BUILDING THE FASTEST DRUPAL OF THE GALAXY

Hello!

I AM MATEU AGUILÓ

I am a senior developer at Lullabot
You can find me at @e0ipso

Hi!

I AM PEDRO GONZÁLEZ

I am a sysadmin at sbit.io

You can find me at @NITEMAN_es

DISCLAIMER

These are our experiences and what we learned so far.

Forget Drupal < 8

Why?

Performance by design

What?

Drupal’s 1.001 caches

How?

Drupal's 1.001 caches

Drupal 8's cache system overview

Swappable cache backends

Cache strategy

SWAPPABLE CACHE BACKENDS

$$\left.\begin{array}{l}\text{Database}\\ \text{MemCache}\\ \text{Redis}\\ \text{etc.}\end{array}\right\}\ \text{CacheBackendInterface}$$

CACHE STRATEGY I: How to cache

- avoid doing the work at all
- avoid doing the work during the critical path
- cache it permanently
- cache it temporarily
- defer executing it after the main content

“

There are only two hard things in Computer Science: cache invalidation and naming things.

— Phil Karlton

CACHE STRATEGY II:

- Content as current as possible
- Cache hit ratio
- Cache invalidation complexity

Batteries included

Drupal 8 comes with caching enabled by default

Page Cache

Dynamic Page Cache

1

Old good Page Cache: Upgraded!

Almost a “poor man’s Varnish”

PAGE CACHE

Pros

- It's super fast
- It shortcuts bootstrap
- It's URL based
- New! Instantly updated when something is changed

Cons

- Only for anonymous users
- Assumes pages are identical for all anonymous users
- Does not use Authentication API
- Poor extensibility

2

Dynamic Page Cache

Built upon render cache, powered by the cacheability metadata.
Personalized parts are excluded automatically: they are turned into placeholders.

DYNAMIC PAGE CACHE

Pros

- Still quite fast
- Works for all users
- Personalized parts are turned into placeholders automatically
- Instantly updated when something is changed

Cons

- Slower than Page Cache
- Require devs to be aware of it

Render API: Caching

Drupal needs to be aware of how dynamic your code is.

metadata

placeholders

1

Cacheability Metadata

Keep Drupal informed about dependencies

CACHE TAGS

Avoid having stale content by using the appropriate cache tags.

CACHE CONTEXTS

Have different versions of a cache entry depending on the context

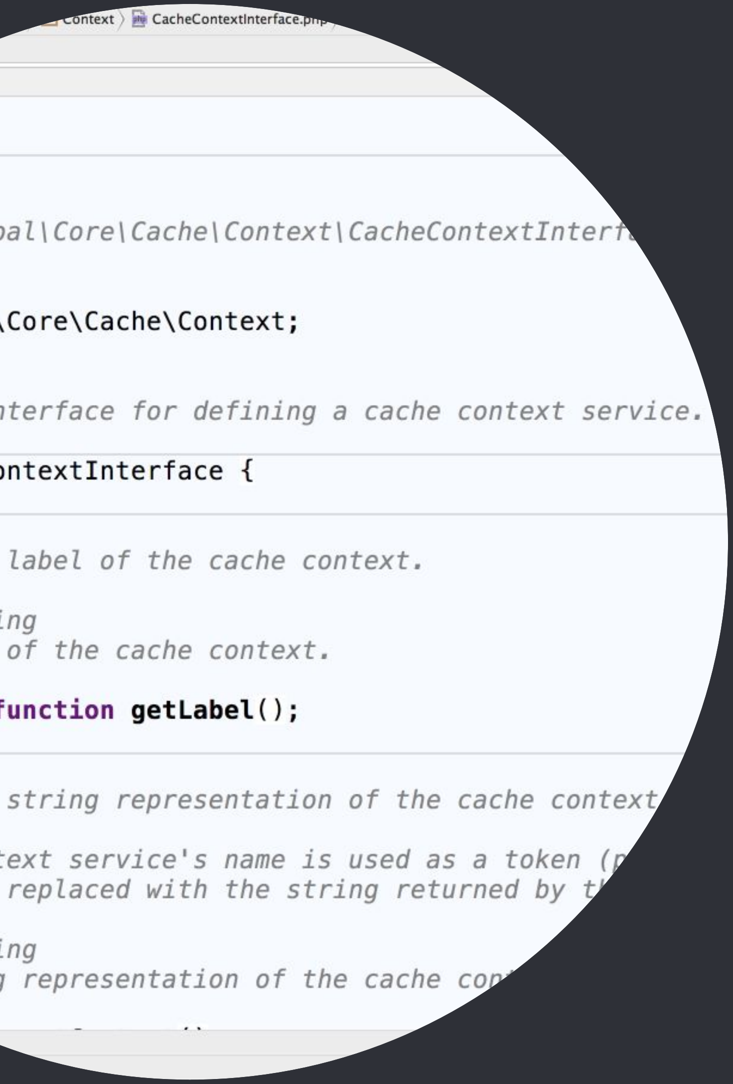
MAX-AGE

How old can your cache entry be before it's considered stale

Drupal 8 requires developers, to think about caching

Mindset

- I’ll always think about cacheability when rendering anything
- If it’s expensive I’ll cache it using cache keys
- If it varies depending on the situation, I’ll use contexts
- If anything will cause it to be outdated I’ll use tags
- If it may become stale I’ll use max-age



```
pal\Core\Cache\Context\CacheContextInterf

\Core\Cache\Context;

nterface for defining a cache context service.
ontextInterface {

 label of the cache context.

ing
 of the cache context.

function getLabel();

 string representation of the cache context

ext service's name is used as a token (p
 replaced with the string returned by t

ing
g representation of the cache con
```

EXAMPLE

Of cacheability metadata

A "REAL LIFE" PROBLEM

First node in the site:

- A block in the sidebar
- Contains a greeting to the user

You want this block to be cached!

Cache Tags

The first node

Cache Context

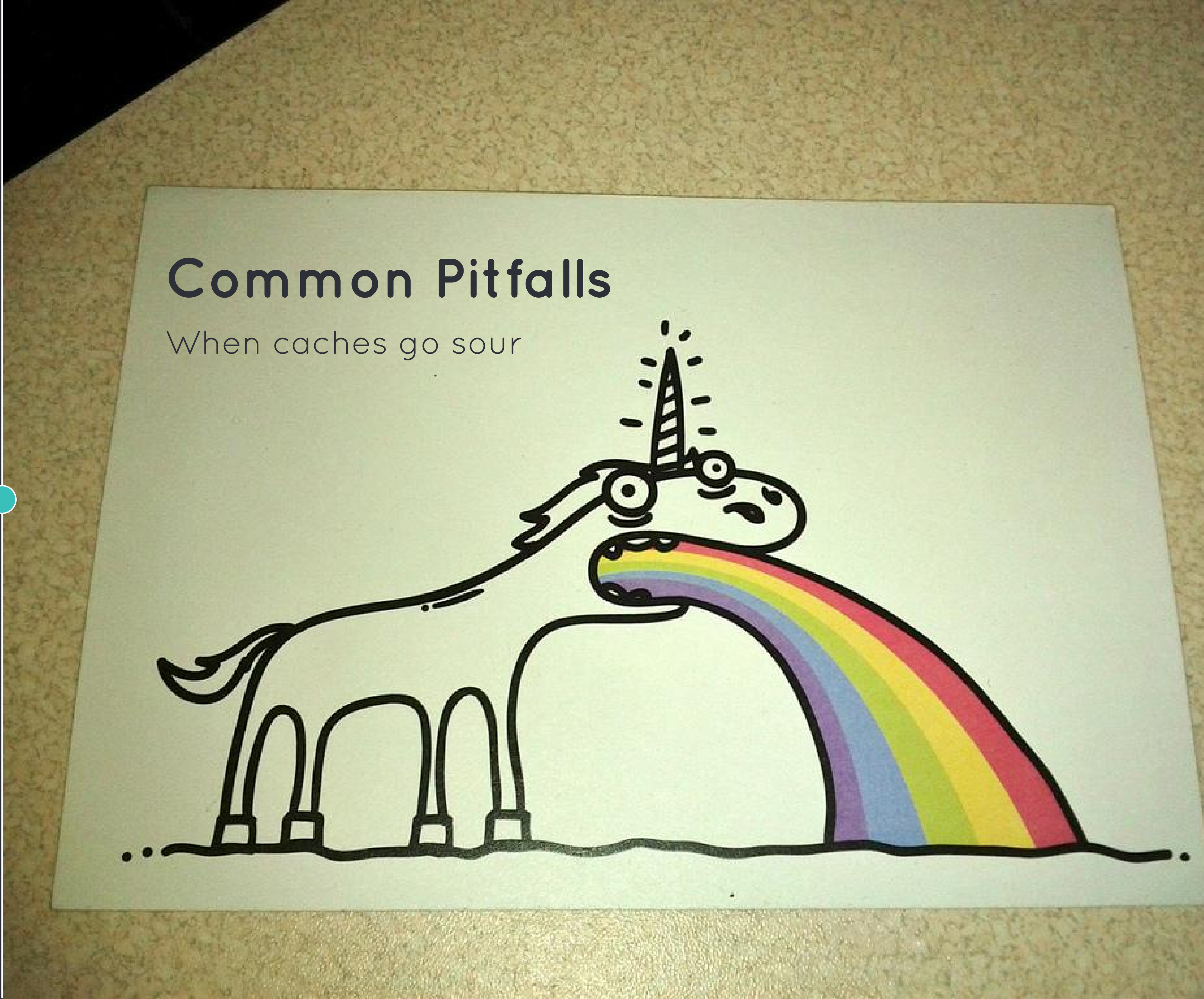
The user display name

Max-Age

Permanent

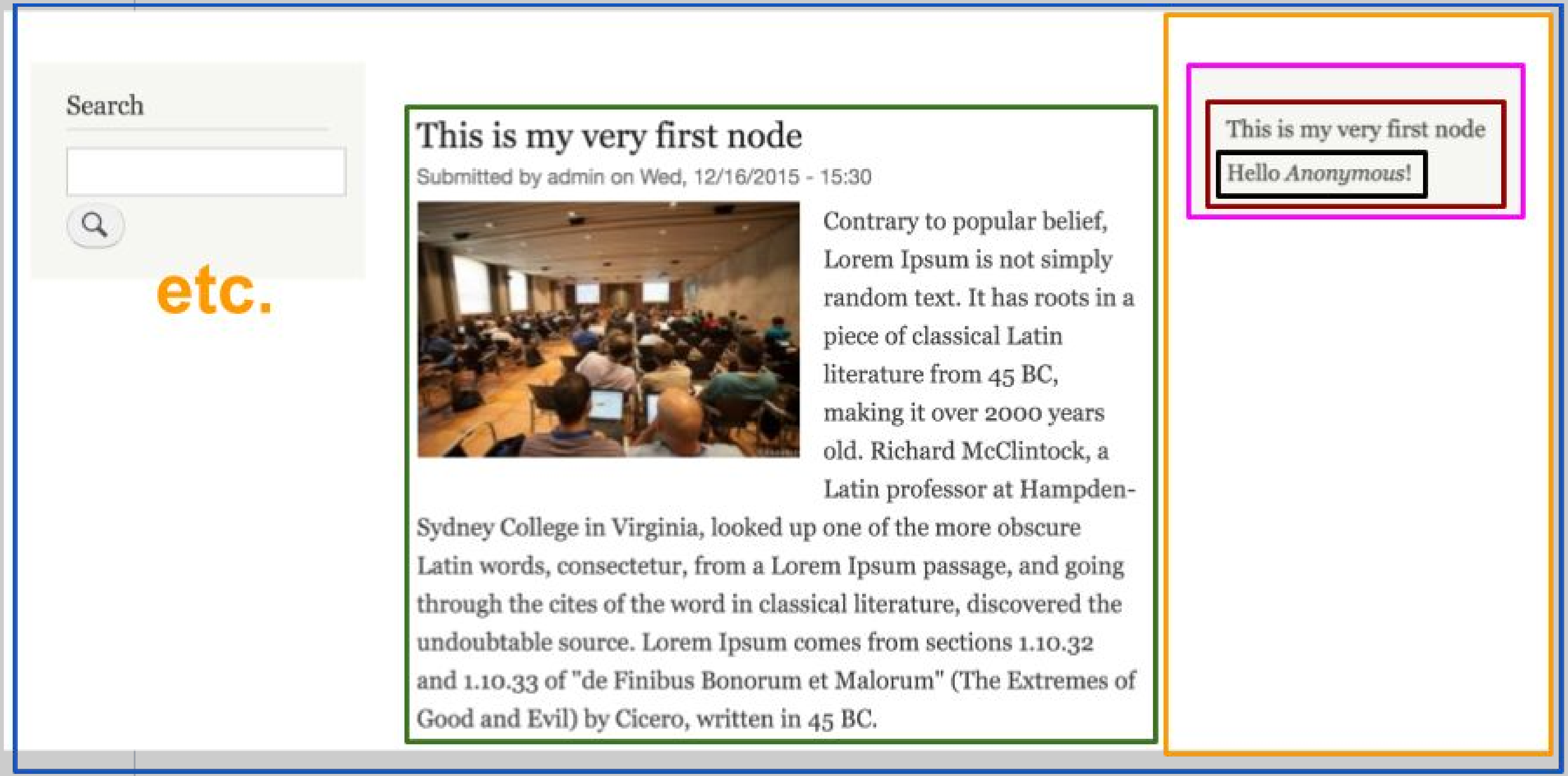
Common Pitfalls

When caches go sour



BUBBLING

Parents get children’s cacheability metadata



Search

etc.

This is my very first node

Submitted by admin on Wed, 12/16/2015 - 15:30

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC.

This is my very first node

Hello *Anonymous*!

Cacheability metadata in the **Black** box is surfaced to the **Maroon** one, and then to the **Pink** one, and then to the **Orange** one, and then to the **Blue** one.

The **Blue** one (page) contains all that, plus the **Green** box, etc. Every box inherits and adds its own.

2

Placeholdering

Drupal’s learning magic

PLACEHOLDERING FOR LAZY BUILDING

Automatic - Manual?

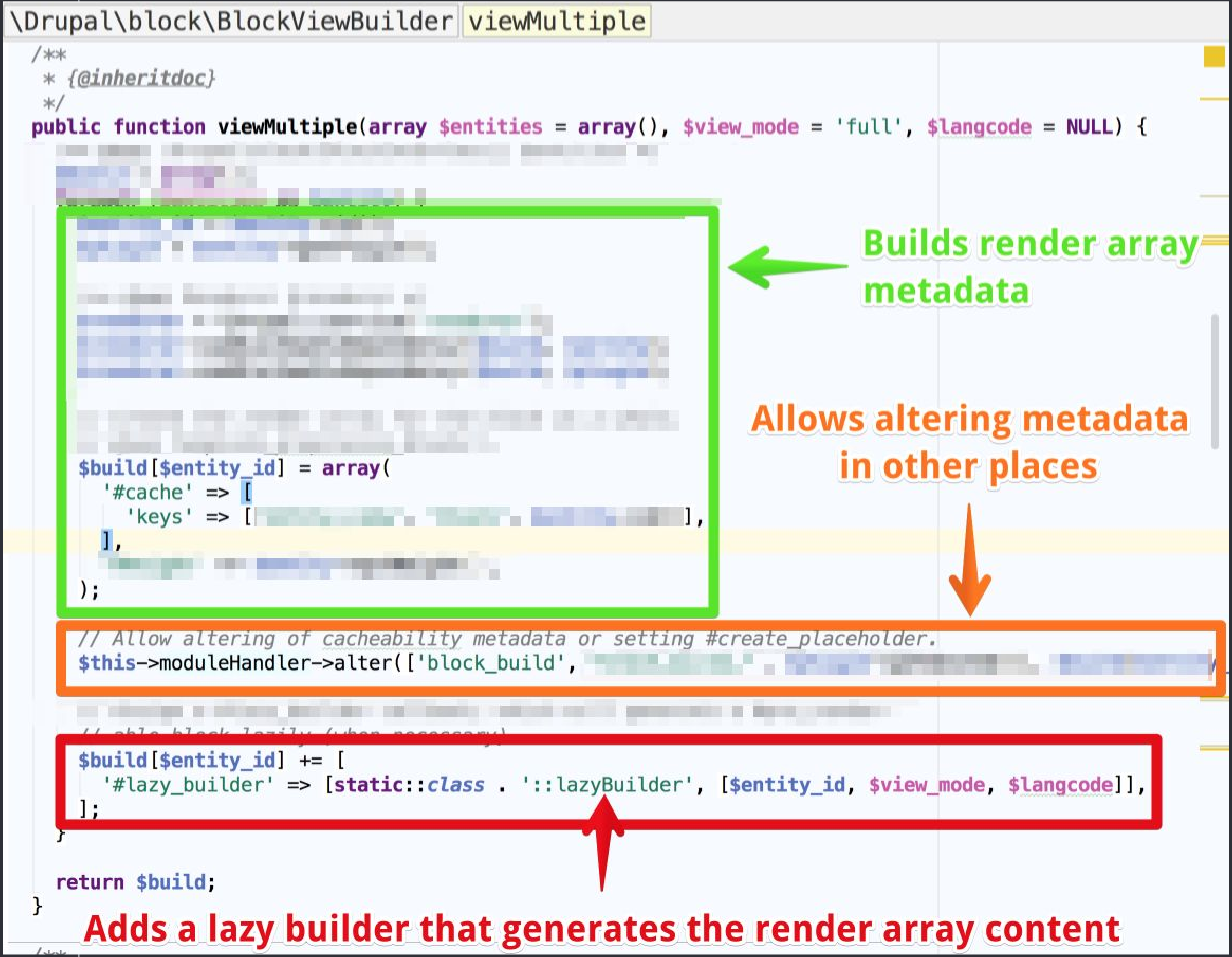
Enables Small & Big Pipes

It all comes down to setting:

`#create_placeholder = TRUE`

When there is:

`#lazy_builder = […, …]`



\Drupal\block\BlockViewBuilder
viewMultiple
Builds render array metadata
Allows altering metadata in other places
Adds a lazy builder that generates the render array content

```php
  /**
   * {@inheritdoc}
   */
  public function viewMultiple(array $entities = array(), $view_mode = 'full', $langcode = NULL) {
    [redacted]
      $build[$entity_id] = array(
        '#cache' => [
          'keys' => [[redacted]],
        ],
        [redacted]
      );

      // Allow altering of cacheability metadata or setting #create_placeholder.
      $this->moduleHandler->alter(['block_build', [redacted]

      [redacted]
      $build[$entity_id] += [
        '#lazy_builder' => [static::class . '::lazyBuilder', [$entity_id, $view_mode, $langcode]],
      ];
    }

    return $build;
  }
```

SETTING `#create_placeholder` to TRUE

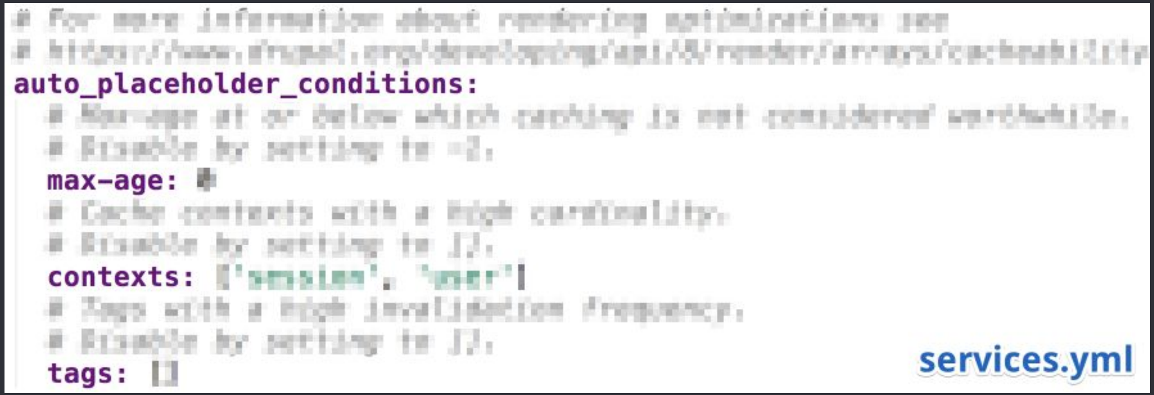
Manually

Via the alter hooks:

- `hook_block_build`
- …

Automagically

Detects configured **conditions** in cache metadata.



```
# For more information about rendering optimizations see
# https://www.drupal.org/developing/api/8/render/arrays/cacheability
auto_placeholder_conditions:
  # Max-age at or below which caching is not considered worthwhile.
  # Disable by setting to -1.
  max-age: 0
  # Cache contexts with a high cardinality.
  # Disable by setting to [].
  contexts: ['session', 'user']
  # Tags with a high invalidation frequency.
  # Disable by setting to [].
  tags: []
```

services.yml

Perceived performance

How all we have seen will make Drupal feels faster

Today:
BigPipe

Tomorrow:
RefreshLess

2.1

Big Pipe

- A placeholdering strategy.
- Applies when there is a session
- Works with and without javascript.

2.2

RefreshLess

Inspired by RoR’s turbolinks.

“

For us, the caching system alone justifies choosing the Drupal 8 platform

— NITEMAN & e0ipso

Thanks!

ANY QUESTIONS?

You can find me at

@e0ipso

@NITEMAN_es