Drupal 8 Caching

A Developer's Guide

Intro

Peter Sawczynec

Senior Customer Success Engineer

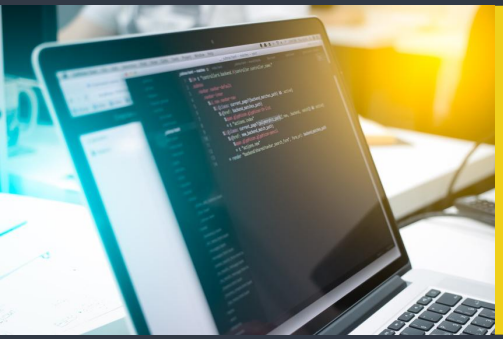
PANTHEON

pantheon.io

peter@pantheon.io

drupal.org/u/**xpsusa**

@sawczynec



Caching Overview

Caching Overview

Every Drupal 8 site is about caching

Caching is about being capable, smart, *and fast*

Begin with the end in mind

Caching Overview

Simplified view of D8 page caching:

Anonymous, monolithic, full-page static

Caching Overview

Real-world view of D8 page caching:

Anonymous, monolithic, full-page static

Anonymous, partially dynamic

News-oriented, more dynamic

Authenticated User, largely dynamic

Admin User, fully dynamic

Caching Overview

PANTHEON®

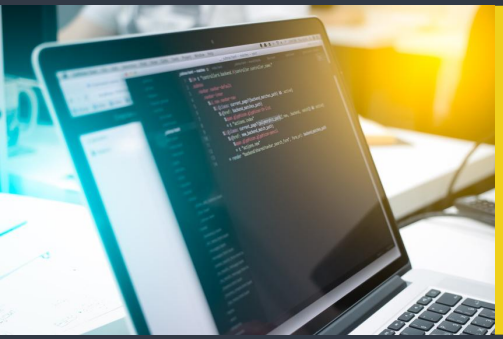
Caching in D8 is the art of trying to never compute or create anything more often than needed. This can refer to *an entire web page* or to *a portion of the web page* (like a block) or just refer to *the data* that is being used to create something.

All these items above can be cached.

Caching Overview

Effective D8 caching has a planned strategy

A caching strategy may seem unplannable, but if you start with a vision of your final web page objectives first, then what needs to be done to get there falls into place



Caching Basics

Edge Architecture Specification

Edge architecture extends the Web infrastructure through the use of HTTP surrogates -- intermediaries (a.k.a. “reverse proxies”) that act on behalf of and with the authority of the origin server.

Surrogates may be deployed close to the origin server, or throughout the network -- a configuration often referred to as “Content Delivery Network” (CDN).

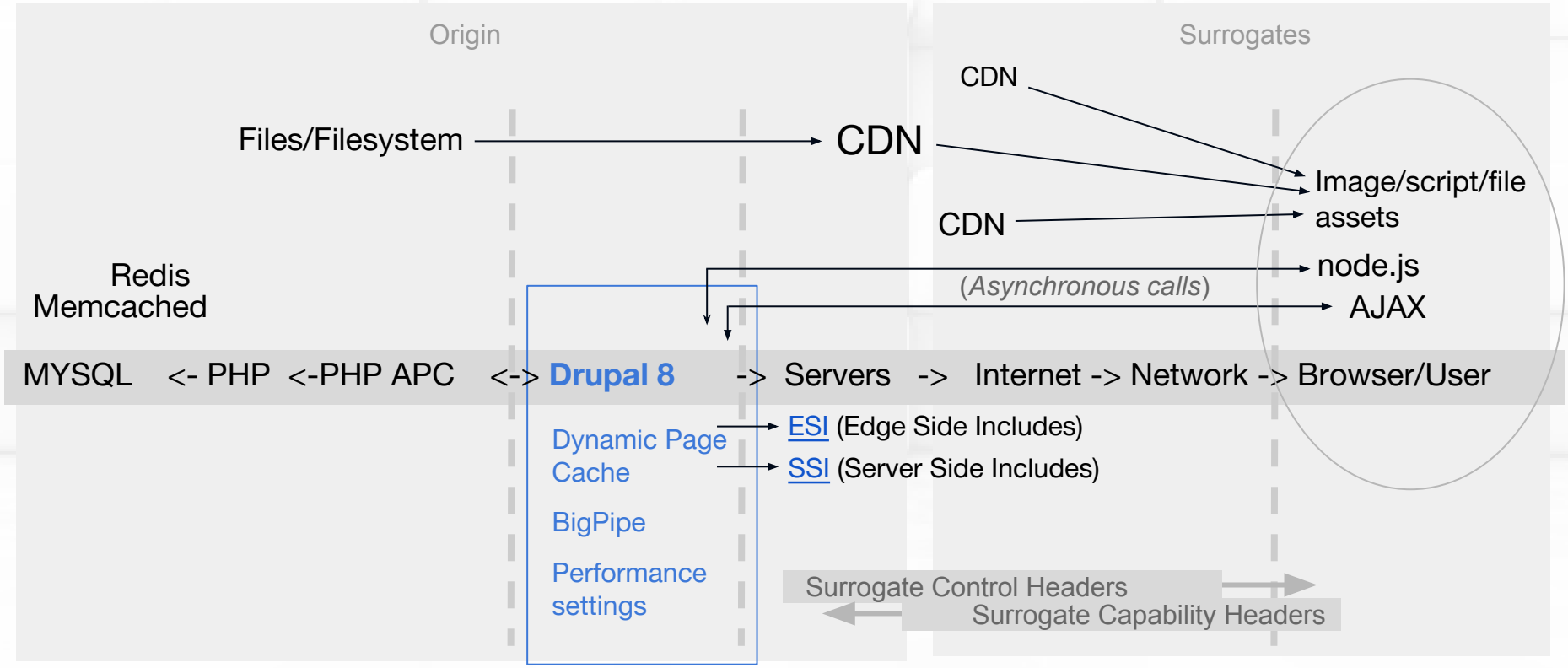
Edge Architecture Specification

HTTP headers are used by the origin to control surrogates and surrogates use HTTP headers to advertise their capabilities:

Surrogate-Control Headers (*origin, response, control*)

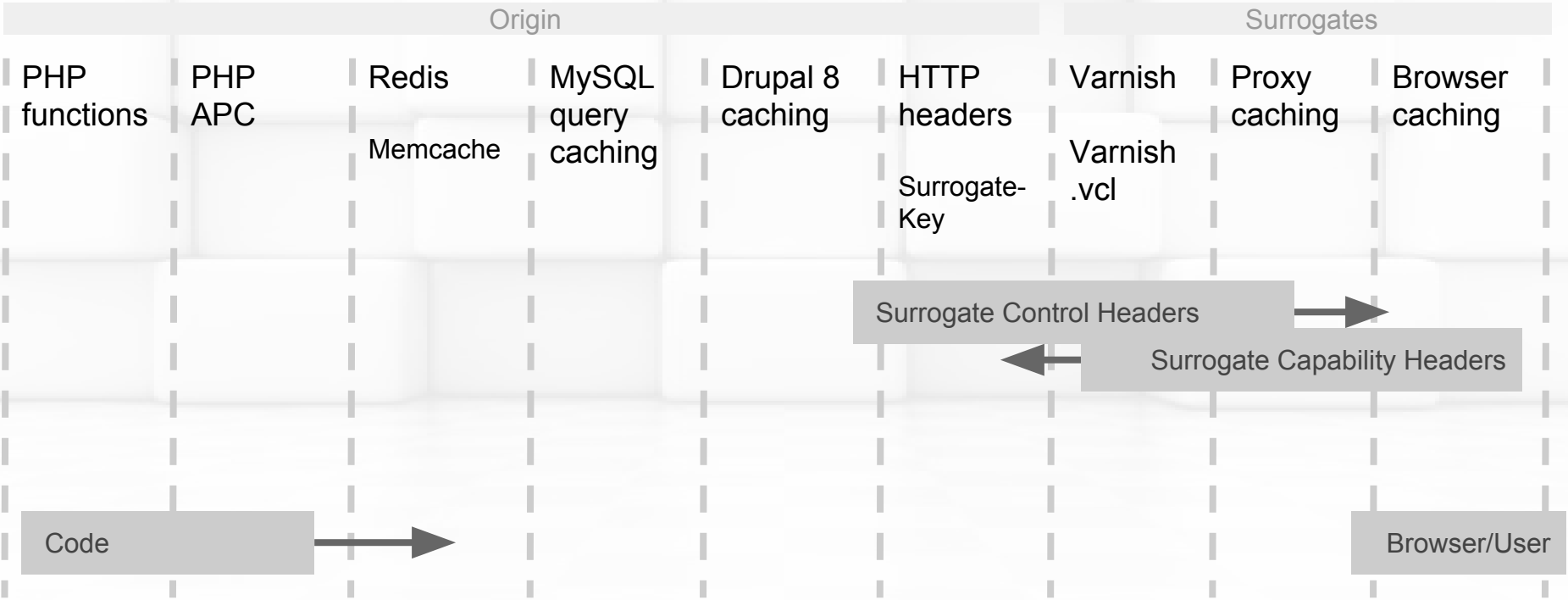
Surrogate-Capability Headers (*surrogates, request, advertise*)

Caching Basics



Caching Basics

CACHING OPPORTUNITY LOCATIONS



PANTHEON

Caching the results of PHP functions

- Cache an API call result (e.g. a Twitter or Facebook API call)
- Cache returned database data or the result of a function calculation

Cached temporarily into a variable in memory to re-use elsewhere on the same page load. Cached into a D8 cache storage bin, session cookie or a db table.

Drupal Cache API overview
Specific Details for D8 Cache Usage
Caching within functions - D8
Caching within functions - D7

PHP APC (Alternative PHP Cache)

Caches compiled PHP bytecode for reuse so that the PHP processing does not need to be recompiled each time

Object Caching

Redis

Redis is an open source, in-memory data structure store written in ANSI C and works in most POSIX systems like Linux, *BSD, OS X without external dependencies.
Predis (a standalone PHP library, requires PHP 5.3+) and PhpRedis (a PHP extension) allow PHP to interface with Redis.
D8 Redis module provides an interface between Drupal and Predis or PhpRedis that then interface with Redis

Memcached

Memcached is the actual daemon, an open source, high-performance, distributed memory object caching system.
Memcache (requires zlib) and Memcached (newer, requires libmemcached and PHP 5.2+) are two different PHP libraries that allow PHP to interface with the Memcached daemon (See: When should I use Memcache instead of Memcached?)
D8 Memcache module provides it's own API to interface Drupal with either the PHP Memcache or Memcached libraries to then interface with Memcached daemon.

Redis vs Memcache Why Redis beats Memcached for caching

MySQL query caching

Only some queries can be cached. The query cache stores the text of a SELECT statement together with the corresponding result that was sent to the client. The query cache is shared among sessions, so a result set generated by one client can be sent in response to the same query issued by another client.

The query cache does not return stale data. When tables are modified, any relevant entries in the query cache are flushed.

See: The MySQL Query Cache

PANTHEON®

Drupal 8 Caching

Dynamic Page Cache, Cache Tags, Performance page settings, BigPipe

Caching Basics

HTTP Headers

Cache-Control: public, max-age=
Expires:
ETag:
Vary:
X-Varnish:
Age:
Via:

[Guide to HTTP Cache Headers]

D8 HTTP Headers

X-Drupal-Cache: HIT
X-Generator: Drupal 8
X-Drupal-Cache-Contexts:
X-Drupal-Cache-Tags:

[How to expose all D8 HTTP response headers for debugging]

HTTP Surrogate-Key Headers

You can mark content with a surrogate key (Fastly specifc terminology) and use it to purge groups of specific URLs/pages at once without purging everything, or purge single URL/page singularly

This is a developer's option to handle. D8 does not automatically create the HTTP header Surrogate-Key as used by Fastly nor the HTTP header Cache-Tag as used by Cloudflare

D8 does automatically create the HTTP headers X-Drupal-Cache-Contexts and X-Drupal-Cache-Tags which at the time of this writing are not outputted into the actual D8 headers unless first enabled

The developer would need to grab the data from X-Drupal-Cache-Contexts and X-Drupal-Cache-Tags put them into the correct HTTP headers that the site's cache server uses (Surrogate-Key or Cache-Tag, for example) and then write those into the outgoing HTTP headers

This header info is captured as caching metadata by the cache server. Subsequently, targeted cache clears can be done by sending cache invalidation commands to the caching server, identifying the page(s) to clear by the caching metadata

HTTP Surrogate-Key Headers

See: Caching at the Edge: CDNs for everyone
Surrogate-Key integration based on URL segments and
X-Drupal-Cache-Tags and -Contexts headers are now only sent when developer explicitly enables them

Fastly module
Surrogate Keys: Part 1 Surrogate Keys: Part 2

Cloudflare module
The Cloudflare Drupal extension

Related docs

Drupal.org search for performance and scalability modules

Varnish Cache

Varnish handles requests in front of your site's backend. If Varnish doesn't have a request cached, the request is passed to your backend. As the response is returned from the backend, Varnish will read the response headers. If the response is cacheable, Varnish will cache it. Upon the next request for the same resource, Varnish will serve the cached version.

Caching Basics

PANTHEON®

Varnish will only cache resources that are requested through an idempotent HTTP verb, which are verbs that do not change the state of the resource.

HTTP verbs that Varnish can handle:

GET, HEAD (which can be cached)

PUT, POST, TRACE, OPTIONS, DELETE (cannot be cached, passed through to the backend)

All other HTTP methods bypass Varnish completely

Varnish Cache

Varnish has three important configuration points.

Built-in default settings, the varnishd startup options file, and the VCL file.

The varnishd startup options file contains default settings, port addresses, backend storage specification, etc.

Varnish VCL file

The varnish.vcl file is used to describe request handling and document caching policies for Varnish Cache.

VCL is the “Varnish Configuration Language” used within the varnish.vcl file

Varnish VCL and D8 Cache Tags
Use Drupal 8 Cache Tags with Varnish and Purge

Caching Basics

Some entries from a .vcl file:

```
backend default {
  .host = "127.0.0.1";
  .port = "8080";
  .connect_timeout = 600s;
  .first_byte_timeout = 600s;
  .between_bytes_timeout = 600s;
}
```

3 backend timeouts

Max time to wait for connecting to the backend (usually the web server)

Max time to wait for getting the first byte after connection above

Max time to wait between each byte of data coming from backend to Varnish

Non-Cacheable Headers

Response --> WWW-Authenticate: Basic realm="Authorized users"

Request <-- Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

Response --> Set-cookie:

Request <-- Cookie:

PANTHEON

Conditional Response and Request Headers

Response --> ETag <a unique hash identifier of this cacheable object that the Browser caches>

Request <-- If-None-Match

Returns 304 Not Modified or new response. Browser returns the original ETag value next time this page is requested again. If it matches the ETag on the cached page a 304 Not Modifeid Status code is returned. If the ETag at Varnish is now different Status Code 200 is returned and the new page with new ETag.

Response --> Last-modified <the datetime this cacheable object was last cached>

Request --> If-Modified-Since

Returns 304 Not Modified or new response. Browser returns the original Last-modified value next time this page is requested again as the If-Modified-Since request header.. If it matches the Last-modified on the cached page a 304 Not Modifeid Status code is returned. If the Last-modified at Varnish is now different Status Code 200 is returned and the new page with new Last-Modified..

PANTHEON

Conditional Response and Request Headers

Vary

Response --> Vary: accept-language

Request <-- Accept-Language

See: [Best Practices for Using the Vary Header]

PANTHEON

Expiration Response Headers

Expires: <absolute timestamp>

Cache-control:
Relative time in sec for object to live in cache.
Syntax example:
Cache-control: public, max-age=3600, s-maxage=86400

Example explained:
Cache-control: browsers and proxies, ttl for browser, ttl for proxies

Age: length of time already in cache

ttl = time to live
Cache-control first paramater:
public = cache by anyone

private | no cache | no store = prevent all caching

Expiration Precedence

Highest priority first:

beresp.ttl (in .vcl file)

S-maxage (cache-control for proxies)

Max-age (cache-control for browsers)

Expires 120 sec (default)

Varnish 5, VMOD, Varnish Plus

HTTP/2

HTTP/2 is not a form of caching, but an improved version of the HTTP 1.1 protocol that allows for more efficient request handling leading to faster delivery of web pages. Some of the software products and services that support HTTP/2:

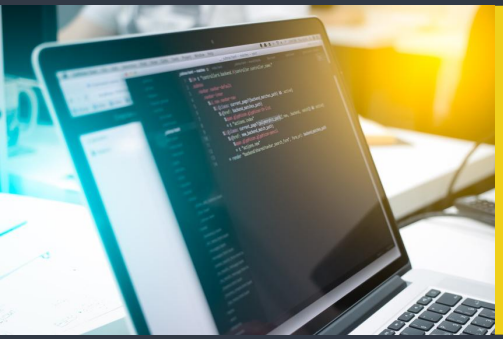
Nginx 1.9.5, Apache 2.4.12, node.js 5.0, Tomcat 8.5, Jetty 9.3, Fastly, Coudflare, CloudFront, Akamai

Proxy and Browser caching

A response after leaving your origin server can be cached at one or more locations (reverse proxies) on the way to the Browser. And the Browser performs caching too

Cookies and caching

The presence of cookies in the HTTP headers can cause Varnish to not cache your page as expected. The Varnish .vcl may have settings that you need to check.



D8 Caching

D8 Caching

PANTHEON

D8 CACHING RESPONSE FLOW

CODE	RENDER ARRAYS	D8 CONFIG	HTTP HEADERS	VARNISH
Custom Modules	Cache contexts	Performance settings	Cache-control: max-age=	Respects HTTP headers. Caches response or not.
Drupal Contrib	Cache Tags	Views cache settings	Surrogate keys	
Drupal core	Cache Keys	Panels cache settings		
	Max-Age			

D8 Caching

D8 caching breaks out into several major sections

Using D8 Caching features:

Cache API, Render Arrays, Cache Contexts, Cache Tags, Cache Keys, Max-Age

Understanding the core D8 modules involved:

Dynamic Page Cache, Page Cache, BigPipe

Configuring Performance Settings

Understanding how D8 communicates caching

D8 Caching - Two Main Concepts

Caching and **Cache Invalidation**

"***Caching*** is easy: it's merely storing the result of an expensive computation, to save time the next time you need it.

Cache invalidation is hard: if you fail to invalidate all the things that should be invalidated, you end up with incorrect results. If you invalidate *too many* things, your cache hit ratio is going to suffer, and you'd be inefficiently using your caches. Invalidating *only the affected things* is very hard."

-- *wim leers*

D8 Caching - Key Concepts/References

Key concepts:

- BigPipe
- Dynamic Page Cache (*formerly SmartCache*)
- Cache Tags
- Cache Contexts
- Cache Keys
- Max-Age

References:

- Cache API (*specifics*)
- Cache API (*overview*)
- D8 Block Cache
- Exploring the Cache API in D8
- Caching in D8
- Cacheability of Render Arrays
- CacheableResponseInterface
- CacheableDependencyInterface

D8 Caching - Caching Thought Process

1. Render Array Caching. I'm rendering something. That means think of cacheability.
   My Render Array Caching
2. Is this something that's expensive to render, and therefore is worth caching? If the answer is "yes", then what identifies this particular representation of the thing I'm rendering? Those are the **cache keys**.
   My Render Array Caching Invalidation (Cacheablility Metadata)
3. Does the representation of the thing I'm rendering vary per combination of permissions, per URL, per interface language, per … something? Those are the **cache contexts**. *Note: cache contexts are completely analogous to HTTP's Vary header.*
4. What causes the representation of the thing I'm rendering to become outdated? I.e. which things does it depend upon, so that when those things change, so should my representation? Those are the **cache tags**.
5. When does the representation of the thing I'm rendering become outdated? I.e. is the data valid for a limited period of time only? That is the **max-age** (maximum age). It defaults to "permanently (forever) cacheable" (Cache::PERMANENT). When the representation is only valid for a limited time, set a max-age, expressed in seconds. Zero means that it's not cacheable at all.

Cache contexts, tags and max-age **must always be set**, because they affect the cacheability of the entire response.
Therefore they "bubble": parents automatically receive them.
Cache keys must only be set if the render array should be cached.

Excerpted from Drupal.org: Cacheability of Render Arrays

D8 Caching - Cache Contexts

PANTHEON

cookies
  :name
headers
  :name
ip
languages
  :type
request_format
route
  .book_navigation
  .menu_active_trails
        :menu_name
  .name
session
theme

timezone
url
  .host
  .query_args
        :key
        .pagers
        :pager_id
  .site
user
  .is_super_user
  .node_grants
        :operation
  .permissions
  .roles
        :role

D8 Caching - Cache Max-Age

What max-age allows you to do:

When **$build['#cache']['max-age']** is not set:

permanent cacheability (Cache::PERMANENT) is assumed.

To indicate that a render array is not cacheable at all, set:
$build['#cache']['max-age'] = 0 (i.e. zero seconds).

And to indicate that a render array is cacheable only for a limited amount of time, e.g. 5 minutes, set:

$build['#cache']['max-age'] = 300; // set in seconds, i.e. 300 / 60 = 5 min.

D8 Caching - Render Arrays

It is advised that custom module code that creates output use render arrays to create output.

Render arrays are used to provide cacheability metadata to Drupal and Drupal decides to serve cached content or re-create content based on the caching metadata found in render arrays.

See: https://www.previousnext.com.au/blog/ensuring-drupal-8-block-cache-tags-bubble-page

D8 Caching - Render Array w/ Caching

```
function my_module_build_array( ) {
$build = [
            '#prefix' => '<aside>',
            '#markup' => t('Hi, %name, welcome back to @site!', [
               ' % name' => $current_user->getUsername( ),
               '@site' => $config->get('name'),
            ]),
            '#suffix' => '</aside>',
            '#theme' => 'my_module_build_arry_theme',
            '#cache' => [
                'contexts' => ['user', 'url.query_args:quantity'],
                'keys' => ['my_module_build_render', 'cache', 'demo'],
                'tags' => ['node:42:en, config.system.performance],
                'max-age' => 300,
            ],
            '#pre-render' => 'my_module_build_pre_render',
            '#attached' => [
               'library' => 'core / jquery',
               'drupalSettings' => ['foo' => 'bar'],
            ],
          ];
}
```

Render Array Reference

Cache will differ based on these variations in context

Terms that identify what this cache is about

Cache expires if any of these other things tagged here change

Cache expires after a specified time

D8 Caching - Block Caching

Block caching is typically set in the render array. It can also be set in code using Cache API functions.

Good reads related to Block caching:

Remove Block Cache API in favor of blocks returning #cache with cache tags

How do I correctly setup caching for my custom block showing content depending on the current node?

D8 Caching - Supporting Contrib Modules

PANTHEON®

Caching: A list of modules that make Drupal scale

Purge

Redis

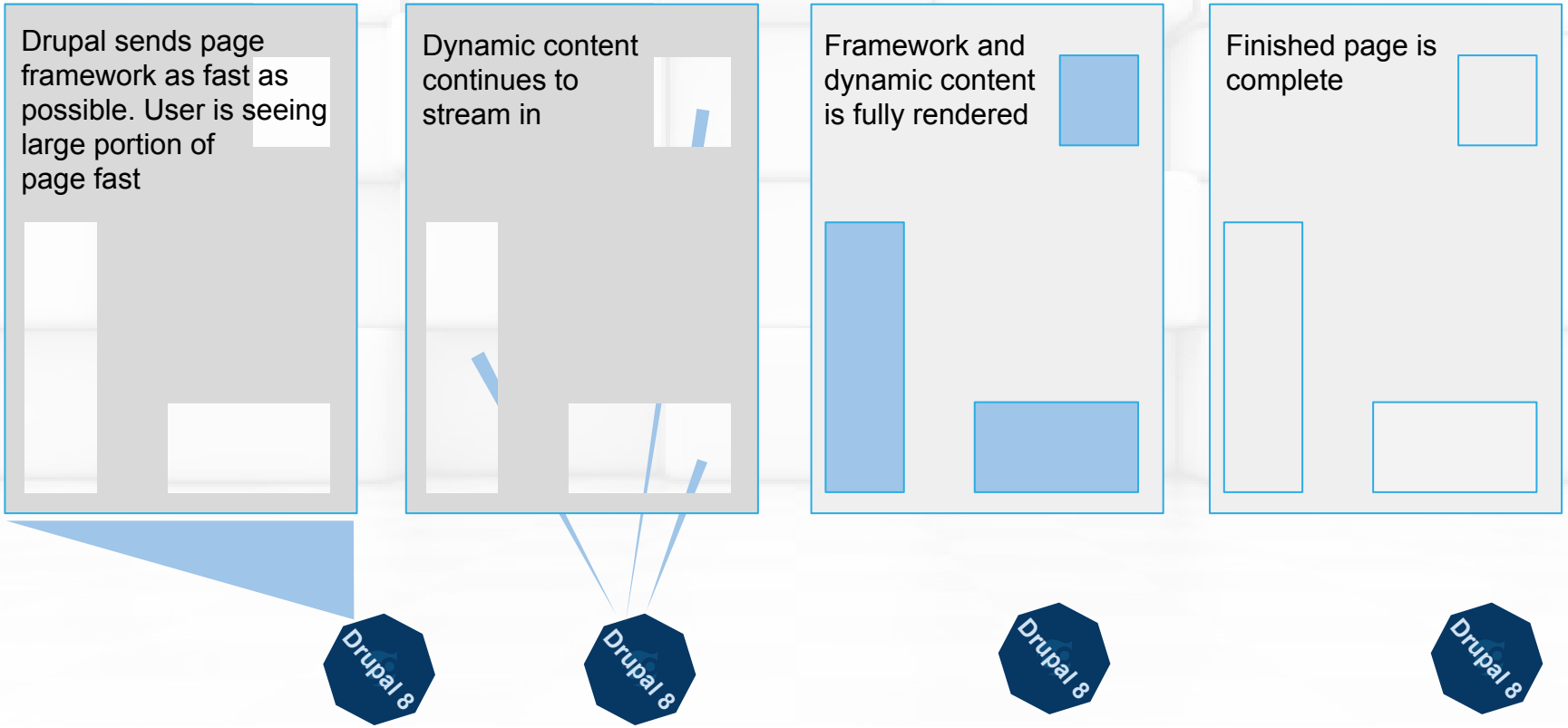
Memcache Storage

Cloudflare

D7:

Boost Cache Control Cache Expiration

D8 Caching - BigPipe Overview



D8 Caching - BigPipe Background

Originally an experimental contrib module. Included as a core module with Drupal 8.1 and later, but needs to be enabled. BigPipe requires JavaScript on the client. BigPipe may need configuration on your server(s), see: BigPipe environment requirements

BigPipe docs

Facebook's BigPipe: Pipelining web pages for high performance describes origin of the technique

Drupal.org BigPipe Overview Drupal.org BigPipe module Drupal.org BigPipe environment requirements page

Drupal.org Dynamic Page Cache module

D8 Caching - Dynamic Page Cache/BigPipe

Dynamic Page Cache (core/modules/dynamic_page_cache):
Caches pages for any user, handling dynamic content correctly

Page Cache (core/modules/page_cache):
Caches pages for Anonymous users. Use when an external page cache isn't avail.

BigPipe (core/modules/big_pipe):
Sends pages using BigPipe technique that allows browsers to show them much faster

D8 Caching - Performance Settings

Needs nothing, don’t look here.

Kidding.

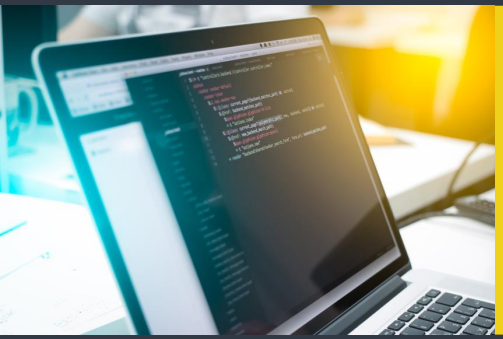
See: Drupal 8 Performance and Varnish Caching Settings

D8 Caching - Twig/Theme Issue

PANTHEON®

[Allow explicit bubbling of cacheability metadata inside Twig template (when accessing data from instead of rendering render arrays)]

[Ensuring Drupal 8 Block Cache Tags bubble up to the Page]



Caching-Related Tech

Other technologies

PANTHEON®

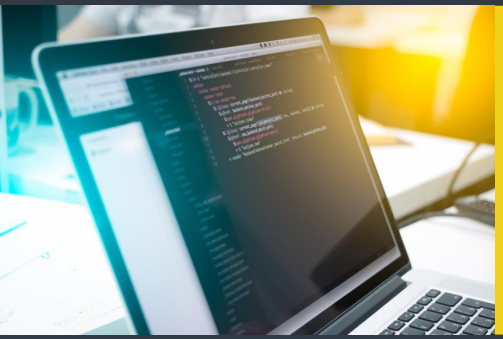
Other technologies to use in tandem with caching:

CDN: Caching assets globally

AJAX: Pull block(s) of content into a web page from within the Browser using asynchronous JavaScript (jQuery) calls

ESI (Edge Side Includes), **SSI** (Server Side Includes): Pull content into a web page on the web or cache server

Cron (Elysia Cron, Ultimate Cron): Manage clearing cache, rebuilding caches



D8 Reading

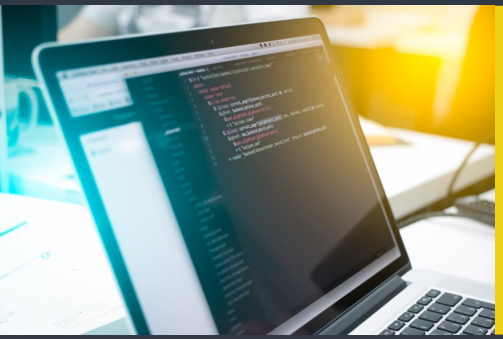
D8 Cache Reading

Drupal 8 API Reference

(https://api.drupal.org/api/drupal/8.2.x)

Curl and other caching verification methods

http://www.isvarnishworking.com Instant Varnish



D8 Training Resources

D8 Training/Resources

Drupalize.me

Buildamodule.com

Knp University

Safaribooksonline.com

Drupal Association - YouTube
(https://www.youtube.com/user/DrupalAssociation)