Layouts for CMS developers

For CMS developers, Exponential Layouts from the developer's side. It covers layouts, zones, blocks and rules, how a page resolves its layout, how layouts meet templates in the pagelayout, layout types, block...

Source file: `doc/eng-us/help/cms-developers/layouts-for-cms-developers.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/cms-developers/layouts-for-cms-developers.md

This page is for developers who decide which parts of a site editors compose themselves. **Exponential Layouts** (the `explayouts` extensions, a port of the upstream layouts packages to this kernel) lets editors build pages from blocks in zones, without code, while your templates and content model stay in charge of what each block looks like.

Contents

1. The pieces
2. How a page gets its layout
3. Where layouts meet your templates
4. What you define as a developer
5. Template blocks: your code inside a layout
6. Shared layouts
7. Layouts between installations
8. Layouts or templates
9. After a change

The pieces

Piece	What it is
Layout	the plan of a page, made by an editor in the layouts editor
Layout type	the zones a layout has (*1 column*, *2 columns*, *Hero + 3 columns* ...), defined in settings with a template
Zone	an area of the layout; holds blocks one under the other
Block	a title, rich text, an image, a list of content, a gallery, a hero and more; each of a block definition with view types
Collection	the items of a list block: picked by hand, or filled by a query
Rule (mapping)	which pages use a layout: targets (a location, a subtree, a module page) and conditions (a class, a siteaccess)
Shared layout	a layout whose zones others link to, such as header and footer

The editor's view: The layouts editor; the tour: Layouts editor tour.

How a page gets its layout

For each request the resolver checks the enabled rules by priority; the first whose targets and conditions match the page wins. Rules live in the database, managed under **Rules** (**Layout mappings**) in the layouts administration, and a node's **Layouts** tab shows which layout it resolves to (Layout rules, Layout mappings).

Where layouts meet your templates

The pagelayout asks for the page's layout and draws it, or falls back to the normal view result:

Template

```
{def $page_layout = fetch( 'explayouts', 'resolve_layout', hash() )}
{if $page_layout}
    {include uri='design:explayouts/layout.tpl' layout=$page_layout module_result=$module_result}
{else}
    {$module_result.content}
{/if}
{undef $page_layout}
```

Blocks of a layout can show the page's own content (the full view in a layout), so a layout can frame the template you wrote for a class rather than replace it. Other fetch functions answer which layout or which rules a node has (Fetching a layout in templates).

What you define as a developer

You define	Where	Read
Layout types and their zones	`[LayoutType_<name>]` in `explayouts.ini` (in your extension's append file) and a layout type template	Layout types and zones
New view types of existing blocks	the block definition's view types and a template; no PHP	Add a view type
New block types	a block definition and its templates	Add a block type, Block definitions
How blocks look	block view templates and stylesheets in your design	Block view templates, Styling layout blocks
How list items look	your view modes per class	View modes

Editors then combine what you defined. Keep the number of layout types and view types small: each one is a choice an editor must understand.

Template blocks: your code inside a layout

A **template block** places a template of your design in a zone. Use it for parts that need code (a search form, a calendar fed by a fetch, a widget) while editors still decide where they appear (Template blocks). This is where the two ways of building meet (Two ways to build).

Shared layouts

A shared layout holds blocks many pages use; a zone of another layout **links** to a zone of the shared layout instead of copying its blocks. Editing the shared layout changes every page that links to it (Shared layouts and zone linking).

Layouts between installations

Layouts and rules live in the layouts tables of the database, not in content packages. They travel as one ZIP file:

Shell

```
./console exp:layouts list
./console exp:layouts export --all --output=var/tmp/layouts.zip
./console exp:layouts import var/tmp/layouts.zip --dry-run
```

(Export and import layouts). Rules that target locations refer to node IDs, which differ between installations; check them after an import.

Layouts or templates

Situation	Build with
Landing pages, campaigns, a front page editors rearrange often	layouts
Every article, event or product page	a template per class, possibly framed by a layout
A list whose content follows rules (newest five news)	a layout block with a query collection, or a fetch in a template
Something that needs code	a template, placed with a template block if editors choose where

The Decision table goes through more cases.

After a change

Layout changes are published in the editor; settings and templates need caches cleared. The resolver keeps its own cache, and pages are cached in the view caches and in Velocity's response cache (Caches after a change, Why doesn't my change appear?).

Related pages

- Two ways to build
- Layouts for designers
- Trainers' course: Exponential Layouts
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

Become an Exponential Partner

Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.

Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.