

# Menu

<https://playdocs1.orangeriver-ad055946.westus2.azurecontainerapps.io/play-docs/docs/ui-components/Navigation/Menu>

## Menu

A flexible and feature-rich dropdown menu component for creating context menus, navigation dropdowns, and action menus. Supports icons, descriptions, multi-column layout, various positioning options, and full accessibility.

## How to use

```
import { AavaMenuComponent } from "@aava/play-core";
```

```
import { AavaMenuComponent } from "@aava/play-core";
```

## Basic Usage

The Menu component provides a dropdown interface that can be positioned relative to a trigger element. It supports various menu items with different states and configurations.

- Default behavior: Bottom positioning, single column layout, with icons and titles.
- Usage: Pass an array of menu items to the items input and control visibility with the visible input.

```
items
```

```
visible
```

```
<div class="menu-container"> <button (click)="toggleMenu()">Open Menu</button> <aava-menu [items]="menuItems" [visible]="showMenu" (itemSelected)="handleItemSelected($event)"> </aava-menu></div>
```

```
<div class="menu-container"> <button (click)="toggleMenu()">Open Menu</button> <aava-menu [items]="menuItems" [visible]="showMenu" (itemSelected)="handleItemSelected($event)"> </aava-menu></div>
```

```
import { Component } from "@angular/core";import { AavaMenuComponent } from "@aava/play-core";@Component({ selector: "app-menu-basic", standalone: true, imports: [AavaMenuComponent],})export class MenuBasicComponent { showMenu = false; menuItems = [ { label: "Dashboard", route: "/dashboard" }, { label: "Profile", route: "/profile" }, { label: "Settings", route: "/settings" }, { label: "Help", route: "/help" }, { label: "Logout", route: "/logout" }, ]; toggleMenu() { this.showMenu = !this.showMenu; } handleItemSelected(event: any) { console.log("Selected:", event.label); this.showMenu = false; }}
```

```
import { Component } from "@angular/core";import { AavaMenuComponent } from "@aava/play-core";@Component({ selector: "app-menu-basic", standalone: true, imports: [AavaMenuComponent],})export class MenuBasicComponent { showMenu = false; menuItems = [ { label: "Dashboard", route: "/dashboard" }, { label: "Profile", route: "/profile" }, { label: "Settings", route: "/settings" }, { label: "Help", route: "/help" }, { label: "Logout", route: "/logout" }, ]; toggleMenu() { this.showMenu = !this.showMenu; } handleItemSelected(event: any) { console.log("Selected:", event.label); this.showMenu = false; }}
```

## With Icons

Menu items can display icons alongside their labels for better visual recognition and user experience.

- How: Add an icon property to menu items with the icon name.
- When to use: For quick recognition, consistent visual language, or when space allows.

```
icon
```

```
<div class="menu-container"> <button (click)="toggleMenu()">Menu with Icons</button> <aava-menu [items]="menuItems" [visible]="showMenu" [displayOptions]="{ showIcon: true, showTitle: true }" (itemSelected)="handleItemSelected($event)"> </aava-menu></div>
```

```
<div class="menu-container"> <button (click)="toggleMenu()">Menu with Icons</button> <aava-menu [items]="menuItems" [visible]="showMenu" [displayOptions]="{{ showIcon: true, showTitle: true }}" (itemSelected)="handleItemSelected($event)"> </aava-menu></div>
```

```
import { Component } from "@angular/core";import { AavaMenuComponent } from "@aava/play-core";@Component({ selector: "app-menu-with-icons", standalone: true, imports: [AavaMenuComponent],})export class MenuWithIconsComponent { showMenu = false; menuItems = [ { label: "Dashboard", icon: "home", route: "/dashboard" }, { label: "Profile", icon: "user", route: "/profile" }, { label: "Settings", icon: "settings", route: "/settings" }, { label: "Notifications", icon: "bell", route: "/notifications" }, { label: "Help", icon: "help-circle", route: "/help" }, { label: "Logout", icon: "log-out", route: "/logout" }]; toggleMenu() { this.showMenu = !this.showMenu; } handleItemSelected(event: any) { console.log("Selected:", event.label); this.showMenu = false; }}
```

```
import { Component } from "@angular/core";import { AavaMenuComponent } from "@aava/play-core";@Component({ selector: "app-menu-with-icons", standalone: true, imports: [AavaMenuComponent],})export class MenuWithIconsComponent { showMenu = false; menuItems = [ { label: "Dashboard", icon: "home", route: "/dashboard" }, { label: "Profile", icon: "user", route: "/profile" }, { label: "Settings", icon: "settings", route: "/settings" }, { label: "Notifications", icon: "bell", route: "/notifications" }, { label: "Help", icon: "help-circle", route: "/help" }, { label: "Logout", icon: "log-out", route: "/logout" }]; toggleMenu() { this.showMenu = !this.showMenu; } handleItemSelected(event: any) { console.log("Selected:", event.label); this.showMenu = false; }}
```

## With Descriptions

Menu items can include descriptions to provide additional context or information about each option.

- How: Add a `description` property to menu items.
- When to use: For complex menus where items need additional explanation or context.

`description`

```
<div class="menu-container"> <button (click)="toggleMenu()">Menu with Descriptions</button> <aava-menu [items]="menuItems" [visible]="showMenu" [displayOptions]="{{ showIcon: true, showTitle: true, showDescription: true }}" (itemSelected)="handleItemSelected($event)"> </aava-menu></div>
```

```
<div class="menu-container"> <button (click)="toggleMenu()">Menu with Descriptions</button> <aava-menu [items]="menuItems" [visible]="showMenu" [displayOptions]="{{ showIcon: true, showTitle: true, showDescription: true }}" (itemSelected)="handleItemSelected($event)"> </aava-menu></div>
```

```
import { Component } from "@angular/core";import { AavaMenuComponent } from "@aava/play-core";@Component({ selector: "app-menu-with-descriptions", standalone: true, imports: [AavaMenuComponent],})export class MenuWithDescriptionsComponent { showMenu = false; menuItems = [ { label: "Dashboard", icon: "home", description: "View your main dashboard and overview", route: "/dashboard" }, { label: "Profile", icon: "user", description: "Manage your account settings and preferences", route: "/profile" }, { label: "Settings", icon: "settings", description: "Configure application settings and options", route: "/settings" }, { label: "Help", icon: "help-circle", description: "Get help and support documentation", route: "/help" }]; toggleMenu() { this.showMenu = !this.showMenu; } handleItemSelected(event: any) { console.log("Selected:", event.label); this.showMenu = false; }}
```

```
import { Component } from "@angular/core";import { AavaMenuComponent } from "@aava/play-core";@Component({ selector: "app-menu-with-descriptions", standalone: true, imports: [AavaMenuComponent],})export class MenuWithDescriptionsComponent { showMenu = false; menuItems = [ { label: "Dashboard", icon: "home", description: "View your main dashboard and overview", route: "/dashboard" }, { label: "Profile", icon: "user", description: "Manage your account settings and preferences", route: "/profile" }, { label: "Settings", icon: "settings", description: "Configure application settings and options", route: "/settings" }, { label: "Help", icon: "help-circle", description: "Get help and support documentation", route: "/help" }]; toggleMenu() { this.showMenu = !this.showMenu; } handleItemSelected(event: any) { console.log("Selected:", event.label); this.showMenu = false; }}
```

## Disabled Items

Menu items can be disabled to indicate unavailable options or actions that require certain conditions.

- How: Set `disabled: true` on menu items.
- When to use: For conditional actions, permission-based menus, or unavailable features.

`disabled: true`

```
<div class="menu-container"> <button (click)="toggleMenu()">Menu with Disabled Items</button> <aava-menu [items]="menuItems" [visible]="showMenu" [displayOptions]="{{ showIcon: true, showTitle: true, showDescription: true }}" (itemSelected)="handleItemSelected($event)"> </aava-menu></div>
```

```
<div class="menu-container"> <button (click)="toggleMenu()">Menu with Disabled Items</button> <aava-menu [items]="menuItems" [visible]="showMenu" [displayOptions]="{{ showIcon: true, showTitle: true, showDescription: true }}" (itemSelected)="handleItemSelected($event)"> </aava-menu></div>
```

```
import { Component } from "@angular/core"; import { AavaMenuComponent } from "@aava/play-core"; @Component({ selector: "app-menu-disabled-items", standalone: true, imports: [AavaMenuComponent], }) export class MenuDisabledItemsComponent { showMenu = false; menuItems = [ { label: "Dashboard", icon: "home", description: "View your main dashboard", route: "/dashboard", }, { label: "Profile", icon: "user", description: "Manage your account settings", route: "/profile", }, { label: "Premium Features", icon: "star", description: "Upgrade to access premium features", disabled: true, }, { label: "Advanced Analytics", icon: "bar-chart", description: "Advanced analytics (requires premium)", disabled: true, }, { label: "Help", icon: "help-circle", description: "Get help and support", route: "/help", }, ]; toggleMenu() { this.showMenu = !this.showMenu; } handleItemSelected(event: any) { console.log("Selected:", event.label); this.showMenu = false; }}
```

```
import { Component } from "@angular/core"; import { AavaMenuComponent } from "@aava/play-core"; @Component({ selector: "app-menu-disabled-items", standalone: true, imports: [AavaMenuComponent], }) export class MenuDisabledItemsComponent { showMenu = false; menuItems = [ { label: "Dashboard", icon: "home", description: "View your main dashboard", route: "/dashboard", }, { label: "Profile", icon: "user", description: "Manage your account settings", route: "/profile", }, { label: "Premium Features", icon: "star", description: "Upgrade to access premium features", disabled: true, }, { label: "Advanced Analytics", icon: "bar-chart", description: "Advanced analytics (requires premium)", disabled: true, }, { label: "Help", icon: "help-circle", description: "Get help and support", route: "/help", }, ]; toggleMenu() { this.showMenu = !this.showMenu; } handleItemSelected(event: any) { console.log("Selected:", event.label); this.showMenu = false; }}
```

## Features

### Flexible Positioning

- 8 Position Options: top, bottom, left, right, and their start/end variants
- Alignment Control: start, center, end alignment within positioned area
- Offset Configuration: Customizable spacing from trigger element
- Auto Flip: Automatic position adjustment when space is limited

### Rich Content Support

- Icons: Lucide icons with customizable size and colors
- Descriptions: Additional text for context and explanation
- Active States: Visual indication of currently active/selected items
- Disabled States: Clear indication of unavailable options

### Layout Options

- Multi-Column: Configurable column layout for extensive menus
- Responsive Design: Adapts to different screen sizes and orientations
- Flexible Sizing: Automatic width adjustment based on content
- Custom Styling: Extensive CSS custom properties for theming

### Accessibility

- Keyboard Navigation: Full keyboard support with arrow keys
- Screen Reader: Proper ARIA attributes and semantic structure
- Focus Management: Clear focus indicators and logical tab order
- High Contrast: Support for high contrast mode and color preferences

# API Reference

## Inputs

```
Property | Type | Description
items | MenuItem[] | [] | Array of menu items to display
visible | boolean | false | Controls menu visibility
itemsPerColumn | number | 3 | Number of items per column in multi-column layout
positionConfig | MenuPositionConfig | See default config below | Configuration for menu positioning
displayOptions | MenuItemDisplayOptions | See default options below | Configuration for item display
customStyles | Record | {} | Custom CSS styles to apply to the menu
```

```
items
```

```
MenuItem[]
```

```
[]
```

```
visible
```

```
boolean
```

```
false
```

```
itemsPerColumn
```

```
number
```

```
3
```

```
positionConfig
```

```
MenuPositionConfig
```

```
displayOptions
```

```
MenuItemDisplayOptions
```

```
customStyles
```

```
Record<string, string>
```

```
{}
```

## Outputs

```
Property | Type | Description
itemSelected | EventEmitter<{route?: string; label: string; item: MenuItem}> | Emitted when a menu item is clicked
```

```
itemSelected
```

```
EventEmitter<{route?: string; label: string; item: MenuItem}>
```

## MenuItem Interface

```
interface MenuItem { label: string; // Display text for the menu item description?: string; // Optional description text route?: string; // Optional route for navigation icon?: string; // Optional icon name (Lucide) disabled?: boolean; // Whether the item is disabled divider?: boolean; // Whether to show a divider line customData?: Record<string, unknown>; // Custom data for the item}
```

```
interface MenuItem { label: string; // Display text for the menu item description?: string; // Optional description text route?: string; // Optional route for navigation icon?: string; // Optional icon name (Lucide) disabled?: boolean; // Whether the item is disabled divider?: boolean; // Whether to show a divider line customData?: Record<string, unknown>; // Custom data for the item}
```

## MenuPositionConfig Interface

```
interface MenuPositionConfig { position?: MenuPosition; // Position relative to trigger alignment?: MenuAlignment; // Alignment within positioned area offset?: number; // Distance from trigger element autoFlip?: boolean; // Auto-adjust position if needed}type MenuPosition = | "top" | "bottom" | "left" | "right" | "top-start" | "top-end" | "bottom-start" | "bottom-end";type MenuAlignment = "start" | "center" | "end";
```

```
interface MenuPositionConfig { position?: MenuPosition; // Position relative to trigger alignment?: MenuAlignment; // Alignment within positioned area offset?: number; // Distance from trigger element autoFlip?: boolean; // Auto-adjust position if needed}type MenuPosition = | "top" | "bottom" | "left" | "right" | "top-start" | "top-end" | "bottom-start" | "bottom-end";type MenuAlignment = "start" | "center" | "end";
```

## MenuItemDisplayOptions Interface

```
interface MenuItemDisplayOptions { showIcon?: boolean; // Whether to show icons showTitle?: boolean; // Whether to show item titles showDescription?: boolean; // Whether to show descriptions iconSize?: number; // Size of icons in pixels titleWeight?: "normal" | "medium" | "semibold" | "bold"; // Font weight for titles descriptionSize?: "xs" | "sm" | "md" | "lg"; // Font size for descriptions}
```

```
interface MenuItemDisplayOptions { showIcon?: boolean; // Whether to show icons showTitle?: boolean; // Whether to show item titles showDescription?: boolean; // Whether to show descriptions iconSize?: number; // Size of icons in pixels titleWeight?: "normal" | "medium" | "semibold" | "bold"; // Font weight for titles descriptionSize?: "xs" | "sm" | "md" | "lg"; // Font size for descriptions}
```

## Default Configurations

```
// Default position config{ position: 'bottom', alignment: 'start', offset: 8,}// Default display options{ showIcon: true, showTitle: true, showDescription: true, iconSize: 24, titleWeight: 'medium', descriptionSize: 'sm',}
```

```
// Default position config{ position: 'bottom', alignment: 'start', offset: 8,}// Default display options{ showIcon: true, showTitle: true, showDescription: true, iconSize: 24, titleWeight: 'medium', descriptionSize: 'sm',}
```

## CSS Custom Properties

The component uses CSS custom properties for theming:

Property	Description
--menu-min-width	Minimum width of the menu
--menu-background	Background color of the menu
--menu-border-radius	Border radius of the menu
--menu-shadow	Box shadow of the menu
--menu-border	Border of the menu
--menu-padding	Padding inside the menu
--menu-margin-top	Top margin for positioning
--menu-transform-y	Transform distance for animations
--menu-transition	Transition timing for animations
--menu-z-index	Z-index for layering
--menu-columns-gap	Gap between columns
--menu-column-min-width	Minimum width of each column
--menu-item-gap	Gap between icon and content
--menu-item-padding	Padding of menu items
--menu-item-border-radius	Border radius of menu items
--menu-item-background	Background of menu items
--menu-item-color	Text color of menu items
--menu-item-hover-background	Background on hover
--menu-item-hover-color	Text color on hover

```
--menu-item-active-background | Background when active
--menu-item-active-color | Text color when active
--menu-item-disabled-background | Background when disabled
--menu-item-disabled-color | Text color when disabled
--menu-item-margin | Margin of menu items
--menu-item-first-margin-top | Top margin of first item
--menu-item-last-margin-bottom | Bottom margin of last item
--menu-item-transition | Transition timing for items
--menu-item-icon-size | Size of item icons
--menu-item-icon-color | Color of item icons
--menu-item-icon-active-color | Color of active item icons
--menu-item-icon-transition | Transition timing for icons
--menu-item-hover-icon-filter | Icon filter on hover
--menu-item-active-icon-filter | Icon filter when active
--menu-item-content-gap | Gap between title and description
--menu-item-title-font-weight | Font weight of titles
--menu-item-title-font-size | Font size of titles
--menu-item-title-color | Color of titles
--menu-item-title-active-color | Color of active titles
--menu-item-title-active-font-weight | Font weight of active titles
--menu-item-title-line-height | Line height of titles
--menu-item-title-transition | Transition timing for titles
--menu-item-description-font-size | Font size of descriptions
--menu-item-description-color | Color of descriptions
--menu-item-description-active-color | Color of active descriptions
--menu-item-description-line-height | Line height of descriptions
--menu-item-description-transition | Transition timing for descriptions
```

```
--menu-min-width
```

```
--menu-background
```

```
--menu-border-radius
```

```
--menu-shadow
```

```
--menu-border
```

```
--menu-padding
```

```
--menu-margin-top
```

```
--menu-transform-y
```

```
--menu-transition
```

```
--menu-z-index
```

```
--menu-columns-gap
```

```
--menu-column-min-width
```

```
--menu-item-gap
```

```
--menu-item-padding
```

```
--menu-item-border-radius
```

```
--menu-item-background
```

```
--menu-item-color
```

--menu-item-hover-background

--menu-item-hover-color

--menu-item-active-background

--menu-item-active-color

--menu-item-disabled-background

--menu-item-disabled-color

--menu-item-margin

--menu-item-first-margin-top

--menu-item-last-margin-bottom

--menu-item-transition

--menu-item-icon-size

--menu-item-icon-color

--menu-item-icon-active-color

--menu-item-icon-transition

--menu-item-hover-icon-filter

--menu-item-active-icon-filter

--menu-item-content-gap

--menu-item-title-font-weight

--menu-item-title-font-size

--menu-item-title-color

--menu-item-title-active-color

--menu-item-title-active-font-weight

--menu-item-title-line-height

--menu-item-title-transition

--menu-item-description-font-size

--menu-item-description-color

```
--menu-item-description-active-color
```

```
--menu-item-description-line-height
```

```
--menu-item-description-transition
```

## Best Practices

### Content Guidelines

- Clear Labels: Use concise, descriptive labels for menu items
- Consistent Icons: Use consistent iconography throughout the menu
- Helpful Descriptions: Provide context when items need explanation
- Logical Grouping: Group related items together in the menu

### Interaction Design

- Trigger Elements: Ensure trigger elements are clearly identifiable
- Positioning: Choose positioning that doesn't obscure important content
- Keyboard Support: Test keyboard navigation thoroughly
- Touch Support: Consider touch interactions on mobile devices

### Accessibility

- ARIA Labels: Provide appropriate ARIA labels for screen readers
- Focus Management: Ensure proper focus handling when menu opens/closes
- Color Contrast: Maintain sufficient contrast for all text and icons
- Motion Sensitivity: Respect user motion preferences

### Performance

- Efficient Rendering: Use OnPush change detection for better performance
- Event Handling: Properly handle click events to prevent bubbling
- Memory Management: Clean up event listeners and subscriptions
- Lazy Loading: Consider lazy loading for large menus

## Use Cases

### Context Menus

Right-click context menus for actions on selected items:

```
<aava-menu [items]="contextMenuItemItems" [visible]="showContextMenu" [positionConfig]="{ position: 'bottom-start' }" (itemSelected)="handleContextMenuAction($event)"></aava-menu>
```

```
<aava-menu [items]="contextMenuItemItems" [visible]="showContextMenu" [positionConfig]="{ position: 'bottom-start' }" (itemSelected)="handleContextMenuAction($event)"></aava-menu>
```

## Navigation Dropdowns

Dropdown navigation menus for main navigation:

```
<aava-menu [items]="navigationItems" [visible]="showNavMenu" [positionConfig]="{ position: 'bottom', alignment: 'center' }" [displayOptions]:"{ showIcon: true, showDescription: false }"></aava-menu>
```

```
<aava-menu [items]="navigationItems" [visible]="showNavMenu" [positionConfig]="{{ position: 'bottom', alignment: 'center' }}" [displayOptions]="{{ showIcon: true, showDescription: false }}></aava-menu>
```

## Action Menus

Action menus for buttons or toolbar items:

```
<aava-menu [items]="actionItems" [visible]="showActionMenu" [positionConfig]="{{ position: 'bottom-end' }}" [itemsPerColumn]="1"></aava-menu>
```

```
<aava-menu [items]="actionItems" [visible]="showActionMenu" [positionConfig]="{{ position: 'bottom-end' }}" [itemsPerColumn]="1"></aava-menu>
```

## Application Launchers

Multi-column menus for application or feature launchers:

```
<aava-menu [items]="appItems" [visible]="showAppMenu" [positionConfig]="{{ position: 'bottom', alignment: 'center' }}" [itemsPerColumn]="4" [displayOptions]="{{ showIcon: true, showDescription: true }}></aava-menu>
```

```
<aava-menu [items]="appItems" [visible]="showAppMenu" [positionConfig]="{{ position: 'bottom', alignment: 'center' }}" [itemsPerColumn]="4" [displayOptions]="{{ showIcon: true, showDescription: true }}></aava-menu>
```

## Accessibility Guidelines

### Keyboard Navigation

The menu supports full keyboard navigation:

- Enter/Space: Activate the currently focused menu item
- Arrow Keys: Navigate between menu items
- Escape: Close the menu
- Tab: Move focus to next focusable element

### Screen Reader Support

The component provides comprehensive screen reader support:

- ARIA Labels: Proper labeling for menu and menu items
- Role Attributes: Correct ARIA roles for menu structure
- State Announcements: Clear announcements of menu state changes
- Focus Indicators: Visible focus indicators for keyboard users

### Color and Contrast

- WCAG Compliance: All text and icons meet WCAG AA contrast ratios
- High Contrast Mode: Component works with system high contrast settings
- Color Independence: Information is not conveyed by color alone

### Motion and Animation

- Respects Preferences: Animation respects user motion preferences
- Reduced Motion: Provides alternative interaction for motion-sensitive users
- Clear Feedback: Visual feedback is immediate and clear