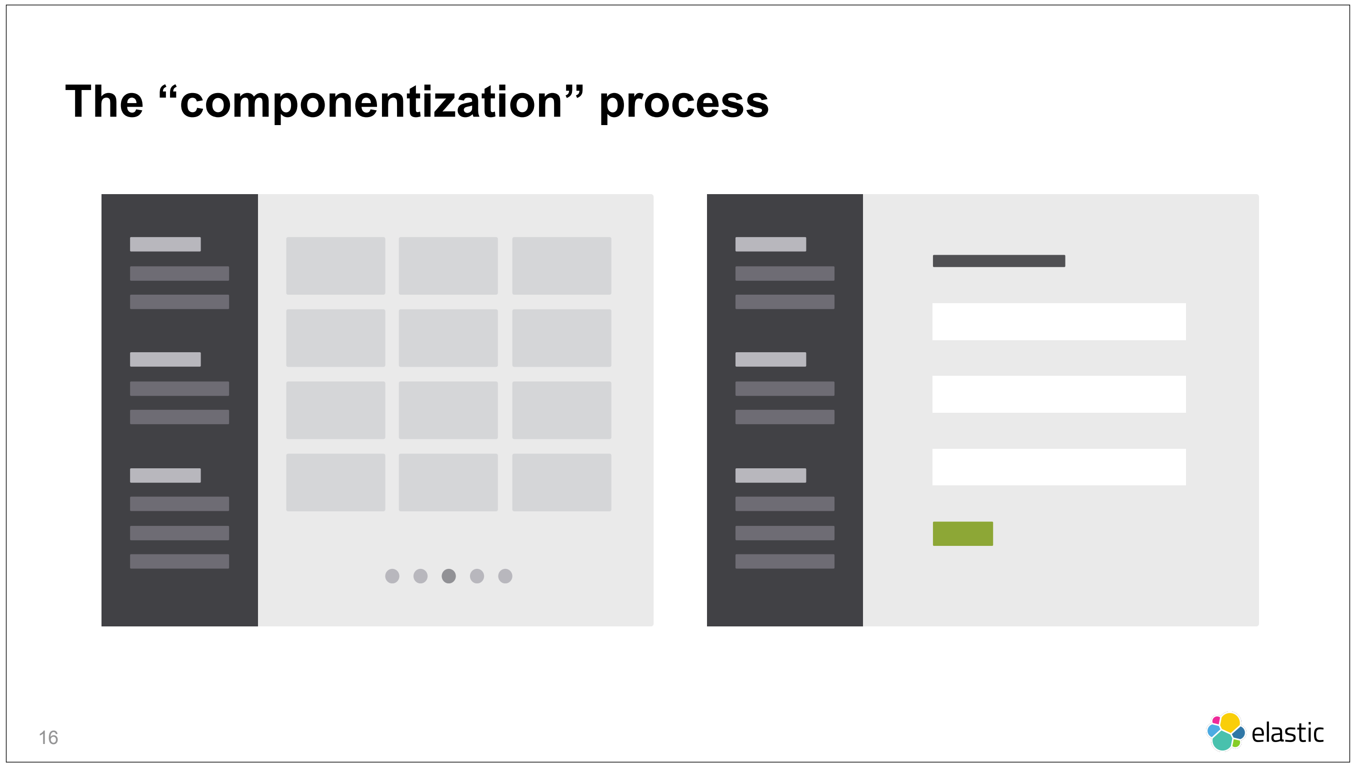


A non-linear, interpretative process

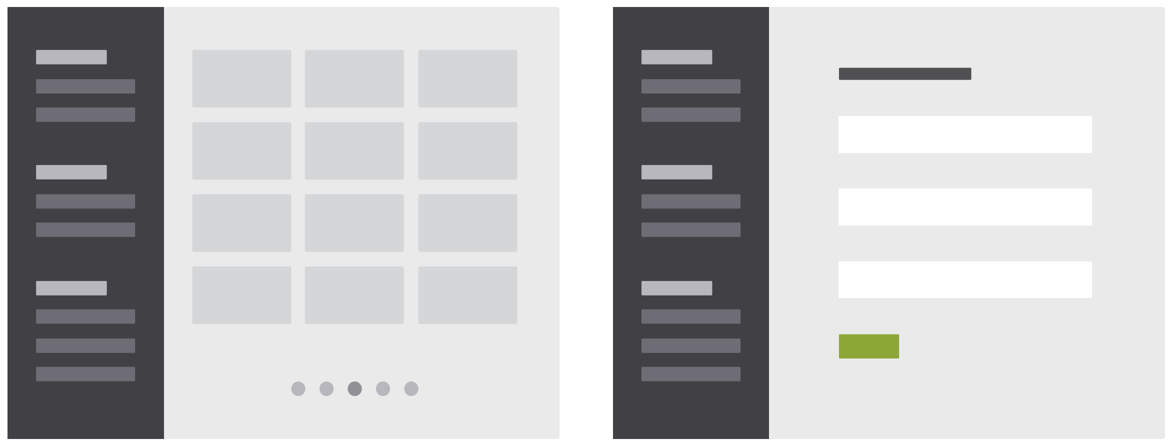
1
2
3
Designers break mockups apart into **components**
Engineers create a **component library**
Engineers use **components** to implement **mockups**

elastic

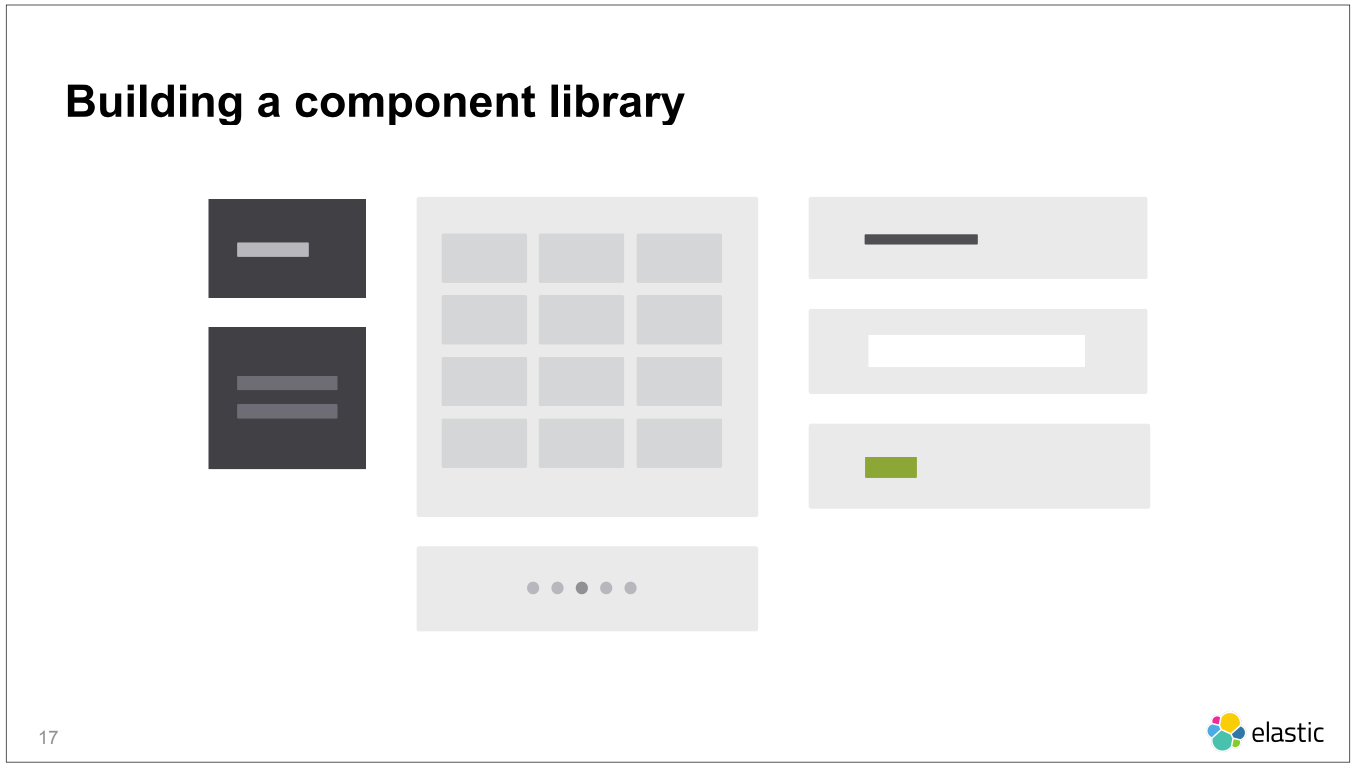
Our new process affects how we think about UI, gives us a consistent design language, and applies engineering best practices to the UI code. When we look at a mockup, we think about individual UI components, not about entire designs. These components are modules. We write code that encapsulates the details of each module. We think in terms of composition of smaller, simple components into larger, complex components.



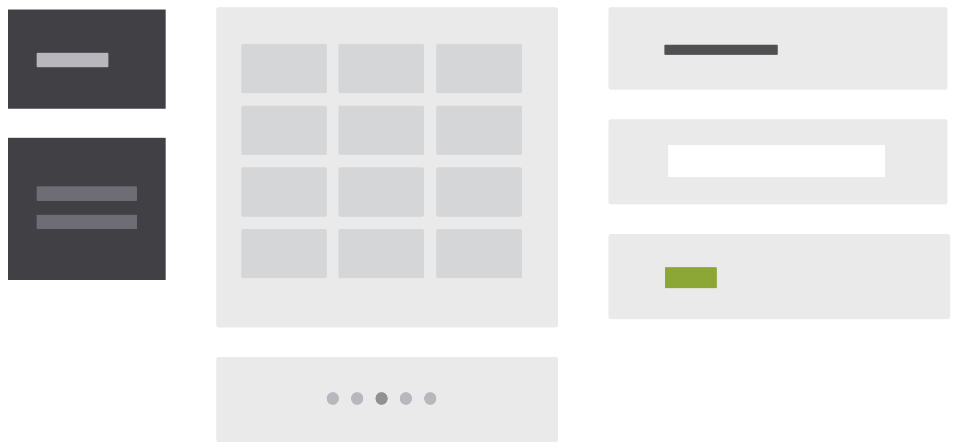
The “componentization” process



Componentization is the process of breaking apart bespoke user interfaces into generalized UI components.
Going back to our earlier example, here’s how we would apply the component-based way of thinking to our designs. We’d take a look at these designs and instead of implementing them in a concrete way…



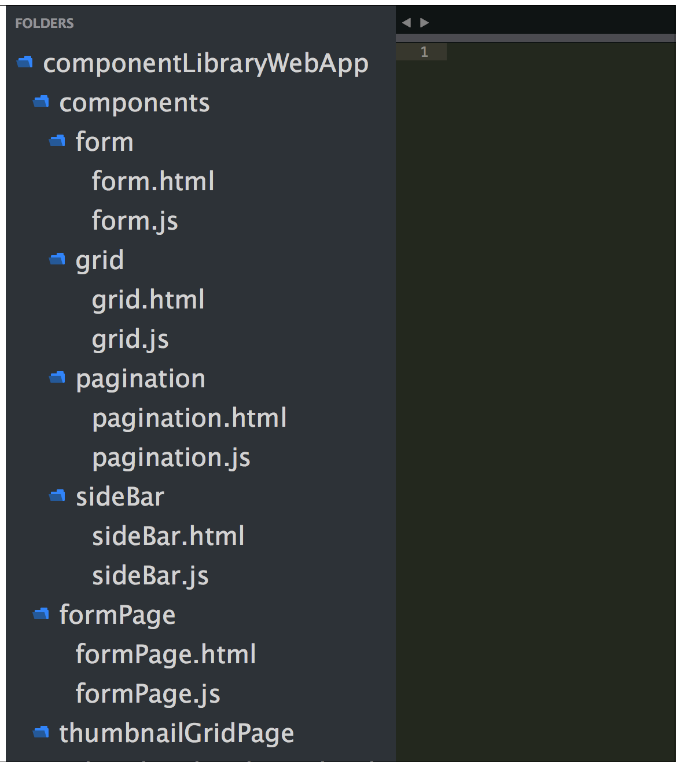
Building a component library



…we'd break them up into the components which comprise them. Page header, column text, table. And then we'd use these components to implement the mockups.

Components form a shared vocabulary.

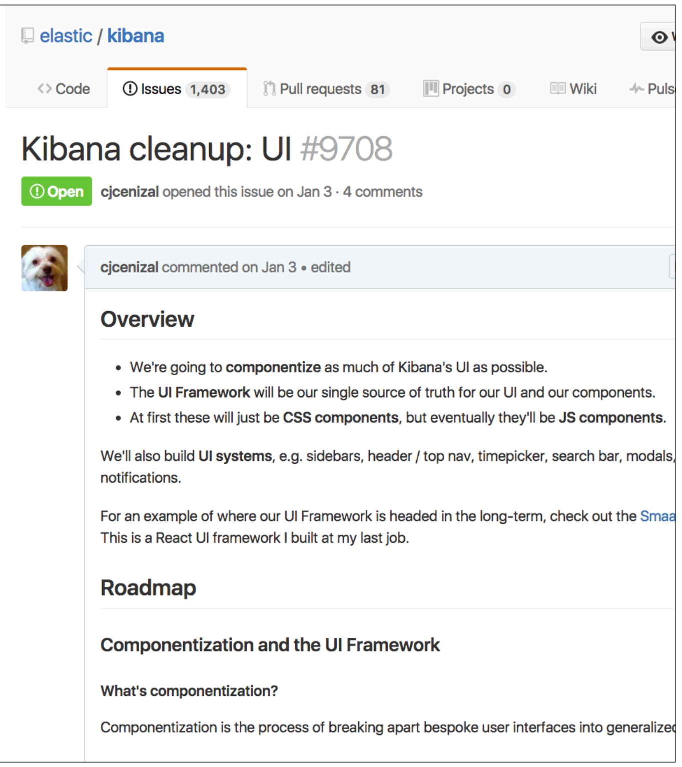
Designers, engineers, and product people can all use it.



```
FOLDERS
componentLibraryWebApp
  components
    form
      form.html
      form.js
    grid
      grid.html
      grid.js
    pagination
      pagination.html
      pagination.js
    sideBar
      sideBar.html
      sideBar.js
  formPage
    formPage.html
    formPage.js
  thumbnailGridPage
```

The result is a library of components that can be used over and over to build new mockups.
Components have both a visual representation and a representation in the code, and they're both identified by the same name. This lets us talk about UI with a common language. Now, whenever a designer, engineer, or product manager refers to a “PageHeader” or “TextColumn”, everybody else in the conversation knows exactly what they're talking about.

This process is, well, a process.



elastic / kibana

Code · Issues 1,403 · Pull requests 81 · Projects 0 · Wiki · Puls

Kibana cleanup: UI #9708

Open · cjcenizal opened this issue on Jan 3 · 4 comments

cjcenizal commented on Jan 3 • edited

Overview

- We're going to **componentize** as much of Kibana's UI as possible.
- The **UI Framework** will be our single source of truth for our UI and our components.
- At first these will just be **CSS components**, but eventually they'll be **JS components**.

We'll also build **UI systems**, e.g. sidebars, header / top nav, timepicker, search bar, modals, notifications.

For an example of where our UI Framework is headed in the long-term, check out the Smaa
This is a React UI framework I built at my last job.

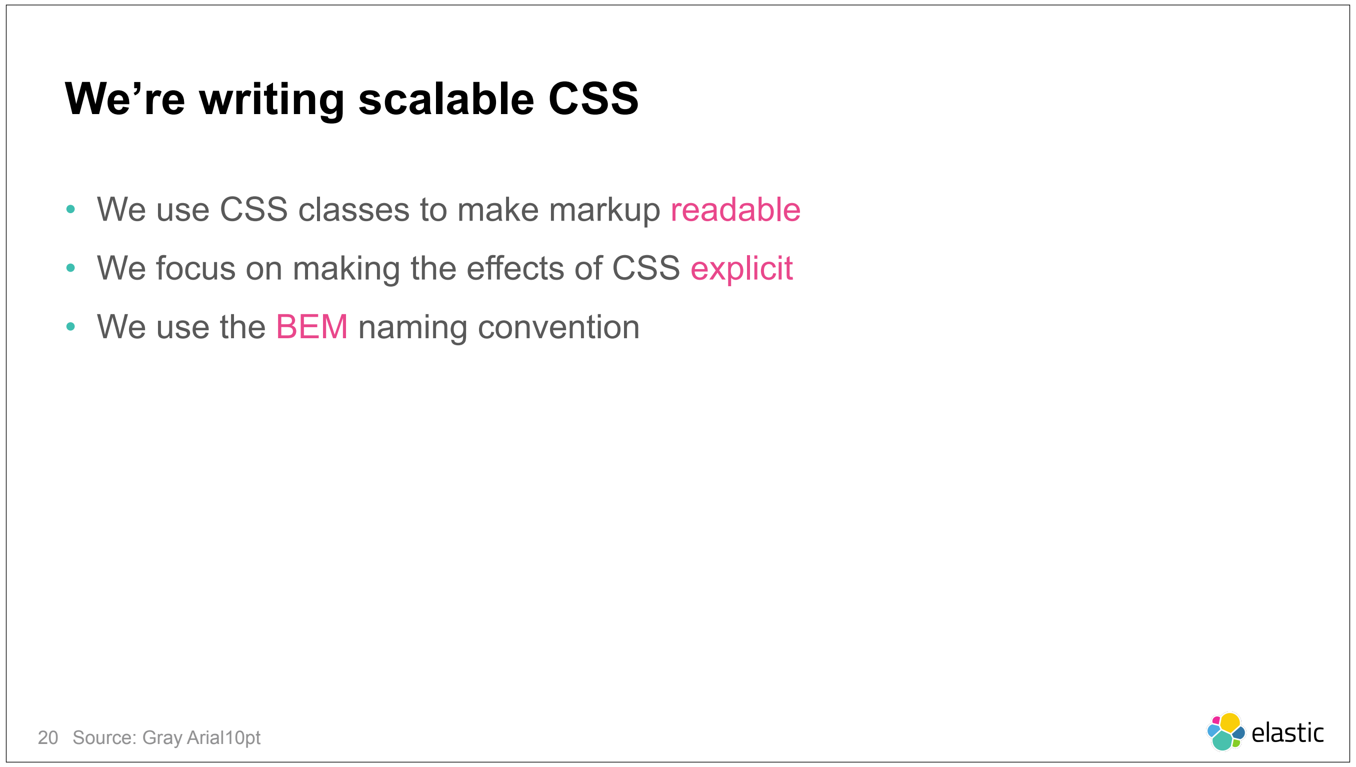
Roadmap

Componentization and the UI Framework

What's componentization?

Componentization is the process of breaking apart bespoke user interfaces into generalized

This is an ongoing process. We have an issue open on the Kibana GitHub repo to track progress if you're interested in following this process or if you want to help us out.



We’re writing scalable CSS

- We use CSS classes to make markup readable
- We focus on making the effects of CSS explicit
- We use the BEM naming convention

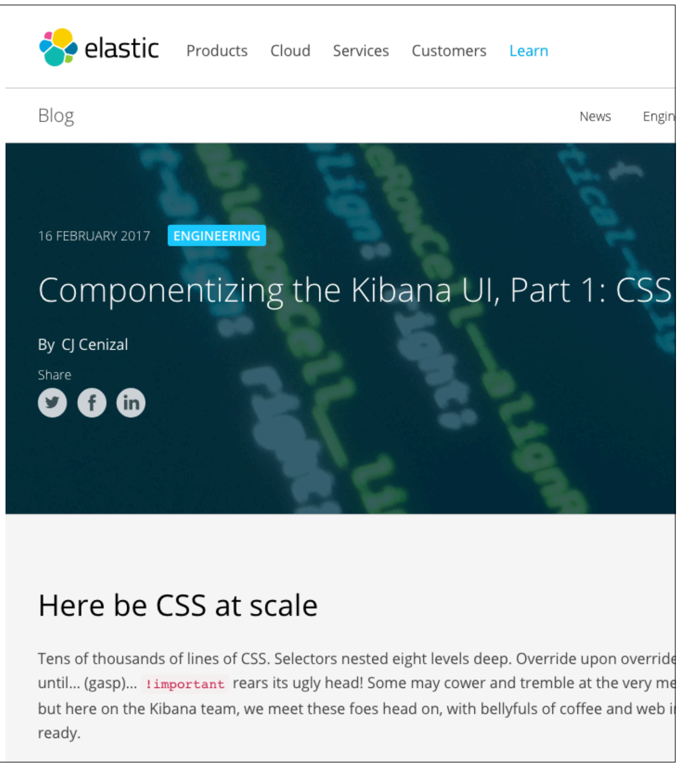
20 Source: Gray Arial10pt

elastic

A huge problem with CSS is trying to read the markup and imagine how it will look when rendered in the browser. Most of us know what a button element generally does, but what if you have 5 different types of button in your app? Using CSS classes, we can give buttons descriptive names that we understand. Just like in code, naming things well makes a big difference.

Another big problem with CSS is figuring out which styles apply to which parts of the markup. This is because some CSS styles can be inherited, which can result in unexpected side effects in your UI. To avoid these surprises, we try to make our CSS as explicit as possible, and avoid relying on inherited styles. We also use the BEM naming convention to name classes with consistent and understandable patterns.

We're sharing what we learn on our blog.


elastic Products Cloud Services Customers Learn

Blog News Engin

16 FEBRUARY 2017 ENGINEERING

Componentizing the Kibana UI, Part 1: CSS

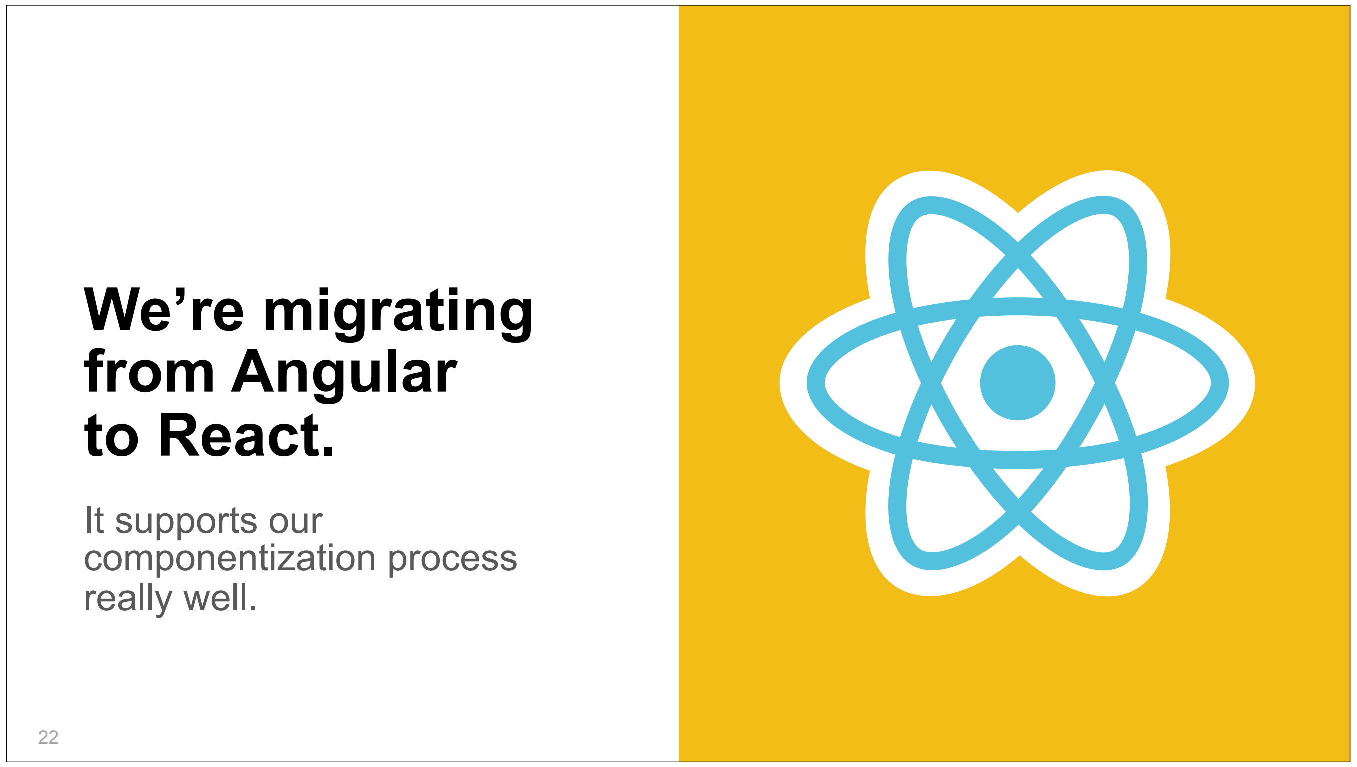
By CJ Cenizal

Share

Here be CSS at scale

Tens of thousands of lines of CSS. Selectors nested eight levels deep. Override upon override until... (gasp)... `!important` rears its ugly head! Some may cower and tremble at the very me but here on the Kibana team, we meet these foes head on, with bellyfuls of coffee and web i ready.

I've written more about how we write CSS on our blog. Feel free to check it out to learn more. Or if you have any questions ask me at the end and I'll be glad to go into more detail.



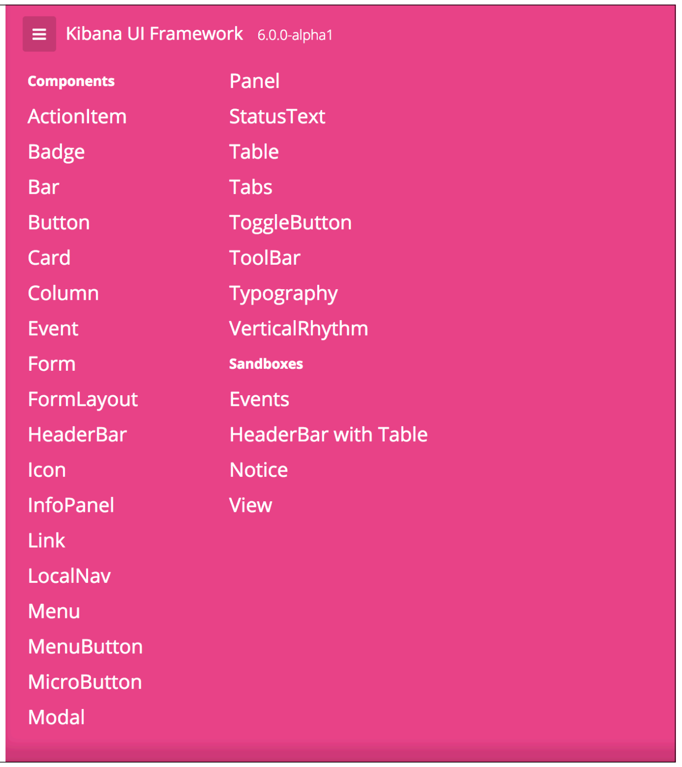
We’re migrating from Angular to React.

It supports our componentization process really well.

Lastly, we’re also migrating from Angular to React. React encourages building small, simple components that can be put together to build more complex components. Components have external dependencies, on things like callbacks and even injected logic and state. This will also help make our code easier to reason about and test.

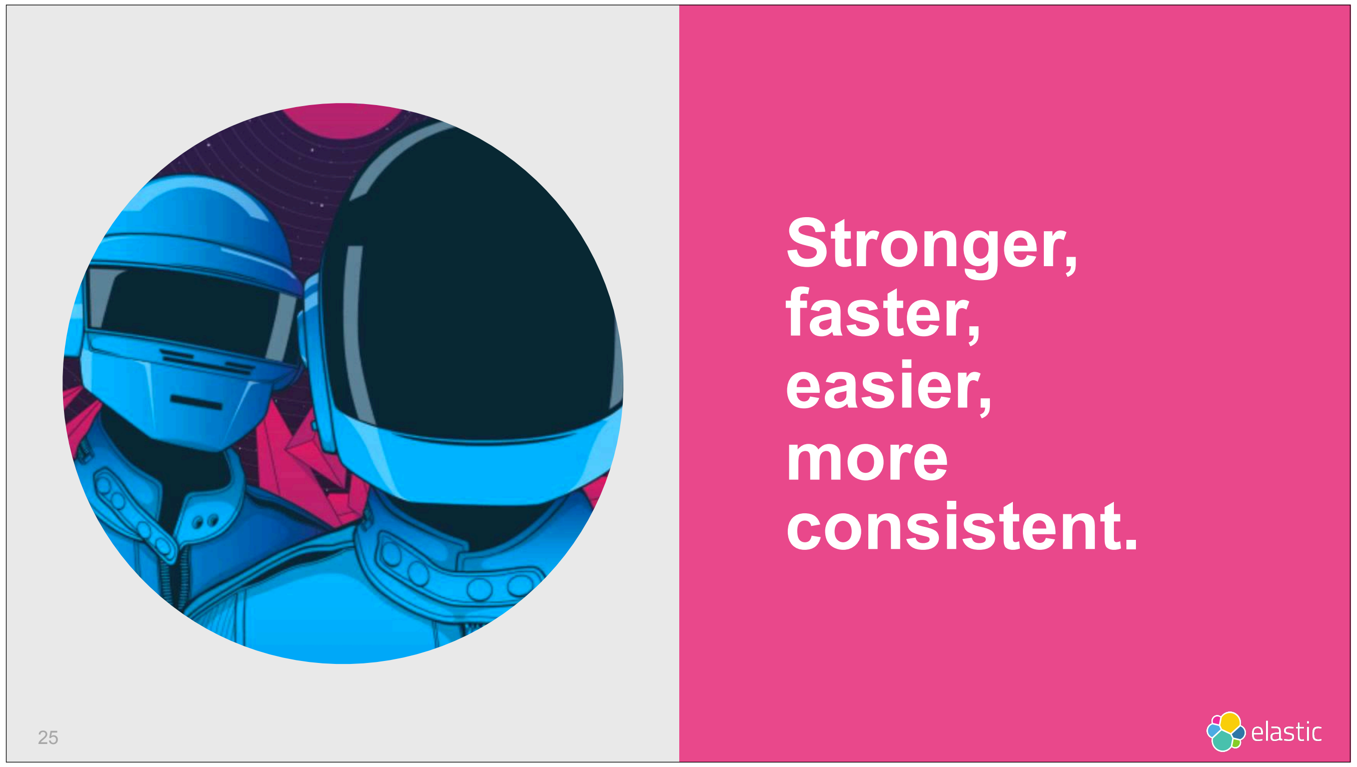
The Kibana UI Framework.

Our single source of truth for components.



All of these components will live in a single UI source of truth: the UI Framework. This is our library of reusable components. An engineer using these components can rest easy knowing that they are tested and demonstrated with interactive examples. Both internal engineers and external contributors can check it out and see what's available, and build user interfaces using these components.

Demo

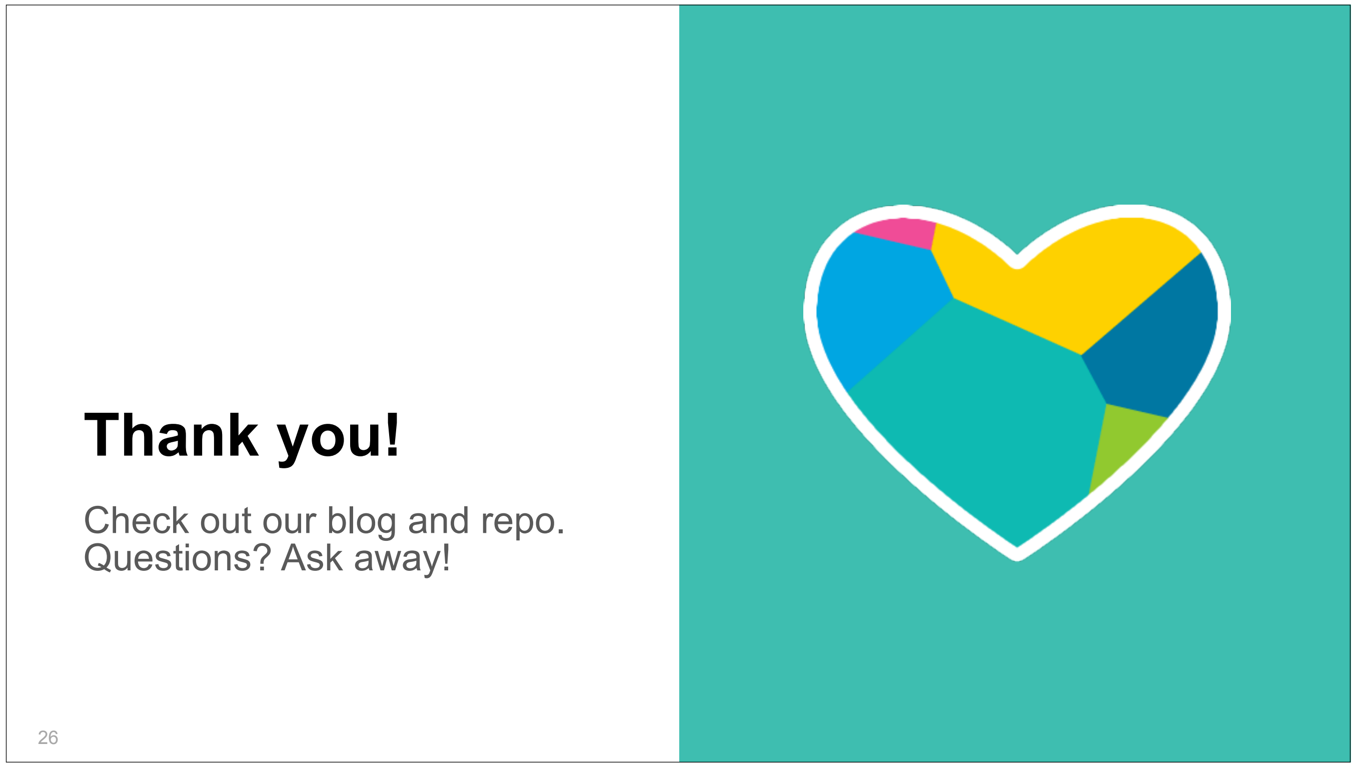


Stronger, faster, easier, more consistent.

elastic

The UI Framework means faster development for us. It also means OSS contributors will be able to write better code, more quickly, the same benefits we gain as team members. Plugin developers: you'll be able to pull components from the UI Framework and build your plugins without having to custom-build the UI. And all of this means a more consistent UI and UX for the end user.

The UI Framework also creates some exciting possibilities:
- Dark theme throughout Kibana
- Easier to create custom “skins”
- Improved global accessibility



Thank you!

Check out our blog and repo.
Questions? Ask away!

So that’s a look at the UI and UX challenges we face on the Kibana team and how we’re solving them. More information is available online on our blog and the Kibana repo on GitHub.
Any questions about anything?