DrupalCamp New Jersey 2016

Decoupled Drupal and the Front End

Preston So

January 30, 2016

Welcome!

- **Preston So** (@prestonso) has designed and developed websites since 2001 and built them in Drupal since 2007. He is Development Manager of Acquia Labs at Acquia and co-founder of the Southern Colorado User Group.

  drupal.org/u/prestonso
  preston.so@acquia.com
  pso@post.harvard.edu

What we'll cover

1. What is decoupled Drupal?
2. Motivating decoupled Drupal
3. Getting started with decoupled
4. Integrating with clients
5. The future of the front end

What is decoupled Drupal?

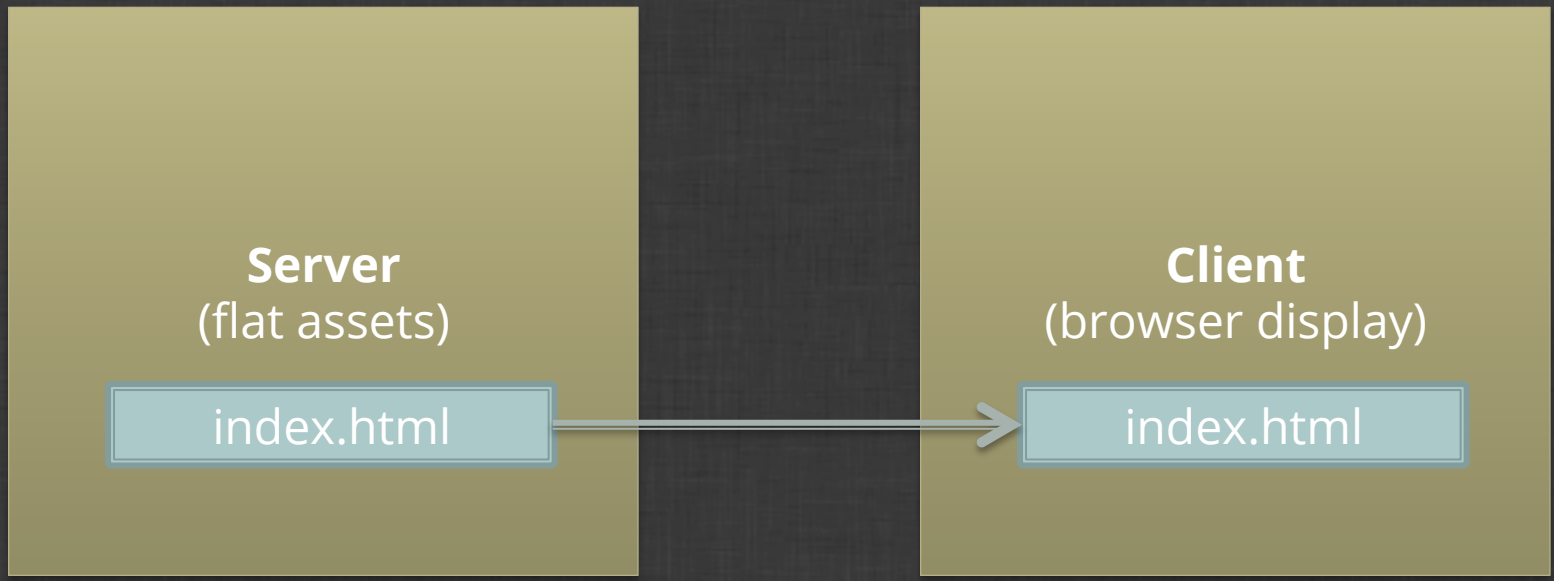
1

The history of content management

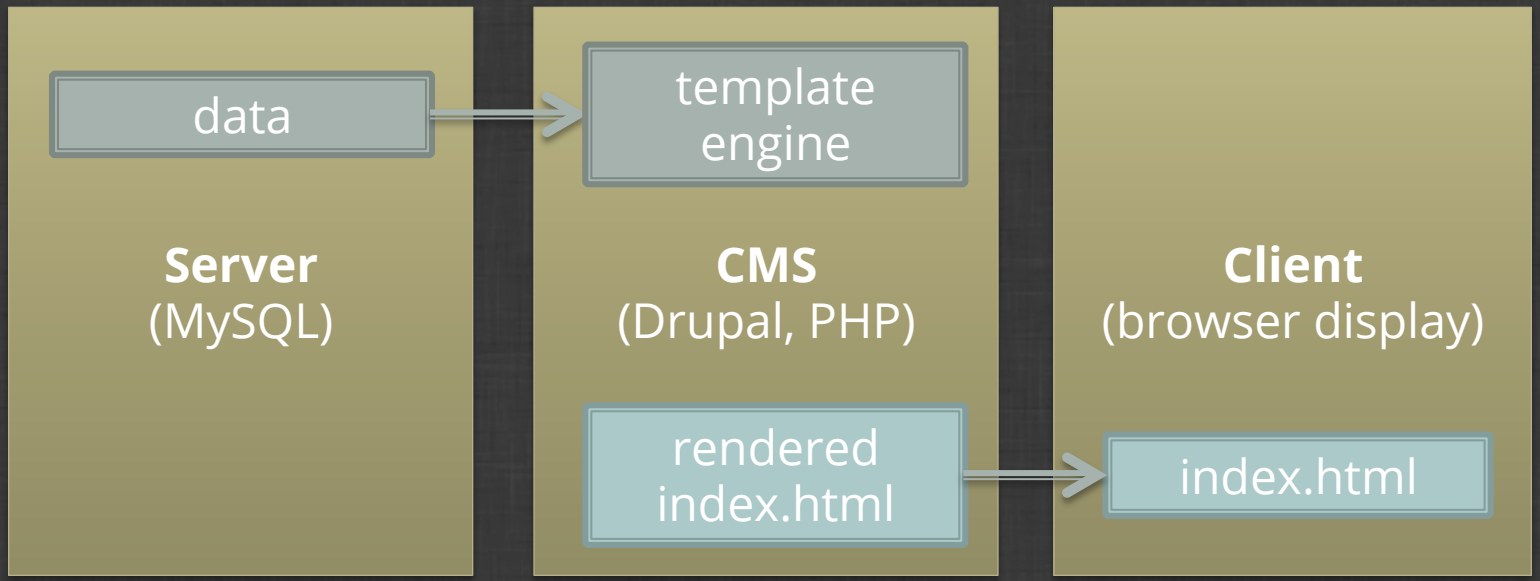
Microservices and API-first methodology

The decoupled front end

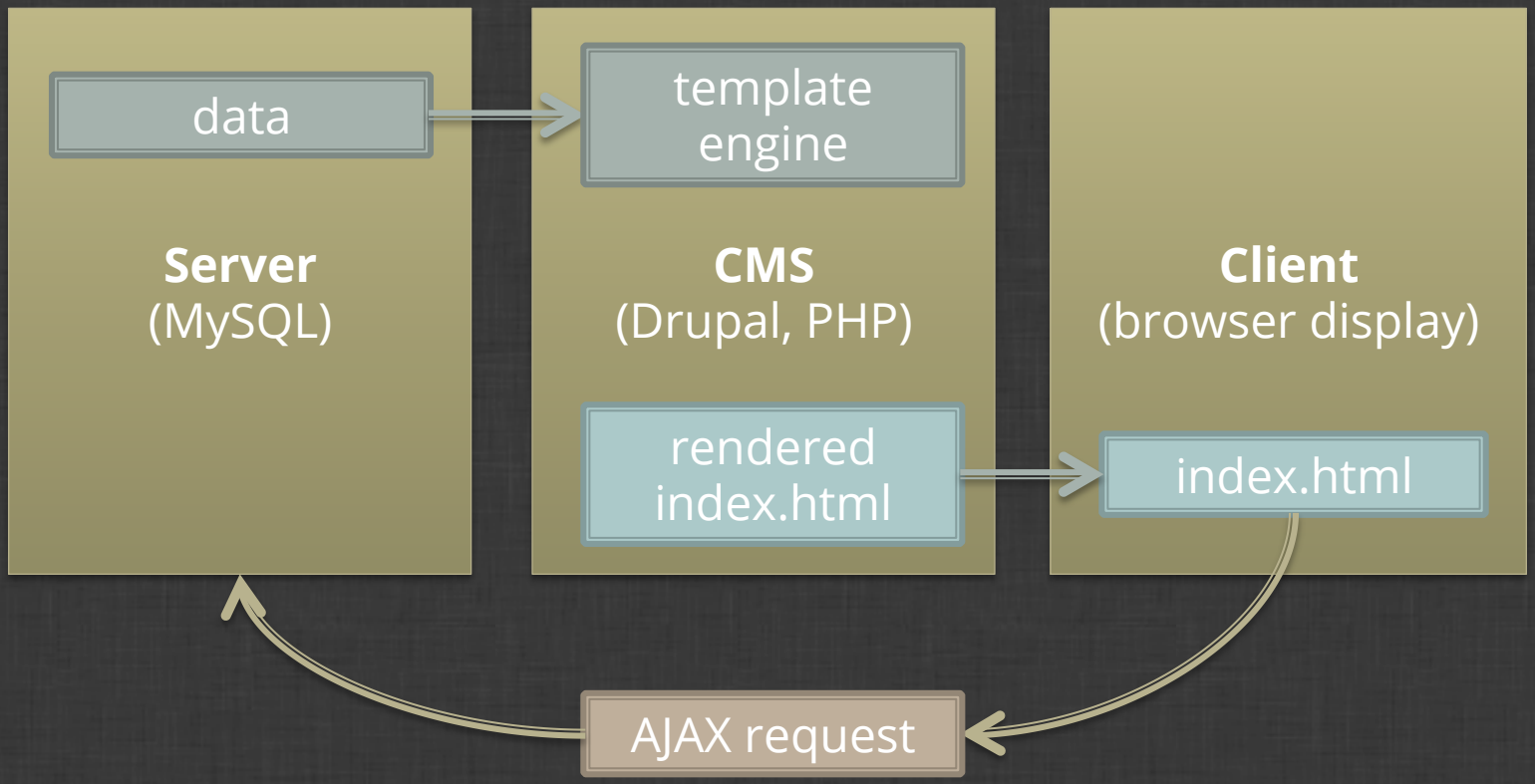
Web 1.0


Server
(flat assets)
index.html
Client
(browser display)
index.html

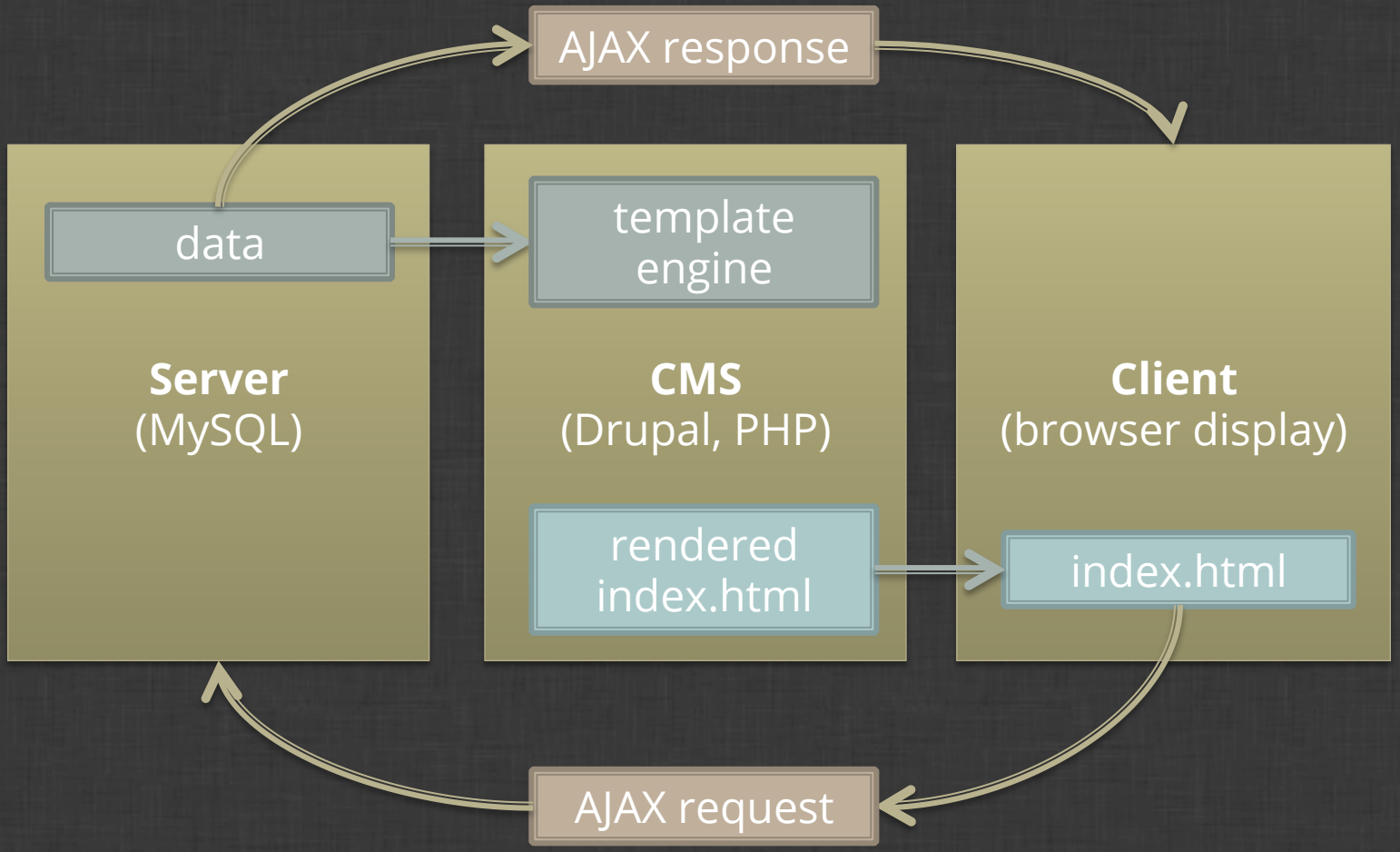
Web 2.0


data
template engine
Server
(MySQL)
CMS
(Drupal, PHP)
Client
(browser display)
rendered index.html
index.html

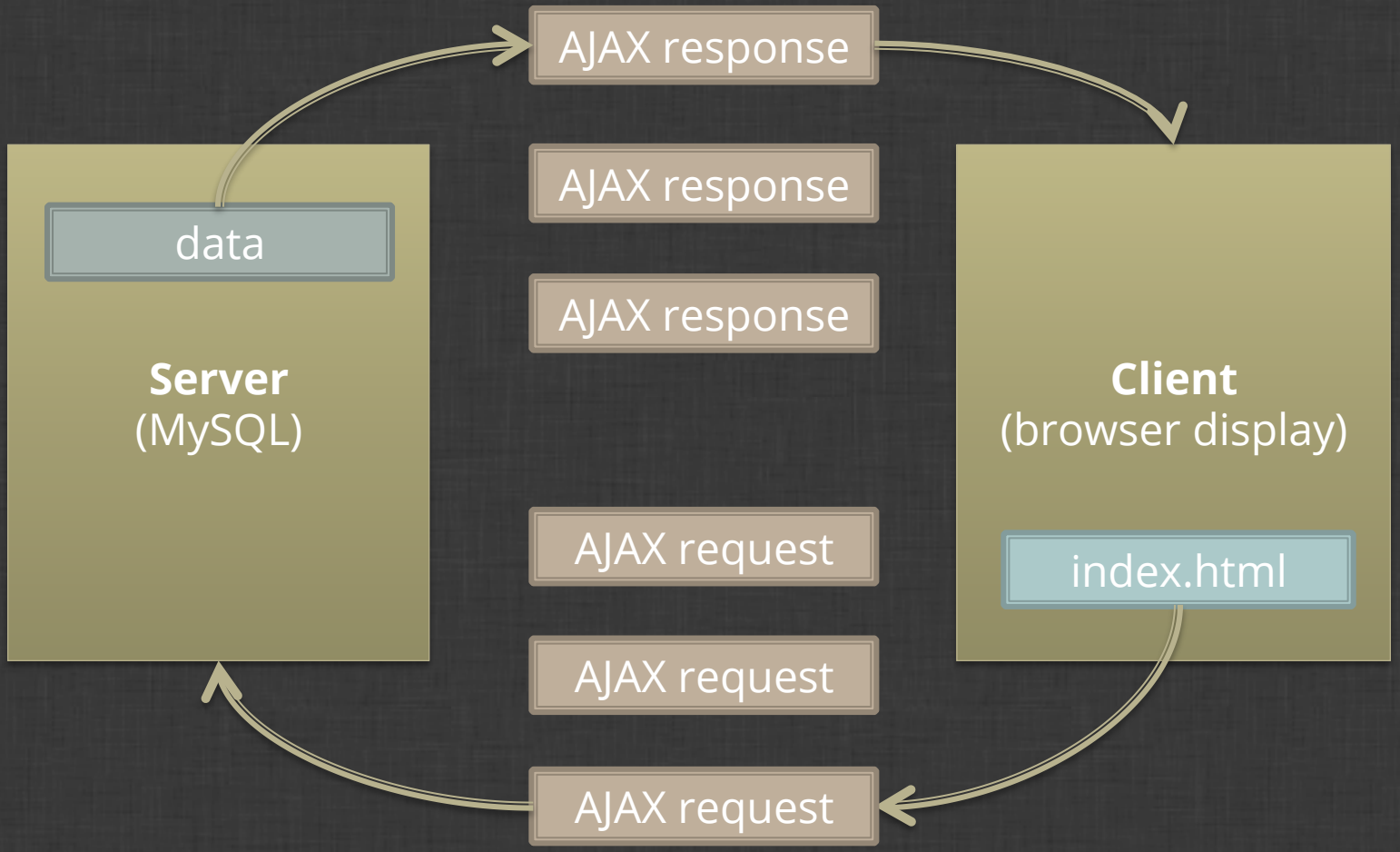
Web 2.0


data
template engine
Server
(MySQL)
CMS
(Drupal, PHP)
Client
(browser display)
rendered index.html
index.html
AJAX request

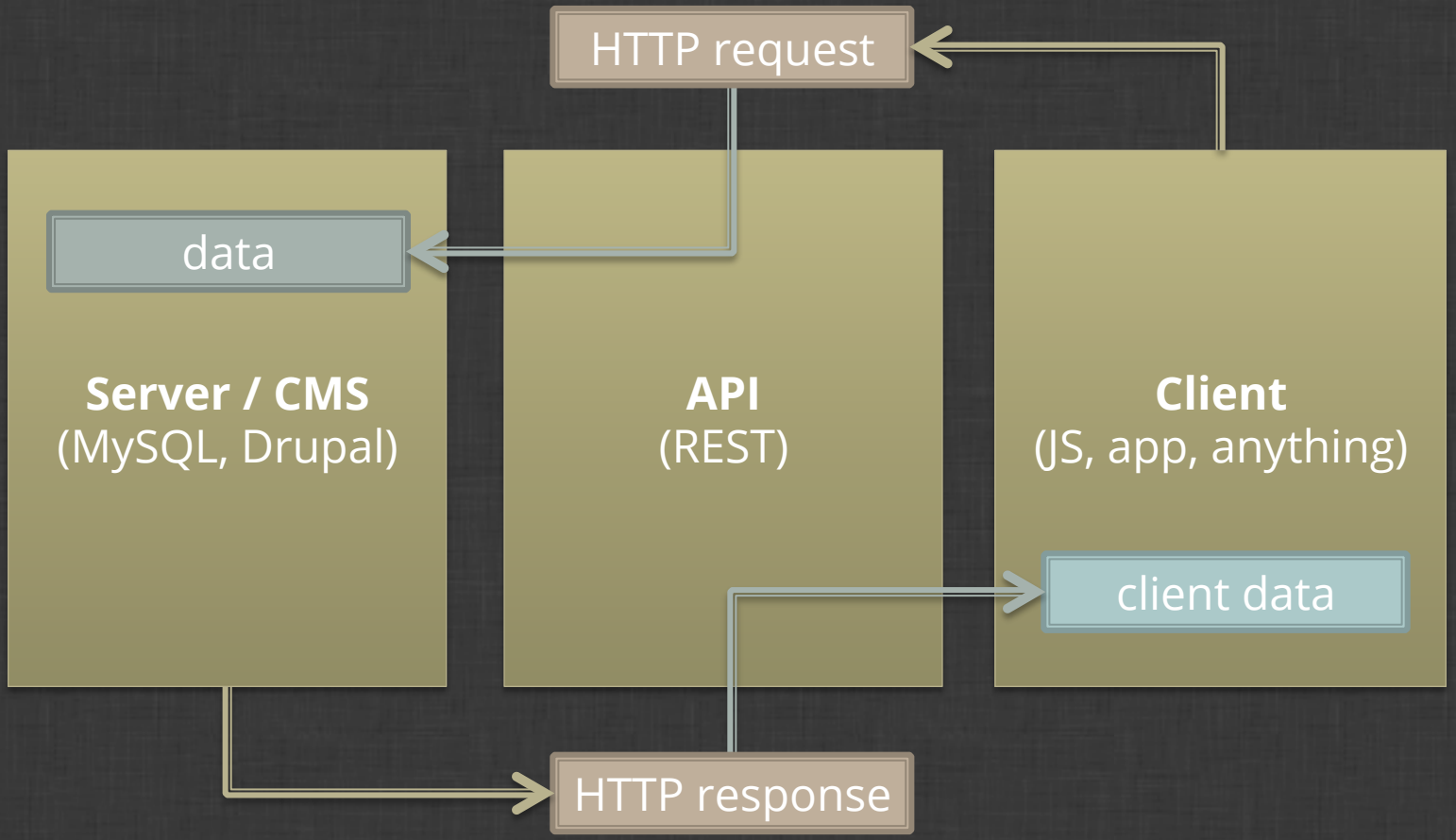
Web 2.0


AJAX response
data
template engine
Server
(MySQL)
CMS
(Drupal, PHP)
Client
(browser display)
rendered index.html
index.html
AJAX request

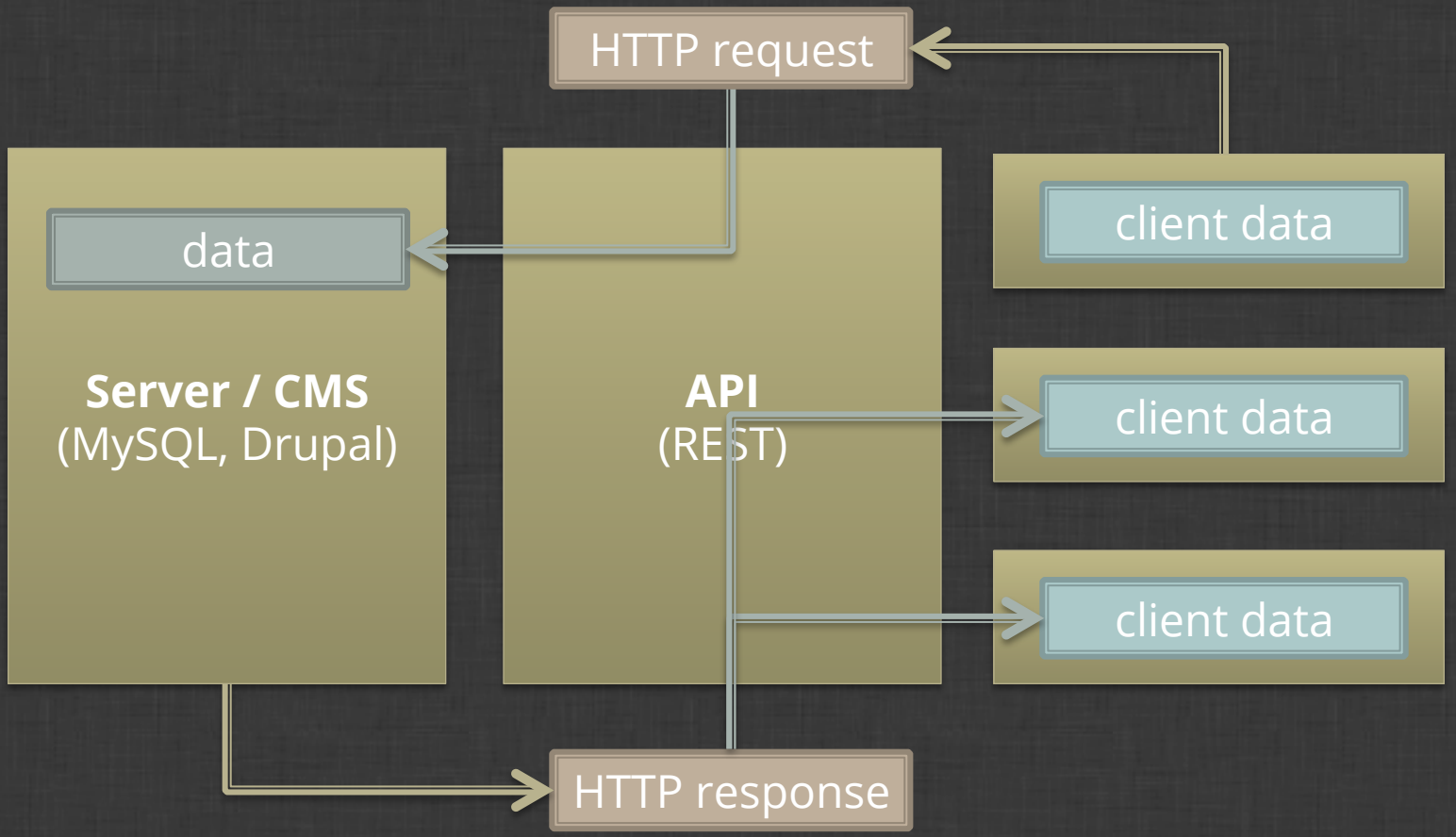
Web 2.x


AJAX response
AJAX response
AJAX response
data
Server
(MySQL)
Client
(browser display)
index.html
AJAX request
AJAX request
AJAX request

“Web 3.0”


HTTP request
data
Server / CMS
(MySQL, Drupal)
API
(REST)
Client
(JS, app, anything)
client data
HTTP response

“Web 3.0”


HTTP request
data
client data
Server / CMS
(MySQL, Drupal)
API
(REST)
client data
client data
HTTP response

JSON

- Package data as JSON or XML instead.

```
[{
    "nid": [{
        "value": "1"
    }],
    "uuid": [{
        "value": "0b65effd-c1fd-426a-b708-8fe43d3d66b1"
    }],
    "vid": [{
        "value": "1"
    }],
    "type": [{
        "target_id": "article"
    }],
```

JSON

- Package data as JSON or XML instead.

```
"langcode": [{
    "value": "en"
}],
"title": [{
    "value": "NEDCamp 2015"
}],
"uid": [{
    "target_id": "1"
}],
"status": [{
    "value": "1"
}],
```

The history of content management

- Content management systems have historically been monolithic.
- *Server-side decoupling* involved the separation of structure and presentation in content delivery.
- Today, content management systems are becoming increasingly decoupled on the client side.

Monolithic content management

Full-featured CMS
(Drupal 6)

Monolithic content management

Front end
(theme layer)

Full-featured CMS
(Drupal 6)

Monolithic content management

- The boundary between the theme layer and functional layer is “highly permeable” (Josh Koenig, Amitai Burstein).
- “Drupalisms leak to the front end”

Decoupled content management

Front end
(theme layer)

Full-featured CMS
(Drupal 6)

Front end
(templates, CSS)

Web editing tool
(Create.js)

Web framework
(Drupal)

Content repository
(PHPCR)

Microservices

- Independent, encapsulated functional components
- Do one thing, and do it well

Client-side framework with CRUD

Front end
(theme layer)

Full-featured CMS
(Drupal 6)

Front end
(templates, CSS)

Web editing tool
(Create.js)

Web framework
(Drupal)

Content repository
(PHPCR)

Front end
(Angular.js)

Back end
(Drupal)

API-first methodology

- Front end and back end interact via REST API
- Design API first (within Drupal), then write your client-side app
- Laravel (Symfony), Flatiron, Sails.js (Node.js)

Multiple clients

Client-side framework
Angular app

Native mobile application
iOS app

Other clients
Internet of things
Other Drupals

Back end
(single Drupal installation)

Headless or decoupled?

- “Headless software is software capable of working on a device without a graphical user interface.”
- … Drush?

Motivating decoupled Drupal

2

Separation of concerns

Write once, publish everywhere

Implications on projects and teams

Motivating decoupled Drupal

- Separation of concerns
- Structure vs. presentation and model, view, controller (MVC)
- Write once, publish everywhere (content syndication)

Problems of decoupled Drupal

- Much of Drupal's robustness is lost
- Cross-origin requests and security
- Authentication and passwords
- Form validation
- Search engine optimization

Problems of decoupled Drupal

- Content workflow and management
- Layout management
- Multilingual and localization
- Accessibility
- User experience

Decoupled Drupal and teams

- Teams will decouple as well, diversifying to work at two different velocities
- Generalists will disappear in favor of specialists, jeopardizing breadth of knowledge
- Multidisciplinary teams will have front-end developers without Drupal knowledge

Decoupled Drupal and projects

- Easier front-end redesigns
- Offline-first applications (PouchDB, Hood.ie)
- Further granularization of page elements

When should you decouple Drupal?

- A site powering one or more other sites
- A site powering one or more applications
- A site powering multiple things
- ~~Standalone applications~~
- ~~Standalone websites~~

Getting started with decoupled Drupal

3

Decoupling Drupal 6 and 7

Decoupling Drupal 8

Decoupled Drupal 6 and 7

- Off-the-shelf Drupal 7 has no opinionated REST layer
- Services module (*demo*)
  - Built-in REST and XML-RPC interfaces
  - Exposes nodes, users, taxonomies, and other core data at custom endpoints
  - Services Entity extends Services to work with all entity types

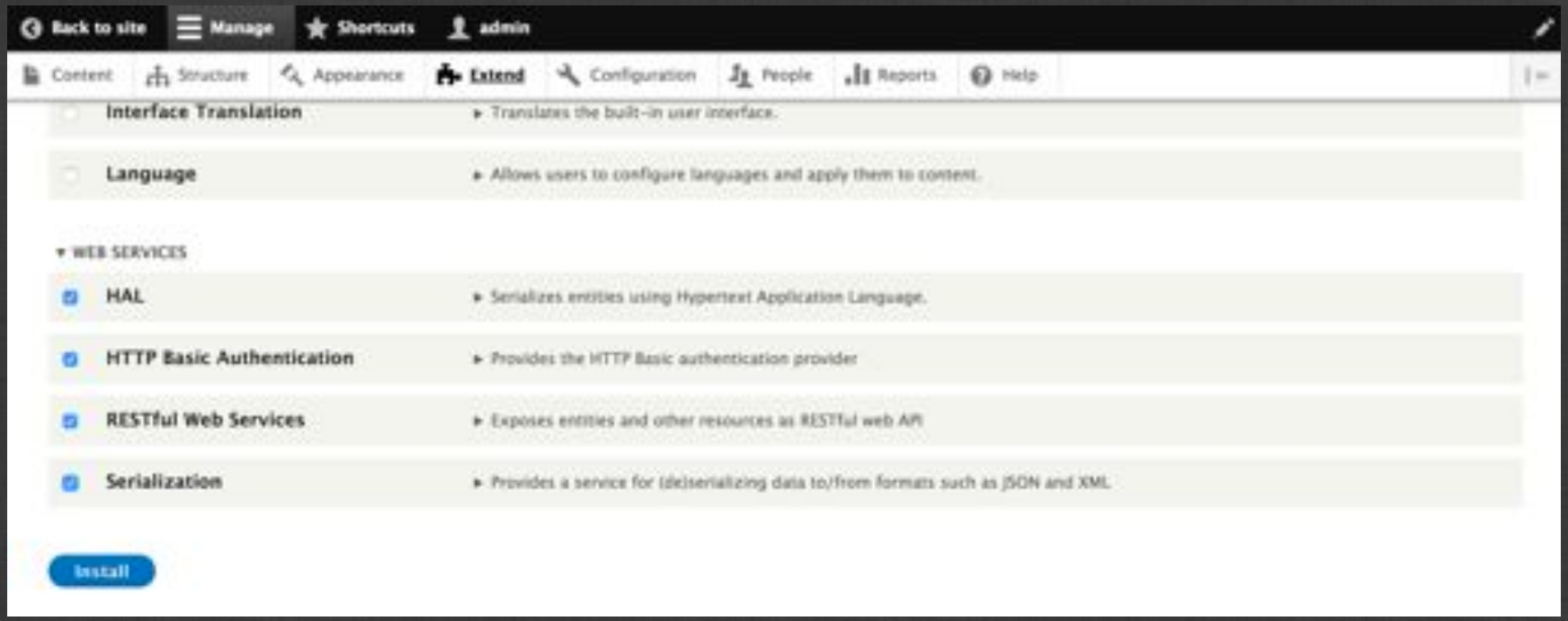
Decoupled Drupal 6 and 7

- restWS exposes any Drupal entity on its existing path based on headers.
- Restful is entity-centric but allows developers to define what data ought to be sent in response to a request.

Decoupled Drupal 8

- WSCCI (Web Services and Context Core Initiative) incorporated Web Services into Drupal 8 core.
- Web Services (core) allows for all core entities to be exposed as JSON+HAL; Views natively supports “REST export” as a new display type.

Decoupled Drupal 8



Decoupled Drupal 8

- RELAXed Web Services (contrib) extends the core REST API to include revisions and file attachments.
- RELAXed's mission aligns with movement in content staging and offline-first applications.

Integrating with clients

4

Native mobile applications

Single-page applications

Client-side frameworks

- Angular.js relies on directives declared in templates to provision MVC relationships and provide two-way data binding
- Ember.js is highly opinionated, less flexible, and has a roadmap to eventually integrate with the fluctuating Web Components standard
- React.js uses a Virtual DOM, JSX as syntactic sugar, and robust state system

Isomorphic JavaScript

- An isomorphic framework can be used both on the client side and server side
- Node.js: Integration of front-end and back-end elements; testing framework
- MEAN stack: MongoDB, Express.js, Angular.js, Node.js

Single-page applications

- Web Services has a built-in CORS Domains UI
- Angular.js: Retrieve JSON using `$rootScope` and `$http`
- Ember.js: Retrieve JSON using `$.getJSON` on a new route

Native mobile applications

- Services Views, Views Datasource (D6 and D7)
- Titanium (write JS, compile Java or Objective-C)
- Drupal iOS SDK
- DrupalCloud (Android)

The future of the front end

5

The Drupal theme layer

Web Components and the DOM

Discussion

Is theming dead?

- Morten's "Angry Themer" efforts
- Does this solve divitis and over-Drupalistic markup?
- Accessibility and user experience (ARIA, etc.)
- Is the Drupal theme layer better off coupled?

Pseudo-decoupling

- Exposing front-end logic from Drupal such as layout configuration to the client-side app
- Duplication of functionality
- Circular dependencies

Progressive decoupling

- Decouple only components of the page
- Maintain Drupal functionality such as site building tools, security, accessibility, and layout management
- “The future of decoupled Drupal” by Dries
  buytaert.net/the-future-of-decoupled-drupal

Is Drupal at risk?

- “We can make Drupal a weapon of choice for the emerging front-end development community. We can become good partners with them in open source. It’ll take us into new markets, it’ll take us into new use cases.”

—Josh Koenig

Food for thought

- How do we make front-end developers excited about using Drupal as a back end? (GraphQL)
- “What back end are you using?”
- Content and site REST API for Drupal
- A better way of provisioning APIs in Drupal

Food for thought

- What if you could manipulate Drupal content and site configuration as easily as we can with Drush?
- Drupal admin UI as a single-page app
- Big plus: Optimistic feedback

Food for thought

- What is the future of Drupal against the backdrop of decoupled Drupal?
- What is the future of the Drupal front-end community against the backdrop of decoupled Drupal?
- What is the future of decoupled Drupal?

Thank you!

- **Preston So** (@prestonso) has designed and developed websites since 2001 and built them in Drupal since 2007. He is Development Manager of Acquia Labs at Acquia and co-founder of the Southern Colorado User Group.

drupal.org/u/prestonso
preston.so@acquia.com
pso@post.harvard.edu