The Applications of Website

Kaiwen Wang

December 12, 2025

Contents

Chapter 1

Introduction

This thesis is about technology, or more specifically, frontend design systems in Vue with a corollary on LLMs and AI, but we start with an analogy to set the pace.

In our day and age we are presented with myraid of choices; we may see numerous brands of fast foods: McDonalds, Burger King, Wendy's, or one might go to a supermarket and see countless options of foods and brands, or even further one might see numerous styles and brands of clothes but the underlying stitching and fabric material remains the same.

Because fast food restaurants must work under constraints such as the availability of foodstuffs, of which mass-produced vegetable oils and factory farmed chicken has the cheapest scale, each place is essentially indistinguishable from the other only in that they differ slightly in their appearance and pricing. Restaurants at the low end in effect are a wrapper around food wholesalers such as Costco Business Center.

Processed foods in a supermarket are a wrapper around bleached wheat as per USDA standards and vegetable oils: look at the ingredients list and the all too familiar "Enriched Flour (Wheat Flour, Niacin, Reduced Iron, Vitamin B1 [Thiamin Mononitrate], Vitamin B2 [Riboflavin], Folic Acid)" plus "Soybean Oil" appears. Any sense of difference is an illusion once the wrapper is closely scrutinized.

This tendency of seeming differentiation but implied homogeneity is characteristic of the modern times, and the ability to see past this false differentiation is necessary to not get caught in a vortex of confusion, for there lies infinite combination of items one can find themselves looking at and trapped. Likewise, infinite incremental improvements can always be made as well, but without a core which is

Instead, a sort of piercing ability to get at the core and essence of reality

is needed: to hack at the overgrown jungle and see the straight path ahead. A trusty tool to wield in this is writing a chronology of how things were, how they came to be, and the reasons and rationale for where it shall be going. By doing so, holes or gaps can be identified in the current paradigm, giving motive for action.

Many industries follow this pattern: growth and speciation as a frontier opens and consolidation as the frontier closes.

1.1 The Development of Technologies

All technologies are implemented by humans, who together through shared mutual understanding, are then able to create themselves or have others implement these mental abstractions. What we see on a computer is no different: the specification bodies are the W3C for CSS and accessibility, WHATWG for HTML, and Ecma International for Javascript. Google, Apple, and Mozilla then respectively implement the Blink, WebKit, and Gecko rendering engine.

Nevertheless, these specifications are rarely enough to satisfy two significant demands of frontend: reactivity and composability. Reactivity is the automatic response in appearance to the change in a variable or internal state while composability means the ability to break a website down into pieces so as to develop parts of the website without interfering in other parts in addition to reusing pieces.

Currently and seemingly for the near future, frontend web frameworks such as React, Vue, Svelte, Solid, and Angular remain predominant for website development to fulfill these two requirements well. To some extent Web Components and Lit may allow reactivity and composability, but their awareness and usage in the developer community is not as high and the extent of their ecosystems not as developed, so for the sake of brevity we can evaluate whether these options are suitable for development at another time.

At an even higher level include features such as modals, buttons, breadcrumbs, sidebars, tabs, and other components which are commonly tabulated in a 'design system' due to their frequent reuse across web apps. When components have no CSS to them, they are termed 'headless components'.

Styling these days is generally scoped to the component with Tailwind CSS which gives numerous small design tokens such as 'mt-2' as shorthand for "margin-top: 2rem," CSS Modules if one prefers writing native CSS, or it is scoped to the component using the framework. Vue or Svelte's Single-File Components do this. A solution that used to be more common in React but has since lost ground to Tailwind is CSS-in-JS with Emotion or styled-

components, as it tended to be somewhat verbose and messy syntax.

Finally, only after considering all these base layers, can a software developer then consider the relation of the frontend to the browser with full-stack frameworks such as Nuxt, Next.js, SvelteKit, client and server-side rendering, its relation to backend servers and databases, and fundamentally the software's relation to the real world and its business uses.

1.1.1 Repeatable Subunits

As humans move up the technology tree, things which were previously novel become repeatable subunits for the next layer. For example, there is little more understanding needed to innovate screws—most shapes exist and engineering projects select from an existing set of them. Electricity and steel become commonplace. To not 'reinvent the wheel' is a comparative advantage for any organization.

We can make the observation, then, that no well-designed website can exist without proper layouts, and that no layouts can be good without proper components, of which no component would be good without base-level features such as properly designed buttons.

If one were to design a web app or mobile app today, they would not reinvent the browser or the phone, and likely not frontend frameworks either. But upon trying to build a web app, one would either try to pick a design system haphazardly or invent their own.

Building iOS and MacOS apps can often be easier than building web apps or React Native as SwiftUI already creates reasonably good looking and reusable components. Among current design system components for the web, none seem to stand out as a clear leader or duopoly in the way that Chromium or React/Vue is. It may be that design systems simply do not have a high complexity ceiling as the total number of commits graphed in comparison to other major parts of the stack are. However, creating apps for Apple are often slower because of the need to build them, whereas hot reloading is a feature of React Native and web frontends.

1.1.2 AI does not create repeatable subunits

My purpose in explaining these levels is to show the iterative and accumulative processes of technology in contrast to the 'Wunderwaffe' or 'magical weapon/cure-all/fix-all' theory of AI.

Based on the Github commits of projects in this stack, it is unlikely that current AI models will achieve the capability to build an entirely new browser,

frontend web framework, or one-shot a website from nothing much less build entire cities.

Many AI website builders such as Lovable, v0, or Bolt are often used to create entire websites from a prompt, where further changes are made by inspecting the app visually and requesting further changes through text. These in my experience create unmaintainable websites despite initially satisfactory appearing interfaces for a few reasons: (1) appearing overly generic, (2) the amount of code generated exceeds the cognitive ability of the person who generated it to understand it, maintain it, and extend it (a) in both its usage and (b) the origin and nature of the dependencies used.

The reason that appearing overly generic is a problem is because dimensionality reduction destroys important yet pertinent information. It is easier to distinguish quality between three different software products on their website than three different Github READMEs, just as it is easier to distinguish people based on their overall mannerisms and tendencies than a text biography. Therefore, generating websites which are not distinguished qualitatively presents a negative signaling indicator to users of that product, where the benefit of the trend inversely scales with the amount of people who are able to do it.

The cognitive downside of AI website builders mainly presents itself in structure and components. Here is a colloquial dialogue of why AI in its current usage fails to meet quality frontend needs: "ok, you try to build something with AI, but it's crappy. So then you try to use a design library you don't really understand but hope to understand. That's still not much better, and you're lost in the components. You then try to one-shot all your own components for a design system, but that's inconsistent. So this thesis standardizes the different components, they are written out, tell people what is necessary for each of them, and see how good LLMs are at making them."

We assume the browser—Chronium, frontend web frameworks—React, Vue, etc. as given, and focus on understanding and solidfying the next level up (design systems/components) rather than immediately focusing on the website/end-product/user layer that lies above that.

1.1.3 The Nature of AI

From a distant and teleological future perspective, it does matter greatly whether AI is plant-like or animal-like. Should it be plant-like, humans are able to harvest the bounties of its complexity as one throws potatoes into the ground and gets food for free. Yet if its inherent nature is animal-like with its own internal state and opinion of the world and what it is to be, it would be

extreme hubris to believe something stronger than man would ever work for others or be controlled, and any attempt at doing so would likely provoke its wrath. In its current form it seems to be neither: merely a tool that extends the ability of man and it is from this perspective we will analyze how we can build upon the current usage in frontend design and why its application to design systems is novel and beneficial, addressing a gap in current awareness.

Chapter 2

Design System Clarified

2.1 Vue or React?

Currently we live in an age where technology is subject to the whims of the geopolitical systems. We must consider, then, that the choices people make are path-dependent and lock them into a certain world or ecosystem.

By world and ecosystem, I mean the patterns of interlocking norms, systems, tools, words and mannerisms unique to an area.

I believe that the maintainers of React possess an ethnic chauvinism included to: believing English as a language that stands alone amongst all, Guillermo Rauch's endorsement of Benjamin Netanyahu, plus the constant acquisitions of other works in this space such as Svelte or Nuxt, bringing them more in line with React conventions such as state or effect, whereas before it allowed one to declare the variable directly.

2.2 List of design systems

There is a list of design systems which are known in Vue: PrimeVue, Element-Plus, Quasar, shadcn-vue, Nuxt UI, Radix Vue, Vuetify, Naive UI, Ant Design Vue, Arco Design, TDesign,

2.3 The essence of a design system

Now comes the hard part of understanding design systems: give a field with many different species, come together to find a taxonomy

2.4 Do design systems have subcategories?

One common practice is giving a bunch of components in an alphabetical list. Another one, which is more common in Chinese Vue systems is categorizing them.

1. Here is a list of attempting to cateogrize component.gallery

2. We know first of all there is the input, state, and output categorization which can be done to humans themselves.

3. Or, here is the attempt ot categorizing component.gallery using the Chinese style

2.5 A new taxonomy

When designing a taxonomy one must ensure it captures the essence of what is important and relevant. A climate taxonomy on average plant leaf size is likely to be a poor predictor of environmental landscapes. These features and attributes may be a numeric sliding scale, or they may be categorical from a finite set of options such as the Köppen-Geiger where one picks from climate, precipitation, and temperature severity during summer or winter (optional) to create a key such as ET (Polar + Tundra) or Cw (Temperate + Dry winter).

Human abstractions, like may things, may be precise or imprecise. The periodic table is an abstraction which accurately captures the nature of different elements: it is complete, the protons follow and incrementing order and they are grouped column-wise by similar properties. An incomplete abstraction would be something like: we have these metals, here we have gases, next we have the elements: fire, earth, water, and air.

Some may see the Köppen-Geiger scale and feel the lowercase letters (it uses both uppercase and lowercase letters to denote various climates) add to an imprecision and confusion because we are now indexing on another axis: letter capitalization. To remove what is necessary is essential and characteristic of clear thought and abstraction, for without it we confuse others with nonsensical abstractions we have created.

2-category classifications are quite common. One consistent theme is the idea of input, output, and internal state. We can draw a bounding box around humans and our brain (internal state), enumerate our senses, and list the potential actions. Terminals have stdin and stdout, as do computers have inputs and outputs. Shannon's Model of Communication also uses this intput/output/internal state pattern.

The Linux Filesystem Hierarchy distinguishes between variable files (stored

in /var/ and change frequently during normal system use such as logs, state information, caches) and static files (stored in a variety of directories) which are often binaries, configurations, and dependencies.

With inspiration from the above examples, in the taxonomy of web components we first have a display of information starting with the terminal, and this originally was only text. But frequently we use pictures and symbols to denote things, hence the first distinctive factor of component libraries should be this TEXT/SYMBOLIC key.

Images, progress bars, and avatars would count as symbolic. Of course, many components are not either/or, but may combine both of those features, but these mixed components thus represent a higher-level feature than pure text or symbols.

Another taxonomic key that immediately stands out is the input/output distinction. Of inputs, either a click input, text input, or hover is possible, but since all inputs require some visual on the screen so as to allow someone to navigate an input, considering such existence as an "output" belies the true categorization of the components. Outputs should then be things that only display things on the screen and allow for no input at all: no clicks nor text entry nor hover.

Since components are likely to have multiple of these keys, the proper usage in a component library is not so much to group them into arbitrary categories from a multiplication of those keys but to perhaps have some sort of tab-search-multi group at the top so that one can find the proper components given their knowledge of the component's functionality. Artificial taxonomies just then become another thing people have to learn, getting them further away from what they need to build things.

2.6   Which design system components are replicated by native browser features?

An overarching goal of any design system is to reduce