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

A comprehensive web components library built with TypeScript and LitElement, providing **90+ production-ready UI components** for building modern web applications with consistent design and behavior. □□

TypeScript LitElement

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.

□ Features

- □□ **Consistent Design System** — Beautiful, cohesive components following modern UI/UX principles
- □□ **Dark/Light Theme Support** — All components automatically adapt to your theme
- ⌨ **Keyboard Accessible** — Full keyboard navigation and ARIA support
- □□ **Responsive** — Mobile-first design that works across all screen sizes
- □□ **TypeScript-First** — Fully typed APIs with excellent IDE support
- □□ **Modular** — Use only what you need, tree-shakeable architecture
- □□ **Full App Shell** — `dees-appui` provides a complete application framework with menus, routing, activity log, and bottom bar
- □□ **Media Components** — Rich tile-based previews for PDFs, images, audio, video, notes, and folders
- □□ **IDE Workspace** — Full workspace component with Monaco editor, file tree, terminal, and diff viewer

Installation

```
npm install @design.estate/dees-catalog
# or
pnpm add @design.estate/dees-catalog
```

Quick Start

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

@customElement('my-app')
class MyApp extends DeesElement {
  render() {
    return html`
      <dees-button type="highlighted" @click=${() => alert('Hello!')}>
        Click me!
      </dees-button>
    `;
  }
}
```

Development Guide

For developers working on this library, please refer to the UI Components Playbook for comprehensive patterns, best practices, and architectural guidelines.

Components Overview

Category	Components
Core UI	`DeesButton`, `DeesButtonExit`, `DeesButtonGroup`, `DeesBadge`, `DeesChips`, `DeesHeading`, `DeesHint`, `DeesIcon`, `DeesLabel`, `DeesPanel`, `DeesSearchbar`, `DeesSpinner`, `DeesToast`, `DeesWindowcontrols`, `DeesActionbar`

Category	Components
Forms	`DeesForm`, `DeesInputText`, `DeesInputCheckbox`, `DeesInputDropdown`, `DeesInputRadiogroup`, `DeesInputFileupload`, `DeesInputIban`, `DeesInputPhone`, `DeesInputQuantitySelector`, `DeesInputMultitoggle`, `DeesInputToggle`, `DeesInputTags`, `DeesInputTypelist`, `DeesInputList`, `DeesInputProfilepicture`, `DeesInputRichtext`, `DeesInputWysiwyg`, `DeesInputDatepicker`, `DeesInputSearchselect`, `DeesInputCode`, `DeesFormSubmit`
App Shell (Layout)	`DeesAppui`, `DeesAppuiMainmenu`, `DeesAppuiSecondarymenu`, `DeesAppuiMaincontent`, `DeesAppuiAppbar`, `DeesAppuiActivitylog`, `DeesAppuiBottombar`, `DeesAppuiProfiledropdown`, `DeesAppuiTabs`, `DeesMobileNavigation`, `DeesDashboardGrid`
Data Display	`DeesTable`, `DeesDataviewCodebox`, `DeesDataviewStatusobject`, `DeesPdf`, `DeesStatsGrid`, `DeesPagination`
Media & Tiles	`DeesTilePdf`, `DeesTileImage`, `DeesTileAudio`, `DeesTileVideo`, `DeesTileNote`, `DeesTileFolder`, `DeesPreview`, `DeesPdfViewer`, `DeesPdfPreview`, `DeesImageViewer`, `DeesAudioViewer`, `DeesVideoViewer`
Visualization	`DeesChartArea`, `DeesChartLog`
Dialogs & Overlays	`DeesModal`, `DeesContextmenu`, `DeesSpeechbubble`, `DeesWindowlayer`
Navigation	`DeesStepper`, `DeesProgressbar`
Workspace / IDE	`DeesWorkspace`, `DeesWorkspaceMonaco`, `DeesWorkspaceDiffEditor`, `DeesWorkspaceFiletree`, `DeesWorkspaceTerminal`, `DeesWorkspaceTerminalPreview`, `DeesWorkspaceMarkdown`, `DeesWorkspaceMarkdownoutlet`, `DeesWorkspaceBottombar`
Theming	`DeesTheme`
Pre-built Templates	`DeesSimpleAppdash`, `DeesSimpleLogin`
Shopping	`DeesShoppingProductcard`

□□ Detailed Component Documentation

Core UI Components

`DeesButton`

A versatile button component supporting multiple styles and states.

```
// Basic usage
const button = document.createElement('dees-button');
button.text = 'Click me';

// With options
<dees-button
  type="highlighted"  // Options: normal, highlighted, discreet
  status="pending"    // Options: normal, pending, success, error
  disabled={false}    // Optional: disables the button
  @click=${handleClick}
>Click me</dees-button>
```

DeesBadge

Display status indicators or counts with customizable styles.

```
<dees-badge
  type="success"  // Options: default, primary, success, warning, error
  text="New"      // Text to display
  rounded        // Optional: applies rounded corners
></dees-badge>
```

DeesChips

Interactive chips/tags with selection capabilities.

```
<dees-chips
  selectionMode="multiple"  // Options: none, single, multiple
  chipsAreRemovable        // Optional: allows removing chips
  .selectableChips=${[
    { key: 'tag1', value: 'Important' },
    { key: 'tag2', value: 'Urgent' }
  ]}
  @selection-change=${handleSelection}
></dees-chips>
```

DeesIcon

Display icons from FontAwesome and Lucide icon libraries with library prefixes.

```
// FontAwesome icons — use 'fa:' prefix
<dees-icon
  icon="fa:check"       // FontAwesome icon with fa: prefix
  iconSize="24"         // Size in pixels
  color="#22c55e"       // Optional: custom color
></dees-icon>

// Lucide icons — use 'lucide:' prefix
<dees-icon
  icon="lucide:menu"    // Lucide icon with lucide: prefix
  iconSize="24"         // Size in pixels
  color="#007bff"       // Optional: custom color
  strokeWidth="2"       // Optional: stroke width for Lucide icons
></dees-icon>

// Legacy API (deprecated but still supported)
<dees-icon
  iconFA="check"        // Without prefix — assumes FontAwesome
></dees-icon>
```

`DeesLabel`

Text label component with optional icon and status indicators.

```
<dees-label
  text="Status"         // Label text
  icon="info-circle"    // Optional: icon name
  type="info"           // Options: default, info, success, warning, error
  size="medium"         // Options: small, medium, large
></dees-label>
```

`DeesSpinner`

Loading indicator with customizable appearance.

```
<dees-spinner
  size="medium"         // Options: small, medium, large
  type="primary"        // Options: primary, secondary, light, dark
  overlay              // Optional: adds a full-screen overlay
></dees-spinner>
```

DeesToast

Notification toast messages with various styles, positions, and auto-dismiss functionality.

```
// Programmatic usage
DeesToast.show({
  message: 'Operation successful',
  type: 'success',      // Options: info, success, warning, error
  duration: 3000,       // Time in milliseconds before auto-dismiss
  position: 'top-right' // Options: top-right, top-left, bottom-right, bottom-left, top-center, bottom-center
});

// Convenience methods
DeesToast.info('Information message');
DeesToast.success('Success message');
DeesToast.warning('Warning message');
DeesToast.error('Error message');

// Advanced control
const toast = await DeesToast.show({
  message: 'Processing...',
  type: 'info',
  duration: 0  // No auto-dismiss
});

// Later dismiss programmatically
toast.dismiss();
```

Key Features:

- Multiple toast types with distinct icons and colors
- 6 position options for flexible placement
- Auto-dismiss with visual progress indicator
- Manual dismiss by clicking
- Smooth animations and transitions
- Automatic stacking of multiple toasts
- Theme-aware styling
- Programmatic control

DeesButtonExit

Exit/close button component with consistent styling.

```
<dees-button-exit
  @click=${handleClose}
></dees-button-exit>
```

DeesButtonGroup

Container for grouping related buttons together.

```
<dees-button-group
  .buttons=${[
    { text: 'Save', type: 'highlighted', action: handleSave },
    { text: 'Cancel', type: 'normal', action: handleCancel }
  ]}
  spacing="medium"    // Options: small, medium, large
></dees-button-group>
```

DeesHeading

Consistent heading component with level and styling options.

```
<dees-heading
  level={1}           // 1-6 for H1-H6
  text="Page Title"
  .subheading=${'Optional subtitle'}
  centered           // Optional: center alignment
></dees-heading>
```

DeesHint

Hint/tooltip component for providing contextual help.

```
<dees-hint
  text="This field is required"
  type="info"        // Options: info, warning, error, success
  position="top"     // Options: top, bottom, left, right
></dees-hint>
```

DeesPanel

Container component for grouping related content with optional title and actions.

```
<dees-panel
  .title=${'Panel Title'}
  .subtitle=${'Optional subtitle'}
  collapsible        // Optional: allow collapse/expand
  collapsed={false}  // Initial collapsed state
  .actions=${[
    { icon: 'settings', action: handleSettings }
  ]}
>
  <!-- Panel content -->
</dees-panel>
```

DeesSearchbar

Search input component with suggestions and search handling.

```
<dees-searchbar
  placeholder="Search..."
  .suggestions=${['item1', 'item2', 'item3']}
  showClearButton    // Show clear button when has value
  @search=${handleSearch}
  @suggestion-select=${handleSuggestionSelect}
></dees-searchbar>
```

DeesWindowcontrols

Window control buttons (minimize, maximize, close) for desktop-like applications.

```
<dees-windowcontrols
  .controls=${['minimize', 'maximize', 'close']}
  @minimize=${handleMinimize}
  @maximize=${handleMaximize}
  @close=${handleClose}
></dees-windowcontrols>
```

DeesActionbar

Floating action bar for contextual actions.

```
<dees-actionbar
  .actions=${[
```

```
    { icon: 'lucide:save', label: 'Save', action: () => handleSave() },
    { icon: 'lucide:trash', label: 'Delete', action: () => handleDelete() }
  ]}
></dees-actionbar>
```

Form Components

`DeesForm`

Container component for form elements with built-in validation and data handling.

```
<dees-form
  @formData=${(e) => handleFormData(e.detail)}  // Emitted when form is submitted
  @formValidation=${(e) => handleValidation(e.detail)}  // Emitted during validation
>
  <dees-input-text required></dees-input-text>
  <dees-form-submit>Submit</dees-form-submit>
</dees-form>
```

`DeesInputText`

Text input field with validation and formatting options.

```
<dees-input-text
  key="email"           // Unique identifier for form data
  label="Email"         // Input label
  value="initial@value.com"  // Initial value
  required             // Makes the field required
  disabled            // Disables the input
  placeholder="Enter your email"
></dees-input-text>
```

`DeesInputCheckbox`

Checkbox input component for boolean values.

```
<dees-input-checkbox
  key="terms"
  label="Accept Terms"
  checked             // Initial checked state
```

```
  required
  @change=${handleChange}
></dees-input-checkbox>
```

DeesInputToggle

Toggle switch component for boolean on/off states.

```
<dees-input-toggle
  key="darkMode"
  label="Enable Dark Mode"
  .value=${true}
  @change=${handleToggle}
></dees-input-toggle>
```

DeesInputDropdown

Dropdown selection component with search and filtering capabilities.

```
<dees-input-dropdown
  key="country"
  label="Select Country"
  .options=${[
    { key: 'us', option: 'United States' },
    { key: 'uk', option: 'United Kingdom' }
  ]}
  searchable          // Enables search functionality
  multiple           // Allows multiple selections
></dees-input-dropdown>
```

DeesInputFileupload

File upload component with drag-and-drop support.

```
<dees-input-fileupload
  key="documents"
  label="Upload Files"
  multiple            // Allow multiple file selection
  accept=".pdf,.doc"  // Accepted file types
  maxSize="5MB"      // Maximum file size
  @upload=${handleUpload}
```

```
></dees-input-fileupload>
```

DeesInputIban

Specialized input for IBAN (International Bank Account Number) with validation.

```
<dees-input-iban
  key="bankAccount"
  label="IBAN"
  country="DE"        // Default country format
  required
  @validate=${handleIbanValidation}
></dees-input-iban>
```

DeesInputPhone

Phone number input with country code selection and formatting.

```
<dees-input-phone
  key="phone"
  label="Phone Number"
  defaultCountry="US"  // Default country code
  required
  @validate=${handlePhoneValidation}
></dees-input-phone>
```

DeesInputQuantitySelector

Numeric input with increment/decrement controls.

```
<dees-input-quantity-selector
  key="quantity"
  label="Quantity"
  min="0"             // Minimum value
  max="100"           // Maximum value
  step="1"            // Increment/decrement step
  value="1"           // Initial value
></dees-input-quantity-selector>
```

DeesInputMultitoggle

Multi-state toggle button group.

```
<dees-input-multitoggle
  key="status"
  label="Status"
  .options=${[
    { key: 'active', label: 'Active' },
    { key: 'pending', label: 'Pending' },
    { key: 'inactive', label: 'Inactive' }
  ]}
  value="active"      // Initial selected value
></dees-input-multitoggle>
```

`DeesInputRadiogroup`

Radio button group for single-choice selections with internal state management.

```
<dees-input-radiogroup
  key="plan"
  label="Select Plan"
  .options=${['Free', 'Pro', 'Enterprise']}
  selectedOption="Pro"
  required
  @change=${handlePlanChange}
></dees-input-radiogroup>

// With custom option objects
<dees-input-radiogroup
  key="priority"
  label="Priority Level"
  .options=${[
    { key: 'low', label: 'Low Priority' },
    { key: 'medium', label: 'Medium Priority' },
    { key: 'high', label: 'High Priority' }
  ]}
  selectedOption="medium"
></dees-input-radiogroup>
```

`DeesInputTags`

Tag input component for managing lists of tags with auto-complete and validation.

```
<dees-input-tags
  key="skills"
  label="Skills"
  .value=${['JavaScript', 'TypeScript', 'CSS']}
  placeholder="Add a skill..."
  .suggestions=${[
    'JavaScript', 'TypeScript', 'Python', 'Go', 'Rust',
    'React', 'Vue', 'Angular', 'Node.js', 'Docker'
  ]}
  maxTags={10}  // Optional: limit number of tags
  required
  @change=${handleTagsChange}
></dees-input-tags>
```

Key Features:

- Add tags by pressing Enter or typing comma/semicolon
- Remove tags with click or backspace
- Auto-complete suggestions with keyboard navigation
- Maximum tag limit support
- Full theme support
- Form validation integration

`DeesInputTypelist`

Dynamic list input for managing arrays of typed values.

```
<dees-input-typelist
  key="features"
  label="Product Features"
  placeholder="Add a feature..."
  .value=${['Feature 1', 'Feature 2']}
  @change=${handleFeaturesChange}
></dees-input-typelist>
```

`DeesInputList`

Advanced list input with drag-and-drop reordering, inline editing, and validation.

```
<dees-input-list
  key="items"
  label="List Items"
```

```
  placeholder="Add new item..."
  .value=${['Item 1', 'Item 2', 'Item 3']}
  maxItems={10}            // Optional: maximum items
  minItems={1}             // Optional: minimum items
  allowDuplicates={false}  // Optional: allow duplicate values
  sortable={true}          // Optional: enable drag-and-drop reordering
  confirmDelete={true}     // Optional: confirm before deletion
  @change=${handleListChange}
></dees-input-list>
```

Key Features:

- Add, edit, and remove items inline
- Drag-and-drop reordering with visual feedback
- Optional duplicate prevention
- Min/max item constraints
- Delete confirmation dialog
- Full keyboard support
- Form validation integration

`DeesInputProfilepicture`

Profile picture input with cropping, zoom, and image processing.

```
<dees-input-profilepicture
  key="avatar"
  label="Profile Picture"
  shape="round"           // Options: round, square
  size={120}              // Display size in pixels
  .value=${imageBase64}   // Base64 encoded image or URL
  allowUpload={true}      // Enable upload button
  allowDelete={true}      // Enable delete button
  maxFileSize={5242880}   // Max file size in bytes (5MB)
  .acceptedFormats=${['image/jpeg', 'image/png', 'image/webp']}
  outputSize={800}        // Output resolution in pixels
  outputQuality={0.95}    // JPEG quality (0-1)
  @change=${handleAvatarChange}
></dees-input-profilepicture>
```

Key Features:

- Interactive cropping modal with zoom and pan
- Drag-and-drop file upload

- Round or square output shapes
- Configurable output size and quality
- File size and format validation
- Delete functionality
- Preview on hover

`DeesInputDatepicker`

Date and time picker component with calendar interface and manual typing support.

```
<dees-input-datepicker
  key="eventDate"
  label="Event Date"
  placeholder="YYYY-MM-DD"
  value="2025-01-15T14:30:00Z"  // ISO string format
  dateFormat="YYYY-MM-DD"        // Display format (default: YYYY-MM-DD)
  enableTime={true}              // Enable time selection
  timeFormat="24h"               // Options: 24h, 12h
  minuteIncrement={15}           // Time step in minutes
  minDate="2025-01-01"          // Minimum selectable date
  maxDate="2025-12-31"          // Maximum selectable date
  .disabledDates=${[            // Array of disabled dates
    '2025-01-10',
    '2025-01-11'
  ]}
  weekStartsOn={1}              // 0 = Sunday, 1 = Monday
  required
  @change=${handleDateChange}
></dees-input-datepicker>
```

Key Features:

- Interactive calendar popup
- Manual date typing with multiple formats
- Optional time selection
- Configurable date format
- Min/max date constraints
- Disable specific dates
- Keyboard navigation
- Today button
- Clear functionality
- 12/24 hour time formats
- Theme-aware styling

- Live parsing and validation

Manual Input Formats:

```
// Date formats supported
"2023-12-20"     // ISO format (YYYY-MM-DD)
"20.12.2023"     // European format (DD.MM.YYYY)
"12/20/2023"     // US format (MM/DD/YYYY)

// Date with time (add space and time after any date format)
"2023-12-20 14:30"
"20.12.2023 9:45"
"12/20/2023 16:00"
```

`DeesInputSearchselect`

Search-enabled dropdown selection component.

```
<dees-input-searchselect
  key="category"
  label="Select Category"
  placeholder="Search categories..."
  .options=${[
    { key: 'tech', label: 'Technology' },
    { key: 'health', label: 'Healthcare' },
    { key: 'finance', label: 'Finance' }
  ]}
  required
  @change=${handleCategoryChange}
></dees-input-searchselect>
```

`DeesInputRichtext`

Rich text editor with formatting toolbar powered by TipTap.

```
<dees-input-richtext
  key="content"
  label="Article Content"
  .value=${htmlContent}
  placeholder="Start writing..."
  minHeight={300}      // Minimum editor height
```

```
  showWordCount={true} // Show word/character count
  @change=${handleContentChange}
></dees-input-richtext>
```

Key Features:

- Full formatting toolbar (bold, italic, underline, strike, etc.)
- Heading levels (H1-H6)
- Lists (bullet, ordered)
- Links with URL editing
- Code blocks and inline code
- Blockquotes
- Horizontal rules
- Undo/redo support
- Word and character count
- HTML output

`DeesInputWysiwyg`

Advanced block-based editor with slash commands and rich content blocks.

```
<dees-input-wysiwyg
  key="document"
  label="Document Editor"
  .value=${documentContent}
  outputFormat="html"  // Options: html, markdown, json
  @change=${handleDocumentChange}
></dees-input-wysiwyg>
```

Key Features:

- Slash commands for quick formatting
- Block-based editing (paragraphs, headings, lists, etc.)
- Drag and drop block reordering
- Multiple output formats
- Keyboard shortcuts
- Collaborative editing ready
- Extensible block types

`DeesInputCode`

Code input component for editing source code with syntax highlighting.

```
<dees-input-code
  key="snippet"
  label="Code Snippet"
  .value=${codeString}
  language="typescript"
  @change=${handleCodeChange}
></dees-input-code>
```

`DeesFormSubmit`

Submit button component specifically designed for `DeesForm`.

```
<dees-form-submit
  disabled            // Optional: disable submit button
  status="normal"     // Options: normal, pending, success, error
>Submit Form</dees-form-submit>
```

App Shell (Layout) Components

`DeesAppui` □□

A comprehensive application shell component providing a complete UI framework with navigation, menus, activity logging, bottom bar, and view management.

> **Full API Documentation**: See ts_web/elements/00group-appui/dees-appui/readme.md for complete documentation including all programmatic APIs, view lifecycle hooks, and TypeScript interfaces.

Quick Start:

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

@customElement('my-app')
class MyApp extends DeesElement {
  private appui: DeesAppui;
```

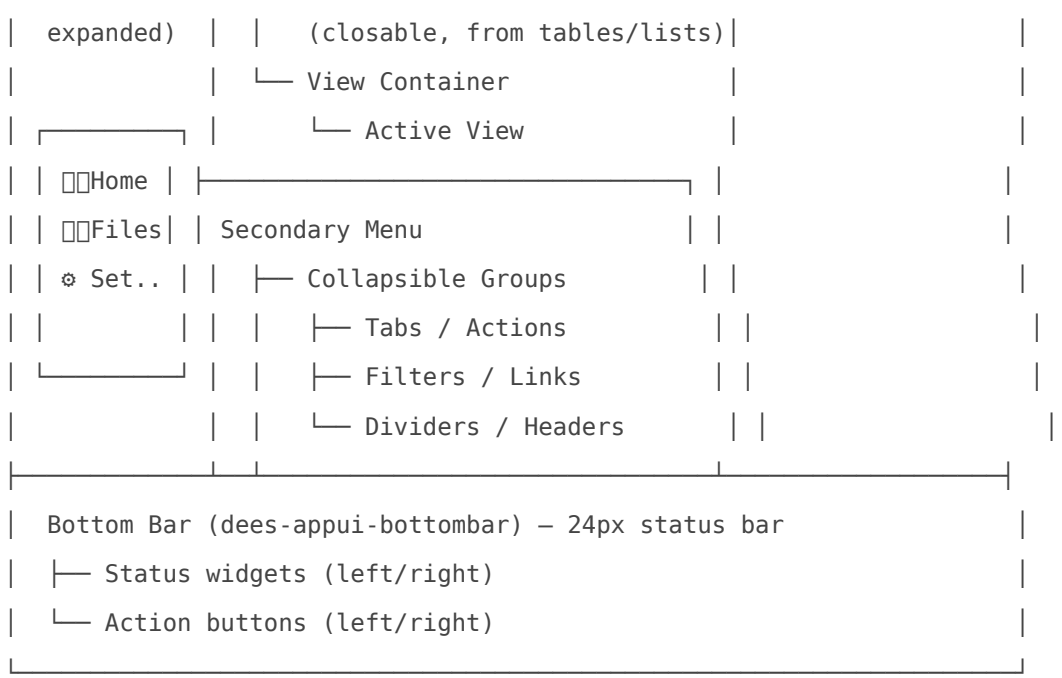
```
  async firstUpdated() {
    this.appui = this.shadowRoot.querySelector('dees-appui');

    this.appui.configure({
      branding: { logoIcon: 'lucide:box', logoText: 'My App' },
      views: [
        { id: 'dashboard', name: 'Dashboard', iconName: 'lucide:home', content: 'my-dashboard'
},
        { id: 'settings', name: 'Settings', iconName: 'lucide:settings', content: 'my-
settings' },
      ],
      mainMenu: {
        sections: [{ name: 'Main', views: ['dashboard', 'settings'] }]
      },
      defaultView: 'dashboard',
      bottomBar: {
        visible: true,
        widgets: [
          { id: 'status', iconName: 'lucide:activity', label: 'Online', status: 'success' }
        ]
      }
    });
  }

  render() {
    return html`<dees-appui></dees-appui>`;
  }
}
```

Architecture Overview:

```
┌──────────────────────────────────────────────────────────────────────┐
│  AppBar (dees-appui-appbar)                                          │
│  ├── Menus (File, Edit, View...)                                     │
│  ├── Breadcrumbs                                                     │
│  ├── User Profile + Dropdown                                         │
│  └── Activity Log Toggle                                             │
├──────────────┬──────────────────────────────────┬────────────────────┤
│ Main Menu    │  Content Area                    │  Activity Log      │
│ (collapsed/  │  ├── Content Tabs                │  (slide panel)     │
```



Configuration (`IAppConfig`):

```
interface IAppConfig {
  branding?: { logoIcon?: string; logoText?: string };
  appBar?: IAppBarConfig;
  views: IViewDefinition[];
  mainMenu?: IMainMenuConfig;
  defaultView?: string;
  activityLog?: IActivityLogConfig;
  bottomBar?: IBottomBarConfig;
  onViewChange?: (viewId: string, view: IViewDefinition) => void;
  onSearch?: (query: string) => void;
}
```

Key Features:

- □□ **Configure API** — Single `configure()` method for complete app setup
- □□ **View Management** — Automatic view caching, lazy loading, and lifecycle hooks (`onActivate`, `onDeactivate`, `canDeactivate`)
- □□ **Hash-based Routing** — Automatic URL synchronization with view navigation and parameterized routes
- □□ **Activity Log** — Slide-out panel with stacked entries, date grouping, search, and filtering
- □□ **Bottom Status Bar** — Configurable widgets and actions with status colors and loading states
- □□ **RxJS Observables** — `viewChanged$` and `viewLifecycle$` for reactive programming
- □□ **TypeScript-first** — Typed `IViewActivationContext` passed to views on activation

Programmatic APIs:

Area	Methods
Navigation	`navigateToView(viewId, params?)`, `getCurrentView()`, `getViewRegistry()`
App Bar	`setAppBarMenus()`, `updateAppBarMenu()`, `setBreadcrumbs()`, `setUser()`, `setProfileMenuItems()`, `setSearchVisible()`, `onSearch()`, `setWindowControlsVisible()`
Main Menu	`setMainMenu()`, `updateMainMenuGroup()`, `addMainMenuItem()`, `removeMainMenuItem()`, `setMainMenuSelection()`, `setMainMenuCollapsed()`, `setMainMenuVisible()`, `setMainMenuBadge()`, `clearMainMenuBadge()`
Secondary Menu	`setSecondaryMenu()`, `updateSecondaryMenuGroup()`, `addSecondaryMenuItem()`, `setSecondaryMenuSelection()`, `setSecondaryMenuCollapsed()`, `setSecondaryMenuVisible()`, `clearSecondaryMenu()`
Content Tabs	`setContentTabs()`, `addContentTab()`, `removeContentTab()`, `selectContentTab()`, `getSelectedContentTab()`, `setContentTabsVisible()`, `setContentTabsAutoHide()`
Activity Log	`activityLog.add()`, `activityLog.addMany()`, `activityLog.clear()`, `activityLog.getEntries()`, `activityLog.filter()`, `activityLog.search()`, `setActivityLogVisible()`, `toggleActivityLog()`, `getActivityLogVisible()`
Bottom Bar	`bottomBar.addWidget()`, `bottomBar.updateWidget()`, `bottomBar.removeWidget()`, `bottomBar.getWidget()`, `bottomBar.clearWidgets()`, `bottomBar.addAction()`, `bottomBar.removeAction()`, `bottomBar.clearActions()`, `setBottomBarVisible()`, `getBottomBarVisible()`
Observables	`viewChanged$`, `viewLifecycle$`

View Lifecycle Hooks:

```
import { DeesElement, customElement } from '@design.estate/dees-element';
import type { IViewActivationContext, IViewLifecycle } from '@design.estate/dees-catalog';

@customElement('my-settings-view')
class MySettingsView extends DeesElement implements IViewLifecycle {
  // Called when view becomes visible
  async onActivate(context: IViewActivationContext) {
    const { appui, viewId, params } = context;

    // Set view-specific secondary menu
    appui.setSecondaryMenu({
      heading: 'Settings',
```

```
      groups: [{ name: 'Options', items: [...] }]
    });

    // Control visibility of other shell parts
    appui.setContentTabsVisible(false);
    appui.setSecondaryMenuVisible(true);
  }

  // Called when navigating away
  onDeactivate() { /* cleanup */ }

  // Return false or a message string to block navigation
  canDeactivate(): boolean | string {
    if (this.hasUnsavedChanges) return 'You have unsaved changes. Leave anyway?';
    return true;
  }
}
```

Secondary Menu Item Types:

The secondary menu supports **8 distinct item types** for building rich contextual sidebars:

Type	Description
Tab (default)	Selectable item that stays highlighted
Action	Executes on click without staying selected (blue styling)
Filter	Checkbox toggle for filtering
MultiFilter	Collapsible multi-select filter box
Divider	Visual separator line
Header	Non-interactive section label
Link	Opens an external URL
Danger Action	Red-styled action with optional confirmation

`DeesAppuiMainmenu`

Main navigation menu component for application-wide navigation. Supports collapsed (icon-only) mode.

```
<dees-appui-mainmenu
  .menuGroups=${[
```

```
    {
      name: 'Main',
      items: [
        { key: 'dashboard', iconName: 'lucide:home', action: () => navigate('dashboard') },
        { key: 'settings', iconName: 'lucide:settings', action: () => navigate('settings') }
      ]
    }
  ]}
  collapsed           // Optional: show collapsed icon-only version
></dees-appui-mainmenu>
```

DeesAppuiSecondarymenu

Secondary navigation component for sub-section selection with collapsible groups, badges, and 8 item types.

```
<dees-appui-secondarymenu
  .heading=${'Projects'}
  .groups=${[
    {
      name: 'Active',
      iconName: 'lucide:folder',
      items: [
        { key: 'Frontend App', iconName: 'lucide:code', action: () => select('frontend'),
badge: 3, badgeVariant: 'warning' },
        { key: 'API Server', iconName: 'lucide:server', action: () => select('api') }
      ]
    }
  ]}
  @item-select=${handleSectionChange}
></dees-appui-secondarymenu>
```

DeesAppuiMaincontent

Main content area with tab management support.

```
<dees-appui-maincontent
  .tabs=${[
    { key: 'Overview', iconName: 'lucide:home', action: () => selectTab('overview') },
    { key: 'Details', iconName: 'lucide:info', action: () => selectTab('details') }
  ]}
```

```
  @tab-select=${handleTabChange}
>
  <!-- Content goes here -->
</dees-appui-maincontent>
```

`DeesAppuiAppbar`

Professional application bar component with hierarchical menus, breadcrumb navigation, user account management, and activity log toggle.

```
<dees-appui-appbar
  .menuItems=${[
    {
      name: 'File',
      action: async () => {},
      submenu: [
        { name: 'New File', shortcut: 'Cmd+N', iconName: 'file-plus', action: async () =>
handleNewFile() },
        { name: 'Open...', shortcut: 'Cmd+O', iconName: 'folder-open', action: async () =>
handleOpen() },
        { divider: true },
        { name: 'Save', shortcut: 'Cmd+S', iconName: 'save', action: async () => handleSave(),
disabled: true }
      ]
    }
  ]}
  .breadcrumbs=${'Project > src > components'}
  .showWindowControls=${true}
  .showSearch=${true}
  .showActivityLogToggle=${true}
  .activityLogCount=${5}
  .activityLogActive=${false}
  .user=${{
    name: 'John Doe',
    avatar: '/path/to/avatar.jpg',
    status: 'online'  // Options: 'online' | 'offline' | 'busy' | 'away'
  }}
  @menu-select=${(e) => handleMenuSelect(e.detail.item)}
  @breadcrumb-navigate=${(e) => handleBreadcrumbClick(e.detail)}
  @activity-toggle=${() => handleActivityToggle()}
```

```
></dees-appui-appbar>
```

Key Features:

- **Hierarchical Menu System** — Top-level menus with dropdown submenus, icons, and keyboard shortcuts
- **Keyboard Navigation** — Full keyboard support (Tab, Arrow keys, Enter, Escape)
- **Breadcrumb Navigation** — Customizable breadcrumb trail with click events
- **User Account Section** — Avatar with status indicator and profile dropdown
- **Activity Log Toggle** — Button with badge count to show/hide activity panel
- **Accessibility** — Full ARIA support with menubar roles

`DeesAppuiActivitylog`

Real-time activity log panel for displaying user actions and system events.

```
<dees-appui-activitylog></dees-appui-activitylog>

// Programmatic API
activityLog.add({
  type: 'update',        // Options: login, logout, view, create, update, delete, custom
  user: 'John Doe',
  message: 'Updated project settings',
  iconName: 'lucide:settings'  // Optional: custom icon
});

activityLog.addMany(entries);  // Add multiple entries
activityLog.clear();           // Clear all entries
activityLog.getEntries();      // Get all entries
activityLog.filter({ user: 'John' });  // Filter by user/type
activityLog.search('settings');        // Search by message
```

Key Features:

- Stacked entry layout with icon, user, timestamp, and message
- Date grouping (Today, Yesterday, etc.)
- Search and filter functionality
- Context menu for entry actions
- Live streaming indicator
- Animated slide-in/out panel
- Theme-aware styling

`DeesAppuiBottombar`

A 24px fixed-height status bar at the bottom of the application shell. Supports status widgets and action buttons positioned left or right.

```
// Configure via DeesAppui
appui.configure({
  bottomBar: {
    visible: true,
    widgets: [
      {
        id: 'status',
        iconName: 'lucide:activity',
        label: 'System Online',
        status: 'success',       // 'idle' | 'active' | 'success' | 'warning' | 'error'
        tooltip: 'All systems operational',
        onClick: () => console.log('Status clicked'),
      },
      {
        id: 'version',
        iconName: 'lucide:gitBranch',
        label: 'v1.2.3',
        position: 'right',
      }
    ],
    actions: [
      {
        id: 'terminal',
        iconName: 'lucide:terminal',
        tooltip: 'Open Terminal',
        position: 'right',
        onClick: () => console.log('Terminal clicked'),
      }
    ]
  }
});

// Programmatic updates
appui.bottomBar.addWidget({ id: 'build', iconName: 'lucide:hammer', label: 'Building...',
loading: true, status: 'active' });
appui.bottomBar.updateWidget('build', { label: 'Build complete', loading: false, status:
'success' });
```

```
appui.bottomBar.removeWidget('build');

appui.bottomBar.addAction({ id: 'refresh', iconName: 'lucide:refreshCw', onClick: () =>
location.reload() });
appui.bottomBar.removeAction('refresh');

appui.setBottomBarVisible(false);
```

Key Features:

- Configurable status widgets with icons, labels, and colored status indicators
- Loading spinner state for widgets
- Contextual actions with icon buttons
- Left/right positioning for both widgets and actions
- Tooltips on hover
- Context menu support per widget

`DeesAppuiTabs`

Reusable tab component with horizontal/vertical layout support.

```
<dees-appui-tabs
  .tabs=${[
    { key: 'Home', iconName: 'lucide:home', action: () => console.log('Home') },
    { key: 'Settings', iconName: 'lucide:settings', action: () => console.log('Settings') }
  ]}
  tabStyle="horizontal"  // Options: horizontal, vertical
  showTabIndicator={true}
  @tab-select=${handleTabSelect}
></dees-appui-tabs>
```

Data Display Components

`DeesTable`

Advanced table component with sorting, filtering, and action support.

```
<dees-table
  .data=${tableData}
  .displayFunction=${(item) => ({
    name: item.name,
```

```
    date: item.date,
    amount: item.amount,
    description: item.description
  })}
  .dataActions=${[
    {
      name: 'Edit',
      icon: 'edit',
      action: (item) => handleEdit(item)
    },
    {
      name: 'Delete',
      icon: 'trash',
      action: (item) => handleDelete(item)
    }
  ]}
  heading1="Transactions"
  heading2="Recent Activity"
  searchable           // Enable search functionality
  dataName="transaction"  // Name for single data item
  @selection-change=${handleSelectionChange}
></dees-table>
```

Advanced Features:

- Schema-first columns or `displayFunction` rendering
- Sorting via header clicks with `aria-sort` + `sortChange`
- Global search with Lucene-like syntax; modes: `table`, `data`, `server`
- Per-column quick filters row; `showColumnFilters` and `column.filterable=false`
- Selection: `none` | `single` | `multi`, with select-all and `selectionChange`
- Sticky header + internal scroll (`stickyHeader`, `--table-max-height`)
- Rich actions: header/in-row/contextmenu/footer/doubleClick; pinned Actions column
- Editable cells via `editableFields`
- Drag & drop files onto rows

`DeesDataviewCodebox`

Code display component with syntax highlighting and line numbers.

```
<dees-dataview-codebox
  progLang="typescript"  // Programming language for syntax highlighting
  .codeToDisplay=${`
```

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

    export const myComponent = () => {
      return html\`<div>Hello World</div>\`;
    };
  `}
></dees-dataview-codebox>
```

DeesDataviewStatusobject

Status display component for complex objects with nested status indicators.

```
<dees-dataview-statusobject
  .statusObject=${{
    id: '1',
    name: 'System Status',
    combinedStatus: 'partly_ok',
    combinedStatusText: 'Partially OK',
    details: [
      { name: 'Database', value: 'Connected', status: 'ok', statusText: 'OK' },
      { name: 'API Service', value: 'Degraded', status: 'partly_ok', statusText: 'Partially
OK' }
    ]
  }}
></dees-dataview-statusobject>
```

DeesPdf

PDF viewer component with navigation and zoom controls.

```
<dees-pdf
  source="path/to/document.pdf"  // URL or base64 encoded PDF
  page={1}            // Current page number
  scale={1.0}         // Zoom level
  .controls=${['zoom', 'download', 'print', 'navigation']}
  @page-change=${handlePageChange}
></dees-pdf>
```

DeesStatsGrid

A responsive grid component for displaying statistical data with various visualization types.

```
<dees-statsgrid
  .tiles=${[
    {
      id: 'revenue',
      title: 'Total Revenue',
      value: 125420,
      unit: '$',
      type: 'number',
      icon: 'lucide:dollarSign',
      description: '+12.5% from last month',
      color: '#22c55e'
    },
    {
      id: 'cpu',
      title: 'CPU Usage',
      value: 73,
      type: 'gauge',
      icon: 'lucide:cpu',
      gaugeOptions: {
        min: 0, max: 100,
        thresholds: [
          { value: 0, color: '#22c55e' },
          { value: 60, color: '#f59e0b' },
          { value: 80, color: '#ef4444' }
        ]
      }
    },
    {
      id: 'requests',
      title: 'API Requests',
      value: '1.2k',
      unit: '/min',
      type: 'trend',
      icon: 'lucide:server',
      trendData: [45, 52, 38, 65, 72, 68, 75, 82, 79, 85, 88, 92]
    },
    {
      id: 'cores',
      title: 'CPU Cores',
      value: 0,
```

```
      type: 'cpuCores',
      icon: 'lucide:cpu',
      columnSpan: 2,
      coresData: [
        { id: 0, usage: 45, label: '0' },
        { id: 1, usage: 72, label: '1' },
        { id: 2, usage: 30, label: '2' },
        { id: 3, usage: 88, label: '3' }
      ]
    }
  ]}
  .minTileWidth=${250}
  .gap=${16}
></dees-statsgrid>
```

Tile Types: `number`, `gauge`, `percentage`, `trend`, `text`, `multiPercentage`, `cpuCores`

`DeesPagination`

Pagination component for navigating through large datasets.

```
<dees-pagination
  totalItems={500}
  itemsPerPage={20}
  currentPage={1}
  maxVisiblePages={7}
  @page-change=${handlePageChange}
></dees-pagination>
```

Media & Tile Components □□

A rich collection of tile-based preview components for displaying media files in grids. All tiles share a consistent base class (`DeesTileBase`) with lazy loading via `IntersectionObserver`, hover interactions, click events, context menus, and three size variants (`small`, `default`, `large`).

All tile badges use a unified styling system with label-awareness — when a `label` is set, bottom badges automatically shift up to avoid overlapping.

`DeesTilePdf`

PDF document tile with live page preview on hover.

```
<dees-tile-pdf
  pdfUrl="/documents/report.pdf"
  label="Annual Report"
  clickable
  @tile-click=${handleClick}
></dees-tile-pdf>
```

Key Features:

- Renders first page as canvas preview
- Hover to scrub through pages (mouse X position maps to page number)
- Shows page count badge, hover page indicator
- Detects A4/Letter vs non-standard aspect ratios

`DeesTileImage`

Image tile with lazy loading and dimension display.

```
<dees-tile-image
  src="/photos/landscape.jpg"
  alt="Mountain landscape"
  label="landscape.jpg"
  clickable
  @tile-click=${handleClick}
></dees-tile-image>
```

Key Features:

- Lazy loads image on scroll into view
- Shows image dimensions on hover (e.g. "1920 × 1080")
- Checkerboard background for transparent images

`DeesTileAudio`

Audio file tile with waveform visualization.

```
<dees-tile-audio
  src="/music/track.mp3"
  title="Summer Vibes"
  artist="DJ Example"
  clickable
  @tile-click=${handleClick}
></dees-tile-audio>
```

Key Features:

- Generates waveform visualization from audio data
- Shows duration badge (e.g. "3:42")
- Displays title and artist metadata
- Play overlay on hover

`DeesTileVideo`

Video tile with thumbnail capture and hover preview.

```
<dees-tile-video
  src="/videos/intro.mp4"
  poster="/thumbs/intro.jpg"
  label="Introduction"
  clickable
  @tile-click=${handleClick}
></dees-tile-video>
```

Key Features:

- Auto-captures first frame as thumbnail (or uses provided `poster`)
- Plays video preview on hover
- Shows duration badge
- Play button overlay

`DeesTileNote`

Code/text snippet tile with syntax-aware display.

```
<dees-tile-note
  title="config.ts"
  language="TypeScript"
  .content=${codeString}
  clickable
  @tile-click=${handleClick}
></dees-tile-note>
```

Key Features:

- Monospace font with line numbers
- Language badge (top-right)
- Scrollable content on hover (mouse X position controls scroll)
- Line indicator badge while scrolling

- Gradient fade at bottom

`DeesTileFolder`

Folder tile with 2×2 content preview grid.

```
<dees-tile-folder
  name="Project Assets"
  .items=${[
    { type: 'image', name: 'logo.png', thumbnailSrc: '/thumbs/logo.png' },
    { type: 'pdf', name: 'spec.pdf' },
    { type: 'audio', name: 'jingle.mp3' },
    { type: 'video', name: 'demo.mp4' },
  ]}
  clickable
  @tile-click=${handleClick}
></dees-tile-folder>
```

Key Features:

- 2×2 preview grid showing first 4 items (thumbnails or type icons)
- Item count badge (e.g. "12 items")
- Folder icon header with name
- Supports: `pdf`, `image`, `audio`, `video`, `note`, `folder`, `unknown` types

`DeesPreview`

Unified preview component that auto-selects the right tile type based on content.

`DeesPdfViewer` / `DeesPdfPreview`

Full PDF viewing components with navigation controls.

`DeesImageViewer`

Full-screen image viewer with zoom and pan.

`DeesAudioViewer`

Audio playback component with waveform and controls.

`DeesVideoViewer`

Video playback component with standard controls.

Visualization Components

`DeesChartArea`

Area chart component built on ApexCharts for visualizing time-series data.

```
<dees-chart-area
  label="System Usage"
  .series=${[
    {
      name: 'CPU',
      data: [
        { x: '2025-01-15T03:00:00', y: 25 },
        { x: '2025-01-15T07:00:00', y: 30 },
        { x: '2025-01-15T11:00:00', y: 20 }
      ]
    }
  ]}
></dees-chart-area>
```

`DeesChartLog`

Specialized chart component for visualizing log data and events.

```
<dees-chart-log
  label="System Events"
  .data=${[
    { timestamp: '2025-01-15T03:00:00', event: 'Server Start', type: 'info' },
    { timestamp: '2025-01-15T03:15:00', event: 'Error Detected', type: 'error' }
  ]}
  .filters=${['info', 'warning', 'error']}
  @event-click=${handleEventClick}
></dees-chart-log>
```

Dialogs & Overlays Components

`DeesModal`

Modal dialog component with customizable content and actions.

```
// Programmatic usage
DeesModal.createAndShow({
  heading: 'Confirm Action',
  content: html`
    <dees-form>
      <dees-input-text .label=${'Enter reason'}></dees-input-text>
    </dees-form>
  `,
  menuOptions: [
    { name: 'Cancel', action: async (modal) => { modal.destroy(); return null; } },
    { name: 'Confirm', action: async (modal) => { /* handle */ modal.destroy(); return null; }
}
  ]
});
```

DeesContextmenu

Context menu component for right-click actions with nested submenu support.

```
// Programmatic usage
DeesContextmenu.openContextMenuWithOptions(mouseEvent, [
  {
    name: 'Edit',
    iconName: 'lucide:edit',
    action: async () => handleEdit()
  },
  { divider: true },
  {
    name: 'More Options',
    iconName: 'lucide:moreHorizontal',
    submenu: [
      { name: 'Duplicate', iconName: 'lucide:copy', action: async () => handleDuplicate() },
      { name: 'Archive', iconName: 'lucide:archive', action: async () => handleArchive() },
    ]
  },
  {
    name: 'Delete',
    iconName: 'lucide:trash2',
    action: async () => handleDelete()
  }
```

```
]);

// Component-based (implement getContextMenuItems on any element)
class MyComponent extends DeesElement {
  getContextMenuItems() {
    return [
      { name: 'View Details', iconName: 'lucide:eye', action: async () => { ... } },
      { name: 'Edit', iconName: 'lucide:edit', action: async () => { ... } },
    ];
  }
}
```

`DeesSpeechbubble`

Tooltip-style speech bubble component for contextual information.

```
// Programmatic usage
const bubble = await DeesSpeechbubble.createAndShow(
  referenceElement,
  'Helpful information about this feature'
);
```

`DeesWindowlayer`

Base overlay component used by modal dialogs and other overlay components.

```
const layer = await DeesWindowLayer.createAndShow({
  blur: true,
});
```

Navigation Components

`DeesStepper`

Multi-step navigation component for guided user flows.

```
<dees-stepper
  .steps=${[
    { key: 'personal', label: 'Personal Info', content: html`<div>Form 1</div>` },
    { key: 'address', label: 'Address', content: html`<div>Form 2</div>` },
```

```
    { key: 'confirm', label: 'Confirmation', content: html`<div>Review</div>` }
  ]}
  currentStep="personal"
  @step-change=${handleStepChange}
  @complete=${handleComplete}
></dees-stepper>
```

`DeesProgressbar`

Progress indicator component for tracking completion status.

```
<dees-progressbar
  value={75}
  label="Uploading"
  showPercentage
  type="determinate"  // Options: determinate, indeterminate
  status="normal"     // Options: normal, success, warning, error
></dees-progressbar>
```

Theming Components

`DeesTheme`

Theme provider component that wraps children and provides CSS custom properties for consistent theming.

```
// Basic usage - wrap your app
<dees-theme>
  <my-app></my-app>
</dees-theme>

// With custom overrides
<dees-theme
  .customColors=${{
    primary: '#007bff',
    success: '#28a745'
  }}
  .customSpacing=${{
    lg: '24px',
```

```
    xl: '32px'
  }}
>
  <my-section></my-section>
</dees-theme>
```

Key Features:

- Provides CSS custom properties for colors, spacing, radius, shadows, and transitions
- Can be nested for section-specific theming
- Works with dark/light mode
- Overrides cascade to all child components

Workspace / IDE Components □□

A full-featured IDE workspace component suite for building browser-based code editors, terminal interfaces, and documentation viewers.

`DeesWorkspace`

Top-level workspace shell that composes editor, file tree, terminal, and bottom bar into an IDE-like layout.

```
<dees-workspace></dees-workspace>
```

`DeesWorkspaceMonaco`

Monaco Editor integration for code editing with full IntelliSense, syntax highlighting, and language support.

```
<dees-workspace-monaco
  .value=${code}
  .language=${'typescript'}
  @change=${handleCodeChange}
></dees-workspace-monaco>
```

`DeesWorkspaceDiffEditor`

Side-by-side diff editor powered by Monaco for comparing file versions.

```
<dees-workspace-diff-editor
  .originalValue=${originalCode}
```

```
  .modifiedValue=${modifiedCode}
  .language=${'typescript'}
></dees-workspace-diff-editor>
```

`DeesWorkspaceFiletree`

File tree navigation component with expand/collapse, file icons, and selection.

```
<dees-workspace-filetree
  .files=${fileTreeData}
  @file-select=${handleFileSelect}
></dees-workspace-filetree>
```

`DeesWorkspaceTerminal`

Terminal emulator component powered by xterm.js.

```
<dees-workspace-terminal></dees-workspace-terminal>
```

`DeesWorkspaceTerminalPreview`

Terminal with integrated preview pane for output visualization.

`DeesWorkspaceMarkdown`

Markdown editor with live preview.

`DeesWorkspaceMarkdownoutlet`

Read-only markdown renderer for documentation display.

`DeesWorkspaceBottombar`

IDE-style bottom status bar for the workspace.

Pre-built Templates

`DeesSimpleAppdash`

Simple application dashboard component for quick prototyping.

```
<dees-simple-appdash
  .appTitle=${'My Application'}
  .menuItems=${[
    { name: 'Dashboard', icon: 'home', route: '/dashboard' },
    { name: 'Settings', icon: 'settings', route: '/settings' }
  ]}
  .user=${{ name: 'John Doe', role: 'Administrator' }}
  @menu-select=${handleMenuSelect}
>
  <!-- Dashboard content -->
</dees-simple-appdash>
```

`DeesSimpleLogin`

Simple login form component with validation and customization.

```
<dees-simple-login
  .appName=${'My Application'}
  .logo=${'./assets/logo.png'}
  .backgroundImage=${'./assets/background.jpg'}
  .fields=${['username', 'password']}
  showForgotPassword
  showRememberMe
  @login=${handleLogin}
  @forgot-password=${handleForgotPassword}
></dees-simple-login>
```

Shopping Components

`DeesShoppingProductcard`

Product card component for e-commerce applications.

```
<dees-shopping-productcard
  .productData=${{
    name: 'Premium Headphones',
    category: 'Electronics',
    description: 'High-quality wireless headphones with noise cancellation',
    price: 199.99,
```

```
    originalPrice: 249.99,
    currency: '$',
    inStock: true,
    imageUrl: '/images/headphones.jpg'
  }}
  quantity={1}
  showQuantitySelector={true}
  @quantityChange=${handleQuantityChange}
></dees-shopping-productcard>
```

□□ TypeScript Interfaces

The library exports unified interfaces for consistent API patterns:

```
// Base menu item interface (used by tabs, menus, etc.)
interface IMenuItem {
  key: string;
  iconName?: string;
  action: () => void;
  badge?: string | number;
  badgeVariant?: 'default' | 'success' | 'warning' | 'error';
  closeable?: boolean;
  onClose?: () => void;
}

// Menu group interface for organized menus
interface IMenuGroup {
  name: string;
  items: IMenuItem[];
  collapsed?: boolean;
  iconName?: string;
}

// View definition for app navigation
interface IViewDefinition {
  id: string;
  name: string;
```

```
  iconName?: string;
  content: string | (new () => HTMLElement) | (() => TemplateResult) | (() => Promise<any>);
  secondaryMenu?: ISecondaryMenuGroup[];
  contentTabs?: IMenuItem[];
  route?: string;
  badge?: string | number;
  badgeVariant?: 'default' | 'success' | 'warning' | 'error';
  cache?: boolean;
}

// Activity log entry
interface IActivityEntry {
  id?: string;
  timestamp?: Date;
  type: 'login' | 'logout' | 'view' | 'create' | 'update' | 'delete' | 'custom';
  user: string;
  message: string;
  iconName?: string;
  data?: Record<string, unknown>;
}

// Bottom bar widget
interface IBottomBarWidget {
  id: string;
  iconName?: string;
  label?: string;
  status?: 'idle' | 'active' | 'success' | 'warning' | 'error';
  tooltip?: string;
  loading?: boolean;
  onClick?: () => void;
  position?: 'left' | 'right';
  order?: number;
}

// Bottom bar action button
interface IBottomBarAction {
  id: string;
  iconName: string;
  tooltip?: string;
  onClick: () => void | Promise<void>;
```

```
  disabled?: boolean;
  position?: 'left' | 'right';
}

// View activation context (passed to onActivate)
interface IViewActivationContext {
  appui: DeesAppui;
  viewId: string;
  params?: Record<string, string>;
}

// Tile folder item (for DeesTileFolder)
interface ITileFolderItem {
  type: 'pdf' | 'image' | 'audio' | 'video' | 'note' | 'folder' | 'unknown';
  thumbnailSrc?: string;
  name: string;
}
```

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 10:49:09 UTC by foss.global Team
Updated 2026-03-28 12:14:32 UTC by foss.global Team