# 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.

<!-- Matrix Table -->

<!-- Size Headers -->

<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

</ng-container>

## 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)>

$

USD

</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"

>

$

</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