Button

https://playdocs1.orangeriver-ad055946.westus2.azurecontainerapps.io/play-docs/docs/ui-components/Inputs/Button

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.

Thecomponent 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.

```
<aava-button>
```

How to use

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

```
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>
```

```
<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>
```

```
type ButtonMode = 'pill' | 'default' | 'action' | 'quick-action';type ButtonFill = 'filled' | 'outline' | 'clear';type ButtonSize = 'xs' | 'sm' | 'md' | 'lg' | 'xl';interface ButtonConfig {  mode: ButtonMode;  fill: ButtonFill;  size: ButtonSize;  pill: boolean;  outlined: boolean;  clear: boolean;  iconPosition?: 'left' | 'right' | 'only';  iconName: string;  label: string;  available: boolean;  iconColor?: string;}modes: ButtonMode[] = ['pill', 'default', 'action', 'quick-action'];  fills: ButtonFill[] = ['filled', 'outline', 'clear'];  sizes: ButtonSize[] = ['xsmall', 'small', 'medium', 'large', 'xlarge'];  sizeTabs: TabItem[] = [    { id: 'xsmall', label: 'XSmall' },    { id: 'small', label: 'Small' },    { id: 'medium', label: 'Medium' },    { id: 'large', label: 'Large' },    { id: 'xlarge', label: 'XLarge' },  ];  variants: ButtonVariant[] = [    'primary', 'secondary',    'success',    'warning',    'danger',    'info',  ];  selectedSize: ButtonSize = 'md'; selectedMode: ButtonMode = 'pill';  selectedFill: ButtonFill = 'filled';  toggleTheme(): void {
```

```
this.isDarkTheme = !this.isDarkTheme;    document.body.setAttribute(      'data-theme',      this.isDarkTheme ?
'dark' : 'light'    );  }  onSizeTabChange(tab: TabItem): void {    this.selectedSize = tab.id as ButtonSize;  }
getSizeLabel(size: ButtonSize): string {    switch (size) {      case 'xs':        return 'XSmall';      case
'sm':        return 'Small';      case 'md':        return 'Medium';      case 'lg':        return 'Large';
case 'xl':        return 'XLarge';      default:        return size;    }  }  getSizeMapping(    size: ButtonSize
): 'xs' | 'sm' | 'md' | 'lg' | 'xl' {    switch (size) {      case 'xs':        return 'xs';      case 'sm':
return 'sm';      case 'md':        return 'md';      case 'lg':        return 'lg';      case 'xl':        return
'xl';      default:        return 'md';    }  }  getButtonConfig(    mode: ButtonMode,    fill: ButtonFill,
size: ButtonSize  ): ButtonConfig {    // All fills are now available    const baseConfig = {      mode,
fill,      size,      outlined: fill === 'outline',      clear: fill === 'clear',      available: true,
iconColor: '#fff',    };    switch (mode) {      case 'pill':        return {          ...baseConfig,
pill: true,          iconPosition: 'left' as const,          iconName: 'star',          label: 'Pill',        };
case 'default':        return {          ...baseConfig,          pill: false,          iconName: '',
iconPosition: 'left' as const,          label: 'Default',        };      case 'action':        return {
...baseConfig,          pill: false,          iconPosition: 'left' as const,          iconName: 'zap',
label: 'Action',        };      case 'quick-action':        return {          ...baseConfig,          pill: true,
iconPosition: 'only' as const,          iconName: 'plus',          label: '',        };      default:
return {          ...baseConfig,          pill: false,          iconPosition: undefined,          iconName: '',
label: mode,          available: false,        };    }  }
```

```
type ButtonMode = 'pill' | 'default' | 'action' | 'quick-action';type ButtonFill = 'filled' | 'outline' |
'clear';type ButtonSize = 'xs' | 'sm' | 'md' | 'lg' | 'xl';interface ButtonConfig {  mode: ButtonMode;  fill:
ButtonFill;  size: ButtonSize;  pill: boolean;  outlined: boolean;  clear: boolean;  iconPosition?: 'left' |
'right' | 'only';  iconName: string;  label: string;  available: boolean;  iconColor?: string;}modes: ButtonMode[]
= ['pill', 'default', 'action', 'quick-action'];  fills: ButtonFill[] = ['filled', 'outline', 'clear'];  sizes:
ButtonSize[] = ['xsmall', 'small', 'medium', 'large', 'xlarge'];  sizeTabs: TabItem[] = [    { id: 'xsmall',
label: 'XSmall' },    { id: 'small', label: 'Small' },    { id: 'medium', label: 'Medium' },    { id: 'large',
label: 'Large' },    { id: 'xlarge', label: 'XLarge' },  ];  variants: ButtonVariant[] = [    'primary',
'secondary',    'success',    'warning',    'danger',    'info',  ];  selectedSize: ButtonSize = 'md';
selectedMode: ButtonMode = 'pill';  selectedFill: ButtonFill = 'filled';  toggleTheme(): void {
this.isDarkTheme = !this.isDarkTheme;    document.body.setAttribute(      'data-theme',      this.isDarkTheme ?
'dark' : 'light'    );  }  onSizeTabChange(tab: TabItem): void {    this.selectedSize = tab.id as ButtonSize;  }
getSizeLabel(size: ButtonSize): string {    switch (size) {      case 'xs':        return 'XSmall';      case
'sm':        return 'Small';      case 'md':        return 'Medium';      case 'lg':        return 'Large';
case 'xl':        return 'XLarge';      default:        return size;    }  }  getSizeMapping(    size: ButtonSize
): 'xs' | 'sm' | 'md' | 'lg' | 'xl' {    switch (size) {      case 'xs':        return 'xs';      case 'sm':
return 'sm';      case 'md':        return 'md';      case 'lg':        return 'lg';      case 'xl':        return
'xl';      default:        return 'md';    }  }  getButtonConfig(    mode: ButtonMode,    fill: ButtonFill,
size: ButtonSize  ): ButtonConfig {    // All fills are now available    const baseConfig = {      mode,
fill,      size,      outlined: fill === 'outline',      clear: fill === 'clear',      available: true,
iconColor: '#fff',    };    switch (mode) {      case 'pill':        return {          ...baseConfig,
pill: true,          iconPosition: 'left' as const,          iconName: 'star',          label: 'Pill',        };
case 'default':        return {          ...baseConfig,          pill: false,          iconName: '',
iconPosition: 'left' as const,          label: 'Default',        };      case 'action':        return {
...baseConfig,          pill: false,          iconPosition: 'left' as const,          iconName: 'zap',
label: 'Action',        };      case 'quick-action':        return {          ...baseConfig,          pill: true,
iconPosition: 'only' as const,          iconName: 'plus',          label: '',        };      default:
return {          ...baseConfig,          pill: false,          iconPosition: undefined,          iconName: '',
label: mode,          available: false,        };    }  }
```

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>
```

```
<aava-button  label="Primary"  variant="primary"  (userClick)="onButtonClick($event)"  [pill]="true"></aava-
button><aava-button  label="Primary"  variant="primary"  (userClick)="onButtonClick($event)"></aava-button>
```

```
  onButtonClick(event: Event) {    console.log('Button clicked:', event);  }
```

```
  onButtonClick(event: Event) {    console.log('Button clicked:', event);  }
```

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>
```

```
<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>
```

```
<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>
```

```
<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>
```

```
<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>
```

```
<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)

```
left
```

```
right
```

```
icon-only
```

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>
```

```
<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>
```

```
<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>
```

```
<aava-button  label="Click Handler"  variant="primary"  (userClick)="onButtonClick()"  [customStyles]="{ 'max-width': '200px' }"  [pill]="true"></aava-button>
```

```
  onButtonClick() {    setTimeout(() => {      alert('Hi There!');    }, 200);  }
```

```
  onButtonClick() {    setTimeout(() => {      alert('Hi There!');    }, 200);  }
```

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

```
aria-disabled
```

```
aria-pressed
```

API Reference

Inputs

```
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 | {} | 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
```

```
label
```

```
string
```

```
''
```

```
variant
```

```
ButtonVariant
```

```
'default'
```

```
'default'
```

```
'primary'
```

```
'secondary'
```

```
'success'
```

```
'warning'
```

```
'danger'
```

```
'info'
```

```
size
```

```
ButtonSize
```

```
'md'
```

```
'xs'
```

```
'sm'
```

```
'md'
```

```
'lg'
```

```
'xl'
```

```
state
```

```
ButtonState
```

```
'default'
```

```
'default'
```

```
'hover'
```

```
'active'
```

```
'disabled'
```

```
'processing'
```

```
'focus'
```

```
hoverEffect
```

```
ButtonHoverEffect
```

```
'torch'
```

```
'torch'
```

```
'glow'
```

```
'tint'
```

```
'scale'
```

```
'none'
```

```
pressedEffect
```

```
ButtonPressedEffect
```

```
'ripple'
```

```
'ripple'
```

```
'inset'
```

```
'solid'
```

```
'none'
```

```
processingEffect
```

```
ButtonProcessingEffect
```

```
'pulse'
```

```
'pulse'
```

```
'none'
```

```
focusEffect
```

```
ButtonFocusEffect
```

```
'border'
```

```
'border'
```

```
'none'
```

```
disabledEffect
```

```
ButtonDisabledEffect
```

```
'dim'
```

```
'dim'
```

```
'none'
```

```
disabled
```

```
boolean
```

```
false
```

```
processing
```

```
boolean
```

```
false
```

```
pill
```

```
boolean
```

```
false
```

```
outlined
```

```
boolean
```

```
false
```

```
clear
```

```
boolean
```

```
false
```

```
width
```

```
string
```

```
''
```

```
height
```

```
string
```

```
''
```

```
gradient
```

```
string
```

```
undefined
```

```
customStyles
```

```
background
```

```
string
```

```
undefined
```

```
customStyles
```

```
color
```

```
string
```

```
undefined
```

```
customStyles
```

```
dropdown
```

```
boolean
```

```
false
```

```
glassVariant
```

```
ButtonGlassVariant
```

```
'glass-10'
```

```
customStyles
```

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

```
{}
```

```
iconName
```

```
string
```

```
''
```

```
iconColor
```

```
string
```

```
''
```

```
iconSize
```

```
number
```

```
20
```

```
iconPosition
```

```
'left' | 'right' | 'only'
```

```
'left'
```

Outputs

```
Event | Type | Description
userClick | EventEmitter | Emitted when button is clicked or activated via keyboard
```

```
userClick
```

```
EventEmitter<Event>
```

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 ofscoped override tokensthat allow you to fine-tune appearance without breaking consistency.

Use theseonly 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

```
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
```

```
--button-size-xsm-padding
```

```
--button-size-sm-padding
```

```
--button-size-md-padding
```

```
--button-size-lg-padding
```

```
--button-size-xlg-padding
```

```
--button-size-xsm-height
```

```
--button-size-sm-height
```

```
--button-size-md-height
```

```
--button-size-lg-height
```

```
--button-size-xlg-height
```

```
--button-border-radius
```

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

```
--button-font-weight
```

```
--button-transition
```

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

```
--button-icon-margin
```

Token Override Example

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

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

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

```
default
```

```
primary
```

```
medium
```

```
extra small
```

```
extra large
```

```
icon-only
```

```
left
```

```
right
```

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