A RESOURCE FOR DESIGN SYSTEM AND ENGINEERING LEADS

Framework decision matrix.

React wrapper or native custom elements?

Score the team, not the trend.

25	04	03
criteria, gates and route rules	architectures scored side by side	worked profiles, three different answers

Decide once, in writing, with the numbers visible.

The web components question is usually argued as taste. Argue it as fit. A single React app with SEO-critical, server-rendered pages gains little from custom elements; a suite spanning React, Angular and server templates for seven more years gains a great deal. The answer turns on a few facts: how many frameworks render product UI, how much markup must arrive from the server, and which React version the slowest consumer runs.

That last fact changed in React 19. React now sets a prop as a property when the element has one, binds custom events through `on` plus the event name, and passes all 16 basic and 16 advanced tests on Custom Elements Everywhere [S01, S03]. On React 18 you still need a wrapper. So "wrapper or native" is no longer a philosophy; it is mostly a version check, and this guide treats it as one.

The kit makes the decision reproducible: a weighted matrix with 11 criteria and 2 gates, a scorer that ranks four architectures and names what decided it, three worked profiles with three different answers, and a JSX type generator for the native route. The component throughout is `ds-button` from Field Guide 15.

Version 1.0 / Sources checked 24 September 2026

Field Guide 13 of the Design × AI series; pairs with 14 and 15. Versions verified 24 September 2026: React 19.3.0, Lit 3.3.3, @lit/react 1.0.8, @lit-labs/ssr 4.1.0, CEM analyzer 0.11.0. Library versions are in Appendix C.

Practical guidance, not a standard. The weights in matrix.json are editorial defaults; change them and re-run the scorer. Prepared with AI assistance and edited by hand.

START HERE

Four architectures, one scorer.

The essay's matrix had three columns. Splitting web components into native and wrapped makes the React question explicit, because that is where teams actually disagree.

Option id	What you ship	React consumers use
`framework-only`	Components in your one framework. No custom elements.	The components directly.
`wc-native`	Custom elements, typed from the Custom Elements Manifest.	`<ds-button>` in JSX (React 19 and later).
`wc-wrapped`	Custom elements plus a generated React package.	`<Button>` from `@lit/react` `createComponent`.
`hybrid`	Framework components on server-rendered paths; custom elements for shared widgets.	Both, by route.

Choosing for the first time

Answer the 11 questions in section 01 as a group, save them as a profile file and run the scorer. Argue about the answers, not the result.

Already on web components

Section 02 and the route rules. The React 19 gate decides whether your wrapper package is still earning its keep.

Already framework-only

Score the team you will be in three years. If C01 or C08 changes, re-run; nothing else usually moves the answer.

Writing the decision record

Paste the scorer's output and the profile into the record at the back. The drivers line is the paragraph your successor needs.

Read the labels before the items.

Label	Meaning
`MATRIX`	A weighted criterion in `matrix.json`, scored by `score.mjs`.
`GATE`	Removes an option outright when a condition holds, with a stated reason.
`KIT TEST`	Observed in a real Chromium by the Field Guide 15 starter's tests.
`SPEC`	Follows from a platform specification or a framework's own documentation.
`PRACTICE`	A working method with a review signal rather than a hard check.

SECTION 01

Score the team you have.

Eleven questions, each with fixed answers. Weight 3 criteria decide most outcomes; weight 1 criteria break ties. Every answer should be checkable by someone outside the room.

Suggested owners: Design-system lead + engineering lead + one product engineer from each framework

- [ ] **How many frameworks render product UI**

  C01 MATRIX

  Answers: one, two, three or more. Count server templates (Java, PHP, Rails, a CMS) as a framework. Weight 3. One framework favours framework-only components by a wide margin; two or more is the strongest single reason to consider custom elements.

  Evidence: A list of repositories that render product UI, with their framework and owner.

  The essay's first criterion: two or more frameworks in active use.

- [ ] **How long the library must outlive today's framework**

  C02 MATRIX

  Answers: under 3 years, 3 to 5, over 5. Weight 2. A long horizon favours browser-native components because the platform changes more slowly than frameworks do.

  Evidence: The roadmap horizon the organisation actually funds, not the one it hopes for.

  The essay's 5+ year horizon.

- [ ] **Whether independently deployed apps share a page**

  C03 MATRIX

  Answers: no, yes. Weight 2. When micro-frontends share a page, a custom element is a stable contract between them; a wrapper package scores lower because each app pins its own React peer.

  Evidence: An architecture diagram that shows which apps render into the same document.

  Essay: micro-frontend architectures.

- [ ] **How much markup must arrive in the server HTML**

  C04 MATRIX

  Answers: under 25%, 25 to 50%, over 50% of pages. Weight 3. Over half, both web component options score 0: React 19 renders custom elements shallowly on the server, and deep rendering of their shadow roots needs Lit's Labs tooling.

  Evidence: Page inventory marked by whether SEO, LCP or no-JS rendering depends on component markup.

  Essay: the 50% threshold. React 19 SSR emits attributes for primitives only (S01); Lit SSR is a Labs package (S08).

- [ ] **Which server renderer produces the HTML**

  C05 MATRIX

  Answers: none (client-rendered), a Lit SSR integration (Astro, Eleventy, Nuxt, Next.js Pages Router), or React Server Components. Weight 2. Lit's Next.js plugin is tested with Next 13 and 14, and deep SSR does not work in server components.

  Evidence: The framework and router version in each app's `package.json`.

  S08, S10.

The lowest React version any consumer runs

C06 MATRIX SPEC

Answers: no React, 18 or lower, 19 or later. Weight 3. This criterion carries gate G1, and it is the one that flips the wrapper question.

Evidence: `npm ls react` in every consuming repository; the minimum, not the average.

React 19 release notes and the react-dom custom element reference (S01, S02).

Custom element experience in the owning team

C07 MATRIX

Answers: none, some, strong. Weight 2. Shadow DOM styling, form association and SSR are learnable, but a team with none of it pays for the learning in production.

Evidence: Who on the team has shipped a custom element with a shadow root, form participation and tests.

Essay: a team without web component experience is a reason to avoid them.

Components that must render outside any framework

C08 MATRIX

Answers: no, yes. Weight 3. CMS pages, server templates and third-party embeds can load a script tag; they cannot host a React tree without shipping React.

Evidence: Pages or partners that need design-system UI with no framework runtime.

Custom elements need only their module; framework components need their runtime.

Participation in native forms

C09 MATRIX KIT TEST

Answers: no, yes. Weight 1. A native submit button inside a shadow root does not submit its form, so every web component that acts in a form needs `ElementInternals`. That is work, not a blocker.

Evidence: The Field Guide 15 test "a submit button inside a shadow root does not submit the outer form" passes in Chromium 153.

Form-associated custom elements, Baseline widely available since September 2025 (S11, S12).

Typed props and autocomplete in JSX

C10 MATRIX

Answers: no, yes. Weight 1. Wrappers carry types. The native route needs JSX types generated from the manifest; the kit's `cem-to-jsx-types.mjs` does that.

Evidence: A `@ts-expect-error` line on an invented prop value that the type check requires.

Kit: `usage.react19.tsx` type-checks with the generated `ds-jsx.d.ts`.

Capacity to release a wrapper package every time

C11 MATRIX

Answers: no, yes. Weight 1. A wrapper package that lags the element by a release is worse than no wrapper, because React consumers see props the element no longer has.

Evidence: A release pipeline that regenerates, tests and publishes wrappers with the core, as Spectrum Web Components does.

S15.

Argue about the answers. The arithmetic is not the interesting part.

SECTION 02

Gates first, then the ranking.

Some answers make an option wrong, not merely weaker. Gates remove it with a reason the scorer prints. The ranking only orders what is left.

Suggested owners: Design-system lead

- [ ] **React 18 anywhere rules out the native route**

  G1 GATE SPEC

  React 18 passes every prop to a custom element as an attribute and cannot listen for its custom events. Objects, arrays and `false` do not survive that. Keep wrappers until the last React consumer is on 19.

  Evidence: The enterprise-suite profile: `wc-native` is excluded by G1 and `wc-wrapped` wins at 92.8%.

  S01, S03. Web Awesome ships wrappers only for React 18 and below (S13).

- [ ] **No React consumers rules out wrappers**

  G2 GATE

  A wrapper package with nobody importing it is maintenance without a user.

  Evidence: `npm ls react` returns nothing across all consumers.

  Kit gate.

- [ ] **Within 5 points is a close call, and the scorer says so**

  D01 PRACTICE

  Weights are opinions. When the top two options land within the tie band, decide on the highest-weight criterion where they differ and write that sentence in the record.

  Evidence: The scorer prints `close call` and the driver list; the record has one sentence per driver.

  Kit: `tieBand` in `matrix.json`.

- [ ] **Re-score when a weight 3 answer changes**

  D02 PRACTICE

  A new framework, a move to server components or the last React 18 app upgrading changes the answer. Re-run the profile, compare, and record the change.

  Evidence: The enterprise suite re-scored with C06 at 19-plus moves to `wc-native` at 92.8% against `wc-wrapped` at 88.4%: a close call.

  Kit run, Appendix B.

A wrapper package is a bridge. Plan the day you stop maintaining it.

SECTION 03

If you wrap: thin, generated, temporary.

The essay's rule stands: the wrapper adds ergonomics, not logic. What changed is how you build it and how long you keep it.

Suggested owners: Engineering lead + release owner

- [ ] **Generate wrappers; never hand-write them**

  W01 PRACTICE

  Use `@lit/react` `createComponent`, the Stencil React output target or a manifest-driven generator. Hand-written `useEffect` and `addEventListener` wrappers drift from the element.

  Evidence: One short file per component, no JSX; the Field Guide 15 wrapper is 17 lines.

  S04, S18.

- [ ] **Map custom events once, in the wrapper**

  W02 SPEC

  `events: { onDsChange: "ds-change" }` gives React consumers camelCase handlers. Native `click` needs no mapping: it bubbles to React's `onClick`.

  Evidence: Field Guide 15 test: the wrapper forwards `onClick` with an empty `events` map.

  S04.

- [ ] **Version the wrapper with the element**

  W03 PRACTICE

  Same repository, same release, same version. A wrapper built from an older manifest offers props the element no longer has. Keep `react` an optional peer so non-React consumers install nothing extra.

  Evidence: One release job regenerates, tests and publishes element and wrapper together.

  Spectrum Web Components does this with `@swc-react` (S15).

- [ ] **Neither wrappers nor React 19 fix SSR**

  W04 SPEC

  React 19 SSR renders the tag with primitive attributes only. `@lit-labs/ssr-react` renders shadow roots, but not for `createComponent` output. Wrappers change ergonomics, not server HTML.

  Evidence: View source: no `<template shadowrootmode>` inside `<ds-button>` unless Lit SSR rendered it.

  S01, S09.

- [ ] **Deprecate the wrapper when G1 stops applying**

  W05 PRACTICE

  Web Awesome and UI5 Web Components for React already branch on React 19. Announce the removal version when the last consumer upgrades.

  Evidence: A deprecation notice with a removal version, and C06 at 19-plus in the current profile.

  S13, S16.

The wrapper adds ergonomics, not logic.

SECTION 04

If you go native: React 19, with care.

React 19 closed the gap, not every gap. These rules come from what the Field Guide 15 starter observed in Chromium 153 with React 19.3.0.

Suggested owners: Engineering lead + one React consumer

- [ ] **Define the element before React renders it**

  N01 KIT TEST SPEC

  React 19 sets a property only if the element already has it. Before definition it writes attributes: strings and `true` survive (the starter saw `loading=""`); objects and functions do not.

  Evidence: Field Guide 15 test "React 19 falls back to attributes when the element is defined after render".

  react-dom custom element reference (S02).

- [ ] **Booleans and rich values reach the element as properties**

  N02 KIT TEST

  `<ds-button loading={saving}>` sets `el.loading = true`, not the string "true". This is the behaviour React 18 lacked and the reason wrappers existed.

  Evidence: Field Guide 15 test "React 19 sets custom element properties without a wrapper".

  S01.

- [ ] **Listen to custom events by their exact name**

  N03 SPEC

  React 19 binds `on` plus the event name, case-sensitive, dashes kept: `onds-change`, not `onDsChange`. Document every event in the manifest.

  Evidence: Every event in `custom-elements.json` has a name and description; the generated JSX types list `"on<event-name>"`.

  S02.

- [ ] **Generate JSX types from the manifest**

  N04 KIT TEST

  Without types, TypeScript rejects `<ds-button>` (TS2339). With `cem-to-jsx-types.mjs` output, the manifest's unions reach JSX: `tone="ghostly"` fails with TS2820 and suggests `"ghost"`.

  Evidence: `tsc --noEmit` over `usage.react19.tsx`: exit 0, with two `@ts-expect-error` lines that must fail.

  Kit run against the Field Guide 15 manifest.

- [ ] **Buttons and fields must be form-associated**

  N05 KIT TEST

  `static formAssociated = true`, `attachInternals()`, and handle submit, reset and fieldset disabling. React form handlers then work unchanged.

  Evidence: Field Guide 15 tests: the ds-button submits its light-DOM form and is disabled by a disabled fieldset.

  S11, S12.

Native means no wrapper, not no work.

APPENDIX A

The weights, and where each option wins.

The maximum is 69 points (23 weight units times 3). The table shows the three weight 3 criteria that decide most profiles, with each option's score per answer.

Criterion and answer	framework-only	wc-native	wc-wrapped	hybrid
C01 one framework	3	1	1	1
C01 two frameworks	1	3	3	2
C01 three or more	0	3	3	2
C04 under 25% server-rendered	3	3	3	2
C04 25 to 50%	3	2	2	3
C04 over 50%	3	0	0	3
C06 no React	0	3	0 (G2)	1
C06 React 18 or lower	3	0 (G1)	3	2
C06 React 19 or later	3	3	2	2

Weights: C01, C04, C06 and C08 are 3; C02, C03, C05 and C07 are 2; C09, C10 and C11 are 1. Full answers and a basis for every criterion are in `matrix.json`.

matrix.json (one criterion and one gate)

```
{
  "id": "C06",
  "question": "What is the lowest React version any consumer runs?",
  "weight": 3,
  "answers": {
    "no-react":    { "framework-only": 0, "wc-native": 3, "wc-wrapped": 0, "hybrid": 1 },
    "18-or-lower": { "framework-only": 3, "wc-native": 0, "wc-wrapped": 3, "hybrid": 2 },
    "19-plus":     { "framework-only": 3, "wc-native": 3, "wc-wrapped": 2, "hybrid": 2 }
  }
},
{
  "id": "G1",
  "excludes": "wc-native",
  "when": { "C06": "18-or-lower" },
  "reason": "React 18 passes every prop to a custom element as an attribute ..."
}
```

The scorer validates the matrix before it scores: weights 1 to 3, scores 0 to 3, a basis on every criterion, gates that point at real answers.

APPENDIX B

Three teams, three answers.

Output of `node scripts/score.mjs profiles/*.json`, unedited apart from line wrapping. Item D02 shows the enterprise suite flipping once its last app reaches React 19.

profiles/checkout-team.json

```
Profile: Checkout team: one React 19 app on the Next.js App Router, SEO-critical

 1. framework-only   61/69   88.4%
 2. hybrid           46/69   66.7%
 3. wc-wrapped       35/69   50.7%
 4. wc-native        34/69   49.3%

Recommendation: framework-only
Why framework-only over hybrid:
  +6 C01 one: How many UI frameworks render product UI today?
  +4 C07 none: How much custom element experience does the team have?
  +3 C06 19-plus: What is the lowest React version any consumer runs?
```

profiles/enterprise-suite.json

```
Profile: Enterprise suite: React 18, Angular and Java server templates,
         micro-frontends, 7-year horizon

 1. wc-wrapped       64/69   92.8%
 2. hybrid           46/69   66.7%
 3. framework-only   40/69   58.0%
 -  wc-native       excluded by G1: React 18 passes every prop to a custom
                    element as an attribute and cannot listen for its custom
                    events. Ship wrappers until the last React consumer is on 19.

Recommendation: wc-wrapped
```

profiles/platform-after-upgrade.json

```
Profile: Platform after the upgrade: React 19 and Vue apps plus CMS pages,
         no wrapper budget

 1. wc-native        63/69   91.3%
 2. wc-wrapped       57/69   82.6%
 3. hybrid           47/69   68.1%
 4. framework-only   39/69   56.5%

Recommendation: wc-native
Why wc-native over wc-wrapped:
  +3 C06 19-plus: What is the lowest React version any consumer runs?
  +3 C11 no: Can you generate, test and release a wrapper package ...?
```

APPENDIX C

What seven web component libraries ship for React.

From each project's documentation, source or npm manifest, checked 24 September 2026. The pattern: generate wrappers from a manifest, and branch on React 19.

Library	React approach	Source
Shoelace 2.20.1	Wrappers at `@shoelace-style/shoelace/dist/react`, events mapped (`sl-input` to `onSlInput`). The project now points to Web Awesome.	S14
Web Awesome 3.13.0	Native use on React 19 ("No wrappers needed"); wrappers kept for React 18 and below; JSX types in `custom-elements-jsx.d.ts`.	S13
Spectrum Web Components 1.12.2	`@swc-react/*` packages built on `@lit/react`, generated from `custom-elements.json` by a manifest plugin.	S15
Carbon web components 2.64.0	The package depends on Lit and no React; forms join through the `formdata` event. IBM's React path is `@carbon/react`.	S20
Ionic 9.0.5	`@ionic/react` generated by the Stencil React output target; routing components hand-written; peers React 18 or 19.	S17, S18
Material Web 2.5.0	In maintenance mode pending new maintainers; no React-specific package.	S19
UI5 Web Components for React 2.27.0	Own `withWebComponent` wrapper that checks for React 19 at runtime and binds events natively there, manually below it.	S16

examples/usage.react19.tsx (wc-native)

```
import "ds-lit-starter"; // defines <ds-button> before React renders it (N01)

export function SaveBar({ saving, onSave }: { saving: boolean; onSave: () => void }) {
  return (
    <form onSubmit={(e) => { e.preventDefault(); onSave(); }}>
      <ds-button type="submit" loading={saving}>Save changes</ds-button>
      <ds-button tone="ghost" onClick={() => history.back()}>Cancel</ds-button>
    </form>
  );
}

// @ts-expect-error "ghost-primary" is not a tone: the manifest's union reaches JSX
export const Invented = () => <ds-button tone="ghost-primary">Nope</ds-button>;
```

Type-checks with TypeScript 7.0.2 against the built Field Guide 15 starter, as does `usage.react18.tsx`, which imports `Button` from `ds-lit-starter/react`. Removing the generated `ds-jsx.d.ts` turns every `<ds-button>` into TS2339.

KEEP WITH THE ARCHITECTURE DECISION

Leave a decision record.

One record per decision. It is the trail that lets the next lead see which facts the choice rested on, and which change would reopen it.

Decision / date / owner	**Profile file (link) and matrix version**
Scorer output (paste)	**Options excluded by gates, and why**
Close call? The deciding criterion	**Answers most likely to change**
Wrapper removal version (if wrapped)	**Pilot component and its test run (link)**
Consumers consulted (one per framework)	**Re-score trigger / next review**

> **Changing a weight is a decision too.**
> If you edit `matrix.json` to get a different answer, record the old and new weights and the reason. A matrix tuned to a conclusion is a slide, not a decision.

SOURCES / MAINTENANCE

Keep the guide current.

Sources checked 24 September 2026. Library versions are the npm latest on that date. Behaviour marked KIT TEST was observed in the Field Guide 15 starter (Chromium 153, React 19.3.0); nothing in this guide was measured on React 18 directly.

S01 / React, React 19 release post
Full custom element support: properties when the instance has them, attributes otherwise; SSR renders primitives as attributes.
https://react.dev/blog/2024/12/05/react-19

S02 / React, react-dom components: custom HTML elements
Property versus attribute rule and on-prefixed, case-sensitive custom event props.
https://react.dev/reference/react-dom/components#custom-html-elements

S03 / Custom Elements Everywhere
React ^19: 16/16 basic, 16/16 advanced, 100%. React 18 is no longer listed.
https://custom-elements-everywhere.com/

S04 / Lit, React integration
@lit/react createComponent: tagName, elementClass, react, events.
https://lit.dev/docs/frameworks/react/

S05 / Custom Elements Manifest schema
Attributes, members, events, slots, CSS properties, parts and states per element.
https://github.com/webcomponents/custom-elements-manifest

S06 / Custom Elements Manifest analyzer
Generates custom-elements.json from source; Lit support behind a flag; JSDoc tags.
https://custom-elements-manifest.open-wc.org/analyzer/getting-started/

S07 / MDN, the template element: shadowrootmode
Declarative shadow DOM; Chrome 111, Firefox 123, Safari 16.4.
https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/template

S08 / Lit, server-side rendering overview
Labs status, limitations and framework integrations.
https://lit.dev/docs/ssr/overview/

S09 / @lit-labs/ssr-react README
Deep SSR inside React; does not work for precompiled JSX or createComponent output.
https://github.com/lit/lit/tree/main/packages/labs/ssr-react

S10 / @lit-labs/nextjs README
Tested with Next.js 13 and 14; no deep SSR in server components.
https://github.com/lit/lit/tree/main/packages/labs/nextjs

S11 / web.dev, More capable form controls
formAssociated, attachInternals, setFormValue, setValidity and form callbacks.
https://web.dev/articles/more-capable-form-controls

S12 / Shoelace discussion 1057: buttons in shadow roots and forms
A submit button inside a shadow root does not submit the outer form, by design.
https://github.com/shoelace-style/shoelace/discussions/1057

S13 / Web Awesome, React
Native on React 19; wrappers for React 18 and below; JSX type file.
https://webawesome.com/docs/frameworks/react/

S14 / Shoelace, React
Wrapper import paths and event name mapping.
https://shoelace.style/frameworks/react

S15 / Spectrum Web Components, using swc-react
React wrappers built on @lit/react, generated per release.
https://opensource.adobe.com/spectrum-web-components/using-swc-react/

S16 / UI5 Web Components for React, withWebComponent
Runtime check for React 19; native event binding there, manual listeners below.
https://github.com/UI5/webcomponents-react/blob/main/packages/base/src/internal/wrapper/withWebComponent.tsx

S17 / Ionic Framework, stencil.config.ts
reactOutputTarget generates @ionic/react components.
https://github.com/ionic-team/ionic-framework/blob/main/core/stencil.config.ts

S18 / Stencil, React integration
Output target with a runtime built around @lit/react; SSR through a hydrate module.
https://stenciljs.com/docs/react

S19 / Material Web repository
Maintenance mode pending new maintainers.
https://github.com/material-components/material-web

S20 / Carbon web components, forms
Form components join forms through the formdata event.
https://github.com/carbon-design-system/carbon/blob/main/packages/web-components/docs/form.md

S21 / WC Toolkit, React wrappers
Manifest-driven wrapper and type generation; notes React 19 can use elements directly.
https://wc-toolkit.com/integrations/react/

S22 / Dave Rupert, the Custom Elements Manifest killer feature
One manifest feeding docs, IDEs, wrappers and types (October 2025).
https://daverupert.com/2025/10/custom-elements-manifest-killer-feature/

Maintenance: re-check Custom Elements Everywhere and the React release notes at each React major, and the library table at each of their majors. Re-run the three profiles after any change to matrix.json and update Appendix B from the output. Update the PDF, HTML, Markdown and JSON together.