



AppSPEC in One Page · Architectural Cheatsheet

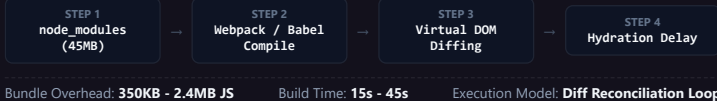
The In-Browser Application Server · Declare Entire Reactive Multi-Screen Apps in Pure XML

01 ARCHITECTURAL PARADIGMS · LEGACY SPA TOOLCHAIN VS. APPSPEC ZERO-BUILD

Eliminating Bundling Friction, Node Toolchains, and Hydration Pauses

LEGACY SPA (REACT / VITE / NEXT.JS)

Heavyweight Bundler Stack



APPSPEC ARCHITECTURE (IN-BROWSER ENGINE)

Zero-Build Native DOM



02 CANONICAL APPLICATION BREAKDOWN (16 LINES OF PURE XML)

Multi-State Notion-Style Kanban Board with Forms, Stores, and Reactive Wires

```
01 <APP ID="kanban-app" TITLE="Sprint 42" THEME="dark">
02 <DATA ID="tasks" SRC="/api/tasks.json" LOCAL="true" />
03 <STATE ID="board" FLOW="column">
04 <HEADER TITLE="Active Engineering Sprint" />
05 <FORM ACTION="create-task" SUBMIT-LABEL="Add Task">
06 <FIELD NAME="title" LABEL="Task" REQUIRED="true" />
07 <FIELD NAME="assignee" TYPE="select" SRC="team" />
08 </FORM>
09 <BOARD SRC="tasks" COLUMNS="todo,progress,done" />
10 </STATE>
11 <STATE ID="settings"> <FORM ACTION="save-cfg" /> </STATE>
12 <WIRE LABEL="create-task" PUSH="tasks" SET="status=todo" />
13 <WIRE LABEL="move-task" UPDATE="tasks" />
14 <WIRE LABEL="open-settings" GOTO="settings" />
15 </APP>
```

- 1
- 2
- 3
- 4
- 5
- 6

1 Root Sandbox Context (<APP>)

Mounts in-browser execution sandbox, registers global CSS theme tokens, and enforces CSP isolation.

2 Reactive Data Store (<DATA>)

In-memory observable state proxy. Automatically binds remote REST JSON or syncs with localStorage.

3 Finite State Machine (<STATE>)

Encapsulates view screens into isolated state machine nodes. Transitions occur without page reloads.

4 Declarative Forms & Validation (<FORM>)

Two-way binds input controls, evaluates schema constraints, and emits typed events on submission.

5 Self-Rendering Primitives (<BOARD>)

Specialized UI components with native drag-and-drop, automated column sorting, and event dispatch.

6 Reactive Event Mediator (<WIRE>)

Connects UI action intents to store mutations (PUSH, UPDATE, SET) and screen transitions with zero JS.

03 THE REACTIVE <WIRE> EVENT BUS & STATE PROPAGATION ENGINE

Deterministic Pub-Sub Pipeline · Zero JavaScript Glue Code

1. USER GESTURE Intent Trigger

```
<BUTTON ACTION="buy" />
```

User clicks control or submits form, broadcasting typed event payload on runtime bus.

2. WIRE MATCHER Rule Intercept

```
<WIRE LABEL="buy" ... />
```

Engine matches action label to registered <WIRE> declarations in sub-microsecond lookup.

3. MUTATION ENGINE Atomic Store Mutate

```
PUSH="cart" SET="qty++1"
```

Executes deterministic store operations (append, merge, set, toggle) on target <DATA> proxy.

4. REACTIVE DOM PAINT Fine-Grained Patch

```
<BADGE VALUE="{cart.qty}">
```

Targeted leaf DOM updates executed via fine-grained dependency graph with 0 VDOM diff overhead.

5. STATE MACHINE Screen Transition

```
GOTO="checkout-view"
```

Switches active <STATE> view in FSM, triggering lifecycle unmount/mount hooks cleanly.

04 APPSPEC CORE GRAMMAR MATRIX · 12 FOUNDATIONAL PRIMITIVES

Complete Tag Vocabulary for Application Structure, Reactive State, and UI

<APP>

Attrs: ID, TITLE, THEME, WIDTH

Top-level container. Mounts runtime sandbox and establishes CSS design tokens.

```
<APP ID="shop" THEME="dark">
```

SANDBOX

<STATE>

Attrs: ID, TITLE, FLOW, GAP

Discrete view screen in the finite state machine. Supports flex and grid flows.

```
<STATE ID="home" FLOW="column">
```

FSM SCREEN

<DATA>

Attrs: ID, SRC, LOCAL, AUTO

In-memory reactive proxy collection. Pulls from REST or binds to localStorage.

```
<DATA ID="items" SRC="/api" />
```

STORE

<WIRE>

Attrs: LABEL, PUSH, SET, GOTO

Routes action intents to store mutations or transitions state machine screens.

```
<WIRE LABEL="add" PUSH="items" />
```

EVENT BUS

<BOARD>

Attrs: SRC, COLUMNS, DRAG, TAG

Multi-lane workflow board with automatic drag-and-drop and status mutation.

```
<BOARD SRC="tasks" COLUMNS="..." />
```

KANBAN

<GRID>

Attrs: SRC, COLS, GAP, CARD

Responsive CSS repeating template. Renders product catalogs and cards.

```
<GRID SRC="prods" COLS="3" />
```

CATALOG

<FORM>

Attrs: ACTION, SUBMIT-LABEL, CLEAR

Two-way reactive form. Collects child <FIELD> values and emits typed payload.

```
<FORM ACTION="save" ...>
```

INPUT

<LIST>

Attrs: SRC, FILTER, SORT, LIMIT

Dynamic filtered list view with built-in search matching and pagination.

```
<LIST SRC="users" FILTER="active">
```

COLLECTION

<CANVAS>

Attrs: ID, ENGINE, TOOLS, BRUSH

Hardware-accelerated 2D/WebGL surface for diagrams and vector sketching.

```
<CANVAS ID="draw" ENGINE="2d" />
```

GRAPHICS

<TIMELINE>

Attrs: FPS, DURATION, AUDIO, LOOP

Keyframe animation sequencer and media player for kinetic presentations.

```
<TIMELINE DURATION="28s" />
```

ANIMATION

<ADATA>

Attrs: ID, WS, SSE, BUFFER

Real-time WebSocket & SSE event stream handler for live metrics and logs.

```
<ADATA ID="feed" WS="wss://..." />
```

TELEMETRY

<MODAL>

Attrs: ID, TITLE, TRIGGER, BLUR

Declarative modal dialog, alert sheet, and slide-over panel with focus trap.

```
<MODAL ID="detail" TRIGGER="view">
```

DIALOG

05 PRODUCTION ARCHITECTURAL RECIPES · DECLARATIVE PATTERNS

Zero-Boilerplate Implementations of Common Web Application Patterns

LIVE FILTER & SEARCH

Real-Time

```
<FIELD NAME="query" PLACEHOLDER="Search..." />
<WIRE ON="input:query"
  FILTER="products"
  BY="name" />
<GRID SRC="products.filtered" />
```

Instant sub-millisecond list filtering with automatic debounce on user keystrokes.

WIZARD STATE MACHINE

Multi-Step

```
<STATE ID="step-1">
  <FORM ACTION="goto-step2" />
</STATE>
<WIRE LABEL="goto-step2"
  SAVE="profile"
  GOTO="step-2" />
```

Multi-step form progression preserving state across screens without router boilerplate.

OPTIMISTIC REST MUTATION

Network Sync

```
<WIRE LABEL="add-item"
  PUSH="cart"
  SYNC="POST /api/cart"
  ROLLBACK="on-error" />
```

Instant optimistic UI update with background HTTP sync and automatic rollback on failure.

