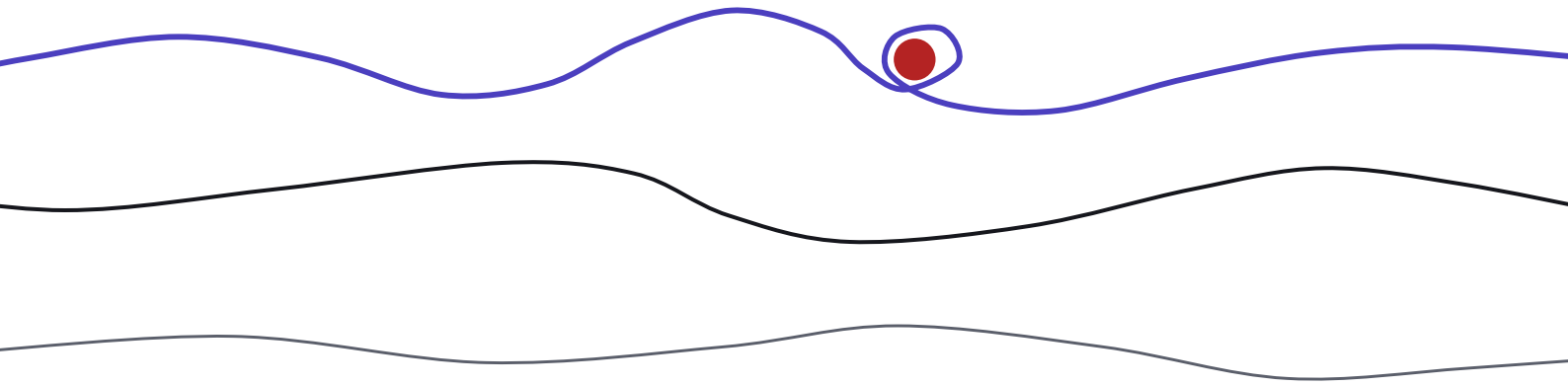
DOCUMENTATION

jupiter.au

Design system

Tokens, components, graphics, fonts and LaTeX classes that every jupiter.au project inherits.



VERSION

v1.5.11

DATE

25 November 2026

AUTHOR

belikh

SOURCE

github.com/belikh/design

Contents

Use

01 **Depending on the design system** 2

01.1 Web / npm projects 2

01.2 Releasing a change 3

01.3 Fonts 3

01.4 Other targets (no npm mechanism) 3

01.5 GitHub (READMEs, profiles, social preview) 3

01.6 Agents 3

Build

02 **Page recipes** 4

02.1 Data 5

02.2 Look and feel 5

SECTION

Use

Depend on the design system from a project.

01 Depending on the design system

How a project takes the design system as a dependency, and how a change reaches every project.

Projects **depend** on the design system; they never copy it. A change to a design constant (`packages/tokens/jupiter.tokens.json`) or a component (`packages/ui`) is released once, and every project picks it up with one update command.

01.1 Web / npm projects

Packages (private, GitHub Packages): `@belikh/jupiter-tokens` (generated CSS and constants), `-graphics` (icons and illustrations), `-ui` (React components, preferences, mock data layer), `-fonts` (the licensed typefaces, private), `-esphome` (display widgets). They are named the same here and in the monorepo, so imports are identical everywhere.

BASH

```
node scripts/new-project.mjs ../my-project         # from a design-system checkout
cd ../my-project
export NODE_AUTH_TOKEN=$(gh auth token)           # token needs read:packages (gh
auth refresh -s read:packages)
npm install && npm run dev
npm run update:design                              # later: take the newest
design-system release
```

The generated `.npmrc` maps the `@belikh` scope to `https://npm.pkg.github.com` and reads the token from the environment. In CI, set `NODE_AUTH_TOKEN` from a secret with read access to the packages.

Why real names rather than aliases such as `@jupiter/ui -> npm:@belikh/jupiter-ui`: npm 12 refuses registry installs through aliases (`allow-remote`), and GitHub Packages requires the scope to be the repo owner.

01.2

Releasing a change

Change the token or component here, then `node scripts/bump-version.mjs X.Y.Z && npm run build && npm run lint && npm test && npm run check:consumer`, push `main`. The release workflow publishes every package at that version (skipping versions already published) and attaches the generated files to a GitHub release. `npm run check:consumer` proves the loop: it changes constants, rebuilds `examples/consumer`, checks the new values arrived, and restores.

01.3

Fonts

SGI Text and SGI Alt are Bitstream typefaces used with the owner's consent. The woff2 files are committed in `apps/admin/public/fonts`, and every release carries them as `jupiter-fonts.tgz` (the `@belikh/jupiter-fonts` package: the files and a `fonts.css`). `node scripts/stage-fonts.mjs` copies them into the gitignored `packages/fonts/files` before the package is packed. Projects import `@belikh/jupiter-fonts/fonts.css`, or unpack `jupiter-fonts.tgz` at build time as docs.jupiter.au does; without the files they fall back to free fonts.

01.4

Other targets (no npm mechanism)

Rust, Go, Slint, GTK, Qt, KDE, LaTeX, ESPHome YAML, email, terminal and mobile consume generated files from the release. Fetch them into a gitignored directory at build time (`gh release download --repo belikh/design --pattern <file> --dir <build-dir> --clobber`); never commit them. The list of files per target is in `packages/ui/skill/SKILL.md` and `README.md`.

01.5

GitHub (READMEs, profiles, social preview)

`jupiter-github.tgz` (also `@belikh/jupiter-github`) carries light and dark SVGs generated from the tokens: banner, divider, stat tile, status badges and the 1280x640 social preview, plus README, profile (`<user>/<user>`) and org profile (`.github/profile/README.md`) templates. Make a project's own with `node make.mjs --title "My project" --subtitle "What it does" --out .github/assets` (add `--logo` for the Jupiter logo, `--badge "build|passing|ok"`, `--tile "Uptime|99.98%|ok"`, `--png` for the social preview, `--palette` for a hex-chip colour table), then reference them with `<picture>` as the templates do. The social preview has no upload API: set it by hand under Settings > Social preview. Regenerate after a token change; nothing is hand-drawn.

A website's own pages get **share cards** from the same kit: the 1200x630 image LinkedIn, Slack and others show when a page is linked. List the pages in a JSON file and run `jupiter-github --cards cards.json --scheme light --png --out <dir>` (one card per entry: `name`, `title`, `subtitle`, `eyebrow`, `footer`, and `art` for one graphic), then point each page's `og:image` at its PNG. Regenerate them after a design-system update, like every other baked file. See the Share cards section of `packages/github/README.md`.

01.6

Agents

`packages/ui/skill/SKILL.md` is the skill development agents use before building anything. New projects get it linked from `node_modules/@belikh/jupiter-ui/skill`, so it updates with the dependency. In this repo it is linked at `.claude/skills/jupiter-design-system`.

SECTION

Build

Page shapes and components for apps.

02

Page recipes

The five page shapes (list, detail, form, dashboard, kiosk) and the scaffold that generates each.

Five page shapes cover almost everything. Generate one, then change the content, not the structure:

BASH

```
npm run new:page -w apps/admin -- <area> <PageName> "<Label>"
list|detail|form|dashboard|kiosk
```

It creates the component, a test and the route. Run `npm test` and `npm run lint:style -w apps/admin` after.

RECIPE	USE IT FOR	BUILT FROM	RULES THAT MATTER
list	Many records: browse, filter, edit, bulk edit	`PageHeader` → `Panel` → `DataTable` with `cols` (editable columns get an `editor`), `onCommit` from `useCollection`	Edit in place, never a separate edit page. Let `DataTable` do bulk edit, paste, fill down, peek, compare
detail	One record	A column of `Editable` fields in a `Panel`; the title is an `Editable` too	No Edit button. Add `showHistory` where "who changed this" matters
form	Creating or configuring something	`Field` + inputs, `useFieldErrors` (validate on blur, summary on submit), `useDraft`, `useUnsavedGuard`	Settings pages autosave (`useAutosave`). Long forms keep a draft. Advanced fields go in a `Disclosure`

RECIPE	USE IT FOR	BUILT FROM	RULES THAT MATTER
dashboard	The numbers that matter	`StatTile` row, then `LineChart` and friends in `Panel`s	Colours come from `--adm-c1..c8` (colour-vision safe); red and green mean status only; every chart has a text alternative
kiosk	Wall displays and touch terminals	A `bare: true` route, big targets (72px+), `100dvh`, no hover-only controls	Works offline; large type from `--jp-fs-xl/2xl`; test with the touch-mode toggle

02.1

Data

Every page reads from a `createCollection` (`@belikh/jupiter-ui/mock.ts`) through `useCollection`, so saves are optimistic, work offline, detect conflicts and go stale with a Refresh. Show that state with `SyncStatus` and `ConflictPrompt`. There is no real backend: replace `mock.ts` when there is one, and the pages do not change.

02.2

Look and feel

Tokens only (`var(--jp-*)`), type from `--jp-fs-*`, logical CSS properties, `Modal/Drawer` for overlays (native `<dialog>`), `Menu` for action lists (popover), an Undo toast for reversible actions. See `AGENTS.md` for the full rules and the component table.