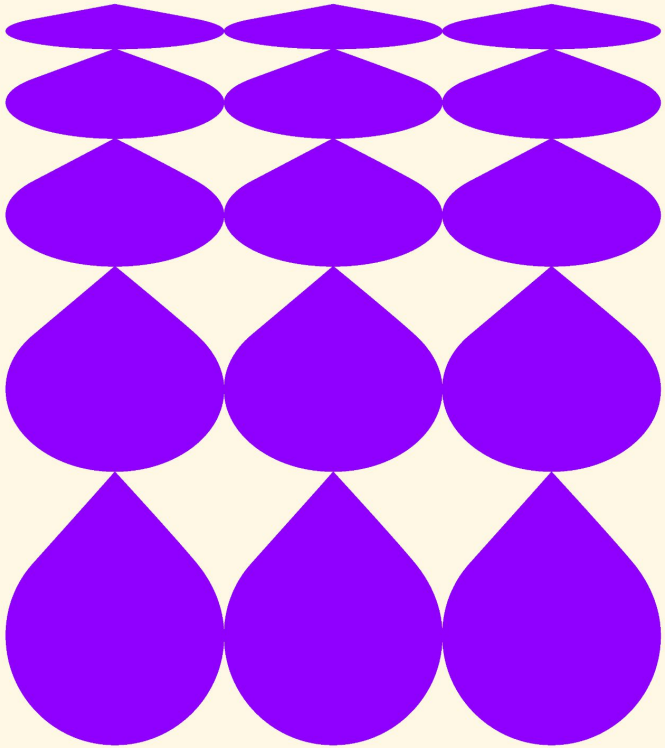
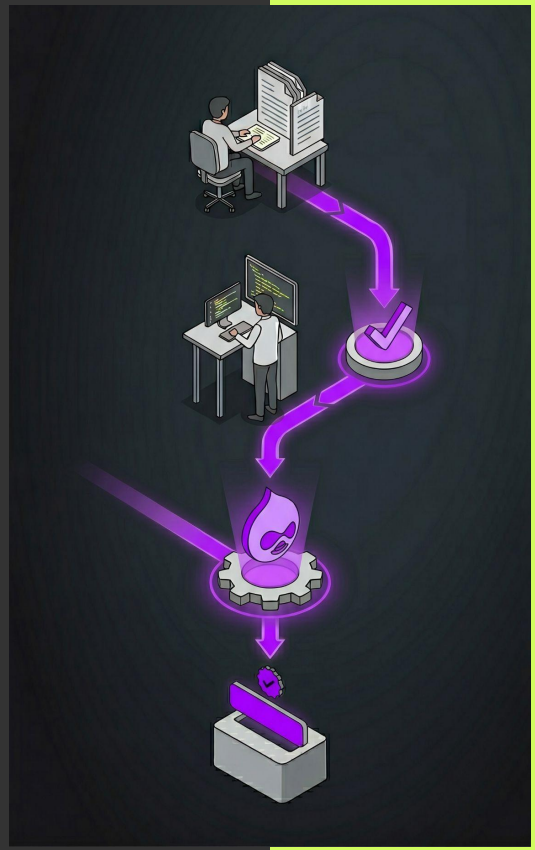
Secure and quality content - Automating moderation tests with Cypress

Marji Cermak

Morpht



The top level concepts of peer review



What Is Peer Review?

- The practice of having a second (or third) set of eyes check work before it reaches its audience
- Used across high-stakes fields: academic publishing, medicine, aviation, law
- In a content context: an editor creates, a reviewer checks, a publisher approves
- The goal is not distrust — it is *systematic quality assurance*

Why Peer Review Matters for Content

- Published content carries the authority and reputation of the organisation
- A single factual error can erode public trust at scale
- Corrections are visible — they signal that the process failed
- Government content often has legal, health or financial implications for citizens

The Review Lifecycle

Content travels through states:

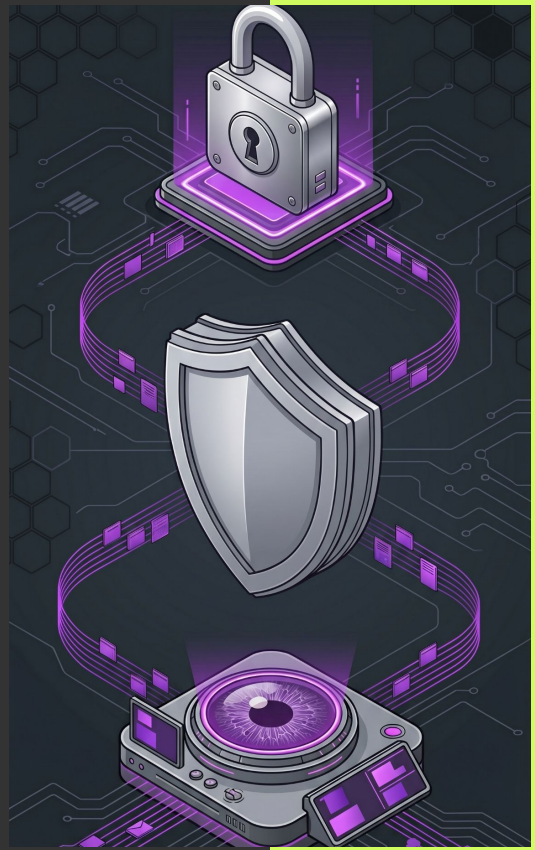
- Draft
- Needs Review
- Published
- Archived

(note, there is no Delete)

Accountability and the Audit Trail

- Every **transition** should be logged: who did what, and when
- Accountability discourages careless approvals
- An audit trail is essential for post-incident review
- It also protects staff - they can demonstrate they followed the correct process

The information security requirements for gov agencies



Gov Content Is a Security Surface

- Content is not just communication — it is infrastructure
- Unauthorised publication of sensitive content is a security incident
- Premature release of policy can have legal and market consequences
- Malicious or accidental misinformation on a .gov.au site carries outsized authority

The Frameworks That Apply

- PSPF (Protective Security Policy Framework) - governs information handling across Australian government agencies
- ISM - agencies are accountable for the accuracy, integrity, and authorised release of their content
- Common requirement: separation of duties, **least privilege**, and access logging
- Web content workflows must demonstrably satisfy these controls

The Principle of Least Privilege in Practice

- Each user role should have access to only what their job requires
- Editors should not be able to publish — only to create and submit
- Approvers should not be able to edit content in the same workflow step they approve
- Admins with elevated access need the most scrutiny, not the least

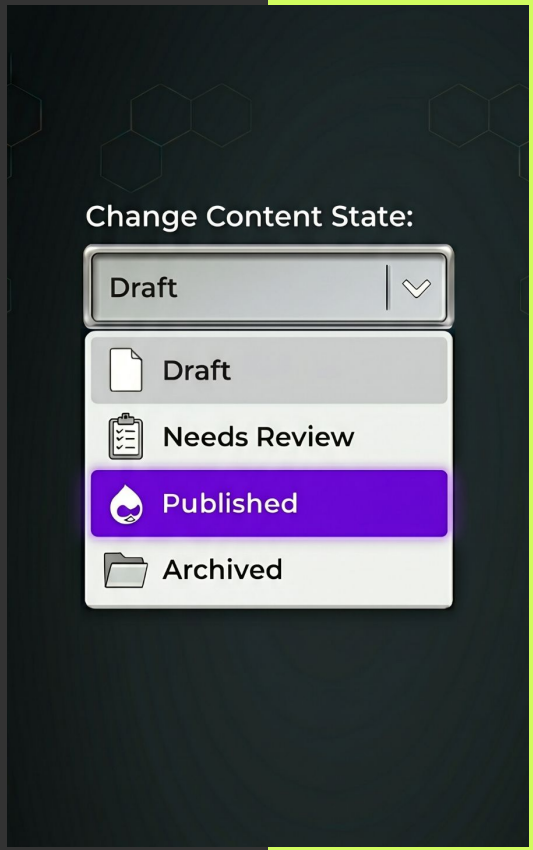
When It Goes Wrong - The Stakes

- Unauthorised disclosure of embargoed policy announcements
- Incorrect health or safety information published and shared widely before correction
- Reputational damage to the agency and the platform team
- Potential legal liability
- Loss of confidence in digital services

The key takeaway

- people need content moderation
- it is non-negotiable

Content Moderation in Drupal



Content Moderation Is a Core Module

- Available in Drupal core since Drupal 8.4 - no contrib module required
- Works in conjunction with the Workflows module (also core)
- Allows content entities to move through defined states via defined transitions
- Built with flexibility in mind - configurable without writing code

States, Transitions and Workflows

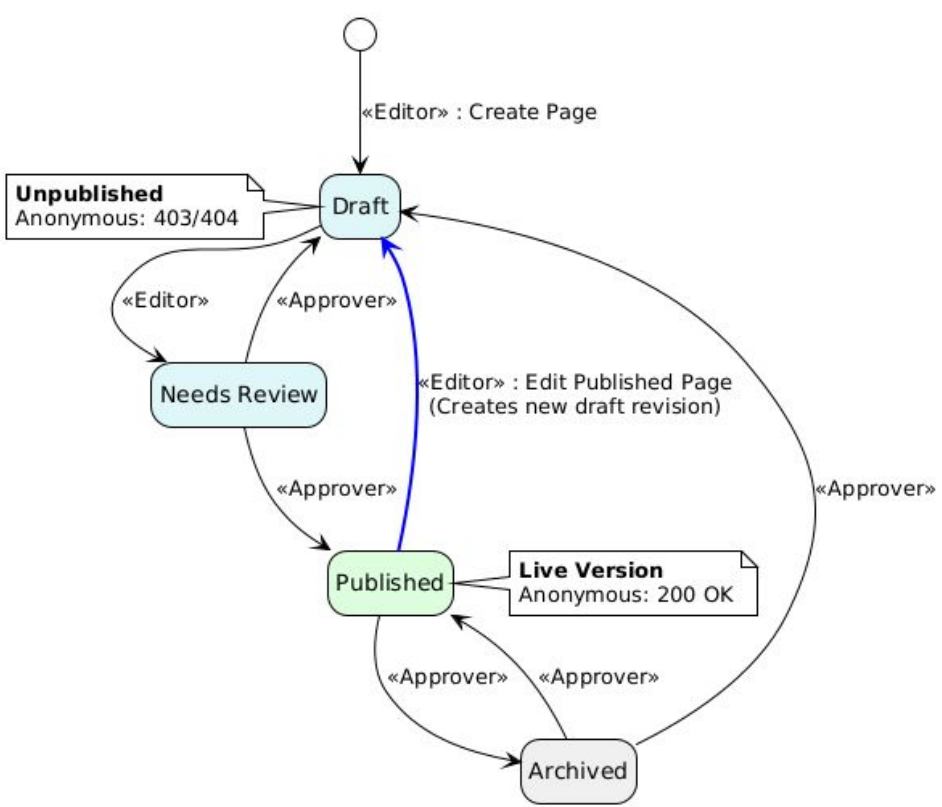
- States are the positions content can be in: Draft, Needs Review, Published, Archived
- Transitions are the allowed movements between states: "Submit for Review", "Publish", "Reject"
- Workflows bundle states and transitions together and are assigned to content types
- Multiple workflows can exist - e.g. different rules for News articles vs. Policy pages

Roles & Permissions
Where the Complexity Lives

- Drupal roles (e.g. Editor, Approver, Administrator) are assigned to users
- Each transition permission is granted to one or more roles
- View, Edit and Transition permissions are separate and must all be correct
- The matrix of roles × states × transitions grows quickly

A Realistic Workflow in a GovCMS website

- Content Type: Page
- Roles: Editor, Approver
- States: Draft → Needs Review → Published → Archived
- Each state has visibility rules
- Each transition is locked to a specific role



Color	Visibility	Meaning
#E0F7FA	**Private**	Work in progress (Draft/Review)
#DDFFDD	**Public**	Currently visible to the world
#F0F0F0	**Private**	Archived/Hidden from public

▬▬▶	**Revision**	Published content remains static while Draft is updated

Common Problematic Areas: Technical and Social

Technical Pitfalls:
Permission Gaps and Bypasses

- Role over-provisioning
- Missing view permissions
- Transition mismatches
- Node access conflicts
- Default state errors

Technical Pitfalls: Configuration Drift

- Workflows degrade over time
- New roles added not having constraint
- Change of behaviour after contrib update
- Staff turnover / configuration intent lost
- Unnoticed change in moderation logic

Technical Challenge

- Define (discovery)
- Implement
- Test it

Social Pitfalls: Process Without Enforcement

- Documented workflow policy != enforced workflow
- If the CMS allows bypassing the process, someone eventually will
- Training alone doesn’t prevent deliberate workarounds

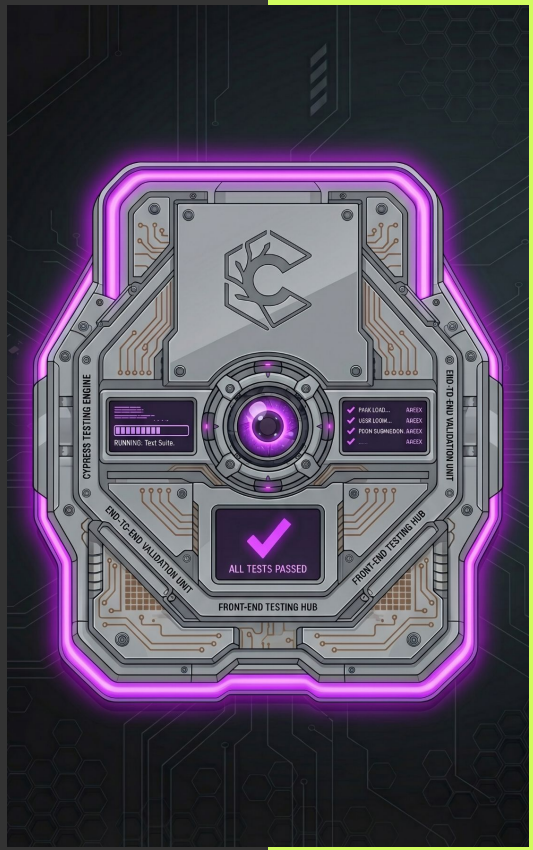
Social Challenge

- The web team wants to be the author and the reviewer
  → If they DO have the two roles, they can be the one rogue.

- People want to "fast forward"
  → want to go from Draft to Publish - 1 rogue
  → they want to go from published to published

- You should tell people we don't want to set it up like that, but we can.

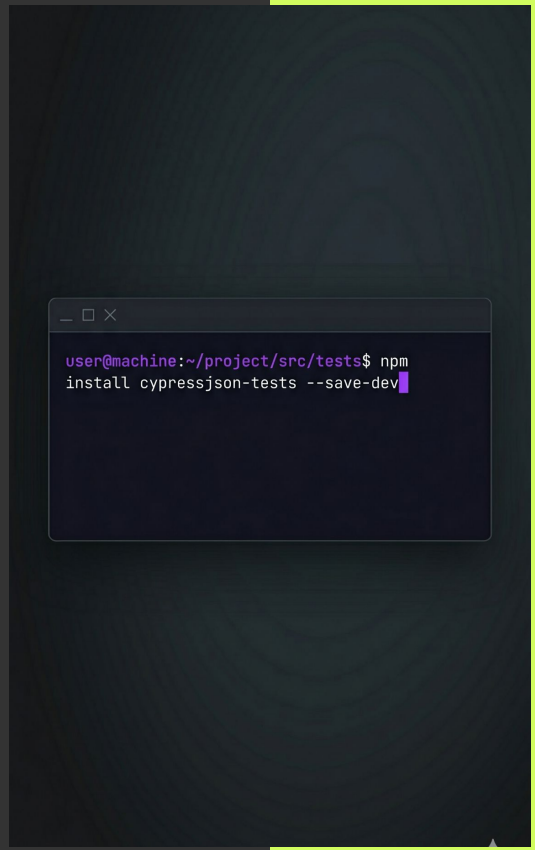
Cypress


CYPRESS TESTING ENGINE
END-TO-END VALIDATION UNIT
RUNNING: Test Suite.
ALL TESTS PASSED
FRONT-END TESTING HUB
END-TO-END VALIDATION UNIT
FRONT-END TESTING HUB

Cypress?

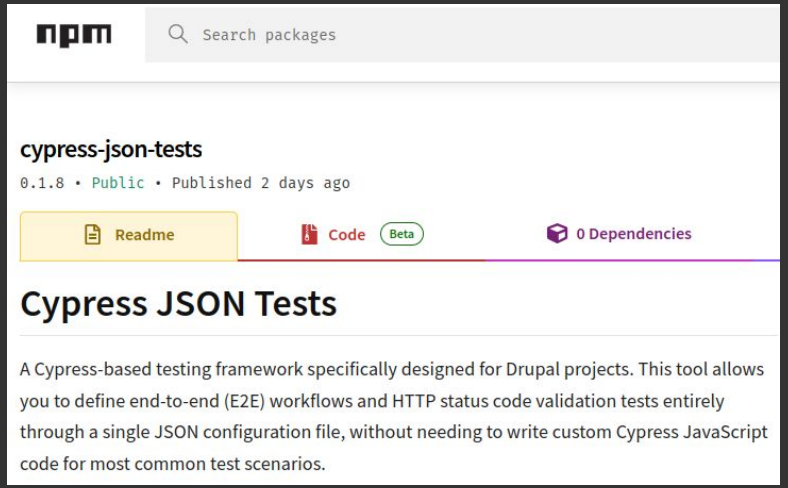
- A modern, front-end testing tool built specifically for the web.
- Operates directly inside the browser alongside your application.
- I.e. has access to the DOM, network requests, local storage, cookies, etc.

The Cypress-JSON-Tests helper library



Cypress-JSON-Tests

- Npm package: cypress-json-tests
- github.com/morpht/cypress-json-tests



Cypress-JSON-Tests

Objective

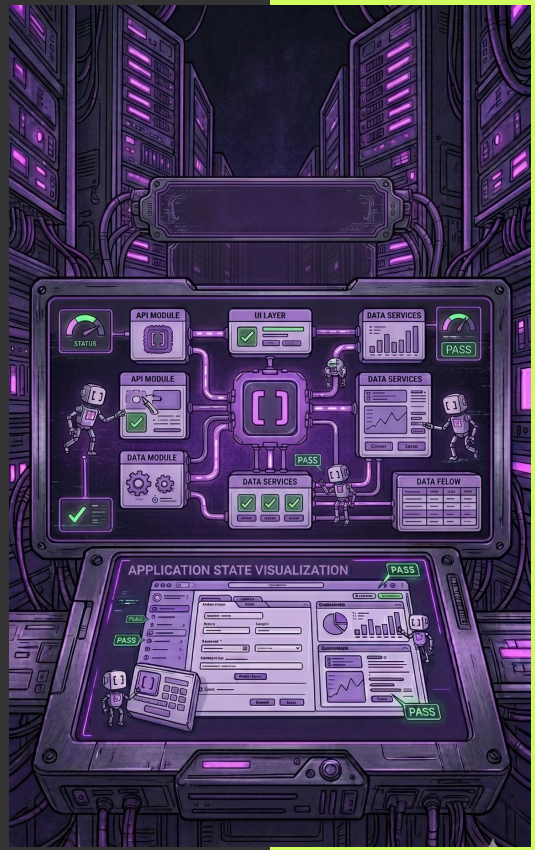
- Be able to quickly add cypress tests to any project
- One command to pull it in
- One config file to get it ready

Cypress-JSON-Tests

Features:

- JSON-Driven Testing
- Drupal Integration
- Status Code Validation
- UI Workflow Validation

DEMO


API MODULE
UI LAYER
DATA SERVICES
STATUS
PASS
API MODULE
DATA SERVICES
DATA MODULE
PASS
DATA SERVICES
DATA FELOW
APPLICATION STATE VISUALIZATION
PASS
PASS
PASS

Demo: existing project

```
npx cypress open --config
   "fixturesFolder=
   ../fixtures,baseUrl=https://convivial-govcms.lndo.site"
   --env env=lando
```

Demo: how to start

```
$ npm install cypress-json-tests --save-dev
$ vim fixtures/config.json # use readme
$ npx cypress open --config
   "fixturesFolder=
   ../../fixtures,baseUrl=https://cnvl.ddev.site"
   --env env=ddev
```

Demo: how to start

Without GUI:

```
$ npx cypress run --config
   "fixturesFolder=
   ../../fixtures,baseUrl=https://cnvl.ddev.site"
   --env env=ddev
   --spec=cypress/e2e/statuses.cy.js
```

Demo: was it fast enough?

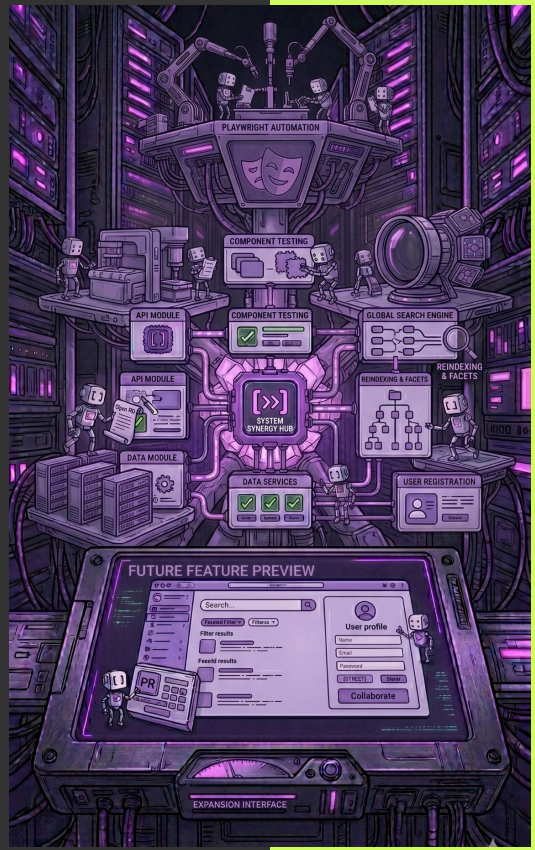
- Copy config.json and it’s ready to go
- Worfklows are often similar, 75% done
- Complex tests can be done in a day

Demo: code

Walk through

- config.json
- cypress/support/commands.js
- Package README

Possible future



Run in CI

cypress_tests.yml

```
  - name: Cypress run
    uses: cypress-io/github-action@v6
    with:
      working-directory: tests
      browser: chrome
```

See my DrupalCon Vienna talk:
“Github Actions + Docker + Cypress = CI/CD Nirvana”

Ideas for the future

- Rewrite this tool for Playwright
- Webform submissions
- Component testing
- Search results
- Search reindex and facets tests
- User registration
- Open to RPs

THANKS

Morpht

morpht.com