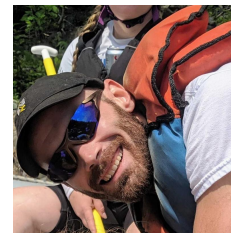


oEmbed


The ecosystem, challenges, and bridging the gaps

Introductions



- I'm Luke.
- Applications Developer with Penn State Outreach Marketing since 2017
 - <https://www.worldcampus.psu.edu>
 - A pile of other sites, products, and microservices
- Active contributor and community member on drupal.org
 - **luke.leber** on drupal.org
 - **lleber** on slack

Agenda

1. An overview of oEmbed itself
 2. A brief history of oEmbed within Drupal
 3. The state of oEmbed in modern versions of Drupal
 4. An honest review of the “off the shelf” functionality of oEmbed in Drupal through the lens of a marketing developer
 5. Introducing: oEmbed Lazyload: a unique, modern, and scalable module created to bridge gaps in the ecosystem.
- 

oEmbed

A 10,000 foot overview

- What is it?
 - What problems does it solve?
 - How does it work?
 - A note on security.
-

oEmbed – What is it?

“oEmbed is an **open** format designed to allow embedding **content** from a website into another page” – [wikipedia](#)


Key concepts:

1. The **open** format is defined by an openly published specification
2. The **content** is intentionally ambiguous – this can be **any** type of content.

Conveniently available in one page at <https://oembed.com>



oEmbed – What problems does it solve?

- There are **tons** of content providers; oEmbed provides structure.
 - Much easier to implement known content provider integrations
 - Allows for no-code integrations with new and unknown providers
 - Less effort for *providers* to make their content widely available
- Operates over common, simple, and portable tooling (REST + JSON / XML)
- The intent to display content and underlying implementation are abstracted.
 - Content consumers don't have to worry about backwards-compatibility problems
 - Content providers have more flexibility in platform evolution and features
 -  <https://www.drupal.org/project/drupal/issues/3085545>

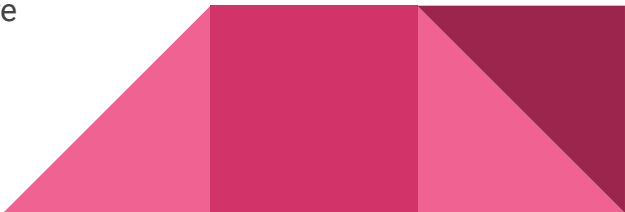
oEmbed – Actors

The oEmbed mechanism includes two actors:

1. Consumers

- a. Parties that wish to display embedded content
- b. Websites, digital billboards, etc...
- c. Analogous to the concept of a “client” in typical client-server architecture

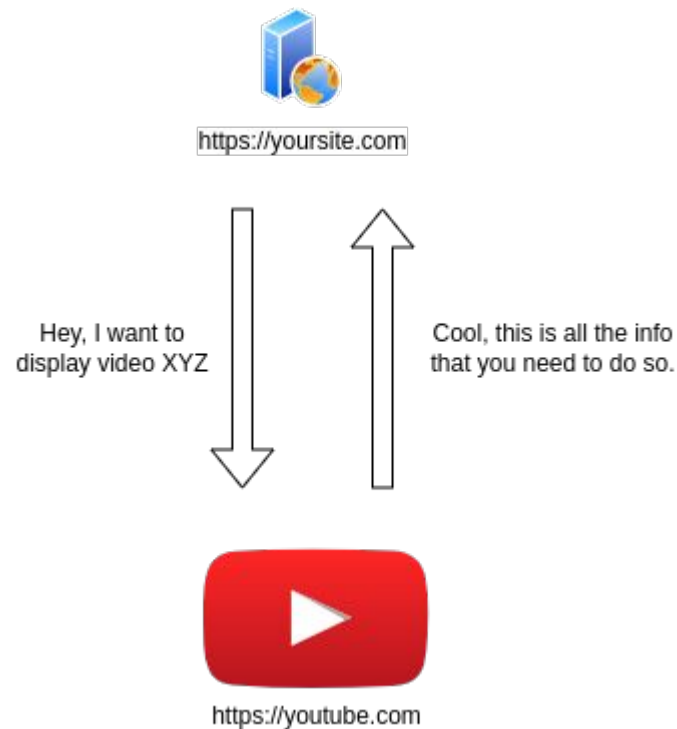
2. Providers

- a. Parties that have content to share
 - b. Image / video services such as YouTube, Vimeo, Flickr, etc...
 - c. Analogous to the concept of a “server” in typical client-server architecture
- 

oEmbed – How does it work?

In a typical oEmbed exchange:

1. A *consumer* requests a resource from a *provider* via HTTP GET
2. The *provider* sends back all of the information needed to display the resource.
 - a. Requested content plus title, thumbnail, authoring information, cacheability metadata, dimensions, etc...
 - b. The spec is open-ended!



oEmbed - Discovery Mechanisms

There are primarily two ways to discover oEmbed provider information:

1. Via `<link>` attributes in the head sections of oEmbed providers' webpages
 - a. `<link rel="alternate" type="application/json+oembed" href="https://..." title="Example" />`
 - b. The oEmbed documentation recommends utilizing this discovery mechanism
2. Via centrally controlled provider repositories
 - a. Typically manifested as static JSON files hosted by reputable parties
 - b. <https://oembed.com/providers.json>
 - c. This is the approach that Drupal core utilizes
 - d. Custom repositories can be used



oEmbed – On security...

- *Third party content is dangerous. **Period.***
- oEmbed requires *some* degree of trust with third party systems.
 - Data exfiltration, DOS, and XSS are risks – just to name a few.
 - Typically requires server-side blocking HTTP calls.
 - The data returned by the provider must be considered arbitrary in nature.
- Several mechanisms can be implemented to mitigate risk
 - Exfiltration can be prevented by serving third party content in an iframe with a different domain
 - A CSP may also be used to harden applications
 - Allow-listing “trusted” third parties is *not* foolproof




oEmbed + Drupal


A brief history (in contrib space)

- The “media” module (for Drupal 7)
 - The “media_entity” module
 - The “video_embed_field” module
-

oEmbed + Drupal – Media (drupal 7)

- Believe it or not, the media module for Drupal 7 is still alive and kicking! 
- Provides oEmbed support via the [media_oembed](#) module
 - Provides off the shelf support for a number of providers
 - Allows site builders to configure others
- Still a fine solution for Drupal 7 sites (you aren't still on Drupal 7, are you? 😜)


Releases

7.x-4.5  released 17 January 2023

Works with Drupal: 7.x
PHP 8.2 compatibility, improve commerce coupon module support

Development version: [7.x-4.x-dev](#) updated 31 Oct 2023 at 00:22 UTC

GitLab CI:  [view all pipelines](#)

7.x-3.5  released 17 January 2023

Works with Drupal: 7.x
PHP 8.2 compatibility and improve support for the commerce coupon module

Development version: [7.x-3.x-dev](#) updated 31 Oct 2023 at 00:21 UTC

GitLab CI:  [view all pipelines](#)

7.x-2.30  released 17 January 2023

Works with Drupal: 7.x
✓ Recommended by the project's maintainer.
PHP 8.2 compatibility, improve support for the commerce coupon module

Development version: [7.x-2.x-dev](#) updated 31 Oct 2023 at 00:18 UTC

GitLab CI:  [view all pipelines](#)


7.x-1.10  released 15 August 2023

Works with Drupal: 7.x
Add support for PHP 8.0 and PHP 8.1 and PHP 8.2

Development version: [7.x-1.x-dev](#) updated 31 Oct 2023 at 00:20 UTC

GitLab CI:  [view all pipelines](#)

oEmbed + Drupal – Media Entity (still contrib space!)

- The contributed [Media Entity module](#) was originally created for Drupal 8
 - oEmbed was originally provided via a large ecosystem of *media_entity_** modules
 - Some oEmbed support was later superseded by the [Video Embed Field](#) module and a large ecosystem of *video_embed_** modules
 - Media Entity was largely absorbed into core
 - Migration paths from these ecosystem to core media exist
- 

oEmbed + Drupal

Modern times

- Media in core
 - oEmbed functionality in core
 - Media integration
 - Provider discovery
 - Resource fetching
 - Displaying oEmbed content (safely?!)
 - Extending oEmbed
 - A look at the contrib ecosystem
-

Modern times: History of media in core

- Media API support ships in 8.4
 - API's existed, but nothing was really usable
 - Hidden from the modules list; no UI
- Reusable media ships in 8.5
 - Had a UI – yay
 - Rather spartan UX compared to alternatives at the time
- oEmbed + experimental media library support ships in 8.6
- Major UX improvements to media library ships in 8.7
 - Finally on par with competitors – yay
- CKEditor4 support for media ships in 8.8 – *Extra yay!*
- Fast forward to Drupal 9.5 – CKEditor5 support arrives



Modern times: oEmbed Media Integration

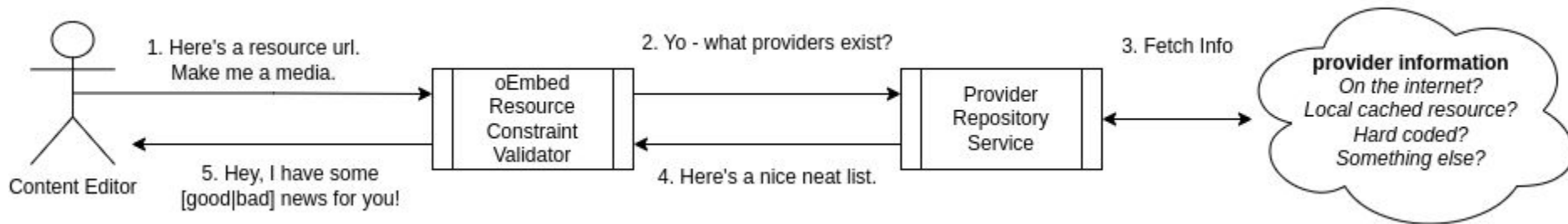
- Site builder + content editor UX similar to documents, images, audio, etc.
 - Completely fieldable including customizable widgets and formatters
 - Media source field mapping works as designed
- Plugs into core media via a dedicated MediaSource plugin
 - Metadata attributes align with the oembed specification
 - Thumbnails are downloaded and stored locally
 - Support is limited to YouTube and Vimeo off the shelf
- The oEmbed media source plugin is presently utilizing the Deriver pattern
 - Additional oEmbed providers can be added via *hook_media_source_info_alter*
 - Can also be extended by contributed or custom code, *but use caution!*



Modern times: oEmbed Provider Discovery

The core provider discovery strategy is...

- Based on an external repository lookup
- Implemented as a regular Drupal service
 - Consumes an oEmbed repository and generates *Provider* objects
 - Used by a validation constraint (💪) to ensure predictable behavior
 - Also used by field formatters, controllers, etc...



Modern times: Provider Discovery (default implementation)

- Repository URL is set on media module install under `media.settings:oembed_providers_url`

```
media.settings:
  type: config object
  label: 'Media settings'
  mapping:
    oembed_providers_url :
      type: uri
      label: 'The URL of the oEmbed providers database in JSON
format'
```

- Set by default to <https://oembed.com/providers.json> (~300 providers therein, but only 2 are used)
- There is no UI for this setting, nor is it able to be updated by hook
- Inherently reliant on a third party resource
 - May block server-side execution for the default http client timeout - *choose your provider carefully!*
 - Data is cached in the KV store, refreshed every 7 days, but stale information may be used as a last resort
 - ***This is a single point of failure for all drupal sites using core media by default!*** 🤖



Error message

An error occurred while trying to retrieve the oEmbed provider database.

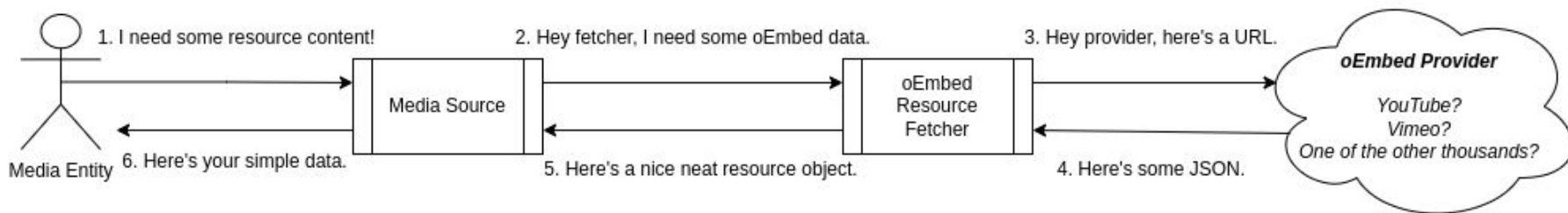


This could happen on every drupal site without cached provider data should oembed.com abruptly shut down or fail.

Modern times: oEmbed Resource Fetching

Resource fetching is...

- Implemented as a regular Drupal service that consumes a URL and generates a *Resource* object
- Resources are used by the oEmbed media source plugin to fulfill the common metadata attribute API that all media types must support



Modern times: Resource Fetching (default implementation)

The default resource fetcher...

- Can make blocking server-side HTTP calls
 - Times out after an unconfigurable 5 seconds elapse
 - Does not cache failed attempts – slow / flaky providers can cause DOS!
- Allows *request URLs* to be altered via *hook_oembed_resource_alter* 🌟
- Core does not offer any hooks or events to alter provider responses
 - Only supports data points explicitly defined by the oEmbed spec
 - Ignores any vendor extensions to the oEmbed spec

Modern times: Displaying oEmbed content

oEmbed content is displayed through a dedicated Field Formatter.

- Allows site builders to configure a max width and height for the resource
- Customized format settings per view mode
- Drupal 10.1 added a control for native lazy loading 🤖
- All oEmbed content is served through an iframe
 - **No security protections are afforded through this by default !!!**
 - Site administrators must set up additional infrastructure for effective risk mitigation
 - Failure to set up the infra results in persistent security warnings in the Drupal dashboard
 - ? - How many in this room have *actually* set this precaution up?

Modern times: Extending oEmbed

Fortunately:

- There are a lot of open issues for improving oEmbed in core
- There are contrib solutions to many functionality gaps
- Almost every part of the oEmbed API is driven by services!
 - Tweak a service *just a little bit* by decorating it.
 - Completely swap out a service!

Unfortunately:

- Many of the open core issues are > 5 years old
- There's overlapping functionality in the contrib ecosystem
- Some contrib modules require core patching to function 🙄

Modern times: Contrib Space

Just a quick shout out to some great contrib modules!

- [oEmbed Providers](#) from Chris Burge
 - Elegantly taps into the oEmbed media source driver pattern to support arbitrary oEmbed providers with no code required! 🎉
 - Can be configured to eliminate blocking server-side I/O for provider discovery. 🛡️
- [Blazy](#) - the venerable go-to for performance optimization over the ages
 - Prior to Drupal 10.1, Blazy was a de-facto solution for lazy loading **almost all** below-the-fold assets
 - *Highly configurable - even works where native **loading** attributes won't*
- ...and [many more](#) - far too many to talk about today!

An Honest Review

A critical look at oEmbed in core – *off the shelf*

- The context of this review
 - What worked well?
 - What didn't work well?
-

An honest review: Context

- The entity in context is Penn State World Campus
 - The main marketing site, <https://www.worldcampus.psu.edu>
 - 400,000 unique visitors per month
 - Hundreds of active marketing campaigns at any given moment, driving web traffic 24/7
 - My primary responsibilities:
 - Maintain an understanding of the business goals of the World Campus marketing strategy
 - Lead custom CMS development to achieve said business goals
 - Collaborate with design, UX, and accessibility experts to implement highly accessible web experiences with exceptional functional and aesthetic quality.
 - Manage integrations between web strategy and various other systems including customer relationship management and analytics
- 

An honest review: What worked well?

1. **Extremely simple** site building / content editing experience
 - a. Videos are managed the same as images
 - b. The video platform doesn't matter – it's all the same
 - c. Consistent video selection UX for entity fields, views, WYSIWYG embeds, and more.
2. Accessible, out of the box
3. Flexible field formatting options
 - a. Able to provide options for full-width videos or limited width in varying contexts
4. Usage tracking (via [Entity Usage](#))
 - a. Removing all traces of a video from content on the website has *never been easier*.



An honest review: What didn't work?

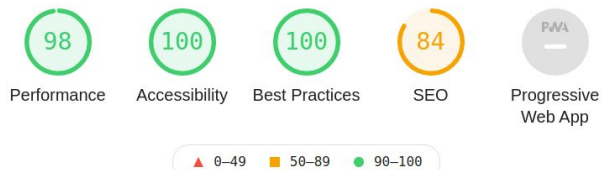
#1 – oEmbed content *can destroy* front-end performance

- An *exceptionally small* of users were found to actually interact with oEmbed videos
 - ~**0.17%** of users in 2023 interacted with video assets
 - Users that did interact with videos were **436%** more likely to convert.
- Existing lazy-load solutions cannot effectively help with above-the-fold assets
- Lead contributor to exorbitant page sizes for mobile users



An honest review: Web Vitals Delta (Bartik)

No oEmbed video



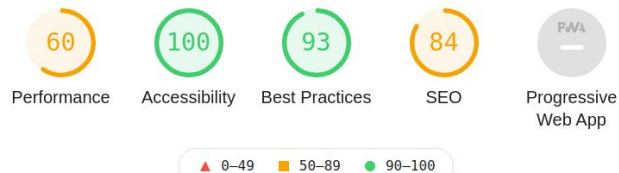
Performance

Metrics

● First Contentful Paint	1.5 s	● Time to Interactive	1.7 s
● Speed Index	1.5 s	● Total Blocking Time	10 ms
● Largest Contentful Paint	2.3 s	● Cumulative Layout Shift	0

Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. [See calculator.](#)

Single above-the-fold oEmbed video(YouTube)



Performance

Metrics

● First Contentful Paint	1.5 s	▲ Time to Interactive	10.5 s
■ Speed Index	4.2 s	▲ Total Blocking Time	1,670 ms
● Largest Contentful Paint	2.2 s	● Cumulative Layout Shift	0

An honest review: What didn't work?

#2 – YouTube Analytics

- The iframe that core wraps oEmbed content with does not load GA by default even when the rest of the site is configured to
- Organizations that use a completely different domain for the iframe also require additional GA work
- Off the shelf YouTube analytics aren't fully compatible, even after somehow injecting GA into the security iframe. 🤨

“Why is every video playing from /media/oembed???”

An honest review: What didn't work?

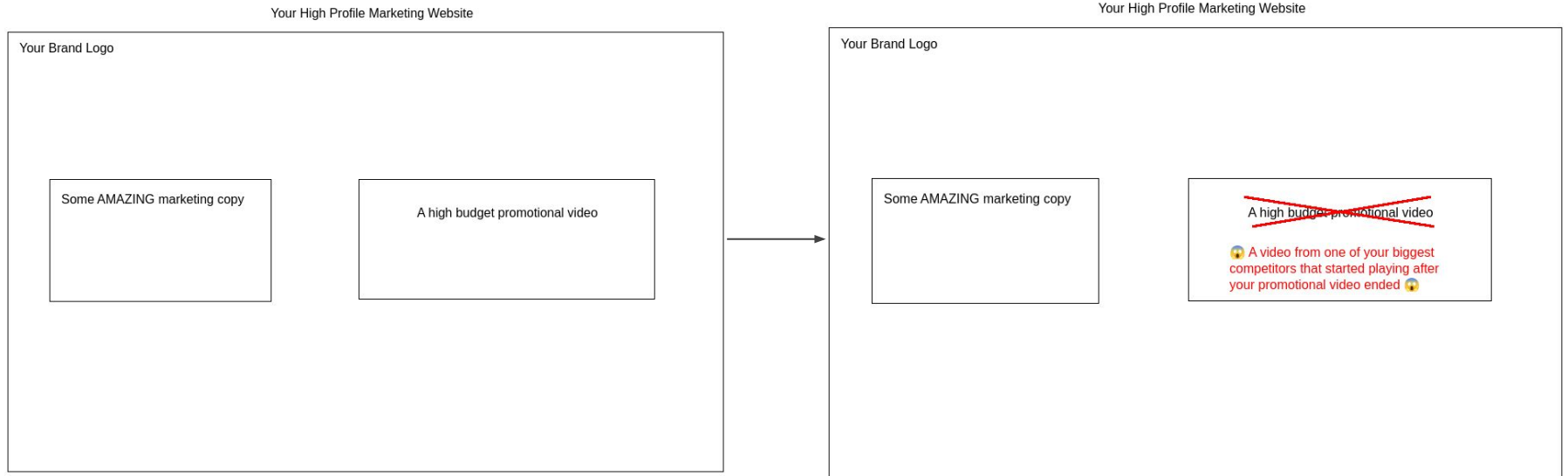
#3 – Generic oEmbed support simply didn't work well enough for marketing use

- The performance impact was far too severe to meet web vitals requirements
- There was an unfortunate logo placement / overlay banner positioning problem across the video inventory with the default vendor-specific settings.



- Removing YouTube branding from the player was desired
- ...and the ultimate marketing horror story deal-breaker

An honest review: What didn't work?



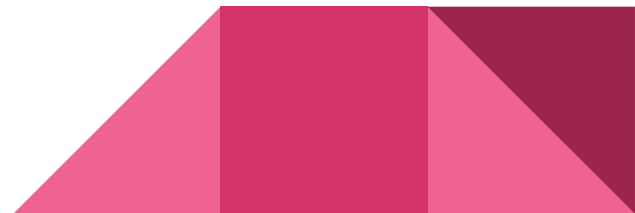
An honest review: What didn't work?

#4 – Thumbnail assets go stale over time

- Drupal downloads a local copy of oEmbed thumbnails on media creation
- There is no way for end-users to update thumbnail assets

#5 – Thumbnail assets were too low resolution for use in lazy-load placeholders

- Mostly downloaded at 480x360
- Quality issues were far more notable on retina displays



An honest review: What didn't work?








Could not retrieve the oEmbed resource

#6 – Inappropriate messaging can be visible to **end-users**.

- How many end-users would understand / care what a “oEmbed resource” is?
- Exception handling for oEmbed is ultimately managed by a flash message
- Some conditions that throw exceptions are **fairly typical**
 - When a YouTube video is made private without first auditing the website for usage
 - Random networking issues between the host and provider.
- <https://www.drupal.org/project/drupal/issues/2972846>

Review Summary

-  Easy for content editors
 -  Accessible, out of the box
 -  Hard to optimize web vitals
 -  Hard to do analytics
 -  Attention to detail
 - Incomplete thumbnail handling
 - Questionable error messages
 - Treating each provider equally lowers the quality of all providers
-

oEmbed Lazyload

A unique, extensible, and opinionated approach to bridging gaps in oEmbed

How to install this module?

Composer.

```
composer require drupal/oembed_lazyload
```



Enabling modules

Only pay for what you need

- oEmbed Lazyload

- oEmbed Lazyload YouTube

How to enable oEmbed Lazyload?

- Via drush: `drush en oembed_lazyload`
- Via the Drupal administration dashboard
 - Sign in as a user that has permission to enable modules
 - Click *Extend* in the administration toolbar
 - Find *oEmbed Lazyload* in the module list and check the corresponding input box
 - Click the *Install* button
 - Note - if any additional modules need to be installed, an additional confirmation step is required. After reviewing the additional dependencies, if everything is agreeable, click the *Continue* button
- Repeat for `oembed_lazyload_youtube` if desired





oEmbed Lazyload

What does oEmbed Lazyload do?

- Provides a new field formatter for oEmbed fields
 - Allows site builders to configure lazy loading strategies
 - Matches core functionality as closely as possible by default
 - Thumbnail assets will never go stale
 - Thumbnail assets may still be low quality
- Provides an extensible API for module developers
 - Allows third party settings to be added to the field formatter plugin type
 - Exposes a *ProviderEnhancer* plugin type



Field Formatter: Lazy load oEmbed video

Site builders can choose between two loading strategies:

1. Load third party assets when they enter the viewport
 - a. Optionally, provide a custom distance from the viewport to load
 - b. Pixels and percentages are supported
 - c. *Note - this predates the core "loading=lazy" feature and uses intersection observer*
2. Load third party assets when the user activates a play button
 - a. Effectively lazy loads above-the-fold third party assets 💪
 - b. *Note - some mobile users **may** have to tap twice to play videos, but this can be mitigated*



back to site Manage Shortcuts admin

content Structure Appearance Extend Configuration People Reports Help

manage news manage form display manage display manage permissions

Default Media library

Show row weights

Field	Label	Format
	- Hidden -	<p>Format settings: Lazy load oEmbed video</p> <p>Maximum width</p> <input type="text" value="550"/> pixels <p>Maximum height</p> <input type="text" value="390"/> pixels <p>Lazy loading strategy</p> <input type="text" value="Load third party assets when it enters the viewport"/> <p>How far away from the viewport should content begin to load?</p> <input type="text" value="100px"/> <p><small>This value must be defined as a number of pixels or percentage, IE. 20px or 4%.</small></p> <p>This strategy can be useful if users do not have to give explicit consent to load third party content.</p> <p>Update Cancel</p>

Lazy load oEmbed video field formatter settings

back to site Manage Shortcuts admin

content Structure Appearance Extend Configuration People Reports Help

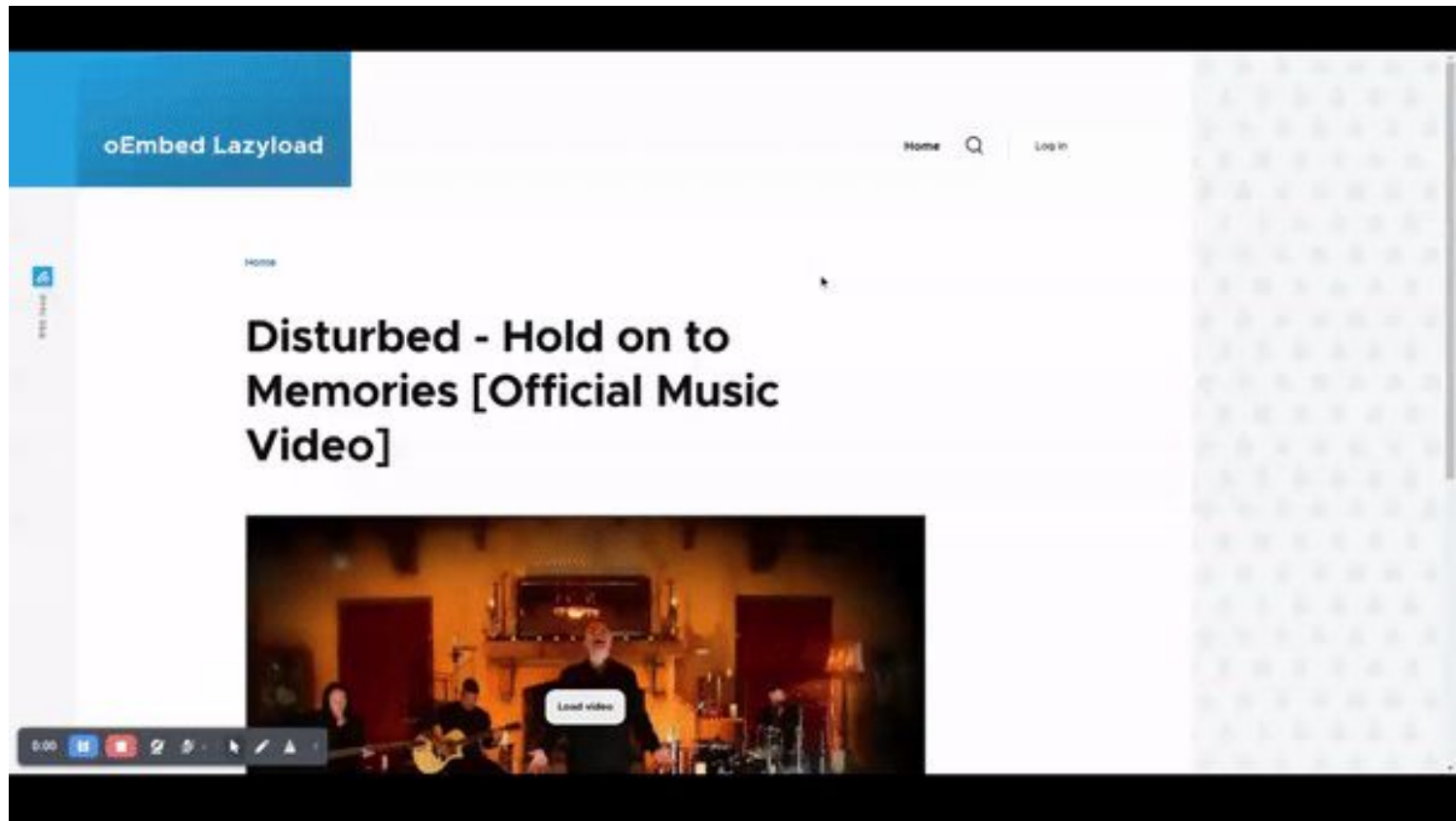
manage news manage form display manage display manage permissions

Default Media library

Show row weights

Field	Label	Format
	- Hidden -	<p>Format settings: Lazy load oEmbed video</p> <p>Maximum width</p> <input type="text" value="550"/> pixels <p>Maximum height</p> <input type="text" value="390"/> pixels <p>Lazy loading strategy</p> <input type="text" value="Load third party assets when the user activates a play button"/>
✚ * Video URL		<p>This strategy can be useful if users are required to give explicit consent to load third party content.</p> <p>Mobile users may have to click play twice if their browser does not allow auto-play (notably, Chrome + Safari).</p> <p>Update Cancel</p>

Lazy load oEmbed video field formatter settings



oEmbed Lazyload: Olivero end-user experience 😞



oEmbed Lazyload Youtube

What does oEmbed Lazyload Youtube do?

- Provides new field formatter settings for oEmbed fields
 - Allows site builders to configure YouTube specific API options
 - Matches core functionality by default
- Provides a buttery smooth off the shelf end-user experience
 - A set of highly opinionated placeholder templates and styles
 - Thumbnail assets will never go stale
 - End-users will likely be oblivious to the fact that the content is loading lazily



Field Formatter: Third Party (YouTube!) Settings

New third party settings will be found on all oEmbed Lazyload formatters

- Attempt to autoplay the video
 - Only recommended with “Load third party assets when the user activates a play button”
 - May not work on all mobile devices
- Hide YouTube branding on the player interface
 - This option was deprecated - no longer functions and will be removed in 3.0.x.
- Allow video to be controlled via the YouTube IFrame API
 - Required for certain YouTube analytics
 - Limiting the domain that can control the iframe is recommended
- Hide the video title and uploader before the video starts playing
 - This option was deprecated, but still seems to function on older accounts
- Only show related videos from the same channel as the current video
 - ***Perhaps the most important option in here!***

Lazy loading strategy

Load third party assets when the user activates a play button ▾

This strategy can be useful if users are required to give explicit consent to load third party content.

Mobile users may have to click play twice if their browser does not allow auto-play (notably, Chrome + Safari).

✚ Video URL

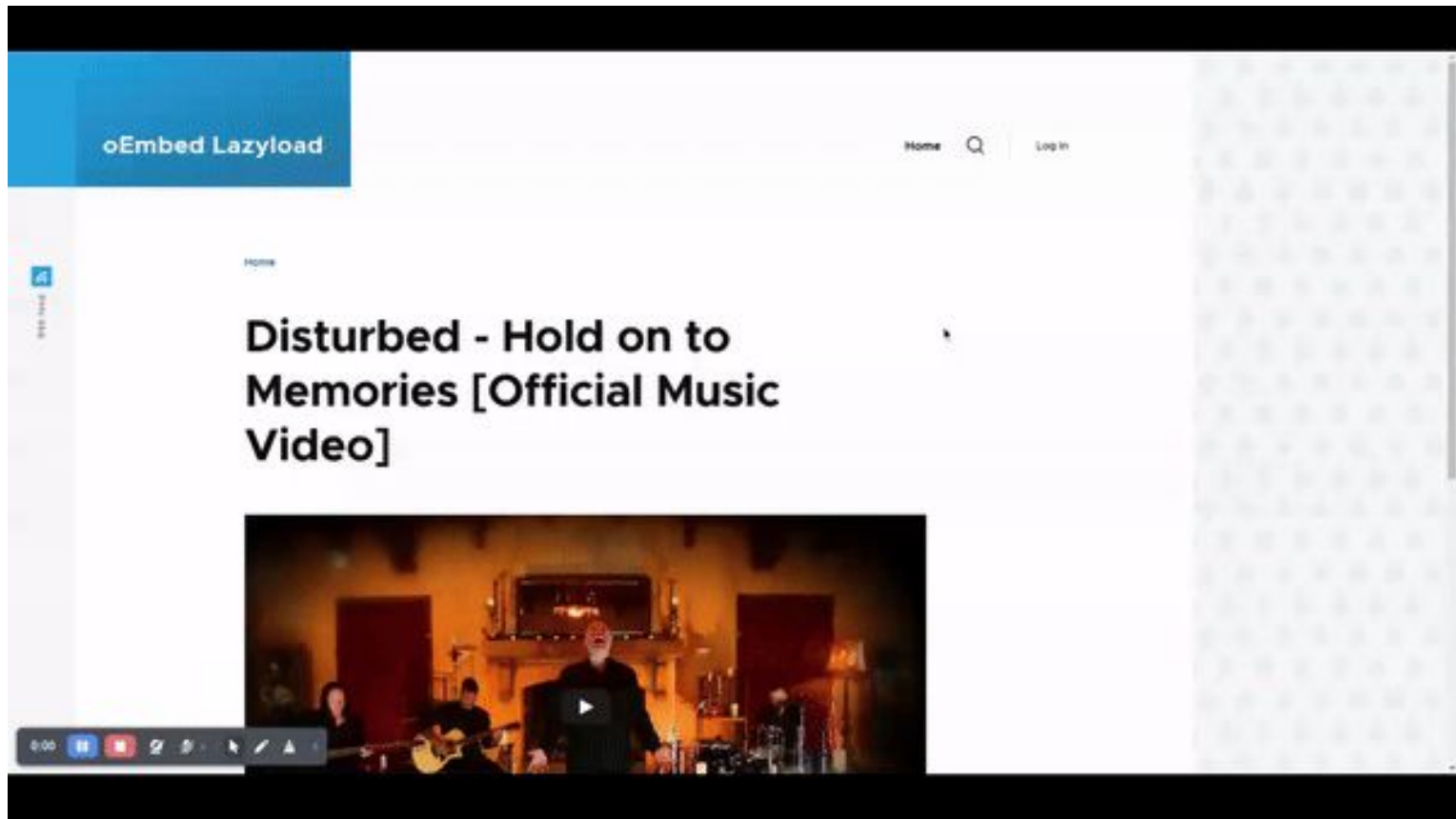
^ YouTube settings

- Attempt to auto-play the video
- Hide YouTube branding on player interface
- Allow video to be controlled via the YouTube IFrame API
- Only allow the oembed iframe domain to control the IFrame API (recommended)
- Hide the video title and uploader before the video starts playing
This option was [deprecated in 2018](#) and while it may work in some certain legacy configurations, it may stop working at any time! If this is necessary for branding reasons, prepare an alternative solution immediately.
- Only show related videos from the same channel as the current video

Update

Cancel

YouTube third party settings added to formatter



oEmbed Lazyload YouTube: Olivero end-user experience 🥰

Advanced Features

Some assembly required

Google Analytics Support

YouTube Autoplay Support



Google Analytics Support

- Google Tag Manager has an off-the-shelf [YouTube video trigger](#)
- Drupal wraps oEmbed content in an iframe
 - By default, this iframe will not contain any custom **<head>** elements
 - This iframe has a customizable twig template, *media-oembed-iframe.html.twig*
- Copy the template from the media module to your module or theme and inject the GTM container.

```
{#
/**
 * @file
 * Template to display an oEmbed resource in an iframe with a GTM container.
 *
 * @ingroup themeable
 */
#}
<!DOCTYPE html>
<html>
  <head>
    <css-placeholder token="{{ placeholder_token }}">
    <script>
      <!-- GTM Snippet Here -->
    </script>
  </head>
  <body style="margin: 0">
    {{ media|raw }}
  </body>
</html>
```

YouTube Autoplay Support


- The *onclick* strategy can lead to a subpar user experience on mobile devices, even with the autoplay option selected in the UI
- The YouTube iframe API can be used to enhance user experience
 - Only works on Android devices
 - Requires additional javascript in the *media-oembed-iframe* twig template
- Copy the template from the media module to your module or theme and inject the appropriate javascript.

```
{#  
/**  
 * @file  
 * Template to display an oEmbed resource with extended autoplay.  
 * @ingroup themeable  
 */  
#}  
<!DOCTYPE html>  
<html>  
  <head>  
    <css-placeholder token="{{ placeholder_token }}">  
  
    {# Add the iframe API script. #}  
    <script src="https://www.youtube.com/iframe_api"></script>  
  </head>  
  <body style="margin: 0">  
    {{ media|raw }}  
    {# Attempt to play the video. #}  
    <script>  
      document.querySelector('iframe').id = 'player';  
      function onYouTubeIframeAPIReady() {  
        new YT.Player('player', {  
          events: {  
            'onReady': function(event) {  
              const player = event.target;  
              player.playVideo();  
            },  
          },  
        });  
      };  
    </script>  
</body>  
</html>
```



oEmbed Lazyload API

Provider Enhancers

- Provider enhancers are plugin types that allow third party code to safely operate on specific oEmbed provider resources
 - Allows for customized placeholder templates, markup, styles, and scripts
 - Allows altering oEmbed metadata
 - Allows altering oEmbed HTML
 - Plugin annotation takes an array of provider strings in the *providers* key
 - Provider Enhancers only fire on the providers specified in this annotation
 - Extensible alternative for an ever-growing if/elseif/elseif chain.
 - A *ProviderEnhancerBase* class exists to reduce boilerplate
 - A fallback enhancer exists for new / unknown providers
- 

Example: YouTube provider enhancer

The YouTube enhancer...

- Uses high resolution thumbnails
- Loads opinionated style and scripting libraries
- Applies vendor-specific settings to oEmbed requests

```
/**
 * An implementation that enhances YouTube content.
 *
 * @ProviderEnhancer(
 *   id = "youtube",
 *   label = "YouTube",
 *   providers = {
 *     "YouTube"
 *   }
 * )
 */
class YoutubeEnhancer extends ProviderEnhancerBase {
    // A bunch of YouTube stuff.
}
```

oEmbed Lazyload Bugs 🐛



Why can we never have our cake and eat it too? 🙄



CKEditor5 integration is *rough* and hard to support



Some web servers require special attention to query parameter sorting



The max-width and max-height intrinsic values are presently ignored

- The off-the-shelf display options are perhaps *too opinionated*
- All videos render at a 16:9 aspect ratio
- Inter-op with custom themes can require custom CSS
- Work is happening in the 2.1.x branch to improve this!

