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.

Revision #6
Created 2026-03-28 10:49:11 UTC by foss.global Team
Updated 2026-03-28 12:14:34 UTC by foss.global Team