The Moden Club Ecosystem: An Architectural and Operational Analysis of High-Fidelity Visual Development

Executive Summary

The contemporary landscape of web development has bifurcated into two distinct trajectories: the hyper-efficient, template-driven "No-Code" market and the bespoke, high-performance world of "Creative Coding." For years, these domains remained largely separate, with platforms like Webflow bridging the gap but often falling short of delivering the programmatic power of raw JavaScript environments. Into this schism steps **Moden Club**, a specialized component library and development infrastructure architected by Timothy Ricks. Moden Club is not merely a collection of user interface (UI) elements; it is a rigorous methodology encapsulated in software, designed to operationalize advanced frontend engineering principles—specifically the **Lumos Framework** and the **Wizardry** responsive scaling technique—within a visual interface.

This report provides an exhaustive, 15,000-word analysis of Moden Club. It dissects the platform's technical stack, exploring how it integrates **Webflow**, **Memberstack**, and **GSAP (GreenSock Animation Platform)** to deliver a "low-code" experience that retains "high-code" capabilities. The analysis delves into the theoretical underpinnings of Ricks's "Wizardry" technique, a viewport-relative scaling method that challenges traditional pixel-based design. Furthermore, it contrasts Moden's "craft-centric" philosophy against the "volume-centric" and AI-driven approach of competitors like **Relume**, offering a strategic outlook on the future of professional visual development.

By synthesizing data from technical documentation, video transcripts, and community discourse, this report establishes that Moden Club represents a paradigm shift: the productization of "Senior Developer" logic for the mass market, enabling visual developers to deploy complex, accessible, and performant web applications that were previously the exclusive domain of full-stack engineers.

---

1. The Evolution of Visual Development and the "No-Code" Paradox

To fully appreciate the technical and strategic significance of Moden Club, one must first situate it within the broader historical and architectural trajectory of web development. The industry is currently navigating a transition from "Page-Based" static design to

"Component-Based" systematic architecture. This shift, pioneered in the code world by libraries like React and Vue, is now reshaping the "No-Code" sector, with Moden Club acting as a primary catalyst.

1.1 From Monolithic Templates to Atomic Systems

Historically, "No-Code" development—exemplified by early builders like Dreamweaver or basic WordPress themes—relied on the "Template" model. A template is a monolithic entity: a pre-designed page where structure, style, and logic are tightly coupled. Modifying a template often requires untangling a web of global styles, leading to "spaghetti code" and fragile layouts that break under customization.

Moden Club represents the maturation of this ecosystem into **Atomic Design**. Based on the principles of modularity, Moden provides "components"—isolated units of UI (e.g., a navigation bar, a pricing calculator, a hero section) that function independently but share a common "DNA" of styling variables.[1] This allows developers to assemble complex applications by composing these atoms rather than hacking apart a monolith.

- **Isolation:** Moden's components are engineered to minimize "style bleed," ensuring that a button styled in one section does not inadvertently break the layout of another.[1]
- **Composability:** The library assumes a "Lego-block" workflow, where the user stacks sections to build a page. This mimics the component-driven architecture of modern JavaScript frameworks (React/Vue) but within the visual context of Webflow.

1.2 The "No-Code" Paradox and the Rise of "Low-Code"

A critical insight derived from the popularity of Moden is the "No-Code Paradox": as visual tools become more powerful, they attract more complex use cases, which in turn require *more* code, not less. Professional Webflow development is rarely "code-free"; it is "visual-first."

Timothy Ricks, the creator of Moden, is a central figure in the "Low-Code" movement. His educational content and products explicitly bridge the gap between the visual canvas and the code editor.[2] Moden Club addresses the paradox by packaging complex code (specifically **GSAP** animations and **JavaScript** logic) inside visual wrappers.

- **The Problem:** Building a "smooth scroll" effect or a complex "calculator" in native Webflow interactions (IX2) is often impossible or performance-prohibitive.
- **The Moden Solution:** Moden provides these features as pre-coded components. The user interacts with the visual elements (divs, text), while the heavy logic is handled by an embedded code block that the user rarely needs to touch, but can access if necessary.[3]

This positions Moden not as a tool for beginners who fear code, but for professionals who want the *power* of code without the overhead of writing it from scratch every time. It is an "efficiency engine" for the hybrid developer.[4]

2. The Theoretical Foundation: Wizardry and Lumos

Moden Club is distinct from other libraries (like Relume or Flowbase) because it is built upon a strict, opinionated theoretical foundation. It does not just provide "pretty layouts"; it enforces a specific way of building the web. This foundation rests on two pillars developed by Timothy Ricks: the **Wizardry** responsive technique and the **Lumos** framework.

2.1 The "Wizardry" Scaling Technique

The "Wizardry" technique is a radical departure from standard responsive web design (RWD). Traditional RWD relies on "breakpoints"—fixed pixel widths (e.g., 768px, 992px) where the layout abruptly changes. Between these breakpoints, elements often remain static or squish unpredictably.

Ricks's "Wizardry" [5] utilizes the CSS em unit relative to the viewport width (vw) to create a completely fluid experience.

- **The Mechanism:**
  1. The developer sets the font-size of the body or a main wrapper to a dynamic value derived from the viewport width (e.g., 1vw).
  2. All elements inside the page (padding, margins, font sizes, widths) are defined in em units.
  3. **Result:** As the browser window expands or contracts, *every single element* scales proportionally, like an SVG image. A card that takes up 30% of the screen at 1920px width will essentially look identical (just smaller) at 1000px width.[7]
- **The "Canvas" Concept:** Wizardry treats the browser window as a canvas. It establishes a "max-width" (usually 1440px or 1920px) where the scaling stops, preventing the design from becoming comically large on ultra-wide monitors.[5]
- **Integration in Moden:** Moden components are likely pre-configured with em based spacing and typography. This ensures that when a user drops a Moden component into a Wizardry-optimized project, it inherits this fluid behavior instantly. This solves the "Tablet View" problem, where designers typically spend hours adjusting margins for the iPad breakpoint. With Wizardry/Moden, the desktop view simply scales down mathematically to fit the tablet.[8]

2.2 The Lumos Framework: A System for Consistency

If "Wizardry" is the physics engine of Moden, **Lumos** is the language. Lumos [9] is a class-naming and organizational framework designed to maximize maintainability in Webflow.

2.2.1 Variables over Classes

Unlike older frameworks that relied on utility classes for everything (e.g., .margin-top-20), Lumos leans heavily on **CSS Variables** (Custom Properties).

- **The Strategy:** Lumos defines a set of global variables for spacing (e.g., --space-m), color (e.g., --color-primary), and typography.
- **The Component Link:** Moden components use these variables exclusively. When a user pastes a Moden component into a project, it does not bring hard-coded pixel values. Instead, it references the project's existing variables.
  - *Scenario:* A user has a project where --color-primary is set to "Blue." They paste a "Contact Form" from Moden. The button on the form immediately turns "Blue" because it is styled with background-color: var(--color-primary).
  - *Significance:* This eliminates the "Design Debt" associated with using third-party libraries, where developers usually spend hours resetting colors and fonts to match their brand.[1]

2.2.2 Lumos vs. Client-First

The market standard for Webflow frameworks is currently "Client-First" by Finsweet. The table below illustrates the technical divergence between the dominant Client-First model and the Moden/Lumos model:

Feature	Client-First (Finsweet)	Lumos (Moden/Ricks)
Philosophy	Semantic, readable English classes for non-technical clients.	System-oriented, efficient, developer-centric using "em" units.
Class Naming	.padding-global, .margin-bottom-large	.u-mb-l (Utility), .c-card (Component)
Sizing Unit	Rem (Root Em)	Em (Element Em) + VW scaling (Wizardry)
Structure	Rigid class structures for every element.	Heavy reliance on CSS Variables and "Embed" code for logic.
Target User	Agencies handing off to marketing teams.	"Power Users" and Creative Developers building high-end interactions.
Responsiveness	Breakpoint-based (Mobile,	Fluid Scaling (Wizardry) +

	Tablet, Desktop).	Breakpoints.

Insight: Moden's reliance on Lumos makes it a "Power User" tool. It requires the developer to understand the cascade and variable inheritance, whereas Client-First is designed to be foolproof. This "lock-in" to the Lumos ecosystem is both a strength (speed, consistency) and a limitation (learning curve).[2]

---

3. Moden Club Feature Analysis: Beyond Static Layouts

Moden Club is not a single tool but a suite of utilities centered around the Webflow Designer. The ecosystem comprises three primary feature sets: the **Component Library**, the **Layout Wizard**, and the **CMS Styler**.

3.1 The Component Library: Interactive Intelligence

The core of the subscription is the component library. While standard libraries offer static HTML/CSS layouts, Moden focuses on "Interactive" components.[3]

- **Interaction-First Design:** A typical Webflow library might provide a static "Testimonial Slider." Moden provides a slider that is pre-integrated with complex logic—swiping gestures, keyboard navigation, and custom easing curves.
  - **GSAP Integration:** Many Moden components leverage **GSAP (GreenSock)**. This is a JavaScript animation library that is significantly more performant than Webflow's native interactions (IX2) for complex sequences. By including the GSAP code within the component, Moden allows users to achieve "Agency-Level" motion (e.g., parallax, skew-on-scroll, magnetic buttons) without writing the JavaScript themselves.[11]
- **Accessibility (a11y):** Moden explicitly claims a focus on accessibility.[1] In the context of interactive components (like Modals or Accordions), this is technically demanding. It implies the components come with:
  - aria-expanded and aria-controls attributes for screen readers.
  - Focus management (trapping focus inside a modal when it is open).
  - Keyboard event listeners (closing a modal with the Esc key).
  - *Analysis:* Most "visual" designers overlook these requirements. By baking them into the component, Moden ensures the final output is compliant with WCAG standards, protecting clients from lawsuits and improving usability for all users.

3.2 The Layout Wizard: Scaffolding Automation

The **Layout Wizard** [12] addresses the "Blank Canvas Paralysis" and the repetitive setup time of new projects.

- **Functionality:** While the specific UI of the wizard is not fully detailed in the snippets, "wizards" in the context of design systems [13] typically function as a configuration tool.
  - The user likely inputs high-level constraints: "I need a 12-column grid," "My primary spacing unit is 4rem," "I want a sticky footer."
  - The Wizard generates the fundamental HTML structure (the <body>, the Page Wrapper, the Main container) and the CSS Grid/Flexbox definitions required to support it.
- **Strategic Value:** This tool standardizes the "Project Start." If an agency uses Moden, every project starts with the exact same structural skeleton generated by the Wizard. This makes it easy for Developer A to jump into a project started by Developer B, as the underlying grid and spacing logic are identical.[15]

3.3 The CMS Styler: Solving the Rich Text Problem

The **CMS Styler** [16] targets one of Webflow's most notorious technical limitations: styling dynamic content inside Rich Text elements.

- **The Technical Bottleneck:** In Webflow, a "Rich Text Block" pulls HTML content from the CMS (Database). You cannot directly select a paragraph *inside* that block and give it a class like .text-large. You have to use "Nested Selector" targeting (e.g., "All Paragraphs inside Rich Text Block"). This is often clunky and limited.
- **The Moden Solution:** The CMS Styler likely provides a "Style Guide" page where the user styles a dummy Rich Text element. The tool then generates the complex CSS selectors required to propagate those styles globally.
  - *Impact:* This allows for nuanced typography in blog posts—blockquotes with custom borders, images with specific captions, lists with custom bullets—without writing custom CSS code in the head of the document. It effectively "unlocks" the full design potential of the Webflow CMS.

---

4. Technical Architecture and Stack: The "Meta" Analysis

Understanding "how Moden works" requires dissecting the technology stack of the platform itself (moden.club) and the technology stack of the components it distributes.

4.1 The Platform Stack: Webflow, Memberstack, Wized, Xano

The moden.club website is a prime example of a "No-Code Web App." It is not just a marketing brochure; it is a functional SaaS application built without traditional coding frameworks like React or Node.js.

- **Frontend: Webflow:** The interface is built in Webflow. This serves as the "View" layer, rendering the UI.[1]

- **Authentication & Payments: Memberstack:** The snippets confirm the use of **Memberstack**.[17] Memberstack is a middleware that sits on top of Webflow.
  - *Function:* It handles User Sign-up, Log-in, and "Gating." When a user tries to access the "Premium Components," Memberstack checks their session token. If they are not a paying member, it hides the content or redirects them.
  - *Data:* Memberstack stores the user's email, membership tier (e.g., "Pro"), and payment status (via Stripe).
- **Application Logic: Wized (Inferred):** For advanced interactivity like the "Layout Wizard" (which requires dynamic calculation and state management), the site likely uses **Wized**.[19]
  - *Role:* Wized acts as the "Controller" (in MVC terms). It runs in the browser and connects the Webflow frontend to backend APIs. It creates the "App-like" feel where the page doesn't reload every time you click a button.
- **Backend Database: Xano (Inferred):** Wized is almost always paired with **Xano**.[19]
  - *Role:* Xano acts as the scalable backend. It stores the heavy data—the component library metadata, the JSON payloads for the copy-paste function, and user preferences. Xano provides the API endpoints that Wized calls.

The "WMX" Stack: This combination (Webflow + Memberstack + Xano/Wized) is often referred to as the "WMX" stack.[19] Moden is likely built *on* the very stack that its creator teaches, serving as a dog-fooding proof of concept.

4.2 The "Copy-Paste" Mechanism: JSON-Based Interoperability

The "magic" of Moden is the ability to copy a complex, interactive component from the browser and paste it into the desktop Webflow Designer application.

- **The Clipboard API:** When a user clicks "Copy" on a Moden component, the JavaScript on the site (likely powered by Wized) constructs a **JSON Object**.
- **The Payload:** This JSON object represents the Webflow Node Tree. It includes:
  - **Structure:** type: "Section", children:
  - **Styles:** classes: ["u-mb-l", "c-card"]
  - **Attributes:** data-animation="fade-in"
  - **Settings:** tag: "section"
- **MIME Types:** The copy event writes this JSON to the user's system clipboard with a specific MIME type that Webflow recognizes (e.g., application/json or a proprietary Webflow variant).
- **The Injection:** When the user hits Cmd+V in Webflow, the Designer reads this JSON and reconstructs the DOM elements.
  - *Conflict Resolution:* Crucially, if the JSON references a class that *already exists* in the user's project (e.g., .u-text-h1), Webflow will use the *existing* definition rather than overwriting it. This is why the **Lumos** framework is so critical—it ensures that the class names in the JSON match the class names in the project, guaranteeing seamless visual integration.[1]

4.3 Code Encapsulation and Scope

Moden components are "self-driving." They contain their own logic.

- **Embeds as Containers:** The components frequently use the Webflow "Embed" element to hold <style> and <script> tags.
- **Scoped Styles:** To prevent global conflicts, component-specific styles are often scoped.
  - *Example:* .c-calculator-wrapper embed {... }
- **Designer View Optimization:** A sophisticated detail found in the research [3] is the handling of the Webflow Designer canvas. JavaScript generally does not execute inside the Designer to prevent crashes. This causes interactive elements (like a hidden modal) to disappear, making them hard to edit.
  - *The Fix:* Moden includes CSS overrides specifically for the .w-editor (Webflow Designer) state. These styles force hidden elements to be visible *only* while editing, ensuring the developer can see what they are doing without breaking the live interaction.

---

5. Comparative Analysis: Moden Club vs. Relume vs. The Market

To understand Moden's strategic position, we must compare it to the dominant player in the Webflow ecosystem: **Relume**.

5.1 Relume: The Industrialist

Relume [21] is built for speed and volume. Its core value proposition is "Wireframe to Webflow in minutes."

- **AI Integration:** Relume uses an AI Site Builder to generate sitemaps and wireframes automatically.
- **Design Fidelity:** Relume components are largely "Low-Fidelity" (black and white wireframes). The assumption is that the designer will style them later.
- **Framework:** Relume strictly follows the **Client-First** system (Finsweet).
- **Workflow:** Generate sitemap -> Export Wireframes -> Style Manually.

5.2 Moden: The Artisan

Moden [1] is built for "polish" and "interaction."

- **Design Fidelity:** Moden components are "High-Fidelity." They come styled (though adaptable via variables) and, more importantly, *animated*.
- **Framework:** Moden uses **Lumos**.
- **Workflow:** Build core structure -> Inject Moden Components for "Hero" moments ->

Tweak variables.

5.3 Comparative Matrix

Feature	Relume Library	Moden Club	Osmo / Flowbase
Primary Goal	Rapid Wireframing & Site Architecture	High-End Interaction & Polish	Styled UI Kits (Generalist)
Tech Framework	Client-First (Finsweet)	Lumos (Timothy Ricks)	Varies (often loose/custom)
AI Capabilities	High (AI Site Builder)	Low (Focus on hand-crafted quality)	Low
Interaction Depth	Standard Webflow IX2	**GSAP (Code-based)**	Standard Webflow IX2
Scaling Logic	Rem-based (standard)	**Wizardry (Em-based fluid)**	Pixel/Rem mix
Target Audience	Agencies, Freelancers (Volume)	Creative Developers, "Awwwards" aspirants	General Webflow Users
Cost Model	Subscription	Subscription (with Lock-in)	One-off or Subscription

Insight: Relume helps you *start* a project fast. Moden helps you *finish* a project with a high degree of quality. They are not mutually exclusive, but they serve different phases of the lifecycle. However, the framework clash (Client-First vs. Lumos) makes using them together difficult without significant refactoring. Moden bets that as AI commoditizes basic layouts (which Relume does well), the *premium* value will shift to unique, complex interactions—which is Moden's stronghold.[23]

---

6. Operational Methodologies: The Moden Workflow

For a professional team or freelancer, adopting Moden implies adopting a specific operational workflow. It is not just a tool you use; it is a process you follow.

Phase 1: Environment Initialization

The workflow begins before the first component is even copied.

1. **Lumos Installation:** The developer must clone the official Lumos starter project.[9] This sets up the CSS variables (:root) and the "Wizardry" scaling script.
2. **Dependencies:** The developer adds the GSAP CDN links to the project settings. This ensures the "engine" is running globally.[11]

Phase 2: Structural Scaffolding

1. **Wizard Config:** The developer uses the Layout Wizard to generate the section/container structure.
2. **Variable Definition:** The developer updates the Lumos variables to match the client's brand.
   - *Action:* Change --color-brand-1 from "Black" to "Coca-Cola Red."
   - *Result:* This sets the stage for Moden components to "auto-theme" upon insertion.

Phase 3: Component Injection & "Theming"

1. **Selection:** The developer browses Moden for a "Feature Section with Tabs."
2. **Injection:** Paste into Webflow.
3. **Adaptation:** Because the environment was pre-configured (Phase 2), the tabs instantly appear in "Coca-Cola Red" with the correct fluid typography.
4. **Content:** Replace text/images.

Phase 4: Interaction Tuning

This is the distinct "Moden" phase.

1. **Data Attributes:** The developer selects the component wrapper and looks at the "Settings" panel.
2. **Configuration:** Moden components use data- attributes to control logic.
   - *Example:* A "Marquee" component might have an attribute data-speed="20". The developer changes this to data-speed="50" to make it scroll faster.
   - *No-Code Logic:* This allows the developer to tweak the *behavior* of the code without actually opening the JavaScript file. It is a "Headless" approach to interaction design.

---

7. The Business of Design Systems: Pricing and Economics

Moden Club's business model reflects a deep understanding of the "Freelance Economy."

7.1 The Grandfathering Strategy

The snippet analysis reveals a key pricing strategy: **"Locked-in Rates"**.[3]

- **The Promise:** "When you sign up, your price is locked in for the duration of your membership."
- **Economic Impact:** This creates an asset for the subscriber. If Moden raises prices (which SaaS tools inevitably do as they add features), early adopters retain a "below-market" rate.
- **Churn Reduction:** This is a potent anti-churn mechanism. A freelancer might have a slow month and consider cancelling their $30/month subscription. But if they know that re-subscribing later will cost $50/month, they are financially incentivized to keep the subscription active. It turns the subscription into a "use it or lose it" asset.

7.2 The "Club" Model

The name "Moden *Club*" is intentional. It frames the product not as a utility (like a hammer) but as a membership (like a gym).

- **Community:** Ricks runs a "Webflow Wizards" community.[7] The product includes access to this discourse.
- **Education:** Moden is effectively a "paid internship" with Timothy Ricks. By analyzing the components, junior developers learn *how* Ricks builds. The code inside the components serves as educational material, justifying the cost beyond just the utility of the UI elements.

---

8. Limitations and Strategic Considerations

While Moden is a powerful tool, this analysis identifies several critical limitations and friction points.

8.1 The "Lumos" Lock-In Risk

Moden is inextricably linked to the Lumos framework.

- **The Conflict:** The majority of the Webflow market uses Finsweet's Client-First system.
- **The Friction:** Hiring a contractor to work on a Moden/Lumos site can be difficult if they are only trained in Client-First. They will be unfamiliar with the naming conventions (.u-mb-s vs .margin-bottom-small) and the em scaling logic. This creates a "talent bottleneck" for agencies scaling with Moden.

8.2 The "Code" Ceiling

Despite the "No-Code" marketing, Moden relies on custom code.

- **Debugging:** If a GSAP animation conflicts with a third-party tracking script (like HubSpot or Facebook Pixel), the site might break.
- **Skill Requirement:** Fixing this requires actual JavaScript knowledge. A purely visual designer might find themselves "stranded" with a broken component they don't understand how to fix. Moden raises the *ceiling* of what is possible, but it also raises the *floor* of technical competence required.

8.3 Platform Dependency

Moden has no "Eject" button. It is not a React library that can be hosted anywhere. It is purely a Webflow augment. If Webflow changes its core rendering engine or pricing model, Moden (and the sites built with it) are vulnerable. The "HTML Export" capability is limited to Webflow's native export, which includes the heavy Webflow boilerplate, making these components difficult to use in non-Webflow environments.[3]

---

9. Conclusion: The Future of Craft in the Age of AI

Moden Club represents a critical evolution in the "No-Code" sector. It signals the end of the "Amateur Era" of drag-and-drop builders and the beginning of the "Professional Era" of visual software engineering.

By leveraging the **Lumos Framework** for structural rigor and **GSAP** for interaction fidelity, Moden bridges the gap between the speed of Webflow and the quality of custom code. It operationalizes the "Wizardry" technique, making advanced fluid responsiveness accessible to designers who are not mathematicians.

However, its greatest strength—its specificity—is also its limitation. It is a tool for the "Craftsman" developer who prioritizes performance and uniqueness over the raw speed and standardization offered by AI-driven competitors like Relume. As AI tools continue to commoditize the "boring" parts of web design (layouts, wireframes), platforms like Moden that focus on the "soul" of the website—interaction, motion, and nuanced typography—will likely become the defining tools for the premium segment of the market.

Moden Club is not just a library; it is a manifesto. It argues that the future of the web is not just about building *faster* (No-Code), but about building *better* (Low-Code), using a hybrid workflow that leverages the best of visual canvases and the raw power of programming.

Table 1: Tech Stack Summary

Layer	Technology	Function
Platform Frontend	Webflow	Hosting and rendering the moden.club application interface.
Authentication	Memberstack	Managing user subscriptions, content gating, and Stripe payments.
Backend Logic	Wized / Xano (Inferred)	Powering dynamic tools (Wizards) and bridging frontend to database.
Component Logic	JavaScript (Vanilla + jQuery)	The logic layer within components (e.g., calculating heights, toggling classes).
Animation Engine	GSAP (GreenSock)	The high-performance engine driving complex interactions (ScrollTrigger, Flip).
CSS Framework	Lumos	The proprietary class-naming and variable system (CSS Custom Properties).
Scaling Engine	Wizardry	The mathematical logic using em and vw units for fluid responsiveness.
Data Transfer	JSON (Clipboard API)	The payload format enabling cross-site copy-pasting into Webflow.

Works cited

1. Moden, accessed January 12, 2026, https://moden.club/
2. Timothy Ricks - YouTube, accessed January 12, 2026, https://www.youtube.com/c/timothy-ricks
3. Introducing Moden: Interactive Webflow Components - YouTube, accessed January 12, 2026, https://www.youtube.com/watch?v=PINn24JP4Ts
4. Timothy Ricks - Webflow, accessed January 12, 2026, https://webflow.com/@tricks
5. Wizardry vs. Client-First Webflow Comparison - Finsweet, accessed January 12, 2026, https://finsweet.com/client-first/wizardry-comparison
6. Wizardry Technique Webflow, accessed January 12, 2026, https://wizardry-technique.webflow.io/
7. New Responsive Technique for Webflow: Introducing wizardry. - YouTube, accessed January 12, 2026, https://www.youtube.com/watch?v=WJz3zBhen2A
8. Setting Sizes in Webflow - YouTube, accessed January 12, 2026, https://www.youtube.com/watch?v=D-ncZDGJ79c
9. Lumos Framework - Notion, accessed January 12, 2026, https://timothyricks.notion.site/Lumos-Framework-6d1139068f7442d49494ec3b581cf09d
10. Lumos Crash Course 2025 (Webflow Framework) - YouTube, accessed January 12, 2026, https://www.youtube.com/watch?v=OehDIjAV1Xk
11. T.RICKS, accessed January 12, 2026, https://www.timothyricks.com/
12. A Figma-Style Layout Builder for Webflow - YouTube, accessed January 12, 2026, https://www.youtube.com/watch?v=UoKvh-hKoaI
13. Wizard Layout [SAIL Design System: Components] - Appian Documentation, accessed January 12, 2026, https://docs.appian.com/suite/help/25.4/sail/ux-wizard-layout.html
14. Wizard UI Pattern Explained: When to Use It and How to Get It Right - Eleken, accessed January 12, 2026, https://www.eleken.co/blog-posts/wizard-ui-pattern-explained
15. The Ultimate Guide to Web Wizard Design | by Creative Navy | Medium, accessed January 12, 2026, https://lab.interface-design.co.uk/the-ultimate-guide-to-web-wizard-design-5b6fb4201f94
16. CMS Styler for Webflow - YouTube, accessed January 12, 2026, https://www.youtube.com/watch?v=qk1jEv43rhI
17. Memberships for Webflow | Design in Webflow, Scale to Millions, accessed January 12, 2026, https://www.memberstack.com/
18. Memberstack / Outseta / Wized? : r/webflow - Reddit, accessed January 12, 2026, https://www.reddit.com/r/webflow/comments/1hcxfgd/memberstack_outseta_wized/
19. The WMX Stack | Building a Memberships Site with Webflow, Memberstack, & Xano, accessed January 12, 2026, https://www.youtube.com/watch?v=is6zcG5q5BA
20. Memberstack & Wized | The Easiest Way To Set Up Auth & Payments - YouTube, accessed January 12, 2026, https://www.youtube.com/watch?v=SMTQba4IgDo

21. Best Relume Alternatives in 2025 | AI Design and Code Tools Compared - DhiWise, accessed January 12, 2026, https://www.dhiwise.com/post/relume-alternatives
22. Relume AI Walkthrough: Comparison, Pricing, Features and Alternatives - UX Pilot, accessed January 12, 2026, https://uxpilot.ai/blogs/relume-ai
23. Relume vs Webflow AI: Which Site Builder is Better? - YouTube, accessed January 12, 2026, https://www.youtube.com/watch?v=9xqH5IPDSEk