Storyblok + Next.js (App Router) POC – Teaching Notes

Goal

CMS-driven marketing site using Next.js App Router + TypeScript + Storyblok rendered via React Server Components.

Key Ideas

- Next route `app/[[...slug]]/page.tsx` maps URLs to Storyblok story slugs.
- Storyblok stories return a `content` object with a `component` field (ex: "page").
- `StoryblokServerComponent` reads `blok.component` and renders the matching React component from the registry.

Routing

- `/` -> Storyblok story slug `home`
- `/faq` -> `faq`
- `/contact` -> `contact` Catch-all params come in as arrays and are joined with `/`.

Draft vs Published

- Development uses `draft` so we can iterate without publishing.
- Production uses `published` so only approved content is visible. Common failure: 404 when story exists but is not published.

RSC (React Server Components)

- We render blocks server-side using `@storyblok/react/rsc`.
- Blocks are server components (no `"use client"`), so no hydration required for content.

Block Rendering

- `page` is the root story component.
- It renders `body` blocks via a small `Blocks` helper to avoid repeating `.map()` logic.

Caching / ISR

- `export const revalidate = 60` keeps pages fast but refreshes content periodically without redeploying.

Extending Blocks

To add a new block:

1. Create component in Storyblok UI
2. Create React component in `/components`

3. Register it in `storyblokInit({ components: { ... } })`
4. Use it inside `page.body` or nested blocks