@design.estate/dees-wcctools

build web component catalogs with viewport preview, property manipulation and precomposed views

- readme.md for @design.estate/dees-wcctools
- changelog.md for @design.estate/dees-wcctools

readme.md for @design.estate/dees-wcctools

□□ **Web Component Development Tools** — A powerful framework for building, testing, documenting, and recording web components

Overview

`@design.estate/dees-wcctools` provides a comprehensive development environment for web components, featuring:

- □□ **Interactive Component Catalogue** — Live preview with customizable sidebar sections
- □□ **Real-time Property Editing** — Modify component props on the fly with auto-detected editors
- □□ **Theme Switching** — Test light/dark modes instantly
- □□ **Responsive Viewport Testing** — Phone, phablet, tablet, and desktop views
- □□ **Screen Recording** — Record component demos with audio support and video trimming
- □□ **Advanced Demo Tools** — Post-render hooks for interactive testing
- □□ **Section-based Organization** — Group components into custom sections with filtering and sorting
- □□ **Zero-config Setup** — TypeScript and Lit support out of the box

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

Installation

```
# Using pnpm (recommended)
pnpm add -D @design.estate/dees-wcctools

# Using npm
npm install @design.estate/dees-wcctools --save-dev
```

Quick Start

1. Create Your Component

```
import { DeesElement, customElement, html, css, property } from '@design.estate/dees-element';

@customElement('my-button')
export class MyButton extends DeesElement {
  // Define a demo for the catalogue
  public static demo = () => html`
    <my-button .label=${'Click me!'} .variant=${'primary'}></my-button>
  `;

  @property({ type: String })
  accessor label: string = 'Button';

  @property({ type: String })
  accessor variant: 'primary' | 'secondary' = 'primary';

  public static styles = [
    css`
      :host {
        display: inline-block;
      }
      button {
        padding: 8px 16px;
        border-radius: 4px;
```

```
        border: none;
        cursor: pointer;
      }
      button.primary {
        background: #3b82f6;
        color: white;
      }
      button.secondary {
        background: #6b7280;
        color: white;
      }
    `
  ];

  public render() {
    return html`
      <button class="${this.variant}">${this.label}</button>
    `;
  }
}
```

2. Set Up Your Catalogue

```
// catalogue.ts
import { setupWccTools } from '@design.estate/dees-wcctools';
import { html } from 'lit';

// Import your components
import * as elements from './components/index.js';
import * as views from './views/index.js';
import * as pages from './pages/index.js';

// Initialize with sections-based configuration
setupWccTools({
  sections: [
    {
      name: 'Pages',
      type: 'pages',
```

```
      items: pages,
    },
    {
      name: 'Views',
      type: 'elements',
      items: views,
      icon: 'web',
    },
    {
      name: 'Elements',
      type: 'elements',
      items: elements,
      sort: ([a], [b]) => a.localeCompare(b),
    },
  ],
});
```

3. Create an HTML Entry Point

```
<!DOCTYPE html>
<html>
<head>
  <title>Component Catalogue</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body style="margin: 0; padding: 0;">
  <script type="module" src="./catalogue.js"></script>
</body>
</html>
```

□□ Sections Configuration

The sections-based API gives you full control over how components are organized in the sidebar.

Section Properties

Property	Type	Description
`name`	`string`	Display name for the section header
`type`	`'elements' | 'pages'`	How items render (`elements` show demos, `pages` render directly)
`items`	`Record<string, any>`	Object containing element classes or page factories
`filter`	`(name, item) => boolean`	Optional filter function to include/exclude items
`sort`	`([a, itemA], [b, itemB]) => number`	Optional sort function for ordering items
`icon`	`string`	Optional Material Symbols icon name
`collapsed`	`boolean`	Start section collapsed (default: `false`)

Advanced Example

```
import { setupWccTools } from '@design.estate/dees-wcctools';
import * as allElements from './elements/index.js';
import * as pages from './pages/index.js';

setupWccTools({
  sections: [
    {
      name: 'Pages',
      type: 'pages',
      items: pages,
    },
    {
      name: 'Form Controls',
      type: 'elements',
      items: allElements,
      icon: 'edit_note',
      filter: (name) => name.startsWith('form-') || name.includes('input'),
      sort: ([a], [b]) => a.localeCompare(b),
    },
    {
      name: 'Layout',
      type: 'elements',
```

```
      items: allElements,
      icon: 'dashboard',
      filter: (name) => name.startsWith('layout-') || name.startsWith('grid-'),
    },
    {
      name: 'Legacy',
      type: 'elements',
      items: allElements,
      filter: (name) => name.startsWith('legacy-'),
      collapsed: true,  // Start collapsed
    },
  ],
});
```

Legacy API (Still Supported)

```
// The old format still works for simple use cases
setupWccTools(elements, pages);
```

Features

□□ Live Property Editing

The properties panel automatically detects and allows editing of:

Property Type	Editor
String	Text input
Number	Number input
Boolean	Checkbox
Enum	Select dropdown
Object/Array	JSON editor modal

□□ Viewport Testing

Test your components across different screen sizes:

- **Phone** — 320px width
- **Phablet** — 600px width
- **Tablet** — 768px width
- **Desktop** — Full width (native)

□□ Theme Support

Components automatically adapt to light/dark themes. Use CSS custom properties with the theme manager:

```
import { cssManager } from '@design.estate/dees-element';

public static styles = [
  css`
    :host {
      color: ${cssManager.bdTheme('#1a1a1a', '#e5e5e5')};
      background: ${cssManager.bdTheme('#ffffff', '#0a0a0a')};
    }
  `
];
```

□□ Screen Recording

Record component demos directly from the catalogue:

- **Viewport Recording** — Record just the component viewport
- **Full Screen Recording** — Capture the entire screen
- **Audio Support** — Add microphone commentary with live level monitoring
- **Video Trimming** — Trim start/end before export with visual timeline
- **WebM Export** — High-quality video output

Click the red record button in the bottom toolbar to start.

□□ Demo Tools

The demotools module provides enhanced testing capabilities with `dees-demowrapper`:

```
import '@design.estate/dees-wcctools/demotools';
```

```
@customElement('my-component')
export class MyComponent extends DeesElement {
  public static demo = () => html`
    <dees-demowrapper .runAfterRender=${async (wrapper) => {
      // Find elements using standard DOM APIs
      const myComponent = wrapper.querySelector('my-component');

      // Simulate user interactions
      myComponent.value = 'Test value';
      await myComponent.updateComplete;

      // Work with multiple elements
      wrapper.querySelectorAll('.item').forEach((el, i) => {
        console.log(`Item ${i}:`, el.textContent);
      });
    }}>
      <my-component></my-component>
      <div class="item">Item 1</div>
      <div class="item">Item 2</div>
    </dees-demowrapper>
  `;
}
```

□□ Multiple Demos

Components can expose multiple demo variations:

```
@customElement('my-button')
export class MyButton extends DeesElement {
  public static demo = [
    () => html`<my-button variant="primary">Primary</my-button>`,
    () => html`<my-button variant="secondary">Secondary</my-button>`,
    () => html`<my-button variant="danger">Danger</my-button>`,
  ];
}
```

Each demo appears as a numbered item in an expandable folder in the sidebar.

□ Async Demos

Return a `Promise` from `demo` for async setup:

```
public static demo = async () => {
  const data = await fetchSomeData();
  return html`<my-component .data=${data}></my-component>`;
};
```

Container Queries

Components can respond to their container size using the `wccToolsViewport` container:

```
public static styles = [
  css`
    @container wccToolsViewport (min-width: 768px) {
      :host {
        flex-direction: row;
      }
    }

    @container wccToolsViewport (max-width: 767px) {
      :host {
        flex-direction: column;
      }
    }
  `
];
```

Component Guidelines

Required for Catalogue Display

1. Components must expose a static `demo` property returning a Lit template (or array of templates)
2. Use `@property()` decorators with the `accessor` keyword for editable properties
3. Export component classes for proper detection

Best Practices

```
@customElement('best-practice-component')
export class BestPracticeComponent extends DeesElement {
  // ✅ Static demo property (single or array)
  public static demo = () => html`
    <best-practice-component
      .complexProp=${{ key: 'value' }}
      simpleAttribute="test"
    ></best-practice-component>
  `;

  // ✅ Typed properties with defaults (TC39 decorators)
  @property({ type: String })
  accessor title: string = 'Default Title';

  // ✅ Complex property without attribute
  @property({ attribute: false })
  accessor complexProp: { key: string } = { key: 'default' };

  // ✅ Enum with proper typing
  @property({ type: String })
  accessor variant: 'small' | 'medium' | 'large' = 'medium';
}
```

URL Routing

The catalogue uses URL routing for deep linking:

```
/wcctools-route/:sectionName/:itemName/:demoIndex/:viewport/:theme

Examples:
/wcctools-route/Elements/my-button/0/desktop/dark
/wcctools-route/Views/view-dashboard/0/tablet/bright
/wcctools-route/Pages/home/0/desktop/dark
```

API Reference

`setupWccTools(config)`

Initialize the WCC Tools dashboard with sections configuration.

```
interface IWccSection {
  name: string;
  type: 'elements' | 'pages';
  items: Record<string, any>;
  filter?: (name: string, item: any) => boolean;
  sort?: (a: [string, any], b: [string, any]) => number;
  icon?: string;
  collapsed?: boolean;
}

interface IWccConfig {
  sections: IWccSection[];
}

setupWccTools(config: IWccConfig): void;

// Legacy (still supported)
setupWccTools(elements: Record<string, any>, pages?: Record<string, TTemplateFactory>): void;
```

`DeesDemoWrapper`

Component for wrapping demos with post-render logic.

Property	Type	Description
`runAfterRender`	`(wrapper) => void | Promise<void>`	Callback after wrapped elements render

The wrapper provides full DOM API access:

- `wrapper.querySelector()` — Find single element
- `wrapper.querySelectorAll()` — Find multiple elements
- `wrapper.children` — Access child elements directly

Recording Components (Advanced)

For custom recording integrations:

```
import { RecorderService } from '@design.estate/dees-wcctools';

const recorder = new RecorderService({
  onDurationUpdate: (duration) => console.log(`${duration}s`),
  onRecordingComplete: (blob) => console.log('Recording done!', blob),
  onAudioLevelUpdate: (level) => console.log(`Audio: ${level}%`),
});

await recorder.startRecording({ mode: 'viewport' });
// ... later
recorder.stopRecording();
```

Project Structure

```
my-component-library/
├── src/
│   ├── elements/          # UI components
│   │   ├── my-button.ts
│   │   ├── my-card.ts
│   │   └── index.ts
│   ├── views/             # Full-page layouts
│   │   ├── view-dashboard.ts
│   │   └── index.ts
│   ├── pages/             # Documentation pages
│   │   ├── home.ts
│   │   └── index.ts
│   └── catalogue.ts       # WCC Tools setup
├── html/
│   └── index.html
└── package.json
```

Browser Support

- Chrome/Edge (latest)
- Firefox (latest)
- Safari (latest)
- Mobile browsers with Web Components support

License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the LICENSE file.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

changelog.md for @design.estate/dees-wcctools

2026-01-27 - 3.8.0 - feat(sidebar)

rename demoGroup to demoGroups, add multi-group support, search by group name, and context menu group navigation

- Static property demoGroup renamed to demoGroups; accepts string | string[] for multi-group membership
- Elements with multiple demoGroups appear in each group's sidebar section simultaneously and show the library_books icon instead of featured_video
- Sidebar search now matches group names in addition to element names; groups are sorted alphabetically by group name
- Context menu for elements includes a "Show in Group:" section with navigable group entries that scroll to and briefly highlight the target group; group headers also have a context menu entry to filter by that group
- Added data-group attribute on .item-group for DOM querying and visual classes for group highlight and filter match
- Updated test elements to use demoGroups and updated docs/changelog/readme.hints to document the new behavior
- Bumped several devDependencies (@api.global/typedserver, @git.zone/tsbuild, @git.zone/tsbundle, @git.zone/tstest, @git.zone/tswatch, @types/node) and adjusted npm script `watch` to use tswatch

2026-01-27 - 3.8.0 - feat(sidebar)

rename demoGroup to demoGroups, add multi-group support, search by group name, and context menu group navigation

- Rename static property `demoGroup` to `demoGroups` on element classes; accepts `string | string[]` for multi-group membership
- Elements with an array of `demoGroups` appear in each group's sidebar section simultaneously
- Search now matches group names in addition to element names (e.g. searching "Buttons" shows all elements in the Buttons group)
- Groups are sorted alphabetically by group name instead of by first element name
- Elements belonging to multiple groups display a `library_books` icon instead of `featured_video`
- Right-click context menu on elements with groups shows "Show in Group:" section with navigable group entries
- Clicking a group name in the context menu scrolls to and briefly highlights that group in the sidebar
- Updated test elements (test-button-primary, test-button-secondary, test-button-danger, test-input-text, test-input-checkbox) to use `demoGroups`

2026-01-04 - 3.7.1 - fix(sidebar)

increase scrolled sidebar header box-shadow intensity and size to improve visual separation

- Changed .sidebar-header.scrolled box-shadow from `0 4px 12px -2px rgba(0, 0, 0, 0.4)` to `0 8px 24px -2px rgba(0, 0, 0, 1)`
- File modified: ts_web/elements/wcc-sidebar.ts — stronger, larger, and fully opaque shadow for better contrast when scrolled

2026-01-04 - 3.7.0 - feat(wcc-sidebar)

add header shadow and scrolled state for sidebar menu to show elevation when content is scrolled

- Introduce isMenuScrolled state to track whether the menu has been scrolled
- Add handleMenuScroll handler and bind it to the menu scroll event
- Apply a 'scrolled' class to .sidebar-header to add box-shadow and border-bottom color with transitions
- Update template to conditionally add scrolled class and attach scroll listener

2026-01-04 - 3.6.2 - fix(wcc-sidebar)

use sidebar's internal .menu element for scroll management and expose scrollableContainer getter

- Add public scrollableContainer getter to wcc-sidebar that returns the .menu element for external scroll control
- Update wcc-dashboard to query wcc-sidebar as WccSidebar and attach scroll listeners to sidebar.scrollableContainer instead of the host element
- Restore sidebar scroll position by setting scrollTop on the scrollableContainer when applying saved positions
- TypeScript casting added to avoid nullable/implicit any issues when querying the sidebar element

2026-01-04 - 3.6.1 - fix(wcc-sidebar)

sort sidebar items alphabetically and unify grouped and ungrouped items for consistent ordering

- Unifies ungrouped elements and groups into a single render list using a RenderItem type with a sortKey.
- Sorts all top-level items alphabetically by element name (case-insensitive via toLowerCase) so sidebar order is deterministic.
- Groups are ordered by the first element's name; individual items inside a group preserve their original order.
- Replaces previous separate rendering paths with a single sorted render pass that returns TemplateResult array.

2026-01-04 - 3.6.0 - feat(sidebar)

restructure sidebar layout, add search clear button, and improve scrolling behavior

- Change sidebar root to a flex column layout and add a .sidebar-header to separate header content from the scrollable menu
- Move pinned section into the header and make .menu flex: 1 with min-height: 0 so the menu becomes the scrollable area

- Replace overflow-y on the root with overflow:hidden to avoid double scrolling and constrain scrolling to .menu
- Add a clear button for the search input (.search-clear) with positioning, hover styles, and a clearSearch() method to reset the query and emit searchChanged
- Adjust search input padding and make .search-container position: relative to correctly position the clear button

2026-01-04 - 3.5.3 - fix(deps)

bump dependency versions: @design.estate/dees-domtools to ^2.3.7, @design.estate/dees-element to ^2.1.5, lit to ^3.3.2; update devDependencies @api.global/typedserver to ^8.1.0 and @git.zone/tstest to ^3.1.4

- Updated runtime dependencies: @design.estate/dees-domtools ^2.3.7, @design.estate/dees-element ^2.1.5, lit ^3.3.2.
- Updated devDependencies: @api.global/typedserver ^8.1.0 (major bump), @git.zone/tstest ^3.1.4.
- @api.global/typedserver major bump affects development tooling only (devDependency), not runtime API.
- Current package version is 3.5.2; recommend a patch release to 3.5.3.

2026-01-04 - 3.5.2 - fix(elements)

delay hiding sidebar and properties panels during native-mode transition and use transparent rgba border for frame to avoid layout jumps

- Add isHidden state to wcc-sidebar and wcc-properties and switch display bindings to use isHidden instead of directly using isNative
- Introduce a 300ms delayed hide when entering native mode so UI hides after frame animation completes; show immediately when exiting native mode
- Replace hardcoded hex border values in wcc-frame with rgba and set native border to a transparent 0px to prevent abrupt visual jumps

2026-01-04 - 3.5.1 - fix(sidebar)

disable frame CSS transition while user is resizing the sidebar to prevent janky animations

- Added isResizing boolean property to wcc-frame to toggle transitions during resize
- Set frame.isResizing = true at resize start and false on mouseup to re-enable transitions

- Updated CSS to skip transition while isResizing is true
- Files changed: ts_web/elements/wcc-frame.ts, ts_web/elements/wcc-sidebar.ts

2026-01-04 - 3.5.0 - feat(wcctools)

add context menu and pinning support, persist pinned state in URL, and add grouped demo test elements

- Add wcc-contextmenu custom element with a static show() API, proper positioning, visibility transitions, outside-click and Escape handling, and menu item actions.
- Introduce pinnedItems (Set) on wcc-dashboard and wcc-sidebar; pass pinnedItems to the sidebar, handle pinnedChanged events, and persist pinned item keys in the URL query param 'pinned'. Changes include defensive updates to avoid unnecessary update loops.
- Enhance wcc-sidebar to render pinned state: new styles for pinned items and pinned sections, contextmenu integration for element items, adjusted layout (grid-template-columns) and improved element/demo rendering logic.
- Add grouped demo test components and exports to demo the demoGroup feature: test-button-primary, test-button-secondary, test-button-danger, test-input-text, and test-input-checkbox.
- Misc: adjust dashboard URL state serialization/deserialization to include pinned items and ensure scroll/search state handling remains stable.

2025-12-30 - 3.4.0 - feat(sidebar)

add searchable sidebar with URL-backed query state and highlighted matches

- Add search input to wcc-sidebar and expose a searchQuery property
- Filter sidebar sections and items client-side based on the search query and hide sections with no matches
- Highlight matching substrings in sidebar item labels
- Emit a 'searchChanged' event from the sidebar and handle it in wcc-dashboard to keep dashboard.searchQuery in sync
- Persist the search query in the route query parameter 'search' when building URLs and restore/clear it on navigation
- Preserve existing scroll-state handling while adding search state to URL updates

2025-12-28 - 3.3.0 - feat(wcctools)

Add section-based configuration API for setupWccTools, new Views, and section-aware routing/sidebar

- Introduce IWccSection and IWccConfig types and migrate setupWccTools to accept a sections config while preserving legacy (elements, pages) format
- WccDashboard and WccSidebar updated to support sections, filtering, sorting, collapsed sections, and section-aware URL routing (uses sectionName in routes with legacy fallbacks)
- Add Views: view-dashboard, view-settings, view-empty-state plus test/views index exports and demo variations
- Add helpers: getSectionItems, convertLegacyToConfig and isWccConfig; update build URL and routing logic to be section-aware
- Update docs and README/readme.hints with sections API, examples, migration notes and UI/UX updates

2025-12-22 - 3.2.0 - feat(wcc-sidebar)

auto-expand sidebar folder when selecting an element with multiple demos

- Add updated() lifecycle to auto-expand folder when selectedItem changes
- Find element name by matching selectedItem against dashboardRef.elements
- Only auto-expand when the element has multiple demos (checks item.demo and hasMultipleDemos)
- Immutably update expandedElements set to trigger re-render and avoid duplicate additions

2025-12-21 - 3.1.2 - fix(wcc-properties)

Use LitElement.updated to recreate properties only when selectedItem changes and handle errors; remove custom scheduleUpdate implementation

- Replaced public async scheduleUpdate() with protected updated(changedProperties) lifecycle method
- Call super.updated(...) and only recreate properties when selectedItem changed to avoid unnecessary work

- Preserve error handling and clear propertyContent on failure
- Removed explicit super.scheduleUpdate() call to rely on LitElement's update lifecycle

2025-12-21 - 3.1.1 - fix(wcc-properties)

Improve wcc-properties CSS to prevent grid overflow, properly size and center icon glyphs, and adjust right-side offset

- Use minmax(0, 1fr) for grid-template-columns (selectorButtons1/2/4/5) to avoid flexbox overflow and ensure consistent column sizing
- Add min-width/min-height and inline-flex centering to .material-symbols-outlined to stabilize icon sizing and vertical/horizontal alignment
- Increase right calc offset from 520px to 600px to accommodate wider content/controls
- Changes applied in ts_web/elements/wcc-properties.ts

2025-12-19 - 3.1.0 - feat(wcc-properties)

add Share selector with inline Record button and adjust properties panel grid layout

- Increase rightmost column width from 70px to 100px in properties grid
- Introduce .shareSelector and .selectorButtons1 CSS classes and markup
- Replace with inline Record button that toggles icon based on isRecording
- Add Share panel heading and integrate record control into selector area

2025-12-19 - 3.0.0 - BREAKING CHANGE(ts_web)

Replace fullscreen boolean with native viewport mode across components, add native viewport selector and toggle, and update dev deps and npmextra config

- Introduce isNative (derived from selectedViewport === 'native') replacing isFullscreen on WccDashboard, WccFrame, WccProperties and WccSidebar (property and styling changes).
- WccDashboard: remove isFullscreen property, add isNative getter, implement toggleNative to switch selectedViewport between 'native' and 'desktop', update ESC handler to exit native mode and call buildUrl().
- WccProperties: add Native button to viewport selector, adjust grid columns and selector layout, replace fullscreen toggle with toggleNative (dispatches 'toggleNative' event).
- WccFrame: rename isFullscreen -> isNative and update style/markup branching to use isNative.
- WccSidebar: visibility now driven by isNative instead of isFullscreen.
- API/event changes: 'toggleFullscreen' event renamed to 'toggleNative' — this is an incompatible change for consumers relying on the old event/property.
- package.json: bump devDependencies @git.zone/tsbuild -> ^4.0.2, @git.zone/tsrun -> ^2.0.1, @git.zone/tswatch -> ^2.3.13, @types/node -> ^25.0.3.
- npmextra.json: rename keys (e.g. "gitzone" -> "@git.zone/cli", "tsdoc" -> "@git.zone/tsdoc"), add @ship.zone/szci entry and add release registries/accessLevel for @git.zone/cli.

2025-12-11 - 2.0.1 - fix(@git.zone/tswatch)

Bump @git.zone/tswatch devDependency to ^2.3.12

- Updated devDependency @git.zone/tswatch from ^2.3.11 to ^2.3.12 in package.json

2025-12-11 - 2.0.0 - BREAKING CHANGE(recorder)

Remove FFmpeg-based MP4 conversion; simplify recorder/export to WebM and improve recorder/editor robustness

- Removed FFmpegService and all client-side MP4 conversion logic — exports are now WebM-only (MP4 conversion and related UI/controls removed).
- ts_web/elements/wcc-recording-panel: dropped outputFormat and conversion states/UI; download flow simplified to always export WebM.
- ts_web/index.ts: removed FFmpegService exports and conversion types from public API.

- package.json: removed @ffmpeg/* dependencies.
- RecorderService: handleRecordingComplete is now async and fixes recorded blob assignment and cleanup timing.
- wcc-properties: improved element detection and robustness — recursive search through light/shadow DOM with retry/delay, plus an advanced JSON editor for Object/Array props (supports multiple open editors and frame resize events).
- wcc-sidebar: force re-render after selecting demos to ensure child demo selection indicators update correctly.
- dees-demowrapper: ensure slotted content is rendered before calling runAfterRender (small timing/stability improvements).
- Test update: demo definitions can be arrays (multiple demos) — test-demoelement updated to use multiple demo entries.

2025-12-11 - 1.3.0 - feat(recording-panel)

Add demo wrapper utilities, improve recording trim behavior, and harden property panel element detection; update documentation

- Add dees-demowrapper (ts_demotools) with runAfterRender callback to run post-render demo logic (supports async callbacks).
- Improve recording UI and trimming: handle WebM files with Infinity/NaN durations by falling back to tracked recording duration; replace numeric handle positioning with CSS calc strings for responsive trim handles.
- Harden property extraction: implement recursive element search (including shadowRoots), add an initial delay and retry loop to wait for demo rendering, and add an advanced JSON editor for Object/Array properties with open/save/cancel and per-editor error reporting.
- Add and expand documentation: new ts_web/ and ts_demotools/ READMEs, reorganized main README with clearer feature list, usage examples, and API reference.
- Minor exports and module/docs housekeeping (index exports, readme reorder, examples updated to import classes).

2025-11-16 - 1.2.1 - fix(dependencies)

Bump dependencies and developer tooling versions

- Bump @design.estate/dees-domtools from ^2.3.3 to ^2.3.6
- Bump @design.estate/dees-element from ^2.1.2 to ^2.1.3
- Upgrade @git.zone/tsbuild from ^2.6.8 to ^2.7.1
- Upgrade @git.zone/tsrun from ^1.2.44 to ^1.6.2
- Upgrade @git.zone/tstest from ^2.3.8 to ^2.7.0

2025-09-19 - 1.2.0 - feat(wcc-properties)

Add advanced property editors, recursive element detection, demo wrapper, UI refresh and test fixtures

- Advanced JSON property editor: multiple side-by-side editors with save/cancel, syntax validation and inline error display; editors affect frame layout (frame bottom increases when editors open).
- Improved properties panel element detection: recursive search through nested children and shadow DOM, initial delay and retry mechanism to handle async Lit rendering.
- Add dees-demowrapper component in ts_demotools to run post-render callbacks and support async demo setup and DOM access for demos.
- UI refresh with shadcn-like styles: CSS variables for theming, redesigned properties panel and sidebar, improved form controls, theme and viewport selectors.
- Viewport and frame improvements: responsive padding based on viewport type, theme-aware background rendering, and scroll position tracking with URL/state restoration for frame and sidebar.
- Add test fixtures and demo elements/pages under test/ to exercise properties, complex types, nested elements and scroll restoration; include node test for resolveTemplateFactory.
- Expose setupWccTools entry point and plugin wiring (wcctools.plugins exports for dees-domtools and smartdelay) for easier integration.

2025-06-27 - 1.1.0 - feat(wcctools)

Enhance component tools with an advanced property editor, improved element detection and modernized UI styling for a more responsive dashboard experience.

- Updated documentation and in-code hints with new shadcn-like design patterns for the dashboard UI.
- Introduced an advanced complex properties editor supporting JSON validation and multi-editor handling.

- Refined recursive element search in the properties panel to improve asynchronous rendering detection.
- Expanded test coverage with scenarios for edge cases, nested elements and wrapper components.

2025-06-26 - 1.0.101 - fix(wcc-dashboard)

Improve scroll listener management and add new test pages

- Removed the pages/ directory entry from .gitignore to allow test pages to be tracked
- Added new test pages: page1 and pageLongScroll for enhanced scroll and navigation testing
- Refactored wcc-dashboard: changed scroll position properties to private variables and added a flag to prevent duplicate scroll listener attachment

2025-06-26 - 1.0.100 - fix(wcc-dashboard)

Prevent duplicate application of scroll positions in dashboard to avoid interfering with user scrolling

- Added a private 'scrollPositionsApplied' property to track if scroll positions have already been applied
- Introduced a guard in the applyScrollPositions method to ensure the scroll state is applied only once

2025-06-26 - 1.0.99 - fix(dashboard)

Fix scroll state preservation in dashboard by tracking frame and sidebar scroll positions and updating the URL accordingly.

- Added frameScrollY and sidebarScrollY properties to capture scroll positions.

- Set up scroll listeners on wcc-frame and wcc-sidebar to update scroll state.
- Implemented debounced updates to modify the URL with current scroll positions without navigation.
- Restored scroll positions from URL query parameters during initialization.

2025-06-16 - 1.0.97 - properties-panel

- Improve element detection timing and value handling in properties panel

2025-06-16 - 1.0.96 - properties-panel

- Enhance element detection and error handling for nested structures

2025-06-16 - 1.0.95 - package

- Correct path for demotools export in package.json

2025-06-16 - 1.0.94 - demotools

- Enhance runAfterRender to provide full DOM API access and improve element selection

2025-06-16 - 1.0.92 - demotools

- Update DeesDemoWrapper to handle multiple slotted elements in runAfterRender callback

2025-06-16 - 1.0.91 - readme

- Update documentation with comprehensive overview, quick start guide, and detailed feature descriptions

2025-06-16 - 1.0.90 - demo/properties/refactor

- Add DeesDemoWrapper component for enhanced demo element handling
- Enhance element detection in properties panel with recursive search and retry mechanism
- Refactor code structure for improved readability and maintainability

2024-05-06 to 2020-05-10 - 1.0.89–1.0.17 - core

- Over a series of releases, trivial core fixes and updates were applied.
- (Note: Version 1.0.87 also included an update to the documentation.)