BUTTON

The Button is one of the most commonly used fundamental components of AAVA Play. It is not just a click target. It's the pulse of the interface—alive, intentional, and radiant with purpose.

The

<aava-button>

component provides a highly customizable clickable element with advanced visual effects including glass morphism, multiple interaction states, and comprehensive theming options. It supports various variants, sizes, icons, and custom styling while maintaining accessibility standards.

How to use

```
import

{

AavaButtonComponent

}

from

"@aava/play-core"

;
```

Interactive Matrix

Explore all button combinations with our interactive playground.

```
<div class="matrix-demo">

<div class="demo-header">

<!-- Matrix Table -->

<div class="matrix-table-container">

<!-- Size Headers -->

<div id="size-tabs" class="size-tabs">

<aava-tabs

[tabs]="sizeTabs"
```

```
[activeTabId]="selectedSize"
variant="button"
size="sm"
[showContentPanels]="false"
(tabChange)="onSizeTabChange($event)"
class="size-tabs-container"
></aava-tabs>
</div>
<!-- Matrix Grid -->
<div class="matrix-grid">
<!-- Column Headers -->
<div class="matrix-header">
<div class="mode-header">Mode</div>
<div class="fill-header" *ngFor="let fill of fills">{{ fill }}</div>
</div>
<!-- Matrix Rows -->
<div class="matrix-row" *ngFor="let mode of modes">
<div class="mode-label">{{ mode }}</div>
<div class="button-cell" *ngFor="let fill of fills">
<ng-container
*ngIf="getButtonConfig(mode, fill, selectedSize) as config"
>
<aava-button
*ngIf="config.available"
```

```
[label]="config.label"
[pill]="config.pill"
[outlined]="config.outlined"
[clear]="config.clear"
[size]="getSizeMapping(selectedSize)"
[iconName]="config.iconName"
[iconPosition]="config.iconPosition || 'left'"
variant="primary"
class="matrix-button"
></aava-button>
<div *ngIf="!config.available" class="unavailable">
■ Not Available
</div>
</ng-container>
</div>
</div>
</div>
</div>
</div>
</div>
```

Basic Usage

Basic button implementation with default settings.

```
<aava-button
label="Primary"
```

```
variant="primary"

(userClick)="onButtonClick($event)"

[pill]="true"

>

</aava-button>

<aava-button

label="Primary"

variant="primary"

(userClick)="onButtonClick($event)"

>

</aava-button>
```

Variants

The button component supports 9 semantic variants that control the visual appearance and meaning of the button.

```
<aava-button label="Primary" variant="primary"></aava-button>

<aava-button label="Secondary" variant="secondary"></aava-button>

<aava-button label="Success" variant="success"></aava-button>

<aava-button label="Warning" variant="warning"></aava-button>

<aava-button label="Danger" variant="danger"></aava-button>

<aava-button label="Info" variant="info"></aava-button>

<avaa-button label="Tertiary" variant="tertiary"></avaa-button>
```

Available Variants

- primary- Main call-to-action button (pink/brand color)

- secondary- Outlined style with transparent background

- success- Positive actions (green)

- warning- Cautionary actions (orange)
- danger- Destructive actions (red)
- info- Informational actions (blue)
- tertiary- Text-only action (transparent)

Sizes

Five size options to fit different layout requirements and visual hierarchies.

```
<aava-button label="Extra Small" variant="primary" size="xs"></aava-button>

<aava-button label="Small" variant="primary" size="sm"></aava-button>

<aava-button label="Medium" variant="primary" size="md"></aava-button>

<aava-button label="Large" variant="primary" size="lg"></aava-button>

<aava-button label="Extra Large" variant="primary" size="xl"></aava-button>
```

Available Sizes

- xs (Extra Small)- Extra compact size for very dense interfaces
- sm (Small)- Compact size for dense interfaces
- md (Medium)- Standard size for most use cases (default)
- lg (Large)- Prominent size for primary actions
- xl (Extra Large)- Extra large size for hero sections and CTAs

Hover Effects

Dynamic hover effects that enhance user interaction feedback.

```
<aava-button

label="Torch (Recommended)"

variant="primary"

hoverEffect="torch"

[pill]="true"
```

```
>
</aava-button>
<aava-button
label="Glow Effect"
ariant="warning"
hoverEffect="glow"
[pill]="true"
>
</aava-button>
<aava-button
label="Tint Effect"
variant="success"
hoverEffect="tint"
[pill]="true"
>
</aava-button>
<aava-button
label="Scale Effect"
variant="danger"
hoverEffect="scale"
[pill]="true"
>
</aava-button>
```

Available Hover Effects

- torch- Internal semicircular sunrise effect (default)

- glow- Outer glow with elevation

- tint- Color overlay with brightness increase

- scale- Scale transformation with elevation

Pressed Effects

Visual feedback for button press interactions with various animation styles.

```
<aava-button
label="Ripple (Recommended)"
variant="primary"
pressedEffect="ripple"
[pill]="true"
>
</aava-button>
<aava-button
label="Inset Effect"
variant="warning"
pressedEffect="inset"
[pill]="true"
>
</aava-button>
```

Available Pressed Effects

- ripple- Multi-layered ripple animation (default)

- inset- Inset shadow effect

Icons

Comprehensive icon support with flexible positioning and customization options.

```
<aava-button
label="Left Icon"
iconName="star"
iconPosition="left"
variant="primary"
iconColor="#fff"
>
</aava-button>
<aava-button
label="Right Icon"
iconName="star"
iconPosition="right"
variant="primary"
iconColor="#fff"
>
</aava-button>
<aava-button
iconName="star"
iconPosition="icon-only"
variant="primary"
iconColor="#fff"
>
</aava-button>
```

Icon Properties

- iconName- Lucide icon name

- iconPosition- Position relative to text:left,right,icon-only

- iconColor- Custom icon color (defaults to currentColor)

- iconSize- Icon size in pixels (default: 20)

States

Interactive states for different user scenarios and feedback.

```
<aava-button label="Default State" variant="primary" size="md" [pill]="true">

</aava-button>

<aava-button

label="Processing State"

variant="primary"

[processing]="true"

size="md"

[pill]="true"

iconName="loader"

iconColor="white"

iconPosition="left"

>

</aava-button>

<aava-button

label="Disabled State"

variant="primary"

[disabled]="true"

size="md"
```

```
[pill]="true"
>
</aava-button>
```

Available States

- default- Programmatically active state
- processing- Loading/async operation state with pulse animation
- disabled- Non-interactive state

Shapes & Styles

Shape modifiers and style variants for different design requirements.

```
<aava-button
label="Primary"
variant="primary"
[pill]="true"
[customStyles]="{ 'max-width': '100px' }"
></aava-button>
<aava-button
label="Secondary"
variant="secondary"
[pill]="true"
[customStyles]="{ 'max-width': '100px' }"
></aava-button>
<aava-button
label="Primary"
variant="primary"
```

```
[customStyles]="{ 'max-width': '100px' }"

></aava-button>

<aava-button

label="Secondary"

variant="secondary"

[customStyles]="{ 'max-width': '100px' }"

></aava-button>
```

Available Shapes

- default- Standard rectangular with border radius
- pill- Fully rounded corners (50px border radius)

Style Variants

- outlined- Transparent background with colored border
- clear- Transparent background with no border, uses variant text colors
- customStyles- Allows to apply inline CSS styles directly to the component.

This property accepts a key-value pair object where the key is the CSS property name and the value is the corresponding CSS value.

Events

The button component emits events for user interactions.

```
<aava-button

label="Click Handler"

variant="primary"

(userClick)="onButtonClick()"

[customStyles]="{ 'max-width': '200px' }"

[pill]="true"

>
```

</aava-button>

Available Events

- userClick- Emitted on button click or keyboard activation (Enter/Space)

Accessibility

The button component follows WAI-ARIA accessibility guidelines:

- Proper keyboard navigation (Enter and Space key support)

- ARIA attributes for screen readers (aria-disabled,aria-pressed)

- Focus management with visible focus indicators

- Semantic button element usage

API Reference

Inputs

[TABLE]

Property | Type | Default | Description

label | string | '' | Button text content

variant | ButtonVariant | 'default' | Visual variant: | 'default' | , | 'primary' | , | 'secondary' | , | 'success' | , | 'warning' | , | 'danger' | , | 'info'

size | ButtonSize | 'md' | Button size: | 'xs' | , | 'sm' | , | 'md' | , | 'lg' | , | 'xl'

state | ButtonState | 'default' | Interaction state: | 'default' | , | 'hover' | , | 'active' | , | 'disabled' | , | 'processing' | , | 'focus'

hoverEffect | ButtonHoverEffect | 'torch' | Hover effect: | 'torch' | , | 'glow' | , | 'tint' | , | 'scale' | , | 'none'

pressedEffect | ButtonPressedEffect | 'ripple' | Pressed effect: | 'ripple' | , | 'inset' | , | 'solid' | , | 'none'

processingEffect | ButtonProcessingEffect | 'pulse' | Processing effect: | 'pulse' | , | 'none'

focusEffect | ButtonFocusEffect | 'border' | Focus effect: | 'border' | , | 'none'

disabledEffect | ButtonDisabledEffect | 'dim' | Disabled effect: | 'dim' | , | 'none'

disabled | boolean | false | Whether button is disabled processing | boolean | false | Whether button is in processing state

pill | boolean | false | Whether to use pill shape

outlined | boolean | false | Whether to use outlined style variant

clear | boolean | false | Whether to use clear style variant (transparent, no border)

width | string | '' | Custom width value

height | string | '' | Custom height value

gradient | string | undefined | Legacy | – use | customStyles | instead

background | string | undefined | Legacy | – use | customStyles | instead

color | string | undefined | Legacy | – use | customStyles | instead

dropdown | boolean | false | Legacy | – use separate dropdown component

glassVariant | ButtonGlassVariant | 'glass-10' | Default recommended glass intensity

customStyles | Record<string, string> | {} | CSS custom properties override

iconName | string | '' | Lucide icon name

iconColor | string | '' | Custom icon color

iconSize | number | 20 | Icon size in pixels

iconPosition | 'left' | 'right' | 'only' | 'left' | Icon position relative to text

[/TABLE]

Outputs

[TABLE]

Event | Type | Description

userClick | EventEmitter<Event> | Emitted when button is clicked or activated via keyboard

[/TABLE]

Design Tokens & Theming

AAVA Play buttons use semantic design tokens for all surfaces, spacing, radius, and motion. While global tokens define the visual language of the system, buttons expose a set of scoped override tokens

that allow you to fine-tune appearance without breaking consistency.

Use these

only when necessary

—for instance, to adjust a button's vertical padding inside a dense UI or to sharpen the radius for compact layouts.

Available Design Tokens for Button

[TABLE]

Token	Purpose	Default Value
--button-size-xsm-padding	Padding for extra small size buttons	Theme-based
--button-size-sm-padding	Padding for small size buttons	Theme-based
--button-size-md-padding	Padding for medium size buttons	Theme-based
--button-size-lg-padding	Padding for large size buttons	Theme-based
--button-size-xlg-padding	Padding for extra large size buttons	Theme-based
--button-size-xsm-height	Height for extra small size buttons	Theme-based
--button-size-sm-height	Height for small size buttons	Theme-based
--button-size-md-height	Height for medium size buttons	Theme-based
--button-size-lg-height	Height for large size buttons	Theme-based
--button-size-xlg-height	Height for extra large size buttons	Theme-based
--button-border-radius	Border radius for default shape	Theme-based

[/TABLE]

[TABLE]

Token	Purpose	Default Value
--button-font-weight	Font weight for button text	Theme-based
--button-transition	Default transition animation	Theme-based

[/TABLE]

[TABLE]

Token	Purpose	Default Value
--button-icon-margin	Margin around icons	Theme-based

[/TABLE]

Token Override Example

You can define overrides in your theme configuration or component styles:

```
/* Custom button theming */
.my-compact-buttons
{
--button-size-md-padding
:
8
px
16
px
;
--button-border-radius
:
4
px
;
--button-transition
:
100
```

ms

ease

;

}

This would make buttons more compact, sharper, and snappier—ideal for dense interfaces or admin tools.

These tokens are opt-in overrides. If not set, the button will inherit global styles via the design system tokens.

Best Practices

Design Guidelines

• Use semantic variants- Choose variants that match the action's intent (primary for main actions, danger for destructive actions)

• Default variant- Usedefaultvariant for standard buttons;primaryfor main CTAs

• Size selection- Usemediumas the default size;extra small/extra largefor extreme cases only

• Effects system- Default effects work well together; customize only when needed

• Icon usage- Useicon-onlyfor compact interfaces,left/rightfor labeled actions

• Consistent sizing- Match button sizes to surrounding elements and visual hierarchy

Accessibility

• Always provide meaningful labels- Even for icon-only buttons, use proper ARIA labels

• Keyboard navigation- Ensure all interactive elements are keyboard accessible

• Focus indicators- Maintain clear focus states for navigation

• Screen reader support- Use semantic HTML and proper ARIA attributes

• Color contrast- Ensure sufficient contrast for all variants and states

Performance

• Avoid excessive custom styling- Use built-in variants when possible

- Debounce rapid clicks- Prevent accidental multiple submissions
- Optimize icon loading- Use icon systems efficiently
- Consider bundle size- Import only needed effects and variants

TEXTBOX

The

<aava-textbox>

component provides a highly sophisticated text input element with glass morphism effects, advanced content projection system, multiple processing animations, and comprehensive Angular forms integration. It supports semantic variants, validation states, and extensive customization options.

How to use

```
import

{

AavaTextboxComponent

}

from

"@aava/play-core"

;
```

Basic Usage

Simple textbox implementation with label, placeholder, and two-way data binding.

```
<aava-textbox

label="Basic Input"

placeholder="Enter text here"

(change)="onTextboxChange($event)"

></aava-textbox>
```

Variants

The textbox component supports 6 semantic variants that control visual appearance and focus colors.

```
<aava-textbox

label="Default Variant"
```

```
placeholder="Default variant..."
variant="default"
(change)="onTextboxChange($event)"
></aava-textbox>
<aava-textbox
label="Primary Variant"
placeholder="Primary variant..."
variant="primary"
(change)="onTextboxChange($event)"
></aava-textbox>
<aava-textbox
label="Success Variant"
placeholder="Success variant..."
variant="success"
(change)="onTextboxChange($event)"
></aava-textbox>
<aava-textbox
label="Error Variant"
placeholder="Error variant..."
variant="error"
(change)="onTextboxChange($event)"
></aava-textbox>
<aava-textbox
label="Warning Variant"
placeholder="Warning variant..."
```

```
variant="warning"
(change)="onTextboxChange($event)"
></aava-textbox>
<aava-textbox
label="Info Variant"
placeholder="Info variant..."
variant="info"
(change)="onTextboxChange($event)"
></aava-textbox>
```

Available Variants

- default- Standard neutral appearance with brand primary focus
- primary- Primary variant with enhanced brand focus color
- success- Positive states and confirmations (green)
- error- Error states and validation failures (red)
- warning- Warning states and cautions (orange/yellow)
- info- Informational states and tips (blue)

Sizes

Five size options to accommodate different interface densities and layout requirements.

```
<aava-textbox
label="Extra Small"
placeholder="Extra Small..."
size="xs"
></aava-textbox>
<aava-textbox label="Small" placeholder="Small..." size="sm"></aava-textbox>
```

```
<aava-textbox label="Medium" placeholder="Medium..." size="md"></aava-textbox>

<aava-textbox label="Large" placeholder="Large..." size="lg"></aava-textbox>

<aava-textbox

label="Extra Large"

placeholder="Extra Large..."

size="xl"

></aava-textbox>
```

Available Sizes

- xs (Extra Small)- Extra small size for very compact interfaces
- sm (Small)- Small size for dense interfaces
- md (Medium)- Medium size for most use cases (default)
- lg (Large)- Large size for prominent inputs
- xl (Extra Large)- Extra large size for emphasis and accessibility

Icons & Affixes

Advanced content projection system supporting icons, prefixes, and suffixes through Angular's content projection.

```
<aava-textbox label="Amount" placeholder="0.00" type="number" (on)>

<span slot="prefix">$</span>

<span slot="suffix">USD</span>

</aava-textbox>

<aava-textbox

label="Password"

[type]="showPassword ? 'text' : 'password'"

placeholder="Enter password"

>
```

```
<aava-icon
slot="icon-end"
[iconName]="showPassword ? 'eye' : 'eye-off'"
(click)="togglePasswordVisibility()"
></aava-icon>
</aava-textbox>
<aava-textbox
label="With Icon Separator (Start)"
placeholder="Search..."
[iconSeparator]="true"
>
<aava-icon slot="icon-start" iconName="search"></aava-icon>
</aava-textbox>
<aava-textbox
label="With Icon Separator (End)"
placeholder="Clear..."
[iconSeparator]="true"
>
<aava-icon slot="icon-end" iconName="x"></aava-icon>
</aava-textbox>
```

Content Projection Slots

- icon-start- Icons at the beginning of the input
- icon-end- Icons at the end of the input
- prefix- Text or elements before the input text
- suffix- Text or elements after the input text

Input Masking

Advanced input masking system powered by ngx-mask for formatted input patterns like phone numbers, dates, currency, and credit cards.

```
<!-- Phone Number Masking -->
<aava-textbox
label="Phone Number"
placeholder="(123) 456-7890"
[mask]="maskPhone"
>
</aava-textbox>
<!-- Currency Masking -->
<aava-textbox
label="Currency"
placeholder="0.00"
[mask]="maskCurrency"
[maskThousandSeparator]="thousand"
[maskDecimalMarker]="decimal"
>
<span slot="prefix">$</span>
</aava-textbox>
<!-- Date Masking -->
<aava-textbox label="Date" placeholder="MM/DD/YYYY" [mask]="maskDate">
</aava-textbox>
<!-- Custom Pattern Masking -->
```

```
<aava-textbox
label="Custom Pattern (AA-0000)"
placeholder="AB-1234"
[mask]="customMask"
[maskPatterns]="customPatterns"
>
</aava-textbox>
<aava-textbox
label="Phone Number (with Built-in Dropdown)"
[placeholder]="currentCountryPlaceholder"
[(ngModel)]="countryPhoneValue"
[mask]="currentCountryMask" [
prefixDropdown]="true"
[prefixDropdownOptions]="countryOptions"
[selectedPrefixOption]="selectedCountry"
(prefixDropdownSelect)="onCountrySelect($event)"
>
</aava-textbox>
<aava-textbox
label="Phone Number (with Custom Slot Dropdown)"
[placeholder]="slotCurrentPlaceholder"
[(ngModel)]="slotCountryPhoneValue"
[mask]="slotCurrentMask"
(clickOutSide)="closeSlotDropdown()"
>
```

```
<!-- Custom dropdown projected into prefix slot -->
<div slot="prefix" class="custom-country-dropdown">
<div class="dropdown-trigger" (click)="toggleSlotDropdown()" tabindex="0" role="button"
[attr.aria-expanded]="slotIsDropdownOpen">
<span class="country-label">{{ slotSelectedCountry.label }}</span>
<svg class="chevron-icon" [class.open]="slotIsDropdownOpen" width="16" height="16" viewBox="0 0 24 24"
fill="none" stroke="currentColor">
<polyline points="6 9 12 15 18 9"></polyline>
</svg>
</div>
</div>
<div slot="dropdown" class="custom-country-dropdown">
<div class="dropdown-menu" *ngIf="slotIsDropdownOpen" role="listbox" tabindex="-1">
<div *ngFor="let option of countryOptions" class="dropdown-item"
[class.selected]="option.value === slotSelectedCountryCode"
(click)="selectSlotCountry(option); $event.stopPropagation()" (keydown.enter)="
selectSlotCountry(option); $event.stopPropagation()
" (keydown.space)="
selectSlotCountry(option); $event.stopPropagation()
" tabindex="0" role="option" [attr.aria-selected]="option.value === slotSelectedCountryCode">
{{ option.label }}
</div>
</div>
</div>
```

```
</aava-textbox>
```

Masking Features

- Pattern-Based Input: Define custom input patterns using mask syntax
- Real-time Formatting: Automatic formatting as users type
- Special Character Handling: Control how special characters are processed
- Prefix/Suffix Support: Add currency symbols, units, or other prefixes/suffixes
- Validation Integration: Works seamlessly with existing validation system
- Accessibility: Maintains proper ARIA attributes and keyboard navigation

Common Mask Patterns

- Phone Numbers:(000) 000-0000for US format
- Dates:00/00/0000for MM/DD/YYYY format
- Currency:separator.2with thousand separators and decimal places
- Credit Cards:0000 0000 0000 0000for card number format
- Custom Patterns: Define your own patterns using0,9,A,Splaceholders

States & Validation

Comprehensive validation system with error messages, helper text, and various input states.

```
<aava-textbox

label="Comments"

placeholder="Share your thoughts..."

helper="Please provide detailed feedback to help us improve our service."

(change)="onTextboxChange($event)"

></aava-textbox>

<aava-textbox

label="Email"
```

```
placeholder="Enter your email..."
error="Please enter a valid email address."
(change)="onTextboxChange($event)"
></aava-textbox>
<aava-textbox
label="Required Field"
placeholder="This field is required..."
[required]="true"
(change)="onTextboxChange($event)"
></aava-textbox>
```

Available States

- Normal- Default input state
- Focused- Active input with enhanced border and shadow
- Disabled- Non-interactive state with dimmed appearance
- Readonly- Display-only state with modified styling
- Error- Validation error state with red styling and error message
- Required- Indicates mandatory fields with asterisk

Validation Features

Error Handling:

- Error messages- Display validation errors with icon
- ARIA compliance- Properaria-invalidandaria-describedbyattributes
- Visual feedback- Error variant styling with red colors
- Icon integration- Alert icons automatically shown with errors

Helper Text:

- Guidance messages- Helpful instructions below input

- Icon support- Info icons automatically shown with helper text
- Conditional display- Helper text hidden when errors are present
- Accessibility- Proper ARIA relationships

Required Fields:

- Visual indicators- Asterisk (*) displayed for required fields
- Label integration- Required indicator integrated with label
- Form validation- Works with Angular form validation

Processing Effects

Advanced processing states with multiple animation options for loading and async operations.

```
<aava-textbox
[(ngModel)]="processingValue"
label="Processing State"
placeholder="Processing..."
[processing]="true"
(change)="onProcessingChange($event)"
></aava-textbox>
<aava-textbox
[(ngModel)]="shimmerValue"
label="Shimmer Effect"
placeholder="Validating..."
[processing]="true"
processingEffect="shimmer"
(change)="onShimmerChange($event)"
></aava-textbox>
```

```
<aava-textbox

[(ngModel)]="gradientValue"

label="Gradient Border"

placeholder="Submitting..."

[processing]="true"

[processingGradientBorder]="true"

(change)="onGradientChange($event)"

></aava-textbox>
```

Available Processing Effects

- border-pulse- Pulsing border animation (default)
- shimmer- Text shimmer effect within input
- gradient-border- Animated multi-color border gradient

Effects System

Modern effects system following Text Input Specifications for consistent visual behavior and accessibility.

Hover Effects

- tint- Subtle color tinting on hover (recommended)
- glow- Enhanced glow effect on hover (for interactive elements)

Processing States

- Default Processing- Border pulse animation with customizable colors
- Shimmer Effect- Text shimmer animation as alternative to border pulse
- Gradient Border- Animated multi-color gradient border for processing state
- Custom Colors- Configurable gradient colors for brand consistency

Accessibility Features

- High Contrast Support- Effects respect high contrast mode settings

- Reduced Motion- Animations respect user's motion preferences

- Focus Indicators- Clear focus states maintained with all effects

- Screen Reader Support- Proper ARIA attributes for all interactive states

API Reference

Inputs

[TABLE]

Property | Type | Default | Description

label | string | '' | Visible label text above input

placeholder | string | '' | Placeholder text shown when empty

variant | TextboxVariant | 'default' | Visual variant: | 'default' | , | 'primary' | , | 'success' | , | 'error' | , | 'warning' | , | 'info'

size | TextboxSize | 'md' | Input size: | 'xs' | , | 'sm' | , | 'md' | , | 'lg' | , | 'xl'

disabled | boolean | false | Whether input is disabled

readonly | boolean | false | Whether input is read-only

error | string | '' | Error message to display

helper | string | '' | Helper text to display

required | boolean | false | Whether input is required

fullWidth | boolean | false | Whether input takes full container width

type | string | 'text' | HTML input type

maxlength | number | '' | Maximum character length

minlength | number | '' | Minimum character length

autocomplete | string | '' | HTML autocomplete attribute

id | string | '' | Custom element ID

name | string | '' | HTML name attribute

icon | string | '' | Icon name to display iconPosition | 'start' | 'end' | 'start' | Position of the icon relative to input

iconSeparator | boolean | false | Whether to show separator between icon and input

iconSpacing | 'compact' | 'normal' | 'relaxed' | 'normal' | Icon spacing variant

inputKind | 'text' | 'phone' | 'currency' | 'password' | 'text' | Type of input for specialized behavior

inputKindLabel | string | '' | Label for specialized input types (e.g., country code)

phone | boolean | false | Enable phone input functionality

labelPosition | 'start' | 'end' | 'start' | Position of country prefix label for phone inputs

[/TABLE]

Masking Properties

[TABLE]

Property | Type | Default | Description

mask | string | null | null | Input mask pattern (e.g., "(000) 000-0000" for phone)

maskPrefix | string | '' | Prefix to add before masked value (e.g., "$" for currency)

maskSuffix | string | '' | Suffix to add after masked value (e.g., "%" for percentage)

maskDropSpecialCharacters | boolean | string[] | true | Whether to drop special characters from value

maskShowMaskTyped | boolean | false | Show mask characters as user types

maskThousandSeparator | string | '' | Thousand separator character (e.g., "," for numbers)

maskDecimalMarker | '.' | ',' | ['.', ','] | '.' | Decimal marker character

maskPatterns | Record<string, object> | {} | Custom mask patterns for special characters

maskValidation | boolean | false | Enable mask validation

maskAllowNegativeNumbers | boolean | false | Allow negative numbers in numeric masks

maskLeadZeroDateTime | boolean | false | Show leading zeros in date/time masks

[/TABLE]

Effects System Properties

[TABLE]

Property | Type | Default | Description

hoverEffect | 'tint' | 'glow' | '' | Hover effect: Tint (recommended) or Glow

pressedEffect | 'solid' | 'solid' | Pressed effect: Solid (recommended and only allowed)

processing | boolean | false | Processing state - triggers border pulse by default

processingEffect | 'shimmer' | '' | Alternative processing effect: Text shimmer animation

processingGradientBorder | boolean | false | Show animated gradient border for processing state

processingGradientColors | string[] | ['#e91e63', '#fee140', '#ff9800', '#047857', '#ff9800', '#fee140', '#e91e63'] | Colors for processing gradient border

decorativeEffect | 'glowBox' | 'borderFlow' | 'attention' | 'wave' | '' | Ambient decorative effects for future use

disabledState | 'grey' | 'grey' | Disabled state appearance: Grey (recommended and only allowed)

customStyles | Record<string, string> | '' | CSS custom properties override for advanced theming

[/TABLE]

Outputs

[TABLE]

Event | Type | Description

iconStartClick | EventEmitter<Event> | Emitted when the start icon is clicked

iconEndClick | EventEmitter<Event> | Emitted when the end icon is clicked

clickOutSide | EventEmitter<boolean> | Emitted when a click occurs outside the input

change | EventEmitter<Event> | Emitted when the input value changes

blur | EventEmitter<Event> | Emitted when the input loses focus

focus | EventEmitter<Event> | Emitted when the input gains focus

input | EventEmitter<Event> | Emitted on every input event

prefixSelect | EventEmitter<{ label: string; value: string; }> | Emitted when a prefix option is selected

suffixSelect | EventEmitter<{ label: string; value: string; }> | Emitted when a suffix option is selected

[/TABLE]

Properties

[TABLE]

Property	Type	Description
value	string	Current input value
isFocused	boolean	Whether input currently has focus
hasError	boolean	Whether input has error state
hasHelper	boolean	Whether input has helper text

[/TABLE]

Methods

[TABLE]

Method	Parameters	Return Type	Description
setValue()	value: string	void	Set input value programmatically
writeValue()	value: string	void	Set input value (ControlValueAccessor)
registerOnChange()	fn: (value: string) => void	void	Register change callback
registerOnTouched()	fn: () => void	void	Register touched callback
setDisabledState()	isDisabled: boolean	void	Set disabled state

[/TABLE]

Content Projection Slots

[TABLE]

Slot	Description
icon-start	Icons displayed at the start of the input
icon-end	Icons displayed at the end of the input
prefix	Content displayed before the input text
suffix	Content displayed after the input text

[/TABLE]

CSS Custom Properties

[TABLE]

Property	Description
--textbox-glass-default-background	Background color with glass effect
--textbox-glass-default-blur	Backdrop blur amount for glass effect
--textbox-glass-default-border	Border color for default state
--textbox-glass-default-shadow	Box shadow for glass effect
--textbox-border-radius	Border radius of the input container
--textbox-transition	Transition animation duration
--textbox-input-font	Font properties for input text
--textbox-input-color	Text color for input
--textbox-input-padding	Padding inside the input
--textbox-input-min-height	Minimum height of the input
--textbox-label-font	Font properties for label
--textbox-label-color	Text color for label
--textbox-label-weight	Font weight for label
--textbox-placeholder-color	Color for placeholder text
--textbox-error-color	Color for error messages and state
--textbox-helper-color	Color for helper text
--textbox-icon-color	Color for icons in normal state
--textbox-icon-focus-color	Color for icons when focused
--textbox-variant-default	Colors for default variant
--textbox-variant-primary	Colors for primary variant

--textbox-variant-success | Colors for success variant

--textbox-variant-error | Colors for error variant

--textbox-variant-warning | Colors for warning variant

--textbox-variant-info | Colors for info variant

--textbox-size-xs-padding | Padding for extra small size

--textbox-size-sm-padding | Padding for small size

--textbox-size-md-padding | Padding for medium size

--textbox-size-lg-padding | Padding for large size

--textbox-size-xl-padding | Padding for extra large size

--textbox-size-xs-height | Height for extra small size

--textbox-size-sm-height | Height for small size

--textbox-size-md-height | Height for medium size

--textbox-size-lg-height | Height for large size

--textbox-size-xl-height | Height for extra large size

[/TABLE]

Best Practices

Design Guidelines

- Choose appropriate variants- Use semantic variants that match the context
- Provide clear labels- Always include descriptive labels for accessibility
- Use helper text wisely- Provide guidance without cluttering the interface
- Handle errors gracefully- Show clear, actionable error messages
- Optimize for touch- Ensure adequate touch targets for mobile users
- Consider glass intensity- Use glass-10 for most cases, glass-50 for emphasis
- Select appropriate sizes- Use xs for very compact layouts, xl for emphasis and accessibility

- Implement masking thoughtfully- Choose mask patterns that match user expectations and data format

- Balance effects and performance- Use effects system for enhanced UX while maintaining performance

Accessibility

- Proper labeling- Label elements correctly associated with inputs

- ARIA attributes- Usearia-invalid,aria-describedby,aria-required

- Error announcement- Error messages announced to screen readers

- Keyboard navigation- Full keyboard support for all interactions

- Focus management- Visible focus indicators and proper focus order

- Icon accessibility- Icons properly labeled and keyboard accessible

Performance

- Validate appropriately- Use client and server validation together

- Debounce input events- For real-time validation and API calls

- Optimize re-renders- Use OnPush change detection strategy

- Efficient icon handling- Load icons efficiently and cache appropriately

Form Integration

- Test with forms- Ensure proper integration with your form handling

- Handle validation- Implement comprehensive validation strategies

- Consider reset behavior- Define clear reset and initial state behavior

- Support reactive forms- Proper ControlValueAccessor implementation

Masking Best Practices

- Choose intuitive patterns- Use mask patterns that users expect (e.g., phone formats)

- Handle edge cases- Consider what happens when users paste or clear masked input

- Provide clear examples- Use placeholders that show the expected format

- Test accessibility- Ensure screen readers can properly announce masked input

- Consider internationalization- Use appropriate separators and formats for different locales
- Balance flexibility- Allow users to edit parts of masked input without losing context

TEXTAREA

A powerful multi-line text input component designed for collecting longer text content with comprehensive form integration, validation support, and accessibility features. Supports Angular Forms (reactive and template-driven), custom styling, and advanced features like processing gradient borders.

How to use

```
import

{

AavaTextareaComponent

}

from

"@aava/play-core"

;
```

Basic Usage

Simple textarea implementations with labels and basic functionality.

```
<aava-textarea

placeholder="Enter your text here..."

[rows]="4"

(change)="onTextareaChange($event)"

[processing]="true"

variant="primary"

></aava-textarea>
```

Sizes

Three size variants to accommodate different interface requirements and content needs.

```
<aava-textarea

label="Extra Small"
```

```
placeholder="Extra small textarea..."
size="xs"
[rows]="2"
(change)="onTextareaChange($event)"
></aava-textarea>
<aava-textarea
label="Small"
placeholder="Small textarea..."
size="sm"
[rows]="2"
(change)="onTextareaChange($event)"
></aava-textarea>
<aava-textarea
label="Medium"
placeholder="Medium textarea..."
size="md"
[rows]="3"
(change)="onTextareaChange($event)"
></aava-textarea>
<aava-textarea
label="Large"
placeholder="Large textarea..."
size="lg"
[rows]="4"
(change)="onTextareaChange($event)"
```

```
></aava-textarea>

<aava-textarea

label="Extra Large"

placeholder="Extra large textarea..."

size="xl"

[rows]="4"

(change)="onTextareaChange($event)"

></aava-textarea>
```

Available Sizes

- xs (Extra Small): Very compact, ideal for tight spaces or dense forms
- sm (Small): Compact for space-constrained interfaces
- md (Medium): Standard size for most use cases (default)
- lg (Large): Prominent for primary content input areas
- xl (Extra Large): Very prominent, used for standout or high-visibility inputs

Variants

Color variants for different contexts, states, and semantic meanings.

```
<aava-textarea

placeholder="Default variant..."

variant="default"

[rows]="3"

(change)="onTextareaChange($event)"

></aava-textarea>

<aava-textarea

placeholder="Primary variant..."
```

```
variant="primary"
[rows]="3"
(change)="onTextareaChange($event)"
></aava-textarea>
<aava-textarea
placeholder="Success variant..."
variant="success"
[rows]="3"
(change)="onTextareaChange($event)"
></aava-textarea>
<aava-textarea
placeholder="Error variant..."
variant="error"
[rows]="3"
(change)="onTextareaChange($event)"
></aava-textarea>
<aava-textarea
placeholder="Warning variant..."
variant="warning"
[rows]="3"
(change)="onTextareaChange($event)"
></aava-textarea>
<aava-textarea
placeholder="Info variant..."
variant="info"
```

```
[rows]="3"
(change)="onTextareaChange($event)"
></aava-textarea>
```

Style Variants

- default: Standard neutral appearance
- primary: Brand color for important inputs
- success: Green for positive confirmation states
- error: Red for validation errors and critical states
- warning: Orange for caution and attention states
- info: Blue for informational content and tips

Accessibility

Built-in accessibility features ensuring inclusive user experience.

```
<!-- Labeled Textarea -->
<aava-textarea
label="Description"
placeholder="Enter your description..."
[rows]="3"
(change)="onTextareaChange($event)"
></aava-textarea>
<!-- Textarea with Helper Text -->
<aava-textarea
label="Comments"
placeholder="Share your thoughts..."
helper="Please provide detailed feedback to help us improve our service."
```

```
[rows]="3"
(change)="onTextareaChange($event)"
></aava-textarea>
<!-- Textarea with Error State -->
<aava-textarea
label="Email"
placeholder="Enter your email..."
error="Please enter a valid email address."
[rows]="3"
(change)="onTextareaChange($event)"
></aava-textarea>
<!-- Textarea with Error State -->
<aava-textarea
label="Required Field"
placeholder="This field is required..."
[required]="true"
[rows]="3"
(change)="onTextareaChange($event)"
></aava-textarea>
<!-- Textarea with Character Limit -->
<aava-textarea
label="Bio"
placeholder="Tell us about yourself (max 200 characters)..."
[maxlength]="200"
[rows]="3"
```

```
(change)="onTextareaChange($event)"

></aava-textarea>
```

Accessibility Features

• Keyboard Navigation: Full keyboard support with proper focus management

• Screen Reader Support: Semantic HTML and ARIA attributes

• Error Announcements: Live regions for dynamic error messaging

• Focus Indicators: Clear visual focus states for navigation

• Label Association: Proper form control labeling and description

• High Contrast: Support for high contrast accessibility modes

API Reference

Inputs

[TABLE]

Property | Type | Default | Description

label | string | '' | Label text displayed above the textarea

placeholder | string | '' | Placeholder text shown when textarea is empty

variant | 'default' | 'primary' | 'success' | 'error' | 'warning' | 'info' | 'default' | Color variant for styling

size | 'xs' | , | 'sm' | , | 'md' | , | 'lg' | , | 'xl' | 'md' | Size variant of the textarea

disabled | boolean | false | Whether the textarea is disabled

readonly | boolean | false | Whether the textarea is read-only

error | string | '' | Error message to display

helper | string | '' | Helper text to provide guidance

rows | number | 3 | Number of visible text lines

id | string | '' | HTML id attribute for the textarea

name | string | '' | HTML name attribute for form submission

maxlength	number	''	Maximum number of characters allowed
minlength	number	''	Minimum number of characters required
required	boolean	false	Whether the field is required
fullWidth	boolean	false	Whether to take full width of container
style	Record<string, string>	''	Custom CSS styles object
resizable	boolean	true	Whether the textarea can be resized vertically
autoResize	boolean	false	Whether to automatically resize height to fit content
processing	boolean	false	Whether to show processing state
processingGradientBorder	boolean	false	Whether to show animated gradient border
processingGradientColors	string[]	See below	Array of colors for gradient animation

[/TABLE]

Outputs

[TABLE]

Event	Type	Description
blur	EventEmitter<Event>	Emitted when the input loses focus
focus	EventEmitter<Event>	Emitted when the input gains focus
input	EventEmitter<Event>	Emitted on every input change
change	EventEmitter<Event>	Emitted when the input value changes
iconStartClick	EventEmitter<Event>	Emitted when the start icon is clicked
iconEndClick	EventEmitter<Event>	Emitted when the end icon is clicked

[/TABLE]

Properties

[TABLE]

Property	Type	Description
value	string	Current textarea value

isFocused	boolean	Whether textarea currently has focus
hasError	boolean	Whether textarea has error state
hasHelper	boolean	Whether textarea has helper text

[/TABLE]

Methods

[TABLE]

Method	Parameters	Return Type	Description
writeValue()	value: string	void	Set textarea value (ControlValueAccessor)
registerOnChange()	fn: (value: string) => void	void	Register change callback
registerOnTouched()	fn: () => void	void	Register touched callback
setDisabledState()	isDisabled: boolean	void	Set disabled state

[/TABLE]

CSS Custom Properties

[TABLE]

Property	Description
--textbox-gap	Base gap between textarea elements
--textbox-gap-xs	Gap for extra small size variant
--textbox-gap-sm	Gap for small size variant
--textbox-gap-lg	Gap for large size variant
--textbox-gap-xl	Gap for extra large size variant
--textbox-label-font	Font family for labels
--textbox-label-color	Color for label text
--textbox-label-weight	Font weight for label text
--textbox-xs-label-font-size	Label font size for extra small variant

--textbox-xs-label-weight | Label font weight for extra small variant

--textbox-xs-label-line-height | Line height for extra small label

--textbox-xs-label-font-style | Font style for extra small label

--textbox-sm-label-font-size | Label font size for small variant

--textbox-sm-label-weight | Label font weight for small variant

--textbox-sm-label-line-height | Line height for small label

--textbox-sm-label-font-style | Font style for small label

--textbox-md-label-font-size | Label font size for medium variant

--textbox-md-label-weight | Label font weight for medium variant

--textbox-md-label-line-height | Line height for medium label

--textbox-md-label-font-style | Font style for medium label

--textbox-lg-label-font-size | Label font size for large variant

--textbox-lg-label-weight | Label font weight for large variant

--textbox-lg-label-line-height | Line height for large label

--textbox-lg-label-font-style | Font style for large label

--textbox-xl-label-font-size | Label font size for extra large variant

--textbox-xl-label-weight | Label font weight for extra large variant

--textbox-xl-label-line-height | Line height for extra large label

--textbox-xl-label-font-style | Font style for extra large label

--textbox-required-color | Color for required field indicator

--textbox-textarea-container-padding | Padding around textarea container

--textbox-background | Background color of the textarea

--glass-50-border | Glass border style for textarea

--textbox-textarea-hover-primary | Primary hover color for textarea

--textbox-textarea-hover-border-width | Border width on hover

--textbox-textarea-focus-primary | Primary color for focus state

--textbox-border-error-color | Border color when there is an error

--textbox-background-disabled | Background for disabled state

--textbox-border-disabled-color | Border color for disabled state

--textbox-background-readonly | Background for readonly state

--textbox-border-readonly-color | Border color for readonly state

--textbox-xs-border-radius | Border radius for extra small variant

--textbox-sm-border-radius | Border radius for small variant

--textbox-md-border-radius | Border radius for medium variant

--textbox-lg-border-radius | Border radius for large variant

--textbox-xl-border-radius | Border radius for extra large variant

--textbox-input-xs-padding | Padding inside extra small input

--textbox-input-sm-padding | Padding inside small input

--textbox-input-md-padding | Padding inside medium input

--textbox-input-lg-padding | Padding inside large input

--textbox-input-xl-padding | Padding inside extra large input

--textbox-input-font | Font family for input text

--textbox-input-color | Text color for input

--textbox-input-padding | General input padding

--textbox-textarea-min-height | Minimum height of textarea

--textbox-textarea-min-height-xs | Minimum height for extra small variant

--textbox-textarea-min-height-sm | Minimum height for small variant

--textbox-textarea-min-height-md | Minimum height for medium variant

--textbox-textarea-min-height-lg | Minimum height for large variant

--textbox-textarea-min-height-xl | Minimum height for extra large variant

--textbox-input-xs-font-size | Font size for extra small input

--textbox-input-xs-font-weight | Font weight for extra small input

--textbox-input-xs-font-style | Font style for extra small input

--textbox-input-xs-line-height | Line height for extra small input

--textbox-input-sm-font-size | Font size for small input

--textbox-input-sm-font-weight | Font weight for small input

--textbox-input-sm-font-style | Font style for small input

--textbox-input-sm-line-height | Line height for small input

--textbox-input-md-font-size | Font size for medium input

--textbox-input-md-font-weight | Font weight for medium input

--textbox-input-md-font-style | Font style for medium input

--textbox-input-md-line-height | Line height for medium input

--textbox-input-lg-font-size | Font size for large input

--textbox-input-lg-font-weight | Font weight for large input

--textbox-input-lg-font-style | Font style for large input

--textbox-input-lg-line-height | Line height for large input

--textbox-input-xl-font-size | Font size for extra large input

--textbox-input-xl-font-weight | Font weight for extra large input

--textbox-input-xl-font-style | Font style for extra large input

--textbox-input-xl-line-height | Line height for extra large input

--textbox-affix-padding | Padding for affix elements

--textbox-affix-color | Color of affix elements

--textbox-affix-font-size | Font size of affix elements

--textbox-affix-background | Background of affix elements

--textbox-affix-border-radius | Border radius of affix elements

--textbox-affix-disabled-color | Disabled color for affix elements

--textbox-affix-disabled-background | Disabled background for affix elements

--textbox-error-gap | Gap between textarea and error text

--textbox-error-color | Color for error messages

--textbox-error-font-size | Font size for error messages

--textbox-xs-error-font-size | Font size for error text in extra small variant

--textbox-sm-error-font-size | Font size for error text in small variant

--textbox-md-error-font-size | Font size for error text in medium variant

--textbox-lg-error-font-size | Font size for error text in large variant

--textbox-xl-error-font-size | Font size for error text in extra large variant

--textbox-helper-gap | Gap between textarea and helper text

--textbox-helper-color | Color for helper text

--textbox-helper-font-size | Font size for helper text

--textbox-xs-helper-font-size | Helper font size for extra small variant

--textbox-sm-helper-font-size | Helper font size for small variant

--textbox-md-helper-font-size | Helper font size for medium variant

--textbox-lg-helper-font-size | Helper font size for large variant

--textbox-xl-helper-font-size | Helper font size for extra large variant

--textbox-border-primary-color | Primary border color

--processing-gradient-colors | Colors for gradient animation

[/TABLE]

Best Practices

Design Guidelines

- Use appropriate row heights for expected content length

- Provide clear labels and helper text for guidance

- Choose semantic variants that match content purpose

- Enable resizing for longer content input areas

- Consider processing states for form submissions

Accessibility

- Always provide meaningful labels for screen readers

- Use helper text to explain input requirements

- Ensure error messages are descriptive and actionable

- Maintain sufficient color contrast for all variants

- Test keyboard navigation and screen reader compatibility

Performance

- Use OnPush change detection strategy for better performance

- Avoid frequent style changes that trigger repaints

- Consider debouncing for real-time validation

- Use trackBy functions for dynamic textarea lists

CHECKBOX

The

<aava-checkbox>

component provides a highly interactive checkbox element with sophisticated animations, multiple visual variants, and comprehensive state management. It supports standard checked/unchecked states, indeterminate state for partial selections, and full accessibility compliance.

How to use

```
import

{

AavaCheckboxComponent

}

from

"@aava/play-core"

;
```

Basic Usage

Simple checkbox implementation with default settings and label.

```
<aava-checkbox

label="Accept terms and conditions"

[isChecked]="false"

(isCheckedChange)="onCheckboxChange($event)"

>

</aava-checkbox>

<aava-checkbox

label="Subscribe to newsletter"

[isChecked]="false"
```

```
(isCheckedChange)="onCheckboxChange($event)"
>
</aava-checkbox>
<aava-checkbox
label="Enable notifications"
[isChecked]="false"
(isCheckedChange)="onCheckboxChange($event)"
>
</aava-checkbox>
```

Variants

The checkbox component supports 3 distinct animation variants, each with unique visual feedback and timing.

```
<aava-checkbox
label="Default Checked"
variant="default"
[isChecked]="true"
></aava-checkbox>
<aava-checkbox
label="With-bg Checked"
variant="with-bg"
[isChecked]="true"
></aava-checkbox>
<aava-checkbox
label="Animated Checked"
variant="animated"
```

```
[isChecked]="true"
></aava-checkbox>
```

Available Variants

- default- Clean checkbox with smooth draw/erase animations (250ms draw, 300ms erase)
- with-bg- Background fill effect with fast transitions (150ms animations)
- animated- Complex scaling background animation with staggered timing (300ms background + 300ms delay + checkmark)

Orientations

Flexible layout options for different design requirements and form arrangements.

```
<div>
<h1>Vertical Layout (Default)</h1>
<div>
<aava-checkbox
alignment="vertical"
label="Development Tools"
[isChecked]="false"
></aava-checkbox>
<aava-checkbox
alignment="vertical"
label="Design Software"
[isChecked]="true"
></aava-checkbox>
<aava-checkbox
alignment="vertical"
label="Project Management"
```

```
[isChecked]="false"
></aava-checkbox>
<aava-checkbox
alignment="vertical"
label="Communication Apps"
[isChecked]="true"
></aava-checkbox>
</div>
<h1>Horizontal Layout</h1>
<div>
<aava-checkbox
alignment="horizontal"
label="React"
[isChecked]="true"
></aava-checkbox>
<aava-checkbox
alignment="horizontal"
label="Angular"
[isChecked]="false"
></aava-checkbox>
<aava-checkbox
alignment="horizontal"
label="Vue.js"
[isChecked]="true"
></aava-checkbox>
```

```
<aava-checkbox
alignment="horizontal"
label="Svelte"
[isChecked]="false"
></aava-checkbox>
<aava-checkbox
alignment="horizontal"
label="Next.js"
[isChecked]="false"
></aava-checkbox>
<aava-checkbox
alignment="horizontal"
label="Nuxt.js"
[isChecked]="false"
></aava-checkbox>
</div>
</div>
```

Available Orientations

- vertical- Default stacked layout with checkboxes arranged vertically
- horizontal- Inline layout with checkboxes arranged horizontally side by side

Sizes

Three size options to accommodate different interface densities and visual hierarchies.

```
<aava-checkbox label="Small" size="sm" variant="default" [isChecked]="false">
</aava-checkbox>
```

```
<aava-checkbox label="Medium" size="md" variant="default" [isChecked]="true">
</aava-checkbox>
<aava-checkbox label="Large" size="lg" variant="default" [isChecked]="false">
</aava-checkbox>
```

Available Sizes

- sm (Small)- 16x16px checkbox for compact interfaces
- md (Medium)- 20x20px checkbox for standard layouts (default)
- lg (Large)- 24x24px checkbox for prominent placements

States

Comprehensive state management including checked, unchecked, disabled, and indeterminate states.

```
<aava-checkbox label="Unchecked" [isChecked]="false"> </aava-checkbox>
<aava-checkbox label="Checked" [isChecked]="true"> </aava-checkbox>
<aava-checkbox label="Disabled Unchecked" [disable]="true" [isChecked]="false">
</aava-checkbox>
<aava-checkbox label="Disabled Checked" [disable]="true" [isChecked]="true">
</aava-checkbox>
<aava-checkbox
label="Disabled Indeterminate"
[disable]="true"
[indeterminate]="true"
>
</aava-checkbox>
<aava-checkbox label="Indeterminate" [indeterminate]="true"> </aava-checkbox>
<aava-checkbox
```

```
label="Indeterminate (With-bg)"
variant="with-bg"
[indeterminate]="true"
>
</aava-checkbox>
<aava-checkbox
label="Indeterminate (Animated)"
variant="animated"
[indeterminate]="true"
>
</aava-checkbox>
```

Available States

- unchecked- Default empty state
- checked- Selected state with checkmark animation
- disabled- Non-interactive state (can be checked or unchecked)
- indeterminate- Partial selection state with horizontal line indicator

State Behavior

Checked State:

- Displays animated checkmark
- Emitstruevalue on change
- Visual feedback varies by variant

Disabled State:

- Non-interactive (no click/keyboard events)
- Dimmed appearance with disabled styling
- Maintains current checked/unchecked state

- Custom disabled colors via CSS tokens

Indeterminate State:

- Shows horizontal line instead of checkmark
- Used for "select all" scenarios in groups
- Clicking transitions to fully checked state
- Emitstruewhen transitioning from indeterminate

Indeterminate State

Special state for representing partial selections in checkbox groups or tree structures.

```
<div class="checkbox-hierarchy">
<aava-checkbox
label="Select All Tasks"
[isChecked]="parentChecked"
[indeterminate]="indeterminate"
(isCheckedChange)="onParentChanged($event)"
>
</aava-checkbox>
<div class="child-checkboxes">
<aava-checkbox
*ngFor="let child of children; let i = index"
[label]="child.label"
[isChecked]="child.checked"
(isCheckedChange)="onChildChanged(i, $event)"
>
</aava-checkbox>
```

```
</div>
</div>
```

Indeterminate Usage

The indeterminate state is particularly useful for:

• Select All checkboxes- When some but not all items are selected

• Tree structures- Parent nodes with partially selected children

• Group controls- Representing mixed states in checkbox groups

• Bulk actions- Indicating partial selection in data tables

Indeterminate Behavior

• Displays horizontal line indicator instead of checkmark

• Clicking indeterminate checkbox transitions to fully checked

• Cannot be set via user interaction (must be programmatically controlled)

• Supports all variants and sizes

• Maintains proper ARIA attributes (aria-checked="mixed")

Labels & Accessibility

Comprehensive accessibility features including keyboard navigation, ARIA attributes, and label support.

```
<aava-checkbox
label="First checkbox (Tab to focus)"
[(isChecked)]="keyboardStates.first"
(isCheckedChange)="onCheckboxChange($event)"
>
</aava-checkbox>
<aava-checkbox
label="Second checkbox (Space/Enter to toggle)"
```

```
[(isChecked)]="keyboardStates.second"
(isCheckedChange)="onCheckboxChange($event)"
>
</aava-checkbox>
<aava-checkbox
label="Third checkbox (Try keyboard navigation)"
[(isChecked)]="keyboardStates.third"
(isCheckedChange)="onCheckboxChange($event)"
>
</aava-checkbox>
<aava-checkbox
label="Standard checkbox with ARIA"
[(isChecked)]="ariaStates.standard"
(isCheckedChange)="onCheckboxChange($event)"
>
</aava-checkbox>
<aava-checkbox
label="Indeterminate checkbox (aria-checked='mixed')"
[indeterminate]="true"
>
</aava-checkbox>
<aava-checkbox
label="Disabled checkbox (aria-disabled='true')"
[disable]="true"
[isChecked]="true"
```

```
>
</aava-checkbox>
<aava-checkbox
label="Checkbox with visible label"
[(isChecked)]="labelStates.visible"
(isCheckedChange)="onCheckboxChange($event)"
>
</aava-checkbox>
<aava-checkbox
label="Clickable label (clicking label toggles checkbox)"
[(isChecked)]="labelStates.clickable"
(isCheckedChange)="onCheckboxChange($event)"
>
</aava-checkbox>
```

Accessibility Features

Keyboard Navigation:

- Space- Toggle checkbox state
- Enter- Toggle checkbox state
- Tab/Shift+Tab- Navigate to/from checkbox

ARIA Compliance:

- role="checkbox"- Semantic role identification
- aria-checked="true|false|mixed"- Current state indication
- aria-disabled="true|false"- Disabled state indication
- aria-label- Accessibility label (falls back to visible label)

Visual Accessibility:

- High contrast focus indicators
- Proper color contrast ratios via CSS tokens
- Scalable vector icons (no pixelation)
- Motion respects user preferences

Label Behavior

- Optional labels- Checkbox works with or without visible labels
- Clickable labels- Clicking label text toggles checkbox
- Accessible labels- Always provide meaningful labels for screen readers
- Flexible positioning- Label appears to the right of checkbox

Events

Event handling for checkbox state changes and user interactions.

```
<aava-checkbox
label="Event Demo Checkbox"
[isChecked]="false"
(isCheckedChange)="onCheckboxChange($event)"
>
</aava-checkbox>
<aava-checkbox
label="Another Event Checkbox"
[isChecked]="false"
(isCheckedChange)="onCheckboxChange($event)"
>
</aava-checkbox>
```

Available Events

- isCheckedChange- Emitted when checkbox state changes (boolean value)

Event Details

isCheckedChange Event:

- Type:EventEmitter<boolean>

- Emitted when:User clicks checkbox or uses keyboard to toggle

- Value:truewhen checked,falsewhen unchecked

- Timing:Emitted after animation completes (timing varies by variant)

- Indeterminate:Emitstruewhen transitioning from indeterminate to checked

Event Timing by Variant

- Default:Immediate for checking, 300ms delay for unchecking

- With-bg:150ms delay for both checking and unchecking

- Animated:600ms delay for checking, 300ms delay for unchecking

API Reference

Inputs

[TABLE]

Property | Type | Default | Description

variant | 'default' | 'with-bg' | 'animated' | 'default' | Animation style variant

size | 'sm' | 'md' | 'lg' | 'md' | Checkbox size

alignment | 'vertical' | 'horizontal' | 'vertical' | Layout orientation for checkbox groups

label | string | '' | Visible label text

isChecked | boolean | false | Whether checkbox is checked

indeterminate | boolean | false | Whether checkbox is in indeterminate state

disable | boolean | false | Whether checkbox is disabled

customStyles | Record<string, string> | {} | CSS custom properties override error | string | '' | Error message text

id | string | '' | Unique checkbox identifier

[/TABLE]

Outputs

[TABLE]

Event | Type | Description

isCheckedChange | EventEmitter<boolean> | Emitted when checkbox state changes

[/TABLE]

Properties

[TABLE]

Property | Type | Description

showIcon | boolean | Whether to show the icon (checkmark or indeterminate)

showCheckmark | boolean | Whether to show checkmark specifically

[/TABLE]

CSS Custom Properties

[TABLE]

Property | Description

--checkbox-box-background | Background color of the checkbox box

--checkbox-box-checked-border | Border color when checked

--checkbox-box-checked-background | Background color when checked

--checkbox-box-border-disabled | Border color when disabled

--checkbox-box-background-disabled | Background color when disabled

--checkbox-icon-color-disabled | Icon color when disabled

--checkbox-box-checked-color | Checkmark color when checked

--checkbox-label-color | Text color for the label

--checkbox-label-color-disabled | Label text color when disabled

--checkbox-label-cursor | Cursor style for clickable label

--checkbox-label-cursor-disabled | Cursor style when disabled

--checkbox-cursor-disabled | Cursor style for disabled checkbox

--checkbox-size-small | Size dimensions for small variant

--checkbox-size-medium | Size dimensions for medium variant

--checkbox-size-large | Size dimensions for large variant

[/TABLE]

Animation Timing

Each variant has carefully tuned animation timing for optimal user experience:

Default Variant

- Check:250ms cubic-bezier(0.25, 0.46, 0.45, 0.94)

- Uncheck:300ms cubic-bezier(0.55, 0.06, 0.68, 0.19)

With-bg Variant

- Check:150ms cubic-bezier(0.25, 0.46, 0.45, 0.94)

- Uncheck:150ms cubic-bezier(0.55, 0.06, 0.68, 0.19)

Animated Variant

- Background:300ms ease-out

- Checkmark:300ms cubic-bezier(0.25, 0.46, 0.45, 0.94) with 300ms delay

- Uncheck:300ms background + 150ms checkmark (simultaneous)

Best Practices

Design Guidelines

- Choose appropriate variants- Usedefaultfor most cases,with-bgfor high contrast needs,animatedfor engaging interactions

• Size consistently- Match checkbox size to surrounding form elements and text size

• Group related checkboxes- Use proper form structure and fieldsets for related options

• Handle indeterminate properly- Use for partial selections in groups, not individual checkboxes

Accessibility

• Label everything- Always provide meaningful labels for accessibility

• Keyboard navigation- Ensure full keyboard support with proper focus management

• ARIA compliance- Use proper ARIA attributes for screen readers

• Visual accessibility- Maintain high contrast and clear focus indicators

• Motion preferences- Consider users who prefer reduced motion

Performance

• Respect motion preferences- Consider users who prefer reduced motion

• Optimize animations- Use CSS transforms and opacity for smooth performance

• Batch state changes- Avoid rapid successive state changes

• Use appropriate variants- Choose variants based on performance requirements

RADIO-BUTTON

The

<aava-radio-button>

component provides a radio button group for single-choice selections with multiple layout orientations, size options, custom color theming, and smooth interaction effects. It includes full Angular forms integration and accessibility compliance.

How to use

```
import

{

AavaRadioButtonComponent

}

from

"@aava/play-core"

;
```

Basic Usage

Simple radio button group implementation with multiple options and two-way data binding.

```
<aava-radio-button

[options]="[

{ label: 'Option 1', value: 'option1' },

{ label: 'Option 2', value: 'option2' },

{ label: 'Option 3', value: 'option3' },

]"

name="basic-radio"

selectedValue="option1"

(selectedValueChange)="onRadioChange($event)"
```

```
>
</aava-radio-button>
```

Orientations

The radio button component supports both horizontal and vertical layouts to accommodate different design requirements.

```
<aava-radio-button
[options]="[
{ label: 'Option 1', value: 'option1' },
{ label: 'Option 2', value: 'option2' },
{ label: 'Option 3', value: 'option3' },
]"
name="vertical-radio"
orientation="vertical"
selectedValue="option2"
(selectedValueChange)="onRadioChange('vertical', $event)"
>
</aava-radio-button>
<aava-radio-button
[options]="[
{ label: 'Option 1', value: 'option1' },
{ label: 'Option 2', value: 'option2' },
{ label: 'Option 3', value: 'option3' },
]"
name="horizontal-radio"
orientation="horizontal"
```

```
selectedValue="option3"
(selectedValueChange)="onRadioChange('horizontal', $event)"
>
</aava-radio-button>
```

Available Orientations

- vertical- Stacked layout with options arranged vertically (default)

- horizontal- Inline layout with options arranged horizontally

Sizes

Three size options to match different interface densities and visual hierarchies.

```
<aava-radio-button
[options]="[
{ label: 'Option 1', value: 'option1' },
{ label: 'Option 2', value: 'option2' },
{ label: 'Option 3', value: 'option3' }
]"
name="small-radio"
size="sm"
selectedValue="option1"
>
</aava-radio-button>
<aava-radio-button
[options]="[
{ label: 'Option 1', value: 'option1' },
{ label: 'Option 2', value: 'option2' },
{ label: 'Option 3', value: 'option3' }

]"

name="medium-radio"

size="md"

selectedValue="option2"

>

</aava-radio-button>

<aava-radio-button

[options]="[

{ label: 'Option 1', value: 'option1' },

{ label: 'Option 2', value: 'option2' },

{ label: 'Option 3', value: 'option3' }

]"

name="large-radio"

size="lg"

selectedValue="option3"

>

</aava-radio-button>
```

Available Sizes

- sm (Small)- Compact size for dense interfaces

- md (Medium)- Standard size for most use cases (default)

- lg (Large)- Prominent size for important selections

States

Various states for different user scenarios including selection, disabled, and interactive states.

```
<aava-radio-button
[options]="[
{ label: 'Option 1', value: 'option1' },
{ label: 'Option 2', value: 'option2' },
{ label: 'Option 3', value: 'option3' },
]"
name="default-states"
selectedValue=""
>
</aava-radio-button>
<aava-radio-button
[options]="[
{ label: 'Option 1 (Disabled)', value: 'option1', disabled: true },
{ label: 'Option 2', value: 'option2' },
{ label: 'Option 3', value: 'option3' }
]"
name="disabled-states"
selectedValue="option2"
>
</aava-radio-button>
<aava-radio-button
[options]="[
{ label: 'Option 1 (Disabled)', value: 'option1', disabled: true },
{ label: 'Option 2 (Disabled)', value: 'option2', disabled: true },
{ label: 'Option 3 (Disabled)', value: 'option3', disabled: true },
]"
name="all-disabled"
selectedValue="option3"
>
</aava-radio-button>
```

Available States

- Unselected- Default state for non-selected options
- Selected- Active state showing the selected option
- Disabled- Non-interactive state for unavailable options

Custom Colors

Comprehensive color customization for radio buttons and labels to match design requirements.

```
<aava-radio-button
[options]="[
{ label: 'Option 1', value: 'option1' },
{ label: 'Option 2', value: 'option2' },
{ label: 'Option 3', value: 'option3' },
]"
name="full-custom"
color="#8e44ad"
labelColor="#1996FC"
selectedValue="option2"
>
</aava-radio-button>
```

Color Customization Options

- labelColor- Custom color for option labels

- color- Cutom color for the dot

Form Integration

Form integration with ControlValueAccessor implementation for reactive from

```
<div class="demo-section center-demo">
<div class="forms-grid">
<h4>Reactive Form</h4>
<form [formGroup]="reactiveForm" (ngSubmit)="onReactiveSubmit()">
<div class="form-group">
<div class="form-field">
<label>Theme Selection:</label>
<aava-radio-button
formControlName="theme"
[options]="themeOptions"
name="theme-selection"
(selectedValueChange)="onReactiveThemeChange($event)"
></aava-radio-button>
</div>
<div class="form-field">
<label>Size Preference:</label>
<aava-radio-button
formControlName="size"
[options]="sizeOptions"
name="size-selection"
```

```
(selectedValueChange)="onReactiveSizeChange($event)"
></aava-radio-button>
</div>
</div>
<aava-button
label="Submit Form"
variant="primary"
[disabled]="!reactiveForm.valid"
>
</aava-button>
<div class="form-status">
<p><strong>Form Status:</strong> {{ reactiveForm.status }}</p>
</div>
<div *ngIf="showFormvalue">
<span>Form Value</span>
<pre>
{{ reactiveForm.value | json }}
</pre
>
</div>
</form>
</div>
</div>
```

Reactive Form Features

- FormControl binding- Direct integration with Angular reactive forms

• Value tracking- Proper change detection and emission

• Disabled control- Programmatic enable/disable support

Accessibility Support

The Radio Button component provides comprehensive accessibility features to ensure compatibility with screen readers, keyboard navigation, and mobile devices.

ARIA Support

• role="radiogroup"- Container identifies as a radio button group

• role="radio"- Each option is properly identified as a radio button

• aria-checked- Indicates selection state (true/false)

• aria-disabled- Identifies disabled options

• aria-label- Describes the radio group ("Radio button group with X options")

• aria-labelledby- Links radio buttons to their text labels

• aria-describedby- Provides additional context for screen readers

• aria-hidden="true"- Hides decorative elements from assistive technology

Keyboard Navigation

• Arrow Keys (↑↓←→)- Navigate between radio options

• Space/Enter- Select the currently focused option

• Tab- Enter and exit the radio group

• Circular Navigation- Arrow keys wrap around when reaching first/last option

• Skip Disabled- Navigation automatically skips disabled options

• Visual Focus- Clear focus indicators show current keyboard position

WCAG 2.1 Compliance

• Level AAcompliant for color contrast

• Keyboard Accessible- All functionality available via keyboard

- Focus Management- Logical focus order and clear indicators

- Semantic Structure- Proper HTML roles and relationships

- Error Prevention- Clear state indication prevents user confusion

API Reference

Inputs

[TABLE]

Property | Type | Default | Description

options | RadioOption[] | [] | Array of radio button options

name | string | '' | HTML name attribute for the radio group

selectedValue | string | '' | Currently selected option value

size | 'sm' | 'md' | 'lg' | 'md' | Size of radio buttons

orientation | 'horizontal' | 'vertical' | 'vertical' | Layout orientation

color | string | '' | Custom color for radio button and glow

labelColor | string | '' | Custom color for option labels

radio | 'dot' | 'none' | 'dot' | Whether to show the inner dot

animation | 'none' | 'shadow' | 'none' | Animation effect for selection

[/TABLE]

Outputs

[TABLE]

Event | Type | Description

selectedValueChange | EventEmitter<string> | Emitted when selection changes

[/TABLE]

CSS Custom Properties

[TABLE]

Property	Description
--radio-group-gap	Gap between radio button options
--radio-checkmark-background	Background color of the radio button circle
--radio-checkmark-border	Border color of the radio button
--radio-checkmark-border-radius	Border radius for the radio button
--radio-checkmark-background-disabled	Background when disabled
--radio-checkmark-border-disabled	Border color when disabled
--radio-dot-background	Background color of the inner dot
--radio-dot-border-radius	Border radius of the inner dot
--radio-dot-background-disabled	Dot background when disabled
--radio-label-color	Text color for option labels
--radio-label-font	Font properties for labels
--radio-label-margin-left	Left margin for labels
--radio-label-color-disabled	Label text color when disabled
--radio-label-cursor-disabled	Cursor style when disabled
--radio-cursor	Cursor style for interactive elements
--radio-cursor-disabled	Cursor style when disabled
--radio-custom-glow-color	Custom glow color for selected state
--radio-size-sm	Size dimensions for small variant
--radio-size-md	Size dimensions for medium variant
--radio-size-lg	Size dimensions for large variant
--radio-size-sm-dot	Dot size for small variant
--radio-size-md-dot	Dot size for medium variant
--radio-size-lg-dot	Dot size for large variant
--radio-size-sm-label	Label font size for small variant

--radio-size-md-label | Label font size for medium variant

--radio-size-lg-label | Label font size for large variant

[/TABLE]

Best Practices

Design Guidelines

- Provide clear options- Use descriptive labels that clearly distinguish choices
- Limit option count- Keep radio groups to 7 or fewer options for usability
- Choose appropriate orientation- Use vertical for longer lists, horizontal for 2-4 items
- Include default selection- Pre-select the most common or safe option when appropriate
- Group related options- Only use radio buttons for mutually exclusive choices
- Consider disabled states- Disable options that aren't currently available

Performance

- Optimize option rendering- Use trackBy functions for dynamic option lists
- Debounce rapid changes- Consider debouncing for real-time validation
- Minimize re-renders- Use OnPush change detection strategy
- Efficient event handling- Use event delegation for large option lists

Form Integration

- Test with forms- Ensure proper integration with your form validation
- Handle validation- Provide clear feedback for required selections
- Consider reset behavior- Define what happens when forms are reset
- Support reactive forms- Use proper ControlValueAccessor implementation