Decoupled site building
Drupal's next challenge

Preston So • 28 Sep 2017 • DrupalCon Vienna 2017

Herzlich Willkommen!

Preston So has been a web developer and designer since 2001, a creative professional since 2004, and a Drupal developer since 2007. As Director of Research and Innovation at Acquia, Preston leads new open-source and research initiatives and helms Acquia Labs, the innovation lab featured in international press.

Previously, Preston contributed to the Spark initiative in the Drupal project, co-founded the Southern Colorado Drupal User Group (est. 2008), and operated an award-winning freelance web and print design studio. Preston has presented keynotes at conferences on three continents in multiple languages.

preston.so@acquia.com • @prestonso • #decoupledsitebuilding

What we'll cover

- The widening gap between developer and marketer
- Editing everywhere and editing everything
- Progressive decoupling and the "black box"
- Fully decoupled front ends: Administrative and public-facing
- The mythical "Edit" button: The case of Prismic
- Decoupled layout: The case of RESTful Panels
- Epilogue: Decoupled content strategy
- Open discussion

1

The widening gap between developer and marketer

Think about how we're using Drupal today.

Drupal's value proposition

Drupal has long prided itself on its unique place at the fulcrum of:

- The **developer**, who benefits from a flexible developer experience and high extensibility.
- The **marketer**, who benefits from contextualized administration tools and editorial access.
- The **user**, who benefits from whatever user experiences are built by both other personas.

Is the CMS as we know it dead?

Yes.

How can we keep up with a widening wilderness of untapped digital experiences?

How can we keep up with CMSes that don't even exist yet?

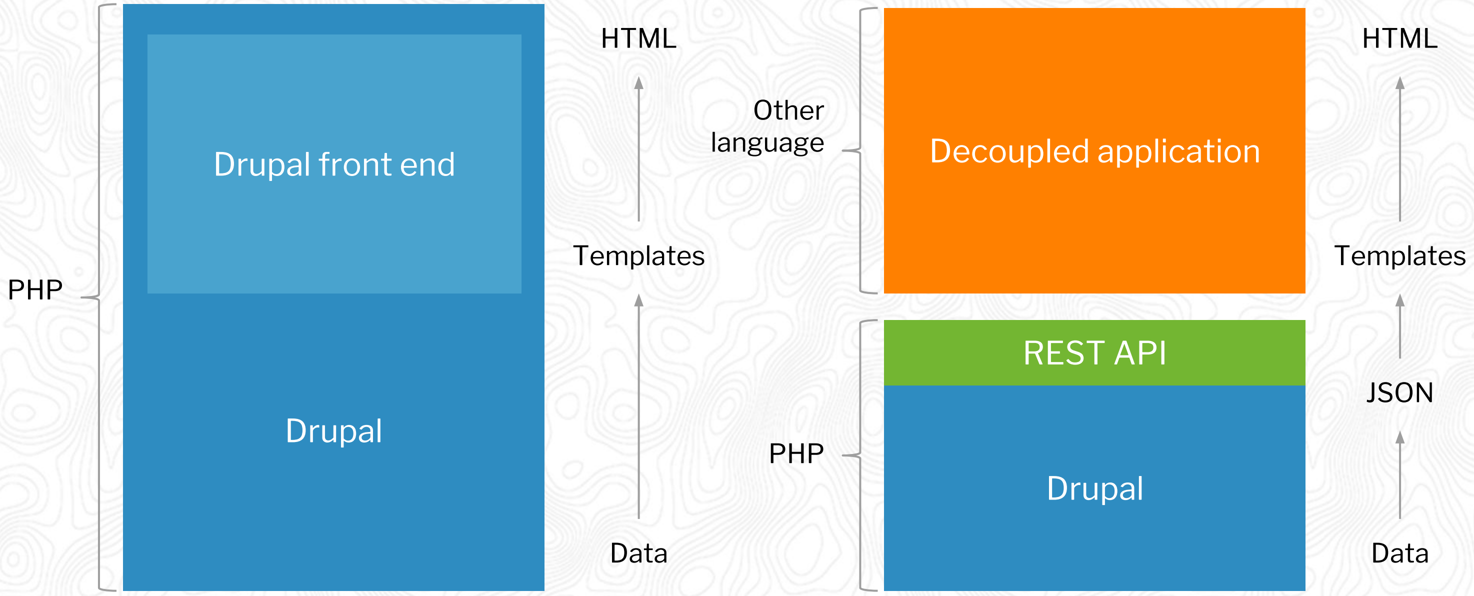
What is the ideal CMS?

It requires a whole new kind of thinking for the **omnichannel**:

- The **developer** retains immense flexibility.
- The **marketer** can use a contextualized editorial experience that is immediately accessible.
- Most critically, the **user** can experience content on any possible channel.

Decoupled Drupal

- **Decoupled Drupal** is the underlying approach that allows for communication with other systems: set-top boxes, augmented reality, etc.
- Decoupled Drupal enables anyone else to "speak Drupal."
- But only having robust APIs is not enough; consider the cautionary tale of headless CMSes.



Decoupled Drupal

Editorial experience

- Often, what developers want is in complete opposition to what marketers want
- **Example:** JavaScript framework agnosticism vs. contextualized front-end tools
- **Cases to consider:** Calypso and React, Laravel and Vue, Prismic

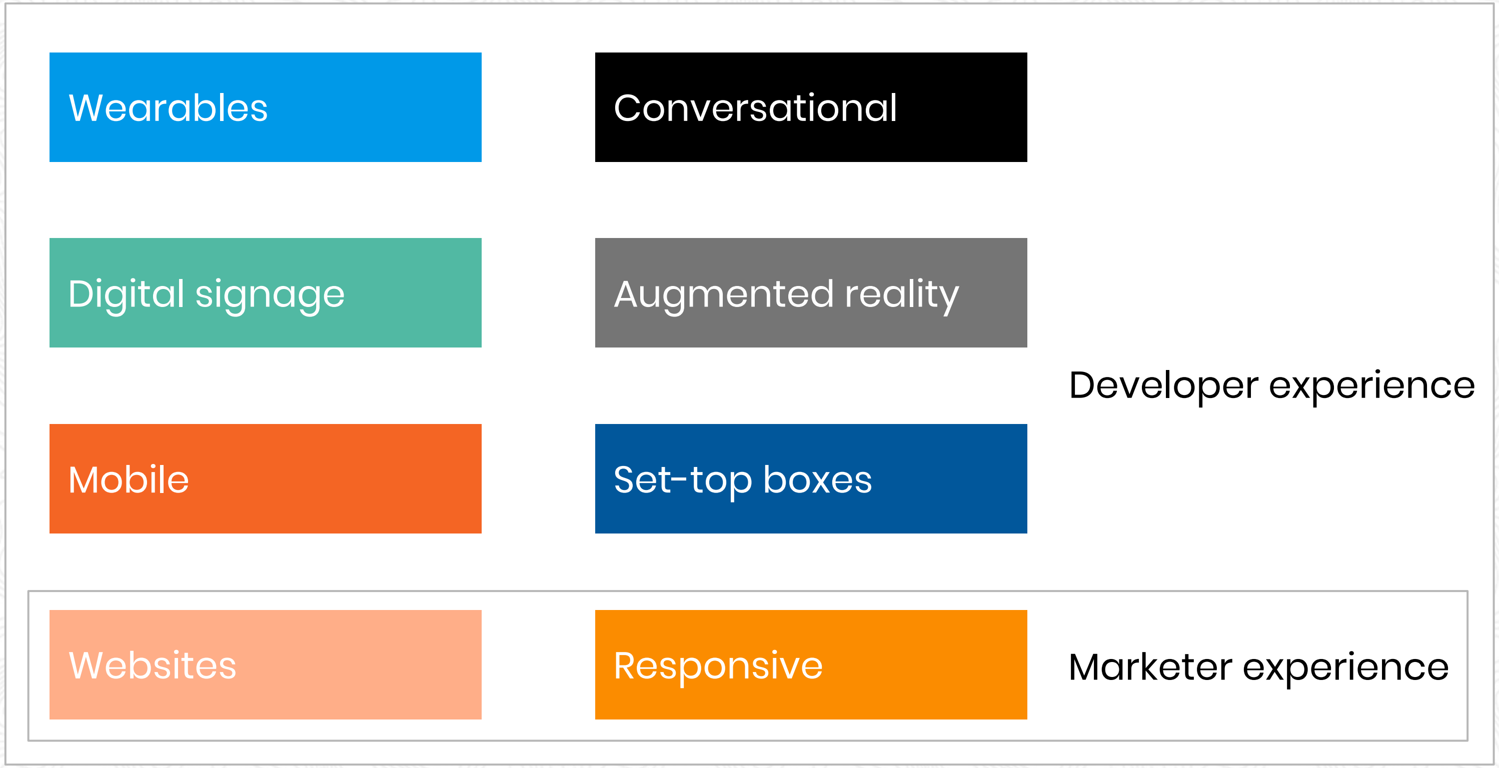
Editorial experience

- How do you edit digital signage?
- How do you edit content driven by augmented and virtual reality?
- How do you edit conversational content?

Drupal's new incongruity

Drupal's value proposition is becoming incongruous between the three personas: the user, the marketer, and the developer.

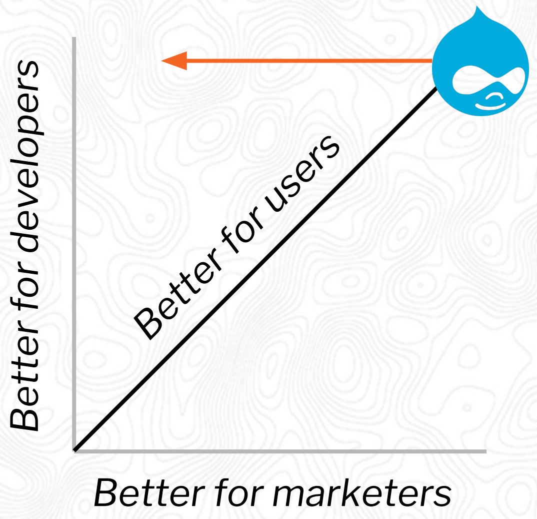
Is this irreconcilable?



Drupal's new incongruity

Drupal's new incongruity

The developer–user axis

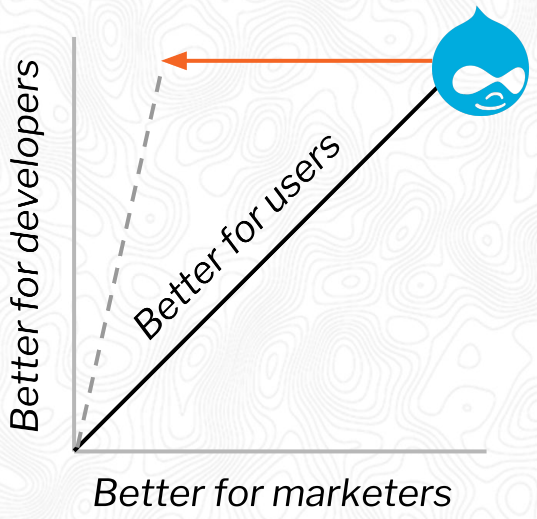


The marketer–user axis

"Better for users" increasingly means "better on more devices."

Drupal's new incongruity

The developer–user axis



The marketer–user axis

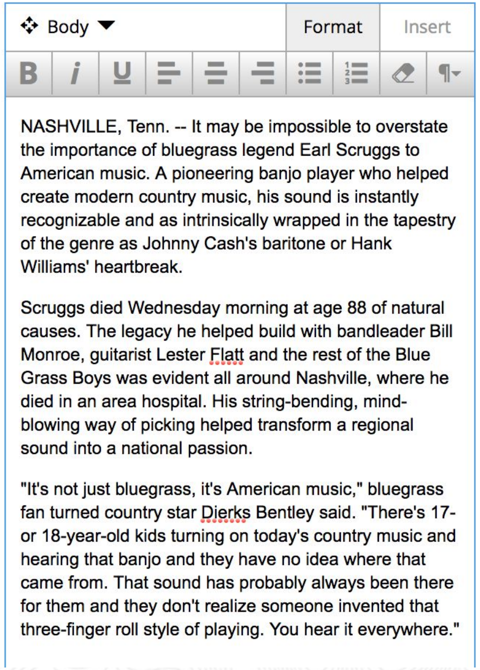
A better outcome for users relies more on custom work by developers.

2

Editing everywhere and editing everything

Edit everywhere

- We already tried it with responsive editing
- Most people will resort to desktop for more complex operations
- The full breadth of functionality isn't available
- User experience deteriorates considerably on certain devices



NASHVILLE, Tenn. -- It may be impossible to overstate the importance of bluegrass legend Earl Scruggs to American music. A pioneering banjo player who helped create modern country music, his sound is instantly recognizable and as intrinsically wrapped in the tapestry of the genre as Johnny Cash's baritone or Hank Williams' heartbreak.

Scruggs died Wednesday morning at age 88 of natural causes. The legacy he helped build with bandleader Bill Monroe, guitarist Lester Flatt and the rest of the Blue Grass Boys was evident all around Nashville, where he died in an area hospital. His string-bending, mind-blowing way of picking helped transform a regional sound into a national passion.

"It's not just bluegrass, it's American music," bluegrass fan turned country star Dierks Bentley said. "There's 17- or 18-year-old kids turning on today's country music and hearing that banjo and they have no idea where that came from. That sound has probably always been there for them and they don't realize someone invented that three-finger roll style of playing. You hear it everywhere."

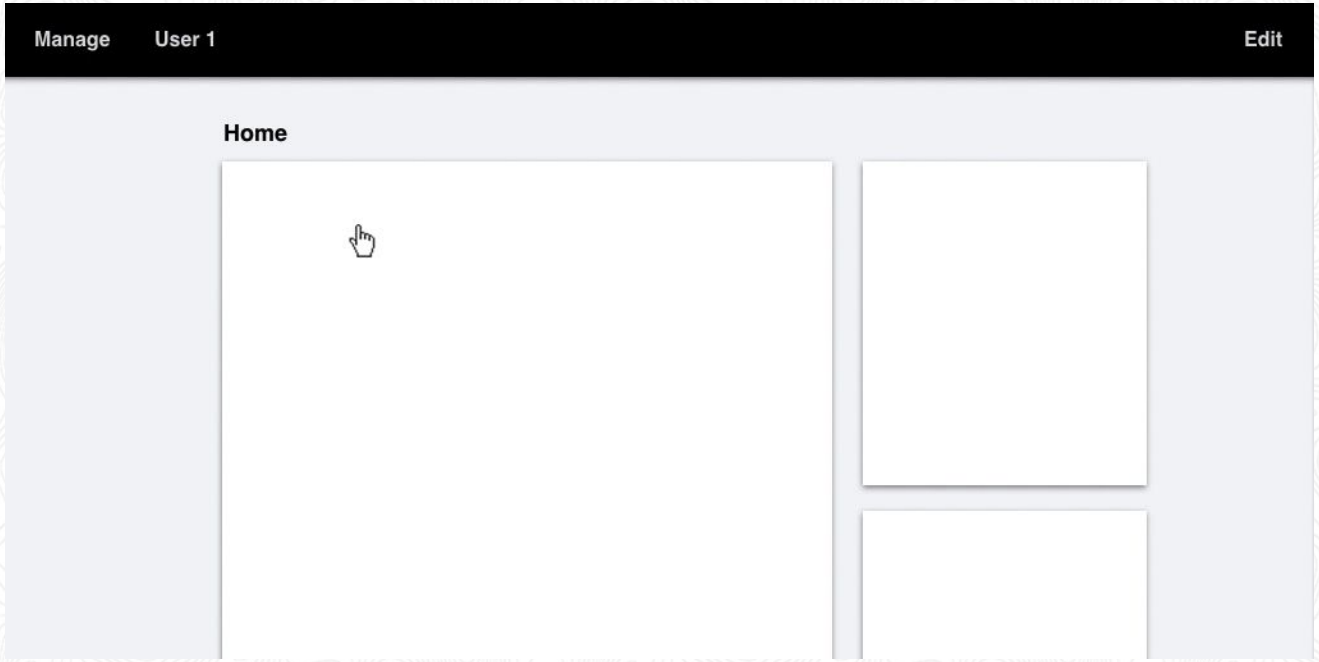
The spectrum of edit everywhere

Fewer channels	More channels
Better usability on fewer devices	**Worse usability on more devices**
Fewer technology stacks to maintain	**More technology stacks to maintain**
Fewer devices that need unique interface design	**More devices that need unique interface design**

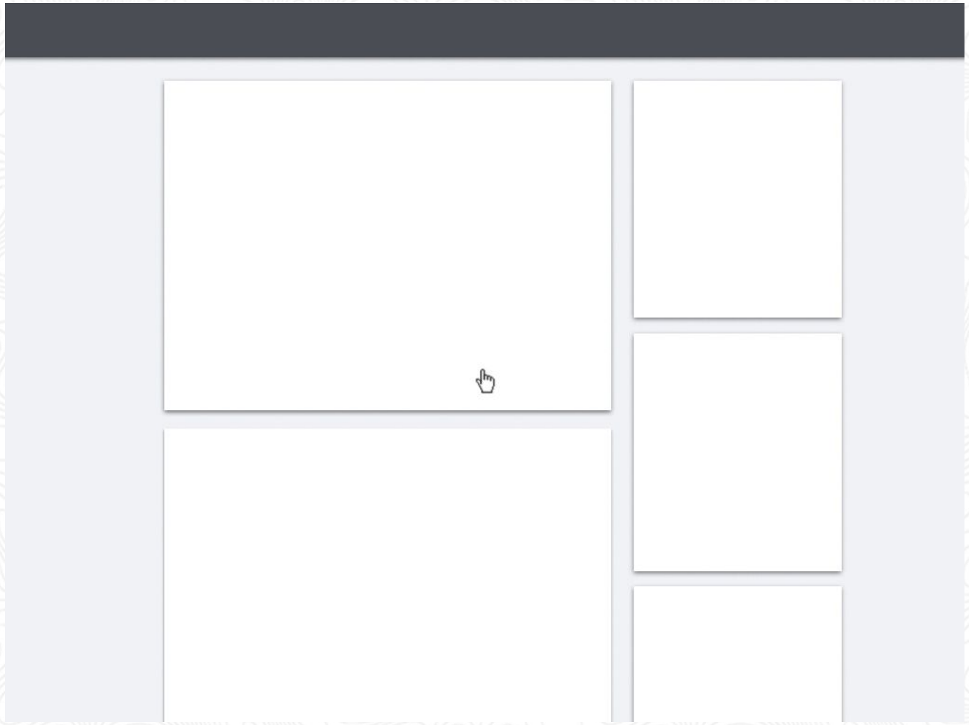
Edit everything

- Edit every channel on "Drupal" through outside-in interfaces and API-first Drupal
- Consistent and seamless user experience across all channel editorial experiences
- Drupal for other experiences should be indistinguishable from Drupal for web experiences

Outside-in interfaces



Outside-in interfaces



Fewer channels	More channels
Better usability on fewer devices	**Worse usability on more devices**
Fewer emulation techniques to maintain	**More emulation techniques to maintain**
Less need for developer assistance for editorial preview	**More need for developer assistance for editorial preview**

The spectrum of edit everything

No silver bullet

- **Editing everywhere** requires us to build editorial interfaces for every device, but it will eventually reach an extreme where interfaces are unusable.
- **Editing everything** requires us to include emulators or provide high-fidelity preview via infrastructure, but it will eventually reach an extreme where infrastructural demand becomes too high.

As the number of channels grows, Drupal is stuck between a rock and a hard place

Drupal currently has examples where this tension is clearly evident

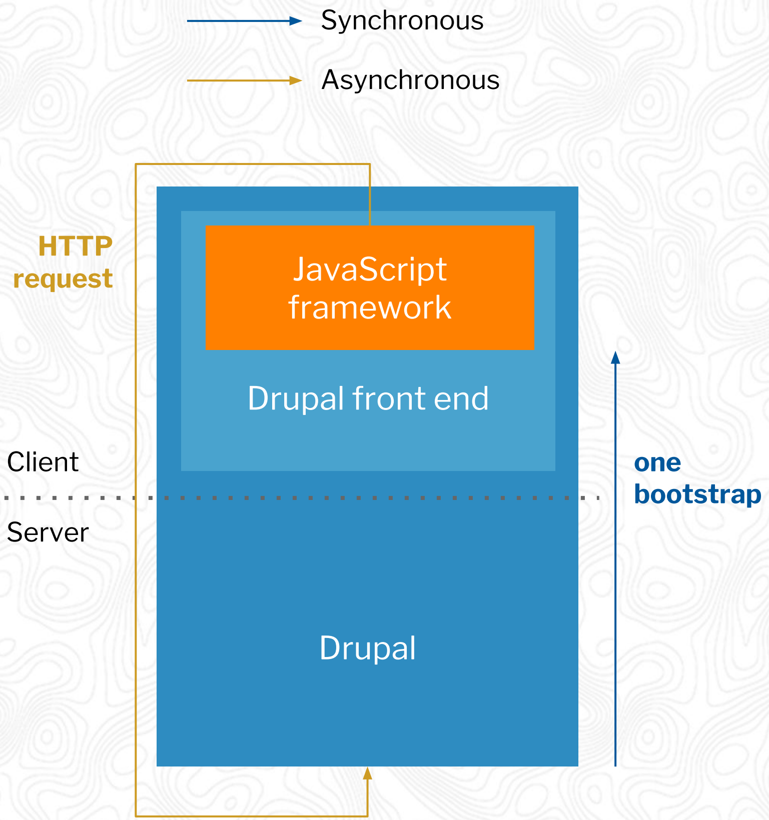
Contextual administration
involves in-context editorial and site building actions within the front end

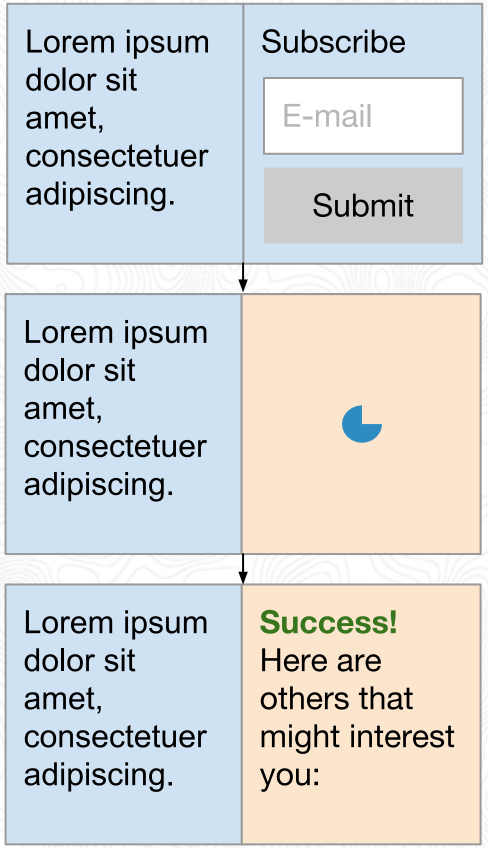
Contextualized administration

- In-place editing
- Contextual links
- Toolbar
- In-context layout management

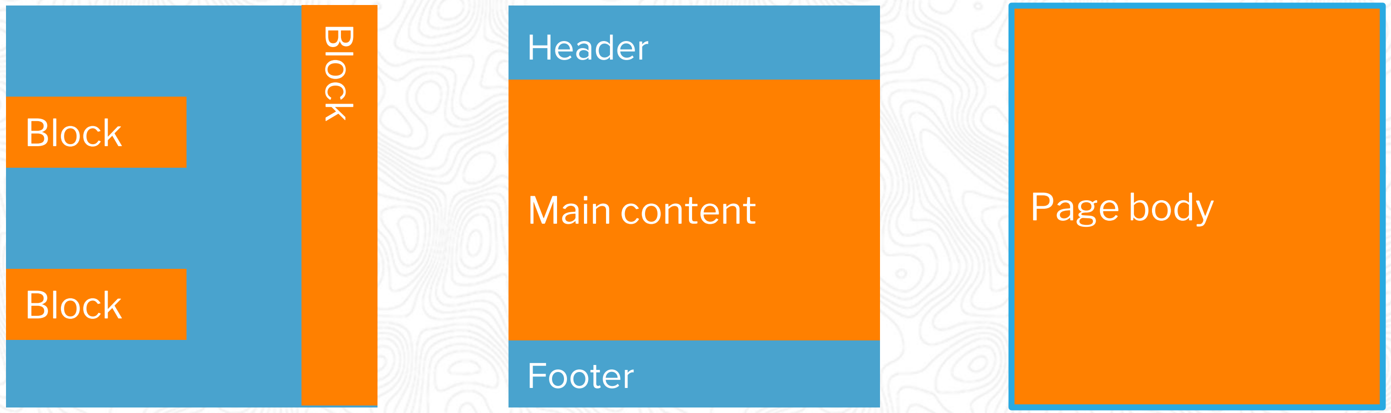
3

Progressive decoupling and the "black box"




Lorem ipsum dolor sit amet, consectetuer adipiscing.
Subscribe
E-mail
Submit
Lorem ipsum dolor sit amet, consectetuer adipiscing.
Lorem ipsum dolor sit amet, consectetuer adipiscing.
Success! Here are others that might interest you:

Progressively decoupled Drupal



Decoupled blocks
Interactivity scoped to blocks; Drupal controls layout

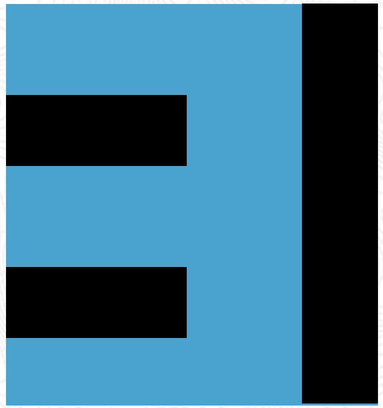
Decoupled main area
Main content handed over to JS; Drupal provides static routes and initial output

Decoupled page body
Entire page body handed over to JS; Drupal provides initial state on page load

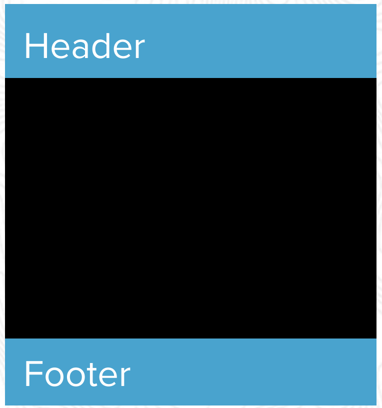
Less **Extent of page controlled by JavaScript** More

🟦 Drupal- (PHP-) controlled 🟧 JavaScript-controlled

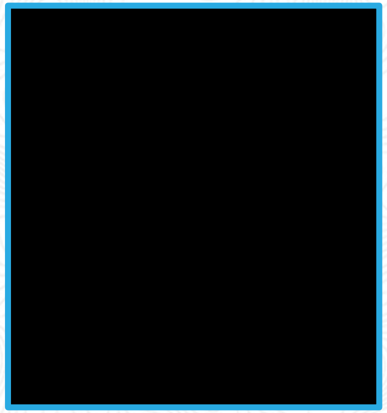
The spectrum of progressive decoupling approaches



Decoupled blocks
Interactivity scoped to blocks; Drupal controls layout



Decoupled main area
Main content handed over to JS; Drupal provides static routes and initial output



Decoupled page body
Entire page body handed over to JS; Drupal provides initial state on page load

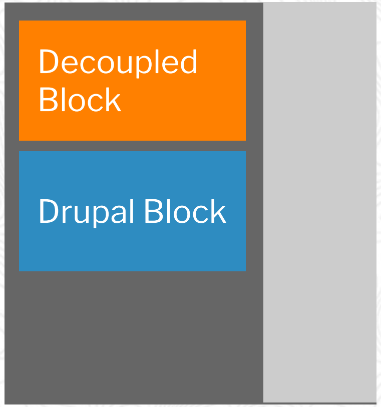
Less **Extent of page controlled by JavaScript** More

🟦 Drupal- (PHP-) controlled ⬛ Drupal "black boxes"

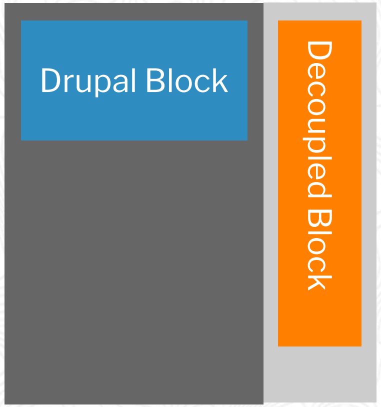
Progressive decoupling "black boxes"

Decoupled Blocks

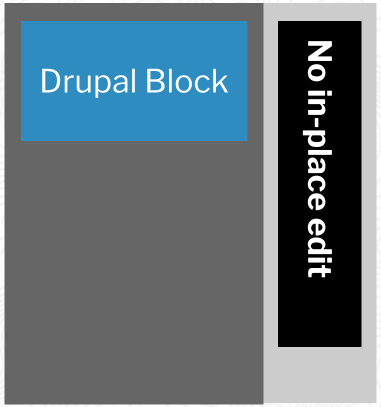
- **Decoupled Blocks** forges an equilibrium between the site builders manipulating layouts and front-end developers manipulating page behavior — in other words, both must compromise on something.
- It's a framework-agnostic module allowing JavaScript components to render into blocks.
- drupal.org/project/pdb



Site builder moves block from one area to another



This enables simple visual assembly for editors and site builders



But these JavaScript components are often "black boxes" and frustrating for editors

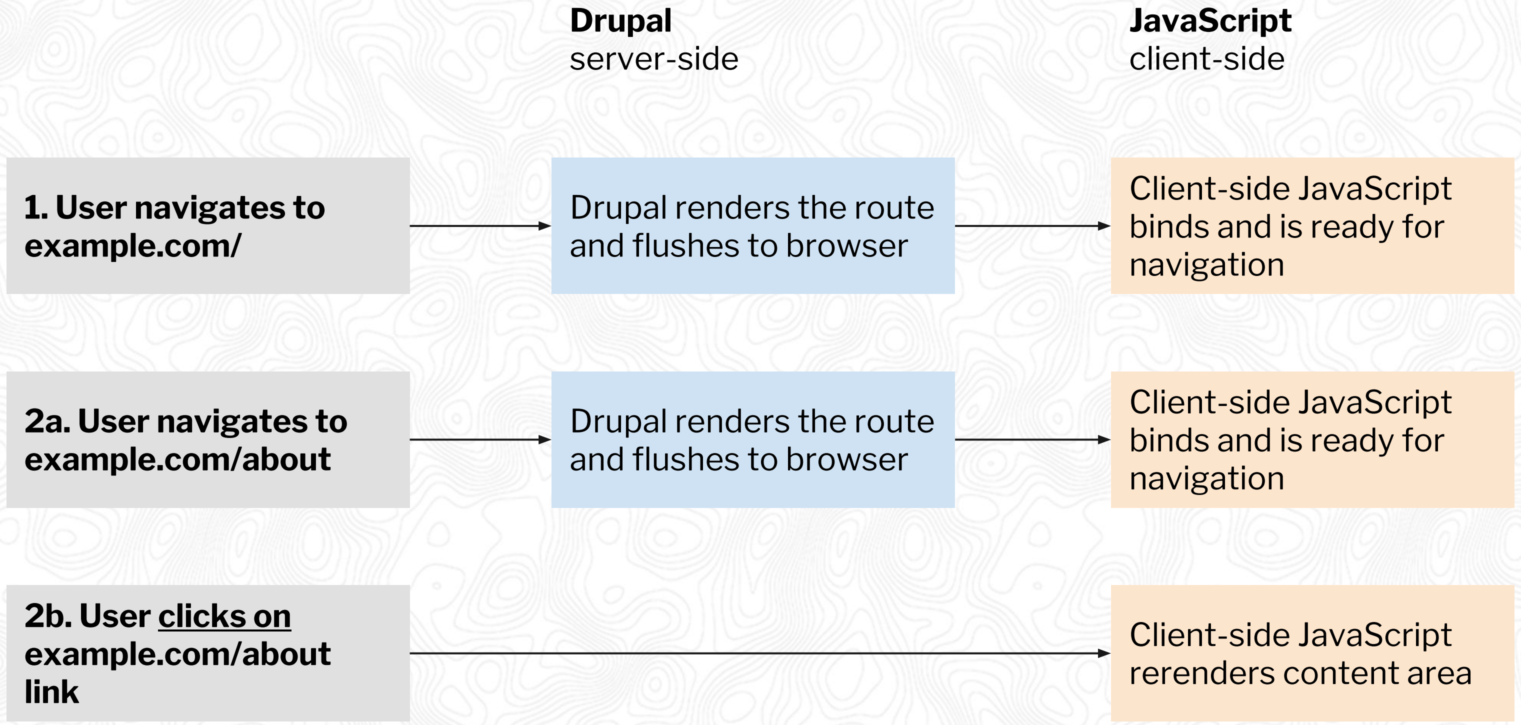
Decoupled Blocks

Also no silver bullet

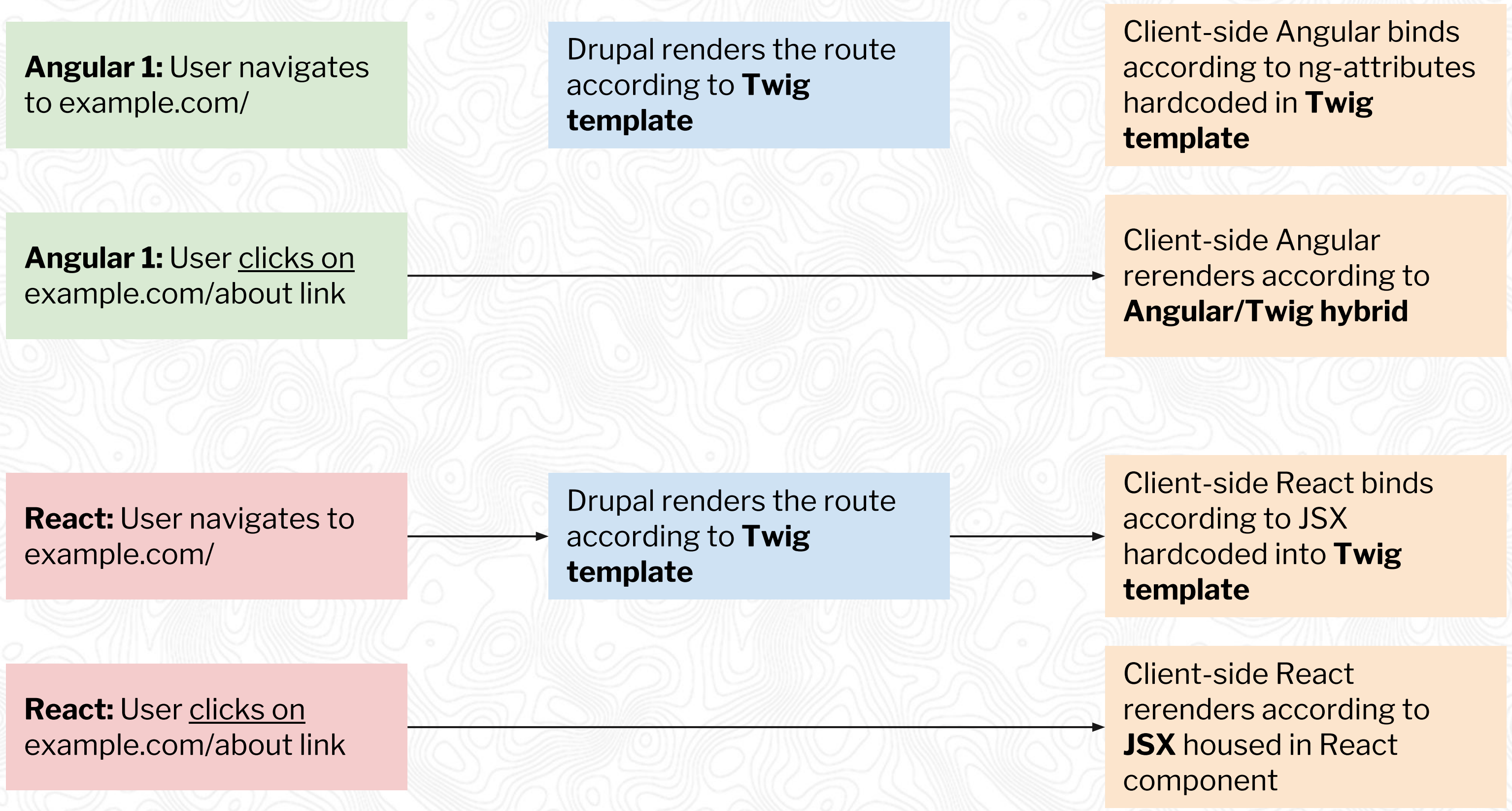
- **Development practices** differ wildly between Drupal and JavaScript frameworks, which presume that front-end developers wield full control over layout and structure.
- **The current lack of harmony** between Drupal's own systems and APIs and those found JavaScript frameworks compounds the gap between the two.

Areas of concern

- Templating
- Routing
- Rendering


Drupal
server-side
JavaScript
client-side
1. User navigates to example.com/
Drupal renders the route and flushes to browser
Client-side JavaScript binds and is ready for navigation
2a. User navigates to example.com/about
Drupal renders the route and flushes to browser
Client-side JavaScript binds and is ready for navigation
2b. User clicks on example.com/about link
Client-side JavaScript rerenders content area

Drupal routes as a superset of JavaScript routes



Angular 1: User navigates to example.com/

Drupal renders the route according to **Twig template**

Client-side Angular binds according to ng-attributes hardcoded in **Twig template**

Angular 1: User clicks on example.com/about link

Client-side Angular rerenders according to **Angular/Twig hybrid**

React: User navigates to example.com/

Drupal renders the route according to **Twig template**

Client-side React binds according to JSX hardcoded into **Twig template**

React: User clicks on example.com/about link

Client-side React rerenders according to **JSX** housed in React component

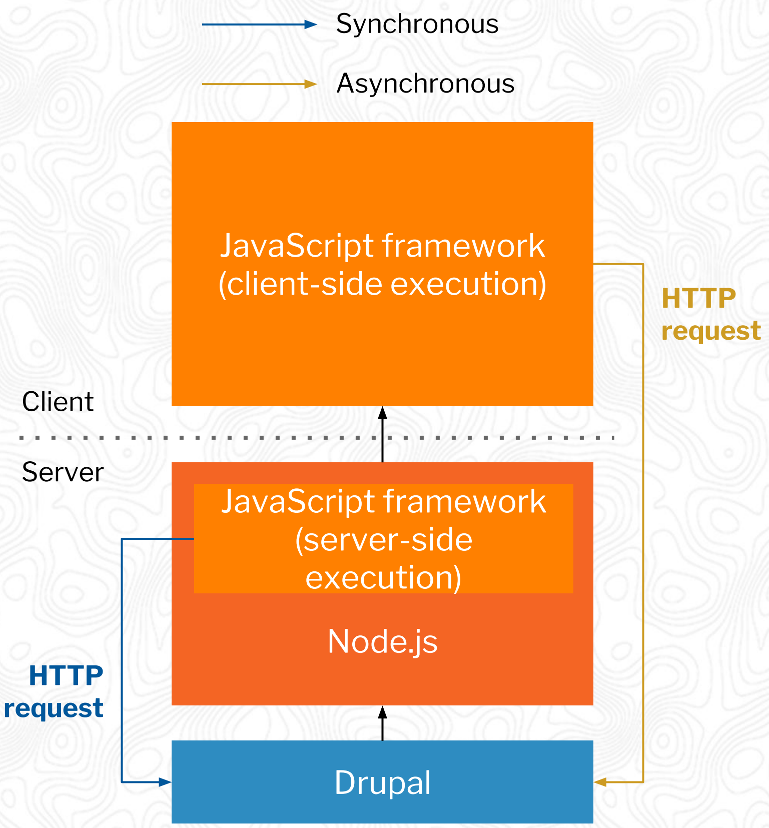
Issue: Template duplication

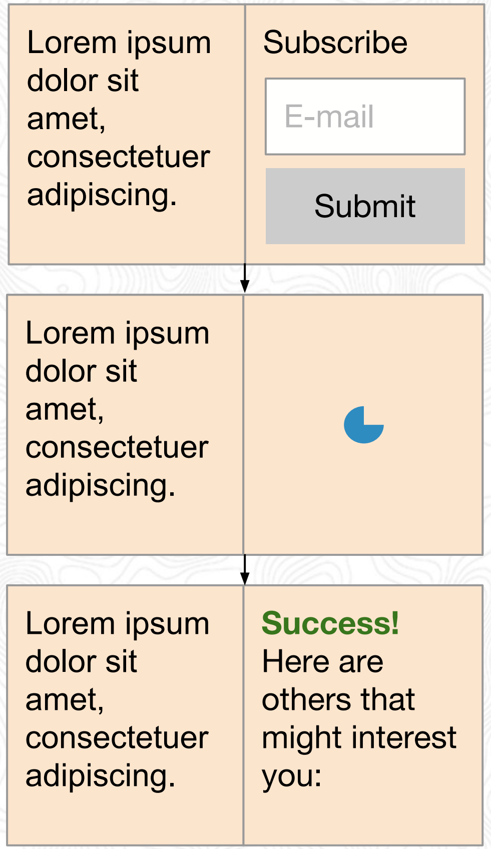
Progressive decoupling comes with expensive tradeoffs

Progressive decoupling may be more trouble than it's worth

4

Fully decoupled front ends
Administrative and public-facing

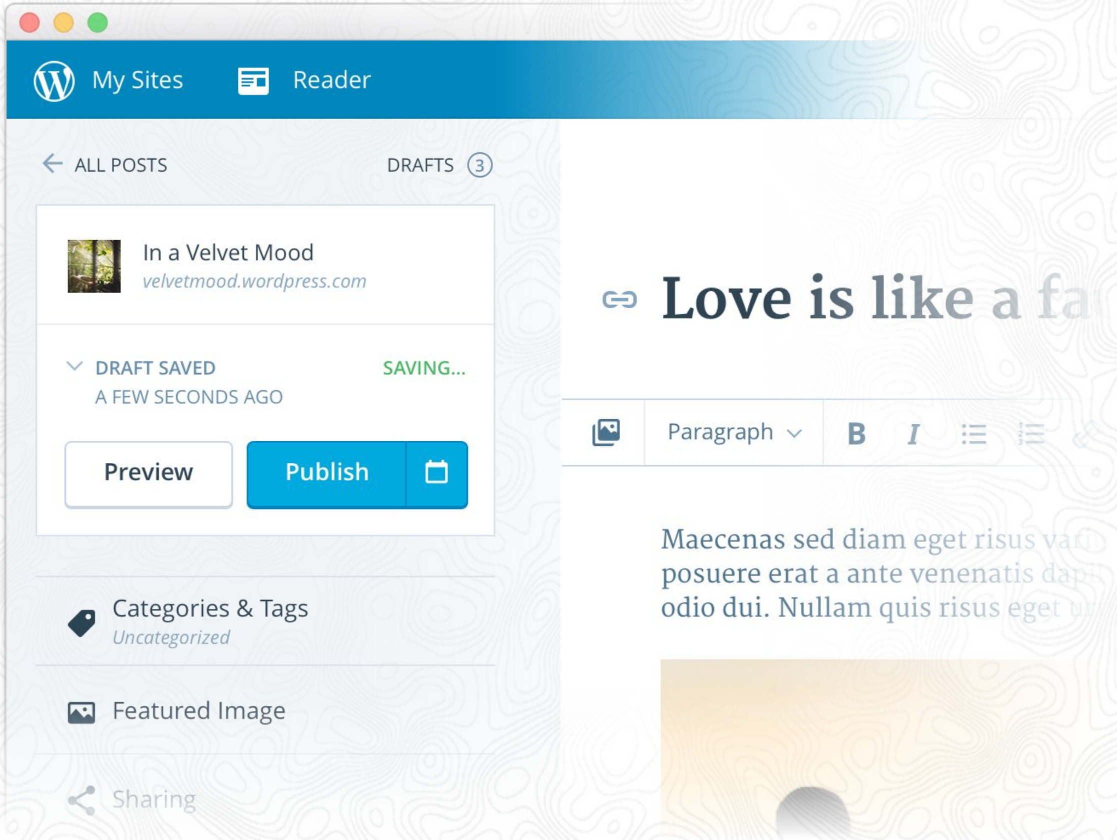

Synchronous
Asynchronous
JavaScript framework (client-side execution)
HTTP request
Client
Server
JavaScript framework (server-side execution)
Node.js
HTTP request
Drupal


Lorem ipsum dolor sit amet, consectetuer adipiscing.
Subscribe
E-mail
Submit
Lorem ipsum dolor sit amet, consectetuer adipiscing.
Lorem ipsum dolor sit amet, consectetuer adipiscing.
Success! Here are others that might interest you:

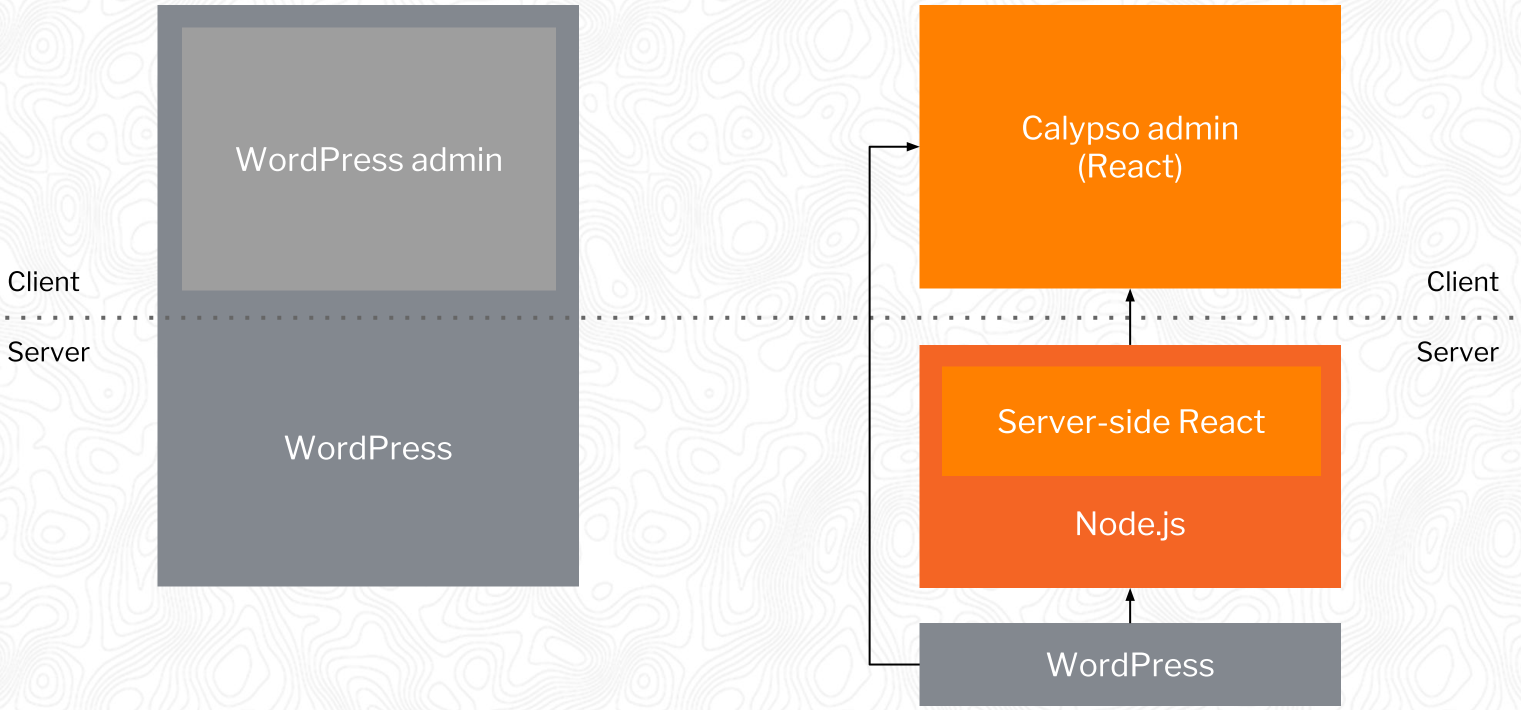
Fully decoupled Drupal

Fully decoupled front ends

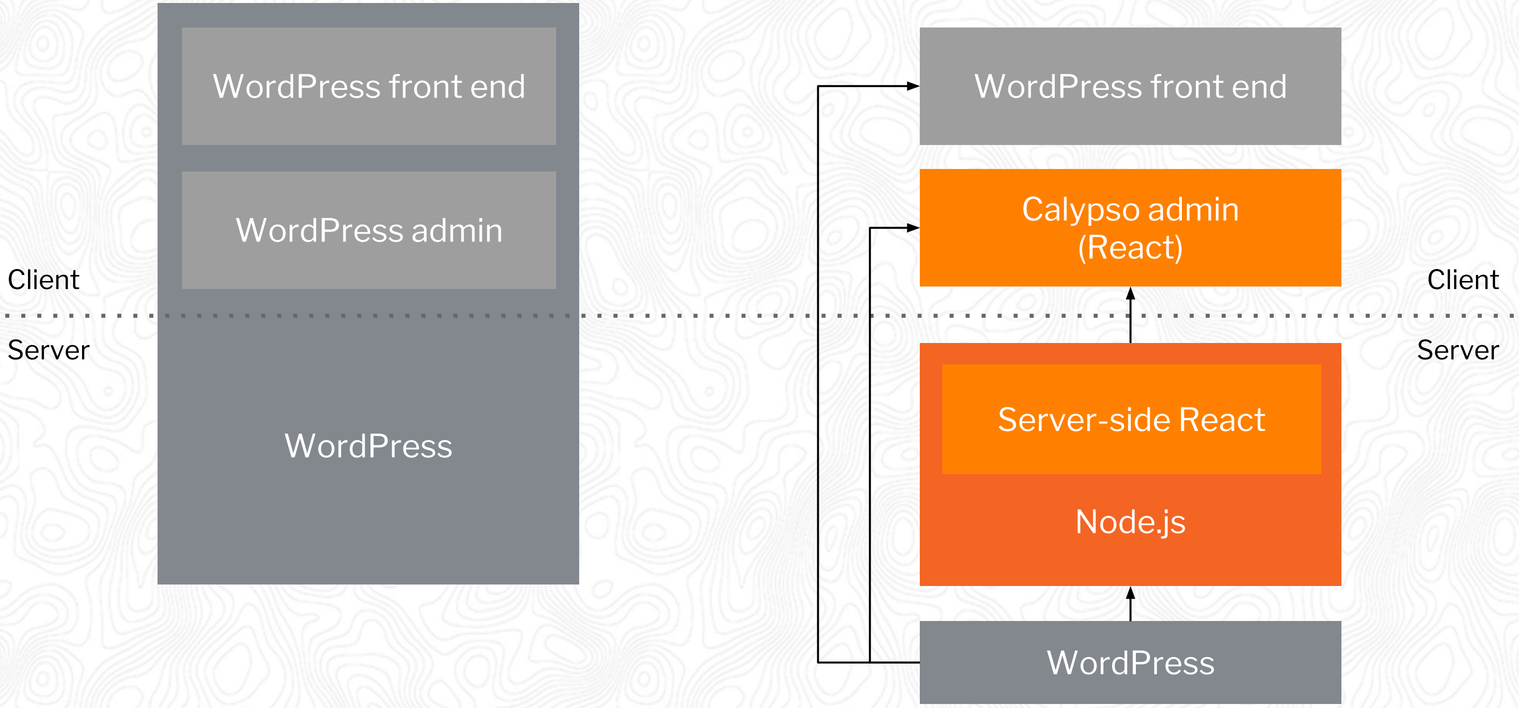
- **Public-facing front ends** are an approach typically chosen to accelerate development of the end user experience by JavaScript developers — and require a strong client understanding of the tradeoffs.
- **Administrative front ends** are replacements for the Drupal editorial interface which provide the same functionality as the traditional administrative "back end" (e.g. Seven OOTB).



WordPress Calypso



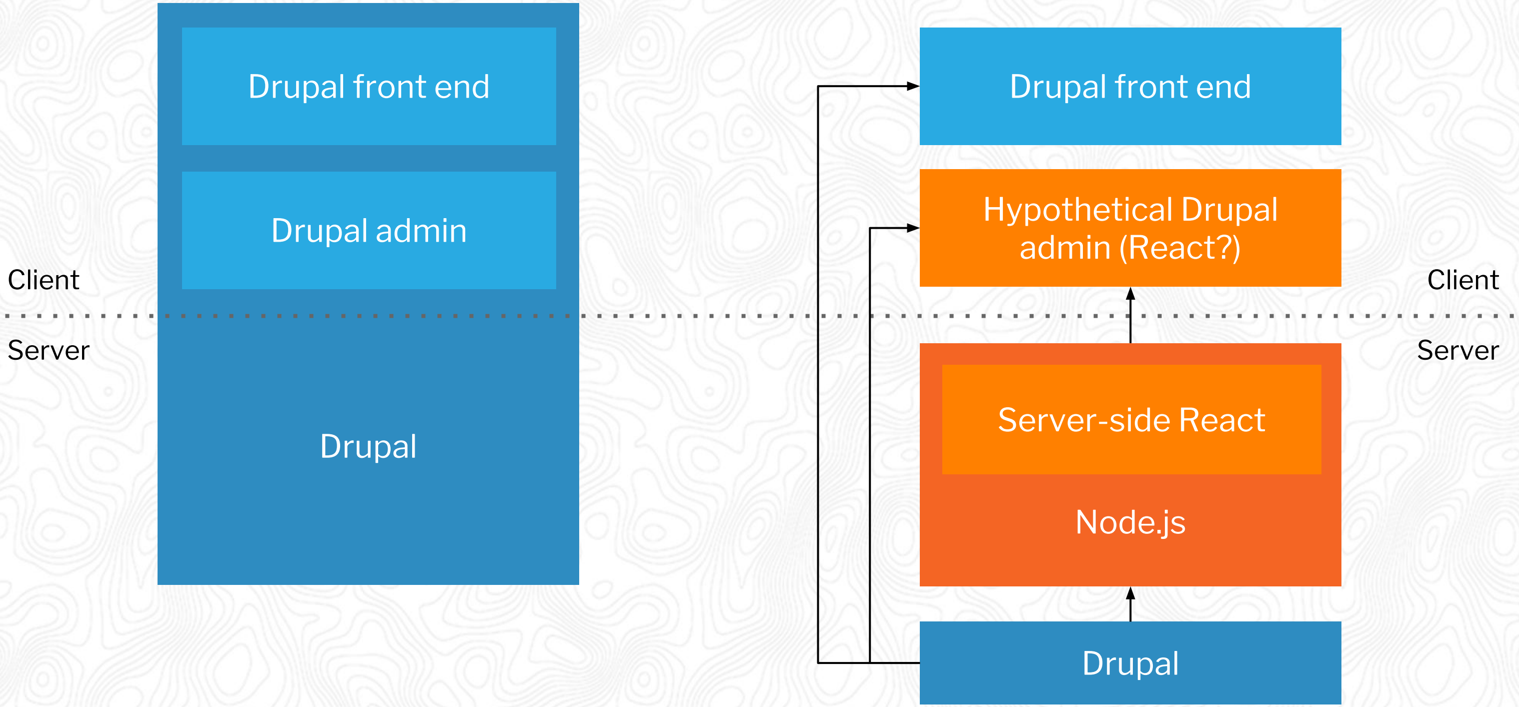
Traditional WordPress vs. WordPress Calypso



WordPress Calypso has no contextualized administration

Calypso considerations

- Calypso made a conscious decision not to tackle the problem of no contextualized administration on WordPress front ends, as features like in-place editing, etc. have been less of a focus.
- A similar editorial interface for Drupal would have the same set of issues.

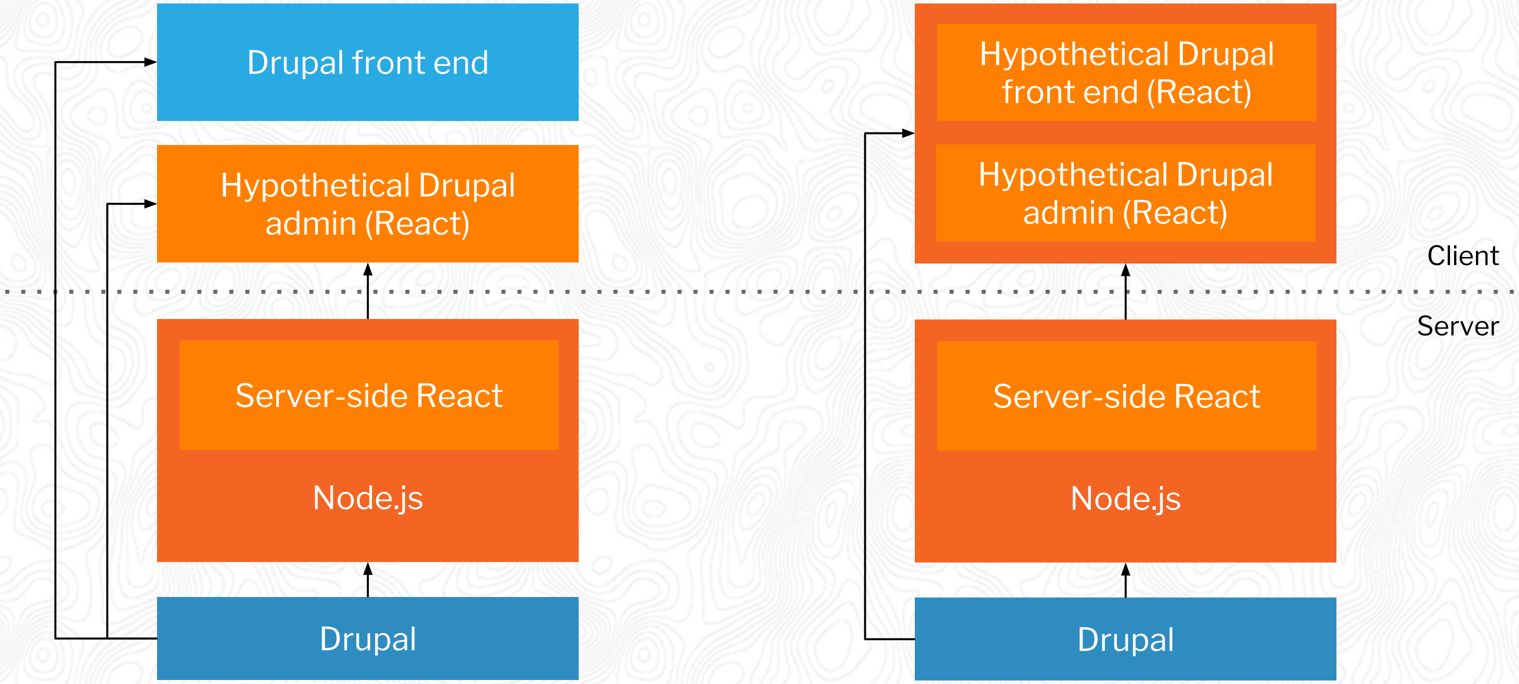


Theoretical Drupal admin

Can you contextually administer fully decoupled front ends?

Edit the fully decoupled front end

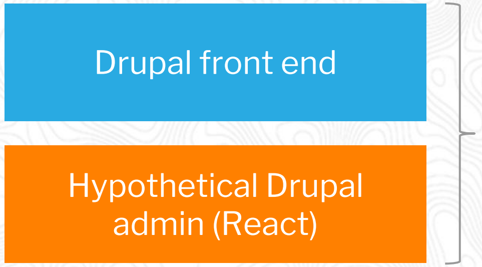
- One option is to make some tools that are native to Drupal's public-facing front end available as part of an entirely decoupled front end.
- This would require us to include Node.js as a dependency for Drupal — a LAMP back end providing APIs and a Node.js-driven front end providing SSR and a contextually administrable front end.
- In other words: a complete front-end rewrite.


Drupal front end
Hypothetical Drupal admin (React)
Server-side React
Node.js
Drupal
Hypothetical Drupal front end (React)
Hypothetical Drupal admin (React)
Client
Server
Server-side React
Node.js
Drupal

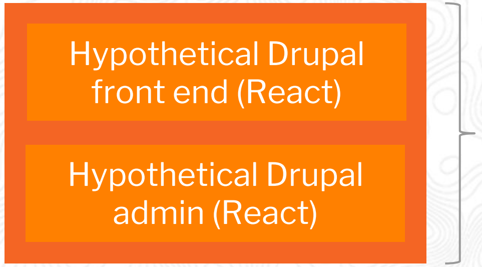
Theoretical contextualized Drupal front end

Drupal's contextual admin history

- Just as there are concerns about forcing authenticated users to download a JavaScript framework when solely viewing Drupal pages (rather than editing them) due to contextual administration …
- … there were concerns about including jQuery and Backbone on the same pages to provide for in-place editing and the toolbar.



When the **Drupal front end and Drupal admin are divergent**, contextual administration is much more difficult.


Hypothetical Drupal front end (React)
Hypothetical Drupal admin (React)

When the **Drupal front end and Drupal admin are one and the same**, contextual administration can take advantage of shared tools.

Contextualized administration is easiest on a shared front end

Divergence is dangerous

A few hypothetical scenarios:

- Imagine building an in-place editing feature in the same toolset and developer workflow as you had built the content editing tool in.
- Imagine building a layout manager feature in the same toolset and developer workflow as you had built the more comprehensive layout manager in.

It's clear that a rewrite in JavaScript
would be a monumental effort

Approximating contextual admin in divergent front ends is possible, but are the available solutions adequate?

5

The mythical "Edit" button
Prismic and Simpla

A quick anecdote …

"Where is my
in-place editing?
contextual links?
toolbar?"

Headless CMS hazards

- Headless CMSes like Contentful and CloudCMS pride themselves on refined and beautiful editorial interfaces which are still fundamentally series of forms.
- But, like WordPress Calypso, none of these interfaces is available in the form of contextual administration.

prismic.io

simpla

Prismic and **Simpla** are the among the first to try it

Prismic: Injected edit button

- Prismic allows JavaScript developers to include a script that provides a deep-linked "Edit" button back to the Prismic administrative back end on any Prismic-provided content.
- These client-side "in-website edit buttons" allow authenticated editors to navigate back to the Prismic back end.

Prismic: Injected edit button

```
<script>
  window.prismic = {
    endpoint: 'https://<your-repository>.prismic.io/api'
  };
</script>
<script src="//code.jquery.com/jquery-2.1.1.min.js"></script>
<script type="text/javascript"
src="//static.cdn.prismic.io/prismic.min.js"></script>
```

```
<article data-wio-id="{insert document id}">
  (...)
</article>
```

Prismic: Contextualized preview

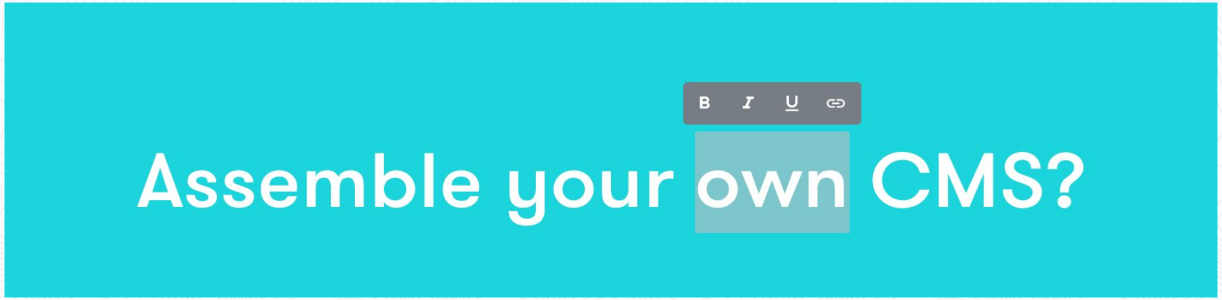
- Prismic allows editors to preview by providing a series of steps (involving some infrastructure) for both editors and developers:
    - Including a client-side prismic.io toolbar JS file
    - Creating a preview API endpoint
    - Adding a dependency for cookies
    - Adding a distinct route for previews

Simpla.io

- **Simpla.io** touts in-place editing, contextual formatting tools, content modeling, and an API in JSON that developers can consume.
- Simpla.io advertises itself as a "replacement for the CMS" and is built in Polymer and Web Components to enable separation of concerns between contextual administration and the public-facing front end.

Simpla.io

This is the future of the CMS


Assemble your own CMS?

Can Drupal stack up with these on contextual editing?

6

Decoupled layout
The case of RESTful Panels

Decoupled layout

- **Decoupled layout** involves the ability to manage layouts for consumption by decoupled front-end applications.
- This can either be a layout manager with no transparency on the public-facing front end or a layout manager solely on the administrative interface.

RESTful Panels

- **RESTful Panels** is an approach that exports Panels configuration as consumable JSON data structures.
- However, it interpolates the desired content into a data structure that mirrors the layout's construction.
- drupal.org/project/restful_panels

More control over desired content by developers (pure data structures)

Less control over layout management and components therein by editors

Better developer experience, worse editorial experience

Less control over desired content by developers (chunks of markup)

More control over layout management and components therein by editors

Better editorial experience, worse developer experience

Decoupled layout can be brittle

Giving JS devs chunks of markup resurfaces the flaws of progressive decoupling and "black boxes"

Better separation of concerns

- Providing both editorially administered layout and raw data structures as separate concerns in a single request might be ideal.
- JSON API could make this work by providing layout as a related entity alongside a raw data structure indistinguishable from a typical content request.

7

Epilogue:
Decoupled content strategy

Food for thought

What does this mean? It means potentially making some difficult decisions:

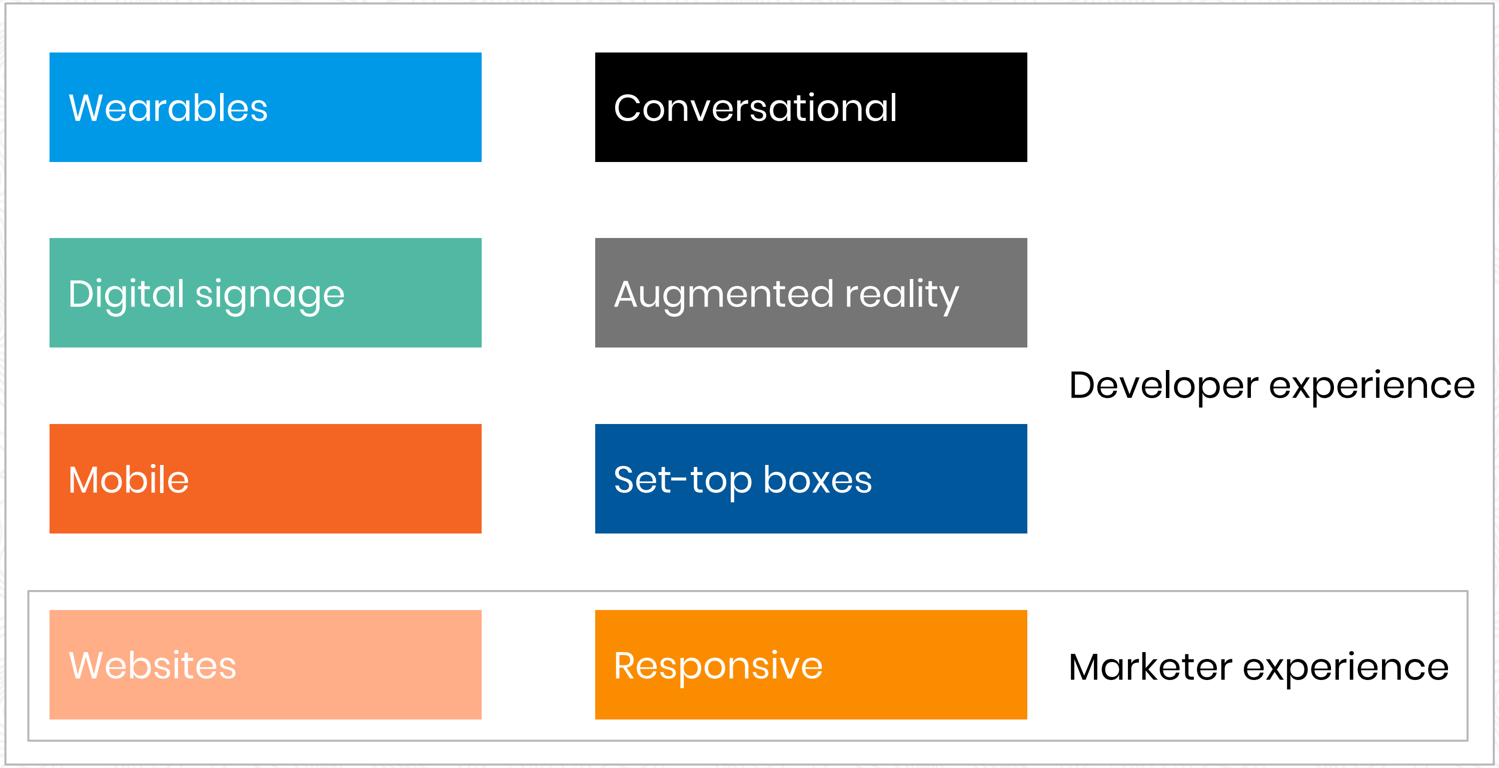
- Maybe it means prestiging the editor and marketer over the developer
- Maybe it means prestiging the developer over the editor and marketer
- Maybe it means attempting to retain the status quo

Prestige the editor and marketer

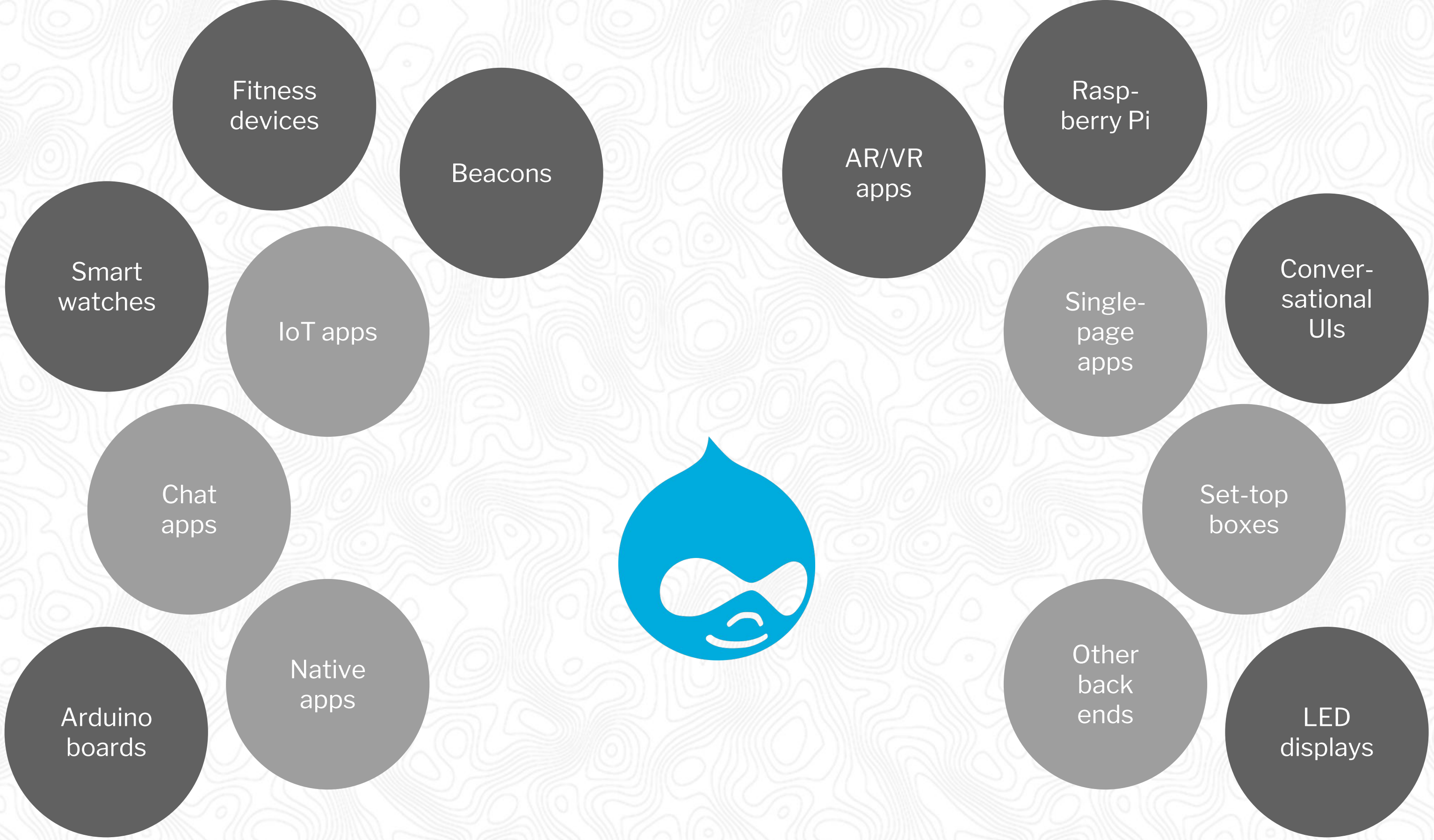
- Maybe it means adopting a JavaScript framework like React for a decoupled front end to enable the marketer — and to focus on only web
- Maybe it means emulating or otherwise approximating other devices in the context of a surrounding outside-in Drupal user interface

Prestige the developer

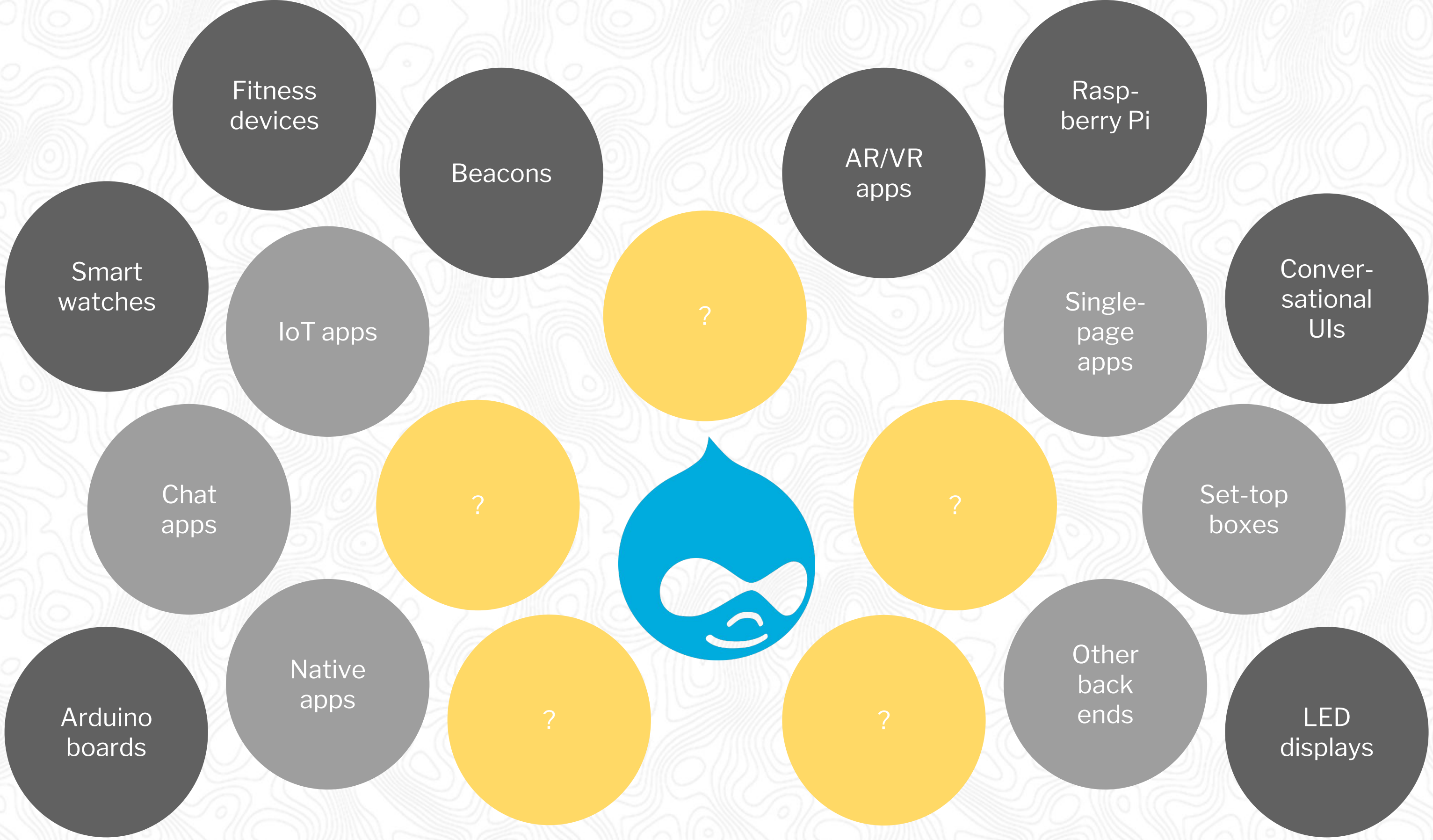
- Maybe it means adopting an agnostic approach and providing components that require the developer to finish the job
- Maybe it means becoming solely an API-first back end and letting developers do the heavy lifting from the front



Drupal's new incongruity



We have new channels every day.

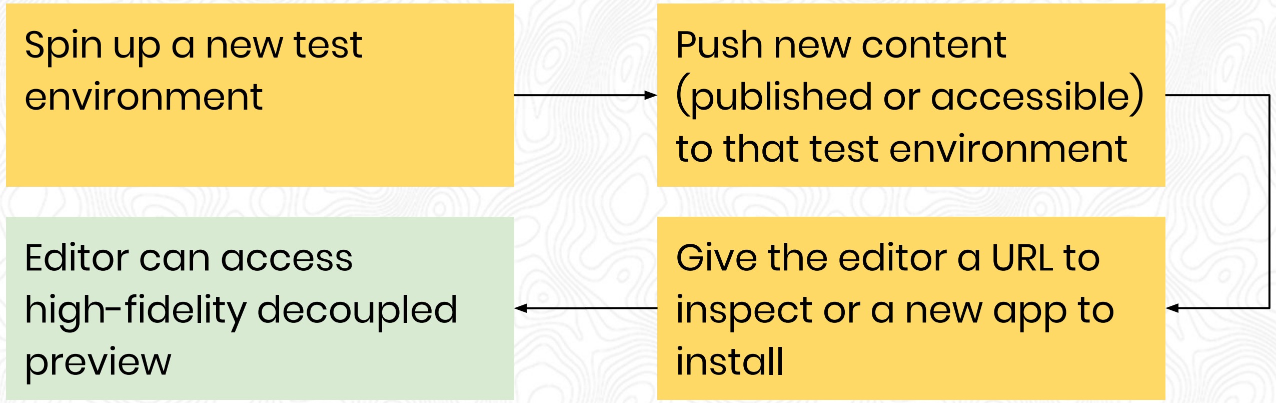


And ones we've never heard of yet.

Web is increasingly
only one facet of editorial concerns

Editorial preview

If an editor wants a high-fidelity preview of content on their single-page application or native mobile application, developers are required, for now.


Spin up a new test environment
Push new content (published or accessible) to that test environment
Give the editor a URL to inspect or a new app to install
Editor can access high-fidelity decoupled preview

Is contextual admin dead?

- Perhaps contextual administration and faithful preview shouldn't be a concern of CMSes; perhaps it should be a platform or infrastructural consideration.
- Fewer editors are using in-place editing and similarly contextualized features.
- But we still ultimately need **seamless** preview for editors and publishers without the aid of a developer.

If contextual admin is dead, decoupled content strategy is the answer

Channel diversity
(differentiated content across channels)

Channel agnosticism
(single piece of content for all channels)

Channel diversity vs. channel agnosticism

Decoupled content strategy

- In this omnichannel age, maybe we need to tell editors to be channel-agnostic with how they write content and manage it visually.
- Maybe it means we need to focus on assembly of just websites and encourage a more decoupled content strategy for everything else.

From decoupled content
to decoupled content strategy

From visual control of everything
to everything visual is uncontrollable

This is the great test that will dictate **the next decade of Drupal**

A multifaceted Drupal is a more future-proof Drupal

Thanks to you, Drupal's story has **only just begun**

8

Open discussion

Join us for contribution sprint

- **Mentored Core Sprint**
  Fri, 29 Sep — 09:00–18:00 — Stolz 2

- **First-time Sprinter Workshop**
  Fri, 29 Sep — 09:00–12:00 — Lehar 1, Lehar 2

- **General Sprint**
  Fri, 29 Sep — 09:00–18:00 — Mall

#drupalsprints

What did you think?

- **Evaluate this session**
  events.drupal.org/vienna2017/sessions/decoupled-site-building-drupals-next-challenge

- **Take the survey!**
  surveymonkey.com/r/drupalconvienna

Vielen Dank! • Thank you!

Preston So has been a web developer and designer since 2001, a creative professional since 2004, and a Drupal developer since 2007. As Director of Research and Innovation at Acquia, Preston leads new open-source and research initiatives and helms Acquia Labs, the innovation lab featured in international press.

Previously, Preston contributed to the Spark initiative in the Drupal project, co-founded the Southern Colorado Drupal User Group (est. 2008), and operated an award-winning freelance web and print design studio. Preston has presented keynotes at conferences on three continents in multiple languages.

preston.so@acquia.com • @prestonso • #decoupledsitebuilding