FASTR

A Practical Checklist of 20 Frontend Tasks That Shouldn’t Go Through Engineering

Built for teams under pressure to move faster – without breaking governance

If your team needs developers to change a homepage banner, launch a campaign landing page, or update localized content, you’re not just moving slowly – you’re operating at a structural disadvantage.

This isn’t a talent problem. It’s a workflow problem.

At enterprise scale, every unnecessary dependency on engineering introduces friction: longer cycles, higher costs, missed revenue windows, and burned-out teams. Meanwhile, your competitors are shipping faster because they’ve decoupled business execution from engineering innovation.

This checklist is designed to be uncomfortable in the best way.

If any of the tasks below still require dev tickets, sprint planning, or release coordination on your team, you’ve found a concrete opportunity to move faster – without replatforming, without rewriting your stack, and without adding yet another tool.

Campaign & Merchandising Tasks

High-velocity teams ship these daily. No pull requests required.

- [ ] Launch or swap homepage promo banners
- [ ] Create campaign-specific landing pages (seasonal, category, brand, or promo-driven)
- [ ] Update homepage modules for sales events (hero, feature blocks, urgency messaging)
- [ ] Reorder or spotlight collections and categories
- [ ] Spin up short-lived campaign experiences without touching core templates

If this requires engineering, campaigns become calendar-driven instead of opportunity-driven.

Personalization & Experimentation

If testing needs dev, you won’t test enough.

- [ ] Create audience-based content variants
- [ ] Adjust personalization rules or logic
- [ ] Launch A/B or multivariate tests
- [ ] Iterate on underperforming experiences in-flight

Teams that depend on engineering test cautiously. Teams with control test aggressively.

Multi-Region & Brand Variants

Global scale demands local autonomy.

- [ ] Localize content by region, market, or language
- [ ] Manage brand or regional experience variations without forked codebases

Centralized dev control doesn’t scale globally. Distributed execution does.

UX & Layout Changes

These are experience optimizations, not engineering initiatives.

- [ ] Reorder page sections or modules
- [ ] Swap components (carousel → grid, editorial → product-driven)
- [ ] Adjust PLP layouts (tile density, hierarchy, module placement)
- [ ] Configure filters and sort logic without custom code
- [ ] Update navigation structures or featured paths

When UX iteration waits for sprints, optimization dies quietly.

Content & Creative Updates

Content velocity directly impacts revenue velocity.

- [ ] Update copy across pages or modules
- [ ] Swap imagery or creative assets
- [ ] Launch seasonal content modules (gift guides, trend edits, collections)
- [ ] Publish editorial or brand storytelling blocks

Every delay here compounds: creative loses relevance, campaigns lose impact.

No Dev ≠ Chaos.

It Equals Speed.

Why These Tasks Shouldn't Require Developers

This isn't about cutting engineering out of the process. It's about protecting their time.
The cost of the old model:

- Costs spike (highly paid resources doing low-leverage work)
- Roadmaps slip (innovation waits behind banners and copy changes)
- Teams slow down (marketing and merchandising operate on engineering timelines)
- Revenue windows close

What high-velocity teams do instead:

- Engineering builds systems, performance, and architecture
- Business teams own execution, iteration, and experimentation

Anything else is organizational drag.

How High-Velocity Teams Actually Operate

What we consistently see with leading enterprise brands:

- Marketing and merchandising ship daily, not quarterly
- UX teams iterate without waiting on sprints
- Personalization and testing are continuous, not precious
- Global teams move independently – without breaking governance

At enterprise scale, speed isn't a tactic. It's an operating model choice.

They don't achieve this by adding more tools or headcount. They achieve it by using an AI-native frontend layer that gives business teams direct control over the experience – without replatforming or fragmenting their stack.

That's the shift: control without chaos. Speed without risk.

The Takeaway

Your engineering team should be building what only they can build.

If they're still shipping banners, swapping components, or launching campaign pages, you're paying a premium for the wrong work – and slowing everyone down in the process.

The fastest teams don't work harder.

They remove friction.

And they stop routing everyday frontend execution through engineering.

Want to see how much this is costing you?

We'll walk through your site, your workflows, and where engineering is still acting as a bottleneck — and show what changes when business teams control execution.

No pitch. Just a practical conversation.

See What This Looks Like on Your Site