Ghost CMS for Custom Theme Development

1. What is Ghost CMS & Why We Use It

Ghost CMS is a modern, open-source Content Management System built specifically for **professional publishing**. Unlike traditional plugin-heavy CMS platforms, Ghost follows a **minimalist, performance-first philosophy**. It is built on **Node.js**, which enables high speed, scalability, and a clean developer experience.

Ghost is purpose-built for **content-centric websites**, including:

- Personal and professional blogs
- Online magazines and publications
- Paid newsletters
- Membership-based platforms

Instead of trying to be a general-purpose website builder, Ghost focuses on **writing, managing, and delivering content efficiently**.

Why We Use Ghost CMS

Ghost offers clear advantages for developers and content-focused organizations:

Performance

- Built on Node.js for faster page loads
- Lightweight core with API-first architecture
- Scales better than legacy CMS platforms

Simplicity

- Distraction-free, block-based editor for writers
- Clean and predictable theme system for developers
- Well-documented APIs and helpers

Content-First Approach

- Publishing-focused by design
- Memberships, subscriptions, and newsletters are built into the core
- No dependency on third-party plugins for core publishing features

---

2. What We Can Do in the Ghost Admin Panel

For Ghost theme developers, the Admin panel is the central control system that manages all content, assets, and configurations consumed by a theme. A well-structured theme must depend entirely on Admin-managed data and avoid hardcoded values.

Key Areas of the Ghost Admin Panel

Dashboard

Provides an overview of site activity, published content, and basic performance insights. It helps confirm that content is being created and rendered correctly through the theme.

Posts & Pages

- **Posts**: Dynamic, chronological content such as blogs or articles.

- **Pages**: Static content like About, Contact, or Policy pages.
  Themes must properly support both post and page templates and render content dynamically.

Tags

Used to organize content and generate archive pages.
Themes should correctly display tag names, descriptions, feature images, and tag-based listings.

Members & Membership

Manages free and paid subscriptions along with content access control.
Themes must respect Ghost membership visibility rules and helpers.

Settings (Critical for Theme Development)

Design

Used to upload and manage the site logo, cover image, and publication icon.
All design assets must be rendered dynamically in the theme. Hardcoded images inside theme files are not allowed.

Navigation

Allows configuration of primary and secondary menus.
Menu updates should reflect in the theme immediately without requiring code changes.

Integrations

Used to manage API keys and external services.
Themes must not contain hardcoded credentials or integration details.

Theme and Routing Management

Theme Upload

Themes are uploaded through the Admin panel using the ZIP file of the Git repository. Manual server-side file edits are discouraged.

Image Handling

Any image added through the Admin panel must use web-optimized image formats, preferably .webP, to ensure better performance and faster loading.

Routing (routes.yaml)

The routes.yaml file can be uploaded or updated via the Admin panel when custom routes or redirects are created or modified. Themes must remain compatible with routing changes without requiring code modifications.

Content migration: JSON files can be uploaded via the Admin panel to migrate posts, pages, tags, and members between Ghost installations.

---

3. Custom Theme Building in Ghost

What is a Ghost Theme?

A Ghost theme is a directory of **Handlebars templates, assets, and configuration files** that define the complete visual presentation of a site. Themes use helpers to dynamically inject content from Ghost into HTML.

Ghost themes follow a **layout inheritance model**, where individual templates extend a shared base layout (default.hbs) to avoid duplication.

Recommended Theme Folder Structure

```
my-theme/
├── assets/
│   ├── css/
│   │   └── screen.css
│   ├── js/
│   │   └── main.js
│   └── images/
├── partials/
├── default.hbs
├── index.hbs
├── author.hbs
├── post.hbs
├── page.hbs
├── package.json
```

Role of Key Files

.hbs Files

- Core templates selected automatically by Ghost routing

default.hbs

- Master layout with <html>, <head>, and <body>
- Must include:
    - {{ghost_head}}
    - {{ghost_foot}}
    - {{body_class}}

index.hbs

- Displays lists of posts
- Used for homepage, tag archives, or author archives if specific templates are missing

post.hbs

- Renders individual blog posts

page.hbs

- Renders static pages

partials/

- Reusable template fragments
- Included using {{> partial-name}}

- Ensures a DRY and maintainable codebase

assets/

- Stores CSS, JavaScript, images, and fonts
- Must use the {{asset}} helper for correct paths and caching

package.json

- Mandatory theme configuration file
- Contains metadata and settings like posts_per_page
- Rules:
    - Double quotes only
    - No trailing commas
    - Valid JSON syntax

Handlebars Helpers & Content Cards

Ghost uses **Handlebars** as its templating language.

Examples

- {{title}} – outputs post title
- {{content}} – outputs full post content
- Block helpers like {{#post}}...{{/post}} define context

Content Cards (kg-cards)

- Ghost editor is block-based
- Each block outputs a specific CSS class:
    - .kg-image-card
    - .kg-gallery-card
    - etc.
- A professional theme must include CSS for all major kg-card types to ensure proper frontend rendering

Common Mistakes & Best Practices

Common Mistakes

- Invalid package.json
- Missing {{ghost_head}} or {{ghost_foot}}
- Incorrect asset linking without {{asset}}
- Not restarting Ghost in production after theme changes

Best Practices

- Start with required files: package.json, default.hbs, index.hbs, post.hbs
- Use partials extensively
- Develop locally for instant template reloads
- Validate themes frequently during development

---

4. Production Requirements, GScan & Referencesweb

Important Things Every Custom Theme Must Have

Layout Structure

- Single source of layout inheritance using default.hbs

Mobile Responsiveness

- Use modern CSS (media queries, Flexbox, Grid)

SEO Essentials

- Semantic HTML
- Mandatory {{ghost_head}} for metadata and structured data

Accessibility

- Semantic elements
- Proper color contrast
- Alt text and keyboard navigation

Performance

- Lightweight CSS and JS
- Optimized images
- Cached assets

Maintainability

- Clean folder structure
- Consistent naming conventions (e.g., BEM)
- Well-documented code

GScan: Theme Validation

GScan is the official Ghost theme validation tool and a mandatory part of the development workflow.

What It Does

- Validates theme structure, syntax, helpers, and security

Why It Matters

- Identifies issues that can break production sites
- Required for Ghost Marketplace submission

Common Errors Detected

- Invalid package.json
- Missing required templates
- Deprecated helpers
- Missing {{ghost_head}}

How to Use

- Validate themes regularly during development
- Always run before deployment

- Online validator:
  https://gscan.ghost.org/

References Theme

Custom Ghost theme repositories for real-world implementation and advanced patterns:

- https://github.com/Sakthi10122004/ghostbase-theme.git
    - **Branch (dev):** Vertical Navigation Bar
    - **Branch (dev1):** Horizontal Navigation Bar