Jacaranda: Grafting User-Editable Web Experiences Across Sites

Xinqi Zhang
Santa Clara University
USA
xzhang22@scu.edu

Mingyuan Zhong
University of Washington
USA
myzhong@cs.washington.edu

Longjie Guo
University of Washington
USA
longjie@uw.edu

Bryan Min
bdmin@ucsd.edu
University of California San Diego
USA

Yuanhao Zhang
yzhangiy@connect.ust.hk
Hong Kong University of Science and Technology
Hong Kong, China

Kai Lukoff
Santa Clara University
USA
klukoff@scu.edu

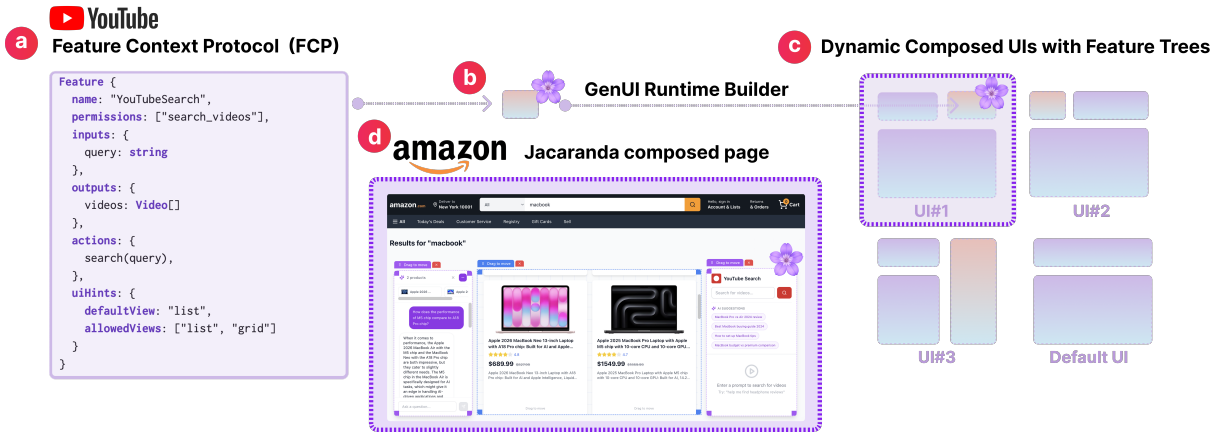


Figure 1: Jacaranda turns web pages into recomposable feature trees. In a shopping workflow, the user can add a YouTube Search feature and a product comparison LLM directly to an Amazon page. Feature Context Protocols expose feature data and UI/UX constraints, while the GenUI Runtime Builder helps connect and reshape features across different sites into a user-defined workflow. Figure 1d: an Amazon page running in Jacaranda with imported YouTube Search and a Comparison LLM feature.

Abstract

Today's web users often pursue goals that span multiple pages, tabs, and platforms, yet web interfaces remain organized around provider-defined pages and application boundaries. A user comparing products may move between shopping pages, review videos, search results, notes, and comparison tools, but the features that support this workflow are locked inside separate sites. We introduce Jacaranda, a working runtime browser environment that reimagines the web as a feature-oriented, user-editable workspace. In Jacaranda, online shoppers can add widgets that enrich a webpage with relevant content, such as product comparisons and YouTube video results. Jacaranda represents web pages as feature trees and enables cross-site capabilities through the Feature Context Protocol (FCP), which defines feature data constraints while supporting diverse UI/UX substrates. We describe its architecture as GRAFT (Generative Runtime Assembly of Feature Trees), a system model for how features are composed, modified, and connected across sites at runtime by a GenUI Runtime Builder.

CCS Concepts

• **Human-centered computing** → **User interface toolkits**; **Web-based interaction**; *Graphical user interfaces.*

Keywords

generative user interfaces, malleable software, web browsers, end-user programming, user-defined workflows

ACM Reference Format:
Xinqi Zhang, Mingyuan Zhong, Longjie Guo, Bryan Min, Yuanhao Zhang, and Kai Lukoff. 2026. Jacaranda: Grafting User-Editable Web Experiences Across Sites. In *The 39th Annual ACM Symposium on User Interface Software and Technology (UIST Adjunct '26), November 02–05, 2026, Detroit, MI, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3830397.3841817

This work is licensed under a Creative Commons Attribution 4.0 International License.

UIST Adjunct '26, Detroit, MI, USA

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2855-6/2026/11
https://doi.org/10.1145/3830397.3841817

1 Introduction

People often need to visit multiple websites to complete meaningful web tasks, engaging across a variety of services, tabs, and interface

contexts [3, 7]. A shopper looking for a laptop may compare Amazon products, watch YouTube reviews, read external articles, and keep notes. A student learning a new topic may watch a lecture, take notes, search references, and suppress distracting popups. The interfaces that support these workflows are fragmented across the web.

What if a system could combine features from multiple websites to support these workflows? The current web presents two blockers to user-defined malleability: platform-controlled feature access and page-centered representation. Platforms such as YouTube and Amazon offer rich capabilities, but their features are bound to their ecosystems rather than reusable across contexts [4, 12]. Meanwhile, the Document Object Model (DOM) exposes pages as document nodes [8], and frontend frameworks organize interfaces into component trees [11]; however, neither represents features such as "compare products," "summarize reviews," or "take notes" as recomposable interface capabilities. Prior work has explored multiple paths toward malleable web experiences: browser extensions, scripts, and mashups let users adapt existing sites in limited ways [1, 6, 14], while Jelly shows how Generative UI can produce bespoke, task-driven applications from evolving data models [2]. However, these approaches do not directly support recombining live web features across application boundaries.

We propose Jacaranda, a runtime browser environment for user-defined web experiences across sites, implemented as a browsing sandbox within Chrome that hosts web features and uses an LLM (GPT-4o) to generate UI code at runtime from direct manipulation (dragging features) and natural-language input (**Figure 1**). Demonstrated on a live shopping workflow spanning Amazon and YouTube, Jacaranda shifts the unit of interaction from fixed page tabs to portable, reusable feature components powered by generative UI. To support interoperability across services, we introduce *Feature Context Protocol (FCP)*, which defines feature data constraints while supporting diverse UI/UX substrates, and *GRAFT: Generative Runtime Assembly of Feature Trees*, a system model for runtime-malleable feature composition and layout adaptation via a GenUI Runtime Builder [5, 9, 15]

2 Feature Substrates: Feature Context Protocols

Interface resources on the current web are largely restricted by service providers: platforms such as YouTube and Amazon expose rich features and data, but their application boundaries limit reuse and recomposition across contexts [4, 12]. Inspired by the Model Context Protocol's effort to standardize how AI applications connect to external tools and data [10], the Feature Context Protocol (FCP) explores how web services might similarly expose structured capabilities for user-defined runtime composition.

Figure 1a shows how YouTube Search is defined as an atomic, portable feature through an FCP. In this example, the FCP specifies not only API data constraints, but also UI constraints as rule-based hints for GenUI. With the FCP as a blueprint, the same feature can be connected to diverse UI/UX presentations that respect the same data constraints.

An FCP can be understood as a universal interface standard for service providers: a "USB-like" connector between platform resources and runtime-generated user experiences. As demonstrated in **Figure 1b**, when YouTube Search is introduced as a new feature on the Amazon site, the GenUI Runtime Builder generates an atomic UI component that adapts to Amazon's layout context. With FCPs, the same underlying data source supports multiple client-side UI/UX presentations, as shown in **Figure 1c**. FCP thus serves as the way sites *declare* features, rather than the general way features *enter* users' web experiences.

3 From DOM to Feature-Tree Pages

A further blocker to malleability is the structure of web presentation itself: the Document Object Model (DOM). While the DOM provides a standardized representation for rendering page elements, it primarily exposes a web page as a presentation-oriented structure rather than as a reusable collection of functional capabilities [1]. As a result, users have limited ability to identify, extract, or recombine specific features and data across contexts [13].

To address this limitation, we take inspiration from component-based frameworks such as React, where interfaces are organized as trees of reusable components. We propose rethinking web representation as a *Feature-Tree Page* rather than as a fixed page. In a Feature-Tree Page, each feature, such as YouTube Search or Video Captions, is represented as an atomic and portable feature node. These nodes can be reused in other contexts when compatible data and UI/UX constraints are guaranteed by the FCP.

The web thus shifts from a set of isolated tabs toward a network of interoperable feature nodes. For example, **Figure 1d** shows an Amazon site represented as a Feature-Tree Page. Users in a shopping workflow can edit Amazon's own features, import features to assist their shopping workflow, such as YouTube Search and a Comparison LLM, and create their own features within the current feature-tree view, such as Notes and Checklist.

4 Jacaranda System Model: GRAFT

Building on FCPs and Feature-Tree Pages, we introduce GRAFT: *Generative Runtime Assembly of Feature Trees*, the system model underlying Jacaranda.

GRAFT maintains three core goals: G1, to decompose pages into meaningful feature units; G2, to support user-defined composition through low-barrier, GenUI-assisted authoring; and G3, to enable portable, cross-site reuse of features. A central tension between these goals concerns who decides the boundaries of feature units. If features are declared only upstream by service providers, then user composition remains limited by provider-chosen granularity. GRAFT separates runtime feature composition into two operations: *defining features* (G1, G2) and *composing pages* (G3).

To **define a feature** is to specify its boundaries from both data and UI perspectives. In GRAFT, features can be defined from three sources: host, imported, and synthetic. *Host features* originate from the current site, such as product cards and product information in **Figure 1** Amazon example. These features may be explicitly declared through an FCP or extracted from the rendered page. *Imported features* come from other services, such as YouTube reviews brought into an Amazon shopping workflow or map results brought into a travel-planning workspace. These features may be exposed through an FCP, embedded directly, or connected through API

adapters. *Synthetic features* do not originate from any single website. Instead, they are generated to support the user's task, such as personal notes or a product-comparison LLM. These features may be instantiated from a built-in component library or generated by an LLM.

To **compose a page** is to decide how features are placed and reused within a Feature-Tree Page. This composition layer is user-editable: users can decide which features to include through natural language and where those features should appear through direct manipulation. From the user interaction perspective, synthetic features are created through a conversational UI with natural-language input, such as "I want to take notes." Host features and imported features are primarily edited through direct-manipulation interactions, such as dragging, resizing, and pinning them in the current view.

5 Discussion and Future Work

Jacaranda reframes the browser as an editable medium for user-defined web experiences, treating features as portable building blocks rather than closed applications. This raises questions about how platforms should expose feature capabilities through FCPs, how users can control GenUI-generated transformations, and how privacy is preserved when application boundaries are crossed. Future work will explore FCP-defined features and Feature-Tree Pages across shopping, video learning, travel planning, and accessibility.

References

[1] Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. 2005. Automation and Customization of Rendered Web Pages. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology (UIST '05)*. Association for Computing Machinery, New York, NY, USA, 163–172. doi:10.1145/1095034.1095062

[2] Yining Cao, Peiling Jiang, and Haijun Xia. 2025. Generative and Malleable User Interfaces with Evolving Task-Driven Data Schema. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, 1–21. doi:10.1145/3706598.3713285

[3] Joseph Chee Chang, Nathan Hahn, Yongsung Kim, Julina Coupland, Bradley Breneisen, Hannah S. Kim, John Hwong, and Aniket Kittur. 2021. When the Tab Comes Due: Challenges in the Cost Structure of Browser Tab Usage. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3411764.3445585

[4] Anne Helmond. 2015. The Platformization of the Web: Making Web Data Platform Ready. *Social Media + Society* 1, 2 (2015), 1–11. doi:10.1177/2056305115603080

[5] Clemens N. Klokmose, James R. Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. 2015. Webstrates: Shareable Dynamic Media. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 280–290. doi:10.1145/2807442.2807446

[6] Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. 2008. CoScripter: Automating and Sharing How-To Knowledge in the Enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. Association for Computing Machinery, New York, NY, USA, 1719–1728. doi:10.1145/1357054.1357323

[7] Bonnie MacKay and Carolyn Watters. 2008. Exploring Multi-session Web Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, 1187–1196. doi:10.1145/1357054.1357243

[8] MDN Web Docs. 2026. Document Object Model (DOM). https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model Web API documentation.

[9] Bryan Min, Allen Chen, Yining Cao, and Haijun Xia. 2025. Malleable Overview-Detail Interfaces. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3706598.3714164

[10] Model Context Protocol. 2025. Model Context Protocol Specification. https://modelcontextprotocol.io/specification/2025-06-18. Version 2025-06-18.

[11] React. [n. d.]. Thinking in React. https://react.dev/learn/thinking-in-react. Accessed 2026-05-20.

[12] Ruben Verborgh and Michel Dumontier. 2018. A Web API Ecosystem through Feature-Based Reuse. *IEEE Internet Computing* 22, 3 (2018), 29–37. doi:10.1109/MIC.2018.032501515

[13] Tim Weninger, Rodrigo Palacios, Valter Crescenzi, Thomas Gottron, and Paolo Merialdo. 2016. Web Content Extraction: a MetaAnalysis of its Past and Thoughts on its Future. *SIGKDD Explor. Newsl.* 17, 2 (Feb. 2016), 17–23. doi:10.1145/2897350.2897353

[14] Jeffrey Wong and Jason I. Hong. 2007. Making mashups with marmite: towards end-user programming for the web *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 1435–1444. doi:10.1145/1240624.1240842

[15] Xinqi Zhang, Bryan Min, Shizhao Lawrence Liu, and Kai Lukoff. 2026. User Agency in Runtime Malleable Software Under Change Aversion. Workshop paper for the CHI 2026 Workshop on GenUI Meets HCI.