Skeleton-Loader

https://playdocs1.orangeriver-ad055946.westus2.azurecontainerapps.io/play-docs/docs/ui-components/Feedback/Skeleton-Loader

Skeleton Loader

Thecomponent is a versatile loading placeholder that provides smooth animations and multiple shape options to create engaging loading states. It helps improve perceived performance by showing users that content is loading, reducing perceived wait times and providing visual feedback during data fetching operations.

```
<aava-skeleton>
```

How to use

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


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


Basic Usage

A simple skeleton with default rectangle shape and wave animation.

```
<!-- Default skeleton for text content --><aava-skeleton  width="100%"  height="20px"  shape="rectangle"
animation="wave"></aava-skeleton><!-- Shorter text line --><aava-skeleton  width="80%"  height="16px"
shape="rectangle"  animation="wave"></aava-skeleton><!-- Even shorter text line --><aava-skeleton  width="60%"
height="16px"  shape="rectangle"  animation="wave"></aava-skeleton>
```


```
<!-- Default skeleton for text content --><aava-skeleton  width="100%"  height="20px"  shape="rectangle"
animation="wave"></aava-skeleton><!-- Shorter text line --><aava-skeleton  width="80%"  height="16px"
shape="rectangle"  animation="wave"></aava-skeleton><!-- Even shorter text line --><aava-skeleton  width="60%"
height="16px"  shape="rectangle"  animation="wave"></aava-skeleton>
```


Shapes

The skeleton component supports multiple shapes for different content types.

```
<!-- Rectangle --><aava-skeleton  width="200px"  height="20px"  shape="rectangle"  animation="wave"></aava-
skeleton><!-- Circle --><aava-skeleton  width="80px"  height="80px"  shape="circle"  animation="pulse"></aava-
skeleton><!-- Rounded --><aava-skeleton  width="150px"  height="100px"  shape="rounded"  animation="wave"></aava-
skeleton><!-- Square --><aava-skeleton  width="100px"  height="100px"  shape="square"  animation="wave"></aava-
skeleton>
```


```
<!-- Rectangle --><aava-skeleton  width="200px"  height="20px"  shape="rectangle"  animation="wave"></aava-
skeleton><!-- Circle --><aava-skeleton  width="80px"  height="80px"  shape="circle"  animation="pulse"></aava-
skeleton><!-- Rounded --><aava-skeleton  width="150px"  height="100px"  shape="rounded"  animation="wave"></aava-
skeleton><!-- Square --><aava-skeleton  width="100px"  height="100px"  shape="square"  animation="wave"></aava-
skeleton>
```


Custom Styling

Customize the skeleton with different colors, sizes, and background colors.

```
<!-- Light blue background --><aava-skeleton  width="200px"  height="20px"  shape="rectangle"  animation="wave"
backgroundColor="#e3f2fd"></aava-skeleton><!-- Light purple background --><aava-skeleton  width="200px"
height="20px"  shape="rectangle"  animation="pulse"  backgroundColor="#f3e5f5"></aava-skeleton><!-- Light green
circle --><aava-skeleton  width="80px"  height="80px"  shape="circle"  animation="wave"
backgroundColor="#e8f5e8"></aava-skeleton><!-- Light orange rounded --><aava-skeleton  width="150px"
height="100px"  shape="rounded"  animation="pulse"  backgroundColor="#fff3e0"></aava-skeleton>
```


```
<!-- Light blue background --><aava-skeleton  width="200px"  height="20px"  shape="rectangle"  animation="wave"
backgroundColor="#e3f2fd"></aava-skeleton><!-- Light purple background --><aava-skeleton  width="200px"
height="20px"  shape="rectangle"  animation="pulse"  backgroundColor="#f3e5f5"></aava-skeleton><!-- Light green
circle --><aava-skeleton  width="80px"  height="80px"  shape="circle"  animation="wave"
```

```
backgroundColor="#e8f5e8"></aava-skeleton><!-- Light orange rounded --><aava-skeleton  width="150px"
height="100px"  shape="rounded"  animation="pulse"  backgroundColor="#fff3e0"></aava-skeleton>
```

Card Skeleton

Create a complete card skeleton with multiple elements.

```
<div  *ngFor="let element of cardSkeletonElements; let i = index"  class="skeleton-element"  [class.title]="i ===
1"  [class.description]="i > 1">  <aava-skeleton    [width]="element.width"    [height]="element.height"
[shape]="element.shape"    [animation]="element.animation"  ></aava-skeleton></div>
```


```
<div  *ngFor="let element of cardSkeletonElements; let i = index"  class="skeleton-element"  [class.title]="i ===
1"  [class.description]="i > 1">  <aava-skeleton    [width]="element.width"    [height]="element.height"
[shape]="element.shape"    [animation]="element.animation"  ></aava-skeleton></div>
```


```
cardSkeletonElements = [  {    width: "100%",    height: "200px",    shape: "rounded" as const,    animation:
"wave" as const,    description: "Card Image",  },  {    width: "70%",    height: "20px",    shape: "rectangle" as
const,    animation: "wave" as const,    description: "Card Title",  },  {    width: "100%",    height: "16px",
shape: "rectangle" as const,    animation: "wave" as const,    description: "Card Description Line 1",  },  {
width: "100%",    height: "16px",    shape: "rectangle" as const,    animation: "wave" as const,    description:
"Card Description Line 2",  },  {    width: "40%",    height: "16px",    shape: "rectangle" as const,
animation: "wave" as const,    description: "Card Description Line 3",  },];
```


```
cardSkeletonElements = [  {    width: "100%",    height: "200px",    shape: "rounded" as const,    animation:
"wave" as const,    description: "Card Image",  },  {    width: "70%",    height: "20px",    shape: "rectangle" as
const,    animation: "wave" as const,    description: "Card Title",  },  {    width: "100%",    height: "16px",
shape: "rectangle" as const,    animation: "wave" as const,    description: "Card Description Line 1",  },  {
width: "100%",    height: "16px",    shape: "rectangle" as const,    animation: "wave" as const,    description:
"Card Description Line 2",  },  {    width: "40%",    height: "16px",    shape: "rectangle" as const,
animation: "wave" as const,    description: "Card Description Line 3",  },];
```


List Skeleton

Build list skeletons for data tables and content lists.

```
<div class="table-skeleton">  <div class="table-header">    <div *ngFor="let element of listItemElements"
class="header-cell">      <aava-skeleton        [width]="element.width"        [height]="element.height"
[shape]="element.shape"        [animation]="element.animation"      ></aava-skeleton>    </div>  </div>  <div
*ngFor="let item of listItems" class="table-row">    <div *ngFor="let element of listItemElements" class="table-
cell">      <aava-skeleton        [width]="element.width"        [height]="element.height"
[shape]="element.shape"        [animation]="element.animation"      ></aava-skeleton>    </div>  </div></div>
```


```
<div class="table-skeleton">  <div class="table-header">    <div *ngFor="let element of listItemElements"
class="header-cell">      <aava-skeleton        [width]="element.width"        [height]="element.height"
[shape]="element.shape"        [animation]="element.animation"      ></aava-skeleton>    </div>  </div>  <div
*ngFor="let item of listItems" class="table-row">    <div *ngFor="let element of listItemElements" class="table-
cell">      <aava-skeleton        [width]="element.width"        [height]="element.height"
[shape]="element.shape"        [animation]="element.animation"      ></aava-skeleton>    </div>  </div></div>
```


```
listItems = [1, 2, 3, 4, 5];listItemElements = [  {    width: "60px",    height: "20px",    shape: "rectangle" as
const,    animation: "wave" as const,    description: "ID",  },  {    width: "150px",    height: "20px",    shape:
"rectangle" as const,    animation: "wave" as const,    description: "Name",  },  {    width: "100px",    height:
"20px",    shape: "rectangle" as const,    animation: "wave" as const,    description: "Status",  },  {    width:
"80px",    height: "20px",    shape: "rectangle" as const,    animation: "wave" as const,    description:
"Actions",  },];
```


```
listItems = [1, 2, 3, 4, 5];listItemElements = [  {    width: "60px",    height: "20px",    shape: "rectangle" as
const,    animation: "wave" as const,    description: "ID",  },  {    width: "150px",    height: "20px",    shape:
"rectangle" as const,    animation: "wave" as const,    description: "Name",  },  {    width: "100px",    height:
"20px",    shape: "rectangle" as const,    animation: "wave" as const,    description: "Status",  },  {    width:
"80px",    height: "20px",    shape: "rectangle" as const,    animation: "wave" as const,    description:
"Actions",  },];
```


Features

Multiple Shapes

• Rectangle: Default shape for text lines and content blocks

• Circle: Perfect for avatars and profile pictures

• Rounded: Soft corners for modern UI elements

• Square: Sharp corners for structured content

Animation Types

• Wave: Smooth shimmer effect that moves across the skeleton

• Pulse: Gentle fade in/out effect for subtle loading states

Customization

• Flexible Sizing: Custom width and height for any content type

• Color Control: Customizable background and animation colors

• Responsive Design: Adapts to different screen sizes

• Performance Optimized: Lightweight animations with minimal impact

Accessibility

• Screen Reader Support: Proper ARIA attributes for loading states

• Reduced Motion: Respects user's motion preferences

• High Contrast: Works with high contrast mode settings

• Focus Management: Proper focus handling during loading

API Reference

Inputs

```
Property | Type | Default | Description
width | string | '100%' | Width of the skeleton element
height | string | '20px' | Height of the skeleton element
shape | ShimmerShape | 'rectangle' | Shape of the skeleton element
animation | ShimmerAnimation | 'wave' | Animation type for the skeleton
backgroundColor | string | '#e0e0e0' | Background color of the skeleton
skeletonType | 'tableList' | 'table' | 'tableList' | Type of skeleton layout
rows | number | 5 | Number of rows in the skeleton layout
columns | number | 5 | Number of columns in the skeleton layout
isLoading | boolean | true | Whether to show the skeleton
```

```
width
```

```
string
```

```
'100%'
```

```
height
```

```
string
```

```
'20px'
```

```
shape
```

```
ShimmerShape
```

```
'rectangle'
```

```
animation
```

```
ShimmerAnimation
```

```
'wave'
```

```
backgroundColor
```

```
string
```

```
'#e0e0e0'
```

```
skeletonType
```

```
'tableList' | 'table'
```

```
'tableList'
```

```
rows
```

```
number
```

```
5
```

```
columns
```

```
number
```

```
5
```

```
isLoading
```

```
boolean
```

```
true
```

Types

```
type ShimmerShape = "rectangle" | "circle" | "rounded" | "square";
```

```
type ShimmerShape = "rectangle" | "circle" | "rounded" | "square";
```

```
type ShimmerAnimation = "wave" | "pulse";
```

```
type ShimmerAnimation = "wave" | "pulse";
```

CSS Classes

The component provides several CSS classes for styling:

```
Class Name | Description
.shimmer-container | Main skeleton container
.shimmer-item | Base skeleton element
.shimmer-rectangle | Rectangle shape styling
```

```
.shimmer-circle | Circle shape styling
.shimmer-rounded | Rounded shape styling
.shimmer-square | Square shape styling
.shimmer-animation-wave | Wave animation styling
.shimmer-animation-pulse | Pulse animation styling
```

```
.shimmer-container
```

```
.shimmer-item
```

```
.shimmer-rectangle
```

```
.shimmer-circle
```

```
.shimmer-rounded
```

```
.shimmer-square
```

```
.shimmer-animation-wave
```

```
.shimmer-animation-pulse
```

CSS Custom Properties

The component uses CSS custom properties for theming:

```
Property | Description
--skeleton-border-radius | Border radius for rectangle shape
--skeleton-rounded-radius | Border radius for rounded shape
--skeleton-wave-duration | Duration of wave animation
--skeleton-pulse-duration | Duration of pulse animation
--skeleton-wave-opacity | Opacity for wave animation
--skeleton-pulse-opacity | Opacity for pulse animation
--skeleton-background-color | Background color of skeleton
--skeleton-gradient-color-start | Start color for gradient animation
--skeleton-gradient-color-end | End color for gradient animation
--skeleton-animation-timing | Timing function for skeleton animations
--skeleton-animation-easing | Easing function for skeleton transitions
```

```
--skeleton-border-radius
```

```
--skeleton-rounded-radius
```

```
--skeleton-wave-duration
```

```
--skeleton-pulse-duration
```

```
--skeleton-wave-opacity
```

```
--skeleton-pulse-opacity
```

```
--skeleton-background-color
```

```
--skeleton-gradient-color-start
```

```
--skeleton-gradient-color-end
```

```
--skeleton-animation-timing
```

```
--skeleton-animation-easing
```

Best Practices

Content Matching

• Match Content Size: Make skeleton dimensions match the actual content

• Use Appropriate Shapes: Choose shapes that represent the actual content

• Maintain Layout: Keep skeleton layout consistent with loaded content

• Consider Spacing: Include proper spacing between skeleton elements

Performance

• Limit Skeleton Count: Don't show too many skeletons at once

• Use Appropriate Duration: Keep animations smooth but not distracting

• Consider Motion Preferences: Respect user's motion preferences

• Optimize for Mobile: Ensure good performance on mobile devices

User Experience

• Show Loading State: Always indicate when content is loading

• Provide Context: Use skeletons that give users an idea of what's coming

• Smooth Transitions: Ensure smooth transition from skeleton to content

• Consistent Timing: Keep skeleton duration consistent across the app

Accessibility

• Screen Reader Support: Provide proper loading announcements

• Reduced Motion: Support users who prefer reduced motion

• High Contrast: Ensure visibility in high contrast mode

• Focus Management: Handle focus properly during loading states

Accessibility Guidelines

Screen Reader Support

• Loading Announcements: Provide clear loading state announcements

• Content Description: Describe what content is loading

• Progress Indication: Indicate loading progress when possible

• State Changes: Announce when content finishes loading

Motion and Animation

• Reduced Motion: Respectprefers-reduced-motionmedia query

• Animation Duration: Keep animations smooth but not distracting

• Motion Alternatives: Provide alternatives for users who can't see animations

• Performance: Ensure animations don't cause performance issues

```
prefers-reduced-motion
```

Visual Design

• High Contrast: Ensure visibility in high contrast mode

• Color Independence: Don't rely solely on color for information

• Focus Indicators: Provide clear focus indicators

• Consistent Styling: Maintain consistent skeleton styling

Keyboard Navigation

• Focus Management: Handle focus properly during loading

• Tab Order: Maintain logical tab order

• Skip Links: Provide skip links for keyboard users

• Loading States: Indicate loading state to keyboard users

Responsive Behavior

Mobile Adaptations

The skeleton component automatically adapts to mobile screens:

• Touch Optimization: Optimized for touch interactions

• Viewport Adaptation: Adapts to different mobile viewport sizes

• Performance: Optimized performance for mobile devices

• Battery Consideration: Efficient animations for battery life

Breakpoint Behavior

• Desktop (>768px): Full skeleton with all features

• Mobile (≤768px): Optimized skeleton for mobile screens

• Content Scaling: Skeleton scales appropriately with content

• Animation Performance: Optimized animations for different devices

Content Considerations

• Flexible Sizing: Skeleton adapts to different content sizes

• Layout Preservation: Maintains layout consistency across devices

• Loading States: Consistent loading experience across platforms

• Performance: Efficient rendering on all device types