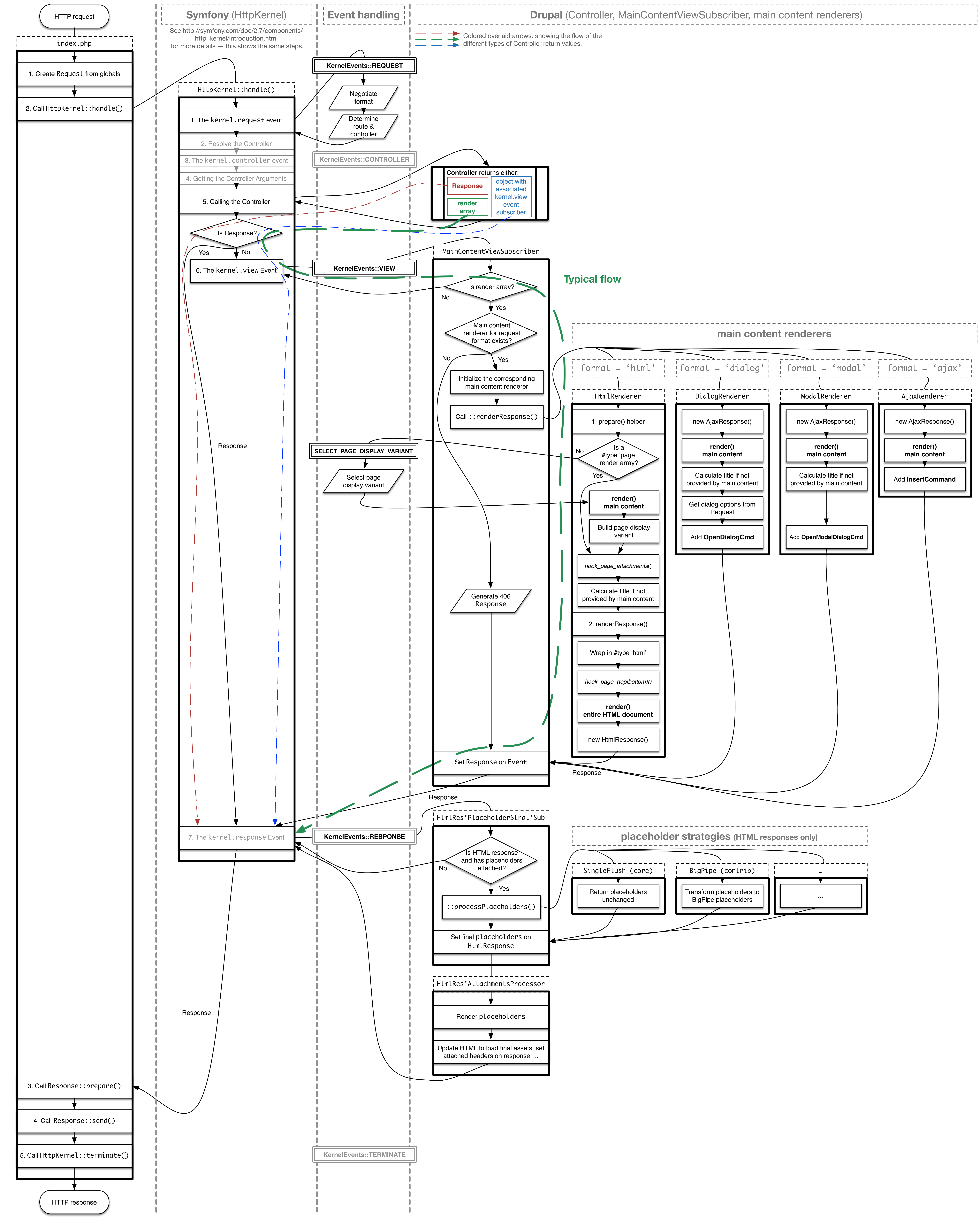
Request handling and rendering flow


HTTP request
index.php
1. Create Request from globals
2. Call HttpKernel::handle()
3. Call Response::prepare()
4. Call Response::send()
5. Call HttpKernel::terminate()
HTTP response
Symfony (HttpKernel)
See http://symfony.com/doc/2.7/components/http_kernel/introduction.html for more details — this shows the same steps.
HttpKernel::handle()
1. The kernel.request event
2. Resolve the Controller
3. The kernel.controller event
4. Getting the Controller Arguments
5. Calling the Controller
Is Response?
Yes
No
6. The kernel.view Event
Response
7. The kernel.response Event
Response
Event handling
KernelEvents::REQUEST
Negotiate format
Determine route & controller
KernelEvents::CONTROLLER
KernelEvents::VIEW
SELECT_PAGE_DISPLAY_VARIANT
Select page display variant
KernelEvents::RESPONSE
KernelEvents::TERMINATE
Drupal (Controller, MainContentViewSubscriber, main content renderers)
Colored overlaid arrows: showing the flow of the different types of Controller return values.
Controller returns either:
Response
render array
object with associated kernel.view event subscriber
MainContentViewSubscriber
Is render array?
No
Yes
Main content renderer for request format exists?
No
Yes
Initialize the corresponding main content renderer
Call ::renderResponse()
Generate 406 Response
Set Response on Event
Response
Typical flow
main content renderers
format = 'html'
format = 'dialog'
format = 'modal'
format = 'ajax'
HtmlRenderer
1. prepare() helper
Is a #type 'page' render array?
No
Yes
render() main content
Build page display variant
hook_page_attachments()
Calculate title if not provided by main content
2. renderResponse()
Wrap in #type 'html'
hook_page_(top|bottom)()
render() entire HTML document
new HtmlResponse()
DialogRenderer
new AjaxResponse()
render() main content
Calculate title if not provided by main content
Get dialog options from Request
Add OpenDialogCmd
ModalRenderer
new AjaxResponse()
render() main content
Calculate title if not provided by main content
Add OpenModalDialogCmd
AjaxRenderer
new AjaxResponse()
render() main content
Add InsertCommand
Response
HtmlRes'PlaceholderStrat'Sub
Is HTML response and has placeholders attached?
No
Yes
::processPlaceholders()
Set final placeholders on HtmlResponse
placeholder strategies (HTML responses only)
SingleFlush (core)
Return placeholders unchanged
BigPipe (contrib)
Transform placeholders to BigPipe placeholders
…
…
HtmlRes'AttachmentsProcessor
Render placeholders
Update HTML to load final assets, set attached headers on response …

Explanation

Some parts are greyed out because they're of lesser importance: they do happen, but are not crucial for understanding the flow.

Example: **GET /admin.**

Typically, the format will be **html**, but it could also be **ajax, dialog, modal, json, hal_json, and more.** See: Symfony's Request::setFormat().

Example: **/admin path → system.admin route**
That route has the following attributes:
_content: '\D\s\C\SystemController::systemAdminMenuBlockPage'
_title: 'Administration'

_controller is set to '\D\s\C\SystemController::systemAdminMenuBlockPage'

\D\s\C\SystemController::systemAdminMenuBlockPage is called, it returns a render array with no #type.

MainContentViewSubscriber only handles render arrays!
If it's anything else (e.g. an object), then it's up to another VIEW event subscriber to turn it into a Response.

MainContentViewSubscriber looks at the available main content renderer services. These are **tagged with *render.main_content_renderer*.** Contributed modules can additional renderers.

The flow for all four formats supported in core in their respect is displayed next, but the **explanations below only apply to the HtmlController!**
If an unsupported format was negotiated, a 406 response is generated.

Example: **BlockPageVariant** selected.

The main content must already be rendered, because it might set the page title.

The result of a built page display variant is a #type 'page' render array.

hook_page_attachments() operates on #type 'page'.

Only accepts render arrays of #type 'page'!

Renders the 3 parts of the HTML body: *page_top, page, page_bottom* (where *page* corresponds to #type 'page' and hence **page.html.twig**). **Then** renders the entire HTML (i.e. the **html.html.twig** template).

Note that HtmlResponse (as well as AjaxResponse) at this point still contains *attachments:* asset libraries, headers … but also *placeholders.*

ChainedPlaceholderStrategy looks at the available placeholder strategies. These are **tagged with *placeholder_strategy*.** Contributed modules can additional renderers.

Placeholder strategies can transform standard Drupal placeholders (designed for non-deferred rendering) into other placeholders, which may be replaced with the final markup using any way possible.
For example: BigPipe, ESI …

BigPipe is a contrib module during 8.0.x, will likely be in 8.1.x core.
See https://www.drupal.org/project/big_pipe

Process all other HtmlResponse attachments: asset libraries, HTML <head> elements, headers …

BigPipe placeholders rendered at the end of::send(), after most is sent.

Example: response for the path **/admin**.