

# Kendo UI for Angular Button Overview

The Kendo UI for Angular Button is an Angular component which performs any action attached to it and triggers a corresponding event when users click it.

To communicate its purpose with the users, the Button shows a piece of information by displaying text, icon and text, or icon only. The Button provides configuration properties and predefined sets of styling options for enhancing its appearance.

The following example demonstrates the Button in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { FormsModule } from "@angular/forms";
import { SVGIcon, folderIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, FormsModule, KENDO_BUTTONS],
  styles: [
    `
      .k-button {
        margin: 5px;
      }
    `,
  ],
  template: `
    <div class="row example-wrapper">
      <div class="col-xs-12 col-sm-6 example-col">

```

```
<p>Default Buttons</p>
<p>
    <button kendoButton (click)="onButtonClick()">Browse</button>
    <button kendoButton (click)="onButtonClick()">
        [svgIcon]="folderSVG">
            Browse
        </button>
        <button
            kendoButton
            (click)="onButtonClick()"
            [svgIcon]="folderSVG"
            title="Browse"
        ></button>
    </p>
    <p>
        <button kendoButton (click)="onButtonClick()">
            [disabled]="true">
                Browse
            </button>
            <button
                kendoButton
                (click)="onButtonClick()"
                [svgIcon]="folderSVG"
                [disabled]="true"
            >
                Browse
            </button>
            <button
                kendoButton
                (click)="onButtonClick()"
                [svgIcon]="folderSVG"
                title="Browse"
                [disabled]="true"
            ></button>
        </p>
    </div>

<div class="col-xs-12 col-sm-6 example-col">
    <p>Default Buttons (Primary)</p>
    <p>
        <button kendoButton (click)="onButtonClick()">
            themeColor="primary">
                Browse
            </button>
            <button
                kendoButton
                (click)="onButtonClick()"
                [svgIcon]="folderSVG"
                themeColor="primary"
            >
                Browse
            </button>
        </p>
    </div>
```

```
</button>
<button
  kendoButton
  (click)="onButtonClick()"
  [svgIcon]="folderSVG"
  title="Browse"
  themeColor="primary"
></button>
</p>
<p>
  <button
    kendoButton
    (click)="onButtonClick()"
    [disabled]="true"
    themeColor="primary"
  >
    Browse
  </button>
  <button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="folderSVG"
    [disabled]="true"
    themeColor="primary"
  >
    Browse
  </button>
  <button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="folderSVG"
    title="Browse"
    [disabled]="true"
    themeColor="primary"
  ></button>
</p>
</div>

<div class="col-xs-12 col-sm-6 example-col">
  <p>Flat Buttons</p>
  <p>
    <button kendoButton (click)="onButtonClick()" fillMode="flat">
      Browse
    </button>
    <button
      kendoButton
      (click)="onButtonClick()"
      [svgIcon]="folderSVG"
      fillMode="flat"
    >
      Browse
    </button>
  </p>
</div>
```

```
</button>
<button
  kendoButton
  (click)="onButtonClick()"
  [svgIcon]="'folderSVG'"
  fillMode="flat"
  title="Browse"
></button>
</p>
<p>
  <button
    kendoButton
    (click)="onButtonClick()"
    fillMode="flat"
    [disabled]="true"
  >
    Browse
  </button>
  <button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="'folderSVG'"
    fillMode="flat"
    [disabled]="true"
  >
    Browse
  </button>
  <button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="'folderSVG'"
    fillMode="flat"
    title="Browse"
    [disabled]="true"
  ></button>
</p>
</div>

<div class="col-xs-12 col-sm-6 example-col">
  <p>Flat Buttons (Primary)</p>
  <p>
    <button
      kendoButton
      (click)="onButtonClick()"
      fillMode="flat"
      themeColor="primary"
    >
      Browse
    </button>
    <button
      kendoButton
```

```
(click)="onButtonClick()"
[svgIcon]="folderSVG"
fillMode="flat"
themeColor="primary"
>
  Browse
</button>
<button
  kendoButton
  (click)="onButtonClick()"
  [svgIcon]="folderSVG"
  fillMode="flat"
  title="Browse"
  themeColor="primary"
></button>
</p>
<p>
  <button
    kendoButton
    (click)="onButtonClick()"
    fillMode="flat"
    [disabled]="true"
    themeColor="primary"
>
  Browse
</button>
<button
  kendoButton
  (click)="onButtonClick()"
  [svgIcon]="folderSVG"
  fillMode="flat"
  [disabled]="true"
  themeColor="primary"
>
  Browse
</button>
<button
  kendoButton
  (click)="onButtonClick()"
  [svgIcon]="folderSVG"
  fillMode="flat"
  title="Browse"
  [disabled]="true"
  themeColor="primary"
></button>
</p>
</div>

<div class="col-xs-12 col-sm-6 example-col">
  <p>Outline Buttons</p>
  <p>
```

```
<button kendoButton (click)="onButtonClick()" fillMode="outline">
    Browse
</button>
<button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="folderSVG"
    fillMode="outline"
>
    Browse
</button>
<button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="folderSVG"
    fillMode="outline"
    title="Browse"
></button>
</p>
<p>
    <button
        kendoButton
        (click)="onButtonClick()"
        fillMode="outline"
        [disabled]="true"
>
    Browse
</button>
<button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="folderSVG"
    fillMode="outline"
    [disabled]="true"
>
    Browse
</button>
<button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="folderSVG"
    fillMode="outline"
    title="Browse"
    [disabled]="true"
></button>
</p>
</div>

<div class="col-xs-12 col-sm-6 example-col">
    <p>Outline Buttons (Primary)</p>
```

```
<p>
  <button
    kendoButton
    (click)="onButtonClick()"
    fillMode="outline"
    themeColor="primary"
  >
    Browse
  </button>
  <button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="folderSVG"
    fillMode="outline"
    themeColor="primary"
  >
    Browse
  </button>
  <button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="folderSVG"
    fillMode="outline"
    title="Browse"
    themeColor="primary"
  ></button>
</p>
<p>
  <button
    kendoButton
    (click)="onButtonClick()"
    fillMode="outline"
    [disabled]="true"
    themeColor="primary"
  >
    Browse
  </button>
  <button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="folderSVG"
    fillMode="outline"
    [disabled]="true"
    themeColor="primary"
  >
    Browse
  </button>
  <button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="folderSVG"
  >
```

```
        fillMode="outline"
        title="Browse"
        [disabled]="true"
        themeColor="primary"
      ></button>
    </p>
</div>

<div class="col-xs-12 col-sm-6 example-col">
  <p>Link Buttons</p>
  <p>
    <button kendoButton (click)="onButtonClick()" fillMode="link">
      Browse
    </button>
    <button
      kendoButton
      (click)="onButtonClick()"
      [svgIcon]="'folderSVG'"
      fillMode="link"
    >
      Browse
    </button>
    <button
      kendoButton
      (click)="onButtonClick()"
      [svgIcon]="'folderSVG'"
      fillMode="link"
      title="Browse"
    ></button>
  </p>
  <p>
    <button
      kendoButton
      (click)="onButtonClick()"
      fillMode="link"
      [disabled]="true"
    >
      Browse
    </button>
    <button
      kendoButton
      (click)="onButtonClick()"
      [svgIcon]="'folderSVG'"
      fillMode="link"
      [disabled]="true"
    >
      Browse
    </button>
    <button
      kendoButton
      (click)="onButtonClick()"
```

```
[svgIcon]="folderSVG"
fillMode="link"
title="Browse"
[disabled]="true"
></button>
</p>
</div>

<div class="col-xs-12 col-sm-6 example-col">
<p>Link Buttons (Primary)</p>
<p>
<button
  kendoButton
  (click)="onButtonClick()"
  fillMode="link"
  themeColor="primary"
>
  Browse
</button>
<button
  kendoButton
  (click)="onButtonClick()"
  [svgIcon]="folderSVG"
  fillMode="link"
  themeColor="primary"
>
  Browse
</button>
<button
  kendoButton
  (click)="onButtonClick()"
  [svgIcon]="folderSVG"
  fillMode="link"
  title="Browse"
  themeColor="primary"
></button>
</p>
<p>
<button
  kendoButton
  (click)="onButtonClick()"
  fillMode="link"
  [disabled]="true"
  themeColor="primary"
>
  Browse
</button>
<button
  kendoButton
  (click)="onButtonClick()"
  [svgIcon]="folderSVG"
```

```
        fillMode="link"
        [disabled]="true"
        themeColor="primary"
    >
    Browse
</button>
<button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="'folderSVG'"
    fillMode="link"
    title="Browse"
    [disabled]="true"
    themeColor="primary"
    ></button>
</p>
</div>

<div class="col-xs-12 col-sm-6 example-col">
    <p>Clear Buttons</p>
    <p>
        <button kendoButton (click)="onButtonClick()"
fillMode="clear">
            Browse
        </button>
        <button
            kendoButton
            (click)="onButtonClick()"
            [svgIcon]="'folderSVG'"
            fillMode="clear"
        >
            Browse
        </button>
        <button
            kendoButton
            (click)="onButtonClick()"
            [svgIcon]="'folderSVG'"
            fillMode="clear"
            title="Browse"
            ></button>
    </p>
    <p>
        <button
            kendoButton
            (click)="onButtonClick()"
            fillMode="clear"
            [disabled]="true"
        >
            Browse
        </button>
        <button
```

```
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="'folderSVG'"
    fillMode="clear"
    [disabled]="true"
  >
    Browse
  </button>
  <button
    kendoButton
    (click)="onButtonClick()"
    [svgIcon]="'folderSVG'"
    fillMode="clear"
    title="Browse"
    [disabled]="true"
  ></button>
</p>
</div>

<div class="col-xs-12 col-sm-6 example-col">
  <p>Clear Buttons (Primary)</p>
  <p>
    <button
      kendoButton
      (click)="onButtonClick()"
      fillMode="clear"
      themeColor="primary"
    >
      Browse
    </button>
    <button
      kendoButton
      (click)="onButtonClick()"
      [svgIcon]="'folderSVG'"
      fillMode="clear"
      themeColor="primary"
    >
      Browse
    </button>
    <button
      kendoButton
      (click)="onButtonClick()"
      [svgIcon]="'folderSVG'"
      fillMode="clear"
      title="Browse"
      themeColor="primary"
    ></button>
  </p>
  <p>
    <button
      kendoButton
```

```

        (click)="onButtonClick()"
        fillMode="clear"
        [disabled]="true"
        themeColor="primary"
      >
    Browse
  </button>
<button
  kendoButton
  (click)="onButtonClick()"
  [svgIcon]="folderSVG"
  fillMode="clear"
  [disabled]="true"
  themeColor="primary"
>
  Browse
</button>
<button
  kendoButton
  (click)="onButtonClick()"
  [svgIcon]="folderSVG"
  fillMode="clear"
  title="Browse"
  [disabled]="true"
  themeColor="primary"
></button>
</p>
</div>
</div>
``,
})
export class AppComponent {
  public folderSVG: SVGIcon = Collanse Code ^
```

# Key Features

- **Disabled Button**—You can use the configuration options of the Button to disable the component so that users are not able to interact with it.
- **Icon Button**—The Button enables you to display various types of icons including the built-in Kendo UI icons as well as FontAwesome and image icons.
- **Toggleable Button**—By using the toggleable feature, you can also show that a Button is inactive.
- **Appearance**—The Button delivers ready-to-use, predefined sets of styling options.
- **Globalization**—All Kendo UI for Angular Buttons provide globalization options.

To learn more about the appearance, anatomy, and accessibility of the Button, visit the [Progress Design System documentation](#)—an information portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

# Support and Learning Resources

- [Button Homepage](#)
- [Getting Started with the Kendo UI for Angular Buttons](#)
- [Button API Documentation](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Button Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular ButtonGroup Overview

The Kendo UI for Angular ButtonGroup is a container for two or more Button components.

Each Button in the ButtonGroup can be separately configured depending on the requirements of your project and according to its [API reference](#).

The following example demonstrates the ButtonGroup in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";

import {
  SVGIcon,
  boldIcon,
  italicIcon,
  underlineIcon,
} from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_BUTTONS],
  template: `
    <div class="row">
      <div class="col-sm-12 col-md-4 example-col">
        <p>Icons only</p>
        <kendo-buttongroup>
          <button
            kendoButton

```

```
[toggleable]="true"
[svgIcon]="boldSVG"
title="Bold"
></button>
<button
  kendoButton
  [toggleable]="true"
  [svgIcon]="italicSVG"
  title="Italic"
></button>
<button
  kendoButton
  [toggleable]="true"
  [svgIcon]="underlineSVG"
  title="Underline"
></button>
</kendo-buttongroup>
</div>
<div class="col-sm-12 col-md-4 example-col">
  <p>Text only</p>
  <kendo-buttongroup>
    <button kendoButton [toggleable]="true">Bold</button>
    <button kendoButton [toggleable]="true">Italic</button>
    <button kendoButton [toggleable]="true">Underline</button>
  </kendo-buttongroup>
</div>
<div class="col-sm-12 col-md-4 example-col">
  <p>Icons + Text</p>
  <kendo-buttongroup>
    <button kendoButton [toggleable]="true" [svgIcon]="boldSVG">
      Bold
    </button>
    <button kendoButton [toggleable]="true" [svgIcon]="italicSVG">
      Italic
    </button>
    <button kendoButton [toggleable]="true"
[svgIcon]="underlineSVG">
      Underline
    </button>
  </kendo-buttongroup>
</div>
</div>
<div class="row">
  <div class="col-sm-12 example-col">
    <p>Buttons with responsive width</p>
    <kendo-buttongroup width="100%">
      <button kendoButton [toggleable]="true" [svgIcon]="boldSVG">
        Bold
      </button>
      <button kendoButton [toggleable]="true" [svgIcon]="italicSVG">
        Italic
      </button>
    </kendo-buttongroup>
  </div>
</div>
```

```
        </button>
        <button kendoButton [toggleable]="true"
[svgIcon]="underlineSVG">
    Underline
    </button>
</kendo-buttongroup>
</div>
</div>
` ,
})
export class AppComponent {
    public boldSVG: SVGIcon = boldIcon;
    public italicSVG: SVGIcon = italicIcon;
    public underlineSVG: SVGIcon = underlineIcon;
}
```

Collapse Code ^

# Key Features

- **Disabled ButtonGroup**—You can use the configuration options of the ButtonGroup to disable the whole group of buttons or a single button so that users are not able to interact with it.
- **Selection mode**—You can render only some of the buttons within the ButtonGroup as active so that the user interaction with the rest of them is restricted.
- **Button collections**—You can iterate on a collection of configuration options and render multiple Button components.
- **Appearance**—The ButtonGroup delivers ready-to-use, predefined sets of styling options for modifying the whole group of buttons.
- **Globalization**—All Kendo UI for Angular Buttons provide globalization options.
- **Accessibility**—The ButtonGroup is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The ButtonGroup supports a number of keyboard shortcuts for processing various commands.

To learn more about the appearance, anatomy, and accessibility of the ButtonGroup, visit the [Progress Design System documentation](#)—an information portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

# Support and Learning Resources

- [ButtonGroup Homepage](#)
- [Getting Started with the Kendo UI for Angular Buttons](#)
- [API Reference of the ButtonGroup](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [ButtonGroup Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Chip Overview

The Kendo UI for Angular Chip allows users to enter information, make selections, filter content, or trigger actions.

The Chip represents a complex piece of information in a compact form—for example, an entity that can be a person, a place, or a thing. The component can be clicked or removed and supports various styling options. The Chip is commonly used in email templates where each chip is a single person.

The following example demonstrates the Chip in action.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { Component, ViewEncapsulation, ViewChild } from "@angular/core";
import {
  AutoCompleteComponent,
  KENDO_DROPDOWNS,
} from "@progress/kendo-angular-dropdowns";
import {
  ChipRemoveEvent,
  KENDO_BUTTONS,
} from "@progress/kendo-angular-buttons";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_BUTTONS, KENDO_DROPDOWNS, KENDO_INPUTS],
  template: `
    <div class="k-block">
```

```
@for (contact of selectedContacts; track contact) {
  <kendo-chip
    [label]= "contact.label"
    [removable]= "true"
    [iconClass]= "contact.iconClass"
    (remove)= "onRemove($event)"
  >
</kendo-chip>
}
<div class="example">
  <kendo-autocomplete
    #contactslist
    [data]= "contacts"
    class="contacts"
    valueField= "label"
    [kendoDropDownFilter]= "{ operator: 'contains' }"
    [filterable]= "true"
    placeholder= "To: Email Adress*"
    (valueChange)= "valueChange($event)"
  >
</kendo-autocomplete>
  <textarea class="k-textarea k-input">
    Hi there! Don't miss out our dinner party!
  </textarea>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None,
styles: [
  .k-chip {
    margin-right: 10px;
  }
  .k-block {
    min-height: 300px;
    padding: 20px;
  }
  .k-textarea {
    width: 100%;
    min-height: 145px;
  }
  .contacts {
    border-width: 0 0 1px 0;
    width: 100%;
    margin: 30px 0;
  }
  .contacts.k-focus {
    box-shadow: 0 4px 2px -2px rgba(0, 0, 0, 0.03);
  }
  .pedro {
    background-image: url("assets/dropdowns/contacts/SPLIR.jpg");
  }
]
```

```
        }
      .thomas {
        background-image: url("^assets/dropdowns/contacts/RICSU.jpg");
      }
      .christina {
        background-image: url("^assets/dropdowns/contacts/BERGS.jpg");
      }
      .paula {
        background-image: url("^assets/dropdowns/contacts/RATTC.jpg");
      }
      .maria {
        background-image: url("^assets/dropdowns/contacts/ALFKI.jpg");
      }
      .yang {
        background-image: url("^assets/dropdowns/contacts/CHOPS.jpg");
      }
      .anna {
        background-image: url("^assets/dropdowns/contacts/EASTC.jpg");
      }
    }
  ],
})
export class AppComponent {
  @ViewChild("contactslist") public list: AutoCompleteComponent;

  public contacts: Array<{ label: string; iconClass: string }> = [
    { label: "Pedro Afonso", iconClass: "k-chip-avatar pedro" },
    { label: "Maria Shore", iconClass: "k-chip-avatar maria" },
    { label: "Thomas Hardy", iconClass: "k-chip-avatar thomas" },
    { label: "Christina Berg", iconClass: "k-chip-avatar christina" },
    { label: "Paula Wilson", iconClass: "k-chip-avatar paula" },
  ];

  public selectedContacts: Array<{ label: string; iconClass: string }> =
  [
    this.contacts[1],
  ];

  public valueChange(contact: string): void {
    if (contact === "") {
      return;
    }

    const contactData = this.contacts.find((c) => c.label === contact);

    if (
      contactData !== undefined &&
      !this.selectedContacts.includes(contactData)
    ) {
      this.selectedContacts.push(contactData);
    }
  }
}
```

```
        this.list.reset();
    }

    public onRemove(e: ChipRemoveEvent): void {
        console.log("Remove event arguments: ", e);
        const index = this.selectedContacts
            .map((c) => c.label)
            .indexOf(e.sender.label);
        this.selectedContacts.splice(index, 1);
    }
}
```

Collanse Code ^

## Key Features

- **Content**—You can specify the content of the Chip by passing a string or using custom configuration options.
- **Sets of Chip for different use cases**—Depending on your requirements, you can implement different sets of Chips for displaying complex information in a compact form such as filters for a collection, multiple chips with multiple selection options, text entries that can be dynamically added or removed by the user, and more.
- **Customization**—You can add a Select or Remove custom icon to the Chip as well as display avatars in it.
- **Appearance**—The Chip delivers ready-to-use, predefined sets of styling options.
- **Globalization**—All Kendo UI for Angular Buttons provide globalization options.

To learn more about the appearance, anatomy, and accessibility of the Chip, visit the [Progress Design System documentation](#)—an information portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

## Support and Learning Resources

- [Chip Homepage](#)
- [Getting Started with the Kendo UI for Angular Buttons](#)

- [API Reference of the Chip](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Chip Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular ChipList Overview

The Kendo UI for Angular ChipList allows you to maintain a set of selected chips.

The ChipList is a container of two or more individual Chip components. They represent a complex piece of information in a compact form—for example, an entity that can be a person, a place, or a thing. Each chip from the list can be selected or removed and supports various styling options. The ChipList component is commonly used for single or multiple selections such as additions to an ordered meal.

The following example demonstrates how to dynamically change the Chips in a ChipList.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { ICON_SETTINGS, KENDO_ICONS } from "@progress/kendo-angular-
icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_BUTTONS, KENDO_ICONS],
  providers: [{ provide: ICON_SETTINGS, useValue: { type: "font" } }],
  template: `
    Order meal:
    <kendo-butongroup class="meals" selection="single">
      @for (meal of meals; track meal) {
        <button
          kendoButton
          [toggleable]="true"
          [selected]="meal.selected"
          (click)="selectedMeal(meal)">
      
```

```
>
  <div class="meal-icon">{{ meal.icon }}</div>
  {{ meal.name }}
</button>
}
</kendo-buttongroup>
<div class="separator">
  <kendo-chiplist selection="multiple">
    <div>Add more:</div>
    @for (addition of additions; track addition) {
      <kendo-chip [label]="addition.label"> </kendo-chip>
    }
  </kendo-chiplist>
</div>
``,
encapsulation: ViewEncapsulation.None,
styles: [
  ``
  .separator {
    margin-top: 12px;
  }
  .meal-icon {
    margin-right: 5px;
  }
  .meals {
    margin-left: 5px;
  }
  ``,
],
`),
})
export class AppComponent {
  public meals: Array<{ name: string; icon: string; selected?: boolean }> = [
    { name: "Pizza", icon: "🍕", selected: true },
    { name: "Sushi", icon: "🍣" },
    { name: "Burger", icon: "🍔" },
  ];
  public additions: Array<{ label: string }> = [
    { label: "Ketchup" },
    { label: "Mustard" },
    { label: "Mayonnaise" },
  ];
}

public selectedMeal(meal: { name: string; icon: string }): void {
  switch (meal.name) {
    case "Pizza":
      this.additions = [
        { label: "Ketchup" },
        { label: "Mustard" },
        { label: "Mayonnaise" },
      ];
    
```

```
        break;
    case "Sushi":
        this.additions = [
            { label: "Wasabi" },
            { label: "Ginger" },
            { label: "Soy sauce" },
        ];
        break;
    case "Burger":
        this.additions = [
            { label: "Onions" },
            { label: "Avocado" },
            { label: "Eggs" },
        ];
        break;
    default:
        break;
}
}
}
```

Collapse Code ^

## Key Features

- **Selection mode**—You can set the selection mode of the ChipList by using the available configuration options.
- **Globalization**—All Kendo UI for Angular Buttons provide globalization options.

## Support and Learning Resources

- [ChipList Homepage](#)
- [Getting Started with the Kendo UI for Angular Buttons](#)
- [API Reference of the Chip](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [ChipList Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)

- Knowledge Base

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular DropDownButton Overview

The Kendo UI for Angular DropDownButton looks like the Button and when clicked, it displays a popup list with action items.

The following example demonstrates the DropDownButton in action.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { Component } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_BUTTONS],
  template: `
    <div class="row">
      <div class="col-xs-12 col-sm-6 col-lg-3 example-col">
        <p>Default Button</p>
        <kendo-dropdownbutton [data]="data">
          User Settings
        </kendo-dropdownbutton>
      </div>
      <div class="col-xs-12 col-sm-6 col-lg-3 example-col">
        <p>Primary Button</p>
        <kendo-dropdownbutton [data]="data" themeColor="primary">
          User Settings
        </kendo-dropdownbutton>
      </div>
      <div class="col-xs-12 col-sm-6 col-lg-3 example-col">
        <p>Outline Button</p>
        <kendo-dropdownbutton [data]="data" fillMode="outline">
          User Settings
        </kendo-dropdownbutton>
      </div>
    </div>
  `)
```

```

<div class="col-xs-12 col-sm-6 col-lg-3 example-col">
  <p>Flat Button</p>
  <kendo-dropdownbutton [data]="data" fillMode="flat">
    User Settings
  </kendo-dropdownbutton>
</div>
</div>
,
})
export class AppComponent {
  data = [
    { text: "My Profile" },
    { text: "Friend Requests" },
    { text: "Account Settings" },
    { text: "Support" },
    { text: "Log Out" },
  ];
}

```

[Collapse Code ^](#)

# Key Features

- **Disabled DropDownButton**—You can use the configuration options of the DropDownButton to disable the component so that users are not able to interact with it.
- **Icon DropDownButton**—The DropDownButton enables you to display various types of icons including the built-in Kendo UI icons as well as FontAwesome and image icons.
- **Popup items and templates**—You can configure the DropDownButton popup and popup items by visually enhancing their rendering and by using templates for controlling their content.
- **Data binding**—You can bind the DropDownButton to data of the primitive (strings and numbers) or of the complex (data inside objects) type.
- **Globalization**—All Kendo UI for Angular Buttons provide globalization options.
- **Accessibility**—The DropDownButton is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The DropDownButton supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [DropDownButton Homepage](#)
- [Getting Started with the Kendo UI for Angular Buttons](#)
- [API Reference of the DropDownButton](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [DropDownButton Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular FloatingActionButton Overview

The FloatingActionButton specifies the primary or the most common action in an application.

The following example demonstrates the FloatingActionButton in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_LAYOUT } from "@progress/kendo-angular-layout";
import { MusicCardComponent } from "./music-card.component";
import { NotesComponent } from "./notes.component";
@Component({
  selector: "my-app",
  standalone: true,
  imports: [
    CommonModule,
    KENDO_BUTTONS,
    KENDO_LAYOUT,
    MusicCardComponent,
    NotesComponent,
  ],
  template: `
    <div class="example-wrap">
      <music-card></music-card>
      <my-notes></my-notes>
    </div>
  `,
  styleUrls: ["./styles.css"],
})
```

# Key Features

- **Disabled FloatingActionButton**—You can use the configuration options of the FloatingActionButton to disable the component so that users are not able to interact with it.
  - **Icon FloatingActionButton**—The FloatingActionButton enables you to display various types of icons including the built-in Kendo UI icons as well as FontAwesome and image icons.
  - **Positioning**—You can align and position the FloatingActionButton in relation to a web page, an HTML element, or an Angular Component.
  - **Dial items**—You can display a stack of related sub-actions when the FloatingActionButton is clicked.
  - **Controlling the open state**—You can control the state of the FloatingActionButton Dial items list by configuring their initial state and preventing the `open` and `close` events.
  - **Templates**—You can define templates to display custom content or customize the appearance of the FloatingActionButton Dial items list.
  - **Appearance**—The FloatingActionButton delivers ready-to-use, predefined sets of styling options.
  - **Globalization**—All Kendo UI for Angular Buttons provide globalization options.
  - **Accessibility**—The FloatingActionButton is accessible for screen readers and supports WAI-ARIA attributes.
  - **Keyboard navigation**—The FloatingActionButton supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [FloatingActionButton Homepage](#)
  - [Getting Started with the Kendo UI for Angular Buttons](#)
  - [API Reference of the FloatingActionButton](#)

- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [FloatingActionButton Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Angular Buttons Overview

The Kendo UI for Angular Buttons provide a clickable UI functionality which can be set to display arbitrary content.

They include a variety of button types and styles that have extensive configuration options. This flexibility allows you to quickly and easily create the exact button you need to fit your specific requirements for functionality and appearance.

The Buttons are built from the ground up and specifically for Angular, so that you get high-performance button controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



## Button

An Angular directive for button elements.



## ButtonGroup

Grouped Buttons with identical functionalities.



## Chip

A component for entering and filtering content, or triggering actions.

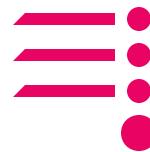


## ChipList

A component for maintaining sets of selected chips.



## DropDownButton



## FloatingActionButton

A component for executing additional action items.



A component for executing primary or most common actions.

## SplitButton

A component for executing default or predefined actions.

# Angular Buttons Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import {
  SVGIcon,
  gearIcon,
  clipboardIcon,
  plusIcon,
  shareIcon,
  googleIcon,
  redditIcon,
  dribbbleIcon,
  clipboardTextIcon,
  clipboardMarkdownIcon,
  clipboardCodeIcon,
  clockIcon,
  lockIcon,
} from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_BUTTONS],
  template: `
    <div class="example-wrapper">
```

```
<div class="example-col">
  <p>Button</p>
  <button kendoButton (click)="onButtonClick()">Default</button>
  <button kendoButton (click)="onButtonClick()" themeColor="primary">
    Primary
  </button>
  <button kendoButton (click)="onButtonClick()" [disabled]="true">
    Disabled
  </button>
</div>
<div class="example-col">
  <p>ButtonGroup</p>
  <kendo-buttongroup>
    <button kendoButton [toggleable]="true">Option A</button>
    <button kendoButton [toggleable]="true">Option B</button>
    <button kendoButton [toggleable]="true">Option C</button>
  </kendo-buttongroup>
</div>
<div class="example-col">
  <p>DropDownButton</p>
  <kendo-dropdownbutton
    [data]="dropDownButtonItem"
    [svgIcon]="gearSVG"
    title="User Settings"
    (itemClick)="onSplitButtonItemClick($event)">
    User Settings
  </kendo-dropdownbutton>
</div>
<div class="example-col">
  <p>SplitButton</p>
  <kendo-splitbutton
    [data]="splitButtonItem"
    [svgIcon]="clipboardSVG"
    title="Paste"
    (itemClick)="onSplitButtonItemClick($event)"
    (buttonClick)="onSplitButtonClick()">
    Paste
  </kendo-splitbutton>
</div>
<div class="example-col">
  <p>Chip</p>
  <kendo-chip-list selection="multiple">
    @for (chip of chips; track chip) {
      <kendo-chip
        [label]="chip.label"
        [svgIcon]="chip.svgIcon">
    </kendo-chip>
  }
</div>
```

```
        </kendo-chip-list>
    </div>
    <div class="example-col">
        <kendo-floatingactionbutton
            [svgIcon]="plusSVG"
            text="Add New"
            title="Add New"
            [align]={ horizontal: 'start', vertical: 'bottom' }"
            (click)="onFabClick()"
        >
        </kendo-floatingactionbutton>
    </div>
    <div class="example-col">
        <kendo-floatingactionbutton [svgIcon]="shareSVG"
[dialItems]="fabItems">
        </kendo-floatingactionbutton>
    </div>
</div>
``,
})
export class AppComponent {
    public gearSVG: SVGIcon = gearIcon;
    public clipboardSVG: SVGIcon = clipboardIcon;
    public plusSVG: SVGIcon = plusIcon;
    public shareSVG: SVGIcon = shareIcon;

    public fabItems = [
        { svgIcon: googleIcon, label: "Google" },
        { svgIcon: redditIcon, label: "Reddit" },
        { svgIcon: dribbbleIcon, label: "Dribbble" },
    ];

    public splitButtonItems = [
        {
            text: "Keep Text Only",
            svgIcon: clipboardTextIcon,
            click: (): void => {
                console.log("Keep Text Only click handler");
            },
        },
        {
            text: "Paste as HTML",
            svgIcon: clipboardCodeIcon,
        },
        {
            text: "Paste Markdown",
            svgIcon: clipboardMarkdownIcon,
        },
        {
            text: "Set Default Paste",
        },
    ];
}
```

```
];

public dropDownButtonItems = [
  { text: "My Profile" },
  { text: "Friend Requests" },
  { text: "Account Settings" },
  { text: "Support" },
  { text: "Log Out" },
];

public chips = [
  {
    label: "Alarm",
    svgIcon: clockIcon,
  },
  {
    label: "Lock",
    svgIcon: lockIcon,
  },
];

public onFabClick(): void {
  console.log("Added");
}

public onSplitButtonClick(): void {
  console.log("Paste");
}

public onSplitButtonItemClick(dataItem: { [key: string]: unknown }): void {
  if (dataItem) {
    console.log(dataItem.text);
  }
}

public onButtonClick(): void {
  console.log("click");
}
}
```

Collapse Code

## Angular Buttons Key Features

Each Kendo UI for Angular Buttons component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts

to improve the performance and add more value to the existing Buttons library as well as develop new features and controls to it.

## Disabled Buttons

You can render the Buttons in their disabled state so that users cannot interact with them. [Read more about the Disabled Button...](#)

## Icon Buttons

All Buttons components provide configuration options that enable the easy rendering of icon, icon-and-text, or text-only buttons. What's more, you can not only use any icon from the rich collection of built-in Kendo UI icons, but also FontAwesome or image icons. [Read more about the Icon Button...](#)

## Customizable Appearance

The color and style of the Buttons are normally picked up by the current Kendo UI theme, but each aspect of the Buttons can be customized by theme variables or configuration options. Kendo UI for Angular delivers a set of popular themes including Bootstrap and Material, all of which can be easily customized with the Progress ThemeBuilder online utility. [Read more about the appearance of the Button...](#)

## Globalization

The Kendo UI for Angular Buttons support globalization to ensure that each Buttons component can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Buttons support rendering in a right-to-left (RTL) direction. [Read more about Buttons globalization...](#)

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular Buttons are part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library —no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of Kendo UI for Angular Buttons, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Buttons](#)
- [API Reference of the Buttons](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)

- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Support and Learning Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular SplitButton Overview

The Kendo UI for Angular SplitButton allows the user to execute a default action which is bound to a Button or to choose a predefined action from a drop-down list.

The following example demonstrates the SplitButton in action.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { Component } from "@angular/core";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { CommonModule } from "@angular/common";

import {
  SVGIcon,
  clipboardIcon,
  clipboardCodeIcon,
  clipboardTextIcon,
  clipboardMarkdownIcon,
} from "@progress/kendo-svg-icons";
import { KENDO_ICONS } from "@progress/kendo-angular-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_BUTTONS, KENDO_ICONS],
  template:
    <div class="row">
      <div class="col-xs-12 col-sm-6 col-lg-3 example-col">
        <p>Default Button</p>
        <kendo-splitbutton
          (buttonClick)="onPaste()"
          [data]="data"
          [svgIcon]="clipboardSVG"
        >
          Paste
        </kendo-splitbutton>
      </div>
    </div>
  </template>
})
```

```
</div>

<div class="col-xs-12 col-sm-6 col-lg-3 example-col">
  <p>Flat Button</p>
  <kendo-splitbutton
    (buttonClick)="onPaste()"
    [data]="data"
    [svgIcon]="clipboardSVG"
    fillMode="flat"
  >
    Paste
  </kendo-splitbutton>
</div>

<div class="col-xs-12 col-sm-6 col-lg-3 example-col">
  <p>Outline Button</p>
  <kendo-splitbutton
    (buttonClick)="onPaste()"
    [data]="data"
    [svgIcon]="clipboardSVG"
    fillMode="outline"
  >
    Paste
  </kendo-splitbutton>
</div>
</div>
``,
})
export class AppComponent {
  public clipboardSVG: SVGIcon = clipboardIcon;

  public data = [
    {
      text: "Keep Text Only",
      svgIcon: clipboardTextIcon,
      click: (): void => {
        console.log("Keep Text Only");
      },
    },
    {
      text: "Paste as HTML",
      svgIcon: clipboardCodeIcon,
      click: (): void => {
        console.log("Paste as HTML");
      },
    },
    {
      text: "Paste Markdown",
      svgIcon: clipboardMarkdownIcon,
      click: (): void => {
        console.log("Paste Markdown");
      },
    },
  ];
}
```

```
        },
      },
      {
        text: "Set Default Paste",
        click: (): void => {
          console.log("Set Default Paste");
        },
      },
    ],
  };

  public onPaste(): void {
    console.log("Paste");
  }
}
```

Collanse Code ^

# Key Features

- **Disabled SplitButton**—You can use the configuration options of the SplitButton to disable the component so that users are not able to interact with it.
- **Icon SplitButton**—The SplitButton enables you to display various types of icons including the built-in Kendo UI icons as well as FontAwesome and image icons.
- **Popup items and templates**—You can configure the SplitButton popup and popup items by visually enhancing their rendering and by using templates for controlling their content.
- **Content**—You can configure the content of the SplitButton by adding it between the opening and the closing tags and by setting its `text` property.
- **Data binding**—You can bind the SplitButton to data of the primitive (strings and numbers) or of the complex (data inside objects) type.
- **Globalization**—All Kendo UI for Angular Buttons provide globalization options.
- **Accessibility**—The SplitButton is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The SplitButton supports a number of keyboard shortcuts for processing various commands.

To learn more about the appearance, anatomy, and accessibility of the SplitButton, visit the [Progress Design System documentation](#)—an information

portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

# Support and Learning Resources

- [SplitButton Homepage](#)
- [Getting Started with the Kendo UI for Angular Buttons](#)
- [API Reference of the SplitButton](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [SplitButton Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Angular Data Query Overview

The Data Query is a package for applying the sorting, filtering, grouping, and aggregate data operations.

The package exports functions which help you perform in-memory data operations. It also serializes the data descriptors according to the OData v4 URL specification.

## Angular Data Query Example

```
import { orderBy } from '@progress/kendo-data-query';

const data = [
  { name: "Pork", category: "Food", subcategory: "Meat" },
  { name: "Pepper", category: "Food", subcategory: "Vegetables" },
  { name: "Beef", category: "Food", subcategory: "Meat" }
];

const result = orderBy(data, [{ field: "name", dir: "asc" }]);

console.log(result);

/* output
[
  { "name": "Beef", "category": "Food", "subcategory": "Meat" },
  { "name": "Pepper", "category": "Food", "subcategory": "Vegetables" },
  { "name": "Pork", "category": "Food", "subcategory": "Meat" }
]
```

JS

## Angular Data Query Key Features

The Kendo UI Data Query package delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI team constantly invests efforts to improve the performance and add more value to the existing Data Query library as well as develop new features to it.

## Helpers for Bulk Operations

The Data Query provides functions that help you handle bulk data operations such as sorting, filtering, grouping, and aggregates. [Read more about the helper functions of the package...](#)

## Licensing

The Kendo UI Data Query package is part of all available Kendo UI libraries. Depending on the trial version and commercial license support that each Kendo UI suite or flavor offers, the Data Query package may be available for trial and commercial users, and as part of the open-source Kendo UI for jQuery Core suite.

## Support Options

For any questions about the use of the Kendo UI Data Query, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI feedback portal](#) and Roadmaps provide information on the features in discussion and also the planned ones for release.

- [Angular roadmap](#)
- [React roadmap](#)
- [Vue roadmap](#)
- Kendo UI uses GitHub Issues as its bug tracker and you can submit any related reports there. Also, check out the closed list.
  - [Angular tracker](#)
  - [React tracker](#)
  - [Vue tracker](#)
- Of course, the Kendo UI team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Data Query Homepage](#)
- [Getting Started with the Kendo UI Data Query](#)
- [API Reference of the Data Query](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Virtual Classroom \(Training Courses for Registered Users\)](#)
- [Data Query Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Calendar Overview

The Kendo UI for Angular Calendar is a form component that represents a graphical Gregorian calendar.

It supports the selection of and navigation between dates as well as data templates and ranges for scheduling appointments.

The following example demonstrates the Angular Calendar in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_DATEINPUTS } from "@progress/kendo-angular-dateinputs";
import { KENDO_LABEL } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_LABEL, KENDO_DATEINPUTS],
  styles: [".k-calendar { margin: 0 auto; }"],
  template: `
    <div class="row example-wrapper">
      <div class="col-xs-12 col-md-6 example-col">
        <kendo-label
          class="k-display-block"
          [for]="calendarInfinite"
          text="Infinite Calendar"
        ></kendo-label>
        <kendo-calendar #calendarInfinite type="infinite"></kendo-
calendar>
      </div>
      <div class="col-xs-12 col-md-6 example-col">
        <kendo-label
          class="k-display-block"
          [for]="calendarClassic"
        ></kendo-label>
      </div>
    </div>
  </template>
})
```

```
        text="Classic Calendar"
      ></kendo-label>
      <kendo-calendar #calendarClassic type="classic"></kendo-
    calendar>
    </div>
  </div>
  ,
})
export class AppComponent {}
```

Collanse Code ^

# Key Features

- **Calendar types**—You can set the type of the Angular Calendar and choose between the default infinite layout and the classic rendering.
- **Disabled Calendar**—You can use the configuration options of the Calendar to disable the component so that users are not able to interact with it.
- **Focused and selected dates**—You can also control the focused and selected dates within the Calendar.
- **Disabled dates**—The Calendar supports specific approaches for disabling a selection of dates such as through using a function, an array of dates, or an array of days.
- **Date ranges**—Within the Calendar, date ranges can be defined by setting a start and end date for a period of time.
- **Fast navigation sidebar**—The Calendar supports configuration options for controlling the default navigation sidebar.
- **View options**—You can toggle the visibility of other month days and control the initially loaded page of the Calendar and render it in a month, year, decade, or year view.
- **View selection depth**—The Angular Calendar provides an option for setting the view depth to which the user can navigate.
- **Selection modes**—Apart from its default single selection mode, the Calendar supports options for `multiple`, `range`, and `week` selection.
- **Templates**—You can customize the content of each Calendar cell and define a cell template.
- **Week number column**—The Calendar also supports options for rendering a column displaying the number of the weeks.

- **Forms support**—You can use the Calendar both in template-driven and reactive Angular forms.
- **Integration with JSON**—As the Calendar works only with `date` JavaScript instances while the received data from the server is serialized in a JSON format, the component provides options for binding it to dates which are serialized as strings.
- **Globalization**—All Kendo UI for Angular Date Inputs provide globalization options.
- **Accessibility**—The Calendar is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The Calendar supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [Calendar Homepage](#)
- [Getting Started with the Kendo UI for Angular Date Inputs](#)
- [API Reference of the Calendar](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Calendar Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular DateInput Overview

The Kendo UI for Angular DateInput represents an input field that recognizes and formats scheduling values such as dates.

The following example demonstrates the DateInput in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_DATEINPUTS } from "@progress/kendo-angular-dateinputs";
import { KENDO_LABEL } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_LABEL, KENDO_DATEINPUTS],
  styleUrls: ["./styles.css"],
  styles: [
    `
      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="profile-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column no-flex">
            <div class="sidebar-container k-skeleton">
              <div class="avatar-name-container">
                <div class="k-skeleton skeleton-avatar"></div>
                <div class="name-container">
                  <div class="k-skeleton skeleton-text"></div>
                  <div class="k-skeleton skeleton-small-text-short">
                    <div class="k-skeleton skeleton-text"></div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  `
})
```

```

        </div>
    </div>
    <div class="description-container">
        <div class="k-skeleton skeleton-small-text"></div>
        <div class="k-skeleton skeleton-small-text"></div>
        <div class="k-skeleton skeleton-small-text"></div>
    </div>
</div>
<div class="card-column">
    <div class="avatar-title-container">
        <div class="k-skeleton skeleton-avatar"></div>
        <h4 class="k-h4">My Profile</h4>
    </div>
    <div class="component-container">
        <kendo-label text="Date of Birth">
            <kendo-dateinput></kendo-dateinput>
        </kendo-label>
    </div>
    <div class="skeleton-container top">
        <div class="k-skeleton skeleton-box-small"></div>
        <div class="k-skeleton skeleton-box-large"></div>
    </div>
    <div class="skeleton-container bottom">
        <div class="k-skeleton skeleton-box-small"></div>
        <div class="k-skeleton skeleton-box-large-double"></div>
    </div>
    </div>
</div>
</div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {}

```

[Collapse Code ^](#)

# Key Features

- **Disabled DateInput**—You can use the configuration options of the DateInput to disable the component so that users are not able to interact with it.
- **Read-only DateInput**—The DateInput provides a configuration option for rendering it in its read-only state.

- **Date ranges**—Within the DateInput, date ranges can be defined by setting a start and end date for a period of time.
- **Spin buttons**—You can enable the spin buttons of the DateInput to increase or decrease the date value by adding or subtracting days.
- **Incremental steps**—The DateInput enables you to change the default step for increasing and decreasing the parts of its date values.
- **Formats**—You can display the DateInput in its single format at all times or configure it to show the value in different formats when the input is focused or blurred.
- **Placeholders**—The DateInput provides options for setting its input field and render a text hint, a floating label, or descriptions for its format sections.
- **Incomplete date validation**—The incomplete date validation feature of the DateInput ensures that users do not accidentally leave a non-required field partially populated.
- **Forms Support**—You can use the DateInput both in template-driven and reactive Angular forms.
- **Integration with JSON**—As the DateInput works only with `date` JavaScript instances while the received data from the server is serialized in a JSON format, the component provides options for binding it to dates which are serialized as strings.
- **Globalization**—All Kendo UI for Angular Date Inputs provide globalization options.
- **Accessibility**—The DateInput is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The DateInput supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [DateInput Homepage](#)
- [Getting Started with the Kendo UI for Angular Date Inputs](#)
- [API Reference of the DateInput](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [DateInput Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)

- Knowledge Base

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular DatePicker Overview

The Kendo UI for Angular DatePicker combines the Kendo UI DateInput and Calendar components.

It enables the user to enter or pick a date value.

The following example demonstrates the Angular DatePicker in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_DATEINPUTS } from "@progress/kendo-angular-dateinputs";
import { KENDO_ICONS } from "@progress/kendo-angular-icons";
import { KENDO_LABEL } from "@progress/kendo-angular-label";
import { ImageIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_LABEL, KENDO_DATEINPUTS, KENDO_ICONS],
  styleUrls: ["./styles.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="visit-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column">
            <h4 class="k-h4">Schedule Your Visit</h4>
            <div class="component-container">
              <kendo-label text="Select Appointment Date">
            </kendo-label>
          </div>
        </div>
      </div>
    </div>
  `
})
```

```

        <kendo-datepicker
            placeholder="Choose a date ..."
        ></kendo-datepicker>
    </kendo-label>
</div>
<div class="skeleton-container top">
    <div class="k-skeleton skeleton-box-small"></div>
    <div class="k-skeleton skeleton-box-large"></div>
</div>
<div class="skeleton-container bottom">
    <div class="k-skeleton skeleton-box-medium"></div>
    <div class="k-skeleton skeleton-box-medium"></div>
</div>
</div>
<div class="card-column image-container">
    <div class="k-skeleton skeleton-image">
        <kendo-svgicon size="xxxlarge" [icon]="imageSVG"></kendo-
svgicon>
    </div>
</div>
</div>
<div class="card-row">
    <div class="k-skeleton skeleton-box-half"></div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
    public imageSVG: SVGIcon = imageIcon;
}

```

[Collapse Code ^](#)

# Getting Started with the Kendo UI for Angular DatePicker

To get started with the Kendo UI for Angular DatePicker, you can follow the steps in the following comprehensive article: [Getting Started with the Kendo UI for Angular Date Inputs](#).

# Key Features

- **Calendar options**—You can use various options to configure the popup calendar within the DatePicker—for example, switch between different calendar layouts, handle the animation of the calendar navigation, set the focused dates and the initially loaded calendar page, and more.
- **Disabled Angular DatePicker**—You can use the configuration options of the DatePicker to disable the component so that users are not able to interact with it.
- **Read-only Angular DatePicker**—The DatePicker provides a configuration option for rendering it or its input field only in a read-only state.
- **Disabled dates**—The DatePicker supports specific approaches for disabling a selection of dates such as through using a function, an array of dates, or an array of days.
- **Date ranges**—Within the DatePicker, date ranges can be defined by setting a start and end date for a period of time.
- **Fast navigation sidebar**—The DatePicker supports configuration options for controlling the default navigation sidebar.
- **Templates**—You can customize the content of each Calendar cell within the DatePicker and define a cell template.
- **Formats**—You can display the DatePicker in its single format at all times or configure it to show the value in different formats when the input is focused or blurred.
- **Placeholders**—The DatePicker provides options for setting its input field and render a text hint or descriptions for its format sections.
- **Incomplete date validation**—The incomplete date validation feature of the DatePicker ensures that users do not accidentally leave a non-required field partially populated.
- **Forms support**—You can use the DatePicker both in template-driven and reactive Angular forms.
- **Integration with JSON**—As the DatePicker works only with `date` JavaScript instances while the received data from the server is serialized in a JSON format, the component provides options for binding it to dates which are serialized as strings.
- **Typing control**—The DatePicker provides options for controlling the typing behavior of the input field like caret mode, auto switch parts, auto switch keys, enable mouse wheel, and auto-fill.

- **Adaptive mode**—The DatePicker provides an adaptive mode for the Calendar component, which allows you to control the visibility of the Calendar based on the screen size.
- **Globalization**—All Kendo UI for Angular Date Inputs provide globalization options.
- **Accessibility**—The DatePicker is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The DatePicker supports a number of keyboard shortcuts for processing various commands.

## Frequently Asked Questions

### What is the Kendo UI for Angular DatePicker?

The Kendo UI for Angular DatePicker is a native Angular component with a modern UI that enables users to enter or pick a date value.

### How to Create a DatePicker in Angular?

To create a Grid in Angular, you need to install the Kendo UI for Angular DateInputs package and then import the `DatePickerModule` or `DateInputsModule` in your application. After that, you can use the `kendo-datepicker` component in your Angular application. For more information, refer to the [Getting Started with the Kendo UI for Angular DatePicker](#) article.

## Support and Learning Resources

- [Angular DatePicker Homepage](#)
- [Getting Started with the Kendo UI for Angular Date Inputs](#)
- [API Reference of the Angular DatePicker Component](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)

- [Video Courses](#)
- [Angular DatePicker Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular DateRange Overview

The Kendo UI for Angular DateRange is a container for holding start and end date inputs, and a date range popup.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_DATEINPUTS } from "@progress/kendo-angular-dateinputs";
import { KENDO_ICONS } from "@progress/kendo-angular-icons";
import { KENDO_LABELS } from "@progress/kendo-angular-label";
import { imageIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_LABELS, KENDO_BUTTONS, KENDO_DATEINPUTS, KENDO_ICONS],
  styleUrls: ["./styles.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="visit-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column">
            <h4 class="k-h4">Reservation Dates</h4>
            <div class="component-container">
              <kendo-daterange>
                <kendo-floatinglabel text="Check-in">
                  <kendo-dateinput
                    kendoDateRangeStartInput

```

```

        [(value)]="range.start"
      ></kendo-dateinput>
    </kendo-floatinglabel>
    <kendo-floatinglabel text="Check-out">
      <kendo-dateinput
        kendoDateRangeEndInput
        [(value)]="range.end"
      ></kendo-dateinput>
    </kendo-floatinglabel>
  </kendo-daterange>
</div>
<div class="skeleton-container top">
  <div class="k-skeleton skeleton-box-small"></div>
  <div class="k-skeleton skeleton-box-large"></div>
</div>
<div class="skeleton-container bottom">
  <div class="k-skeleton skeleton-box-medium"></div>
  <div class="k-skeleton skeleton-box-medium"></div>
</div>
</div>
<div class="card-column image-container">
  <div class="k-skeleton skeleton-image">
    <kendo-svgicon size="xxxlarge" [icon]="imageSVG"></kendo-
svgicon>
  </div>
</div>
</div>
<div class="card-row">
  <div class="k-skeleton skeleton-box-half"></div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
  public range = { start: null, end: null };
  public imageSVG: SVGIcon = imageIcon;
}

```

[Collapse Code ^](#)

# Key Features

- **Selection of dates**—The DateRange allows the selection of dates by linking its default popup with the MultiViewCalendar component with the help of the directives it

delivers.

- **Selection of date ranges**—The DateRange uses the MultiViewCalendar component to provide a date range selection functionality.
- **Popup options**—The DateRange provides a number of options for customizing the popup such as controlling its visibility and animation, setting its alignment and position, and more.
- **Auto-correction of date ranges**—You can use the DateRange directives and enable the automatic correction of the selected date range.
- **Globalization**—All Kendo UI for Angular Date Inputs provide globalization options.

## Support and Learning Resources

- [DateRange Homepage](#)
- [Getting Started with the Kendo UI for Angular Date Inputs](#)
- [API Reference of the DateRange](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [DateRange Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular DateTimePicker Overview

The Kendo UI for Angular DateTime Picker enables the user to select date and time values.

The component combines the Kendo UI DateInput, Calendar, and TimePicker.

The following example demonstrates the Angular DateTimePicker in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_DATEINPUTS } from "@progress/kendo-angular-dateinputs";
import { KENDO_ICONS } from "@progress/kendo-angular-icons";
import { KENDO_LABEL } from "@progress/kendo-angular-label";
import { ImageIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_LABEL, KENDO_DATEINPUTS, KENDO_ICONS],
  styleUrls: ["./styles.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="visit-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column">
            <h4 class="k-h4">Schedule Your Visit</h4>
            <div class="component-container">
              <kendo-label text="Date and Time">

```

```

        <kendo-datetimepicker
            [format]="format"
            [(value)]="value"
        ></kendo-datetimepicker>
    </kendo-label>
</div>
<div class="skeleton-container top">
    <div class="k-skeleton skeleton-box-small"></div>
    <div class="k-skeleton skeleton-box-large"></div>
</div>
<div class="skeleton-container bottom">
    <div class="k-skeleton skeleton-box-medium"></div>
    <div class="k-skeleton skeleton-box-medium"></div>
</div>
</div>
<div class="card-column image-container">
    <div class="k-skeleton skeleton-image">
        <kendo-svgicon size="xxxlarge" [icon]="imageSVG"></kendo-
svgicon>
    </div>
</div>
</div>
<div class="card-row">
    <div class="k-skeleton skeleton-box-half"></div>
</div>
</div>
</div>
,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
    public value: Date;
    public format = "MM/dd/yyyy HH:mm";
    public imageSVG: SVGIcon = imageIcon;
}

```

[Collapse Code ^](#)

# Key Features

- **Calendar options**—You can use various options to configure the popup calendar within the Angular DateTimePicker—for example, switch between different calendar layouts, handle the animation of the calendar navigation, set the focused dates, and more.

- **Disabled DateTimePicker**—You can use the configuration options of the DateTimePicker to disable the component so that users are not able to interact with it.
- **Read-only DateTimePicker**—The DateTimePicker provides a configuration option for rendering it or its input field only in a read-only state.
- **Date and time ranges**—Within the DateTimePicker, date and time ranges can be defined by setting a start and end date for a period of time.
- **Disabled dates**—The Angular DateTimePicker supports specific approaches for disabling a selection of dates such as through using a function, an array of dates, or an array of days.
- **Formats**—You can display the DateTimePicker in its single format at all times or configure it to show the value in different formats when the input is focused or blurred.
- **Placeholders**—The DateTimePicker provides options for setting its input field and render a text hint, a floating label, or descriptions for its format sections.
- **Incremental steps**—The DateTimePicker enables you to change the default step for increasing and decreasing the parts of its date values.
- **Popup options**—The Angular DateTimePicker provides a number of options for customizing the popup, toggling its Cancel button, and more.
- **Templates**—You can customize the content of each Calendar cell within the DateTimePicker and define a cell template.
- **Forms support**—You can use the DateTimePicker both in template-driven and reactive Angular forms.
- **Integration with JSON**—As the DateTimePicker works only with `date` JavaScript instances while the received data from the server is serialized in a JSON format, the component provides options for binding it to dates which are serialized as strings.
- **Globalization**—All Kendo UI for Angular Date Inputs provide globalization options.
- **Accessibility**—The Angular DateTimePicker is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The DateTimePicker supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [DateTimePicker Homepage](#)
- [Getting Started with the Kendo UI for Angular Date Inputs](#)
- [API Reference of the DateTimePicker](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [DateTimePicker Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular DateInputs

This guide provides the information you need to start using the Kendo UI for Angular DateInputs—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#).

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_DATEINPUTS } from "@progress/kendo-angular-dateinputs";
import { KENDO_LABEL } from "@progress/kendo-angular-label";

@Component({
  standalone: true,
  imports: [KENDO_DATEINPUTS, KENDO_LABEL],
  selector: "my-app",
```

```
template: `<kendo-label  
    class="k-display-block"  
    [for]="datepicker"  
    text="Select a date:"  
></kendo-label>  
<kendo-datepicker #datepicker [style.width.px]="170"></kendo-  
datepicker>  
,  
}  
export class AppComponent {}
```

Collanse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular DateInputs package:

1. Run the following command.

```
ng add @progress/kendo-angular-dateinputs
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-dateinputs` package as a dependency to the `package.json` file.
- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the required components, directives and utility arrays:

The utility arrays are available starting from **v16.6.0**. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

- To add all components from the `DateInputs` package, import the `KENDO_DATEINPUTS` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_DATEINPUTS } from '@progress/kendo-angular-
dateinputs';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_DATEINPUTS]
})
```

TS

- To add individual `DateInputs` components, import the corresponding utility arrays in your standalone component. See the list of [available utility arrays](#).

For example if you only need the `Calendar` component, import `KENDO_CALENDAR`.

```
import { Component } from '@angular/core';
import { KENDO_CALENDAR } from '@progress/kendo-angular-
dateinputs';

@Component({
  standalone: true,
```

TS

```
        selector: 'my-app',
        imports: [KENDO_CALENDAR]
    })
```

# Using the Components

1. After successfully installing the `DateInputs` package and importing the required components and directives, add the corresponding tags of the components you need in the `app.component.html`. For example, if you need the Calendar, add the following code:

```
<kendo-calendar type="infinite"></kendo-calendar>
```

HTML

2. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

3. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular DropDownList component on the page.

# Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

# Next Steps

- [DateInputs Dependencies & Standalone Utilities](#)

- [Calendar Overview](#)
- [DateInput Overview](#)
- [DatePicker Overview](#)
- [DateRange Overview](#)
- [DateTimePicker Overview](#)
- [MultiViewCalendar Overview](#)
- [TimePicker Overview](#)
- [Globalization](#)
- [DateInputs API Documentation](#)

# Learning Resources

- [DateInputs Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Kendo UI for Angular MultiViewCalendar Overview

The Kendo UI for Angular MultiViewCalendar is an Angular form control that represents a graphical Gregorian calendar with multiple horizontal views.

It supports the selection of and navigation between dates as well as data templates and ranges for scheduling appointments.

The following example demonstrates the MultiViewCalendar in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_DATEINPUTS } from "@progress/kendo-angular-dateinputs";
import { KENDO_LABEL } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_LABEL, KENDO_DATEINPUTS],
  styles: [".k-calendar { margin: 0 auto; }"],
  template: `
    <kendo-label
      class="k-display-block"
      [for]="multiviewcalendar"
      text="Select a date:"
    ></kendo-label>
    <kendo-multiviewcalendar #multiviewcalendar></kendo-
    multiviewcalendar>
  `,
})
export class AppComponent {}
```

 Collapsse Code 

# Key Features

- **Disabled MultiViewCalendar**—You can use the configuration options of the MultiViewCalendar to disable the component so that users are not able to interact with it.
- **Focused and selected dates**—You can also control the focused and selected dates within the MultiViewCalendar.
- **Disabled dates**—The MultiViewCalendar supports specific approaches for disabling a selection of dates such as through using a function, an array of dates, or an array of days.
- **Active view**—You can control the initially loaded page of the Calendar and render it in a month, year, decade, or year view.
- **Multiple views**—You can also control the number of the horizontally rendered MultiViewCalendar views.
- **View header**—The MultiViewCalendar provides options for rendering a header for each view.
- **View selection depth**—The MultiViewCalendar provides an option for setting the view depth to which the user can navigate.
- **Navigation animation**—You can enable the navigation animation for the MultiViewCalendar.
- **Selection modes**—Apart from its default single selection mode, the MultiViewCalendar supports options for `multiple`, `range`, and `week` selection.
- **Templates**—You can customize the content of each Calendar cell and define a cell template.
- **Week number column**—The Calendar also supports options for rendering a column displaying the number of the weeks.
- **Forms support**—You can use the Calendar both in template-driven and reactive Angular forms.
- **Integration with JSON**—As the Calendar works only with `date` JavaScript instances while the received data from the server is serialized in a JSON format, the component provides options for binding it to dates which are serialized as strings.
- **Globalization**—All Kendo UI for Angular Date Inputs provide globalization options.
- **Accessibility**—The Calendar is accessible for screen readers and supports WAI-ARIA attributes.

- [Keyboard navigation](#)—The Calendar supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [MultiViewCalendar Homepage](#)
- [Getting Started with the Kendo UI for Angular Date Inputs](#)
- [API Reference of the MultiViewCalendar](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [MultiViewCalendar Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Angular Date Inputs Overview

The Kendo UI for Angular Date Inputs are components which provide user-friendly interface for selecting dates and times when scheduling appointments.

The Date Inputs are built from the ground up and specifically for Angular, so that you get high-performance date-input controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



## Calendar

A component for date selection and navigation



## DateInput

An input field which recognizes and formats values as dates



## DatePicker

A component which allows the fast selection of date values



## DateRange

A set of directives and components which provide date range selection



## DateRange

A component which provides date range selection



## DateTimePicker

A component which allows the fast selection of date and time values



## MultiView Calendar

A component with multiple month views which provides date selection and navigation



## TimePicker

A component for selecting time values from a time-list

# Angular Date Inputs Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import { KENDO_DATEINPUTS } from "@progress/kendo-angular-dateinputs";
import { KENDO_LABELS } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_LABELS, KENDO_DATEINPUTS],
  styles: [
```

```
.k-calendar {
    margin: 0 auto;
}

kendo-label {
    font-weight: bold;
}

kendo-dateinput,
kendo-datepicker,
kendo-timepicker,
kendo-datetimepicker {
    width: 175px;
}
],
template:
<div class="row example-wrapper" style="min-height: 450px;">
    <div class="col-xs-12 col-md-6 example-col">
        <kendo-label
            class="k-display-block"
            [for]="dateinput"
            text="DateInput"
        ></kendo-label>
        <kendo-dateinput #dateinput [value]="value"></kendo-dateinput>
        <p>
            (use <code>↔</code> and <code>→</code> to navigate,
<code>↑</code> and
            <code>↓</code> to update)
        </p>

        <kendo-label
            class="k-display-block"
            [for]="datepicker"
            text="DatePicker"
        ></kendo-label>
        <kendo-datepicker #datepicker [value]="value"></kendo-
datepicker>
        <p>(use <code>Alt</code>+<code>↓</code> to open the Calendar)
    </p>

        <kendo-label
            class="k-display-block"
            [for]="timepicker"
            text="TimePicker"
        ></kendo-label>
        <kendo-timepicker #timepicker [value]="value"></kendo-
timepicker>
        <p>
            (use <code>Alt+↓</code> to open the time list, Tab to move to
```

the next time section in the popup, `↑` to increment and `↓` to decrement the value)

</p>

<kendo-label  
  class="k-display-block"  
  [for]="datetimepicker"  
  text="DateTimePicker"  
></kendo-label>  
<kendo-datetimepicker #datetimepicker [value]="value">  
</kendo-datetimepicker>  
<p>  
  (use `Alt+↓` to open the popup, `Alt+←`, `→`, `↑`, `↓` to switch between the tabs when open)  
</p>

and

<p><b>DateRange</b></p>  
<kendo-daterange>  
  <kendo-floatinglabel text="Start">  
    <kendo-dateinput  
      kendoDateRangeStartInput  
      [(value)]="range.start"  
    ></kendo-dateinput>  
  </kendo-floatinglabel>  
  <kendo-floatinglabel text="End">  
    <kendo-dateinput  
      kendoDateRangeEndInput  
      [(value)]="range.end"  
    ></kendo-dateinput>  
  </kendo-floatinglabel>  
</kendo-daterange>  
<p>  
  (use `←`, `→`, `↑`, `↓`, `ENTER` to update)  
<code>↑</code> to  
  navigate, and <code>ENTER</code> to update)  
</p>  
</div>  
<div class="col-xs-12 col-md-6 example-col">  
  <kendo-label  
    class="k-display-block"  
    [for]="calendar"  
    text="Calendar"  
></kendo-label>  
  <kendo-calendar #calendar [value]="value"></kendo-calendar>  
  <br />  
  <br />  
  <kendo-label  
    class="k-display-block"

```
[for]="multiviewcalendar"
  text="MultiViewCalendar"
></kendo-label>
<kendo-multiviewcalendar
  #multiviewcalendar
  [value]="value"
></kendo-multiviewcalendar>
</div>
</div>
` ,
})
export class AppComponent {
  public value: Date = new Date();
  public range = { start: null, end: null };
}
```

Collapse Code ^

# Angular Date Inputs Key Features

Each Kendo UI for Angular Date Inputs component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Date Inputs library as well as develop new features and controls to it.

## Forms Support

All Date Inputs provide support for both asynchronous template-driven Angular forms and the predominantly synchronous reactive Angular forms. This feature allows you to draw on the logic set either in the template, or in the component's Typescript code. [Read more about the forms support of the Calendar...](#)

## Date and Time Ranges

Except for the MultiViewCalendar, all Date Inputs provide configuration options for setting a start and end date or time and, in this way, defining date or time ranges. [Read more about the date ranges in the Calendar...](#)

# Disabled Date Inputs

You can render the Date Inputs in a disabled state so that users cannot interact with them. [Read more about the Disabled Calendar...](#)

## Selection Modes

The Date Inputs provide a number of selection modes which allow users to select a single date, a range of dates, or multiple dates. [Read more about the selection modes of the Calendar...](#)

## Formats

The input-based Date Inputs let you to configure a desired date format for their displayed value. This feature also allows you to display different formats for the date value depending on whether the component is currently focused or blurred. [Read more about the formats of the DatePicker...](#)

## Placeholders

The Date Inputs conveniently enable you to render a text hint inside the their input field to give the users an indication of what type of value they should provide and in what format. [Read more about the placeholders of the DatePicker...](#)

## Typing

You can control the user typing experience of the input-based Date Inputs by configuring the available typing options depending on your specific requirements—for example, displaying a blinking caret, auto-correcting the date segments, etc. [Read more about the typing options of the DatePicker...](#)

## View Options

The calendar-based Date Inputs provide various view options for configuring the type of the calendar, the initially displayed view, the currently focused date, and the view depth, to which the user can navigate. [Read more about the view options of the DatePicker...](#)

## Templates

You can visually customize the content and the overall appearance of the calendar-based Date Inputs by utilizing ready-to-use templates and applying them to the individual cells of the components—for example, to the month, year, decade, or century calendar cells. [Read more about the templates used by the Calendar...](#)

## Integration with JSON

Since the Date Inputs components only work with JavaScript **Date** instances, and the received data from the server is serialized in a JSON format, the components provide options for binding them to dates that are serialized as strings. [Read more about the integration with JSON provided by the DatePicker...](#)

## Globalization

The Kendo UI for Angular Date Inputs support globalization to ensure that they can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Date Inputs support rendering in a right-to-left (RTL) direction. [Read more about Date Inputs globalization...](#)

## Accessibility

The Date Inputs are accessible for screen readers and support WAI-ARIA attributes. [Read more about accessibility support of the Calendar...](#)

## Keyboard Navigation

The Date Inputs support a number of keyboard shortcuts which allow users to accomplish various commands. [Read more about the keyboard navigation of the Calendar...](#)

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular Date Inputs are part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library —no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of Kendo UI for Angular Date Inputs, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

# Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Date Inputs](#)
- [API Reference of the Date Inputs](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular TimePicker Overview

The Kendo UI for Angular TimePicker represents a time-list where the user can enter or pick time values.

The following example demonstrates the Angular TimePicker in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_DATEINPUTS } from "@progress/kendo-angular-dateinputs";
import { KENDO_ICONS } from "@progress/kendo-angular-icons";
import { KENDO_LABEL } from "@progress/kendo-angular-label";
import { imageIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_LABEL, KENDO_BUTTONS, KENDO_DATEINPUTS, KENDO_ICONS],
  styleUrls: ["./styles.css"],
  styles: [
    `
      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="visit-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column">
            <h4 class="k-h4">Schedule Your Visit</h4>
            <div class="component-container">
              <kendo-label text="Time">
                <kendo-timepicker [(value)]="value"></kendo-timepicker>
              </kendo-label>
            </div>
          </div>
        </div>
      </div>
    </div>
  `
})
```

```

        </kendo-label>
    </div>
    <div class="skeleton-container top">
        <div class="k-skeleton skeleton-box-small"></div>
        <div class="k-skeleton skeleton-box-large"></div>
    </div>
    <div class="skeleton-container bottom">
        <div class="k-skeleton skeleton-box-medium"></div>
        <div class="k-skeleton skeleton-box-medium"></div>
    </div>
</div>
<div class="card-column image-container">
    <div class="k-skeleton skeleton-image">
        <kendo-svgicon size="xxxlarge" [icon]="imageSVG"></kendo-
svgicon>
    </div>
</div>
</div>
<div class="card-row">
    <div class="k-skeleton skeleton-box-half"></div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
    public value: Date;
    public imageSVG: SVGIcon = imageIcon;
}

```

[Collapse Code ^](#)

# Key Features

- **Disabled TimePicker**—You can use the configuration options of the Angular TimePicker to disable the component so that users are not able to interact with it.
- **Read-only TimePicker**—The TimePicker provides a configuration option for rendering it in its read-only state.
- **Time ranges**—Within the Angular TimePicker, time ranges can be defined by setting a start and end time value.
- **Incremental steps**—The TimePicker enables you to change the default step for increasing and decreasing the parts of its time values.

- **Formats**—You can control the format of the TimePicker by using its `format` property.
- **Placeholders**—The TimePicker provides options for setting its input field and render a text hint or descriptions for its format sections.
- **Incomplete date validation**—The incomplete date validation feature of the TimePicker ensures that users do not accidentally leave a non-required field partially populated.
- **Forms Support**—You can use the Angular TimePicker both in template-driven and reactive Angular forms.
- **Integration with JSON**—As the TimePicker works only with `date` JavaScript instances while the received data from the server is serialized in a JSON format, the component provides options for binding it to dates which are serialized as strings.
- **Globalization**—All Kendo UI for Angular Date Inputs provide globalization options.
- **Accessibility**—The TimePicker is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The Angular TimePicker supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [TimePicker Homepage](#)
- [Getting Started with the Kendo UI for Angular Date Inputs](#)
- [API Reference of the TimePicker](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [TimePicker Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)

- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Angular Date Math Overview

The Kendo UI Date Math is a package for JavaScript Date manipulations.

The package exports numerous functions that support the performance of different date manipulation tasks. For example, adding and removing days or getting the week number.

## Angular Date Math Example

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import {
  addDays,
  addWeeks,
  addMonths,
  addYears,
  addDecades,
  addCenturies,
} from "@progress/kendo-date-math";

@Component({
  selector: "my-app",
  template: `
    <div class="row example-config">
      <div class="col-xs-12 col-md-3 example-col">
        <p>Start Date</p>
        <kendo-datepicker
          id="startDate"
          [(value)]="value"
          (valueChange)="calc()"
        ></kendo-datepicker>
      </div>

      <div class="col-xs-12 col-md-3 example-col">
        <p>Operation</p>
```

```
<kendo-buttongroup selection="single">
  <button
    kendoButton
    [toggleable]="true"
    (click)="setOp('add')"
    [selected]="true"
  >
    Add
  </button>
  <button kendoButton [toggleable]="true"
  (click)="setOp('subtract')">
    Subtract
  </button>
</kendo-buttongroup>
</div>

<div class="col-xs-12 col-md-2 example-col">
  <p>Days</p>
  <kendo-numerictextbox
    id="days"
    [(ngModel)]="days"
    (valueChange)="calc()"
    format="N0"
    style="width: 80px;">
  </kendo-numerictextbox>
</div>
<div class="col-xs-12 col-md-2 example-col">
  <p>Weeks</p>
  <kendo-numerictextbox
    id="weeks"
    [(ngModel)]="weeks"
    (valueChange)="calc()"
    format="N0"
    style="width: 80px;">
  </kendo-numerictextbox>
</div>
<div class="col-xs-12 col-md-2 example-col">
  <p>Months</p>
  <kendo-numerictextbox
    id="months"
    [(ngModel)]="months"
    (valueChange)="calc()"
    format="N0"
    style="width: 80px;">
  </kendo-numerictextbox>
</div>
<div class="col-xs-12 col-md-2 example-col">
  <p>Years</p>
```

```
<kendo-numerictextbox
  id="years"
  [(ngModel)]="years"
  (valueChange)="calc()"
  format="N0"
  style="width: 80px;">
</kendo-numerictextbox>
</div>
<div class="col-xs-12 col-md-2 example-col">
  <p>Decades</p>
  <kendo-numerictextbox
    id="decades"
    [(ngModel)]="decades"
    (valueChange)="calc()"
    format="N0"
    style="width: 80px;">
</kendo-numerictextbox>
</div>
<div class="col-xs-12 col-md-2 example-col">
  <p>Centuries</p>
  <kendo-numerictextbox
    id="centuries"
    [(ngModel)]="centuries"
    (valueChange)="calc()"
    format="N0"
    style="width: 80px;">
</kendo-numerictextbox>
</div>
</div>
<div>The new date is {{ result | kendoDate }}</div>
``,
})
export class AppComponent {
  public value: Date = new Date();
  public op: string = "add";
  public days: number = 1;
  public weeks: number = 0;
  public months: number = 0;
  public years: number = 0;
  public decades: number = 0;
  public centuries: number = 0;
  public result: Date;

  constructor() {
    this.calc();
  }

  public calc(): void {
```

```
const sign = this.op === "add" ? 1 : -1;
let value = this.value;

value = addDays(value, sign * this.days);
value = addWeeks(value, sign * this.weeks);
value = addMonths(value, sign * this.months);
value = addYears(value, sign * this.years);
value = addDecades(value, sign * this.decades);
value = addCenturies(value, sign * this.centuries);

this.result = value;
}

public setOp(value: string): void {
    this.op = value;
    this.calc();
}
}
```

Collapse Code ^

# Date Math Key Features

The Kendo UI Date Math package delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI team constantly invests efforts to improve the performance and add more value to the existing Date Math library as well as develop new features to it.

## Date Calculations

The Date Math package provides options for adding and removing specific time periods (for example, days and weeks), getting the first or last time periods (for example, the first date in a month), and also creating and comparing dates. [Read more about the available date-calculating methods of the Date Math...](#)

## Timezones

The Date Math package includes a timezone database and utilities for working with dates in different timezones. You can create a date in any timezone and perform

calculations or convert between timezones without creating intermediate local dates. [Read more about the timezone adjustments methods of the Date Math...](#)

# Licensing

The Kendo UI Date Math package is part of all available Kendo UI libraries. Depending on the trial version and commercial license support that each Kendo UI suite or flavor offers, the Date Math package may be available for trial and commercial users, and as part of the open-source Kendo UI for jQuery Core suite.

# Support Options

For any questions about the use of the Kendo UI Date Math, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI feedback portal](#) and Roadmaps provide information on the features in discussion and also the planned ones for release.
  - [Angular roadmap](#)
  - [React roadmap](#)
  - [Vue roadmap](#)
- Kendo UI uses GitHub Issues as its bug tracker and you can submit any related reports there. Also, check out the closed list.
  - [Angular tracker](#)
  - [React tracker](#)
  - [Vue tracker](#)

- Of course, the Kendo UI team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Date Math Homepage](#)
- [Getting Started with the Kendo UI Date Math](#)
- [API Reference of the Date Math](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Virtual Classroom \(Training Courses for Registered Users\)](#)
- [Date Math Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Dialog Overview

The Kendo UI for Angular Dialog communicates specific information and prompts users to take certain actions by interacting with an Angular modal component.

The following example demonstrates the Dialog—a feature-rich Angular modal component.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";

import { KENDO_DIALOGS } from "@progress/kendo-angular-dialog";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { FormsModule } from "@angular/forms";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_DIALOGS, KENDO_BUTTONS, KENDO_INPUTS, FormsModule],
  template: `
    @if (!opened) {
      <button kendoButton (click)="open()">Open dialog</button>
    } @if (opened) {
      <kendo-dialog
        title="Please confirm"
        (close)="close('cancel')"
        [minWidth]="250"
        [width]="450"
      >
        <p style="margin: 30px; text-align: center;">
          Are you sure you want to continue?
        </p>
        <kendo-dialog-actions>
          <button kendoButton (click)="close('no')">No</button>
          <button kendoButton (click)="close('yes')" themeColor="primary">Yes</button>
        </kendo-dialog-actions>
      </kendo-dialog>
    }
  `
})
```

```
        Yes
      </button>
    </kendo-dialog-actions>
  </kendo-dialog>
}
`,
})
export class AppComponent {
  public opened = true;

  public close(status: string): void {
    console.log(`Dialog result: ${status}`);
    this.opened = false;
  }

  public open(): void {
    this.opened = true;
  }
}
```

Collapse Code ^

# Key Features

- **Title**—The Dialog provides configuration options for displaying a title as well as customizing the content of that title.
- **Visibility**—Like other Angular modal components, the Dialog is visible by default, but allows you to control this functionality.
- **Initial focus**—The Dialog enables you to set different elements as focused upon its loading.
- **Action buttons**—To further customize the Dialog, you can use the available approaches for displaying its action buttons.
- **Dimensions**—To control the size of the Dialog, the component provides specific configuration options for setting its height and width.
- **Angular service**—You can also create the Dialog dynamically by using the Angular service integration of the component.
- **Globalization**—All Kendo UI for Angular Dialogs provide globalization options.
- **Accessibility**—The Dialog is accessible for screen readers and supports WAI-ARIA attributes.

- [Keyboard navigation](#)—The Dialog supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [Dialog Homepage](#)
- [Getting Started with the Kendo UI for Angular Dialogs](#)
- [API Reference of the Dialog](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Dialog Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Dialogs

This guide provides the information you need to start using the Kendo UI for Angular Dialogs—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#).

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_DIALOGS } from "@progress/kendo-angular-dialog";
import { KENDO_BUTTON } from "@progress/kendo-angular-buttons";

@Component({
  standalone: true,
  imports: [KENDO_DIALOGS, KENDO_BUTTON],
  selector: "my-app",
  template: `
    @if (!opened) {
```

```
<button kendoButton (click)="open()">Open dialog</button>
} @if (opened) {
<kendo-dialog title="Please confirm" (close)="close()">
  <p>Are you sure you want to continue?</p>
  <kendo-dialog-actions>
    <button kendoButton (click)="close()">No</button>
    <button kendoButton (click)="close()" themeColor="primary">Yes</button>
  </kendo-dialog-actions>
</kendo-dialog>
}
,
})
export class AppComponent {
  public opened = true;

  public close(): void {
    this.opened = false;
  }

  public open(): void {
    this.opened = true;
  }
}
```

Collanse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

# Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Dialogs package:

## 1. Run the following command.

```
ng add @progress/kendo-angular-dialog
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-dialog` package as a dependency to the `package.json` file.
- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the required components, directives and utility arrays:

The utility arrays are available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

- To add all components from the Dialogs package, import the `KENDO_DIALOGS` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_DIALOGS } from '@progress/kendo-angular-dialog';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_DIALOGS]
})
```

TS

- To add individual Dialogs components, import the corresponding utility arrays in your standalone component. See the list of [available utility](#)

arrays.

For example if you only need the Dialog component, import KENDO\_DIALOG.

```
import { Component } from '@angular/core';
import { KENDO_DIALOG } from '@progress/kendo-angular-dialog';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_DIALOG]
})
```

TS

## Using the Components

- After successfully installing the Dialogs package and importing the required components and directives, add the corresponding tags of the components you need in the `app.component.html`. For example, if you need the Dialog, add the following code:

```
<kendo-dialog title="Please confirm" >
  <p>Are you sure you want to continue?</p>
  <kendo-dialog-actions>
    <button kendoButton (click)="close()">No</button>
    <button kendoButton (click)="close()"
themeColor="primary">Yes</button>
  </kendo-dialog-actions>
</kendo-dialog>
```

HTML

- Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

3. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular Dialog component on the page.

# Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

## Next Steps

- [Dialogs Dependencies & Standalone Utilities](#)
- [Dialog Overview](#)
- [Window Overview](#)
- [Globalization](#)
- [Dialogs API Documentation](#)

## Learning Resources

- [Dialogs Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Angular Dialogs Overview

The Kendo UI for Angular Dialogs components communicate specific information to the users, and prompt them to take actions.

The Dialogs are built from the ground up and specifically for Angular, so that you get high-performance dialog controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



**Dialog**

A prompt for users to take specific actions by interacting with a modal dialog.



**Window**

A non-modal HTML window which provides information and can be moved and resized by users.

## Angular Dialogs Example

EXAMPLE

VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { DialogsComponent } from "./dialogs.component";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [DialogsComponent],
  template: `<my-dialogs></my-dialogs> `,
})
export class AppComponent {}
```

[Collapse Code ^](#)

# Angular Dialogs Key Features

Each Kendo UI for Angular Dialogs component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Dialogs library as well as develop new features and controls to it.

## Action Buttons

The Dialogs support different approaches for rendering action buttons and also for customizing their content and appearance. [Read more about the action buttons of the Dialog...](#)

## Initial Focus

You can use the options provided by the Dialogs components to define which element will be focused upon their initial loading. [Read more about the initial element focus of the Dialog...](#)

## State Management

The Kendo UI for Angular Dialogs support various rendering states. The Dialog always displays an overlay element preventing the interaction with other elements on the page, while the Window supports maximized and minimized state. [Read more about the maximized and minimized state of the Window...](#)

## Sizing

All Kendo UI for Angular Dialogs allow you to customize their dimensions. By default, the components automatically calculate their height so they can fit the displayed content. Read more about the dimensions of the [Dialog](#) and [Window](#).

## Angular Service Support

All Kendo UI for Angular Dialogs enable you to create their instances dynamically avoiding, in this way, the necessity for defining the components in templates. [Read more about the service implementation in the Dialog...](#)

## Appearance

You can use the options provided by the Dialogs components to customize their appearance. Read more about the customizable appearance of the [Dialog](#) and [Window](#).

## Globalization

The Kendo UI for Angular Dialogs support globalization to ensure that each Dialogs component can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Dialogs support rendering in a right-to-left (RTL) direction. [Read more about Dialogs globalization...](#)

## Accessibility

The Dialogs are accessible for screen readers and support WAI-ARIA attributes. [Read more about accessibility support of the Dialog...](#)

## Keyboard Navigation

The Dialogs support a number of keyboard shortcuts which allow users to accomplish various commands. [Read more about the keyboard navigation of the Dialog...](#)

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular Dialogs are part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library —no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of Kendo UI for Angular Dialogs, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

# Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Dialogs](#)
- [API Reference of the Dialogs](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Window Overview

The Kendo UI for Angular Window displays content in a non-modal HTML window which can be moved and resized.

The following example demonstrates the Window in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮

```
import { Component } from "@angular/core";

import { KENDO_DIALOGS } from
"@progress/kendo-angular-dialog";
import { KENDO_BUTTONS } from
"@progress/kendo-angular-buttons";
import { KENDO_INPUTS } from
"@progress/kendo-angular-inputs";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_DIALOGS, KENDO_BUTTONS,
  KENDO_INPUTS],
  template: `
    <div class="example-wrapper">
      @if (!opened) {
        <button kendoButton
        (click)="open()">Open window</button>
      } @if (dataSaved) {
        <p>Data has been saved</p>
      } @if (opened) {
        <kendo-window
          title="Please provide additional
          data"
          (close)="close()"
          [minWidth]="250"
          [width]="450"
        ></kendo-window>
      }
    </div>
  
```

CONFIGURATOR

X

File

app.component.ts

```
>
  <form class="k-form">
    <fieldset>
      <legend>User Details</legend>
      <label class="k-form-field">
        <span>First Name</span>
        <kendo-textbox
placeholder="Your Name"></kendo-textbox>
      </label>
      <label class="k-form-field">
        <span>Last Name</span>
        <kendo-textbox
placeholder="Your Last Name"></kendo-textbox>
      </label>
    </fieldset>
    <div class="k-actions k-actions-end">
      <button kendoButton type="button"
(click)="close()">Cancel</button>
      <button
        kendoButton
        themeColor="primary"
        type="button"
        (click)="submit()">
        >
          Submit
        </button>
    </div>
  </form>
</kendo-window>
}
</div>
``,
})
export class AppComponent {
  public opened = true;
  public dataSaved = false;

  public close(): void {
    this.opened = false;
  }

  public open(): void {
    this.opened = true;
  }

  public submit(): void {
    this.dataSaved = true;
    this.close();
  }
}
```

# Key Features

- [Title](#)—The Window provides configuration options for displaying a title as well as customizing the content of that title.
- [Visibility](#)—Even though the Window is visible by default, you can still overwrite and control this functionality.
- [Initial focus](#)—The Window enables you to set different elements as focused upon its loading.
- [Action buttons](#)—To further customize the Window, you can use the available approaches for displaying its action buttons.
- [Dimensions and resizing](#)—To control the size of the Window, the component provides specific configuration options for setting its height and width as well as allowing users to resize it.
- [Positioning and dragging](#)—You can specify the position of the Window and also allow it to be moved by dragging.
- [Minimizing and maximizing](#)—The Window provides title bar commands which enable the user to minimize and maximize the component when needed.
- [Angular service](#)—You can also create the Window dynamically by using the Angular service integration of the component.
- [Globalization](#)—All Kendo UI for Angular Dialogs provide globalization options.
- [Accessibility](#)—The Window is accessible for screen readers and supports WAI-ARIA attributes.
- [Keyboard navigation](#)—The Window supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [Window Homepage](#)
- [Getting Started with the Kendo UI for Angular Dialogs](#)
- [API Reference of the Window](#)

- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Window Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular AutoComplete Overview

The Kendo UI for Angular AutoComplete is a form component that provides suggestions depending on the typed text.

It is a richer version of the `<input>` element and supports data binding, filtering, and templates.

The following example demonstrates the AutoComplete in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_DROPDOWNS } from "@progress/kendo-angular-dropdowns";
import { KENDO_ICONS } from "@progress/kendo-angular-icons";
import { KENDO_LABELS } from "@progress/kendo-angular-label";
import { ImageIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_DROPDOWNS, KENDO_LABELS, KENDO_ICONS],
  styleUrls: ["./skeleton-app.component.css"],
  styles: [
    `my-app {
      padding-bottom: 0;
    }
  `,
  ],
  template: `
    <div class="food-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column">
            <h4 class="k-h4">Time to order food</h4>
          </div>
        </div>
      </div>
    </div>
  `
})
```

```
<div class="component-container">
    <kendo-label text="Find restaurants in your area">
        <kendo-autocomplete
            [data]="areaList"
            placeholder="e.g. New York"
        >
        </kendo-autocomplete>
    </kendo-label>
</div>
<div class="skeleton-container top">
    <div class="k-skeleton skeleton-box-small"></div>
    <div class="k-skeleton skeleton-box-large"></div>
</div>
<div class="skeleton-container bottom">
    <div class="k-skeleton skeleton-box-medium"></div>
    <div class="k-skeleton skeleton-box-medium"></div>
</div>
</div>
<div class="card-column image-container">
    <div class="k-skeleton skeleton-image">
        <kendo-svgicon size="xxxlarge" [icon]="imageIcon"></kendo-
svgicon>
    </div>
</div>
</div>
<div class="card-row">
    <div class="k-skeleton skeleton-box-half"></div>
</div>
</div>
</div>
,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
    public areaList: Array<string> = [
        "Amsterdam",
        "Athens",
        "Barcelona",
        "Berlin",
        "Brussels",
        "Chicago",
        "Copenhagen",
        "Dublin",
        "Helsinki",
        "Houston",
        "Lisbon",
        "London",
        "Los Angeles",
        "Madrid",
        "Miami",
    ]
}
```

```
"Montreal",
"New York",
"Paris",
"Philadelphia",
"Prague",
"Rome",
"Sao Paulo",
"Seattle",
"Stockholm",
"Toronto",
"Vancouver",
"Vienna",
"Vienna",
"Warsaw",
];
public imageIcon: SVGIcon = imageIcon;
}
```

Collasne Code ^

# Key Features

- **Data binding**—You can bind the AutoComplete to a list of possible values containing arrays of primitive or complex data.
- **Value binding**—The AutoComplete enables you to set its value to arrays of primitive data.
- **Filtering**—Apart from its default filter functionality, the AutoComplete provides options for setting a minimum length of the search symbols and a built-in filter directive.
- **Grouping**—You can group AutoComplete items and display them as a grouped result.
- **Virtualization**—The AutoComplete supports a virtualization mechanism which noticeably improves the performance of the component when it handles large datasets.
- **Disabled items**—You can disable any of the AutoComplete items and prevent users from interacting with them.
- **Suggestions**—The AutoComplete renders suggestions which appear while the user types in the input field and also provides ways for controlling the behavior of the suggestion list such as preventing its opening and closing, and managing the focus of the displayed suggestion items.

- **Templates**—You can customize the content of the AutoComplete by using templates for its items, header, footer, and other elements.
- **Forms support**—You can use the AutoComplete both in template-driven and reactive Angular forms.
- **Globalization**—All Kendo UI for Angular Dropdowns provide globalization options.
- **Accessibility**—The AutoComplete is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The AutoComplete supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [AutoComplete Homepage](#)
- [Getting Started with the Kendo UI for Angular Dropdowns](#)
- [API Reference of the AutoComplete](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [AutoComplete Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular ComboBox Overview

The Kendo UI for Angular ComboBox is a form component that lets you choose from a list of options.

It is a richer version of the `<select>` element and supports data binding, filtering, templates, and the entering of custom values.

The following example demonstrates the Angular ComboBox in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { FormsModule } from "@angular/forms";
import { KENDO_DROPDOWNS } from "@progress/kendo-angular-dropdowns";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_LABELS } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [FormsModule, KENDO_DROPDOWNS, KENDO_LABELS, KENDO_INPUTS],
  styleUrls: ["./hobbies-skeleton-app.component.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="hobbies-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column no-flex">
            <div class="sidebar-container k-skeleton">

```

```
<div class="avatar-name-container">
  <div class="k-skeleton skeleton-avatar"></div>
  <div class="name-container">
    <div class="k-skeleton skeleton-text"></div>
    <div class="k-skeleton skeleton-small-text-short">
</div>
</div>
</div>
<div class="description-container">
  <div class="k-skeleton skeleton-small-text"></div>
  <div class="k-skeleton skeleton-small-text"></div>
  <div class="k-skeleton skeleton-small-text"></div>
</div>
</div>
</div>
<div class="card-column">
  <div class="avatar-title-container">
    <div class="k-skeleton skeleton-avatar"></div>
    <h4 class="k-h4">Hobbies</h4>
  </div>
  <div class="component-container">
    <kendo-formfield showHints="always">
      <kendo-label text="Favorite sport:">
        <kendo-combobox
          [data]="listItems"
          [allowCustom]="allowCustom"
          [(ngModel)]="selectedValues"
        >
        </kendo-combobox>
      </kendo-label>
      <kendo-formhint
        >Add your favourite sport, if it is not in the
        list.</kendo-formhint
      >
    </kendo-formfield>
  </div>
  <div class="skeleton-container top">
    <div class="k-skeleton skeleton-box-small"></div>
    <div class="k-skeleton skeleton-box-large"></div>
  </div>
  <div class="skeleton-container bottom">
    <div class="k-skeleton skeleton-box-small"></div>
    <div class="k-skeleton skeleton-box-large-double"></div>
  </div>
</div>
</div>
</div>
``,
  encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
  for demo purposes only.
```

```
)  
export class AppComponent {  
  public allowCustom = true;  
  public selectedValues: string = "Baseball";  
  public listItems: Array<string> = [  
    "Baseball",  
    "Basketball",  
    "Cricket",  
    "Field Hockey",  
    "Football",  
    "Table Tennis",  
    "Tennis",  
    "Volleyball",  
  ];  
}  
Collanse Code ^
```

# Key Features

- **Data binding**—You can bind the Angular ComboBox to a list of possible values containing arrays of primitive or complex data.
- **Value binding**—The ComboBox enables you to set its value to primitive or complex data.
- **Custom values**—You can override the default behavior of the ComboBox and configure it to accept custom values.
- **Filtering**—Apart from its default filter functionality, the Angular ComboBox provides options for performing server-side filtering, setting a minimum length of the search symbols, and using its built-in filter directive.
- **Grouping**—You can group ComboBox items and display them as a grouped result.
- **Virtualization**—The ComboBox supports a virtualization mechanism which noticeably improves the performance of the component when it handles large datasets.
- **Disabled items**—You can disable any of the ComboBox items and prevent users from interacting with them.
- **Suggestions**—The ComboBox renders suggestions which appear while the user types in the input field and also provides ways for controlling the behavior of the suggestion list such as preventing its opening and closing.
- **Templates**—You can customize the content of the ComboBox by using templates for its items, header, footer, and other elements.

- **Forms support**—You can use the Angular ComboBox both in template-driven and reactive Angular forms.
- **Cascading ComboBoxes**—The ComboBox enables you to render a series of two or more ComboBoxes where each ComboBox is filtered based on the selected option in the previous ComboBox.
- **Globalization**—All Kendo UI for Angular Dropdowns provide globalization options.
- **Accessibility**—The ComboBox is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The Angular ComboBox supports a number of keyboard shortcuts for processing various commands.

To learn more about the appearance, anatomy, and accessibility of the ComboBox, visit the [Progress Design System documentation](#)—an information portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

## Support and Learning Resources

- [ComboBox Homepage](#)
- [Getting Started with the Kendo UI for Angular Dropdowns](#)
- [API Reference of the ComboBox](#)
- [Angular ComboBoxList Features](#)
- ["How to Build Modern Angular Dropdowns in Minutes with Kendo UI"](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [ComboBox Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular DropDownList Overview

The Kendo UI for Angular DropDownList is a form component that lets you choose a single predefined value from a list.

It is a richer version of the `<select>` element and supports data binding, filtering, templates, and default items.

The following example demonstrates the Angular DropDownList in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_DROPDOWNS } from "@progress/kendo-angular-dropdowns";
import { KENDO_ICONS } from "@progress/kendo-angular-icons";
import { KENDO_LABELS } from "@progress/kendo-angular-label";
import { ImageIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_DROPDOWNS, KENDO_LABELS, KENDO_ICONS],
  styleUrls: ["./skeleton-app.component.css"],
  styles: [
    `my-app {
      padding-bottom: 0;
    }
  `,
  ],
  template: `
    <div class="food-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column">
            <h4 class="k-h4">Time to order food</h4>
          </div>
        </div>
      </div>
    </div>
  `
})
```

```
<div class="component-container">
    <kendo-label text="Find restaurants in your area">
        <kendo-dropdownlist
            [data]="areaList"
            defaultItem="Select city...">
        </kendo-dropdownlist>
    </kendo-label>
</div>
<div class="skeleton-container top">
    <div class="k-skeleton skeleton-box-small"></div>
    <div class="k-skeleton skeleton-box-large"></div>
</div>
<div class="skeleton-container bottom">
    <div class="k-skeleton skeleton-box-medium"></div>
    <div class="k-skeleton skeleton-box-medium"></div>
</div>
</div>
<div class="card-column image-container">
    <div class="k-skeleton skeleton-image">
        <kendo-svgicon size="xxxlarge" [icon]="imageIcon"></kendo-
svgicon>
    </div>
</div>
</div>
<div class="card-row">
    <div class="k-skeleton skeleton-box-half"></div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
    public areaList: Array<string> = [
        "Boston",
        "Chicago",
        "Houston",
        "Los Angeles",
        "Miami",
        "New York",
        "Philadelphia",
        "San Francisco",
        "Seattle",
    ];
    public imageIcon: SVGIcon = imageIcon;
}
```

Collanse Code ^

# Key Features

- **Data binding**—You can bind the Angular DropDownList to a list of possible values containing arrays of primitive or complex data.
- **Value binding**—The DropDownList enables you to set its value to primitive or complex data.
- **Filtering**—Apart from its default filter functionality, the DropDownList provides options for setting a minimum length of the search symbols and a built-in filter directive.
- **Grouping**—You can group DropDownList items and display them as a grouped result.
- **Virtualization**—The Angular DropDownList supports a virtualization mechanism which noticeably improves the performance of the component when it handles large datasets.
- **Default item**—You can configure the default item of the DropDownList by using primitive or complex values.
- **Disabled items**—You can disable any of the DropDownList items and prevent users from interacting with them.
- **Templates**—You can customize the content of the DropDownList by using templates for its value, header, footer, and other elements.
- **Forms support**—You can use the DropDownList both in template-driven and reactive Angular forms.
- **Adding new items**—The Angular DropDownList allows you to render a button for the user to add new items to the list.
- **Controlling the open state**—The DropDownList enables you to configure the initially opened item and prevent the opening and closing of its options list.
- **Cascading DropDownList**s—The DropDownList enables you to render a series of two or more DropDownList where each DropDownList is filtered based on the selected option in the previous DropDownList.
- **Globalization**—All Kendo UI for Angular Dropdowns provide globalization options.
- **Accessibility**—The DropDownList is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The DropDownList supports a number of keyboard shortcuts for processing various commands.

To learn more about the appearance, anatomy, and accessibility of the DropDownList, visit the [Progress Design System documentation](#)—an information portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

## Support and Learning Resources

- [DropDownList Homepage](#)
- [Getting Started with the Kendo UI for Angular Dropdowns](#)
- [API Reference of the DropDownList](#)
- [Angular DropDownList Features](#)
- ["How to Build Modern Angular Dropdowns in Minutes with Kendo UI"](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [DropDownList Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular DropDownTree Overview

The Kendo UI for Angular DropDownTree is a form component that renders data in a tree-like structure and lets you choose a single predefined value.

It is a richer version of the `<select>` element and supports data binding, filtering and templates.

The following example demonstrates the DropDownTree in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_DROPDOWNS } from "@progress/kendo-angular-dropdowns";
import { KENDO_ICONS } from "@progress/kendo-angular-icons";
import { KENDO_LABELS } from "@progress/kendo-angular-label";
import { ImageIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_DROPDOWNS, KENDO_LABELS, KENDO_ICONS],
  styleUrls: ["./skeleton-app.component.css"],
  styles: [
    `my-app {
      padding-bottom: 0;
    }
  `,
  ],
  template: `
    <div class="food-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column">
            <h4 class="k-h4">Time to order food</h4>
          </div>
        </div>
      </div>
    </div>
  `
})
```

```
<div class="component-container">
    <kendo-label text="Find restaurants in your area">
        <kendo-dropdowntree
            kendoDropDownTreeExpandable
            [kendoDropDownTreeHierarchyBinding]="areaData"
            textField="text"
            valueField="id"
            childrenField="areas"
        >
        </kendo-dropdowntree>
    </kendo-label>
</div>
<div class="skeleton-container top">
    <div class="k-skeleton skeleton-box-small"></div>
    <div class="k-skeleton skeleton-box-large"></div>
</div>
<div class="skeleton-container bottom">
    <div class="k-skeleton skeleton-box-medium"></div>
    <div class="k-skeleton skeleton-box-medium"></div>
</div>
<div class="card-column image-container">
    <div class="k-skeleton skeleton-image">
        <kendo-svgicon size="xxxlarge" [icon]="imageIcon"></kendo-
svgicon>
    </div>
</div>
</div>
<div class="card-row">
    <div class="k-skeleton skeleton-box-half"></div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
    public areaData: AreaData[] = [
        {
            text: "America",
            id: 1,
            areas: [
                { text: "Chicago", id: 4 },
                { text: "Los Angeles", id: 3 },
                { text: "New York", id: 2 },
                { text: "San Francisco", id: 5 },
            ],
        },
        {
            text: "Europe",

```

```

        id: 6,
        areas: [
            { text: "Amsterdam", id: 7 },
            { text: "Barcelona", id: 10 },
            { text: "London", id: 8 },
            { text: "Paris", id: 9 },
        ],
    },
];
public imageIcon: SVGIcon = imageIcon;
}
type AreaData = {
    text: string;
    id: number;
    areas: Area[];
};

type Area = {
    text: string;
    id: number;
};

```

[Collapse Code ^](#)

# Key Features

- **Data binding**—The DropDownTree enables you to work with various types of data and provides a number of directives which cover various data-binding scenarios.
- **Value binding**—The DropDownTree enables you to set its value to primitive or complex data.
- **Filtering**—Apart from its default filter functionality, the DropDownTree provides built-in filter directives, and options for auto-expanding its items while filtering and performing manual filtering.
- **Persisting the expanded state**—You can persist the expanded state of DropDownTree items when the popup TreeView is re-rendered.
- **Disabled items**—You can disable any of the DropDownTree items and prevent users from interacting with them.
- **Templates**—You can customize the content of the DropDownTree by using templates for its value, header, footer, and other elements.
- **Controlling the open state**—The DropDownTree enables you to configure the initially opened item and prevent the opening and closing of its options list.

- **Forms support**—You can use the DropDownTree both in template-driven and reactive Angular forms.
- **Globalization**—All Kendo UI for Angular Dropdowns provide globalization options.
- **Accessibility**—The DropDownTree is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The DropDownTree supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [DropDownTree Homepage](#)
- [Getting Started with the Kendo UI for Angular Dropdowns](#)
- [API Reference of the DropDownTree](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [DropDownTree Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Drop Downs

This guide provides the information you need to start using the Kendo UI for Angular Drop Downs—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#).

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_DROPDOWNS } from "@progress/kendo-angular-dropdowns";
import { KENDO_LABEL } from "@progress/kendo-angular-label";

@Component({
  standalone: true,
  imports: [KENDO_DROPDOWNS, KENDO_LABEL],
  selector: "my-app",
```

```
template: `
  <kendo-label
    class="k-display-block"
    [for]="dropdownlist"
    text="Choose a sport:"
  ></kendo-label>
  <kendo-dropdownlist #dropdownlist [data]="listItems"> </kendo-
dropdownlist>
`,
styles: [
  kendo-dropdownlist {
    width: 200px;
  }
],
})
export class AppComponent {
  public listItems: Array<string> = [
    "Baseball",
    "Basketball",
    "Cricket",
    "Field Hockey",
    "Football",
    "Table Tennis",
    "Tennis",
    "Volleyball",
  ];
}
```

Collapse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

# Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular DropDowns package:

### 1. Run the following command.

```
ng add @progress/kendo-angular-dropdowns
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-dropdowns` package as a dependency to the `package.json` file.
- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

### 2. Import the required components, directives and utility arrays:

The utility arrays are available starting from **v16.6.0**. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

- To add all components from the DropDowns package, import the `KENDO_DROPDOWNNS` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_DROPDOWNNS } from '@progress/kendo-angular-
dropdowns';

@Component({
  standalone: true,
  selector: 'my-app',
```

TS

```
        imports: [KENDO_DROPDOWNS]
    })
```

- To add individual DropDowns components, import the corresponding utility arrays in your standalone component. See the list of [available utility arrays](#).

For example if you only need the DropDownList component, import `KENDO_DROPDOWNLIST`.

```
import { Component } from '@angular/core';
import { KENDO_DROPDOWNLIST } from '@progress/kendo-angular-
dropdowns';

@Component({
    standalone: true,
    selector: 'my-app',
    imports: [KENDO_DROPDOWNLIST]
})
```

TS

## Using the Components

1. After successfully installing the DropDowns package and importing the required components and directives, add the corresponding tags of the components you need in the `app.component.html`. For example, if you need the DropDownList, add the following code:

```
<kendo-dropdownlist [data]="listItems"></kendo-dropdownlist>
```

HTML

2. Create the data source in your `app.component.ts` file:

```
export class AppComponent {
    public listItems: Array<string> = [
        "Item 1",
        "Item 2",
```

TS

```
        "Item 3"  
    ];  
}
```

### 3. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

### 4. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular DropDownList component on the page.

## Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

## Next Steps

- [DropDowns Dependencies & Standalone Utilities](#)
- [AutoComplete Overview](#)
- [ComboBox Overview](#)
- [DropDownList Overview](#)
- [DropDownTree Overview](#)
- [MultiColumnComboBox Overview](#)
- [MultiSelect Overview](#)
- [MultiSelectTree Overview](#)
- [Globalization](#)
- [DropDowns API Documentation](#)

# Learning Resources

- [DropDowns Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Kendo UI for Angular MultiColumnComboBox Overview

The Kendo UI for Angular MultiColumnComboBox is a form component that lets you choose from a table-structured list of options.

It is a richer version of the `<select>` element and supports data binding, filtering, templates, and the entering of custom values.

The following example demonstrates the MultiColumnComboBox in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_DROPDOWNS } from "@progress/kendo-angular-dropdowns";
import { KENDO_ICONS } from "@progress/kendo-angular-icons";
import { KENDO_LABELS } from "@progress/kendo-angular-label";
import { chartPieIcon, SVGIcon } from "@progress/kendo-svg-icons";
import { Contact, contacts } from "./contacts";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_DROPDOWNS, KENDO_LABELS, KENDO_ICONS],
  styleUrls: ["./orders-skeleton-app.component.css"],
  styles: [
    my-app {
      padding-bottom: 0;
    }
  ],
  template: `
    <!--Load the utils package for the UI elements:-->
    <link
```

```
    rel="stylesheet"
    href="https://unpkg.com/@progress/kendo-theme-
utils@dev/dist/all.css"
  />

  <div class="orders-demo card-container">
    <div class="k-card custom-card !k-flex-row">
      <div class="custom-card-header k-skeleton">
        <div
          class="card-actions-container k-pt-4 k-gap-2.5 k-flex-layout
k-align-items-center k-flex-col"
        >
          <div
            class="custom-card-header-action k-skeleton k-rounded-
full"
          ></div>
          <div
            class="custom-card-header-action k-skeleton k-rounded-
full"
          ></div>
        </div>
      </div>
      <div class="card-content k-p-8 k-w-full">
        <h4 class="k-h4">Order History</h4>
        <div class="card-columns k-gap-8 k-flex-layout">
          <div class="card-content-column k-flex-1">
            <kendo-label text="Contact person:">
              <kendo-multicolumncombobox
                [data]="contacts"
                [listHeight]="145"
                [popupSettings]="{ width: '22%' }"
                textField="name"
                valueField="id"
              >
                <kendo-combobox-column
                  field="name"
                  title="Contact Name"
                  [width]="200"
                >
                  <ng-template
                    kendoMultiColumnComboBoxColumnCellTemplate
                    let-dataItem
                  >
                    <img
                      class="contact-image"
                      [src]="getContactImageUrl(dataItem.id)"
                    />
                    <span>{{ dataItem.name }}</span>
                  </ng-template>
                </kendo-combobox-column>
                <kendo-combobox-column
                  field="image"
                  title="Contact Image"
                  [width]="200"
                >
                  <ng-template
                    kendoMultiColumnComboBoxColumnCellTemplate
                    let-dataItem
                  >
                    <img
                      class="contact-image"
                      [src]="getContactImageUrl(dataItem.id)"
                    />
                    <span>{{ dataItem.name }}</span>
                  </ng-template>
                </kendo-combobox-column>
              </kendo-multicolumncombobox>
            </kendo-label>
          </div>
        </div>
      </div>
    </div>
  </div>
```

```
        field="jobTitle"
        title="Title"
        [width]="200"
    >
</kendo-combobox-column>
<kendo-combobox-column
    field="company"
    title="Company"
    [width]="200"
>
</kendo-combobox-column>
<ng-template kendoMultiColumnComboBoxFooterTemplate>
    <strong> {{ contacts.length }} records in total
</strong>
    </ng-template>
</kendo-multicolumncombobox>
</kendo-label>
<div
    class="skeleton-wrapper k-d-flex k-flex-col k-align-items-start k-gap-1 k-flex-layout k-mt-5"
>
    <div class="k-skeleton k-rounded skeleton-text-small">
</div>
    <div
        class="k-skeleton k-rounded skeleton-text-medium k-w-full"
        ></div>
    </div>
    <div
        class="skeleton-wrapper k-d-flex k-flex-col k-align-items-start k-gap-1 k-mt-5 padding-top-5"
        >
        <div class="k-skeleton k-rounded skeleton-text-small">
</div>
        <div
            class="k-skeleton k-rounded skeleton-text-large k-w-full"
            ></div>
        </div>
    </div>
    <div class="card-content-column card-image-column k-flex-1">
        <div
            class="skeleton-wrapper k-h-full k-flex-col k-align-items-start k-gap-1"
            >
            <div
                class="k-skeleton k-h-full k-rounded skeleton-image-large !k-d-flex k-justify-content-center k-align-items-center"
                >
                    <kendo-svg-icon
                        size="xxlarge"

```

```

        [icon]="chartPieIcon"
      ></kendo-svg-icon>
    </div>
  </div>
</div>
</div>
</div>
</div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
  public contacts: Contact[] = contacts;
  public chartPieIcon: SVGIcon = chartPieIcon;

  public getContactImageUrl(contactId: string): string {
    return `assets/dropdowns/contacts/${contactId}.jpg`;
  }
}

```

[Collapse Code ▲](#)

# Key Features

- **Data binding**—You can bind the MultiColumnComboBox to a list of possible values containing arrays of complex data.
- **Value binding**—The MultiColumnComboBox enables you to set its value to primitive or complex data.
- **Columns**
- **Custom values**—You can override the default behavior of the MultiColumnComboBox and configure it to accept custom values.
- **Filtering**—Apart from its default filter functionality, the MultiColumnComboBox provides options for performing server-side filtering and using its built-in filter directive.
- **Grouping**—You can group MultiColumnComboBox items and display them as a grouped result.
- **Virtualization**—The MultiColumnComboBox supports a virtualization mechanism which noticeably improves the performance of the component when it handles large datasets.

- **Disabled items**—You can disable any of the MultiColumnComboBox items and prevent users from interacting with them.
- **Suggestions**—The MultiColumnComboBox renders suggestions which appear while the user types in the input field.
- **Controlling the open state**—The MultiColumnComboBox allows you to implement a manual toggle functionality for its options list and prevent its opening and closing.
- **Templates**—You can customize the content of the MultiColumnComboBox by using templates for its cells, header, footer, and other elements.
- **Forms support**—You can use the MultiColumnComboBox both in template-driven and reactive Angular forms.
- **Globalization**—All Kendo UI for Angular Dropdowns provide globalization options.
- **Accessibility**—The MultiColumnComboBox is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The MultiColumnComboBox supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [MultiColumnComboBox Homepage](#)
- [Getting Started with the Kendo UI for Angular Dropdowns](#)
- [API Reference of the MultiColumnComboBox](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [MultiColumnComboBox Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)

- Kendo UI for Angular Roadmap

# Kendo UI for Angular MultiSelect Overview

The Kendo UI for Angular MultiSelect is a form component that displays a list of options and allows for multiple selections from this list.

It is a richer version of the `<select>` element and supports item and tag templates, and configurable options for controlling the list behavior.

The following example demonstrates the MultiSelect in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { FormsModule } from "@angular/forms";
import { KENDO_DROPDOWNS } from "@progress/kendo-angular-dropdowns";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_LABELS } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [FormsModule, KENDO_DROPDOWNS, KENDO_LABELS, KENDO_INPUTS],
  styleUrls: ["./hobbies-skeleton-app.component.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="hobbies-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column no-flex">
            <div class="sidebar-container k-skeleton">

```

```
        <div class="avatar-name-container">
          <div class="k-skeleton skeleton-avatar"></div>
          <div class="name-container">
            <div class="k-skeleton skeleton-text"></div>
            <div class="k-skeleton skeleton-small-text-short">
</div>
        </div>
      </div>
      <div class="description-container">
        <div class="k-skeleton skeleton-small-text"></div>
        <div class="k-skeleton skeleton-small-text"></div>
        <div class="k-skeleton skeleton-small-text"></div>
      </div>
    </div>
</div>
<div class="card-column">
  <div class="avatar-title-container">
    <div class="k-skeleton skeleton-avatar"></div>
    <h4 class="k-h4">Hobbies</h4>
  </div>
  <div class="component-container">
    <kendo-formfield showHints="always">
      <kendo-label text="Favorite sport:">
        <kendo-multiselect
          [data]="listItems"
          [(ngModel)]="value"
        ></kendo-multiselect>
      </kendo-label>
      <kendo-formhint
        >Add your favourite sport, if it is not in the
        list.</kendo-formhint
      >
    </kendo-formfield>
  </div>
  <div class="skeleton-container top">
    <div class="k-skeleton skeleton-box-small"></div>
    <div class="k-skeleton skeleton-box-large"></div>
  </div>
  <div class="skeleton-container bottom">
    <div class="k-skeleton skeleton-box-small"></div>
    <div class="k-skeleton skeleton-box-large-double"></div>
  </div>
</div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
```

```
public listItems: Array<string> = [
    "Baseball",
    "Basketball",
    "Cricket",
    "Field Hockey",
    "Football",
    "Table Tennis",
    "Tennis",
    "Volleyball",
];
public value: any = ["Baseball"];
}
```

Collanse Code ^

# Key Features

- **Data binding**—You can bind the MultiSelect to a list of possible values containing arrays of primitive or complex data.
- **Value binding**—The MultiSelect enables you to set its value to primitive or complex data.
- **Custom values**—You can override the default behavior of the MultiSelect and configure it to accept custom values.
- **Checkboxes**—The MultiSelect allows you to implement checkboxes for managing its items selection.
- **Filtering**—Apart from its default filter functionality, the MultiSelect provides options for setting a minimum length of the search symbols and a built-in filter directive.
- **Grouping**—You can group MultiSelect items and display them as a grouped result.
- **Virtualization**—The MultiSelect supports a virtualization mechanism which noticeably improves the performance of the component when it handles large datasets.
- **Disabled items**—You can disable any of the MultiSelect items and prevent users from interacting with them.
- **Summary-tag mode**—The MultiSelect allows you to customize the display of its selected tags.
- **Templates**—You can customize the content of the MultiSelect by using templates for its tag, header, footer, and other elements.

- **Forms support**—You can use the MultiSelect both in template-driven and reactive Angular forms.
- **Controlling the open state**—The MultiSelect enables you to configure the initially opened item and prevent the opening and closing of its options list.
- **Globalization**—All Kendo UI for Angular Dropdowns provide globalization options.
- **Accessibility**—The MultiSelect is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The MultiSelect supports a number of keyboard shortcuts for processing various commands.

To learn more about the appearance, anatomy, and accessibility of the MultiSelect, visit the [Progress Design System documentation](#)—an information portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

## Support and Learning Resources

- [MultiSelect Homepage](#)
- [Getting Started with the Kendo UI for Angular Dropdowns](#)
- [API Reference of the MultiSelect](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [MultiSelect Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)



# Kendo UI for Angular MultiSelectTree Overview

The Kendo UI for Angular MultiSelectTree is a form component that renders data in a tree-like structure and allows for multiple selection.

It is a richer version of the `<select>` element and supports templates, and configurable options for controlling the component's behavior.

The following example demonstrates the MultiSelectTree in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { FormsModule } from "@angular/forms";
import { KENDO_DROPDOWNS } from "@progress/kendo-angular-dropdowns";
import { KENDO_ICONS } from "@progress/kendo-angular-icons";
import { KENDO_LABELS } from "@progress/kendo-angular-label";
import { chartPieIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [FormsModule, KENDO_DROPDOWNS, KENDO_LABELS, KENDO_ICONS],
  styleUrls: ["./orders-skeleton-app.component.css"],
  styles: [
    my-app {
      padding-bottom: 0;
    }
  ],
  template: `
    <!--Load the utils package for the UI elements:-->
    <link
      rel="stylesheet"
      href="https://unpkg.com/@progress/kendo-theme-
```

```
utils@dev/dist/all.css"
  />

  <div class="orders-demo card-container">
    <div class="k-card custom-card !k-flex-row">
      <div class="custom-card-header k-skeleton">
        <div
          class="card-actions-container k-pt-4 k-gap-2.5 k-flex-layout
k-align-items-center k-flex-col"
        >
          <div
            class="custom-card-header-action k-skeleton k-rounded-
full"
          ></div>
          <div
            class="custom-card-header-action k-skeleton k-rounded-
full"
          ></div>
        </div>
      </div>
      <div class="card-content k-p-8 k-w-full">
        <h4 class="k-h4">Order History</h4>
        <div class="card-columns k-gap-8 k-flex-layout">
          <div class="card-content-column k-flex-1">
            <kendo-label text="Contact person:">
              <kendo-multiselecttree
                kendoMultiSelectTreeExpandable
                [kendoMultiSelectTreeHierarchyBinding]="data"
                childrenField="employees"
                textField="text"
                valueField="id"
                [(ngModel)]="value"
                [tagMapper]="tagMapper"
                [expandedKeys]="['0']"
                class="furniture"
              >
                </kendo-multiselecttree>
            </kendo-label>
            <div
              class="skeleton-wrapper k-d-flex k-flex-col k-align-
items-start k-gap-1 k-flex-layout k-mt-5"
            >
              <div class="k-skeleton k-rounded skeleton-text-small">
</div>
              <div
                class="k-skeleton k-rounded skeleton-text-medium k-w-
full"
              ></div>
            </div>
            <div
              class="skeleton-wrapper k-d-flex k-flex-col k-align-
```

```
        items-start k-gap-1 k-mt-5 padding-top-5"
            >
                <div class="k-skeleton k-rounded skeleton-text-small">
</div>
                <div
                    class="k-skeleton k-rounded skeleton-text-large k-w-
full"
                    ></div>
                </div>
            </div>
            <div class="card-content-column card-image-column k-flex-1">
                <div
                    class="skeleton-wrapper k-h-full k-flex-col k-align-
items-start k-gap-1"
                    >
                        <div
                            class="k-skeleton k-h-full k-rounded skeleton-image-
large !k-d-flex k-justify-content-center k-align-items-center"
                            >
                                <kendo-svg-icon
                                    size="xxlarge"
                                    [icon]="chartPieIcon"
                                    ></kendo-svg-icon>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
    ,
    encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
    public value: { text: string; id: number }[] = [
        { text: "Ana Trujillo", id: 2 },
    ];
    public chartPieIcon: SVGIcon = chartPieIcon;

    public data: JobPosition[] = [
        {
            text: "Owner",
            id: 1,
            employees: [
                { text: "Ana Trujillo", id: 2 },
                { text: "Antonio Moreno", id: 3 },
                { text: "Martín Sommer", id: 4 },
            ],
        },
        {
    
```

```

        text: "Order Administrator",
        id: 5,
        employees: [
            { text: "Christina Berglund", id: 6 },
            { text: "Sven Ottlieb", id: 7 },
        ],
    },
    {
        text: "Sales Manager",
        id: 8,
        employees: [{ text: "Roland Mendel", id: 9 }],
    },
];

public tagMapper(tags: any[]): any[] {
    return tags.length < 3 ? tags : [tags];
}
}

type JobPosition = {
    text: string;
    id: number;
    employees: Employee[];
};

type Employee = {
    text: string;
    id: number;
};

```

[Collapse Code ▲](#)

# Key Features

- **Data binding**—The MultiSelectTree enables you to work with various types of data and provides a number of directives which cover various data-binding scenarios.
- **Value binding**—The MultiSelectTree enables you to set its value to primitive or complex data.
- **Filtering**—Apart from its default filter functionality, the MultiSelectTree provides built-in filter directives, and options for auto-expanding its items while filtering and performing manual filtering.
- **Checkboxes**—The MultiSelectTree allows you to implement checkboxes for managing its items selection.
- **Persisting the expanded state**—You can persist the expanded state of MultiSelectTree items when the popup TreeView is re-rendered.

- **Disabled items**—You can disable any of the MultiSelectTree items and prevent users from interacting with them.
- **Summary-tag mode**—The MultiSelectTree allows you to customize the display of its selected tags.
- **Templates**—You can customize the content of the MultiSelectTree by using templates for its items, header, footer, and other elements.
- **Controlling the open state**—The MultiSelectTree enables you to configure the initially opened item and prevent the opening and closing of its options list.
- **Forms support**—You can use the MultiSelectTree both in template-driven and reactive Angular forms.
- **Globalization**—All Kendo UI for Angular Dropdowns provide globalization options.
- **Accessibility**—The MultiSelectTree is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The MultiSelectTree supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [MultiSelectTree Homepage](#)
- [Getting Started with the Kendo UI for Angular Dropdowns](#)
- [API Reference of the MultiSelectTree](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [MultiSelectTree Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)

- Kendo UI for Angular Roadmap

# Angular Dropdowns Overview

The Kendo UI for Angular Dropdowns are components for displaying predefined lists of options.

They include a variety of drop-down types and styles that have extensive configuration options. This flexibility allows you to quickly and easily create the exact Dropdowns component you need to fit your specific requirements for functionality and appearance.

The Dropdowns are built from the ground up and specifically for Angular, so that you get high-performance drop-down controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



## AutoComplete

A list of suggestions for typed content.



## ComboBox

A list for picking predefined items or entering custom values.



## DropDownList

A predefined list of options for picking single values.



## DropDownTree

A predefined list rendered in a tree-like structure for single item selection.



## MultiColumnComboBox



## MultiSelect

A table-structured list for picking predefined items or entering custom values.

A predefined list of options for multiple item selection.



### MultiSelect

A predefined list rendered in a tree-like structure for multiple item selection.

# Angular Dropdowns Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import { KENDO_DROPDOWNS } from "@progress/kendo-angular-dropdowns";
import { KENDO_LABEL } from "@progress/kendo-angular-label";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { Employee, employees } from "./employees";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_DROPDOWNS, KENDO_LABEL, KENDO_INPUTS, KENDO_BUTTONS],
  template: `
    <div class="example-wrapper" style="min-height: 400px;">
      <div class="col-xs-12 col-sm-6 example-col">
        <kendo-label text="AutoComplete">
          <br />
          <kendo-autocomplete
            [data]="listItems"
            placeholder="Your favorite sport"
          >
          </kendo-autocomplete>
        </kendo-label>
      </div>
    </div>
```

```
<div class="col-xs-12 col-sm-6 example-col">
  <kendo-label text="ComboBox">
    <br />
    <kendo-combobox [data]="listItems" value="Basketball">
      </kendo-combobox>
    </kendo-label>
  </div>

<div class="col-xs-12 col-sm-6 example-col">
  <kendo-label text="DropDownList">
    <br />
    <kendo-dropdownlist [data]="listItems" value="Basketball">
      </kendo-dropdownlist>
    </kendo-label>
  </div>

<div class="col-xs-12 col-sm-6 example-col">
  <kendo-label text="DropDownTree">
    <br />
    <kendo-dropdowntree
      kendoDropDownTreeExpandable
      [kendoDropDownTreeHierarchyBinding]="treeItems"
      textField="text"
      valueField="id"
      childrenField="items"
      [value]="complexValue"
      [popupSettings]="{ width: '230px' }"
      [expandedKeys]=["'1']"
    ></kendo-dropdowntree>
  </kendo-label>
</div>

<div class="col-xs-12 col-sm-6 example-col">
  <kendo-label text="MultiColumnComboBox">
    <br />
    <kendo-multicolumncombobox
      [data]="gridData"
      textField="name"
      valueField="id"
      placeholder="Select an employee"
    >
      <kendo-combobox-column field="name" title="Name"
      [width]="200">
        </kendo-combobox-column>
      <kendo-combobox-column field="title" title="Title"
      [width]="200">
        </kendo-combobox-column>
      <kendo-combobox-column field="phone" title="Phone"
      [width]="200">
        </kendo-combobox-column>
    </kendo-multicolumncombobox>
```

```
        </kendo-label>
    </div>

    <div class="col-xs-12 col-sm-6 example-col">
        <kendo-label text="MultiSelect">
            <br />
            <kendo-multiselect
                [data]="listItems"
                [value]="value"
                placeholder="Your favorite sports"
            ></kendo-multiselect>
        </kendo-label>
    </div>

    <div class="col-xs-12 col-sm-6 example-col">
        <kendo-label text="MultiSelectTree">
            <br />
            <kendo-multiselecttree
                kendoMultiSelectTreeExpandable
                [kendoMultiSelectTreeHierarchyBinding]="treeItems"
                textField="text"
                valueField="id"
                childrenField="items"
                [expandedKeys]="['0']"
                [(value)]="complexArrayValue"
            >
            </kendo-multiselecttree>
        </kendo-label>
    </div>
</div>
,
styles: [
    kendo-autocomplete,
    kendo-combobox,
    kendo-dropdownlist,
    kendo-dropdowntree,
    kendo-multicolumncombobox {
        width: 210px;
    }
],
])
export class AppComponent {
    public listItems: Array<string> = [
        "Baseball",
        "Basketball",
        "Cricket",
        "Field Hockey",
        "Football",
        "Table Tennis",
    ]
}
```

```

    "Tennis",
    "Volleyball",
];

public gridData: Employee[] = employees;

public treeItems: any[] = [
{
  text: "Furniture",
  id: 1,
  items: [
    { text: "Tables & Chairs", id: 2 },
    { text: "Sofas", id: 3 },
    { text: "Occasional Furniture", id: 4 },
  ],
},
{
  text: "Decor",
  id: 5,
  items: [
    { text: "Bed Linen", id: 6 },
    { text: "Carpets", id: 7 },
  ],
},
];
}

public value = ["Basketball", "Cricket"];
public complexValue = { text: "Decor", id: 5 };
public complexArrayValue = [{ text: "Sofas", id: 3 }];
}

```

[Collapse Code ^](#)

## Angular Dropdowns Key Features

Each Kendo UI for Angular Dropdowns component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Dropdowns library as well as develop new features and controls to it.

## Data Binding

The Dropdowns support the binding of their data to lists of possible values which can contain primitive (strings and numbers) or complex (objects) items. [Read more](#)

about the data binding of the AutoComplete...

## Forms Support

All Dropdowns provide support both for the asynchronous template-driven Angular forms and the predominantly synchronous reactive Angular forms. This feature allows you to draw on the logic set either in the template, or in the component or typescript code. [Read more about the forms support of the AutoComplete...](#)

## Filtering

Apart from their built-in filter functionality, the Dropdowns provide further options for fine-tuning and customizing the mechanism such as setting a minimum length of the search symbols, using the built-in filter directive, and more. [Read more about filtering the AutoComplete items...](#)

## Virtualization

All Kendo UI for Angular Dropdowns that do not visualize hierarchical data in their popups provide a virtualization option which significantly improves the performance when working with large datasets. [Read more about the virtualization functionality in the AutoComplete...](#)

## Grouping

All Kendo UI for Angular Dropdowns that do not visualize hierarchical data in their popups provide a grouping option. [Read more about the grouping functionality in the AutoComplete...](#)

## Disabled Options

You can choose to disable some of the predefined list options of the Dropdowns so that, if need be present, users will not be able to interact with them. [Read more](#)

[about the disabled options in the AutoComplete...](#)

## Adaptive Mode

All Kendo UI for Angular Dropdowns support a mobile-friendly rendering mode of their popups. Enabling the adaptive mode allows the components to automatically fit to the current screen size. [Read more about the adaptive mode in the AutoComplete...](#)

## Appearance

Each Kendo UI for Angular Dropdowns component offers extensive customization options, allowing you to tailor the look and feel to your preferences (size, border radius, and fill modes) for a refined and cohesive design. [Read more about the customizable appearance of the AutoComplete...](#)

## Templates

You can visually customize the content and the general look and feel of the items and elements list by utilizing the ready-to-use header, footer, and other templates. [Read more about the templates used by the AutoComplete...](#)

## Globalization

The Kendo UI for Angular Dropdowns support globalization to ensure that they can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Dropdowns support rendering in a right-to-left (RTL) direction. [Read more about Dropdowns globalization...](#)

## Accessibility

The Dropdowns are accessible for screen readers and support WAI-ARIA attributes. [Read more about accessibility support of the AutoComplete...](#)

# Keyboard Navigation

The Dropdowns support a number of keyboard shortcuts which allow users to accomplish various commands. [Read more about the keyboard navigation of the AutoComplete...](#)

# Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Dropdowns](#)
- [API Reference of the Dropdowns](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI File Saver

This guide provides the information you need to start using the Kendo UI File Saver—it includes instructions about the installation approach, how to import the required methods, and links to additional resources.

## Installing the Package

To add the Kendo UI File Saver package, run the following command:

```
npm install --save @progress/kendo-file-saver
```

SH

## Using the Package

1. After successfully installing the File Saver, import the required individual functions from the package:

```
import { saveAs, encodeBase64 } from '@progress/kendo-file-saver';
```

SH

2. Then you can save a file on the client machine, by using the `saveAs` method, and pass a valid `data URI` and a file name:

```
const dataURI = "data:text/plain;base64," + encodeBase64("Hello World!");
```

JS

```
saveAs(dataURI, "test.txt");
```

# Dependencies

The Kendo UI File Saver package has no external dependencies.

# Next Steps

- [Implementing Server Proxies](#)

# Suggested Links

- [File Saver Overview](#)
- [File Saver API Index](#)

# Angular File Saver Overview

The File Saver component enables you to save files on the client machine.

The saving of files is done through the `saveAs` method. You can consume the package from TypeScript and JavaScript projects alike.

## Angular File Saver Example

```
import { saveAs, encodeBase64 } from '@progress/kendo-file-saver';

const dataURI = "data:text/plain;base64," + encodeBase64("Hello
World!");
saveAs(dataURI, "test.txt");
```

JS

## Licensing

The Kendo UI File Saver package is part of all available Kendo UI libraries. Depending on the trial version and commercial license support that each Kendo UI suite or flavor offers, the File Saver package may be available for trial and commercial users, and as part of the open-source Kendo UI for jQuery Core suite.

## Support Options

For any questions about the use of the Kendo UI File Saver, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI outstanding customer support delivered by the actual

developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).

- [Kendo UI forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI feedback portal](#) and Roadmaps provide information on the features in discussion and also the planned ones for release.
  - [Angular roadmap](#)
  - [React roadmap](#)
  - [Vue roadmap](#)
- Kendo UI uses GitHub Issues as its bug tracker and you can submit any related reports there. Also, check out the closed list.
  - [Angular tracker](#)
  - [React tracker](#)
  - [Vue tracker](#)
- Of course, the Kendo UI team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [File Saver Homepage](#)
- [Getting Started with the Kendo UI File Saver](#)
- [API Reference of the File Saver](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)

- [Virtual Classroom \(Training Courses for Registered Users\)](#)
- [File Saver Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Badge Overview

The Kendo UI for Angular Badge is a visual indicator for UI elements.

The Badge enables you to easily show statuses, notifications, and short messages in your application, and also provides additional contextual information for other elements on the page.

The following example demonstrates the Badge in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_INDICATORS } from "@progress/kendo-angular-indicators";
import { ContactsComponent } from "./contacts.component";
import { SocialAppsComponent } from "./social-apps.component";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [
    KENDO_BUTTONS,
    KENDO_INDICATORS,
    ContactsComponent,
    SocialAppsComponent,
  ],
  template: `
    <!--Load the font icons package:-->
    <link
      rel="stylesheet"
      href="https://unpkg.com/@progress/kendo-font-icons/dist/index.css"
    />

    <div class="example">
```

```
<div class="example-wrap">
  <team-contacts></team-contacts>
  <social-apps></social-apps>

  <div class="status">
    <button kendoButton>
      New Updates
      <kendo-badge rounded="medium" themeColor="info"></kendo-
badge>
    </button>

    <span class="k-icon k-font-icon k-i-bell notifications">
      <kendo-badge rounded="medium" themeColor="warning"></kendo-
badge>
    </span>
  </div>
</div>
</div>

,
encapsulation: ViewEncapsulation.None,
styles: [
  .example {
    display: flex;
    align-items: center;
    flex-direction: column;
  }
  .example .status {
    display: flex;
    justify-content: space-between;
    padding: 12px 14px;
  }
  .example-wrap {
    border: 1px solid #ccc;
    width: 230px;
  }
  .notifications {
    position: relative;
    align-self: center;
    font-size: 21px;
    color: #6c757d;
  }
],
})
export class AppComponent {}
```

Collapse Code ^

# Key Features

- [Badge container](#)—
- [Aligning and positioning](#)—You can align the Badge in relation to its parent container whether it is an HTML element or an Angular Component.
- [Appearance](#)—The Badge delivers ready-to-use, predefined sets of styling options for modifying its roundness, size, theme color, and more.
- [Globalization](#)—All Kendo UI for Angular Indicators provide globalization options.

# Support and Learning Resources

- [Badge Homepage](#)
- [Getting Started with the Kendo UI for Angular Indicators](#)
- [API Reference of the Badge](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Badge Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Indicators

This guide provides the information you need to start using the Kendo UI for Angular Indicators—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#).

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_INDICATORS } from "@progress/kendo-angular-indicators";

@Component({
  standalone: true,
  imports: [KENDO_INDICATORS],
  selector: "my-app",
  template: ` <kendo-loader> </kendo-loader> `,
})
```

```
})  
export class AppComponent {}
```

Collapse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Indicators package:

1. Run the following command.

```
ng add @progress/kendo-angular-indicators
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-indicators` package as a dependency to the `package.json` file.
- Add all required peer dependencies to the `package.json` file.
- Register the Kendo UI Default theme in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the required components, directives and utility arrays:

The utility arrays are available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

- To add all components from the Indicators package, import the `KENDO_INDICATORS` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_INDICATORS } from '@progress/kendo-angular-indicators';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_INDICATORS]
})
```

TS

- To add individual Indicators components, import the corresponding utility arrays in your standalone component. See the list of [available utility arrays](#).

For example if you only need the Loader component, import `KENDO_LOADER`.

```
import { Component } from '@angular/core';
import { KENDO_LOADER } from '@progress/kendo-angular-indicators';

@Component({
  standalone: true,
```

TS

```
        selector: 'my-app',
        imports: [KENDO_LOADER]
    })
```

# Using the Components

1. After successfully installing the Indicators package and importing the required components and directives, add the corresponding tags of the components you need in the `app.component.html`. For example, if you need the Loader, add the following code:

```
<kendo-loader> </kendo-loader>
```

HTML

2. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

3. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular Loader component on the page.

# Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

# Next Steps

- [Badge Overview](#)

- [Loader Overview](#)
- [Skeleton Overview](#)
- [Globalization](#)
- [Indicators API Documentation](#)

# Learning Resources

- [Indicators Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Kendo UI for Angular Loader Overview

The Kendo UI for Angular Loader is a visual indicator that shows an indeterminate wait time.

The Loader informs users about the status of ongoing processes, such as loading an application, submitting a form, saving updates, or fetching data.

The following example demonstrates the Loader in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { CommonModule } from "@angular/common";
import {
  KENDO_INDICATORS,
  LoaderType,
  LoaderThemeColor,
  LoaderSize,
} from "@progress/kendo-angular-indicators";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_INDICATORS],
  template: `
    <div class="example">
      <div class="wrap">
        @for (loader of loaders; track loader) {
          <div class="example-item">
            <div class="example-item-title">{{ loader.type | titlecase }}</div>
        </div>
        <div class="k-block">
          <kendo-loader
            [type]="loader.type"
            [themeColor]="loader.themeColor"
          ></kendo-loader>
        </div>
      </div>
    </div>
  `,
})
```

```
          [size]="loader.size"
        >
      </kendo-loader>
    </div>
  </div>
}
</div>
</div>
,
styles: [
  .wrap {
    display: flex;
    align-items: center;
    margin: 0 -10px;
  }
  .example-item {
    flex: 0 0 33%;
    padding: 10px;
  }
  .example-item-title {
    margin-bottom: 10px;
    text-align: center;
  }
  .k-block {
    display: flex;
    align-items: center;
    justify-content: center;
    min-height: 80px;
  }
],
),
export class AppComponent {
  public loaders = [
    {
      type: <LoaderType>"pulsing",
      themeColor: <LoaderThemeColor>"primary",
      size: <LoaderSize>"medium",
    },
    {
      type: <LoaderType>"infinite-spinner",
      themeColor: <LoaderThemeColor>"secondary",
      size: <LoaderSize>"medium",
    },
    {
      type: <LoaderType>"converging-spinner",
      themeColor: <LoaderThemeColor>"info",
      size: <LoaderSize>"medium",
    },
  ],
}
```

```
    ];
```

Collapse Code ^



# Key Features

- [Integration](#)—You can integrate the Loader in another component, such a button or a loading panel, as well as use it part of a more complex functionality.
- [Appearance](#)—The Loader delivers ready-to-use, predefined sets of styling options for modifying its animation type, size, and more.
- [Globalization](#)—All Kendo UI for Angular Indicators provide globalization options.

# Support and Learning Resources

- [Loader Homepage](#)
- [Getting Started with the Kendo UI for Angular Indicators](#)
- [API Reference of the Loader](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Loader Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Angular Indicators Overview

The Kendo UI for Angular Indicators are components for decorating other elements by creating a visual indication for the current status, an ongoing process, or a state change.

The Indicators are built from the ground up and specifically for Angular, so that you get high-performance indicator controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



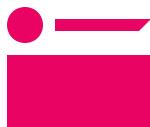
## Badge

A component that represents a visual indicator such as status or notification.



## Loader

A component that represents a visual indicator that expresses an indeterminate wait time.



## Skeleton

A component that represents a placeholder rendered on a page before the actual content loads.

# Angular Indicators Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮

A screenshot of a web application interface. At the top left is a navigation bar with the Kendo UI logo and the text "Angular Indicators Example". Below the navigation bar is a large button with the text "Get Started". Underneath the button is a section titled "Angular Indicators Example". This section contains three cards: "Badge", "Loader", and "Skeleton". Each card has a small icon to its left and a brief description below it. At the bottom of the page is a footer with the Kendo UI logo and the text "Angular Indicators Example".

```
import { Component } from "@angular/core";
import { BadgeComponent } from "./badge.component";
import { LoaderComponent } from "./loader.component";
import { SkeletonComponent } from "./skeleton.component";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [BadgeComponent, LoaderComponent, SkeletonComponent],
  template: `
    <div class="example">
      <div class="component">
        <div class="component-name">Badge</div>
        <badge-component></badge-component>
      </div>

      <div class="component">
        <div class="component-name">Loader</div>
        <loader-component></loader-component>
      </div>

      <div>
        <div class="component-name">Skeleton</div>
        <skeleton-component></skeleton-component>
      </div>
    `,
    styles: [
      .component {
        margin-bottom: 40px;
      }
      .component-name {
        margin: 0 0 20px;
      }
      loader-component,
      skeleton-component {
        display: block;
        margin-top: -25px;
      }
    ],
  })
export class AppComponent {}
```

Collanse Code ^

# Angular Indicators Key Features

Each Kendo UI for Angular Indicators component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Indicators library as well as develop new features and controls to it.

## Appearance

The color and style of the Indicators are normally picked up by the current Kendo UI theme, but each aspect of the Indicators can be customized by theme variables or configuration options. Kendo UI for Angular delivers a set of popular themes including Bootstrap and Material, all of which can be easily customized with the [Progress ThemeBuilder](#) online utility. [Read more about the appearance of the Badge...](#)

## Globalization

The Kendo UI for Angular Indicators support globalization to ensure that each Indicators component can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Indicators support rendering in a right-to-left (RTL) direction. [Read more about Indicators globalization...](#)

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular Indicators are part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library —no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of Kendo UI for Angular Indicators, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Indicators](#)
- [API Reference of the Indicators](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)

- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Skeleton Overview

The Kendo UI for Angular Skeleton helps developers to speed up the initial page-load time by displaying a simplified preview of the content.

This preview acts as a placeholder for the content while it is loading, and, at the same time, the user gets a general idea about what will appear on the page.

The following example demonstrates the Skeleton in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_INDICATORS } from "@progress/kendo-angular-indicators";
import { KENDO_LAYOUT } from "@progress/kendo-angular-layout";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_INDICATORS, KENDO_LAYOUT],
  template:
    <kendo-card width="45%" style="max-width: 350px;">
      <kendo-card-header class="k-hbox">
        <kendo-skeleton
          shape="circle"
          animation="pulse"
          [width]="40"
          [height]="40"
        ></kendo-skeleton>
      <div [ngStyle]={`${ flex: '1 1 50%', marginLeft: '16px' }`}>
        <kendo-skeleton
          shape="text"
          animation="pulse"
          width="100%"
        ></kendo-skeleton>
```

```
<kendo-skeleton  
    shape="text"  
    animation="pulse"  
    width="40%"  
    ></kendo-skeleton>  
  </div>  
</kendo-card-header>  
  
<kendo-skeleton  
    shape="rectangle"  
    animation="pulse"  
    width="100%"  
    height="143.86px"  
    ></kendo-skeleton>  
  
<kendo-card-footer class="k-hbox">  
  <kendo-skeleton  
    shape="text"  
    animation="pulse"  
    width="100%"  
    ></kendo-skeleton>  
  </kendo-card-footer>  
</kendo-card>  
<kendo-card width="45%" style="max-width: 350px; margin-left:  
15px;">  
  <kendo-card-header class="k-hbox">  
    <kendo-avatar  
      width="40px"  
      height="40px"  
      [imageSrc]="avatarUrl"  
      shape="circle"  
      ></kendo-avatar>  
    <div>  
      <h1 kendoCardTitle>Tom Smith</h1>  
      <p kendoCardSubtitle>5 hours ago</p>  
    </div>  
  </kendo-card-header>  
  
<img  
  [src]="thumbnailUrl"  
  alt="description"  
  [ngStyle]="{{ width: '100%' }}"  
/>  
  
<kendo-card-footer class="k-hbox">  
  <span>{{ description }}</span>  
</kendo-card-footer>  
</kendo-card>  
,  
  styles: [  
`
```

```
:host {
  display: flex;
}
],
})
export class AppComponent {
  public loading = true;

  public thumbnailUrl = "assets/indicators/skeleton/card-thumbnail.jpg";
  public avatarUrl = "assets/indicators/skeleton/user-avatar.jpg";
  public description = "Having so much fun in Prague! #NaZdravi";
}
```

Collapse Code ▾

# Key Features

- **Appearance**—The Skeleton delivers ready-to-use, predefined sets of styling options for modifying its shape and animation.
- **Globalization**—All Kendo UI for Angular Indicators provide globalization options.

# Support and Learning Resources

- [Skeleton Homepage](#)
- [Getting Started with the Kendo UI for Angular Indicators](#)
- [API Reference of the Skeleton](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Skeleton Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)

- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular CheckBox Overview

The Kendo UI for Angular CheckBox component allows the user to toggle between checked and unchecked states and supports all regular `<input type="checkbox">` HTML attributes and Angular bindings.

The following example demonstrates the CheckBox in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";

@Component({
  selector: "my-app",
  template: `
    <div class="row">
      <div class="col-xs-12 col-sm-6 example-col">
        <h4>Checkbox Component</h4>
        <p>
          <kendo-checkbox #checkedState [checkedState]="true"></kendo-
checkbox>
          <kendo-label
            class="k-checkbox-label"
            [for]="checkedState"
            text="Checked"
          ></kendo-label>
        </p>
        <p>
          <kendo-checkbox #uncheckedState></kendo-checkbox>
          <kendo-label
            class="k-checkbox-label"
            [for]="uncheckedState"
            text="Unchecked"
          ></kendo-label>
        </p>
        <p>
          <kendo-checkbox></kendo-checkbox>
        </p>
      </div>
    </div>
  `,
  styles: [
    ".example-col { border: 1px solid #ccc; padding: 10px; }"
  ]
})
```

```
        #indentState
        checkedState="indeterminate"
    ></kendo-checkbox>
    <kendo-label
        class="k-checkbox-label"
        [for]="indentState"
        text="Indeterminate"
    ></kendo-label>
</p>
<p>
    <kendo-checkbox
        #disabledState
        [disabled]="true"
        [checkedState]="true"
    ></kendo-checkbox>
    <kendo-label
        class="k-checkbox-label"
        [for]="disabledState"
        text="Disabled"
    ></kendo-label>
</p>
</div>
<div class="col-xs-12 col-sm-6 example-col">
    <h4>HTML Checkbox</h4>
    <p>
        <input type="checkbox" #checked kendoCheckBox [checked]="true"
    />
        <kendo-label
            class="k-checkbox-label"
            [for]="checked"
            text="Checked"
        ></kendo-label>
    </p>
    <p>
        <input type="checkbox" #unchecked kendoCheckBox />
        <kendo-label
            class="k-checkbox-label"
            [for]="unchecked"
            text="Unchecked"
        ></kendo-label>
    </p>
    <p>
        <input type="checkbox" #indent kendoCheckBox
[ineterminate]="true" />
        <kendo-label
            class="k-checkbox-label"
            [for]="indent"
            text="Indeterminate"
        ></kendo-label>
    </p>
    <p>
```

```
<input  
    type="checkbox"  
    #disabled  
    kendoCheckBox  
    [disabled]="true"  
    [checked]="true"  
/>  
<kendo-label  
    class="k-checkbox-label"  
    [for]="disabled"  
    text="Disabled"  
></kendo-label>  
</p>  
</div>  
</div>  
,  
})  
export class AppComponent {}
```

Collapse Code ^

# Key Features

- **Indeterminate CheckBox**—Whether or not the CheckBox is checked, you can render it in its indeterminate state.
- **Labels association**—The CheckBox provides approaches for associating it with an HTML `label` element.
- **Forms support**—You can use the CheckBox both in template-driven and reactive Angular forms.
- **Disabled CheckBox**—You can use the configuration options of the CheckBox to disable the component so that users are not able to interact with it.
- **Required CheckBox**—The CheckBox provides options for setting it as required.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.

To learn more about the appearance, anatomy, and accessibility of the CheckBox, visit the [Progress Design System documentation](#)—an information

portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

# Support and Learning Resources

- [CheckBox Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the CheckBox](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [CheckBox Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular ColorGradient Overview

The Kendo UI for Angular ColorGradient renders a gradient (a hue and an alpha slider) and inputs to manually enter a desired color.

The component is independently used by the [Kendo UI ColorPicker for Angular](#) and can be directly added to the page instead of rendered in a popup.

The following example demonstrates the ColorGradient in action.

EXAMPLE

VIEW SOURCE

⋮

```
import { Component, ViewEncapsulation } from
"@angular/core";

@Component({
  selector: "my-app",
  styleUrls: ["./clothes-skeleton-
app.component.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="clothes-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column no-flex">
            <div class="image-container">
              <div class="k-skeleton"></div>
              <kendoka-svg
[color]="currentColor"></kendoka-svg>
            </div>
          </div>
        </div>
      </div>
    </div>
  
```

CONFIGURATOR

File

app.component.ts

app.module.ts

clothes-skeleton-app.comp  
onent.css

kendoka-svg.component.ts

```
</div>
<div class="card-column">
    <h4 class="k-h4">T-shirt
Design</h4>
    <div class="component-container">
        <kendo-label text="Choose
color">
            <kendo-colorgradient
                [(value)]="currentColor"
            ></kendo-colorgradient>
        </kendo-label>
    </div>
</div>
</div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, //  
Encapsulation is disabled for demo purposes  
only.
})
export class AppComponent {
    public currentColor = "rgba(226, 30, 54,
1)";
}
```

Collasne Code ^

# Key Features

- **Customizing the ColorGradient**—You can enable the user to choose whether to render an alpha slider and an alpha input in the ColorGradient.
- **Forms support**—You can use the ColorGradient both in template-driven and reactive Angular forms.
- **Disabled ColorGradient**—You can use the configuration options of the ColorGradient to disable the component so that users are not able to interact with it.
- **Read-only ColorGradient**—The ColorGradient provides an option for overriding its default active state.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.

- [Accessibility](#)—The ColorGradient is accessible for screen readers and supports WAI-ARIA attributes.
- [Keyboard navigation](#)—The ColorGradient supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [ColorGradient Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the ColorGradient](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [ColorGradient Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular ColorPalette Overview

The Kendo UI for Angular ColorPalette renders colors by using color presets (sets of predefined colors) or by implementing a custom color palette.

The component is independently used by the [Kendo UI ColorPicker for Angular](#) and can be directly added to the page instead of rendered in a popup.

The following example demonstrates the ColorPalette in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";

@Component({
  selector: "my-app",
  template: `
    <div class="example">
      <div class="img" [ngStyle]="{ 'background-color': selected }">
    </div>
      <div class="description">
        <h1>Comfy T-shirt with a cut-away neckline</h1>
        <p class="price">$19.99</p>
        <span>incl. VAT</span>

        <p class="selected-color">{{ colorNames[selected] }}</p>
        <kendo-label text="Choose color">
          <kendo-colorpalette
            [palette]="palette"
            [value]="selected"
            [tileSize]="40"
            (valueChange)="onChange($event)">
          </kendo-colorpalette>
        </kendo-label>
        <button kendoButton>Add to cart</button>
    </div>
  `,
  styles: [
    ".example { border: 1px solid #ccc; padding: 10px; width: fit-content; margin: auto; }
    .img { width: 100%; height: 100px; background-color: #f0f0f0; }
    .description { margin-top: 10px; }
    .selected-color { color: green; font-weight: bold; }
    .kendo-label { margin-bottom: 10px; }
    .kendo-colorpalette { width: 100%; }
  ]
})
```

```
        </div>
    </div>
    ,
  styles: [
    .example {
      display: flex;
      justify-content: space-evenly;
      align-items: center;
      margin: 0 auto;
      width: 700px;
      color: #44403f;
    }
    .k-colorpalette .k-item {
      border-radius: 50%;
    }
    .description {
      width: 340px;
      margin-top: 35px;
      align-self: flex-start;
    }
    h1 {
      font-size: 32px;
      font-weight: bold;
      margin-bottom: 20px;
    }
    p {
      font-size: 20px;
      letter-spacing: 0.025em;
      margin-bottom: 0;
    }
    .description span {
      color: #8e8682;
      display: block;
      font-size: 13px;
    }
    .selected-color {
      margin: 40px 0 10px 0;
    }
    .img {
      width: 100%;
      height: 480px;
      margin-right: 10px;
      background-size: cover;
      background-image: url("assets/inputs/colorpalette/demo-
img.png");
    }
    button {
      display: block;
      background: transparent;
      color: #44403f;
    }
  ]
}
```

```
        border-color: currentColor;
        width: 200px;
        margin-top: 40px;
        padding: 8px;
        font-size: 20px;
    }
    button:hover {
        background: #0083ff;
        color: #fff;
        border-color: currentColor;
    }
],
})
export class AppComponent {
    public selected = "#a21616";
    public palette: Array<string> = [
        "#37399b",
        "#a81c85",
        "#0ab3cc",
        "#2f7d20",
        "#a21616",
    ];
    public colorNames: { [Key: string]: string } = {
        "#37399b": "Navy blue",
        "#a81c85": "Violet",
        "#0ab3cc": "Light blue",
        "#2f7d20": "Forest green",
        "#a21616": "Dark red",
    };
    public onChange(color: string): void {
        this.selected = color;
    }
}
```

Collanse Code ^

# Key Features

- **Color presets**—The ColorPalette delivers predefined color palettes as a faster and easier configuration approach.
- **Forms support**—You can use the ColorPalette both in template-driven and reactive Angular forms.

- **Customizing the ColorPalette**—You can enable the user to choose whether to render an alpha slider and an alpha input in the ColorPalette.
- **Disabled ColorPalette**—You can use the configuration options of the ColorPalette to disable the component so that users are not able to interact with it.
- **Read-only ColorPalette**—The ColorPalette provides an option for overriding its default active state.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Accessibility**—The ColorPalette is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The ColorPalette supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [ColorPalette Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the ColorPalette](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [ColorPalette Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)



# Kendo UI for Angular ColorPicker Overview

The Kendo UI for Angular ColorPicker enables the user to select and submit color values.

The ColorPicker component provides a rich interface to choose a color from Palette and Gradient views rendered in its popup. It enables the user to preview the selected color before submit and to ensure that certain contrast requirements are met.

The following example demonstrates the ColorPicker in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component, ViewEncapsulation } from "@angular/core";

@Component({
  selector: "my-app",
  styleUrls: ["./clothes-skeleton-app.component.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="clothes-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column no-flex">
            <div class="image-container">
              <div class="k-skeleton"></div>
              <kendoka-svg [color]="selected"></kendoka-svg>
            </div>
          </div>
        <div class="card-column">
      
```

```

<h4 class="k-h4">T-shirt Design</h4>
<div class="card-row card-row-inner">
  <div class="component-container">
    <kendo-label text="Color">
      <kendo-colorpicker
        [value]="selected"
        (valueChange)="onChange($event)"
      ></kendo-colorpicker>
    </kendo-label>
  </div>
  <div class="skeleton-container">
    <div class="k-skeleton skeleton-box-small"></div>
    <div class="k-skeleton skeleton-box-large"></div>
  </div>
  <div class="skeleton-container">
    <div class="k-skeleton skeleton-box-small"></div>
    <div class="k-skeleton skeleton-box-xlarge"></div>
  </div>
  <div class="skeleton-container bottom">
    <div class="k-skeleton skeleton-box-medium"></div>
    <div class="k-skeleton skeleton-box-medium"></div>
  </div>
  </div>
</div>
``,
  encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
  public selected = "#E21E36";

  public onChange(color: string): void {
    this.selected = color;
  }
}

```

[Collapse Code ^](#)

# Key Features

- **Views**—The ColorPicker provides options for displaying gradient and palette views, as well as switching between them in a seamless way.

- **Color Preview**—You can enable the user to preview and compare the selected color with the ColorPicker value, before applying it. Additionally, the current selection can be reverted to its initial state.
- **Contrast Tool**—The ColorPicker provides an option for displaying a contrast tool. It ensures the user that the selected color meets certain contrast requirements e.g. AA or AAA.
- **Customizing the ColorPicker**—You can customize almost any building block of the ColorPicker like the views toggle buttons, clear button, action buttons layout, alpha slider etc.
- **Disabled ColorPicker**—You can use the configuration options of the ColorPicker to disable the component so that users are not able to interact with it.
- **Read-only ColorPicker**—The ColorPicker provides an option for overriding its default active state.
- **Controlling the open state**—The ColorPicker enables you to configure the open state of its popup.
- **Forms support**—You can use the ColorPicker both in template-driven and reactive Angular forms.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Accessibility**—The ColorPicker is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The ColorPicker supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [ColorPicker Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the ColorPicker](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)

- [ColorPicker Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular FlatColorPicker Overview

The Kendo UI for Angular FlatColorPicker enables the user to select and submit color values.

The FlatColorPicker component provides a rich interface to choose a color from Palette and Gradient views. It enables the user to preview the selected color before submit and to ensure that a certain contrast requirements are met.

The following example demonstrates the FlatColorPicker in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component, ViewEncapsulation } from "@angular/core";

@Component({
  selector: "my-app",
  styleUrls: ["./clothes-skeleton-app.component.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="clothes-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column no-flex">
            <div class="image-container">
              <div class="k-skeleton"></div>
              <kendoka-svg [color]="currentColor"></kendoka-svg>
            </div>
          </div>
        <div class="card-column">
      
```

```

<h4 class="k-h4">T-shirt Design</h4>
<div class="component-container">
    <kendo-label text="Choose color">
        <kendo-flatcolorpicker
            [(value)]="currentColor"
        ></kendo-flatcolorpicker>
    </kendo-label>
</div>
</div>
</div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
    public currentColor = "rgba(226, 30, 54, 1)";
}

```

[Collapse Code ^](#)

# Key Features

- **Views**—The FlatColorPicker provides options for displaying gradient and palette views, as well as switching between them in a seamless way.
- **Color Preview**—You can enable the user to preview and compare the selected color with the FlatColorPicker value, before applying it. Additionally, the current selection can be reverted to its initial state.
- **Color Contrast Tool**—The FlatColorPicker provides an option for displaying a contrast tool. It ensures the user that the selected color meets certain contrast requirements like AA or AAA.
- **Customizing the FlatColorPicker**—You can customize almost any building block of the FlatColorPicker like the views toggle buttons, clear button, action buttons layout, alpha slider etc.
- **Disabled FlatColorPicker**—You can use the configuration options of the FlatColorPicker to disable the component so that users are not able to interact with it.
- **Read-only FlatColorPicker**—The FlatColorPicker provides an option for overriding its default active state.

- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Forms support**—You can use the FlatColorPicker both in template-driven and reactive Angular forms.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Accessibility**—The FlatColorPicker is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The FlatColorPicker supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [FlatColorPicker Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the FlatColorPicker](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [FlatColorPicker Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular FormField Overview

The Kendo UI for Angular `FormField` enables you to group and provide configurable behavior for form-related content such as inputs, labels, hint and error messages.

Each `kendo-formfield` element can contain a single form-control element, a label, or multiple hint or error messages, but cannot accommodate a group of `RadioButtons`.

The following example demonstrates the `FormField` in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component, ViewEncapsulation } from "@angular/core";
import { FormGroup, FormControl, Validators } from "@angular/forms";

@Component({
  selector: "my-app",
  template: `
    <div class="example-wrap">
      <form class="k-form k-form-md" [FormGroup]="registerForm">
        <fieldset class="k-form-fieldset">
          <legend class="k-form-legend">User Details</legend>
          <kendo-formfield>
            <kendo-label
              labelCssClass="k-form-label"
              [for]="firstName"
              text="First Name"
            ></kendo-label>
            <kendo-textbox
              formControlName="firstName"
              #firstName
              required
            ></kendo-textbox>

            <kendo-formhint>Your first name</kendo-formhint>
          </kendo-formfield>
        </fieldset>
      </form>
    </div>
  `
```

```
<kendo-formerror>Error: First name is required</kendo-
formerror>
</kendo-formfield>

<kendo-formfield>
  <kendo-label
    labelCssClass="k-form-label"
    [for]="lastName"
    text="Last Name"
  ></kendo-label>
  <kendo-textbox
    formControlName="lastName"
    #lastName
    required
  ></kendo-textbox>

  <kendo-formhint>Your last name</kendo-formhint>
  <kendo-formerror>Error: Last name is required</kendo-
formerror>
</kendo-formfield>

<kendo-formfield>
  <kendo-label
    labelCssClass="k-form-label"
    [for]="birthDate"
    [optional]="true"
    text="Birth Date"
  ></kendo-label>
  <kendo-datepicker
    #birthDate
    formControlName="birthDate"
    [min]="min"
    [max]="max"
  >
  </kendo-datepicker>

  <kendo-formhint>Your birth date</kendo-formhint>
</kendo-formfield>

<kendo-formfield>
  <kendo-label
    labelCssClass="k-form-label"
    [for]="email"
    text="Email"
  ></kendo-label>
  <kendo-textbox
    formControlName="email"
    #email
    required
  ></kendo-textbox>
```

```
        <kendo-formhint>Your active email</kendo-formhint>
        <kendo-formerror>Error: Not valid email format</kendo-
formerror>
    </kendo-formfield>

    <kendo-formfield>
        <div class="k-checkbox-wrap">
            <input
                #acceptNews
                type="checkbox"
                kendoCheckBox
                formControlName="acceptNews"
            />
            <kendo-label
                [for]="acceptNews"
                class="k-checkbox-label"
                text="I want to receive notifications"
            ></kendo-label>
        </div>

        <kendo-formhint
            >You will receive our latest updates and promotions on
your
            email</kendo-formhint
        >
    </kendo-formfield>

    <div class="k-form-buttons">
        <button kendoButton themeColor="primary"
(click)="submitForm()">
            Submit
        </button>
        <button kendoButton (click)="clearForm()">Clear</button>
    </div>
    </fieldset>
</form>
</div>
,
styles: [
    .example-wrap {
        display: flex;
        justify-content: center;
    }
    .k-form {
        width: 400px;
    }
],
encapsulation: ViewEncapsulation.None,
})
```

```
export class AppComponent {
  public min: Date = new Date(1917, 0, 1);
  public max: Date = new Date(2020, 4, 31);

  public registerForm: FormGroup = new FormGroup({
    firstName: new FormControl(),
    lastName: new FormControl(),
    birthDate: new FormControl(new Date(2000, 10, 10)),
    email: new FormControl("", Validators.email),
    acceptNews: new FormControl(),
  });

  public submitForm(): void {
    this.registerForm.markAllAsTouched();
  }

  public clearForm(): void {
    this.registerForm.reset();
  }
}
```

Collanse Code ^

## Key Features

- **Hints and errors**—The `FormField` provides options for displaying hint and error messages.
- **Horizontal fields**—You can change the layout of the `FormField` content by configuring its orientation.
- **Forms support**—You can use the `FormField` both in template-driven and reactive Angular forms.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.

## Support and Learning Resources

- [FormField Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)

- [API Reference of the FormField](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [FormField Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Inputs

This guide provides the information you need to start using the Kendo UI for Angular Inputs—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#).

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_LABEL } from "@progress/kendo-angular-label";

@Component({
  standalone: true,
  imports: [KENDO_INPUTS, KENDO_LABEL],
  selector: "my-app",
  template: `
    <kendo-label
      class="k-display-block"
    >My Label</kendo-label>
  `})
export class AppComponent {
  constructor() {
    console.log("App component initialized");
  }
}
```

```
[for]="picker"
  text="ColorPicker"
></kendo-label>
<kendo-colorpicker #picker> </kendo-colorpicker>
  ,
})
export class AppComponent {}
```

Collanse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Inputs package:

1. Run the following command.

```
ng add @progress/kendo-angular-inputs
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-inputs` package as a dependency to the `package.json` file.

- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the required components, directives and utility arrays:

The utility arrays are available starting from **v16.6.0**. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

- To add all components from the **Inputs** package, import the `KENDO_INPUTS` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_INPUTS } from '@progress/kendo-angular-inputs';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_INPUTS]
})
```

TS

- To add individual **Inputs** components, import the corresponding utility arrays in your standalone component. See the list of [available utility arrays](#).

For example if you only need the **TextBox** component, import `KENDO_TEXTBOX`.

```
import { Component } from '@angular/core';
import { KENDO_TEXTBOX } from '@progress/kendo-angular-inputs';

@Component({
  standalone: true,
  selector: 'my-app',
```

TS

```
    imports: [KENDO_TEXTBOX]
  })
```

# Using the Components

1. After successfully installing the Inputs package and importing the required components and directives, add the corresponding tags of the components you need in the `app.component.html`. For example, if you need the ColorPicker, add the following code:

```
<kendo-colorpicker> </kendo-colorpicker>
```

HTML

2. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

3. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular Inputs component on the page.

# Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

# Next Steps

- [CheckBox Overview](#)

- [ColorGradient Overview](#)
- [ColorPalette Overview](#)
- [ColorPicker Overview](#)
- [FlatColorPicker Overview](#)
- [FormField Overview](#)
- [MaskedTextBox Overview](#)
- [NumericTextBox Overview](#)
- [RadioButton Overview](#)
- [RangeSlider Overview](#)
- [Signature Overview](#)
- [Slider Overview](#)
- [Switch Overview](#)
- [TextArea Overview](#)
- [TextBox Overview](#)
- [Globalization](#)
- [Inputs API Documentation](#)

# Learning Resources

- [Inputs Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Kendo UI for Angular MaskedTextBox Overview

The Kendo UI for Angular MaskedTextBox enables you to control the user input by using a mask.

The following example demonstrates the MaskedTextBox in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";

@Component({
  selector: "my-app",
  styleUrls: ["./profile-skeleton-app.component.css"],
  styles: [
    `
      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="profile-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column no-flex">
            <div class="sidebar-container k-skeleton">
              <div class="avatar-name-container">
                <div class="k-skeleton skeleton-avatar"></div>
                <div class="name-container">
                  <div class="k-skeleton skeleton-text"></div>
                  <div class="k-skeleton skeleton-small-text-short">
                    <div>
                      </div>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  
```

```

        <div class="k-skeleton skeleton-small-text"></div>
        <div class="k-skeleton skeleton-small-text"></div>
    </div>
</div>
<div class="card-column">
    <div class="avatar-title-container">
        <div class="k-skeleton skeleton-avatar"></div>
        <h4 class="k-h4">My Profile</h4>
    </div>
    <div class="skeleton-container top">
        <div class="k-skeleton skeleton-box-small"></div>
        <div class="k-skeleton skeleton-box-large"></div>
    </div>
    <div class="component-container">
        <kendo-label text="Phone Number">
            <kendo-maskedtextbox [mask]="mask" [value]="value">
            </kendo-maskedtextbox>
        </kendo-label>
    </div>
    <div class="skeleton-container bottom">
        <div class="k-skeleton skeleton-box-small"></div>
        <div class="k-skeleton skeleton-box-large-double"></div>
    </div>
    </div>
</div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
    public value = "359884123321";
    public mask = "(999) 000-00-00-00";
}

```

[Collapse Code ^](#)

# Key Features

- **Masks**—You can use the predefined mask rules of the MaskedTextBox to require a specific type of user input such as digits, spaces, letters, and more.
- **Validation**—The MaskedTextBox delivers a built-in mask validator which ensures that the user input is valid.

- **Forms support**—You can use the MaskedTextBox both in template-driven and reactive Angular forms.
- **Value**—The MaskedTextBox provides settings for formatting its visible (masked) value.
- **Disabled MaskedTextBox**—You can use the configuration options of the MaskedTextBox to disable the component so that users are not able to interact with it.
- **Read-only MaskedTextBox**—The MaskedTextBox provides an option for overriding its default active state.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Accessibility**—The MaskedTextBox is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The MaskedTextBox supports a number of keyboard shortcuts for processing various commands.

To learn more about the appearance, anatomy, and accessibility of the MaskedTextBox, visit the [Progress Design System documentation](#)—an information portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

## Support and Learning Resources

- [MaskedTextBox Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the MaskedTextBox](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [MaskedTextBox Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)

- Knowledge Base

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular NumericTextBox Overview

The Kendo UI for Angular NumericTextBox enables the user to edit and submit specific numeric values by typing or by using the spin buttons.

The following example demonstrates the NumericTextBox in action.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { Component, ViewEncapsulation } from "@angular/core";

@Component({
  selector: "my-app",
  styleUrls: ["./profile-skeleton-app.component.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="profile-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column no-flex">
            <div class="sidebar-container k-skeleton">
              <div class="avatar-name-container">
                <div class="k-skeleton skeleton-avatar"></div>
                <div class="name-container">
                  <div class="k-skeleton skeleton-text"></div>
                  <div class="k-skeleton skeleton-small-text-short">
                    <div class="k-skeleton skeleton-small-text"></div>
                  </div>
                </div>
              </div>
            <div class="description-container">
              <div class="k-skeleton skeleton-small-text"></div>
            </div>
          </div>
        </div>
      </div>
    </div>
  `
})
```

```

        <div class="k-skeleton skeleton-small-text"></div>
        <div class="k-skeleton skeleton-small-text"></div>
    </div>
</div>
<div class="card-column">
    <div class="avatar-title-container">
        <div class="k-skeleton skeleton-avatar"></div>
        <h4 class="k-h4">My Profile</h4>
    </div>
    <div class="skeleton-container top">
        <div class="k-skeleton skeleton-box-small"></div>
        <div class="k-skeleton skeleton-box-large"></div>
    </div>
    <div class="component-container">
        <kendo-label text="Weight">
            <kendo-numerictextbox
                format="#.00 kg"
                [step]="0.1"
                [value]="value"
            ></kendo-numerictextbox>
        </kendo-label>
    </div>
    <div class="skeleton-container bottom">
        <div class="k-skeleton skeleton-box-small"></div>
        <div class="k-skeleton skeleton-box-large-double"></div>
    </div>
    </div>
</div>
</div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {
    public value = 76;
}

```

[Collapse Code ^](#)

# Key Features

- **Restriction of user input**—You can use the `NumericTextBox` settings which enable you to control the user input such as restricting the length of the fraction, setting value ranges, and more.

- **Formats**—The NumericTextBox provides formatting options for its input value and supports all date and number formats provided by the Kendo UI Internationalization package.
- **Predefined steps**—You can use the configuration options of the NumericTextBox to define the step which the component uses to increase or decrease its value.
- **Forms support**—You can use the NumericTextBox both in template-driven and reactive Angular forms.
- **Spin buttons**—The NumericTextBox enables you control its spin buttons which allow users to increase or decrease the value with a predefined step.
- **Disabled NumericTextBox**—You can use the configuration options of the NumericTextBox to disable the component so that users are not able to interact with it.
- **Read-only NumericTextBox**—The NumericTextBox provides an option for overriding its default active state.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Accessibility**—The NumericTextBox is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The NumericTextBox supports a number of keyboard shortcuts for processing various commands.

To learn more about the appearance, anatomy, and accessibility of the NumericTextBox, visit the [Progress Design System documentation](#)—an information portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

## Known Limitations

To keep its value, the NumericTextBox uses the [Number](#) JavaScript object. The [Number](#) JavaScript object persists its precision up to 16 digits, and units which are longer than that get converted to exponential numbers and lose their precision.

Because the component relies on `Number`, it inherits the precision limitation and because the limitation is caused by the JavaScript logic, the NumericTextBox does not provide a workaround to handle it.

# Support and Learning Resources

- [NumericTextBox Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the NumericTextBox](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [NumericTextBox Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular OTP Input Overview

The Kendo UI for Angular OTP Input enables the user to enter one-time passwords during multi-factor authentication, which enhances the security of login processes.

The following example demonstrates the Kendo UI for Angular OTP Input in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_OTPINPUT } from "@progress/kendo-angular-inputs";
import { KENDO_LABEL } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_OTPINPUT, KENDO_LABEL],
  styleUrls: ["./profile-skeleton-app.component.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="profile-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column no-flex">
            <div class="sidebar-container k-skeleton">
              <div class="avatar-name-container">
                <div class="k-skeleton skeleton-avatar"></div>
                <div class="name-container">
                  <div class="k-skeleton skeleton-text"></div>
                  <div class="k-skeleton skeleton-small-text-short">
                    <div class="k-skeleton skeleton-text"></div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  `
})
```

```

        </div>
    </div>
    <div class="description-container">
        <div class="k-skeleton skeleton-small-text"></div>
        <div class="k-skeleton skeleton-small-text"></div>
        <div class="k-skeleton skeleton-small-text"></div>
    </div>
</div>
<div class="card-column">
    <div class="avatar-title-container">
        <div class="k-skeleton skeleton-avatar"></div>
        <h4 class="k-h4">Profile Authentication</h4>
    </div>
    <div class="skeleton-container top">
        <div class="k-skeleton skeleton-box-small"></div>
        <div class="k-skeleton skeleton-box-large"></div>
    </div>
    <div class="component-container">
        <kendo-label text="Enter the code sent to your phone">
            <kendo-otpinput
                [length]="6"
                [groupLength]="3"
                separator="-"
            ></kendo-otpinput>
        </kendo-label>
    </div>
    <div class="skeleton-container bottom">
        <div class="k-skeleton skeleton-box-small"></div>
        <div class="k-skeleton skeleton-box-large-double"></div>
    </div>
    </div>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {}

```

Collanse Code ^

# Key Features

- **Types**—You can specify different types of user input in the OTP Input component.

- **Separator**—The OTP Input enables you to render a desired separator between groups of input fields.
- **Forms support**—You can use the OTP Input both in template-driven and reactive Angular forms.
- **Placeholder**—The OTP Input allows you to render hint characters in its input fields.
- **Appearance**—The OTP Input provides built-in customization options that allow you to modify its appearance.
- **Disabled OTP Input**—You can use the configuration options of the OTP Input to disable the component so that users are not able to interact with it.
- **Read-Only OTP Input**—The OTP Input provides an option for overriding its default active state.
- **Adaptiveness**—You can configure the type of an on-screen keyboard that will display upon touch interaction with the OTP Input.
- **Accessibility**—The OTP Input is accessible for screen readers and supports WAI-ARIA attributes.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Keyboard navigation**—The OTP Input supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the OTP Input](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [OTP Input Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Angular Inputs Overview

The Kendo UI for Angular Inputs are fields for allowing the input of data based on specific and predefined formats.

They include a variety of input types and styles that have extensive configuration options. This flexibility allows you to quickly and easily create the exact input you need to fit your specific requirements for functionality and appearance.

The Inputs are built from the ground up and specifically for Angular, so that you get high-performance input controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



**CheckBox**

A directive for rendering a checkbox element



**ColorPallete**

A component for selecting one or more items from a set



**FlatColorPicker**

A component for selecting and submitting color values, through different views



**ColorGradient**

A component for displaying gradient, a hue and an alpha slider used to pick colors



**ColorPicker**

A component for selecting and submitting color values



**FormField**

A component for grouping form-related content such as inputs, labels, hint and error



## MaskedTextBox

A component for controlling the input of the user



## OTP Input

A component for entering a one-time password



## RangeSlider

A component for selecting range-values from a predefined range



## Signature

A component for creating handwritten signatures



## Switch

A component for toggling between checked and unchecked states

messages



## NumericTextBox

A component for editing and submitting specific numeric values



## RadioButton

A directive for rendering a radiobutton element



## Rating

A component for rating



## Slider

A component for selecting values from a predefined range



## TextArea

A component for getting or displaying multiline text input submitted by the user



## TextBox

A component for getting or displaying text input submitted by the user

# Angular Inputs Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";

@Component({
  selector: "my-app",
  template: `
    <div class="example-wrapper">
      <div class="row">
        <div class="col-xs-12 col-sm-6 example-col">
          <kendo-label
            class="k-display-block"
            [for]="gradient"
            text="ColorGradient"
          ></kendo-label>
          <kendo-colorgradient #gradient value="#7e16c8"></kendo-
colorgradient>
        </div>
        <div class="col-xs-12 col-sm-6 example-col">
          <kendo-label
            class="k-display-block"
            [for]="palette"
            text="ColorPalette"
          ></kendo-label>
          <kendo-colorpalette #palette [tileSize]="30"></kendo-
colorpalette>
        </div>
        <div class="col-xs-12 col-sm-6 example-col">
          <kendo-label
            class="k-display-block"
            [for]="flat"
          ></kendo-label>
        </div>
    </div>
  `)
```

```
        text="FlatColorPicker"
    ></kendo-label>
    <kendo-flatcolorpicker #flat value="#326393"></kendo-
flatcolorpicker>
</div>
<div class="col-xs-12 col-sm-6 example-col">
    <kendo-label
        class="k-display-block"
        [for]="picker"
        text="ColorPicker"
    ></kendo-label>
    <kendo-colorpicker #picker value="#16c8aa"></kendo-
colorpicker>
</div>
<div class="col-xs-12 col-sm-6 example-col">
    <kendo-label
        class="k-display-block"
        [for]="masked"
        text="MaskedTextBox"
    ></kendo-label>
    <kendo-maskedtextbox
        #masked
        [(ngModel)]="maskedValue"
        mask="9-000"
    ></kendo-maskedtextbox>
</div>
<div class="col-xs-12 col-sm-6 example-col">
    <kendo-label
        class="k-display-block"
        [for]="numeric"
        text="NumericTextBox"
    ></kendo-label>
    <kendo-numerictextbox
        #numeric
        [(ngModel)]="numericValue"
    ></kendo-numerictextbox>
</div>
<div class="col-xs-12 col-sm-6 example-col">
    <kendo-label
        class="k-display-block"
        [for]="switch"
        text="Switch"
    ></kendo-label>
    <kendo-switch #switch [(ngModel)]="switchValue"></kendo-
switch>
</div>
</div>
<div class="row">
    <div class="col-xs-12 col-sm-6 example-col">
        <kendo-label
            class="k-display-block"
```

```
[for]="slider"
text="Slider"
></kendo-label>
<p>(value = {{ sliderValue }})</p>
<kendo-slider
#slider
[min]="min"
[max]="max"
[smallStep]="smallStep"
[(ngModel)]="sliderValue"
></kendo-slider>
</div>
<div class="col-xs-12 col-sm-6 example-col">
<kendo-label
class="k-display-block"
[for]="rangeslider"
text="RangeSlider"
></kendo-label>
<p>(value = {{ rangeSliderValue | json }})</p>
<kendo-rangeslider
#rangeslider
[smallStep]="smallStep"
[(ngModel)]="rangeSliderValue"
></kendo-rangeslider>
</div>
<div class="col-xs-12 col-sm-6 example-col">
<kendo-label
class="k-display-block"
[for]="textbox"
text="TextBox"
></kendo-label>
<kendo-textbox
#textbox
[style.width.px]="200"
[clearButton]="true"
[(ngModel)]="textboxValue"
></kendo-textbox>
</div>
<div class="col-xs-12 col-sm-6 example-col">
<kendo-label
class="k-display-block"
[for]="textarea"
text="TextArea"
></kendo-label>
<kendo-textarea
#textarea
[(ngModel)]="textareaValue"
></kendo-textarea>
</div>
</div>
<div class="row">
```

```
<div class="col-xs-12 col-sm-6 example-col">
  <kendo-label
    class="k-display-block"
    [for]="checkbox"
    text="CheckBox"
  ></kendo-label>
  <p>(checked = {{ checked }})</p>
  <input
    #checkbox
    type="checkbox"
    kendoCheckBox
    [(ngModel)]="checked"
  />
</div>
<div class="col-xs-12 col-sm-6 example-col">
  <kendo-label class="k-display-block" text="RadioButton">
</kendo-label>
  <p>(value = {{ radioValue }})</p>
  <div>
    <input
      #foo
      type="radio"
      name="demo"
      value="foo"
      kendoRadioButton
      [(ngModel)]="radioValue"
    />
    <kendo-label
      class="k-radio-label"
      [for]="foo"
      text="Foo"
    ></kendo-label>

    <input
      #bar
      type="radio"
      name="demo"
      value="bar"
      kendoRadioButton
      [(ngModel)]="radioValue"
    />
    <kendo-label
      class="k-radio-label"
      [for]="bar"
      text="Bar"
    ></kendo-label>
  </div>
</div>
<div class="row">
  <div class="col-xs-12 col-sm-6 example-col">
```

```
<kendo-label  
    class="k-display-block"  
    [for]="rating"  
    text="Rating"  
></kendo-label>  
<p>(value = {{ ratingValue }})</p>  
<kendo-rating #rating [(ngModel)]="ratingValue"></kendo-  
rating>  
    </div>  
    </div>  
    </div>  
,  
}  
export class AppComponent {  
    public maskedValue: string;  
    public sliderValue = 5;  
    public ratingValue = 3;  
    public numericValue = 5;  
    public switchValue = false;  
    public textBoxValue = "";  
    public textareaValue = "";  
    public min = 0;  
    public max = 10;  
    public smallStep = 1;  
    public checked = true;  
    public radioValue = "foo";  
    public rangeSliderValue = [3, 6];  
}  
Collanse Code ^
```

# Angular Inputs Key Features

Each Kendo UI for Angular Inputs component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Inputs library as well as develop new features and controls to it.

## Delaying Input Value Changes

By default, value updates are processed immediately after the user input. However, if you need to provide for some more complex scenarios and operations, you can

still implement a slight delay before the components accept the new input value. [Read more about debouncing value changes in the Inputs...](#)

## Disabled Inputs

You can choose to render the Inputs in their disabled state so that, if need be present, users will not be able to interact with them. [Read more about the Disabled MaskedTextBox...](#)

## Forms Support

The Inputs provide support both for the asynchronous template-driven Angular forms and the predominantly synchronous reactive Angular forms. This feature allows you to draw on the logic set either in the template, or in the component or typescript code. [Read more about the forms support of the MaskedTextBox...](#)

## Appearance

The color and style of the Inputs are normally picked up by the current Kendo UI theme, but each aspect of the Inputs can be customized by theme variables or configuration options. Kendo UI for Angular delivers a set of popular themes including Bootstrap and Material, all of which can be easily customized with the Progress ThemeBuilder online utility. [Read more about styling the Inputs...](#)

## Customization

The color-picking Inputs provide various configuration options for the user to specify their rendering and customize certain aspects of the components' appearance. [Read more about the customization options of the ColorPicker...](#)

## Adornments

You can enhance the visual and functional aspects of the text-based Inputs by using custom adornments in the form of prefixes and suffixes. Decorating the components allows you to make them more user-friendly and engaging. [Read more about the adornments of the TextBox...](#)

## Labels

You have the ability to display labels for the Inputs by associating them with a label element, ensuring that the components follow best practices for adding input elements to your Angular application. Some of the Inputs also provide built-in properties for customizing their internal labels. [Read more about labels association of the CheckBox...](#)

## Globalization

The Kendo UI for Angular Inputs support globalization to ensure that each Inputs component can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Inputs support rendering in a right-to-left (RTL) direction. [Read more about Inputs globalization...](#)

## Accessibility

The Inputs are accessible for screen readers and support WAI-ARIA attributes. [Read more about the accessibility support of the MaskedTextBox...](#)

## Keyboard Navigation

The Inputs support a number of keyboard shortcuts which allow users to accomplish various commands. [Read more about the keyboard navigation of the MaskedTextBox...](#)

# Trial Version of Kendo UI for Angular

The Kendo UI for Angular Inputs are part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of Kendo UI for Angular Inputs, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the Inputs](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)

- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular RadioButton Overview

The Kendo UI for Angular RadioButton component allows you to create a list of radio buttons where you can only select one of the predefined options. It supports all regular `<input type="radio">` HTML attributes and Angular bindings.

The following example demonstrates the RadioButton in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";

@Component({
  selector: "my-app",
  template: `
    <div class="row">
      <div class="col-xs-12 col-sm-6 example-col">
        <h4>RadioButton Component</h4>
        <p>Choose page layout</p>
        <p>
          <kendo-radiobutton
            #checked
            value="Landscape"
            [(ngModel)]="radio1.layout"></kendo-radiobutton>
          <kendo-label
            class="k-radio-label"
            [for]="checked"
            text="Landscape"></kendo-label>
        </p>
        <p>
          <kendo-radiobutton
            #unchecked
            value="Portrait"
            [(ngModel)]="radio1.layout"></kendo-radiobutton>
        </p>
    </div>
  </div>
}
```

```
></kendo-radiobutton>
<kendo-label
  class="k-radio-label"
  [for]="unchecked"
  text="Portrait"
></kendo-label>
</p>
<p>
  <kendo-radiobutton
    #disabled
    value="Square"
    [(ngModel)]="radio1.layout"
    [disabled]="true"
  ></kendo-radiobutton>
  <kendo-label
    class="k-radio-label"
    [for]="disabled"
    text="Square (disabled)"
  ></kendo-label>
</p>
</div>
<div class="col-xs-12 col-sm-6 example-col">
  <h4>HTML Radio Button</h4>
  <p>Choose page layout</p>
  <p>
    <input
      type="radio"
      kendoRadioButton
      value="Landscape"
      #landscape
      [(ngModel)]="radio2.layout"
    />
    <kendo-label
      class="k-radio-label"
      [for]="landscape"
      text="Landscape"
    ></kendo-label>
  </p>
  <p>
    <input
      type="radio"
      kendoRadioButton
      value="Portrait"
      #portrait
      [(ngModel)]="radio2.layout"
    />
    <kendo-label
      class="k-radio-label"
      [for]="portrait"
      text="Portrait"
    ></kendo-label>
  </p>
</div>
```

```

        </p>
        <p>
            <input
                type="radio"
                kendoRadioButton
                value="Square"
                #square
                [(ngModel)]="radio2.layout"
                [disabled]="true"
            />
            <kendo-label
                class="k-radio-label"
                [for]="square"
                text="Square (disabled)"
            ></kendo-label>
        </p>
    </div>
</div>
` ,
})
export class AppComponent {
    public radio1 = {
        layout: "Landscape",
    };

    public radio2 = {
        layout: "Landscape",
    };
}

```

[Collapse Code ▲](#)

# Key Features

- **Forms support**—You can use the RadioButton both in template-driven and reactive Angular forms.
- **Labels association**—The RadioButton provides approaches for associating it with an HTML `label` element.
- **Disabled RadioButton**—You can use the configuration options of the RadioButton to disable the component so that users are not able to interact with it.
- **Required RadioButton**—The RadioButton provides options for setting it as required.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.

- [Appearance](#)—All Kendo UI for Angular Inputs enable you to set their dimensions.
- [Globalization](#)—All Kendo UI for Angular Inputs provide globalization options.

To learn more about the appearance, anatomy, and accessibility of the RadioButton, visit the [Progress Design System documentation](#)—an information portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

## Support and Learning Resources

- [RadioButton Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the RadioButtonComponent](#)
- [API Reference of the RadioButton directive](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [RadioButton Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular RangeSlider Overview

The Kendo UI for Angular RangeSlider enables the user to select ranges by dragging its handles or by clicking its track.

The following example demonstrates the RangeSlider in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮

```
import { Component } from "@angular/core";

@Component({
  selector: "my-app",
  template: `
    <div class="example-config">
      Selected Range: <strong> {{ value[0] }} - {{ value[1] }}</strong>
    </div>
    <kendo-label
      class="k-display-block"
      [for]="range"
      text="Choose a price range">
    </kendo-label>
    <kendo-rangeslider
      #range
      [(ngModel)]="value"
      [min]="min"
      [max]="max"
      [smallStep]="smallStep"
      [largeStep]="largeStep">
    </kendo-rangeslider>
  `,
})
export class AppComponent {
  public value: [number, number] = [50, 100];
  public min = 0;
  public max = 200;
}
```

CONFIGURATOR X

File

app.component.ts

app.module.ts

```
    public largeStep = 2;  
    public smallStep = 20;  
}
```

Collapse Code ^



# Key Features

- **Orientation**—The RangeSlider allows you to display it in a vertical or horizontal direction.
- **Forms support**—You can use the RangeSlider both in template-driven and reactive Angular forms.
- **Ticks**—The RangeSlider provides configuration options for setting the placement, titles, and width of its ticks, which indicate the values resulting from each incremented predefined step.
- **Predefined steps**—You can use the configuration options of the RangeSlider to define the step which the component uses to increase or decrease its value.
- **Disabled RangeSlider**—You can use the configuration options of the RangeSlider to disable the component so that users are not able to interact with it.
- **Read-only RangeSlider**—The RangeSlider provides an option for overriding its default active state.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Accessibility**—The RangeSlider is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The RangeSlider supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [RangeSlider Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)

- [API Reference of the RangeSlider](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [RangeSlider Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Rating Overview

The Kendo UI for Angular Rating component enables the user to increase, decrease, and select predefined values by clicking its icons along or using the arrow keys.

The following example demonstrates the Angular Rating in action.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { Component } from "@angular/core";

@Component({
  selector: "my-app",
  template: `<kendo-rating></kendo-rating>`,
})
export class AppComponent {}
```

[Collapse Code ^](#)

## Key Features

- **Selection**—The Rating allows you to configure it for continuous or single selection mode.
- **Precision**—You can configure the Rating and allow it to render half-full items.
- **Number of Items**—The Rating allows you to specify the number of items that should be rendered.
- **Label**—The Rating allows you to enable or disable a label.
- **Icon**—The Rating allows you to change the default icon.
- **Disabled State**—The Rating allows you to change the default icon.
- **Read-Only State**—The Rating provides an option for overriding its default active state.
- **Templates**—You can customize the icon by using templates.
- **Form Support**—You can use the Rating both in template-driven and reactive Angular forms.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Accessibility**—The Rating is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The Rating supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [Angular Rating Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the Angular Rating](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Angular Rating Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Signature Overview

The Kendo UI for Angular Signature enables the user to create handwritten signatures and submit them as part of a form.

The following example demonstrates the Signature in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, OnDestroy, ViewEncapsulation } from "@angular/core";
import { DomSanitizer, SafeUrl } from "@angular/platform-browser";
import { saveAs } from "@progress/kendo-file-saver";
import {
  SVGIcon,
  brushIcon,
  uploadIcon,
  saveIcon,
  imageIcon,
} from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  template: `
    <div class="example">
      <div class="content-wrapper">
        <kendo-toolbar>
          <ng-container *ngIf="!showUpload">
            <colorpicker-tool [(value)]="color">Brush:</colorpicker-
tool>
            <kendo-toolbar-splitbutton text="Size" [data]="sizes">
            </kendo-toolbar-splitbutton>
            <kendo-toolbar-separator></kendo-toolbar-separator>
            <colorpicker-tool [(value)]="backgroundColor">
              >Background:</colorpicker-tool
            >
          </ng-container>
          <upload-tool *ngIf="showUpload" />
        </kendo-toolbar>
      </div>
    </div>
  `,
  encapsulation: ViewEncapsulation.None
})
```

```
(valueChange)="onImageUpload($event)">
    Upload your signature:
</upload-tool>
<kendo-toolbar-spacer></kendo-toolbar-spacer>
<kendo-toolbar-butongroup selection="single">
    <kendo-toolbar-button
        [toggleable]="true"
        [selected]="!showUpload"
        (click)="showUpload = false"
        [svgIcon]="brushSvg"
        text="Draw"
    >
        Draw
    </kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable]="true"
        [selected]="showUpload"
        (click)="showUpload = true"
        [svgIcon]="uploadSvg"
        text="Upload"
    >
    </kendo-toolbar-button>
</kendo-toolbar-butongroup>
</kendo-toolbar>

<div class="signature-wrapper">
    <kendo-signature
        #signature
        *ngIf="!showUpload"
        [(value)]="value"
        [color]="color"
        [backgroundColor]="backgroundColor"
        [strokeWidth]="strokeWidth"
        [smooth]="true"
        [maximizable]="false"
        [hideLine]="true"
    ></kendo-signature>

    <ng-container *ngIf="showUpload">
        <div *ngIf="!imageURL" class="placeholder">
            <kendo-svgicon [icon]="imageIcon" size="xlarge"></kendo-
        <img
            *ngIf="imageURL"
            [src]="imageURL"
            title="Uploaded signature"
            draggable="false"
        />
    </ng-container>
```

```
</div>

<div class="notes">
  By using the Kendo UI for Angular signature component, you can
enable
  your end-users to draw handwritten signatures using touch or
pointer
  devices.
</div>

<kendo-toolbar>
  <kendo-toolbar-button
    text="Save"
    [svgIcon]="saveSvg"
    themeColor="primary"
    [disabled]="!value"
    (click)="onSave()"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button text="Clear" (click)="onClear()">
  </kendo-toolbar-button>
</kendo-toolbar>
</div>
</div>
``,
encapsulation: ViewEncapsulation.None,
styles: [
  .content-wrapper,
  .content-wrapper img {
    width: 750px;
    height: 250px;
  }
  .example {
    display: flex;
    justify-content: center;
  }
  .k-signature,
  .notes {
    border-color: rgba(0, 0, 0, 0.08);
    border-width: 0 1px;
    border-radius: 0;
  }
  .k-upload .k-dropzone {
    padding: 0;
  }
  .signature-wrapper {
```

```
        height: 270px;
    }

.signature-wrapper kendo-signature {
    width: 100%;
    height: 100%;
}

.signature-wrapper img {
    user-select: none;
}

.placeholder {
    height: 100%;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
}

.notes {
    font-style: italic;
    border-width: 1px;
    border-bottom-width: 0;
    border-style: solid;
    padding: 1em;
}
```,
],
})
export class AppComponent implements OnDestroy {
    public value = "";
    public showUpload = false;
    public brushSvg: SVGIcon = brushIcon;
    public uploadSvg: SVGIcon = uploadIcon;
    public saveSvg: SVGIcon = saveIcon;
    public imageIcon: SVGIcon = imageIcon;

    // Use theme colors
    public color = "";
    public backgroundColor = "";
    public strokeWidth = 3;

    public sizes = [
        {
            text: "Normal",
            click: () => (this.strokeWidth = 1),
        },
        {
            text: "Wide",
            click: () => (this.strokeWidth = 3),
        }
    ]
}
```

```
    },  
];  
  
public imageURL?: SafeUrl;  
private rawImageURL?: string;  
  
constructor(private sanitizer: DomSanitizer) {}  
  
public ngOnDestroy() {  
  this.cleanupImage();  
}  
  
public onSave() {  
  saveAs(this.value, "signature.png");  
}  
  
public onClear() {  
  this.value = "";  
  this.cleanupImage();  
}  
  
public onImageUpload(file: File) {  
  this.cleanupImage();  
  
  this.readImage(file);  
  this.rawImageURL = URL.createObjectURL(file);  
  this.imageURL =  
this.sanitizer.bypassSecurityTrustUrl(this.rawImageURL);  
}  
  
private cleanupImage() {  
  if (this.rawImageURL) {  
    URL.revokeObjectURL(this.rawImageURL);  
    this.imageURL = undefined;  
    this.rawImageURL = "";  
  }  
}  
  
public readImage(file: File) {  
  const reader = new FileReader();  
  
  const onLoad = () => {  
    this.value = reader.result as string;  
    reader.removeEventListener("load", onLoad);  
  };  
  
  reader.addEventListener("load", onLoad);  
  reader.readAsDataURL(file);  
}  
}
```

Collapse Code ^

# Key Features

- [Appearance](#)—All Kendo UI for Angular Inputs enable you to set their appearance settings.
- [Line Smoothing](#)—The Signature can smooth out the user's drawing for better appearance.
- [Value Binding](#)—The Signature supports two-way binding to images.
- [Forms support](#)—You can use the Signature both in template-driven and reactive Angular forms.
- [Disabled Signature](#)—You can use the configuration options of the Signature to disable the component so that users are not able to interact with it.
- [Read-only Signature](#)—The Signature provides an option for overriding its default active state.

# Support and Learning Resources

- [Signature Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the Signature](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Signature Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)



# Kendo UI for Angular Slider Overview

The Kendo UI for Angular Slider enables the user to increase, decrease, and select predefined values by dragging its handle along the track, or by clicking its side arrow buttons.

The following example demonstrates the Slider in action.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { Component } from "@angular/core";

@Component({
  selector: "my-app",
  template: `
    <div style="text-align: center;">
      <kendo-label
        class="k-display-block"
        [for]="balance"
        text="Balance"
      ></kendo-label>
      <kendo-slider
        #balance
        [fixedTickWidth]="10"
        [min]="-10"
        [max]="10"
        [smallStep]="smallStep"
        [(ngModel)]="valueHorizontal"
      >
        <kendo-slider-messages
          increment="Right"
          decrement="Left"
        ></kendo-slider-messages>
      </kendo-slider>
      <div>{{ valueHorizontal }}</div>
    </div>
  `)
```

```
<div style="height: 300px; text-align: center;">
  <kendo-label
    class="k-display-block"
    [for]="equalizer"
    text="EQUALIZER"
  ></kendo-label>
  <div class="eqSlider">
    <kendo-slider
      #equalizer
      [vertical]="true"
      [showButtons]="false"
      [min]="min"
      [max]="max"
      [smallStep]="smallStep"
      [largeStep]="largeStep"
      [(ngModel)]="valuesVertical[0]"
    ></kendo-slider>
    <p>{{ valuesVertical[0] }}</p>
  </div>
  <div class="eqSlider">
    <kendo-slider
      #equalizer
      [vertical]="true"
      [showButtons]="false"
      [min]="min"
      [max]="max"
      [smallStep]="smallStep"
      [largeStep]="largeStep"
      [(ngModel)]="valuesVertical[1]"
    ></kendo-slider>
    <p>{{ valuesVertical[1] }}</p>
  </div>
  <div class="eqSlider">
    <kendo-slider
      aria-label="Equalizer"
      [vertical]="true"
      [showButtons]="false"
      [min]="min"
      [max]="max"
      [smallStep]="smallStep"
      [largeStep]="largeStep"
      [(ngModel)]="valuesVertical[2]"
    ></kendo-slider>
    <p>{{ valuesVertical[2] }}</p>
  </div>
  <div class="eqSlider">
    <kendo-slider
      aria-label="Equalizer"
      [vertical]="true"
      [showButtons]="false"
      [min]="min"
```

```
[max]="max"
[smallStep]="smallStep"
[largeStep]="largeStep"
[(ngModel)]="valuesVertical[3]"
></kendo-slider>
<p>{{ valuesVertical[3] }}</p>
</div>
<div class="eqSlider">
  <kendo-slider
    aria-label="Equalizer"
    [vertical]="true"
    [showButtons]="false"
    [min]="min"
    [max]="max"
    [smallStep]="smallStep"
    [largeStep]="largeStep"
    [(ngModel)]="valuesVertical[4]"
  ></kendo-slider>
  <p>{{ valuesVertical[4] }}</p>
</div>
</div>
,
styles: [
  .eqSlider {
    display: inline-block;
    margin: 2em;
    height: 122px;
    vertical-align: top;
  }
  .eqSlider p {
    padding-top: 10px;
  }
],
})
export class AppComponent {
  public valueHorizontal = 0;
  public valuesVertical: number[] = [10, 5, 0, 10, 15];
  public min = -20;
  public max = 20;
  public largeStep = 20;
  public smallStep = 1;
}
```

Collapse Code ^

# Key Features

- **Orientation**—The Slider allows you to display it in a vertical or horizontal direction.
- **Forms support**—You can use the Slider both in template-driven and reactive Angular forms.
- **Ticks**—The Slider provides configuration options for setting the placement, titles, and width of its ticks, which indicate the values resulting from each incremented predefined step.
- **Predefined steps**—You can use the configuration options of the Slider to define the step which the component uses to increase or decrease its value.
- **Side buttons**—The Slider enables you to show or hide, and set the titles of its side buttons.
- **Disabled Slider**—You can use the configuration options of the Slider to disable the component so that users are not able to interact with it.
- **Read-only Slider**—The Slider provides an option for overriding its default active state.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Accessibility**—The Slider is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The Slider supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [Slider Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the Slider](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)

- [Slider Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Switch Overview

The Kendo UI for Angular Switch enables the user to toggle between checked and unchecked states.

The following example demonstrates the Switch in action.

EXAMPLE VIEW SOURCE

```
import { Component, ViewEncapsulation } from
"@angular/core";

@Component({
  selector: "my-app",
  template: `
    <kendo-label
      class="k-display-block"
      [for]="switch"
      text="Toggle Switch"
    ></kendo-label>
    <kendo-switch #switch
      [(ngModel)]="checked"></kendo-switch>
  `,
})
export class AppComponent {
  public checked = true;
}
```

Collapse Code ^

**CONFIGURATOR**

File

app.component.ts

app.module.ts

## Key Features

- **Labels**—The Switch provides options for setting and styling the titles of its labels.

- **Forms support**—You can use the Switch both in template-driven and reactive Angular forms.
- **Checked Switch**—You can use the configuration options of the Switch to set its initial value.
- **Disabled Switch**—You can use the configuration options of the Switch to disable the component so that users are not able to interact with it.
- **Read-only Switch**—The Switch provides an option for overriding its default active state.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Accessibility**—The Switch is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The Switch supports a number of keyboard shortcuts for processing various commands.

To learn more about the appearance, anatomy, and accessibility of the Switch, visit the [Progress Design System documentation](#)—an information portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

## Support and Learning Resources

- [Switch Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the Switch](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Switch Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular TextArea Overview

The Kendo UI for Angular TextArea provides highly customizable interface for displaying multi-line text.

You can integrate the component in forms or use it as a standalone item, and also customize its appearance by rendering elements such as buttons and icons.

The following example demonstrates the TextArea in action.

EXAMPLE

VIEW SOURCE

⋮

```
import { Component, ViewEncapsulation } from
"@angular/core";

@Component({
  selector: "my-app",
  styleUrls: ["./profile-skeleton-
app.component.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="profile-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column no-flex">
            <div class="sidebar-container k-
skeleton">
              <div class="avatar-name-
container">
                <div class="k-skeleton
`
```

CONFIGURATOR

File

app.component.ts

app.module.ts

profile-skeleton-app.compo
nent.css

```
skeleton-avatar"></div>
    <div class="name-container">
        <div class="k-skeleton
skeleton-text"></div>
            <div class="k-skeleton
skeleton-small-text-short"></div>
                </div>
            </div>
        <div class="description-
container">
            <div class="k-skeleton
skeleton-small-text"></div>
            <div class="k-skeleton
skeleton-small-text"></div>
            <div class="k-skeleton
skeleton-small-text"></div>
                </div>
            </div>
        </div>
        <div class="card-column">
            <div class="avatar-title-
container">
                <div class="k-skeleton
skeleton-avatar"></div>
                <h4 class="k-h4">My
Profile</h4>
            </div>
            <div class="component-container">
                <kendo-label text="About me">
                    <kendo-textarea
                        placeholder="Tell us a
little bit about yourself...">
                        [rows]="3"
                        resizable="vertical"
                    </kendo-textarea>
                </kendo-label>
            </div>
            <div class="skeleton-container
top">
                <div class="k-skeleton
skeleton-box-small"></div>
                <div class="k-skeleton
skeleton-box-large"></div>
            </div>
            <div class="skeleton-container
bottom">
                <div class="k-skeleton
skeleton-box-small"></div>
                <div class="k-skeleton
skeleton-box-large"></div>
            </div>
```

```
        </div>
      </div>
    </div>
  </div>
  ,
  encapsulation: ViewEncapsulation.None, //  
Encapsulation is disabled for demo purposes  
only.  
})  
export class AppComponent {  
  Collase Code ^
```

# Key Features

- **Character counter**—You can display a character counter for the `TextArea` to restrict the user input to a specified length.
- **Adornments**—The `TextArea` enables you to display custom items, such as buttons, icons, and more, as additional decorations for the component.
- **Sizing**—You can use the `TextArea` configuration options to specify its initial width and height, its directions for resizing or fixed size, and more.
- **Forms support**—You can use the `TextArea` both in template-driven and reactive Angular forms.
- **Disabled `TextArea`**—You can use the configuration options of the `TextArea` to disable the component so that users are not able to interact with it.
- **Read-Only `TextArea`**—The `TextArea` provides an option for overriding its default active state.
- **`TextArea` directive**—The `TextArea` component delivers the `TextArea` directive which provides options for styling and auto-resizing `textarea` elements.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Accessibility**—The `TextArea` is accessible for screen readers and supports WAI-ARIA attributes.

To learn more about the appearance, anatomy, and accessibility of the `TextArea`, visit the [Progress Design System documentation](#)—an information

portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

# Support and Learning Resources

- [TextArea Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the TextArea](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [TextArea Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular TextBox Overview

The Kendo UI for Angular TextBox provides options for creating composite inputs that you can integrate within forms or use as standalone items.

The following example demonstrates the TextBox in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";

@Component({
  selector: "my-app",
  styleUrls: ["./profile-skeleton-app.component.css"],
  styles: [
    `

      my-app {
        padding-bottom: 0;
      }
    `,
  ],
  template: `
    <div class="profile-demo card-container">
      <div class="k-card custom-card">
        <div class="card-row">
          <div class="card-column no-flex">
            <div class="sidebar-container k-skeleton">
              <div class="avatar-name-container">
                <div class="k-skeleton skeleton-avatar"></div>
                <div class="name-container">
                  <div class="k-skeleton skeleton-text"></div>
                  <div class="k-skeleton skeleton-small-text-short">
                    <div>
                      </div>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
        <div class="description-container">
          <div class="k-skeleton skeleton-small-text"></div>
        </div>
      </div>
    </div>
  `
}
```

```

        <div class="k-skeleton skeleton-small-text"></div>
        <div class="k-skeleton skeleton-small-text"></div>
    </div>
</div>
<div class="card-column">
    <div class="avatar-title-container">
        <div class="k-skeleton skeleton-avatar"></div>
        <h4 class="k-h4">My Profile</h4>
    </div>
    <div class="component-container">
        <kendo-label text="Name">
            <kendo-textbox placeholder="John Smith"></kendo-textbox>
        </kendo-label>
    </div>
    <div class="skeleton-container top">
        <div class="k-skeleton skeleton-box-small"></div>
        <div class="k-skeleton skeleton-box-large"></div>
    </div>
    <div class="skeleton-container bottom">
        <div class="k-skeleton skeleton-box-small"></div>
        <div class="k-skeleton skeleton-box-large-double"></div>
    </div>
    </div>
</div>
</div>
</div>
,
encapsulation: ViewEncapsulation.None, // Encapsulation is disabled
for demo purposes only.
})
export class AppComponent {}
```

[Collapse Code ^](#)

# Key Features

- **Character counter**—You can display a character counter for the TextBox to restrict the user input to a specified length.
- **Validation icons**—You can indicate a successful or erroneous operation by using the built-in validation icons of the TextBox or create custom ones.
- **Adornments**—The TextBox enables you to display custom items as prefix or suffix adornments and supports a set of built-in adornments such as a **Clear** button and more.

- **Forms support**—You can use the TextBox both in template-driven and reactive Angular forms.
- **Disabled TextBox**—You can use the configuration options of the TextBox to disable the component so that users are not able to interact with it.
- **Read-Only Textbox**—The TextBox provides an option for overriding its default active state.
- **TextBox directive**—The TextBox component delivers the TextBox directive which provides options for styling `input` elements.
- **Debouncing value changes**—All Kendo UI for Angular Inputs enable you to implement a slight delay before they accept a new input value.
- **Appearance**—All Kendo UI for Angular Inputs enable you to set their dimensions.
- **Globalization**—All Kendo UI for Angular Inputs provide globalization options.
- **Accessibility**—The TextBox is accessible for screen readers and supports WAI-ARIA attributes.

To learn more about the appearance, anatomy, and accessibility of the TextBox, visit the [Progress Design System documentation](#)—an information portal offering rich component usage guidelines, descriptions of the available style variables, and globalization support details.

## Support and Learning Resources

- [TextBox Homepage](#)
- [Getting Started with the Kendo UI for Angular Inputs](#)
- [API Reference of the TextBox](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [TextBox Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular FloatingLabel Overview

The Kendo UI for Angular FloatingLabel enables you to provide a floating label functionality to focusable form controls, which could be Angular components and HTML input elements.

The following example demonstrates the FloatingLabel in action.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { Component } from "@angular/core";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_FLOATINGLABEL } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_INPUTS, KENDO_FLOATINGLABEL],
  template: `
    <kendo-floatinglabel text="Enter Age">
      <kendo-numerictextbox [style.width.px]="180"></kendo-
numerictextbox>
    </kendo-floatinglabel>
  `,
})
export class AppComponent {}
```

[Collapse Code ^](#)

## Key Features

- **Association**—You can apply a floating label functionality to Kendo UI for Angular or other Angular components, or HTML input elements.
- **Optional text**—The FloatingLabel enables you to toggle a chunk of text, which indicates that a form field is optional.
- **Styling**—With the FloatingLabel, you can set custom CSS styles and classes on the actual label element.

## Support and Learning Resources

- [FloatingLabel Homepage](#)
- [Getting Started with the Kendo UI for Angular Labels](#)
- [API Reference of the FloatingLabel Component](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [FloatingLabel Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Labels

This guide provides the information you need to start using the Kendo UI for Angular Labels—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#).

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

EXAMPLE

VIEW SOURCE

⋮

CONFIGURATOR

File

[app.component.ts](#)

```
import { Component } from "@angular/core";
import { KENDO_FLOATINGLABEL } from
"@progress/kendo-angular-label";
import { KENDO_INPUTS } from
"@progress/kendo-angular-inputs";

@Component({
  standalone: true,
  imports: [KENDO_FLOATINGLABEL,
  KENDO_INPUTS],
  selector: "my-app",
```

```
template: `<kendo-floatinglabel text="Enter Age">
  <kendo-numerictextbox
[style.width.px]="180"></kendo-
numerictextbox>
</kendo-floatinglabel>
` ,
})
export class AppComponent {}
```

[Collapse Code ^](#)

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Labels package:

1. Run the following command.

```
ng add @progress/kendo-angular-label
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-label` package as a dependency to the `package.json` file.
- Add all required peer dependencies to the `package.json` file.
- Register the Kendo UI Default theme in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the required components, directives and utility arrays:

The utility arrays are available starting from [v16.6.0](#). If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

- To add all components from the Labels package, import the `KENDO_LABEL` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_LABEL } from '@progress/kendo-angular-label';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_LABEL]
})
```

TS

- To add individual Labels components, import the corresponding utility arrays in your standalone component. See the list of [available utility arrays](#).

For example if you only need the FloatingLabel component, import `KENDO_FLOATINGLABEL`.

```
import { Component } from '@angular/core';
import { KENDO_FLOATINGLABEL } from '@progress/kendo-
angular-label';

@Component({
  standalone: true,
  selector: 'my-app',
```

TS

```
    imports: [KENDO_FLOATINGLABEL]
  })
```

# Using the Components

1. After successfully installing the Labels package and importing the required components and directives, add the corresponding tags of the components you need in the `app.component.html`. For example, if you need the `FloatingLabel`, add the following code:

```
<kendo-floatinglabel text="Enter Age">
  <kendo-numerictextbox></kendo-numerictextbox>
</kendo-floatinglabel>
```

HTML

2. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

3. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular `FloatingLabel` component on the page.

# Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

# Next Steps

- [Labels Dependencies & Standalone Utilities](#)
- [Label Overview](#)
- [FloatingLabel Overview](#)
- [Globalization](#)
- [Labels API Documentation](#)

# Learning Resources

- [Labels Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Kendo UI for Angular Label Overview

The Kendo UI for Angular Label component enables you to associate `labels` with focusable Kendo UI for Angular components or regular HTML elements.

The following example demonstrates the Label in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import {
  ReactiveFormsModule,
  FormGroup,
  FormControl,
  Validators,
} from "@angular/forms";

import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_LABEL } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [ReactiveFormsModule, KENDO_INPUTS, KENDO_BUTTONS,
    KENDO_LABEL],
  template: `
    <div class="example">
      <form class="k-form" [formGroup]="form">
        <h5>COURSE REGISTRATION FORM</h5>
        <kendo-label text="Full Name">
          <kendo-textbox formControlName="fullName"></kendo-textbox>
        </kendo-label>
        <kendo-label text="Email">
          <kendo-textbox formControlName="email" #email></kendo-textbox>
        </kendo-label>
        <kendo-label style="margin-bottom: -10px" [for]="#age">
          <kendo-select formControlName="age"></kendo-select>
        </kendo-label>
      </form>
    </div>
  `)
```

```
        text="Age">
            </kendo-label>
            <kendo-numerictextbox
                #age
                formControlName="age"
                format="n0"
            ></kendo-numerictextbox>
            <div class="k-form-buttons">
                <button
                    kendoButton
                    themeColor="primary"
                    [disabled]="!form.valid"
                    (click)="submitForm()"
                >
                    Register
                </button>
                <button kendoButton (click)="clearForm()">Clear</button>
            </div>
        </form>
    </div>
    ,
    encapsulation: ViewEncapsulation.None,
    styles: [
        .
        .example {
            display: flex;
            justify-content: center;
        }

        form {
            display: flex;
            flex-direction: column;
            justify-content: center;
            gap: 10px;
        }

        .k-form-buttons {
            margin-top: 15px;
        }
    ],
})
export class AppComponent {
    public form: FormGroup;

    public data = {
        fullName: "",
        email: "",
        age: 0,
    };
}
```

```
constructor() {
    this.form = new FormGroup({
        fullName: new FormControl(this.data.fullName,
[Validators.required]),
        email: new FormControl(this.data.email, [
            Validators.required,
            Validators.email,
        ]),
        age: new FormControl(this.data.age, [
            Validators.required,
            Validators.min(18),
        ]),
    });
}

public submitForm(): void {
    console.log(this.form.value);
}

public clearForm(): void {
    this.form.reset();
}
}
```

Collanse Code ^

## Key Features

- **Association**—You can associate the Label with a Kendo UI for Angular or other Angular component, as well as with an HTML element.
- **Optional Text**—The Label component allows you to indicate a form field as optional.

## Support and Learning Resources

- [Label Homepage](#)
- [Getting Started with the Kendo UI for Angular Labels](#)
- [API Reference of the Label](#)
- [API Reference of the Label directive](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)

- [Label Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Angular Labels Overview

The Kendo UI for Angular Labels associate focusable components or HTML elements with an HTML Label element.

The Labels are built from the ground up and specifically for Angular, so that you get high-performance label controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



## FloatingLabel

A component that enables the floating label functionality for Kendo UI and HTML input elements.



## Label

A component that associates a focusable component or HTML element with a HTML Label element.

# Angular Labels Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import {
  ReactiveFormsModule,
  FormGroup,
  FormControl,
  Validators,
} from "@angular/forms";

import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_LABEL } from "@progress/kendo-angular-label";
```

```
@Component({
  selector: "my-app",
  standalone: true,
  imports: [ReactiveFormsModule, KENDO_INPUTS, KENDO_BUTTONS,
KENDO_LABEL],
  template:
    <div class="example">
      <form class="k-form" [formGroup]="form">
        <h5>COURSE REGISTRATION FORM</h5>
        <kendo-label text="Full Name">
          <kendo-textbox formControlName="fullName"></kendo-textbox>
        </kendo-label>
        <kendo-label text="Email">
          <kendo-textbox formControlName="email" #email></kendo-textbox>
        </kendo-label>
        <kendo-label style="margin-bottom: -10px" [for]="age"
text="Age">
          </kendo-label>
          <kendo-numerictextbox
            #age
            formControlName="age"
            format="n0"
          ></kendo-numerictextbox>
        <div class="k-form-buttons">
          <button
            kendoButton
            themeColor="primary"
            [disabled]="!form.valid"
            (click)="submitForm()"
          >
            Register
          </button>
          <button kendoButton (click)="clearForm()">Clear</button>
        </div>
      </form>
    </div>
  ,
  encapsulation: ViewEncapsulation.None,
  styles: [
    .example {
      display: flex;
      justify-content: center;
    }

    form {
      display: flex;
      flex-direction: column;
      justify-content: center;
      gap: 10px;
    }
  ]
})
```

```

        }

    .k-form-buttons {
        margin-top: 15px;
    }
``,
],
})
export class AppComponent {
    public form: FormGroup;

    public data = {
        fullName: "",
        email: "",
        age: 0,
    };

    constructor() {
        this.form = new FormGroup({
            fullName: new FormControl(this.data.fullName,
[Validators.required]),
            email: new FormControl(this.data.email, [
                Validators.required,
                Validators.email,
            ]),
            age: new FormControl(this.data.age, [
                Validators.required,
                Validators.min(18),
            ]),
        });
    }

    public submitForm(): void {
        console.log(this.form.value);
    }

    public clearForm(): void {
        this.form.reset();
    }
}

```

[Collapse Code ^](#)

## Angular Labels Key Features

Each Kendo UI for Angular Labels component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts

to improve the performance and add more value to the existing Labels library as well as develop new features and controls to it.

## Association

You can use the Labels to associate Kendo UI for Angular components, other Angular components, and regular HTML inputs. [Read more about the association options of the Label component...](#)

## Optional Text

The Labels provide settings for toggling a text in a way it indicates that a form field is optional. [Read more about optional text of the Label...](#)

## Styling

The Labels components allow you to apply custom CSS styles and classes directly to the label element, giving you full control over their appearance and allowing for seamless customization. [Read more about the styling options of the FloatingLabel component...](#)

## Globalization

The Kendo UI for Angular Labels support globalization to ensure that each component can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Labels support rendering in a right-to-left (RTL) direction. [Read more about Labels globalization...](#)

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular Labels are part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no restrictions! What's more, you are eligible for full technical support during your trial

period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

# Support Options

For any questions about the use of Kendo UI for Angular Labels, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

# Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Labels](#)
- [API Reference of the Label](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)

- Knowledge Base

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Avatar Overview

The Kendo UI for Angular Avatar is typically used to display images, icons or initials representing people or other entities.

The following example demonstrates the Avatar in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { SVGIcon, userIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  template: `
    <div class="list">
      <div class="contact-list">Contacts</div>
      <div class="contact" *ngFor="let contact of contactInitials">
        <div class="k-hstack">
          <kendo-avatar [initials]="contact.avatar"></kendo-avatar>
          <div>
            <h2>{{ contact.name }}</h2>
            <p>{{ contact.position }}</p>
          </div>
        </div>
      </div>
      <div class="contact" *ngFor="let contact of contactImages">
        <div class="k-hstack">
          <kendo-avatar [imageSrc]="contact.avatar"></kendo-avatar>
          <div>
            <h2>{{ contact.name }}</h2>
            <p>{{ contact.position }}</p>
          </div>
        </div>
      </div>
      <div class="contact">
        <div class="k-hstack">
```

```
<kendo-avatar [svgIcon]="userSvg"> </kendo-avatar>
<div class="mate-info">
  <h2>Unknown</h2>
  <p>Not specified</p>
</div>
<div class="k-hstack"></div>
</div>
</div>
``,
styles: [
  .contact {
    padding: 8px 14px;
    margin-bottom: 4px;
    box-shadow: 0 1px 2px #ccc;
  }
  .contact h2 {
    font-size: 1.3em;
    font-weight: normal;
    margin: 0;
  }
  .contact p {
    margin: 0;
    font-size: 0.8em;
  }
  .k-hstack div {
    margin-left: 1em;
  }
  .mate-info {
    display: inline-block;
    vertical-align: top;
  }
  .list {
    width: 280px;
    margin: auto;
    padding: 4px 10px;
    box-shadow: 0 1px 5px 0 rgba(0, 0, 0, 0.26),
      0 2px 2px 0 rgba(0, 0, 0, 0.12), 0 3px 1px -2px rgba(0, 0, 0, 0.08);
  }
  .contact-list {
    text-align: center;
    padding: 4px;
    font-size: 20px;
  }
``,
],
})
export class AppComponent {
  public firstContactImage = "assets/dropdowns/contacts/RICSU.jpg";
```

```
public secondContactImage = "assets/dropdowns/contacts/GOURL.jpg";
public userSvg: SVGIcon = userIcon;

public contactImages: Array<{
  avatar: string;
  name: string;
  position: string;
}> = [
  {
    avatar: this.firstContactImage,
    name: "Michael Holz",
    position: "Manager",
  },
  {
    avatar: this.secondContactImage,
    name: "André Stewart",
    position: "Product Manager",
  },
];
}

public contactInitials: Array<{
  avatar: string;
  name: string;
  position: string;
}> = [
  { avatar: "JS", name: "Jason Smith", position: "UX Designer" },
  { avatar: "GP", name: "George Porter", position: "Software Engineer" }
];
}
```

Collapse Code ^

## Key Features

- **Types**—The Avatar provides a set of rendering options such as displaying image, icon, and initials avatars, setting its size, roundness, theme color, and more.
- **Appearance**—The Avatar provides a set of rendering options such as displaying image, icon, and initials avatars, setting its size, roundness, theme color, and more.
- **Globalization**—All Kendo UI for Angular Layout components provide globalization options.

## Support and Learning Resources

- [Avatar Homepage](#)
- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the Avatar](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Avatar Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Card Overview

The Kendo UI for Angular Card represents any type of content and all kinds of actions about a single subject.

It provides clarity, categorization, and an attractive way of presentation.

The following example demonstrates the Card in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component, ViewEncapsulation } from "@angular/core";

export interface MyComment {
    likes: number;
    text: string;
}

export interface MyCardComponent {
    thumbnailSrc: string;
    headerTitle: string;
    headerSubtitle: string;
    commentsExpanded: boolean;
    postLiked: boolean;
    comments: Array<MyComment>;
    newCommentTextValue: string;
    postLikes: number;
    scrollViewItems: Array<Record<string, unknown>>;
}

@Component({
    selector: "my-app",
    template: `
        <div class="card-list">
            <div *ngFor="let card of scrollViewCards">
                <scrollview-card [card]="card"></scrollview-card>
            </div>
    `
```

```
        </div>
    ,
  styles: [
    .card-list {
      display: flex;
      justify-content: space-evenly;
      flex-wrap: wrap;
    }
    .k-textarea {
      resize: none;
    }
    .comment-box {
      display: flex;
      flex-direction: column;
    }
    .card-comment-wrapper .box {
      display: flex;
      justify-content: space-between;
    }
    .card-post-comment-wrapper {
      padding: 16px 16px 0;
    }
    .card-comment-wrapper {
      margin-bottom: 8px;
      padding: 0 16px;
    }
    .card-comment-container {
      padding: 4px 0;
      align-items: center;
      display: flex;
    }
    .comment-box .comment-text {
      font-size: 13px;
      font-weight: bold;
      max-width: 100px;
    }
    .comment-box span {
      font-size: 10px;
      font-weight: normal;
      word-break: break-all;
    }
    .time span {
      margin-left: 8px;
    }
    .box .k-button-icon {
      font-size: 16px;
      align-self: center;
    }
    .post-likes-count {
      font-size: 13px;
    }
  ]
}
```

```
    align-self: center;
    color: #656565;
}
.k-button.k-flat:focus::after {
    display: none;
}
.k-button {
    outline: none;
}
.k-button .k-i-heart {
    color: #ff6358;
}
.k-card-subtitle .k-icon {
    vertical-align: baseline;
    font-size: 12px;
}
.k-card-header .k-card-title + .k-card-subtitle {
    margin-top: 0;
}
.k-card-title {
    margin-bottom: 4px;
}
.k-card-media li img {
    width: 285px;
}
comment-actions {
    width: 100%;
    display: flex;
    justify-content: space-between;
}
.k-card-action {
    flex: 0 0 auto;
}
.k-avatar {
    margin-right: 1em;
}
`,
],
encapsulation: ViewEncapsulation.None,
})
export class AppComponent {
public scrollViewCards: Array<MyCardComponent> = [
{
    thumbnailSrc: "assets/layout/card/bg_flag.jpg",
    headerTitle: "bg_traditions",
    headerSubtitle: "Bulgaria, Europe",
    commentsExpanded: false,
    postLiked: false,
    comments: [],
    newCommentTextValue: "",
    postLikes: 674,
```

```
scrollViewItems: [
    { url: "assets/layout/card/kukeri.jpg" },
    { url: "assets/layout/card/martenitsa.jpg" },
    { url: "assets/layout/card/rose_festival.jpg" },
],
},
{
    thumbnailSrc: "assets/layout/card/rila_lakes.jpg",
    headerTitle: "bg_mountains",
    headerSubtitle: "Bulgaria, Europe",
    commentsExpanded: false,
    postLiked: false,
    comments: [],
    newCommentTextValue: "",
    postLikes: 962,
    scrollViewItems: [
        { url: "assets/layout/card/rila.jpg" },
        { url: "assets/layout/card/pamporovo.jpg" },
        { url: "assets/layout/card/camping.jpg" },
    ],
},
];
}
```

Collanse Code ^

## Key Features

- **Building blocks**—The Card consists of content elements related to a single subject which allow you to build multiple layout types and provide tailored project solutions.
- **Orientation**—You can arrange the content of the Card in a vertical or horizontal direction.
- **Media display**—You can display an image or a video within the content of the Card.
- **Action buttons**—The Card allows you to render action buttons and customize their content, orientation, and layout.
- **Custom layout**—The Card enables you to build on top of the default flow of its inner containers by overriding the common CSS property of its building blocks.
- **Globalization**—All Kendo UI for Angular Layout components provide globalization options.

## Support and Learning Resources

- [Card Homepage](#)
- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the Card](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Card Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Drawer Overview

The Kendo UI for Angular Drawer is a dismissible navigation panel in responsive web applications.

It also enables the user to change the content of a specific section of the page.

The following example demonstrates the Drawer in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { DrawerItem, DrawerSelectEvent } from "@progress/kendo-angular-layout";
import {
  SVGIcon,
  bellIcon,
  calendarIcon,
  envelopeLinkIcon,
  inboxIcon,
  menuIcon,
  starOutlineIcon,
} from "@progress/kendo-svg-icons";

@Component({
  encapsulation: ViewEncapsulation.None,
  selector: "my-app",
  styles: [
    `
      html,
      body,
      my-app {
        padding: 0;
        height: 100%;
      }
    `,
    `
```

```
        display: flex;
        flex-direction: column;
    }

    kendo-drawer-container {
        flex: 1 1 auto;
        overflow-y: auto;
    }
    .k-icon {
        font-size: 20px;
    }
    .custom-toolbar {
        width: 100%;
        background-color: #f6f6f6;
        line-height: 10px;
        border-bottom: inset;
        border-bottom-width: 1px;
        padding: 3px 8px;
        color: #656565;
    }
    .mail-box {
        margin-left: 20px;
        font-weight: bold;
        font-size: 17px;
    }
}
],
template: `
<div class="custom-toolbar">
    <button
        kendoButton
        [svgIcon]="menuSvg"
        fillMode="flat"
        (click)="drawer.toggle()"
    ></button>
    <span class="mail-box">Mail Box</span>
</div>
<kendo-drawer-container>
    <kendo-drawer
        #drawer
        [items]="items"
        mode="push"
        [mini]="true"
        [expanded]="true"
        (select)="onSelect($event)"
    >
    </kendo-drawer>

    <kendo-drawer-content>
        <my-content [selectedItem]="selected"></my-content>
    </kendo-drawer-content>

```

```

        </kendo-drawer-container>
      ,
    })
export class AppComponent {
  public selected = "Inbox";
  public menuSvg: SVGIcon = menuIcon;

  public items: Array<DrawerItem> = [
    { text: "Inbox", svgIcon: inboxIcon, selected: true },
    { separator: true },
    { text: "Notifications", svgIcon: bellIcon },
    { text: "Calendar", svgIcon: calendarIcon },
    { separator: true },
    { text: "Attachments", svgIcon: envelopeLinkIcon },
    { text: "Favourites", svgIcon: starOutlineIcon },
  ];
}

public onSelect(ev: DrawerSelectEvent): void {
  this.selected = ev.item.text;
}
}

```

[Collapse Code ^](#)

# Key Features

- **Display modes**—The Drawer provides overlay and push expand modes as well as a compact mini view.
- **Interaction modes**—You can set the initial expanded state of the Drawer and then change it on demand by using the available API options.
- **Positioning**—The Drawer enables you to specify its position with regard to the page content.
- **Templates**—You can apply custom styles to the Drawer by using templates for its collection of items, header and footer, and other elements.
- **Routing**—The Drawer can utilize the Angular Router in this way enabling you to use it as a navigational component.
- **Hierarchical Drawer**—Even though by design the Drawer works with a flat structure of items, it provides options for hierarchical data navigation.
- **Globalization**—All Kendo UI for Angular Layout components provide globalization options.

- **Accessibility**—The Drawer is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The Drawer supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [Drawer Homepage](#)
- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the Drawer](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Drawer Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular ExpansionPanel Overview

The Kendo UI for Angular ExpansionPanel provides a details-summary view that enables the user to expand or collapse the content.

The following example demonstrates the ExpansionPanel in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { ExpansionPanelComponent } from "@progress/kendo-angular-layout";
import { Component, QueryList, ViewChildren } from "@angular/core";
import { countries } from "./countries";

@Component({
  selector: "my-app",
  template: `
    <div class="wrapper">
      <kendo-expansionpanel
        *ngFor="let item of items; index as i"
        [title]="item.country"
        [subtitle]="item.continent"
        [expanded]="item.expanded"
        (action)="onAction(i)"
      >
        <div class="content">
          <div class="image-container">
            <img [src]="item.imageUrl" [alt]="item.country" />
          </div>
          <span class="content-text">{{ item.text }}</span>
        </div>
      </kendo-expansionpanel>
    </div>
  `,
  styleUrls: ['./styles.css'],
})
```

```
export class AppComponent {  
    @ViewChildren(ExpansionPanelComponent)  
    panels: QueryList<ExpansionPanelComponent>;  
  
    public items = countries;  
  
    public onAction(index: number): void {  
        this.panels.forEach((panel, idx) => {  
            if (idx !== index && panel.expanded) {  
                panel.toggle();  
            }  
        });  
    }  
}
```

Collapse Code ^

# Key Features

- **Title**—The ExpansionPanel enables you to define the content inside its header by using a title, subtitle, or a title template.
- **Interaction modes**—You can set the initial expanded state of the ExpansionPanel and then change it on demand by using the available API options.
- **Animations**—You can define the duration of the ExpansionPanel expand and collapse animations, and also fully disable the animations.
- **Icons**—The ExpansionPanel enables you to customize its expand and collapse icon indicators.
- **Disabled ExpansionPanel**—You can use the configuration options of the ExpansionPanel to disable the whole group of buttons or a single button so that users are not able to interact with it.
- **Globalization**—All Kendo UI for Angular Layout components provide globalization options.
- **Accessibility**—The ExpansionPanel is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The ExpansionPanel supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [ExpansionPanel Homepage](#)
- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the ExpansionPanel](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [ExpansionPanel Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular GridLayout Overview

The Kendo UI for Angular GridLayout allows you to easily arrange its contents in rows and columns, forming a grid structure. It is based on the CSS Grid Layout system, which allows setting the specific row, column and size of each item.

The following example demonstrates the GridLayout in action.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { Component } from "@angular/core";
import { tags, articles, recommendedArticles } from "./articles-info";

@Component({
  selector: "my-app",
  template: `
    <div class="grid-layout-container">
      <kendo-gridlayout
        [gap]="{ rows: 6, cols: 10 }"
        [rows]="[
          { height: 20 },
          { height: 100 },
          { height: 100 },
          { height: 20 },
          { height: 660 }
        ]"
        [cols]="[{ width: 270 }, { width: 270 }, { width: 310 }]"
      >
        <kendo-gridlayout-item [row]="1" [col]="1" [colSpan]="3">
          <div class="k-text-inverse k-text-uppercase k-font-weight-bold">
            Trending articles
          </div>
        </kendo-gridlayout-item>
        <kendo-gridlayout-item [row]="2" [col]="1">
          <trending-article
            [position]="1"

```

```
        [article]="articles[0]"
      ></trending-article>
</kendo-gridlayout-item>
<kendo-gridlayout-item [row]="2" [col]="2">
  <trending-article
    [position]="2"
    [article]="articles[1]"
  ></trending-article>
</kendo-gridlayout-item>
<kendo-gridlayout-item [row]="2" [col]="3">
  <trending-article
    [position]="3"
    [article]="articles[2]"
  ></trending-article>
</kendo-gridlayout-item>
<kendo-gridlayout-item [row]="3" [col]="1">
  <trending-article
    [position]="4"
    [article]="articles[3]"
  ></trending-article>
</kendo-gridlayout-item>
<kendo-gridlayout-item [row]="3" [col]="2">
  <trending-article
    [position]="5"
    [article]="articles[4]"
  ></trending-article>
</kendo-gridlayout-item>
<kendo-gridlayout-item [row]="3" [col]="3">
  <trending-article
    [position]="6"
    [article]="articles[5]"
  ></trending-article>
</kendo-gridlayout-item>
<kendo-gridlayout-item [row]="4" [col]="1" [colSpan]="2">
  <div class="k-text-inverse k-text-uppercase k-font-weight-
bold">
    Recommended for you
  </div>
</kendo-gridlayout-item>
<kendo-gridlayout-item [row]="5" [col]="1" [colSpan]="2">
  <recommended-article
    *ngFor="let article of recommendedArticles"
    [article]="article"
  ></recommended-article>
</kendo-gridlayout-item>
<kendo-gridlayout-item [row]="4" [col]="3">
  <div class="k-text-inverse k-text-uppercase k-font-weight-
bold">
    Events this month
  </div>
</kendo-gridlayout-item>
```

```
<kendo-gridlayout-item [row]="5" [col]="3">
  <kendo-calendar
    type="infinite"
    class="event-calendar"
    [(value)]="selectedDate"
    (valueChange)="handleValueChange()"
    bottomView="year"
    topView="decade"
    [navigation]="false"
  >
</kendo-calendar>
<div
  class="k-text-inverse k-text-uppercase k-font-weight-bold k-
mt-4">
  >
  Discover more
</div>
<kendo-chiplist class="k-flex-wrap" selection="multiple">
  <kendo-chip
    *ngFor="let tag of tags"
    [label]="tag.name"
    fillMode="outline"
    >
    </kendo-chip>
  </kendo-chiplist>
</kendo-gridlayout-item>
</kendo-gridlayout>
</div>
,
styles: [
  .grid-layout-container {
    overflow-x: auto;
  }

  .k-grid-layout {
    width: 870px;
    margin: auto;
  }

  .k-chip-list {
    padding: 5px 0px;
    grid-gap: 6px;
    margin-top: 8px;
  }

  .k-chip {
    margin: 0px;
    border-radius: 2px;
  }
]
```

```
    .event-calendar {
      width: 100%;
    }
  ],
})
export class AppComponent {
  public tags = tags;
  public articles = articles;
  public recommendedArticles = recommendedArticles;

  public selectedDate = new Date();

  handleValueChange(): void {
    articles.sort(() => Math.random() - 0.5);
    recommendedArticles.sort(() => Math.random() - 0.5);
  }
}
```

Collanse Code ^

## Key Features

- **Items**—The GridLayout enables you to control the size and position of its items.
- **Rows and Columns**—To configure the appearance of the GridLayout, use the `rows` and `cols` properties.
- **Gaps**—Customize the spacing between the rows and columns of the GridLayout by setting its `gap` property.
- **Alignment**—You can control the alignment of the items in the GridLayout based on the X and Y axes.
- **Globalization**—All Kendo UI for Angular Layout components provide globalization options.

## Support and Learning Resources

- [GridLayout Homepage](#)
- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the GridLayout](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)

- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [GridLayout Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Layout

This guide provides the information you need to start using the Kendo UI for Angular Layout—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#).

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_LAYOUT } from "@progress/kendo-angular-layout";

@Component({
  standalone: true,
  imports: [KENDO_LAYOUT],
  selector: "my-app",
  template: `
    <kendo-panelbar>
      <kendo-panelbar-item title="Teams">
        <kendo-panelbar-item title="Team 1"></kendo-panelbar-item>
      </kendo-panelbar-item>
    </kendo-panelbar>
  `
```

```
    <kendo-panelbar-item title="Team 2"></kendo-panelbar-item>
</kendo-panelbar-item>
<kendo-panelbar-item title="Releases">
    <kendo-panelbar-item title="Q1 2021"> </kendo-panelbar-item>
    <kendo-panelbar-item title="Q2 2021"> </kendo-panelbar-item>
</kendo-panelbar-item>
</kendo-panelbar>
,
})
export class AppComponent {}
```

Collapse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Layout package:

1. Run the following command.

```
ng add @progress/kendo-angular-layout
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-layout` package as a dependency to the `package.json` file.
- Add all required peer dependencies to the `package.json` file.
- Register the Kendo UI Default theme in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the required components, directives and utility arrays:

The utility arrays are available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

- To add all components from the Layout package, import the `KENDO_LAYOUT` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_LAYOUT } from '@progress/kendo-angular-layout';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_LAYOUT]
})
```

TS

- To add individual Layout components, import the corresponding utility arrays in your standalone component. See the list of [available utility arrays](#).

For example if you only need the PanelBar component, import `KENDO_PANELBAR`.

```
import { Component } from '@angular/core';
import { KENDO_PANELBAR } from '@progress/kendo-angular-layout';

@Component({
  standalone: true,
```

TS

```
        selector: 'my-app',
        imports: [KENDO_PANELBAR]
    })
```

# Using the Components

- After successfully installing the Layout package and importing the required components and directives, add the corresponding tags of the components you need in the `app.component.html`. For example, if you need the PanelBar, add the following code:

```
<kendo-panelbar>
  <kendo-panelbar-item title="Teams">
    <kendo-panelbar-item title="Team 1"></kendo-panelbar-item>
    <kendo-panelbar-item title="Team 2"></kendo-panelbar-item>
  </kendo-panelbar-item>
  <kendo-panelbar-item title="Releases">
    <kendo-panelbar-item title="Q1 2021"> </kendo-panelbar-item>
    <kendo-panelbar-item title="Q2 2021"> </kendo-panelbar-item>
  </kendo-panelbar-item>
</kendo-panelbar>
```

HTML

- Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

- Point your browser to <http://localhost:4200> to see the Kendo UI for Angular PanelBar component on the page.

# Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key.

If your application does not contain a Kendo UI license file, [activate your license key](#).

# Next Steps

- [Layout Dependencies & Standalone Utilities](#)
- [Avatar Overview](#)
- [Card Overview](#)
- [Drawer Overview](#)
- [ExpansionPanel Overview](#)
- [GridLayout Overview](#)
- [PanelBar Overview](#)
- [Splitter Overview](#)
- [StackLayout Overview](#)
- [Stepper Overview](#)
- [TabStrip Overview](#)
- [TileLayout Overview](#)
- [Timeline Overview](#)
- [Globalization](#)
- [Layout API Documentation](#)

# Learning Resources

- [Layout Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Angular Layout Overview

The Kendo UI for Angular Layout components create a perceptive and intuitive layout for web projects and provide for an easier navigation.

They include a variety of layout types and styles that have extensive configuration options. This flexibility allows you to quickly and easily create the exact layout control you need to fit your specific requirements for functionality and appearance.

The Layout components are built from the ground up and specifically for Angular, so that you get high-performance input controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



**Avatar**

A component used to represent people or entities



**Card**

A component which displays content built of different elements



**Drawer**

A dismissible navigation panel



**ExpansionPanel**

An expandable details-summary view component



**GridLayout**

A component with rows and columns, based on the CSS grid-layout system



**PanelBar**

A multi-level component for hierarchical data



## Splitter

A component for resizable layouts



## Stepper

A step collection for sequential layout



## TileLayout

A collection of resizable and reorderable tiles



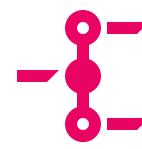
## StackLayout

A component for aligning elements in a stack, based on the CSS flexbox system



## TabStrip

A tab collection for associated content



## Timeline

A collection of events in a chronological succession for each year

# Angular Layout Example

EXAMPLE

VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { DrawerSelectEvent } from "@progress/kendo-angular-layout";
import {
  SVGIcon,
  alignJustifyIcon,
  cellsMergeVerticallyIcon,
  colResizeIcon,
  connectorIcon,
  gridIcon,
  imageIcon,
```

```
insertTopIcon,
layoutIcon,
listOrderedIcon,
menuIcon,
thumbnailsUpIcon,
userIcon,
} from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  template: `
    <div class="wrapper">
      <div class="custom-toolbar">
        <button
          kendoButton
          [svgIcon]="layoutSvg"
          fillMode="flat"
          (click)="drawer.toggle()"
        >
          <span class="mail-box">Drawer component</span>
        </button>
      </div>
      <kendo-drawer-container class="customClass">
        <kendo-drawer
          #drawer
          [items]="items"
          mode="push"
          [mini]="true"
          [expanded]="true"
          (select)="onSelect($event)"
        >
        </kendo-drawer>
        <kendo-drawer-content>
          <layout-component [selectedItem]="selected"></layout-
component>
        </kendo-drawer-content>
      </kendo-drawer-container>
    </div>
  `,
  encapsulation: ViewEncapsulation.None,
  styles: [
    html,
    body,
    #parent,
    my-app,
    .wrapper {
      margin: 0;
      padding: 0;
      border-width: 0;
      height: 100%;
    }
  ]
})
```

```
        }
      }

      html {
        overflow: hidden;
      }

      .customClass {
        flex: 1;
        overflow: auto;
        align-items: stretch;
      }

      .wrapper {
        display: flex;
        flex-direction: column;
        height: 100%;
      }

      .customClass .k-drawer-content {
        overflow: auto;
      }

      .custom-toolbar {
        width: 100%;
        background-color: #f6f6f6;
        line-height: 10px;
        border-bottom: inset;
        border-bottom-width: 1px;
        padding: 3px 8px;
        color: #656565;
        flex-shrink: 0;
      }

      .mail-box {
        font-weight: bold;
        font-size: 17px;
      }

    ],
  ],
})
export class AppComponent {
  public selected = "Avatar";
  public layoutSvg: SVGIcon = layoutIcon;

  public items: Array<
    | { text: string; svgIcon: SVGIcon; selected?: boolean }
    | { separator: boolean }
  > = [
    { text: "Avatar", svgIcon: userIcon, selected: true },
    { separator: true },
    { text: "Card", svgIcon: imageIcon },
    { separator: true },
  ]
}
```

```
{ text: "ExpansionPanel", svgIcon: insertTopIcon },
{ separator: true },
{ text: "GridLayout", svgIcon: cellsMergeVerticallyIcon },
{ separator: true },
{ text: "PanelBar", svgIcon: menuIcon },
{ separator: true },
{ text: "Splitter", svgIcon: colResizeIcon },
{ separator: true },
{ text: "StackLayout", svgIcon: alignJustifyIcon },
{ separator: true },
{ text: "Stepper", svgIcon: listOrderedIcon },
{ separator: true },
{ text: "TabStrip", svgIcon: thumbnailsUpIcon },
{ separator: true },
{ text: "TileLayout", svgIcon: gridIcon },
{ separator: true },
{ text: "Timeline", svgIcon: connectorIcon },
];
}

public onSelect(ev: DrawerSelectEvent): void {
    this.selected = ev.item.text;
}
}
```

Collanse Code ^

# Angular Layout Key Features

Each Kendo UI for Angular Layout component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Layout library as well as develop new features and controls to it.

## Appearance

The colors of the Layout components are normally picked up by the current Kendo UI theme, but each aspect of the controls can be customized using theme variables or configuration options. Kendo UI for Angular offers a variety of popular themes, including Bootstrap and Material. These themes can be easily tailored to your needs using the [Progress ThemeBuilder](#), an online utility application. [Read more about styling the Avatar...](#)

# Globalization

The Kendo UI for Angular Layout package supports globalization to ensure that each Layout component can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Layout controls support rendering in a right-to-left (RTL) direction. [Read more about Layout globalization...](#)

# Accessibility

Most Layout components are accessible for screen readers and support WAI-ARIA attributes. [Read more about the accessibility support of the Drawer...](#)

# Keyboard Navigation

Apart from the Avatar, Card, and TileLayout, the Layout components support a number of keyboard shortcuts which allow users to accomplish various commands. [Read more about the keyboard navigation of the Drawer...](#)

# Trial Version of Kendo UI for Angular

The Kendo UI for Angular Layout controls are part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

# Support Options

For any questions about the use of Kendo UI for Angular Layout controls, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual

developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).

- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the Layout](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular PanelBar Overview

The Kendo UI for Angular PanelBar displays hierarchical data as a multi-level, expandable component.

To describe its items children, the PanelBar enables you to nest them as **PanelBarItem** components or to set the **items** property.

The following example demonstrates the PanelBar in action.

EXAMPLE

VIEW SOURCE

⋮

```
import { Component } from "@angular/core";

@Component({
  selector: "my-app",
  template: `
    <div class="panelbar-wrapper">
      <kendo-panelbar>
        <kendo-panelbar-item title="My Teammates" [expanded]="true">
          <ng-template kendoPanelBarContent>
            <div>
              <div class="teamMate">
                <img [src]="imageUrl('andrew')" alt="Andrew Fuller" />
                <span class="mate-info">
                  <h2>Andrew Fuller</h2>
                  <p>Team Lead</p>
                </span>
              </div>
              <div class="teamMate">
                <img [src]="imageUrl('nancy')" alt="Nancy Leverling" />
                <span class="mate-info">
                  <h2>Nancy Leverling</h2>
                  <p>Sales Associate</p>
                </span>
              </div>
            </div>
          </ng-template>
        </kendo-panelbar-item>
      </kendo-panelbar>
    </div>
  `
```

```
        <div class="teamMate">
            <img [src]="imageUrl('robert')" alt="Robert King" />
            <span class="mate-info">
                <h2>Robert King</h2>
                <p>Business System Analyst</p>
            </span>
        </div>
    </div>
</ng-template>
</kendo-panelbar-item>
<kendo-panelbar-item title="Projects">
    <kendo-panelbar-item title="New Business Plan"></kendo-
panelbar-item>
    <kendo-panelbar-item title="Sales Forecasts">
        <kendo-panelbar-item title="Q1 Forecast"></kendo-panelbar-
item>
        <kendo-panelbar-item title="Q2 Forecast"></kendo-panelbar-
item>
        <kendo-panelbar-item title="Q3 Forecast"></kendo-panelbar-
item>
        <kendo-panelbar-item title="Q4 Forecast"></kendo-panelbar-
item>
    </kendo-panelbar-item>
    <kendo-panelbar-item title="Sales Reports"></kendo-panelbar-
item>
</kendo-panelbar-item>
<kendo-panelbar-item title="Programs">
    <kendo-panelbar-item title="Monday"></kendo-panelbar-item>
    <kendo-panelbar-item title="Tuesday"></kendo-panelbar-item>
    <kendo-panelbar-item title="Wednesday"></kendo-panelbar-item>
    <kendo-panelbar-item title="Thursday"></kendo-panelbar-item>
    <kendo-panelbar-item title="Friday"></kendo-panelbar-item>
</kendo-panelbar-item>
<kendo-panelbar-item
    title="Communication"
    [disabled]="true"
    ></kendo-panelbar-item>
</kendo-panelbar>
</div>
,
styles: [
    .teamMate:after {
        content: ".";
        display: block;
        height: 0;
        line-height: 0;
        clear: both;
        visibility: hidden;
    }
    .teamMate h2 {
```

```

        font-size: 1.3em;
        font-weight: normal;
        padding-top: 17px;
        margin: 0;
    }
    .teamMate p {
        margin: 0;
        font-size: 0.8em;
    }
    .teamMate img {
        display: inline-block;
        vertical-align: top;
        width: 50px;
        height: 50px;
        margin: 10px;
        border: 1px solid #ccc;
        border-radius: 50%;
    }
    .mate-info {
        display: inline-block;
        vertical-align: top;
    }
    .panelbar-wrapper {
        max-width: 300px;
        margin: 0 auto;
    }
}
],
})
export class AppComponent {
    public imageUrl(imageName: string): string {
        return `assets/layout/panelbar/${imageName}.jpg`;
    }
}

```

[Collapse Code ▲](#)

# Key Features

- **Data binding**—You can bind the PanelBar to an array of `PanelBarItemModel` objects from a local or remote data source.
- **Items**—The PanelBar provides a number of configuration options for its items such as setting their titles and disabled state, rendering icons and images next to them, and more.
- **Expand modes**—The PanelBar supports the single, multiple, and full expand modes.

- [Templates](#)—You can apply templates to instantiate the PanelBar items and customize its styles.
- [Animations](#)—You can control the expand and collapse animations of the PanelBar.
- [Routing](#)—The PanelBar can utilize the Angular Router in this way enabling you to use it as a navigational component.
- [Globalization](#)—All Kendo UI for Angular Layout components provide globalization options.
- [Accessibility](#)—The PanelBar is accessible for screen readers and supports WAI-ARIA attributes.
- [Keyboard navigation](#)—The PanelBar supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [PanelBar Homepage](#)
- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the PanelBar](#)
- [API Reference of the PanelBarItem](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [PanelBar Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Splitter Overview

The Kendo UI for Angular Splitter divides the web page into sections and allows the user to control its layout.

The following example demonstrates the Splitter in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";

@Component({
  selector: "my-app",
  template: `
    <kendo-splitter orientation="vertical" style="height: 340px;">
      <kendo-splitter-pane>
        <kendo-splitter>
          <kendo-splitter-pane [collapsible]="true" size="30%">
            <div class="pane-content">
              <h3>Inner splitter / left pane</h3>
              <p>Resizable and collapsible.</p>
            </div>
          </kendo-splitter-pane>

          <kendo-splitter-pane>
            <div class="pane-content">
              <h3>Inner splitter / center pane</h3>
              <p>Resizable only.</p>
            </div>
          </kendo-splitter-pane>

          <kendo-splitter-pane [collapsible]="true" size="30%">
            <div class="pane-content">
              <h3>Inner splitter / right pane</h3>
              <p>Resizable and collapsible.</p>
            </div>
          </kendo-splitter-pane>
    </kendo-splitter>
  `
```

```

        </kendo-splitter>
    </kendo-splitter-pane>

    <kendo-splitter-pane size="100px">
        <div class="pane-content">
            <h3>Outer splitter / Middle pane</h3>
            <p>Resizable only.</p>
        </div>
    </kendo-splitter-pane>

    <kendo-splitter-pane [resizable]="false" size="100px">
        <div class="pane-content">
            <h3>Outer splitter / Bottom pane</h3>
            <p>Non-resizable and non-collapsible.</p>
        </div>
    </kendo-splitter-pane>
</kendo-splitter>
,
styles: [
    .pane-content {
        padding: 0 10px;
    }
    h3 {
        font-size: 1.2em;
        margin: 10px 0;
        padding: 0;
    }
    p {
        margin: 0;
        padding: 0;
    }
],
})
export class AppComponent {}

```

[Collapse Code ▲](#)

# Key Features

- **Orientation**—The Splitter enables you to arrange its panes horizontally or vertically.
- **Pane settings**—You can control the behavior of the Splitter panes and set their dimensions, resize and collapse them, and more.

- **Globalization**—All Kendo UI for Angular Layout components provide globalization options.
- **Accessibility**—The Splitter is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The Splitter supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [Splitter Homepage](#)
- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the Splitter](#)
- [API Reference of the Splitter Pane](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Splitter Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular StackLayout Overview

The Kendo UI for Angular StackLayout allows you to easily align vertically or horizontally multiple elements in a stack.

The following example demonstrates the StackLayout in action and also showcases other Kendo UI for Angular components such as the ButtonGroup, Avatar and Card components.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import { Orientation } from "@progress/kendo-angular-layout";

@Component({
  selector: "my-app",
  template: `
    <div class="demo-header">
      <span class="demo-title">Popular creators</span>
      <div class="btn-group">
        <kendo-buttongroup selection="single">
          <button
            kendoButton
            [selected]="true"
            [toggleable]="true"
            (click)="toggleOrientation('horizontal')"
          >
            Horizontal
          </button>
          <button
            kendoButton
            [toggleable]="true"
            (click)="toggleOrientation('vertical')"
          >
            Vertical
          </button>
        </kendo-buttongroup>
      </div>
    </div>
  `,
  styles: [
    ".demo-header { border-bottom: 1px solid #ccc; padding-bottom: 10px; margin-bottom: 10px; }"
  ]
})
```

```
        </kendo-buttongroup>
    </div>
</div>
<kendo-stacklayout [gap]="15" [orientation]="orientation">
    <kendo-card width="auto">
        <kendo-card-header class="k-hstack">
            <kendo-avatar
                width="40px"
                height="40px"
                [imageSrc]="lindseyAvatar"
                shape="circle"
            ></kendo-avatar>
            <div>
                <h1 kendoCardTitle>Lindsey Mango</h1>
                <p kendoCardSubtitle>34K followers</p>
            </div>
        </kendo-card-header>
        <img
            class="card-img"
            [src]="lindseyImage"
            kendoCardMedia
            alt="lindsey_img"
        />
        <kendo-card-body>
            <p class="photo-desc">
                Photos by Rikonavt and by Ivana La on Unsplash
            </p>
        </kendo-card-body>
        <kendo-card-actions layout="center">
            <button kendoButton fillMode="flat" themeColor="primary">
                See more from Lindsey
            </button>
        </kendo-card-actions>
    </kendo-card>
    <kendo-card width="auto">
        <kendo-card-header class="k-hstack">
            <kendo-avatar
                width="40px"
                height="40px"
                [imageSrc]="vincenzoAvatar"
                shape="circle"
            ></kendo-avatar>
            <div>
                <h1 kendoCardTitle>Vincenzo Mays</h1>
                <p kendoCardSubtitle>32K followers</p>
            </div>
        </kendo-card-header>
        <img
            class="card-img"
            [src]="vincenzoImage"
            kendoCardMedia
        />
    </kendo-card>
</kendo-stacklayout>
```

```
        alt="vincenzo_img"
    />
<kendo-card-body>
    <p class="photo-desc">
        Photos by Irene Strong and by the BlackRabbit on Unsplash
    </p>
</kendo-card-body>
<kendo-card-actions layout="center">
    <button kendoButton fillMode="flat" themeColor="primary">
        See more from Vincenzo
    </button>
</kendo-card-actions>
</kendo-card>
<kendo-card width="auto">
    <kendo-card-header class="k-hstack">
        <kendo-avatar
            width="40px"
            height="40px"
            [imageSrc]="marissaAvatar"
            shape="circle"
        ></kendo-avatar>
        <div>
            <h1 kendoCardTitle>Marissa Webb</h1>
            <p kendoCardSubtitle>36K followers</p>
        </div>
    </kendo-card-header>
    <img
        class="card-img"
        [src]="marissaImage"
        kendoCardMedia
        alt="marissa_img"
    />
    <kendo-card-body>
        <p class="photo-desc">
            Photos by JJ Jordan and by Silvana Carlos on Unsplash
        </p>
    </kendo-card-body>
    <kendo-card-actions layout="center">
        <button kendoButton fillMode="flat" themeColor="primary">
            See more from Marissa
        </button>
    </kendo-card-actions>
</kendo-card>
</kendo-stacklayout>
`,
styles: [
    .photo-desc {
        text-align: right;
        font-size: 10px;
        font-style: italic;
    }
]
```

```

        }

    .card-img {
        height: 375px;
        object-fit: cover;
    }

    .demo-header {
        display: flex;
        justify-content: space-between;
    }

    .demo-title {
        font-size: 30px;
        font-weight: bold;
    }

    .btn-group {
        align-self: center;
    }
}

],
})
export class AppComponent {
    public orientation: Orientation = "horizontal";

    public lindseyAvatar = "assets/layout/stack-layout/avatar1.jpg";
    public vincenzoAvatar = "assets/layout/stack-layout/avatar2.jpg";
    public marissaAvatar = "assets/layout/stack-layout/avatar3.jpg";

    public lindseyImage = "assets/layout/stack-layout/dog1.jpg";
    public vincenzoImage = "assets/layout/stack-layout/summer.jpg";
    public marissaImage = "assets/layout/stack-layout/dog2.jpg";

    public toggleOrientation(orientation: Orientation): void {
        this.orientation = orientation;
    }
}

```

[Collapse Code ▲](#)

# Key Features

- **Orientation**—The StackLayout enables you to arrange its content horizontally or vertically.
- **Gap**—The `gap` property allows you to set gaps between the inner elements of the StackLayout.

- **Alignment**—You can control the alignment of the items in the StackLayout based on the X and Y axes.
- **Nested StackLayouts**—Nesting StackLayouts is supported to build more complex layouts.
- **Globalization**—All Kendo UI for Angular Layout components provide globalization options.

## Support and Learning Resources

- [StackLayout Homepage](#)
- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the StackLayout](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [StackLayout Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Stepper Overview

The Kendo UI for Angular Stepper visualizes the progress of a process by dividing the content into logical steps.

It enables wizard-like workflow and supports multiple step types, linear flow, step validation, templates, and more.

The following example demonstrates the Stepper in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import {
  bookIcon,
  eyeIcon,
  fileAddIcon,
  paperclipIcon,
  userIcon,
} from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  template: `
    <kendo-stepper
      [steps]="steps"
      stepType="full"
      [(currentStep)]="current"
      [linear]="false"
      [style.width.px]="570"
    >
    </kendo-stepper>
  `,
})
export class AppComponent {
  public current = 1;
```

```
public steps = [
    { label: "Personal Info", svgIcon: userIcon },
    { label: "Education", svgIcon: bookIcon },
    { label: "Attachments", svgIcon: paperclipIcon, optional: true },
    { label: "Preview", svgIcon: eyeIcon },
    { label: "Submit", svgIcon: fileAddIcon },
];
}
```

Collapse Code ^

## Key Features

- **Types**—You can display each step of the Stepper as a circle indicator, a text label, or a combination of the two.
- **Appearance**—The Stepper enables you to customize the appearance of its steps, for example, to display icons as indicators, add custom text inside the indicators, disable steps, and more.
- **Validation**—The Stepper allows you to implement validation logic and render a success or an error icon for previous steps.
- **Linear flow**—You can override the default linear flow of steps and enable the user to access all stages of the process.
- **Orientation**—The Stepper enables you to arrange its panes horizontally or vertically.
- **Templates**—You can apply templates to display custom content and customize the appearance of the Stepper indicators, labels, and steps.
- **Globalization**—All Kendo UI for Angular Layout components provide globalization options.
- **Accessibility**—The Stepper is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard Navigation**—The Stepper supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [Stepper Homepage](#)

- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the Stepper Component](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Stepper Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular TabStrip Overview

The Kendo UI for Angular TabStrip displays a collection of tabs, containing associated content, which enable the user to switch between different views inside a single component.

The following example demonstrates the TabStrip in action.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { Component } from "@angular/core";
import { SelectEvent } from "@progress/kendo-angular-layout";

@Component({
  selector: "my-app",
  template: `
    <div class="wrapper">
      <kendo-tabstrip (tabSelect)="onTabSelect($event)">
        <kendo-tabstrip-tab title="Paris" [selected]="true">
          <ng-template kendoTabContent>
            <div class="content">
              
              <h2>17<span>°C</span></h2>
              <span>Rainy weather in Paris.</span>
            </div>
          </ng-template>
        </kendo-tabstrip-tab>
        <kendo-tabstrip-tab title="New York City">
          <ng-template kendoTabContent>
            <div class="content">
              
              <h2>19<span>°C</span></h2>
              <span>Cloudy weather in New York City.</span>
            </div>
          </ng-template>
        </kendo-tabstrip-tab>
        <kendo-tabstrip-tab title="Tallinn">
          <ng-template kendoTabContent>
            <div class="content">
              
              <h2>22<span>°C</span></h2>
              <span>Sunny weather in Tallinn.</span>
            </div>
          </ng-template>
        </kendo-tabstrip-tab>
      </kendo-tabstrip>
    </div>
  `
```

```
        <ng-template kendoTabContent>
            <div class="content">
                
                <h2>23<span>°C</span></h2>
                <span>Sunny weather in Tallinn.</span>
            </div>
        </ng-template>
    </kendo-tabstrip-tab>
    <kendo-tabstrip-tab title="London">
        <ng-template kendoTabContent>
            <div class="content">
                
                <h2>16<span>°C</span></h2>
                <span>Rainy weather in London.</span>
            </div>
        </ng-template>
    </kendo-tabstrip-tab>
</kendo-tabstrip>
</div>
,
styles: [
    .wrapper {
        display: flex;
        justify-content: center;
    }
    kendo-tabstrip {
        width: 380px;
    }
    .content {
        padding: 30px;
        display: flex;
        flex-direction: column;
        align-items: center;
    }
    img {
        width: 120px;
        height: 120px;
    }
    h2 {
        font-size: 4em;
        font-weight: lighter;
    }
    h2 > span {
        padding-left: 5px;
        font-size: 0.3em;
        vertical-align: top;
    }
],
})
```

```
export class AppComponent {  
  public onTabSelect(e: SelectEvent): void {  
    console.log(e);  
  }  
}
```

Collapse Code ^

# Key Features

- **Tabs**—The TabStrip provides a set of options for controlling its tabs such as setting the such as displaying tabs on initial load, defining their titles and position, and more.
- **Close tab icons**—You can add an customize an icon indicating a close action to each tab of the TabStrip.
- **Configuration components**—The TabStrip enables you to build configuration components with the Angular structural directives.
- **Globalization**—All Kendo UI for Angular Layout components provide globalization options.
- **Accessibility**—The TabStrip is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The TabStrip supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [TabStrip Homepage](#)
- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the TabStrip](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [TabStrip Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular TileLayout Overview

The Kendo UI for Angular TileLayout displays a collection of tiles and aligns them into columns and rows.

The tiles are highly customizable in terms of content, size, positioning, and the spacing that separates them. The TileLayout also allows end users to rearrange and resize any of the tiles while providing full control over the flow of the component during these events to the developer.

The following example demonstrates the TileLayout in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
@Component({
  selector: "my-app",
  template: `
    <kendo-tilelayout
      [columns]="4"
      [rowHeight]="255"
      [resizable]="true"
      [reorderable]="true"
    >
      <kendo-tilelayout-item title="Page Views" [col]="1" [colSpan]="3">
        <kendo-tilelayout-item-body>
          <page-views-chart></page-views-chart>
        </kendo-tilelayout-item-body>
      </kendo-tilelayout-item>

      <kendo-tilelayout-item title="Conversion Rate" [col]="4">
        <kendo-tilelayout-item-body>
          <h3>9%</h3>
          <div>Visitor to Customer</div>
        </kendo-tilelayout-item-body>
      </kendo-tilelayout-item>
    
```

```
<kendo-tilelayout-item title="Most Visited Pages" [col]="1"
[colSpan]="2">
    <kendo-tilelayout-item-body>
        <most-visited-chart></most-visited-chart>
    </kendo-tilelayout-item-body>
</kendo-tilelayout-item>

<kendo-tilelayout-item title="Currently" [col]="3">
    <kendo-tilelayout-item-body>
        <h3>2399</h3>
        <div>Active users right now</div>
    </kendo-tilelayout-item-body>
</kendo-tilelayout-item>

<kendo-tilelayout-item title="Visitors" [col]="4" [rowSpan]="2">
    <kendo-tilelayout-item-body>
        <visitors-chart></visitors-chart>
    </kendo-tilelayout-item-body>
</kendo-tilelayout-item>

<kendo-tilelayout-item
    title="Users By Channel"
    [col]="1"
    [colSpan]="2"
    [rowSpan]="2"
>
    <kendo-tilelayout-item-body>
        <users-grid></users-grid>
    </kendo-tilelayout-item-body>
</kendo-tilelayout-item>

<kendo-tilelayout-item title="Bounce Rate" [col]="3">
    <kendo-tilelayout-item-body>
        <h3>55%</h3>
        <div>
            The percentage of all sessions on your site in which users
viewed
            only a single page.
        </div>
    </kendo-tilelayout-item-body>
</kendo-tilelayout-item>

<kendo-tilelayout-item
    title="Conversion This Month"
    [col]="3"
    [colSpan]="2"
>
    <kendo-tilelayout-item-body>
        <conversion-chart></conversion-chart>
    </kendo-tilelayout-item-body>
```

```
</kendo-tilelayout-item>
</kendo-tilelayout>
}
export class AppComponent {}
```

Collanse Code ^

# Key Features

- **Tiles configuration**—The TileLayout supports a number of settings for configuring the size and position of the tile rows and columns.
- **Columns and rows dimensions**—The TileLayout provides options for defining the number of visual columns and for setting the layout of its rows and columns.
- **Auto flow**—The TileLayout enables you to choose whether the component will use the auto-placement rules or the CSS grid-auto-flow configuration option.
- **Tiles reordering**—You can use the tile reordering options provided by the TileLayout such as enabling or disabling the functionality, using a programmatic approach to implement it, and more.
- **Tiles resizing**—You can use the tile resizing options provided by the TileLayout such as enabling or disabling the functionality, using a programmatic approach to implement it, and more.
- **Tiles collection**—The TileLayout delivers an alternative approach to the manual declaration of each container and enables you to render all tiles as a collection of configurable objects.
- **Globalization**—All Kendo UI for Angular Layout components provide globalization options.

# Known Limitations

- Placing the TileLayout within an element with `position` set to `absolute` or `relative` will result in undesired offsetting of the Tiles during resizing.

# Support and Learning Resources

- [TileLayout Homepage](#)
- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the TileLayout](#)
- [API Reference of the TileLayoutItem](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [TileLayout Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Timeline Overview

The Kendo UI for Angular Timeline displays a collection of events and their data in a chronological succession for each year.

The following example demonstrates the Timeline in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import { TimelineEvent } from "@progress/kendo-angular-layout";
import { events } from "./events";

@Component({
  selector: "my-app",
  template: `
    <kendo-timeline
      [events]="events"
      [collapsibleEvents]="true"
      [alterMode]="true"
    >
    </kendo-timeline>
  `,
})
export class AppComponent {
  public events: TimelineEvent[] = events;
}
```

[Collapse Code](#) ^

# Key Features

- **Layout**—The [Angular Timeline component](#) displays a collection of events and their data in a chronological succession for each year.
- **Horizontal and Vertical Modes**—The Kendo UI for Angular Timeline allows you to render its events in a vertical or horizontal list.
- **Templates**—The Kendo UI for Angular Timeline allows you to define your own template for rendering the events data so that you can customize it to your needs.
- **Events**—The Kendo UI for Angular Timeline allows you to handle the events and further customize the behavior of the component.
- **Keyboard Navigation**—The Kendo UI for Angular TimeLine supports various keyboard shortcuts that improve the ease of navigation.

# Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Layout](#)
- [API Reference of the Timeline](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular ListView

This guide provides the information you need to start using the Kendo UI for Angular ListView—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#)

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_LISTVIEW } from "@progress/kendo-angular-listview";
import { products } from "./products";

@Component({
  standalone: true,
  selector: "my-app",
  imports: [KENDO_LISTVIEW],
```

```
template: `<kendo-listview
  [data]="products"
  [style.height.px]="280"
  [itemClass]="{ 'item-border': true }"
>
  <ng-template kendoListViewItemTemplate let-dataItem="dataItem">
    {{ dataItem.ProductName }}
  </ng-template>
</kendo-listview>
`,
encapsulation: ViewEncapsulation.None,
styles: [
  .item-border {
    padding: 10px;
    border-bottom: 1px solid lightgrey;
  }
],
})
export class AppComponent {
  public products = products;
}
```

[Collapse Code](#) ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Component

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular ListView package:

## 1. Run the following command:

```
ng add @progress/kendo-angular-listview
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-listview` package as a dependency to the `package.json` file.
- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the `KENDO_LISTVIEW` utility array in your standalone component to enable the entire feature set of the ListView:

The utility array is available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

```
import { Component } from '@angular/core';
import { KENDO_LISTVIEW } from '@progress/kendo-angular-listview';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_LISTVIEW]
})
```

TS

# Using the Component

1. After successfully installing the ListView package and importing its component, add the following code in the `app.component.html` file:

```
<kendo-listview  
  [data]="products"  
  [itemClass]="{ 'item-border': true }"  
>  
  <ng-template kendoListViewItemTemplate let-dataItem="dataItem">  
    {{ dataItem.productName }}  
  </ng-template>  
</kendo-listview>
```

HTML

2. Bind the `data` property to a collection that will be used to populate the ListView in the `app.component.ts` file.

```
public products = [  
  {  
    ProductID: 1,  
    ProductName: 'Chai'  
  },  
  {  
    ProductID: 2,  
    ProductName: 'Chang'  
  },  
  ...  
]
```

TS

3. Apply a custom CSS class to customize the layout of each item by setting the `itemClass` property of the component:

```
.item-border {  
  padding: 10px;  
  border-bottom: 1px solid lightgrey;  
}
```

CSS

4. Build and serve the application by running the following command in the root folder.

5. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular ListView component on the page.

## Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

## Next Steps

- [ListView Dependencies & Standalone Utilities](#)
- [Paging the ListView content](#)
- [Applying the ListView scroll modes](#)
- [Editing the ListView](#)
- [Using the ListView templates](#)
- [API Reference of the ListView](#)

## Learning Resources

- [ListView Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)



# Angular ListView Overview

The Kendo UI for Angular ListView component visualizes repeated data content, providing a variety of configuration options.

The ListView is built from the ground up and specifically for Angular, so that you get a high-performance control which delivers lightning-fast performance, integrates tightly with your application and with the rest of the Kendo UI for Angular components, and is highly customizable.

The following example demonstrates the ListView in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_LISTVIEW } from "@progress/kendo-angular-listview";
import { ContactComponent } from "./contact.component";
import { contacts } from "./contacts";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_LISTVIEW, ContactComponent],
  template: `
    <kendo-listview
      [data]="contacts"
      containerClass="k-d-flex k-float-col k-float-wrap"
    >
      <ng-template kendoListViewHeaderTemplate>
        <div class="header">Contact list</div>
      </ng-template>
      <ng-template kendoListViewItemTemplate
        let-dataItem="dataItem"
        let-isFirst="isFirst"
      >
        <contact-card
          class="contact"
        >
      
```

```

        [contact]="dataItem"
        [borderTop]!="isFirst"
      >
    </contact-card>
  </ng-template>
  <ng-template kendoListViewFooterTemplate>
    <div class="footer">25 unread messages in total</div>
  </ng-template>
</kendo-listview>
,
styles: [
  .k-listview {
    font-family: sans-serif;
    width: 400px;
    margin: auto;
  }
  .header,
  .footer {
    color: #a0a0a0;
    font-size: 16px;
    padding: 6px 10px;
  }
  .footer {
    font-size: 14px;
  }
  .contact {
    width: 100%;
  }
],
})
export class AppComponent {
  public contacts: any[] = contacts;
}

```

[Collapse Code ▾](#)

## Angular ListView Key Features

The Kendo UI for Angular ListView component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing ListView library as well as develop new features to it.

## Paging

The ListView enables you to split its content into pages by adding a pager and by using its extensive set of configuration options such as implementing a responsive behavior, customizing its appearance, and more. [Read more about the pager options of the ListView...](#)

## Editing

Depending on your user requirements, you can edit, add, or delete the ListView items by using Reactive or Template-Driven Angular Forms. [Read more about the editing feature of the ListView...](#)

## Scroll Modes

You can benefit from the default scrolling or enable endless scrolling mode which allows you to load more records by appending additional data items on demand. [Read more about the scroll modes of the ListView...](#)

## Templates

You can customize the headers, footers, and the ListView items by utilizing the available templates. [Read more about the available templates of the ListView...](#)

## Accessibility

The ListView is accessible for screen readers and supports WAI-ARIA attributes. [Read more about the accessibility of the ListView...](#)

## Keyboard Navigation

The ListView supports a number of keyboard shortcuts which allow users to accomplish various commands. [Read more about the keyboard navigation of the](#)

# Trial Version of Kendo UI for Angular

The Kendo UI for Angular ListView is part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of the Kendo UI for Angular ListView, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [ListView Homepage](#)
- [Getting Started with the Kendo UI for Angular ListView](#)
- [API Reference of the ListView](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [ListView Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Paging

The ListView provides built-in integration with the [Kendo UI for Angular Pager component](#), which allows you to quickly set up the pager content and handle local data with the built-in page change.

To enable paging:

1. Set the [pageable](#), [pageSize](#), and [skip](#) options of the ListView.
2. Handle the page changes in either of the following ways:
  - Use the built-in [kendoListViewBinding](#) directive. It handles page changes internally and works for local data only.
  - Manually handle the [pageChange](#) event of the ListView. It is suitable for remote binding.

# Binding Directive

To use the built-in binding directive, apply the [kendoListViewBinding](#) directive to the ListView. The directive selector accepts as a single argument a plain array of objects ([any\[\]](#)).

The following example demonstrates the directive in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";  
  
import {  
  KENDO_LISTVIEW,  
  PagerSettings,  
}
```

```
    } from "@progress/kendo-angular-listview";
import { DestinationComponent } from "./destination.component";
import { destinations, Destination } from "./destinations";

@Component({
  encapsulation: ViewEncapsulation.None,
  selector: "my-app",
  standalone: true,
  imports: [KENDO_LISTVIEW, DestinationComponent],
  template:
    <kendo-listview
      [kendoListViewBinding]="destinations"
      [pageable]="pagerSettings"
      [pageSize]="pageSize"
      containerClass="listview-content"
    >
      <ng-template
        kendoListViewItemTemplate
        let-dataItem="dataItem"
        let-isLast="isLast"
      >
        <destination-card [destination]="dataItem"></destination-card>
      </ng-template>
      <ng-template kendoListViewFooterTemplate>
        <div class="footer-note">Source:
          www.european.destinations</div>
      </ng-template>
    </kendo-listview>
  ,
  styles: [
    .k-listview-header,
    .k-listview-content {
      padding: 20px 0;
      justify-content: center;
    }

    .k-listview-footer {
      border-width: 0;
    }

    .listview-content {
      display: grid;
      grid-template-columns: repeat(auto-fill, 180px);
      gap: 15px 30px;
    }

    .footer-note {
      text-align: right;
      font-style: italic;
      font-size: 11px;
    }
  ]
})
```

```

        padding-right: 4px;
    }
},
],
})
export class AppComponent {
    public destinations: Destination[] = destinations;

    public pagerSettings: PagerSettings = {
        previousNext: false,
        pageSizeValues: false,
        buttonCount: 9,
    };
    public pageSize = 6;
}

```

[Collapse Code ^](#)

# Remote Binding

When you deal with a large data set, it's better to perform the paging on the server and fetch only parts of the data on demand.

To configure the ListView for such a scenario, you have to handle the following:

1. Provide the ListView `data` input with a `ListViewDataResult` object with the current batch of data items and information for the total number of records that are available on the server.
2. Hook to the `pageChange` event and manually update the `pageSize` and `skip` property values.
3. (Optional) Update the `loading` property value to render a loading indicator while the data is being fetched.

 EXAMPLE

 VIEW SOURCE

⋮

```

import { Component, OnInit, OnDestroy } from "@angular/core";
import { Subscription } from "rxjs";
import { finalize } from "rxjs/operators";
import {

```

```
KENDO_LISTVIEW,
ListViewDataResult,
PageChangeEvent,
} from "@progress/kendo-angular-listview";
import { ProductComponent } from "./product.component";
import { ProductsService } from "./products.service";

@Component({
  selector: "my-app",
  standalone: true,
  providers: [ProductsService],
  imports: [KENDO_LISTVIEW, ProductComponent],
  template:
    <kendo-listview
      [data]="view"
      [loading]="loading"
      [itemStyle]="{
        display: 'inline-flex'
      }"
      [pageable]="showPager"
      [pageSize]="pageSize"
      [skip]="skip"
      (pageChange)="handlePageChange($event)"
    >
      <ng-template kendoListViewItemTemplate let-dataItem="dataItem">
        <product-tile [product]="dataItem"></product-tile>
      </ng-template>
    </kendo-listview>
  ,
  styles: [
    .k-listview {
      height: 340px;
    }
  ],
})
export class AppComponent implements OnInit, OnDestroy {
  public view: ListViewDataResult;
  public loading = false;

  public skip = 0;
  public pageSize = 20;

  public get showPager(): boolean {
    return this.view && this.view.total > 0;
  }

  private productsSubscription = new Subscription();

  constructor(private productService: ProductsService) {}
```

```
public ngOnInit(): void {
    this.fetchData();
}

public ngOnDestroy(): void {
    if (this.productsSubscription) {
        this.productsSubscription.unsubscribe();
    }
}

public handlePageChange(event: PageChangeEvent): void {
    this.skip = event.skip;
    this.pageSize = event.take;

    this.fetchData();
}

public fetchData(): void {
    if (this.productsSubscription) {
        this.productsSubscription.unsubscribe();
    }

    this.loading = true;
    this.productsSubscription = this.productsService
        .get({ skip: this.skip, take: this.pageSize })
        .pipe(finalize(() => (this.loading = false)))
        .subscribe((response) => (this.view = response));
}
}
```

Collanse Code ^

# Pager Options

You can customize the built-in pager content according to the application needs. To configure the pager, pass a `PagerSettings` object to the `pageable` input of the `ListView`.

The `PagerSettings` object has the following fields:

- `position`—Sets the pager position relative to the `ListView` content section.
- `pageSizeValues`—Sets the list of drop-down values from which the end-user can choose for current page size. If the input is set to `false`, the page sizes drop-down

will not be rendered.

- `buttonCount`—Sets the maximum numeric buttons count before the buttons are collapsed.
- `info`—Toggles the information about the current page and the total number of records.
- `previousNext`—Toggles the Previous and Next buttons.
- `type`—Accepts the `numeric` (buttons with numbers) and `input` (input for typing the page number) values.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { FormsModule } from "@angular/forms";

import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import {
    KENDO_LISTVIEW,
    PagerSettings,
    PagerPosition,
    PagerType,
} from "@progress/kendo-angular-listview";
import { products } from "./products";

@Component({
    selector: "my-app",
    standalone: true,
    imports: [FormsModule, KENDO_INPUTS, KENDO_LISTVIEW],
    template: `
        <div class="example-config">
            <div>
                <label class="radio-button">
                    <input
                        type="radio"
                        kendoRadioButton
                        name="position"
                        value="bottom"
                        [(ngModel)]="position"
                    />
                    Bottom pager
                </label>
                <label class="radio-button">
                    <input
```

```
        type="radio"
      kendoRadioButton
      name="position"
      value="top"
      [(ngModel)]="position"
    />
    Top pager
  </label>
<label class="radio-button">
  <input
    type="radio"
    kendoRadioButton
    name="position"
    value="both"
    [(ngModel)]="position"
  />
  Both
</label>
</div>
<div>
  <label
    ><input type="checkbox" kendoCheckBox [(ngModel)]="pageSizes"
/> Show
    page size options</label>
  >
</div>
<div>
  <label
    ><input type="checkbox" kendoCheckBox [(ngModel)]="info" />
Show
  info</label>
  >
</div>
<div>
  <label
    ><input type="checkbox" kendoCheckBox [(ngModel)]="prevNext"
/> Show
    previous/next buttons</label>
  >
</div>
<div>
  <label class="radio-button">
    <input
      type="radio"
      kendoRadioButton
      value="numeric"
      [(ngModel)]="type"
    />
    Numeric buttons
  </label>
<label class="radio-button">
```

```
<input
  type="radio"
  kendoRadioButton
  value="input"
  [(ngModel)]="type"
/>
  Numeric input
</label>
</div>
</div>

<kendo-listview
  [kendoListViewBinding]="products"
  [pageable]="pagerSettings"
  [pageSize]="pageSize"
  containerClass="k-d-flex k-float-col k-float-nnowrap"
>
  <ng-template
    kendoListViewItemTemplate
    let-dataItem="dataItem"
    let-isLast="isLast"
  >
    <div class="product" [class.last]="isLast">
      {{ dataItem.productName }}
    </div>
  </ng-template>
</kendo-listview>
,
styles: [
  .product {
    padding: 10px;
    font-size: 16px;
    border-bottom-color: lightgrey;
    border-bottom-style: solid;
    border-bottom-width: 1px;
    width: 100%;
  }
  .product.last {
    border-bottom-width: 0;
  }
  .radio-button {
    margin-right: 5px;
  }
],
})
export class AppComponent {
  public products: any[] = products;
  public pageSize = 5;
```

```
public position: PagerPosition = "bottom";
public pageSizes = false;
public info = true;
public prevNext = true;
public type: PagerType = "numeric";

public get pagerSettings(): PagerSettings {
    return {
        position: this.position,
        pageSizeValues: this.pageSizes,
        info: this.info,
        previousNext: this.prevNext,
        type: this.type,
    };
}
```

[Collapse Code ^](#)

## Suggested Links

- [API Reference of the ListViewComponent](#)
- [API Reference of the DataBindingDirective](#)
- [API Reference of the PagerSettings](#)
- [API Index of the ListView](#)

# Kendo UI for Angular ContextMenu Overview

The ContextMenu displays a hierarchical list of items in a popup.

The following example demonstrates the ContextMenu in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { items } from "./contextmenu-items";

@Component({
  selector: "my-app",
  template: `
    <div #target class="target">
      <p class="placeholder">Right-click to open Context menu</p>
    </div>
    <kendo-contextmenu [target]="target" [items]="items"> </kendo-
contextmenu>
  `,
  styles: [
    .target {
      border-radius: 5px;
      height: 100px;
      width: 400px;
      background-color: #f6f6f6;
      display: flex;
      justify-content: center;
      align-items: center;
      box-shadow: 0 1px 5px 0 rgba(0, 0, 0, 0.26),
        0 2px 2px 0 rgba(0, 0, 0, 0.12), 0 3px 1px -2px rgba(0, 0, 0,
        0.08);
    }
    .placeholder {
      font-size: 20px;
      color: #656565;
    }
  ]
})
```

```
        margin: 0;
    }
},
],
})
export class AppComponent {
    public items: any[] = items;
}
```

Collanse Code ^

# Key Features

- **Data binding**—The ContextMenu provides configuration options for binding it to an array of objects.
- **Items**—You can instantiate the ContextMenu items by initializing them as item components and then use their corresponding properties.
- **Opening**—You can change the event on which the ContextMenu component opens.
- **Target**—The ContextMenu enables you to configure its target to various values and also allows you to filter these targets within the container for which the component will open.
- **Orientation**—You can render the items of the ContextMenu horizontally.
- **Using with data-bound components**—The ContextMenu can be used with other data-bound Kendo UI components such as the Grid, Chart, and TreeView.
- **Templates**—You can customize the content of the ContextMenu, its items, and the content of its items by using templates.
- **Globalization**—All Kendo UI for Angular Menus provide globalization options.
- **Keyboard navigation**—The ContextMenu supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [ContextMenu Homepage](#)
- [Getting Started with the Kendo UI for Angular Menus](#)
- [API Reference of the ContextMenu](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)

- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [ContextMenu Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Menus

This guide provides the information you need to start using the Kendo UI for Angular Menus—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#).

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_MENUS } from "@progress/kendo-angular-menu";

@Component({
  standalone: true,
  selector: "my-app",
  imports: [KENDO_MENUS],
  template: `<kendo-menu [items]="items"> </kendo-menu> `,
})
export class AppComponent {
  public items: any[] = [
```

```
{  
  text: "Item1",  
  items: [  
    { text: "Item1.1" },  
    { text: "Item1.2", items: [{ text: "Item1.2.1" }] },  
  ],  
},  
{  
  text: "Item2",  
  items: [{ text: "Item2.1" }],  
},  
];  
}
```

Collapse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Menus package:

1. Run the following command.

```
ng add @progress/kendo-angular-menu
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-menu` package as a dependency to the `package.json` file.
- Add all required peer dependencies to the `package.json` file.
- Register the Kendo UI Default theme in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the required components, directives and utility arrays:

The utility arrays are available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

- To add all components from the Menus package, import the `KENDO_MENUS` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_MENUS } from '@progress/kendo-angular-menu';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_MENUS]
})
```

TS

- To add individual Menus components, import the corresponding utility arrays in your standalone component. See the list of [available utility arrays](#).

For example if you only need the `ContextMenu` component, import `KENDO_CONTEXTMENU`.

```
import { Component } from '@angular/core';
import { KENDO_CONTEXTMENU } from '@progress/kendo-angular-menu';
```

TS

```
@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_CONTEXTMENU]
})
```

# Using the Components

- After successfully installing the Menus package and importing the required components and directives, add the corresponding tags of the components you need in the `app.component.html`. For example, if you need the Menu component, add the following code:

```
<kendo-menu [items]="items"> </kendo-menu>
```

HTML

- Bind the `items` property to specify the Menu items in the `app.component.ts` file:

```
public items: any[] = [
  {
    text: 'Item1',
    items: [{ text: 'Item1.1' }, { text: 'Item1.2', items: [{ text: 'Item1.2.1' }] }]
  },
  {
    text: 'Item2',
    items: [{ text: 'Item2.1' }]
  }
];
```

TS

- Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

4. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular Menu component on the page.

# Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

# Next Steps

- [Menus Dependencies & Standalone Utilities](#)
- [Menu Overview](#)
- [ContextMenu Overview](#)
- [Globalization](#)
- [Menus API Documentation](#)

# Learning Resources

- [Menus Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Kendo UI for Angular Menu Overview

The Menu displays a hierarchical list of items.

The following example demonstrates the Menu in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { items } from "./menu-items";

@Component({
  selector: "my-app",
  template: `<kendo-menu [items]="items"></kendo-menu>`,
})
export class AppComponent {
  public items: any[] = items;
}
```

[Collapse Code ^](#)

## Key Features

- **Data binding**—The Menu provides configuration options for binding it to an array of objects.

- **Items**—You can instantiate the Menu items by initializing them as item components and then use their corresponding properties.
- **Vertical Menu**—You can use the configuration option of the Menu and render it vertically.
- **Opening and closing**—You can change the opening and closing behavior of the Menu by configuring the hover delay and open the component on click.
- **Templates**—You can customize the items and the item content of the Menu by using templates.
- **Routing**—The Menu can utilize the Angular Router in this way enabling you to use it as a navigational component.
- **Globalization**—All Kendo UI for Angular Menus provide globalization options.
- **Accessibility**—The Menu is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The Menu supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [Menu Homepage](#)
- [Getting Started with the Kendo UI for Angular Menus](#)
- [API Reference of the Menu](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Menu Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)

- Kendo UI for Angular Roadmap

# Angular Menus Overview

The Kendo UI for Angular Menus display a hierarchical list of items in a popup. The package contains the `Menu` and `ContextMenu` components.

The Menus are built from the ground up and specifically for Angular, so that you get high-performance controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



**ContextMenu**

A component for displaying hierarchical data as a multi-level menu in a popup



**Menu**

A component for displaying hierarchical data as a multi-level menu

The following example demonstrates all available Kendo UI for Angular Menus components in action.

EXAMPLE

VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { items } from "./menu-items";

@Component({
  selector: "my-app",
  template: `
    <div #target class="target">
      <p class="placeholder">Right-click to open Context menu</p>
    </div>
    <kendo-contextmenu [target]="target" [items]="items"> </kendo-
contextmenu>
      <kendo-menu [items]="items"> </kendo-menu>
    `,
  styles: [
```

```
.target {
    border-radius: 5px;
    height: 100px;
    width: 400px;
    background-color: #f6f6f6;
    display: flex;
    justify-content: center;
    align-items: center;
    box-shadow: 0 1px 5px 0 rgba(0, 0, 0, 0.26),
    0 2px 2px 0 rgba(0, 0, 0, 0.12), 0 3px 1px -2px rgba(0, 0, 0,
0.08);
}
.placeholder {
    font-size: 20px;
    color: #656565;
    margin: 0;
}
],
})
export class AppComponent {
    public items: any[] = items;
}
```

Collanse Code ^

# Angular Menus Key Features

Each Kendo UI for Angular Menus component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Menus library as well as develop new features and controls to it.

## Data Binding

You can bind the Menu components to a local or a remote objects array. [Read more about how to bind the Menu component...](#)

# Orientation

You can set the orientation of the components to vertical or horizontal based on your needs. [Read more about orientation of the ContextMenu component...](#)

# Opening

The Menus open and close on different user interactions but you can change the event on which the desired menu will open or close. [Read more about the opening behavior of the ContextMenu...](#)

# Templates

The Menus enable you to customize their content, items, and the content of the items by using the available templates. [Read more about the templates of the ContextMenu...](#)

# Globalization

The Kendo UI for Angular Menus support globalization to ensure that each component can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Menus support rendering in a right-to-left (RTL) direction. [Read more about Menus globalization...](#)

# Accessibility

The Menus are accessible for screen readers and support WAI-ARIA attributes. [Read more about accessibility support of the Menu...](#)

# Keyboard Navigation

The Menus support a number of keyboard shortcuts which allow users to accomplish various commands. [Read more about the keyboard navigation of the](#)

# Trial Version of Kendo UI for Angular

The Kendo UI for Angular Menus are part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of Kendo UI for Angular Menus, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Menus](#)
- [API Reference of the Menus](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular ActionSheet Overview

The ActionSheet component allows you to display a predefined set of options in a modal view. It is commonly used to present choices related to an action that the user initiates.

The following example demonstrates the ActionSheet in practice.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { Component } from "@angular/core";
import { ActionSheetItem } from "@progress/kendo-angular-navigation";
import {
  cancelIcon,
  eyedropperIcon,
  heartIcon,
  SVGIcon,
  uploadIcon,
} from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  template: `
    <div class="demo-app" style="position:relative">
      <div id="city" class="demo-view" style="position:absolute">
        <div class="title">
          <h3>Current City</h3>
        </div>
        <div class="cards-container">
          <div class="k-card">
            <div class="k-card-header">
              <h5 class="k-card-title">Rome</h5>
              <h6 class="k-card-subtitle">Capital of Italy</h6>
            </div>
            
          </div>
        </div>
      </div>
    </div>
  `,
})
```

```
        />
      <div class="k-card-body">
        <p>
          Rome is a sprawling, cosmopolitan city with nearly 3,000
years
          of globally influential art, architecture and culture on
display.
        </p>
        <p>
          Ancient ruins such as the Forum and the Colosseum evoke
the
          power of the former Roman Empire.
        </p>
      </div>
    </div>
  </div>
<button
  kendoButton
  id="openActionSheetBtn"
  (click)="clickHandler()"
  themeColor="tertiary"
>
  OPEN ACTION SHEET
</button>
<kendo-actionsheet
  [expanded]="expanded"
  (itemClick)="onItemClick()"
  (overlayClick)="onOverlayClick()"
  [title]="title"
  [items]="items"
>
  </kendo-actionsheet>
</div>
</div>
``,
styles: [
  .demo-view {
    transform: scale(1);
    overflow: hidden;
  }
  .k-actionsheet-container {
    width: 100%;
    height: 100%;
    z-index: 1;
  }
  .k-actionsheet-container > .k-overlay {
    border-bottom-left-radius: 30px;
    border-bottom-right-radius: 30px;
  }
]
```

```
        }

.cards-container {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
}

.k-card {
  width: 285px;
  margin: 5%;
}

#openActionSheetBttn {
  margin: 0 auto;
  display: block;
}

.demo-app {
  margin: auto;
  width: 380px;
  height: 630px;
  background: #ffffff;
  box-shadow: 0px 10px 20px #00000016;
  border-radius: 30px;
  font-family: "Roboto", sans-serif;
  font-size: 14px;
}
.demo-app .title {
  width: 100%;
}

#city {
  background: #f9f9f9;
  height: inherit;
  width: inherit;
  border-radius: 30px;
  border: 10px solid white;
  box-sizing: border-box;
}

.demo-app h3 {
  padding-top: 24px;
  text-align: center;
  font-size: 28px;
  letter-spacing: 0.28px;
  color: #3d57d8;
  font-weight: 400;
}
``,
],

```

```
)  
export class AppComponent {  
  public expanded = false;  
  public title = "Select item";  
  public editIcon: SVGIcon = eyedropperIcon;  
  public uploadIcon: SVGIcon = uploadIcon;  
  public cancelIcon: SVGIcon = cancelIcon;  
  public heartIcon: SVGIcon = heartIcon;  
  public items: ActionSheetItem[] = [  
    {  
      title: "Edit Item",  
      svgIcon: this.editIcon,  
    },  
    {  
      title: "Add to Favorites",  
      svgIcon: this.heartIcon,  
    },  
    {  
      title: "Upload New",  
      svgIcon: this.uploadIcon,  
    },  
    {  
      title: "Cancel",  
      svgIcon: this.cancelIcon,  
      group: "bottom",  
    },  
  ];  
  
  public clickHandler(): void {  
    this.expanded = true;  
  }  
  
  public onItemClick(): void {  
    this.expanded = false;  
  }  
  
  public onOverlayClick(): void {  
    this.expanded = false;  
  }  
}
```

Collanse Code ^

# Key Features

- **Items**—The ActionSheet enables you to configure the items based on specific project requirements.

- **Templates**—You can also customize the content of the ActionSheet items using templates.
- **Accessibility**—The ActionSheet is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The ActionSheet supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [ActionSheet Homepage](#)
- [Getting Started with the Kendo UI for Angular Navigation](#)
- [API Reference of the ActionSheet](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [ActionSheet Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular AppBar Overview

The AppBar provides information and actions related to the current application screen. It is typically used to show page titles or brand identity, and can contain navigation items.

The following example demonstrates the AppBar in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { bellIcon, menuIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  template: `
    <kendo-appbar position="top">
      <kendo-appbar-section>
        <button kendoButton fillMode="flat">
          <kendo-svgicon [icon]="menuIcon"></kendo-svgicon>
        </button>
      </kendo-appbar-section>

      <kendo-appbar-section>
        <h1 class="title">Kendo UI for Angular</h1>
      </kendo-appbar-section>

      <kendo-appbar-spacer width="32px"></kendo-appbar-spacer>

      <kendo-appbar-section>
        <ul>
          <li><span>What's New</span></li>
          <li><span>About</span></li>
          <li><span>Contacts</span></li>
        </ul>
      </kendo-appbar-section>

      <kendo-appbar-spacer></kendo-appbar-spacer>
    </kendo-appbar>
  `
```

```
<kendo-appbar-section class="actions">
  <kendo-badge-container>
    <button kendoButton fillMode="flat">
      <kendo-svgicon [icon]="bellIcon"></kendo-svgicon>
    </button>
    <kendo-badge
      shape="dot"
      themeColor="warning"
      size="small"
      position="inside"
    ></kendo-badge>
  </kendo-badge-container>
  <span class="k-appbar-separator"></span>
</kendo-appbar-section>

<kendo-appbar-section>
  <kendo-avatar
    [imageSrc]="kendokaAvatar"
    shape="circle"
    width="26px"
    height="26px"
  ></kendo-avatar>
</kendo-appbar-section>
</kendo-appbar>
``,
encapsulation: ViewEncapsulation.None,
styles: [
  body {
    background: #adadb1;
  }
  .title {
    font-size: 18px;
    margin: 0;
  }
  kendo-badge-container {
    margin-right: 8px;
  }
  ul {
    font-size: 14px;
    list-style-type: none;
    padding: 0;
    margin: 0;
    display: flex;
  }
  li {
    margin: 0 9px;
  }
  li:hover {
    cursor: pointer;
  }
]
```

```
        color: #d6002f;
    }
.actions .k-button {
    padding: 0;
}
`,
],
})
export class AppComponent {
    public menuIcon: SVGIcon = menuIcon;
    public bellIcon: SVGIcon = bellIcon;
    public kendokaAvatar = "assets/navigation/appbar/kendoka-angular.png";
}
```

Collanse Code ^

## Key Features

- **Position**—The AppBar enables you to specify its position in relation to the page content.
- **Content arrangement**—You can also arrange the layout of the AppBar content by splitting it into sections, defining spacings, and more.
- **Theme color**—The AppBar delivers predefined theme colors, which you can apply as its background.
- **Responsive design**—The AppBar provides options for automatically adjusting it to different screen sizes.
- **Globalization**—All Kendo UI for Angular Navigation components provide globalization options.

## Support and Learning Resources

- [AppBar Homepage](#)
- [Getting Started with the Kendo UI for Angular Navigation](#)
- [API Reference of the AppBar](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)

- [AppBar Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular BottomNavigation Overview

The BottomNavigation component allows the user to navigate between primary destinations in an application. The component consists of items with text and icons, which offer a navigation element.

The following example demonstrates the BottomNavigation in action.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { BottomNavigationSelectEvent } from "@progress/kendo-angular-navigation";
import { ActivatedRoute, Route, Router } from "@angular/router";
import { Component } from "@angular/core";
import { bottomNavigationRoutes } from "./app.module";

export interface CustomRoute extends Route {
    text?: string;
    svgIcon?: string;
}

@Component({
    selector: "my-app",
    styleUrls: ["./styles.css"],
    template: `
        <div class="example-wrapper">
            <router-outlet></router-outlet>
            <kendo-bottomnavigation
                [items]="items"
                positionMode="sticky"
                [border]="true"
                (select)="onSelect($event)">
            </kendo-bottomnavigation>
        </div>
    `,
})

```

```
export class AppComponent {
    public items = [];

    constructor(private router: Router, private route: ActivatedRoute) {
        this.items = this.mapItems(bottomNavigationRoutes);
        this.items[0].selected = true;
    }

    public onSelect(ev: BottomNavigationSelectEvent): void {
        this.router.navigate([`./${ev.item.path}`], { relativeTo: this.route });
    }

    public mapItems(routes: any[]): CustomRoute[] {
        return routes.map((item) => {
            return {
                text: item.text,
                svgIcon: item.svgIcon,
                path: item.path ? item.path : "",
            };
        });
    }
}
```

Collanse Code ^

# Key Features

- **Items**—The BottomNavigation allows you to specify a collection of items and enables you to define their content and content flow.
- **Appearance**—You can also use the predefined settings for applying different styling options to the BottomNavigation.
- **Position**—The BottomNavigation enables you to specify its position in relation to the page content.
- **Templates**—The BottomNavigation provides options for customizing the appearance of its items.
- **Routing**—You can utilize the Angular Router service and use the BottomNavigation as a navigational component.
- **Globalization**—All Kendo UI for Angular Navigation components provide globalization options.

- **Accessibility**—The BottomNavigation is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The BottomNavigation supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [BottomNavigation Homepage](#)
- [Getting Started with the Kendo UI for Angular Navigation](#)
- [API Reference of the BottomNavigation](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [BottomNavigation Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Breadcrumb Overview

The Angular Breadcrumb component allows you to navigate within a folder structure or web page and provides an easy way to navigate backwards by one or multiple steps.

The following example demonstrates the Angular Breadcrumb in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";

import { BreadCrumbItem } from "@progress/kendo-angular-navigation";
import {
  arrowRotateCcwIcon,
  homeIcon,
  SVGIcon,
} from "@progress/kendo-svg-icons";

const defaultItems: BreadCrumbItem[] = [
  {
    text: "Home",
    title: "Home",
    svgIcon: homeIcon,
  },
  {
    text: "Products",
    title: "Products",
  },
  {
    text: "Computer peripherals",
    title: "Computer peripherals",
  },
  {
    text: "Keyboards",
    title: "Keyboards",
  },
]
```

```

        text: "Gaming keyboards",
        title: "Gaming keyboards",
    },
];

@Component({
  selector: "my-app",
  template:
    <button kendoButton [svgIcon]="rotateIcon"
(click)="refreshBreadCrumb()">
  Refresh breadcrumb
</button>
<kendo-breadcrumb
  [items]="items"
  (itemClick)="onItemClick($event)"
></kendo-breadcrumb>
`,
  styles: [
    kendo-breadcrumb {
      margin-top: 20px;
    }
  ],
})
export class AppComponent {
  public items: BreadCrumbItem[] = [...defaultItems];
  public homeIcon: SVGIcon = homeIcon;
  public rotateIcon: SVGIcon = arrowRotateCcwIcon;
  public onItemClick(item: BreadCrumbItem): void {
    const index = this.items.findIndex((e) => e.text === item.text);
    this.items = this.items.slice(0, index + 1);
  }

  public refreshBreadCrumb(): void {
    this.items = [...defaultItems];
  }
}

```

[Collapse Code ^](#)

# Key Features

- **Collapse modes**—The Angular Breadcrumb enables you to specify the way its items will be rendered when their total width is larger than the width of the component.

- [Templates](#)—The Angular Breadcrumb provides options for customizing the appearance of its items.
- [Item appearance](#)—The Angular Breadcrumb delivers a set of options for customizing the appearance of its items such as adding an item icon, and more.
- [Separator](#)—You can use the built-in Kendo UI icons to change the default Angular Breadcrumb separator.
- [Routing](#)—You can utilize the Angular Router service and use the Angular Breadcrumb as a navigational component.
- [Globalization](#)—All Kendo UI for Angular Navigation components provide globalization options.
- [Accessibility](#)—The Angular Breadcrumb is accessible for screen readers and supports WAI-ARIA attributes.

## Support and Learning Resources

- [Breadcrumb Homepage](#)
- [Getting Started with the Kendo UI for Angular Navigation](#)
- [API Reference of the Breadcrumb](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Breadcrumb Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Navigation

This guide provides the information you need to start using the Kendo UI for Angular Navigation—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#).

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import {
  KENDO_NAVIGATION,
  BreadCrumbItem,
} from "@progress/kendo-angular-navigation";
import { homeIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
```

```
standalone: true,
selector: "my-app",
imports: [KENDO_NAVIGATION],
template: `<kendo-breadcrumb [items]="items"></kendo-breadcrumb> `,
})
export class AppComponent {
  public hIcon: SVGIcon = homeIcon;
  public items: BreadCrumbItem[] = [
    {
      text: "Home",
      title: "Home",
      svgIcon: this.hIcon,
    },
    {
      text: "Products",
      title: "Products",
    },
    {
      text: "Keyboards",
      title: "Keyboards",
    },
  ],
};
```

Collanse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

# Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Navigation package:

## 1. Run the following command.

```
ng add @progress/kendo-angular-navigation
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-navigation` package as a dependency to the `package.json` file.
- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the required components, directives and utility arrays:

The utility arrays are available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

- To add all components from the Navigation package, import the `KENDO_NAVIGATION` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_NAVIGATION } from '@progress/kendo-angular-navigation';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_NAVIGATION]
})
```

TS

- To add individual Navigation components, import the corresponding utility arrays in your standalone component. See the list of [available](#)

utility arrays.

For example if you only need the AppBar component, import KENDO\_APPBAR.

```
import { Component } from '@angular/core';
import { KENDO_APPBAR } from '@progress/kendo-angular-navigation';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_APPBAR]
})
```

TS

## Using the Components

1. After successfully installing the Navigation package and importing the required components and directives, add the corresponding tags of the components you need in the `app.component.html`. For example, if you need the BreadCrumb, add the following code:

```
<kendo-breadcrumb [items]="items"></kendo-breadcrumb>
```

HTML

2. Bind the `items` property to a collection of items in the `app.component.ts` file:

```
public items: BreadCrumbItem[] = [
  {
    text: 'Home',
    title: 'Home',
    icon: 'home'
  },
  {
    text: 'Products',
    title: 'Products'
  },
  {
```

TS

```
        text: 'Keyboards',
        title: 'Keyboards'
    }
];
```

3. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

4. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular Breadcrumb component on the page.

## Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

## Next Steps

- [AppBar Overview](#)
- [BottomNavigation Overview](#)
- [Breadcrumb Overview](#)
- [Globalization](#)
- [Navigation API Documentation](#)

## Learning Resources

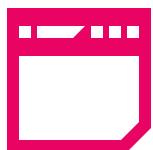
- [Navigation Overview](#)
- [Get Started with Kendo UI for Angular](#)

- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Angular Navigation Overview

The Kendo UI for Angular Navigation components provide easy and intuitive page navigation.

The Navigation components are built from the ground up and specifically for Angular, so that you get high-performance upload controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



## AppBar

A component which provides information and actions related to the current application screen.



## BottomNavigation

A component for navigating among primary destinations in an application.



## BreadCrumb

A component which allows you to navigate within a folder structure or web page.



## ActionSheet

A component which provides a set of predefined options in modal view.

# Angular Navigation Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮

A large blue-bordered placeholder area with a top-left corner icon containing a grid of lines.

```
import { Component } from "@angular/core";
import { ActionSheetItem } from "@progress/kendo-angular-navigation";

@Component({
  selector: "my-actionsheet",
  template: `
    <div class="actionsheet-container">
      <button kendoButton (click)="clickHandler()">Open
      ActionSheet</button>

      <kendo-actionsheet
        title="ActionSheet"
        subtitle="Kendo UI for Angular"
        [items]="actionSheetItems"
        (overlayClick)="onOverlayClick()"
        (itemClick)="onItemClick()"
        [expanded]="opened"
      >
      </kendo-actionsheet>
    </div>
  `,
  styles: [
    `
      .actionsheet-container {
        width: 100%;
        height: 86%;
        transform: scale(1);
      }

      .actionsheet-container > button {
        margin: 20px;
      }
    `,
  ],
})
export class ActionSheetComponent {
  public opened = true;

  public actionSheetItems: ActionSheetItem[] = [
    { title: "Edit Item" },
    { title: "Add to Favorites" },
    { title: "Upload New" },
    { title: "Cancel" },
  ];

  public clickHandler(): void {
    this.opened = true;
  }

  public onOverlayClick(): void {
    this.opened = false;
  }
}
```

```
}

public onItemClick(): void {
    this.opened = false;
}
}
```

Collapse Code ^

# Angular Navigation Key Features

Each Kendo UI for Angular Navigation component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Navigation library as well as develop new features and controls to it.

## Items Configuration

You can supply a collection of elements to the Navigation components to define and configure their items. [Read more about the items configuration of the ActionSheet...](#)

## Appearance

You can customize the appearance of the Navigation items by displaying icons or images, or by changing their colors. [Read more about the BottomNavigation appearance...](#)

## Templates

The Navigation components offer built-in options that allow you to customize their appearance to match your preferences and design requirements. [Read more about the templates of the ActionSheet...](#)

## Position Modes

The Navigation components allow you to specify their position in relation to the page content. [Read more about the position modes of the BottomNavigation...](#)

## Routing

You can utilize [Angular Router](#) and use the components from the Kendo UI for Angular Navigation as a navigational component. [Read more about the routing of the BottomNavigation...](#)

## Globalization

The Kendo UI for Angular Navigation support globalization to ensure that each component can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Navigation components support rendering in a right-to-left (RTL) direction. [Read more about the globalization feature of the Navigation components...](#)

## Accessibility

The Navigation components are accessible for screen readers and support WAI-ARIA attributes. [Read more about accessibility support of the BottomNavigation...](#)

## Keyboard Navigation

The Navigation components (except the AppBar) support a number of keyboard shortcuts which allow users to accomplish various commands. [Read more about the keyboard navigation of the BottomNavigation...](#)

# Trial Version of Kendo UI for Angular

The Kendo UI for Angular Navigation components are part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no restrictions! What's more, you are eligible for full technical

support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

# Support Options

For any questions about the use of Kendo UI for Angular Navigation, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

# Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Navigation](#)
- [API Reference of the Gantt](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)

- Knowledge Base

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Notification

This guide provides the information you need to start using the Kendo UI for Angular Notification—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version **17.0.0**, Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#)

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import {
  KENDO_NOTIFICATION,
  NotificationService,
} from "@progress/kendo-angular-notification";
import { KENDO_BUTTON } from "@progress/kendo-angular-buttons";

@Component({
```

```
standalone: true,
selector: "my-app",
imports: [KENDO_NOTIFICATION, KENDO_BUTTON],
template: `<button kendoButton (click)="show()">Save data</button> `,
})
export class AppComponent {
  constructor(private notificationService: NotificationService) {}

  public show(): void {
    this.notificationService.show({
      content: "Your data has been saved. Time for tea!",
    });
  }
}
```

Collapse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Component

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Notification package:

1. Run the following command:

```
ng add @progress/kendo-angular-notification
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-notification` package as a dependency to the `package.json` file.
- Add all required peer dependencies to the `package.json` file.
- Register the Kendo UI Default theme in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

2. Import the `KENDO_NOTIFICATION` utility array in your standalone component to enable the entire feature set of the Notification:

The utility array is available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

```
import { Component } from '@angular/core';
import { KENDO_NOTIFICATION } from '@progress/kendo-angular-notification';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_NOTIFICATION]
})
```

TS

## Using the Component

1. After successfully installing the Notification package and importing its component, add a custom button in the `app.component.html` file:

```
<button kendoButton (click)="show()">Save data</button>
```

HTML

2. Add the following code in the `app.component.ts` file to show the Notification:

```
constructor(private notificationService: NotificationService) {}

public show(): void {
    this.notificationService.show({
        content: 'Your data has been saved. Time for tea!'
    });
}
```

TS

3. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

4. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular Notification component on the page.

## Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

## Next Steps

- [Notification Dependencies & Standalone Utilities](#)
- [API Reference of the Notification](#)

## Learning Resources

- [Notification Overview](#)
- [Angular Notification Homepage](#)

- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Angular Notification Overview

The Kendo UI for Angular Notification is a brief message, which provides information about the status of an application process.

The Notification service is built from the ground up and specifically for Angular, so that you get a high-performance control which integrates tightly with your application and with the rest of the Kendo UI for Angular components.

## Angular Notification Example

 EXAMPLE

 VIEW SOURCE

⋮

```
import {  
  Component,  
  ViewChild,  
  TemplateRef,  
  ViewEncapsulation,  
} from "@angular/core";  
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";  
import { KENDO_ICONS } from "@progress/kendo-angular-icons";  
import {  
  KENDO_NOTIFICATION,  
  NotificationRef,  
  NotificationService,  
  NotificationSettings,  
} from "@progress/kendo-angular-notification";  
import {  
  chevronDoubleUpIcon,  
  questionCircleIcon,  
  SVGIcon,  
} from "@progress/kendo-svg-icons";
```

```
@Component({
  selector: "my-app",
  encapsulation: ViewEncapsulation.None,
  standalone: true,
  imports: [KENDO_NOTIFICATION, KENDO_BUTTONS, KENDO_ICONS],
  styleUrl: "./styles.css",
  template: `
    <link
      href="https://fonts.googleapis.com/css2?
family=Montserrat:wght@400;700&display=swap"
      rel="stylesheet"
    />
    <div class="row example-wrapper">
      <div class="col-sm-12 example-col">
        <p>Show Default Notification Types</p>
        <p>
          <button
            kendoButton
            (click)="showNotification('success')"
            themeColor="success"
          >
            Success
          </button>
          <button
            kendoButton
            (click)="showNotification('error')"
            themeColor="error"
          >
            Error
          </button>
          <button
            kendoButton
            (click)="showNotification('info')"
            themeColor="info"
          >
            Info
          </button>
          <button
            kendoButton
            (click)="showNotification('warning')"
            themeColor="warning"
          >
            Warning
          </button>
          <button kendoButton (click)="showNotification('default')">
            Default
          </button>
        </p>
      </div>
      <div class="col-sm-12 example-col">
        <p>Show Styled Notification</p>
      </div>
    </div>
  </div>
})
```

```

<p>
  <button
    kendoButton
    (click)="showStyledNotification()"
    style="background-color: #a87716; color: white">
    </button>
</p>
<ng-template #notification>
  <div class="notification-box">
    <div class="text-content">
      <div>
        <kendo-svgicon
          [icon]="svgIcon"
          size="large"
          style="margin-top: -5px;"></kendo-svgicon>
        <span class="header">Did you check our latest release?</span>
        <kendo-svgicon
          class="close"
          [icon]="closeIcon"
          (click)="close()"
          style="height: 20px; width: 20px;"></kendo-svgicon>
      </div>
      <div style="margin-left:25px;">
        <p>Don't miss the new features!</p>
        <a
          href="https://www.telerik.com/kendo-angular-ui/components/changelogs/kendo-angular-ui"
          target="blank"
          >Learn more</a>
      </div>
    </div>
  </ng-template>
</div>
</div>
`,
})
export class AppComponent {
  @ViewChild("notification", { read: TemplateRef }) notificationTemplate: TemplateRef<any>;
  public notificationReference: NotificationRef;
  public svgIcon: SVGIcon = questionCircleIcon;
  public closeIcon: SVGIcon = chevronDoubleUpIcon;
  public state: NotificationSettings = {
    content: "Your data has been saved.",
  }
}

```

```
        type: { style: "success", icon: true },
        animation: { type: "slide", duration: 400 },
        hideAfter: 3000,
    };

constructor(private notificationService: NotificationService) {}

public close(): void {
    this.notificationReference.hide();
}

public showStyledNotification(): void {
    this.notificationReference = this.notificationService.show({
        content: this.notificationTemplate,
        cssClass: "wrapper",
        animation: { type: "fade", duration: 200 },
        position: { horizontal: "right", vertical: "top" },
        hideAfter: 5000,
        width: 400,
    });
}

public showNotification(type): void {
    switch (type) {
        case "success":
            this.state.content = "Your data has been saved.";
            this.state.type = { style: "success", icon: true };
            break;
        case "error":
            this.state.content = "Oops, something went wrong...";
            this.state.type = { style: "error", icon: true };
            break;
        case "warning":
            this.state.content = "Your password is about to expire.";
            this.state.type = { style: "warning", icon: true };
            break;
        case "info":
            this.state.content = "You have 1 new message.";
            this.state.type = { style: "info", icon: true };
            break;
        case "default":
            this.state.content = "John Smith reacts on your photo.";
            this.state.type = { style: "none", icon: true };
            break;
    }
    this.notificationService.show(this.state);
}
}
```

Collanse Code ^

# Angular Notification Key Features

The Kendo UI for Angular Notification service delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Notification library as well as develop new features to it.

## Angular Notification Types

The Notification features built-in types, which enable you to display different notification variants such as success or error. [Read more about the built-in Notification types...](#)

## Angular Notification Animations

The Notification allows you to use different built-in animation effects, which enable you to visualize the display and disappearance of the notifications. [Read more about the built-in animations of the Notification...](#)

## Controlling the Angular Notification Content

The Notification provides options for controlling its content. [Read more about the different content options of the Notification...](#)

## Angular Notification Positioning

The Notification enables you to align it to the browser viewport and to position it to a specific container. [Read more about the positioning options of the Notification...](#)

## Hiding The Angular Notification

You can hide the Notification by defining a delay period before the hiding occurs or by defining a closable Notification component. [Read more about the hiding options of the Notification...](#)

## Appending to Dynamic Containers

With the Notification service, you can also define a container to which the Notification will be appended. [Read more about the appending functionality of the Notification...](#)

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular Notification is part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library —no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of the Kendo UI for Angular Notification, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).

- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Notification Homepage](#)
- [Getting Started with the Kendo UI for Angular Notification](#)
- [API Reference of the Pager](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Notification Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Pager

This guide provides the information you need to start using the Kendo UI for Angular Pager—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default.

If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#)

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_PAGER, PageChangeEvent } from "@progress/kendo-angular-pager";

@Component({
  standalone: true,
  selector: "my-app",
  imports: [KENDO_PAGER],
  template: `
    <kendo-pager
      [total]="50"
    >
  
```

```
[pageSize]="pageSize"
[skip]="skip"
(pageChange)="onPageChange($event)"
>
</kendo-pager>
,
})
export class AppComponent {
public skip = 0;
public pageSize = 10;

public onPageChange(e: PageChangeEvent): void {
  this.skip = e.skip;
  this.pageSize = e.take;
}
}
```

Collanse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Component

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Pager package:

1. Run the following command:

```
ng add @progress/kendo-angular-pager
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-pager` package as a dependency to the `package.json` file.
- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

2. Import the `KENDO_PAGER` utility array in your standalone component to enable the entire feature set of the Pager:

The utility array is available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

```
import { Component } from '@angular/core';
import { KENDO_PAGER } from '@progress/kendo-angular-pager';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_PAGER]
})
```

TS

# Using the Component

1. After successfully installing the Pager package and importing its component, add the following code in the `app.component.html` file:

```
<kendo-pager  
    [total]="50"  
    [pageSize]="pageSize"  
    [skip]="skip"  
    (pageChange)="onPageChange($event)"  
>  
</kendo-pager>
```

HTML

2. Bind the `total` option to the total number of data items in the collection, the `pageSize` option to define the number of the data items per page, and the `skip` option to define the number of the items that will be skipped.

3. Handle the `pageChange` event and update the `skip` and `pageSize` options.

```
public skip = 0;  
public pageSize = 10;  
  
public onPageChange(e: PageChangeEvent): void {  
    this.skip = e.skip;  
    this.pageSize = e.take;  
}
```

TS

4. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

5. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular Pager component on the page.

## Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key.

If your application does not contain a Kendo UI license file, [activate your license key](#).

# Next Steps

- [Utilizing the Pager settings and types](#)
- [Rendering the responsive Pager design](#)
- [Using the Pager template](#)
- [API Reference of the Pager](#)

# Learning Resources

- [Pager Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Angular Pager Overview

The Kendo UI for Angular Pager component enables you to split a set of data into pages, providing a flexible and intuitive UI.

The Pager is built from the ground up and specifically for Angular, so that you get a high-performance control which integrates tightly with your application and with the rest of the Kendo UI for Angular components.

## Angular Pager Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, OnInit } from "@angular/core";
import { FormsModule } from "@angular/forms";
import { KENDO_PAGER, PageChangeEvent } from "@progress/kendo-angular-pager";
import { DestinationComponent } from "./destination.component";
import { destinations, Destination } from "./destinations";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_PAGER, FormsModule, DestinationComponent],
  template:
    <span class="title"> Top European Destinations </span>
    <div class="wrapper">
      <div class="content-container" [id]="contentId">
        @for (destination of pagedDestinations; track destination) {
          <destination-component [data]="destination"> </destination-component>
        }
      </div>
      <kendo-pager>
```

```

[attr.aria-controls]="contentId"
[style.width.%] = "100"
[pageSize] = "pageSize"
[skip] = "skip"
[total] = "total"
(pageChange) = "onPageChange($event)"
>
</kendo-pager>
</div>
` ,
styleUrls: ["styles.css"],
})
export class AppComponent implements OnInit {
public pageSize = 8;
public skip = 0;
public pagedDestinations: Destination[] = [];
public total = destinations.length;
public contentId = "content-1";

public ngOnInit(): void {
  this.pageData();
}

public onPageChange(e: PageChangeEvent): void {
  this.skip = e.skip;
  this.pageSize = e.take;
  this.pageData();
}

private pageData(): void {
  this.pagedDestinations = destinations.slice(
    this.skip,
    this.skip + this.pageSize
  );
}
}

```

[Collapse Code ^](#)

## Angular Pager Key Features

The Kendo UI for Angular Pager component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Pager library as well as develop new features to it.

# Pager Settings and Types

The Pager component features multiple settings to customize the paging UI. [Read more about the available settings configurations of the Pager...](#)

## Pager Appearance

You can change the size of the Pager or fully customize its appearance by mixing custom components and built-in pager elements by using the built-in Pager template. Read more about the [Pager template](#) and the [built-in sizing options...](#)

## Responsive Design

The Pager supports responsive web-design by adapting its layout based on the set width or the available screen size. [Read more about the adaptive layout of the Pager...](#)

## Globalization

The Kendo UI for Angular Pager supports globalization to ensure that it can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Pager supports rendering in a right-to-left (RTL) direction. [Read more about the globalization of the Pager...](#)

## Keyboard Navigation

The Pager supports a number of keyboard shortcuts which allow users to accomplish various commands. [Read more about the keyboard navigation of the Pager...](#)

## Accessibility

The Kendo UI for Angular Pager component is [WCAG 2.2 AA](#) and [Section 508](#) compliant, incorporating [WAI-ARIA](#) best practices for modern web accessibility. The Pager is also tested against the popular screen readers, including JAWS and NVDA. Read more about the accessibility of the Grid...

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular Pager is part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of the Kendo UI for Angular Pager, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

# Support and Learning Resources

- [Pager Homepage](#)
- [Getting Started with the Kendo UI for Angular Pager](#)
- [API Reference of the Pager](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Pager Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Pager Settings and Types

The Pager component features multiple settings to customize the paging UI.

Starting with [v17.0.0](#), the selector for the Pager has been updated from `kendo-datapager` to `kendo-pager`.

## Types

The pager types are:

- `Numeric`—Renders buttons with numbers.
- `Input`—Renders a `NumericUpDown` for entering the page number.

To configure the pager type and number of numeric buttons, use the `type` and `buttonCount` options:

- `type`—Accepts the `numeric` (buttons with numbers) and `input` (a `NumericUpDown` for entering the page number) values.
- `buttonCount`—Sets the maximum number of numeric buttons that will be rendered when `type` is `numeric`.

## Settings

The following settings enable you to determine which built-in pager elements will be rendered:

- `pageSizeValues`—Toggles a `DropDownList` with page size options, and allows you to provide the list of available page sizes.
- `previousNext`—Toggles a set of `Previous` and `Next` buttons.

- **info**—Toggles the information about the current page and the total number of records.

The following example demonstrates all available Pager options in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component, OnInit, ViewChild, TemplateRef } from
"@angular/core";

import { FormsModule } from "@angular/forms";

import {
  KENDO_PAGER,
  PageChangeEvent,
  PagerType,
} from "@progress/kendo-angular-pager";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_NOTIFICATION } from "@progress/kendo-angular-
notification";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { NotificationService } from "@progress/kendo-angular-
notification";

import { ArticleComponent } from "./article.component";
import { articles } from "./articles";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [
    KENDO_PAGER,
    FormsModule,
    KENDO_INPUTS,
    KENDO_NOTIFICATION,
    KENDO_BUTTONS,
    ArticleComponent,
  ],
  template: `
    <div class="demo-wrapper">
      <div>
        <div class="checkbox-wrapper">
          <label>
            <><input type="checkbox" kendoCheckBox
[(ngModel)]="pageSizes" />
```

```
        Show page size options</label>
    >
<label>
    <input type="checkbox" kendoCheckBox [(ngModel)]="info" />
Show
    info</label>
    >
<label>
    <input type="checkbox" kendoCheckBox [(ngModel)]="prevNext" />
/> Show
    previous/next buttons</label>
    >
</div>
<label>
    <input type="radio" kendoRadioButton value="numeric" [(ngModel)]="type" />
    Numeric buttons</label>
    >
<label>
    <input type="radio" kendoRadioButton value="input" [(ngModel)]="type" />
    Numeric input</label>
    >
</div>
<ng-template #template>Article {{ notificationText }}.</ng-template>
<div class="articles-wrapper">
    <span class="articles-title"> Trending Articles This Week </span>
</div>
<div class="articles-container" [id]="contentId">
    @for (article of pagedArticles; track article) {
        <article-component
            [article]="article"
            (showNotification)="this.onShowNotification($event)"></article-component>
    }
</div>
<kendo-pager
    style="width: 770px"
    [attr.aria-controls]="contentId"
    [pageSize]="pageSize"
    [skip]="skip"
    [total]="total"
```

```
[pageSizeValues]="pageSizes"
[info]="info"
[previousNext]="prevNext"
[type]="type"
(pageChange)="onPageChange($event)"
>
</kendo-pager>
</div>
</div>
`,
styleUrls: ["styles.css"],
})
export class AppComponent implements OnInit {
@ViewChild("template", { read: TemplateRef })
public notificationTemplate: TemplateRef<unknown>;
public pageSize = 2;
public skip = 0;
public pagedArticles = [];
public total = articles.length;
public pageSizes = false;
public info = true;
public prevNext = true;
public type: PagerType = "numeric";
public notificationText: string;
public contentId = "content-1";

constructor(private notificationService: NotificationService) {}

public ngOnInit(): void {
  this.pageData();
}

public onPageChange(e: PageChangeEvent): void {
  this.skip = e.skip;
  this.pageSize = e.take;
  this.pageData();
}

public onShowNotification(text: string): void {
  this.notificationText = text;
  this.notificationService.show({
    content: this.notificationTemplate,
    hideAfter: 600,
    position: { horizontal: "center", vertical: "bottom" },
    animation: { type: "fade", duration: 400 },
    type: { style: "info", icon: true },
  });
}

private pageData(): void {
  this.pagedArticles = articles.slice(this.skip, this.skip +
```

```
this.pageSize);  
}  
}
```

Collanse Code ^

# Suggested Links

- [API Reference of the Pager](#)

# Pager Template

To customize the pager appearance, use the [PagerTemplateDirective](#).

The directive enables you to take full control over the content of the Pager by mixing the built-in pager elements and custom components.

Starting with [v17.0.0](#), the Pager selectors have been updated to improve clarity and consistency. The following breaking changes affect the pager template:

Deprecated Selector	Updated Selector
kendoDataPagerTemplate	kendoPagerTemplate
kendo-datapager-pager-prev-buttons	kendo-pager-prev-buttons
kendo-datapager-pager-numeric-buttons	kendo-pager-numeric-buttons
kendo-datapager-pager-next-buttons	kendo-pager-next-buttons
kendo-datapager-pager-input	kendo-pager-input
kendo-datapager-pager-info	kendo-pager-info
kendo-datapager-pager-page-sizes	kendo-pager-page-sizes
kendo-datapager-pager-spacer	kendo-pager-spacer

The template context exposes the following fields:

- `currentPage`—The index of the displayed page.
- `pageSize`—The value of the current `pageSize`.
- `skip`—The current `skip` value.
- `total`—The total number of records.
- `totalPages`—The total number of available pages.

# Using Built-in Pager Components

You can customize the Pager by using the following built-in components:

- `PagerNumericButtonsComponent`—Displays numeric buttons for navigation between the pages.
- `PagerInputComponent`—Displays an input element that allows typing and rendering of page numbers.
- `PagerPageSizesComponent`—Displays a drop-down list for the page size selection.
- `PagerInfoComponent`—Shows information about the current page and the total number of records.
- `PagerPrevButtonsComponent`—Displays buttons for navigating to the first and previous page.
- `PagerNextButtonsComponent`—Displays buttons for navigating to the next and the last page.

Using the Pager template disables its default responsive behavior.

The following example demonstrates how to define the built-in pager components inside an `<ng-template>` tag with applied `kendoPagerTemplate` directive.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";

import { KENDO_PAGER, PageChangeEvent } from "@progress/kendo-angular-
pager";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_PAGER],
  template: `
    <kendo-pager
      style="width: 100%;"
      [pageSize]="pageSize"
    >
  
```

```

[skip]="skip"
[total]="total"
(pageChange)="onPageChange($event)"
>
<ng-template
  kendoPagerTemplate
  let-totalPages="totalPages"
  let-currentPage="currentPage"
>
  <div class="k-pager-numbers-wrap">
    <kendo-pager-prev-buttons></kendo-pager-prev-buttons>
    <kendo-pager-numeric-buttons
      [buttonCount]="buttonCount"
    ></kendo-pager-numeric-buttons>
    <kendo-pager-next-buttons></kendo-pager-next-buttons>
  </div>

  <kendo-pager-input></kendo-pager-input>
  <kendo-pager-info></kendo-pager-info>
  <kendo-pager-page-sizes [pageSizes]="sizes"></kendo-pager-page-
sizes>
</ng-template>
</kendo-pager>
,
})
export class AppComponent {
  public pageSize = 5;
  public buttonCount = 2;
  public sizes = [10, 20, 50];
  public skip = 0;
  public total = 100;

  public onPageChange(e: PageChangeEvent): void {
    this.skip = e.skip;
    this.pageSize = e.take;
  }
}

```

[Collapse Code ^](#)

## Using Other Components

In some scenarios, the built-in pager elements do not entirely meet your requirements. In such cases, the [PagerTemplateDirective](#) enables you to use any set of Angular components and DOM elements that fit the specific use case.

The following example demonstrates how to use the Kendo UI for Angular Slider to paginate the data.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { FormsModule } from "@angular/forms";
import { KENDO_PAGER, PageChangeEvent } from "@progress/kendo-angular-pager";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_PAGER, FormsModule, KENDO_INPUTS],
  template:
    <kendo-pager
      style="width: 100%;"
      [pageSize]="pageSize"
      [skip]="skip"
      [total]="total"
      (pageChange)="onPageChange($event)"
    >
      <ng-template
        kendoPagerTemplate
        let-totalPages="totalPages"
        let-currentPage="currentPage"
      >
        <kendo-pager-prev-buttons
          style="margin-right: 10px;"
        ></kendo-pager-prev-buttons>
        <kendo-slider
          [showButtons]="false"
          tickPlacement="none"
          [max]="totalPages"
          [min]="1"
          (valueChange)="sliderChange($event)"
          [value]="currentPage"
        >
        </kendo-slider>
        <kendo-pager-next-buttons
          style="margin-left: 10px;"
        ></kendo-pager-next-buttons>
      </ng-template>
    </kendo-pager>
  </div>
  Page {{ currentPage }} of {{ totalPages }}
```

```
        <kendo-pager-info></kendo-pager-info>
        </ng-template>
    </kendo-pager>
  ,
})
export class AppComponent {
  public pageSize = 10;
  public skip = 0;
  public total = 100;

  public sliderChange(pageIndex: number): void {
    this.skip = (pageIndex - 1) * this.pageSize;
  }

  public onPageChange(e: PageChangeEvent): void {
    this.skip = e.skip;
    this.pageSize = e.take;
  }
}
```

Collanse Code ^

## Suggested Links

- [API Reference of the Pager](#)
- [API Reference of the PagerTemplateDirective](#)

# Getting Started with the Kendo UI for Angular Ripple

This guide provides the information you need to start using the Kendo UI for Angular Ripple—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#)

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_RIPPLE } from "@progress/kendo-angular-ripple";
import { KENDO_BUTTON } from "@progress/kendo-angular-buttons";

@Component({
  standalone: true,
  selector: "my-app",
  imports: [KENDO_RIPPLE, KENDO_BUTTON],
  template: `
    <div kendoRippleContainer>
      <button kendoButton>Default Button</button>
```

```
</div>
},
})
export class AppComponent {}
```

Collanse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Component

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Ripple package:

### 1. Run the following command:

```
ng add @progress/kendo-angular-ripple
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-ripple` package as a dependency to the `package.json` file.
- Add all required `peer dependencies` to the `package.json` file.

- Register the Kendo UI Default theme in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

2. Import the `KENDO_RIPPLE` utility array in your standalone component to enable the entire feature set of the Ripple:

The utility array is available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

```
import { Component } from '@angular/core';
import { KENDO_RIPPLE } from '@progress/kendo-angular-ripple';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_RIPPLE]
})
```

TS

## Using the Component

1. After successfully installing the Ripple package and importing its directive, add the following code in the `app.component.html` file:

```
<div kendoRippleContainer>
  <button kendoButton>Default Button</button>
</div>
```

HTML

2. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

3. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular Ripple component on the page.

# Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

## Next Steps

- [Ripple Dependencies & Standalone Utilities](#)
- [RippleContainerDirective](#)
- [API Reference of the Ripple](#)

## Learning Resources

- [Ripple Overview](#)
- [Angular Ripple Homepage](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Angular Ripple Overview

The Kendo UI for Angular Ripple directive provides the [Material ink ripple effect](#) for the Kendo UI components for Angular and is fully compatible with all available Kendo UI themes.

The Ripple directive is built from the ground up and specifically for Angular, so that you get a high-performance directive which integrates tightly with your application and with the rest of the Kendo UI for Angular components.

## Angular Ripple Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";

import { KENDO_RIPPLE } from "@progress/kendo-angular-ripple";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_RIPPLE, KENDO_BUTTONS, KENDO_INPUTS],
  template: `
    <div class="example-wrapper">
      <div kendoRippleContainer>
        <!-- You can apply this directive to any container element --&gt;

        &lt;div class="row"&gt;
          &lt;div class="col-xs-12 col-sm-4 example-col"&gt;
            &lt;p&gt;Ripple on Buttons&lt;/p&gt;
            &lt;button kendoButton&gt;Default Button&lt;/button&gt;&lt;br /&gt;
            &lt;button kendoButton themeColor="primary"&gt;Primary
              Button&lt;/button&gt;
            &gt;&lt;br /&gt;
        &lt;/div&gt;
    &lt;/div&gt;
  `})
export class AppComponent { }</pre>
```

```

        <button kendoButton fillMode="flat">Flat Button</button>
    </div>
    <div class="col-xs-12 col-sm-4 example-col">
        <p>Ripple on Checkboxes</p>
        <p style="line-height: 2.5em;">
            <input kendoCheckBox type="checkbox" id="c1" />
            <label class="k-checkbox-label" for="c1">Checkbox
1</label><br />
            <input kendoCheckBox type="checkbox" id="c2" />
            <label class="k-checkbox-label" for="c2">Checkbox
2</label><br />
            <input kendoCheckBox type="checkbox" id="c3" />
            <label class="k-checkbox-label" for="c3">Checkbox
3</label>
        </p>
    </div>
    <div class="col-xs-12 col-sm-4 example-col">
        <p>Ripple on Radio Buttons</p>
        <p style="line-height: 2.5em;">
            <input kendoRadioButton type="radio" id="r1" name="rg"
checked />
            <label class="k-radio-label" for="r1">Radio 1</label><br />
            <input kendoRadioButton type="radio" id="r2" name="rg" />
            <label class="k-radio-label" for="r2">Radio 2</label><br />
            <input kendoRadioButton type="radio" id="r3" name="rg" />
            <label class="k-radio-label" for="r3">Radio 3</label>
        </p>
    </div>
    </div>
    ,
    styles: [
        ,
        button,
        input {
            margin: 1em 0;
        }
        ,
    ],
})
export class AppComponent {}

```

[Collapse Code ▲](#)

# Trial Version of Kendo UI for Angular

The Kendo UI for Angular Ripple is part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of the Kendo UI for Angular Ripple, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Ripple Homepage](#)
- [Getting Started with the Kendo UI for Angular Ripple](#)
- [API Reference of the Ripple](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)

- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Ripple Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Active Items

The ScrollView enables you to set an active item to it.

To control the active ScrollView item, use the `activeIndex` property. By default, `activeIndex` displays the first ScrollView item.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { CommonModule } from "@angular/common";
import { Component } from "@angular/core";
import { FormsModule } from "@angular/forms";
import { KENDO_SCROLLVIEW } from "@progress/kendo-angular-scrollview";
import { data, Item } from "./data";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_SCROLLVIEW, FormsModule],
  template: `
    <kendo-scrollview
      [data]="items"
      [width]="width"
      [height]="height"
      [activeIndex]="activeIndex"
    >
      <ng-template let-item="item">
        <h2 class="demo-title">{{ item.title }}</h2>
        
      </ng-template>
    </kendo-scrollview>
  `,
  styles: [
    ".k-scrollbar-wrap {"
  ]
})
```

```
    margin: 0 auto;
}
.demo-title {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  margin: 0;
  padding: 15px;
  color: #fff;
  background-color: rgba(0, 0, 0, 0.4);
  text-align: center;
  font-size: 24px;
}
,
])
export class AppComponent {
  public items: Item[] = data;
  public width = "100%";
  public height = "400px";
  public activeIndex = 1;
}
```

[Collapse Code ^](#)

# Suggested Links

- [API Reference of the ScrollView](#)

# Data Sources

The ScrollView enables you to provide it with a data source.

To determine a data source for the ScrollView, use its `data` property.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { CommonModule } from "@angular/common";
import { Component } from "@angular/core";
import { KENDO_SCROLLVIEW } from "@progress/kendo-angular-scrollview";
import { data, Item } from "./data";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_SCROLLVIEW],
  template: `
    <kendo-scrollview [data]="items" [width]="width" [height]="height">
      <ng-template let-item="item">
        <h2 class="demo-title">{{ item.title }}</h2>
        
      </ng-template>
    </kendo-scrollview>
  `,
  styles: [
    .k-scrollbar-wrap {
      margin: 0 auto;
    }
    .demo-title {
      position: absolute;
      top: 0;
      left: 0;
      right: 0;
    }
  ]
})
```

```
        margin: 0;
        padding: 15px;
        color: #fff;
        background-color: rgba(0, 0, 0, 0.4);
        text-align: center;
        font-size: 24px;
    }
},
],
})
export class AppComponent {
    public items: Item[] = data;
    public width = "100%";
    public height = "400px";
}
```

Collanse Code ^

# Suggested Links

- [API Reference of the ScrollView](#)

# Dimensions

The ScrollView enables you to define its dimensions.

To specify its width and height, use the `width` and `height` properties of the ScrollView. The `width` and `height` settings do not have default values and you have to set both of them.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { CommonModule } from "@angular/common";
import { Component } from "@angular/core";
import { KENDO_SCROLLVIEW } from "@progress/kendo-angular-scrollview";
import { data, Item } from "./data";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_SCROLLVIEW],
  template: `
    <kendo-scrollview [data]="items" [width]="width" [height]="height">
      <ng-template let-item="item">
        <h2 class="demo-title">{{ item.title }}</h2>
        
      </ng-template>
    </kendo-scrollview>
  `,
  styles: [
    .k-scrollbar-wrap {
      margin: 0 auto;
    }
    .demo-title {
      position: absolute;
    }
  ]
})
```

```
        top: 0;
        left: 0;
        right: 0;
        margin: 0;
        padding: 15px;
        color: #fff;
        background-color: rgba(0, 0, 0, 0.4);
        text-align: center;
        font-size: 24px;
    }
},
],
})
export class AppComponent {
    public items: Item[] = data;
    public width = "100%";
    public height = "400px";
}
```

Collanse Code ^

# Suggested Links

- [API Reference of the ScrollView](#)

# Endless Scrolling

The ScrollView provides a scrolling mode which loops through its views in an endless fashion.

To enable or disable the endless scrolling functionality, set the `endless` property. By default, `endless` is set to `false` and the endless scrolling mode is disabled.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { CommonModule } from "@angular/common";
import { Component } from "@angular/core";
import { FormsModule } from "@angular/forms";
import { KENDO_SCROLLVIEW } from "@progress/kendo-angular-scrollview";
import { KENDO_LABEL } from "@progress/kendo-angular-label";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { data, Item } from "./data";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [
    CommonModule,
    FormsModule,
    KENDO_SCROLLVIEW,
    KENDO_LABEL,
    KENDO_INPUTS,
  ],
  template: `
    <div class="example-config">
      <input #checkbox kendoCheckBox type="checkbox"
        [(ngModel)]="endless" />
      <kendo-label
        [for]="checkbox"
        class="k-checkbox-label"
        text="Enable Endless Scrolling Mode"
      ></kendo-label>
    </div>
    <kendo-scrollview
```

```

        [data]="items"
        [width]="width"
        [height]="height"
        [arrows]="true"
        [endless]="endless"
    >
    <ng-template let-item="item">
        <h2 class="demo-title">{{ item.title }}</h2>
        
    </ng-template>
</kendo-scrollview>
,
styles: [
    .k-scrollbar-wrap {
        margin: 0 auto;
    }
    .demo-title {
        position: absolute;
        top: 0;
        left: 0;
        right: 0;
        margin: 0;
        padding: 15px;
        color: #fff;
        background-color: rgba(0, 0, 0, 0.4);
        text-align: center;
        font-size: 24px;
    }
],
})
export class AppComponent {
    public items: Item[] = data;
    public width = "100%";
    public height = "400px";
    public endless = false;
}

```

[Collapse Code ^](#)

# Automatic Endless Scrolling

To implement an automatic endless scrolling, use the `next` method of the `ScrollView`, wrapped in a `setInterval` callback.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮

```
import { CommonModule } from "@angular/common";
import { AfterViewInit, Component, OnDestroy, ViewChild } from
"@angular/core";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_SCROLLVIEW } from "@progress/kendo-angular-scrollview";
import { data, Item } from "./data";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_SCROLLVIEW, KENDO_BUTTONS],
  template:
    <kendo-scrollview
      #sv
      [data]="items"
      [width]="width"
      [height]="height"
      [arrows]="true"
      [endless]="true"
    >
      <ng-template let-item="item">
        <h2 class="demo-title">{{ item.title }}</h2>
        
      </ng-template>
    </kendo-scrollview>
    <button kendoButton (click)="paused = !paused">
      {{ paused ? "Resume" : "Pause" }}
    </button>
  ,
  styles: [
    .k-scrollbar-wrap {
      margin: 0 auto;
    }
    .demo-title {
      position: absolute;
    }
  ]
})
```

```
        top: 0;
        left: 0;
        right: 0;
        margin: 0;
        padding: 15px;
        color: #fff;
        background-color: rgba(0, 0, 0, 0.4);
        text-align: center;
        font-size: 24px;
    }
},
],
})
export class AppComponent implements OnDestroy, AfterViewInit {
    @ViewChild("sv") private scrollView;
    public paused = false;
    public items: Item[] = data;
    public width = "100%";
    public height = "500px";
    private interval;

    public ngAfterViewInit(): void {
        this.interval = setInterval(() => {
            if (!this.paused) {
                this.scrollView.next();
            }
        }, 3000);
    }

    public ngOnDestroy(): void {
        clearInterval(this.interval);
    }
}
```

Collanse Code ^

# Suggested Links

- [API Reference of the ScrollView](#)

# Getting Started with the Kendo UI for Angular ScrollView

This guide provides the information you need to start using the Kendo UI for Angular ScrollView—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#)

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_SCROLLVIEW } from "@progress/kendo-angular-scrollview";

@Component({
  standalone: true,
  selector: "my-app",
  imports: [KENDO_SCROLLVIEW],
  template: `
```

```
<kendo-scrollview
  [data]="items"
  [width]="width"
  [height]="height"
  [arrows]="true"
>
  <ng-template let-item="item">
    
  </ng-template>
</kendo-scrollview>
,
})
export class AppComponent {
  public items: Array<{ url: string }> = [
    {
      url: "assets/layout/card/kukeri.jpg",
    },
    {
      url: "assets/layout/card/martenitsa.jpg",
    },
    {
      url: "assets/layout/card/rose_festival.jpg",
    },
  ];
  public width = "100%";
  public height = "500px";
}
```

Collanse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

# Installing the Component

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves

time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular ScrollView package:

### 1. Run the following command:

```
ng add @progress/kendo-angular-scrollview
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-scrollview` package as a dependency to the `package.json` file.
- Add all required `peer dependencies` to the `package.json` file.
- Register the [Kendo UI Default theme](#) in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

### 2. Import the `KENDO_SCROLLVIEW` utility array in your standalone component to enable the entire feature set of the ScrollView:

The utility array is available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

```
import { Component } from '@angular/core';
import { KENDO_SCROLLVIEW } from '@progress/kendo-angular-
scrollview';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_SCROLLVIEW]
})
```

TS

# Using the Component

- After successfully installing the `ScrollView` package and importing its component, add the following code in the `app.component.html` file:

```
<kendo-scrollview  
  [data]="items"  
  width="356px"  
  height="238px"  
  [arrows]="true"  
>  
<ng-template let-item="item">  
  <img width="100%" src='{{item.url}}'/>  
</ng-template>  
</kendo-scrollview>
```

HTML

- Bind the `data` property to array of objects by adding the following code in the `app.component.ts` file:

```
public items: Array<{ url: string }> = [  
  {  
    url: 'https://www.telerik.com/kendo-angular-ui-  
develop/components/layout/card/assets/kukeri.jpg',  
  },  
  {  
    url: 'https://www.telerik.com/kendo-angular-ui-  
develop/components/layout/card/assets/martenitsa.jpg',  
  },  
  {  
    url: 'https://www.telerik.com/kendo-angular-ui-  
develop/components/layout/card/assets/rose_festival.jpg',  
  },  
];
```

TS

- Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

4. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular ScrollView component on the page.

# Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

# Next Steps

- Setting the ScrollView [animation](#) and [active item](#)
- [Using the arrows of the ScrollView](#)
- Applying the [paging feature](#) and the [endless scrolling mode](#) of the ScrollView
- Defining [dimensions](#) and a [data source](#) for the ScrollView
- [Globalization](#)
- [Keyboard navigation](#)
- [Accessibility](#)
- [API Reference of the ScrollView](#)

# Learning Resources

- [ScrollView Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Angular ScrollView Overview

The Kendo UI for Angular ScrollView represents a horizontal collection of content or image views with built-in navigation between them.

The ScrollView is built from the ground up and specifically for Angular, so that you get a high-performance control which delivers lightning-fast performance, integrates tightly with your application and with the rest of the Kendo UI for Angular components, and is highly customizable.

## Angular ScrollView Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { CommonModule } from "@angular/common";
import { Component } from "@angular/core";
import { KENDO_SCROLLVIEW } from "@progress/kendo-angular-scrollview";
import { data, Item } from "./data";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_SCROLLVIEW],
  template: `
    <kendo-scrollview
      [data]="items"
      [width]="width"
      [height]="height"
      [arrows]="true"
      [pageable]="true"
    >
      <ng-template let-item="item">
        <h2 class="demo-title">{{ item.title }}</h2>
        
      </ng-template>
    
  `,
  styles: [
    ".demo-title { font-size: 1.2em; margin-bottom: 10px; }"
  ]
})
export class AppComponent {
  items = data;
  width = "1000px";
  height = "300px";
}
```

```

        [ngStyle]={ minWidth: width }"
        draggable="false"
      />
    </ng-template>
  </kendo-scrollview>
  ,
  styles: [
    .k-scrollbar-wrap {
      margin: 0 auto;
    }
    .demo-title {
      position: absolute;
      top: 0;
      left: 0;
      right: 0;
      margin: 0;
      padding: 15px;
      color: #fff;
      background-color: rgba(0, 0, 0, 0.4);
      text-align: center;
      font-size: 24px;
    }
  ],
)
export class AppComponent {
  public items: Item[] = data;
  public width = "100%";
  public height = "500px";
}

```

[Collapse Code ^](#)

## Angular ScrollView Key Features

The Kendo UI for Angular ScrollView component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests ongoing efforts to improve the performance and add more value to the existing ScrollView component as well as develop new features to it.

## Animations

The ScrollView allows you to enable or disable its built-in animations, which are enabled by default. [Read more about the animations of the ScrollView...](#)

## Arrows

You can also control the behavior of the built-in navigation arrows of the ScrollView, which are disabled by default. [Read more about the navigation arrows functionality of the ScrollView...](#)

## Paging

The ScrollView provides options for enabling or disabling its built-in paging functionality as well as for adding a pager overlay. [Read more about the paging of the ScrollView...](#)

## Endless Scrolling

Furthermore, you can enable the endless scrolling mode which loops through its views in an endless fashion and which is disabled by default. [Read more about the endless scrolling of the ScrollView...](#)

## Dimensions

The ScrollView enables you to define its width and height. [Read more about the dimensions of the ScrollView...](#)

## Data Sources

You can set a data source for the ScrollView to provide it with data. [Read more about the data sources of the ScrollView...](#)

## Active Items

You can render an active item in the ScrollView by using the available configuration of the component. [Read more about setting the active item of the ScrollView...](#)

## Globalization

The Kendo UI for Angular ScrollView supports globalization to ensure that it can fit well in any application, no matter what languages and locales need to be supported. Additionally, the ScrollView supports rendering in a right-to-left (RTL) direction. [Read more about the globalization of the ScrollView...](#)

## Accessibility

The ScrollView is accessible by screen readers and provides WAI-ARIA support. [Read more about the accessibility of the ScrollView...](#)

## Keyboard Navigation

The ScrollView supports a number of keyboard shortcuts for processing various commands. [Read more about the keyboard navigation of the ScrollView...](#)

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular ScrollView is part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library —no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of the Kendo UI for Angular ScrollView, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [ScrollView Homepage](#)
- [Getting Started with the Kendo UI for Angular ScrollView](#)
- [API Reference of the ScrollView](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [ScrollView Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)

- Kendo UI for Angular Roadmap

# Data Binding

The Sortable enables you to bind it to an array of strings or objects, as well as to use its built-in directive to decrease the repetitive boilerplate code.

## Binding to Arrays of Strings and Objects

You can set the `data` property of the Sortable component to an array of strings or objects.

Sorting items by using either drag-and-drop operations or the [keyboard navigation](#) will emit [events](#), such as `dragStart`, `dragOver`, `dragEnd`, and others. The events provide the necessary information for manipulating the dataset manually or through the API of the Sortable.

[EXAMPLE](#)[VIEW SOURCE](#)

:



```
import { Component, ViewEncapsulation, ViewChild } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_SORTABLE } from "@progress/kendo-angular-sortable";
import {
  DragOverEvent,
  NavigateEvent,
  SortableComponent,
} from "@progress/kendo-angular-sortable";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_SORTABLE],
  template: `
    <div class="example-config">
      <h6>Items: {{ items | json }}</h6>
```

```

        </div>
      <div class="container-fluid">
        <kendo-sortable
          #sortable
          [data]="items"
          [navigable]="true"
          [animation]="true"
          (dragOver)="onDragOver($event)"
          (navigate)="onNavigate($event)"
          class="row"
          itemClass="item col-xs-6 col-sm-3"
          activeItemClass="item col-xs-6 col-sm-3 active">
        </kendo-sortable>
      </div>
    ,
  encapsulation: ViewEncapsulation.None,
  styleUrls: ['./styles.css'],
)
export class AppComponent {
  @ViewChild("sortable") public sortable: SortableComponent;

  public items: string[] = [
    "Item 1",
    "Item 2",
    "Item 3",
    "Item 4",
    "Item 5",
    "Item 6",
    "Item 7",
    "Item 8",
  ];
  public onDragOver(event: DragOverEvent): void {
    event.preventDefault();
    this.sortable.moveItem(event.oldIndex, event.index);
  }
  public onNavigate(event: NavigateEvent): void {
    this.sortable.moveItem(event.oldIndex, event.index);
  }
}

```

[Collapse Code ^](#)

# Using the Built-In Directive

The Sortable provides the built-in `SortableBindingDirective` directive, which simplifies the handling of data operations by cutting down the repetitive boilerplate code.

The `SortableBindingDirective` will seamlessly keep the Sortable items and the data array that is passed to it in sync.

The following example demonstrates how to set up the binding.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_SORTABLE } from "@progress/kendo-angular-sortable";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_SORTABLE],
  template: `
    <div class="example-config">
      <h6>Items: {{ items | json }}</h6>
    </div>
    <div class="container-fluid">
      <kendo-sortable
        [kendoSortableBinding]="items"
        [navigable]="true"
        [animation]="true"
        class="row"
        itemClass="item col-xs-6 col-sm-3"
        activeItemClass="item col-xs-6 col-sm-3 active"
      >
      </kendo-sortable>
    </div>
  `,
  encapsulation: ViewEncapsulation.None,
  styleUrls: ["./styles.css"],
})
export class AppComponent {
  public items: string[] = [
    "Item 1",
    "Item 2",
    "Item 3",
    "Item 4",
  ]
}
```

```
    "Item 5",
    "Item 6",
    "Item 7",
    "Item 8",
];
}
```

Collapse Code ^

# Suggested Links

- [API Reference of the SortableBindingDirective](#)

# Getting Started with the Kendo UI for Angular Sortable

This guide provides the information you need to start using the Kendo UI for Angular Sortable—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#)

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { KENDO_SORTABLE } from "@progress/kendo-angular-sortable";

@Component({
  standalone: true,
  selector: "my-app",
  imports: [KENDO_SORTABLE],
  template: `
```

```
<kendo-sortable
  [kendoSortableBinding]="items"
  [animation]="true"
  itemClass="item"
  activeItemClass="item active"
>
</kendo-sortable>
,
encapsulation: ViewEncapsulation.None,
styleUrls: ['./styles.css'],
})
export class AppComponent {
  public items: string[] = ["Item 1", "Item 2", "Item 3"];
}
```

Collapse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Component

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Sortable package:

1. Run the following command:

```
ng add @progress/kendo-angular-sortable
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-sortable` package as a dependency to the `package.json` file.
- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

2. Import the `KENDO_SORTABLE` utility array in your standalone component to enable the entire feature set of the Sortable:

The utility array is available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

```
import { Component } from '@angular/core';
import { KENDO_SORTABLE } from '@progress/kendo-angular-sortable';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_SORTABLE]
})
```

TS

## Using the Component

1. After successfully installing the Sortable package and importing its module, add the following code in the `app.component.html` file:

```
<kendo-sortable
  [kendoSortableBinding]="items"
  [animation]="true"
  itemClass="item"
  activeItemClass="item active"
```

HTML

```
>  
</kendo-sortable>
```

- Bind the `kendoSortableBinding` directive to an array of strings by adding the following code in the `app.component.ts` file:

```
public items: string[] = [  
    "Item 1", "Item 2", "Item 3", "Item 4",  
    "Item 5", "Item 6", "Item 7", "Item 8"  
];
```

TS

- Style the Sortable items by setting the `itemClass` and `activeItemClass` properties to the respective CSS classes. For example add the following styles:

```
.item {  
    background-color: #bfe7f9;  
    color: #1494d0;  
    border: 1px solid #fff;  
    width: 200px;  
    line-height: 68px;  
    font-size: 16px;  
    text-align: center;  
    cursor: move;  
}  
  
.item.active {  
    background-color: #27aceb;  
    color: #fff;  
    border-color: #27aceb;  
    z-index: 10;  
    width: 200px;  
}
```

CSS

- Build and serve the application by running the following command in the root folder.

ng serve

SH

5. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular Sortable component on the page.

## Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

## Next Steps

- [Sortable Dependencies & Standalone Utilities](#)
- [Binding the Sortable to data](#)
- [Configuring Sortable items](#)
- [Customizing the Sortable](#)
- [API Reference of the Sortable](#)

## Learning Resources

- [Sortable Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Items

The Sortable allows you to [disable its items and to move items between two or more Sortable components](#).

## Disabling Items

To disable items, use the [disabledIndexes](#) property of the Sortable.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_SORTABLE } from "@progress/kendo-angular-sortable";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_SORTABLE],
  template: `
    <div class="example-config">
      <h5>Items: {{ items | json }}</h5>
      <h5>Disabled items: {{ disabledIndexes }}</h5>
    </div>
    <div class="container-fluid">
      <kendo-sortable
        [kendoSortableBinding]="items"
        [navigable]="true"
        [animation]="true"
        [disabledIndexes]="disabledIndexes"
        class="row"
        itemClass="item col-xs-6 col-sm-3"
        activeItemClass="item col-xs-6 col-sm-3 active"
        disabledItemClass="item col-xs-6 col-sm-3 disabled"
      >
        >
      </kendo-sortable>
    </div>
  </div>
)
```

```
</div>
  ,
  styleUrls: ['./styles.css'],
  encapsulation: ViewEncapsulation.None,
})
export class AppComponent {
  public disabledIndexes = [0, 2, 5, 7];
  public items: string[] = [];

  constructor() {
    for (let i = 0; i < 8; i++) {
      this.items[i] = "Item " + i;
    }
  }
}
```

Collanse Code ^

# Transferring of Items

To enable the user to move and transfer items between two or more Sortable components by dragging and dropping, utilize the drag zones of the components and set the `zone` and `acceptZones` properties.

When an item is dragged from the source to the target component, hide the currently active item in the source Sortable through the `visible:false` and `display:none` inline styles. While the dragging of the item happens, the item is removed from the source component on `dragEnd` (mouse up) and added to the target component on `dragOver` (mouse release). As a result, between the `dragEnd` and the `dragOver` actions, and at the same time, in both the source and the target Sortable components two identical items exist. This functionality is handled by the `SortableBindingDirective` directive. You can implement custom solutions by utilizing the `SortableService` and the events and methods of the Sortable.

The following example demonstrates how to enable the dragging of items between different item lists.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_SORTABLE } from "@progress/kendo-angular-sortable";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_SORTABLE],
  template:
    <div class="example-config">
      <h5>Team A: {{ palettes.TeamA | json }}</h5>
      <h5>Team B: {{ palettes.TeamB | json }}</h5>
    </div>
    <div class="container-fluid">
      <div class="row">
        <div class="col-xs-12 col-sm-6 team">
          <h5>Team A</h5>
          <kendo-sortable
            [kendoSortableBinding]="palettes.TeamA"
            zone="twoWay"
            emptyText="Move employees from Team B to Team A."
            class="row"
            itemClass="employee"
            [emptyItemStyle]="{{ 'min-height': '150px', width: '100%' }}"
            emptyItemClass="empty"
            activeItemClass="employee active"
          >
            </kendo-sortable>
          </div>
          <div class="col-xs-12 col-sm-6 team team-b">
            <h5>Team B</h5>
            <kendo-sortable
              [kendoSortableBinding]="palettes.TeamB"
              zone="twoWay"
              emptyText="Move employees from Team A to Team B."
              class="row"
              itemClass="employee"
              [emptyItemStyle]="{{ 'min-height': '150px', width: '100%' }}"
              emptyItemClass="empty"
              activeItemClass="employee active"
            >
              </kendo-sortable>
            </div>
          </div>
        </div>
      <, styleUrls: ['./styles.css'], encapsulation: ViewEncapsulation.None,
    })
export class AppComponent {
```

```
public palettes = {  
    TeamA: ["Peter Franken", "Simon Crowther", "Catherine Dewey"],  
    TeamB: ["Lino Rodriguez", "Paolo Accorti"],  
};
```

Collapse Code ^

## Reordering of Items

To enable the reordering of items, mark some of the elements in the item template as draggable—for example, `button` and `a`.

```
import { Component } from '@angular/core';  
  
@Component({  
    selector: 'my-app',  
    template: `  
        <kendo-sortable [kendoSortableBinding]="items">  
            <ng-template let-item="item">  
                <button draggable="true">  
                    {{item}}  
                </button>  
            </ng-template>  
        </kendo-sortable>  
    `,  
})  
export class AppComponent {  
    public items: string[] = [  
        'Item 1', 'Item 2', 'Item 3', 'Item 4'  
    ];  
}
```

TS

## Suggested Links

- [API Reference of the Sortable](#)

# Angular Sortable Overview

The Kendo UI for Angular Sortable is a component for dragging and dropping an element within a list to a new location within that list while the rest of the items correspondingly reorder.

The provided functionality is suitable for users who want to sort various elements such as, for example, photos within a gallery or items within a menu.

The Sortable is built from the ground up and specifically for Angular, so that you get a high-performance control which integrates tightly with your application and with the rest of the Kendo UI for Angular components.

## Angular Sortable Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_SORTABLE } from "@progress/kendo-angular-sortable";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_SORTABLE],
  template: `
    <div class="example-config">
      <h6>Items: {{ items | json }}</h6>
    </div>
    <div class="container-fluid">
      <kendo-sortable
        [kendoSortableBinding]="items"
        [navigable]="true"
        [animation]="true"
        class="row">
      </kendo-sortable>
    </div>
  `})
export class AppComponent {
  items = [
    { id: 1, name: "Item 1" },
    { id: 2, name: "Item 2" },
    { id: 3, name: "Item 3" },
    { id: 4, name: "Item 4" },
    { id: 5, name: "Item 5" }
  ];
}
```

```
        itemClass="item col-xs-6 col-sm-3"
        activeItemClass="item col-xs-6 col-sm-3 active"
      >
    </kendo-sortable>
  </div>
  ,
  encapsulation: ViewEncapsulation.None,
  styleUrls: ['./styles.css'],
})
export class AppComponent {
  public items: string[] = [
    "Item 1",
    "Item 2",
    "Item 3",
    "Item 4",
    "Item 5",
    "Item 6",
    "Item 7",
    "Item 8",
  ];
}
```

Collapse Code ^

# Angular Sortable Key Features

The Kendo UI for Angular Sortable component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Sortable library as well as develop new features to it.

## Data Binding

You can bind the Sortable to an array of strings or objects by setting the `data` property, or utilize the built-in directive, which cuts down the repetitive boilerplate code and simplifies the handling of data operations. [Read more about the data binding options of the Sortable...](#)

## Managing the Items

You can choose to render Sortable items in their disabled state so that, if need be present, users will not be able to interact with them. Also, you may have two or more Sortables and enable users to transfer items between all of the components. [Read more about the Sortable items...](#)

## Templates

You can visually customize the content, color, structure, and the general look and feel of the Sortable by implementing templates in the component. [Read more about the templates which are used by the Sortable...](#)

## Accessibility

The Sortable is accessible for screen readers and supports WAI-ARIA attributes. [Read more about the accessibility of the Sortable...](#)

## Globalization

The Kendo UI for Angular Sortable supports globalization to ensure that it can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Sortable supports rendering in a right-to-left (RTL) direction. [Read more about Sortable globalization...](#)

## Keyboard Navigation

The Sortable supports a number of keyboard shortcuts which allows users to accomplish various commands. [Read more about the keyboard navigation of the Sortable...](#)

# Trial Version of Kendo UI for Angular

The Kendo UI for Angular Sortable is part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no

restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of the Kendo UI for Angular Sortable, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Sortable Homepage](#)
- [Getting Started with the Kendo UI for Angular Sortable](#)
- [API Reference of the Sortable](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)

- [Sortable Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Templates

The Sortable enables you to customize its content by using templates.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_SORTABLE } from "@progress/kendo-angular-sortable";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_SORTABLE],
  encapsulation: ViewEncapsulation.None,
  template: `
    <ul>
      <li>Reorder the palettes</li>
      <li>Reorder the colors in palettes</li>
      <li>Move colors between palettes</li>
    </ul>
    <kendo-sortable
      [kendoSortableBinding]="palettes"
      [itemStyle]="{
        float: 'left',
        display: 'inline-block',
        width: '125px',
        'background-color': '#ffffad',
        margin: '4px',
        border: '1px solid black',
        cursor: 'move'
      }"
    >
      <ng-template let-palette="item">
        {{ palette.name }}
        <br /><br />
        <div style="width: 120px;">
          <kendo-sortable
            [kendoSortableBinding]="palette.data"
            [itemStyle]="{ border: '0px', opacity: '1', cursor: 'move' "
          >
        </div>
      </ng-template>
    </kendo-sortable>
  </div>
)
```

```
"}
      [emptyItemStyle] = { height: '30px', border: '2px dashed black' }
    [activeItemStyle] = { border: '2px dashed black', opacity: '0.7' }
      zone="innerZone"
    >
      <ng-template let-item="item">
        <div
          [ngStyle] = {
            'background-color': item,
            color: 'white',
            height: '30px',
            margin: '2px',
            border: '1px solid black'
          }
        >
          {{ item }}
        </div>
      </ng-template>
    </kendo-sortable>
  </div>
</ng-template>
</kendo-sortable>
``,
})
export class AppComponent {
  public colorsA: string[] = ["Violet", "Magenta", "Purple",
  "SlateBlue"];
  public colorsB: string[] = [
    "SteelBlue",
    "CornflowerBlue",
    "RoyalBlue",
    "MediumBlue",
  ];
  public colorsC: string[] = ["LimeGreen", "SeaGreen", "Green",
  "OliveDrab"];
  public colorsD: string[] = [
    "LightSalmon",
    "Salmon",
    "IndianRed",
    "FireBrick",
  ];
  public palettes = [
    { data: this.colorsA, name: "Palette A" },
    { data: this.colorsB, name: "Palette B" },
    { data: this.colorsC, name: "Palette C" },
    { data: this.colorsD, name: "Palette D" },
  ];
}
```

[Collapse Code ^](#)

# Suggested Links

- [API Reference of the Sortable](#)

# Control Types

You can add specific tools and controls to the ToolBar container.

## Button Controls

The ToolBar supports the following button-type controls:

- [Buttons](#)
- [Toggle buttons](#)
- [Button groups](#)
- [Split buttons](#)
- [Drop-down buttons](#)
- [Separators](#)
- [Spacer](#)

## Buttons

To render a button in the ToolBar container, add the `kendo-toolbar-button` tag during initialization. For more information, refer to the [ToolBarButton API](#).

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";

import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";
import { SVGIcon, filterIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
```

```
imports: [KENDO_TOOLBAR],
template: `
  <kendo-toolbar>
    <kendo-toolbar-button
      text="Button"
      showText="both"
      [svgIcon]="filterIcon"
      showIcon="both"
      themeColor="primary"
      [disabled]="false"
      (click)="onClick()"
    >
    </kendo-toolbar-button>
  </kendo-toolbar>
` ,
})
export class AppComponent {
  public filterIcon: SVGIcon = filterIcon;

  public onClick(): void {
    console.log("on click");
  }
}
```

Collapse Code ^

## Toggle Buttons

To render a toggle button in the ToolBar container, add the `kendo-toolbar-button` tag during initialization and set `toggleable` to `true`. For more information, refer to the [ToolBarButton API](#).

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";

import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_TOOLBAR],
```

```

template: `
  <kendo-toolbar>
    <kendo-toolbar-button
      text="Toggle"
      showText="both"
      showIcon="both"
      themeColor="primary"
      [disabled]="false"
      [toggleable]="true"
      [selected]="true"
      (selectedChange)="onChange()"
    >
    </kendo-toolbar-button>
  </kendo-toolbar>
`,

})
export class AppComponent {
  public onChange(): void {
    console.log("on change");
  }
}

```

[Collapse Code ▾](#)

## Button Groups

A ButtonGroup consists of multiple button elements that are visually separated in a group. In the overflow popup of the command, the ButtonGroup is shown as a list of commands.

The ButtonGroup supports **single** and **multiple** selection modes. The **single** selection mode acts as a radio group and allows only one button to be selected at a time. When a user clicks on a button, it becomes selected and all other buttons that are part of the group become unselected. The **multiple** selection mode allows more than one button to be selected at a time. If the user clicks a button that has been already selected, it will become unselected. By default the selection mode is set to **multiple**.

To render a ButtonGroup in the ToolBar container, add the **kendo-toolbar-buttongroup** tag during initialization. You can separately configure each Button in the ButtonGroup by using the **kendo-toolbar-button** tag.

- [ToolBarButtonGroup API](#)

- [ToolBarButton API](#)

 [EXAMPLE](#) [VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";
import {
    SVGIcon,
    alignCenterIcon,
    alignJustifyIcon,
    alignLeftIcon,
    alignRightIcon,
    boldIcon,
    italicIcon,
    underlineIcon,
} from "@progress/kendo-svg-icons";

@Component({
    selector: "my-app",
    standalone: true,
    imports: [KENDO_TOOLBAR],
    template: `
        <kendo-toolbar>
            <kendo-toolbar-buttongroup selection="multiple">
                <kendo-toolbar-button
                    [toggleable]="true"
                    [svgIcon]="alignLeftIcon"
                ></kendo-toolbar-button>
                <kendo-toolbar-button
                    [toggleable]="true"
                    [svgIcon]="alignCenterIcon"
                ></kendo-toolbar-button>
                <kendo-toolbar-button
                    [toggleable]="true"
                    [svgIcon]="alignRightIcon"
                ></kendo-toolbar-button>
                <kendo-toolbar-button
                    [toggleable]="true"
                    [svgIcon]="alignJustifyIcon"
                ></kendo-toolbar-button>
            </kendo-toolbar-buttongroup>
            <kendo-toolbar-separator></kendo-toolbar-separator>
            <kendo-toolbar-buttongroup selection="single">
                <kendo-toolbar-button
                    [toggleable]="true"

```

```

        [svgIcon]="boldIcon"
      ></kendo-toolbar-button>
      <kendo-toolbar-button
        [toggleable]="true"
        [svgIcon]="italicIcon"
      ></kendo-toolbar-button>
      <kendo-toolbar-button
        [toggleable]="true"
        [svgIcon]="underlineIcon"
      ></kendo-toolbar-button>
    </kendo-toolbar-buttongroup>
  </kendo-toolbar>
  ,
})
export class AppComponent {
  public alignLeftIcon: SVGIcon = alignLeftIcon;
  public alignRightIcon: SVGIcon = alignRightIcon;
  public alignCenterIcon: SVGIcon = alignCenterIcon;
  public alignJustifyIcon: SVGIcon = alignJustifyIcon;
  public boldIcon: SVGIcon = boldIcon;
  public italicIcon: SVGIcon = italicIcon;
  public underlineIcon: SVGIcon = underlineIcon;
}

```

[Collapse Code ^](#)

## Split Buttons

A SplitButton allows the user to execute a default action, which is bound to a button or to choose a predefined action from a drop-down list. To render a SplitButton in the ToolBar container, add the **kendo-toolbar-splitbutton** tag during initialization. For more information, refer to the [ToolBarSplitButton API](#).

 EXAMPLE

 VIEW SOURCE

⋮



```

import { Component } from "@angular/core";

import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";
import {
  SVGIcon,
  clipboardCodeIcon,
  clipboardMarkdownIcon,

```

```
clipboardTextIcon,
clipboardIcon,
} from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_TOOLBAR],
  template: `
    <kendo-toolbar>
      <kendo-toolbar-splitbutton
        text="Paste"
        [data]="data"
        [svgIcon]="clipboardSVG"
        (buttonClick)="onPaste()"
      >
      </kendo-toolbar-splitbutton>
    </kendo-toolbar>
  `,
})
export class AppComponent {
  public clipboardSVG: SVGIcon = clipboardIcon;
  public data = [
    {
      text: "Keep Text Only",
      svgIcon: clipboardTextIcon,
      click: (): void => {
        console.log("Keep Text Only");
      },
    },
    {
      text: "Paste as HTML",
      svgIcon: clipboardCodeIcon,
      click: (): void => {
        console.log("Paste as HTML");
      },
    },
    {
      text: "Paste Markdown",
      svgIcon: clipboardMarkdownIcon,
      click: (): void => {
        console.log("Paste Markdown");
      },
    },
    {
      text: "Set Default Paste",
      click: (): void => {
        console.log("Set Default Paste");
      },
    },
  ];
}
```

```
public onPaste(): void {
    console.log("Paste");
}
}
```

Collanse Code ^

## Drop-Down Buttons

When the DropDownButton is clicked, it displays a popup list with action items. To render a DropDownButton in the ToolBar container, add the **kendo-toolbar-dropdownbutton** tag during initialization. For more information, refer to the [ToolBarDropDownButton API](#).

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";

import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";
import {
    clipboardCodeIcon,
    clipboardIcon,
    clipboardMarkdownIcon,
    clipboardTextIcon,
} from "@progress/kendo-svg-icons";

@Component({
    selector: "my-app",
    standalone: true,
    imports: [KENDO_TOOLBAR],
    template: `
        <kendo-toolbar>
            <kendo-toolbar-dropdownbutton text="Paste Variations"
[data]="data">
            </kendo-toolbar-dropdownbutton>
        </kendo-toolbar>
    `,
})
export class AppComponent {
    public data = [
        {
            text: "Paste",
        }
    ]
}
```

```
        svgIcon: clipboardIcon,
        click: () => {
          console.log("Paste");
        },
      },
      {
        text: "Keep Text Only",
        svgIcon: clipboardTextIcon,
        click: () => {
          console.log("Keep Text Only");
        },
      },
      {
        text: "Paste as HTML",
        svgIcon: clipboardCodeIcon,
        click: () => {
          console.log("Paste as HTML");
        },
      },
      {
        text: "Paste Markdown",
        svgIcon: clipboardMarkdownIcon,
        click: () => {
          console.log("Paste Markdown");
        },
      },
      {
        text: "Set Default Paste",
        click: () => {
          console.log("Set Default Paste");
        },
      },
    ],
  }

  public onPaste(): void {
    console.log("Paste");
  }
}
```

Collapse Code ^

# Separators

You can configure separators to act as delimiters between the ToolBar tools.

 EXAMPLE

 VIEW SOURCE





```
import { Component } from "@angular/core";
import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_TOOLBAR],
  template:
    <kendo-toolbar>
      <kendo-toolbar-button text="Button 1"></kendo-toolbar-button>
      <kendo-toolbar-separator></kendo-toolbar-separator>
      <kendo-toolbar-button text="Button 2"></kendo-toolbar-button>
    </kendo-toolbar>
  ,
})
export class AppComponent {}
```

Collanse Code ^

# Spacer

You can add a spacers to take up all available space between ToolBar tools.

EXAMPLE

VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_TOOLBAR],
  template:
    <kendo-toolbar>
      <kendo-toolbar-button text="Button 1"></kendo-toolbar-button>
      <kendo-toolbar-spacer></kendo-toolbar-spacer>
```

```
<kendo-toolbar>
  <kendo-toolbar-button text="Button 2"></kendo-toolbar-button>
</kendo-toolbar>
}
export class AppComponent {}
```

Collanse Code ^

# Suggested Links

- [Implementing Custom Control Types](#)
- [API Reference of the ToolBar](#)
- [API Reference of the ToolBarButton](#)
- [API Reference of the ToolBarButtonGroup](#)
- [API Reference of the ToolBarSplitButton](#)
- [API Reference of the ToolBarDropDownButton](#)

# Getting Started with the Kendo UI for Angular ToolBar

This guide provides the information you need to start using the Kendo UI for Angular ToolBar—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#)

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";

@Component({
  standalone: true,
  imports: [KENDO_TOOLBAR],
  selector: "my-app",
  template: `
    <kendo-toolbar>
      <kendo-toolbar-button text="Button"></kendo-toolbar-button>
      <kendo-toolbar-button>
```

```
        text="Toggle"
        [toggleable]="true"
      ></kendo-toolbar-button>
    </kendo-toolbar>
  ,
})
export class AppComponent {}
```

Collanse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Component

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular ToolBar package:

1. Run the following command:

```
ng add @progress/kendo-angular-toolbar
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-toolbar` package as a dependency to the `package.json` file.

- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

**2. Import the `KENDO_TOOLBAR` utility array in your standalone component to enable the entire feature set of the ToolBar:**

The utility array is available starting from **v16.6.0**. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

```
import { Component } from '@angular/core';
import { KENDO_TOOLBAR } from '@progress/kendo-angular-toolbar';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_TOOLBAR]
})
```

TS

## Using the Component

**1. After successfully installing the ToolBar package and importing its component, add the following code in the `app.component.html` file:**

```
<kendo-toolbar>
  <kendo-toolbar-button text="Button"></kendo-toolbar-button>
  <!-- . . . -->
</kendo-toolbar>
```

HTML

**2. Build and serve the application by running the following command in the root folder.**

ng serve

SH

3. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular ToolBar component on the page.

## Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

## Next Steps

- [ToolBar Dependencies & Standalone Utilities](#)
- [API Reference of the ToolBar](#)

## Learning Resources

- [ToolBar Overview](#)
- [Angular ToolBar Homepage](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Kendo UI for Angular ToolBar Overview

The Kendo UI for Angular ToolBar provides a graphical container for holding button elements and allows users to effortlessly structure and access them.

The ToolBar is built from the ground up and specifically for Angular, so that you get a high-performance control which integrates tightly with your application and with the rest of the Kendo UI for Angular components.

The following example demonstrates the ToolBar in action.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";
import {
  SVGIcon,
  alignCenterIcon,
  alignJustifyIcon,
  alignLeftIcon,
  alignRightIcon,
} from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_TOOLBAR],
  template:
    <kendo-toolbar>
      <kendo-toolbar-button text="Button"></kendo-toolbar-button>
      <kendo-toolbar-button
        text="Toggle"
        [toggleable]="true"
      ></kendo-toolbar-button>
      <kendo-toolbar-splitbutton text="Split Button">
```

```
[data]="splitButtonData">
  </kendo-toolbar-splitbutton>
  <kendo-toolbar-dropdownbutton
    text="DropDownButton"
    [data]="dropdownButtonData"
  >
  </kendo-toolbar-dropdownbutton>
<kendo-toolbar-buttongroup selection="single">
  <kendo-toolbar-button
    [toggleable]="true"
    title="Align left"
    [svgIcon]="alignLeftIcon"
  ></kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    title="Align center"
    [svgIcon]="alignCenterIcon"
  ></kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    title="Align right"
    [svgIcon]="alignRightIcon"
  ></kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    title="Align justify"
    [svgIcon]="alignJustifyIcon"
  ></kendo-toolbar-button>
</kendo-toolbar-buttongroup>
</kendo-toolbar>
``,
})
export class AppComponent {
  public alignLeftIcon: SVGIcon = alignLeftIcon;
  public alignRightIcon: SVGIcon = alignRightIcon;
  public alignCenterIcon: SVGIcon = alignCenterIcon;
  public alignJustifyIcon: SVGIcon = alignJustifyIcon;

  public splitButtonData: Array<{ text: string }> = [
    {
      text: "Option 1",
    },
    {
      text: "Option 2",
    },
    {
      text: "Option 3",
    },
  ];
  public dropdownButtonData: Array<{ text: string }> = [
```

```
{  
    text: "Option 1",  
},  
{  
    text: "Option 2",  
},  
{  
    text: "Option 3",  
},  
];  
}  
]  
};
```

Collance Code ^

# Angular ToolBar Key Features

The Kendo UI for Angular ToolBar service delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing ToolBar library as well as develop new features to it.

## Control Types

The ToolBar features built-in control types, which enable you to display specific tools and controls to the ToolBar container. [Read more about the built-in ToolBar control types...](#)

## Custom Control Types

The ToolBar allows you to create custom tools that can be added to the ToolBar and Overflow Popup. [Read more about the custom control types of the ToolBar...](#)

## Responsive ToolBar

The ToolBar provides options to hide the tools that do not fit its width in an overflow popup. [Read more about the responsive behavior of the ToolBar...](#)

## Appearance

The ToolBar provides built-in options that allow you to customize its appearance. [Read more about customizing the appearance of the ToolBar...](#)

## Globalization

The Kendo UI for Angular ToolBar supports globalization to ensure that it can fit well in any application, no matter what languages and locales need to be supported. Additionally, the ToolBar supports rendering in a right-to-left (RTL) direction. [Read more about the globalization of the ToolBar...](#)

## Accessibility

The ToolBar is accessible for screen readers and supports WAI-ARIA attributes. [Read more about the accessibility of the ToolBar...](#)

## Keyboard Navigation

The ToolBar supports a number of keyboard shortcuts which allow users to accomplish various commands. [Read more about the keyboard navigation of the ToolBar...](#)

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular ToolBar is part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of the Kendo UI for Angular ToolBar, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [ToolBar Homepage](#)
- [Getting Started with the Kendo UI for Angular ToolBar](#)
- [API Reference of the ToolBar](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [ToolBar Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Responsive ToolBar

The ToolBar enables you to hide the tools that do not fit its width and allows you to programmatically determine their rendering.

By default, the ToolBar allows you to choose from three different overflow modes for smaller screen sizes:

- [Scroll Mode](#)
- [Section Mode](#)
- [Menu Mode](#)

## Scroll Mode

Setting the `overflow` property of the ToolBar to `scroll` allows you to display all tools regardless of the component's width. In this mode, the ToolBar tools become scrollable and the component displays two scroll buttons for easier navigation.

The following example demonstrates how to make the ToolBar tools scrollable.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import {
  SVGIcon,
  alignCenterIcon,
  alignJustifyIcon,
  alignLeftIcon,
  alignRightIcon,
  boldIcon,
  clipboardCodeIcon,
  clipboardIcon,
  clipboardMarkdownIcon,
  clipboardTextIcon,
```

```
gearIcon,  
italicIcon,  
redoIcon,  
underlineIcon,  
undoIcon,  
} from "@progress/kendo-svg-icons";  
  
import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";  
  
import { FormsModule } from "@angular/forms";  
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";  
import { KENDO_LABELS } from "@progress/kendo-angular-label";  
  
@Component({  
  selector: "my-app",  
  standalone: true,  
  imports: [FormsModule, KENDO_TOOLBAR, KENDO_INPUTS, KENDO_LABELS],  
  template: `  
    <kendo-label [for]="width" text="Set toolbar width"></kendo-label>  
    <kendo-slider  
      #width  
      [(ngModel)]="toolbarWidth"  
      style="width: 100%; display: block;"  
      [showButtons]="false"  
      [min]="'0'"  
      [max]="'100'"  
      [largeStep]="'1'"  
      tickPlacement="none"  
    ></kendo-slider>  
  
    <kendo-toolbar overflow="scroll" [style.width.%]="'toolbarWidth'">  
      <kendo-toolbar-buttongroup>  
        <kendo-toolbar-button  
          [toggleable]="'true'"  
          [svgIcon]="'boldIcon'"  
          (click)="onItemClick('Bold button clicked')"  
        >  
        </kendo-toolbar-button>  
        <kendo-toolbar-button  
          [toggleable]="'true'"  
          [svgIcon]="'italicIcon'"  
          (click)="onItemClick('Italic button clicked')"  
        >  
        </kendo-toolbar-button>  
        <kendo-toolbar-button  
          [toggleable]="'true'"  
          [svgIcon]="'underlineIcon'"  
          (click)="onItemClick('Underline button clicked')"  
        >  
        </kendo-toolbar-button>  
      </kendo-toolbar-buttongroup>  
    </kendo-toolbar>  
  `
```

```
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-butongroup selection="single">
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignLeftIcon"
    text="Left"
    (click)="onItemClick('Align left button clicked')"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignCenterIcon"
    text="Center"
    (click)="onItemClick('Align center button clicked')"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignRightIcon"
    text="Right"
    (click)="onItemClick('Align right button clicked')"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignJustifyIcon"
    text="Justify"
    (click)="onItemClick('Align justify button clicked')"
  >
  </kendo-toolbar-button>
</kendo-toolbar-butongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-splitbutton
  [svgIcon]="clipboardIcon"
  text="Paste"
  [data]="splitButtonData"
  (buttonClick)="onItemClick('Paste splitbutton clicked')"
>
</kendo-toolbar-splitbutton>
<kendo-toolbar-dropdownbutton
  [svgIcon]="gearIcon"
  text="Set default paste"
  [data]="dropdownButtonData"
>
</kendo-toolbar-dropdownbutton>
</kendo-toolbar>

  <p>{{ clickMessage }}</p>
``,
})
export class AppComponent {
```

```
public clickMessage: string;
public toolbarWidth = 60;
public undoIcon: SVGIcon = undoIcon;
public redoIcon: SVGIcon = redoIcon;
public boldIcon: SVGIcon = boldIcon;
public italicIcon: SVGIcon = italicIcon;
public underlineIcon: SVGIcon = underlineIcon;
public alignLeftIcon: SVGIcon = alignLeftIcon;
public alignCenterIcon: SVGIcon = alignCenterIcon;
public alignRightIcon: SVGIcon = alignRightIcon;
public alignJustifyIcon: SVGIcon = alignJustifyIcon;
public clipboardIcon: SVGIcon = clipboardIcon;
public gearIcon: SVGIcon = gearIcon;

public splitButtonData: Array<{ [key: string]: unknown }> = [
  {
    text: "Keep Text Only",
    svgIcon: clipboardTextIcon,
    click: (): void =>
      this.onItemClick("Keep Text Only splitbutton button clicked"),
  },
  {
    text: "Paste as HTML",
    svgIcon: clipboardCodeIcon,
    click: (): void =>
      this.onItemClick("Paste as HTML splitbutton button clicked"),
  },
  {
    text: "Paste Markdown",
    svgIcon: clipboardMarkdownIcon,
    click: (): void =>
      this.onItemClick("Paste Markdown splitbutton button clicked"),
  },
];

public dropdownButtonData: Array<{ [key: string]: unknown }> = [
  {
    text: "Keep Text Only",
    svgIcon: clipboardTextIcon,
    click: (): void =>
      this.onItemClick("Keep Text Only dropdownbutton button clicked"),
  },
  {
    text: "Paste as HTML",
    svgIcon: clipboardCodeIcon,
    click: (): void =>
      this.onItemClick("Paste as HTML dropdownbutton button clicked"),
  },
  {
    text: "Paste Markdown",
```

```

    svgIcon: clipboardMarkdownIcon,
    click: () =>
      this.onItemClick("Paste Markdown dropdownbutton button
clicked"),
  ],
};

public onItemClick(message: string): void {
  console.log(message);
  this.clickMessage = message;
}
}

```

[Collapse Code ^](#)

## Controlling Scroll Buttons Visibility

When the ToolBar is in scroll mode, you can control the visibility of the scroll buttons. To determine whether the buttons should always be visible, automatically displayed, or hidden based on your requirements, use the built-in `scrollButtons` property which accepts the following values:

- `auto` (default)—The scroll buttons will be rendered only when the tools list overflows its container.
- `visible`—The scroll buttons will be always visible.
- `hidden`—No scroll buttons will be rendered.

The following example demonstrates how to dynamically update the value of the `scrollButtons` property.

EXAMPLE

VIEW SOURCE

⋮

```

import { Component } from "@angular/core";
import {
  SVGIcon,
  alignCenterIcon,
  alignJustifyIcon,
  alignLeftIcon,
  alignRightIcon,
  boldIcon,
  clipboardCodeIcon,

```

```
clipboardIcon,
clipboardMarkdownIcon,
clipboardTextIcon,
gearIcon,
italicIcon,
redoIcon,
underlineIcon,
undoIcon,
} from "@progress/kendo-svg-icons";

import {
KENDO_TOOLBAR,
ToolbarScrollButtonsVisibility,
} from "@progress/kendo-angular-toolbar";

import { FormsModule } from "@angular/forms";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_LABELS } from "@progress/kendo-angular-label";

@Component({
selector: "my-app",
standalone: true,
imports: [FormsModule, KENDO_TOOLBAR, KENDO_INPUTS, KENDO_LABELS],
styles: [
```
.k-checkbox-label {
margin-right: 10px;
margin-left: 2px;
}
```,
],
template: `
<div class="example-config">
<p>Scroll buttons visibility:</p>
<input
id="auto"
name="pos"
kendoRadioButton
type="radio"
[(ngModel)]="scrollButtonsVisibility"
value="auto"
/>
<kendo-label
for="auto"
class="k-checkbox-label"
text="Auto"
></kendo-label>
<input
id="visible"
name="pos"
kendoRadioButton

```

```
        type="radio"
        [(ngModel)]="scrollButtonsVisibility"
        value="visible"
    />
    <kendo-label
        for="visible"
        class="k-checkbox-label"
        text="Visible"
    ></kendo-label>
    <input
        id="hidden"
        name="pos"
        kendoRadioButton
        type="radio"
        [(ngModel)]="scrollButtonsVisibility"
        value="hidden"
    />
    <kendo-label
        for="hidden"
        class="k-checkbox-label"
        text="Hidden"
    ></kendo-label>
</div>

<kendo-toolbar
    [overflow]={`${ mode: 'scroll', scrollButtons:
scrollButtonsVisibility }`}
    [style.width.%]="${toolbarWidth}"
>
    <kendo-toolbar-butongroup>
        <kendo-toolbar-button
            [toggleable]="true"
            [svgIcon]="boldIcon"
            (click)="onItemClick('Bold button clicked')"
        >
        </kendo-toolbar-button>
        <kendo-toolbar-button
            [toggleable]="true"
            [svgIcon]="italicIcon"
            (click)="onItemClick('Italic button clicked')"
        >
        </kendo-toolbar-button>
        <kendo-toolbar-button
            [toggleable]="true"
            [svgIcon]="underlineIcon"
            (click)="onItemClick('Underline button clicked')"
        >
        </kendo-toolbar-button>
    </kendo-toolbar-butongroup>
    <kendo-toolbar-separator></kendo-toolbar-separator>
    <kendo-toolbar-butongroup selection="single">
```

```
<kendo-toolbar-button  
  [toggleable]="true"  
  [svgIcon]="alignLeftIcon"  
  text="Left"  
  (click)="onItemClick('Align left button clicked')"  
>  
</kendo-toolbar-button>  
<kendo-toolbar-button  
  [toggleable]="true"  
  [svgIcon]="alignCenterIcon"  
  text="Center"  
  (click)="onItemClick('Align center button clicked')"  
>  
</kendo-toolbar-button>  
<kendo-toolbar-button  
  [toggleable]="true"  
  [svgIcon]="alignRightIcon"  
  text="Right"  
  (click)="onItemClick('Align right button clicked')"  
>  
</kendo-toolbar-button>  
<kendo-toolbar-button  
  [toggleable]="true"  
  [svgIcon]="alignJustifyIcon"  
  text="Justify"  
  (click)="onItemClick('Align justify button clicked')"  
>  
</kendo-toolbar-button>  
</kendo-toolbar-buttongroup>  
<kendo-toolbar-separator></kendo-toolbar-separator>  
<kendo-toolbar-splitbutton  
  [svgIcon]="clipboardIcon"  
  text="Paste"  
  [data]="splitButtonData"  
  (buttonClick)="onItemClick('Paste splitbutton clicked')"  
>  
</kendo-toolbar-splitbutton>  
<kendo-toolbar-dropdownbutton  
  [svgIcon]="gearIcon"  
  text="Set default paste"  
  [data]="dropdownButtonData"  
>  
</kendo-toolbar-dropdownbutton>  
</kendo-toolbar>  
  
<p>{{ clickMessage }}</p>  
,
```

)

```
export class AppComponent {  
  public scrollButtonsVisibility: ToolbarScrollButtonsVisibility =  
  "auto";
```

```
public clickMessage: string;
public toolbarWidth = 60;
public undoIcon: SVGIcon = undoIcon;
public redoIcon: SVGIcon = redoIcon;
public boldIcon: SVGIcon = boldIcon;
public italicIcon: SVGIcon = italicIcon;
public underlineIcon: SVGIcon = underlineIcon;
public alignLeftIcon: SVGIcon = alignLeftIcon;
public alignCenterIcon: SVGIcon = alignCenterIcon;
public alignRightIcon: SVGIcon = alignRightIcon;
public alignJustifyIcon: SVGIcon = alignJustifyIcon;
public clipboardIcon: SVGIcon = clipboardIcon;
public gearIcon: SVGIcon = gearIcon;

public splitButtonData: Array<{ [key: string]: unknown }> = [
  {
    text: "Keep Text Only",
    svgIcon: clipboardTextIcon,
    click: (): void =>
      this.onItemClick("Keep Text Only splitbutton button clicked"),
  },
  {
    text: "Paste as HTML",
    svgIcon: clipboardCodeIcon,
    click: (): void =>
      this.onItemClick("Paste as HTML splitbutton button clicked"),
  },
  {
    text: "Paste Markdown",
    svgIcon: clipboardMarkdownIcon,
    click: (): void =>
      this.onItemClick("Paste Markdown splitbutton button clicked"),
  },
];

public dropdownButtonData: Array<{ [key: string]: unknown }> = [
  {
    text: "Keep Text Only",
    svgIcon: clipboardTextIcon,
    click: (): void =>
      this.onItemClick("Keep Text Only dropdownbutton button clicked"),
  },
  {
    text: "Paste as HTML",
    svgIcon: clipboardCodeIcon,
    click: (): void =>
      this.onItemClick("Paste as HTML dropdownbutton button clicked"),
  },
];
```

```
text: "Paste Markdown",
svgIcon: clipboardMarkdownIcon,
click: (): void =>
  this.onItemClick("Paste Markdown dropdownbutton button
clicked"),
},
];
public onItemClick(message: string): void {
  console.log(message);
  this.clickMessage = message;
}
```

Collapse Code ▾

## Adjusting Scroll Buttons Position

The scroll mode allows you to adjust the position of the scroll buttons in the ToolBar component. To render the buttons either at the beginning of the component, the end of the component, or at each side of the tools list, use the `scrollButtonsPosition` property and set its value to either of the following possible options:

- `split` (default)—The scroll buttons will be rendered at each side of the tools list.
- `start`—The scroll buttons will be rendered at the start before all tools.
- `end`—The scroll buttons will be rendered at the end after all tools.

The following example demonstrates how to use the `scrollButtonsPosition` property to adjust the scroll buttons' position.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import {
  SVGIcon,
  alignCenterIcon,
  alignJustifyIcon,
  alignLeftIcon,
  alignRightIcon,
  boldIcon,
  clipboardCodeIcon,
```

```
clipboardIcon,
clipboardMarkdownIcon,
clipboardTextIcon,
gearIcon,
italicIcon,
redoIcon,
underlineIcon,
undoIcon,
} from "@progress/kendo-svg-icons";

import {
KENDO_TOOLBAR,
ToolbarOverflowSettings,
ToolbarScrollButtonsPosition,
} from "@progress/kendo-angular-toolbar";

import { FormsModule } from "@angular/forms";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_LABELS } from "@progress/kendo-angular-label";

@Component({
selector: "my-app",
standalone: true,
imports: [FormsModule, KENDO_TOOLBAR, KENDO_INPUTS, KENDO_LABELS],
styles: [
.k-checkbox-label {
margin-right: 10px;
margin-left: 2px;
}
],
template: `
<div class="example-config">
<p>Scroll buttons position:</p>
<input
id="start"
name="pos"
kendoRadioButton
type="radio"
[(ngModel)]="scrollButtonsPosition"
value="start"
/>
<kendo-label
for="start"
class="k-checkbox-label"
text="Start"
></kendo-label>
<input
id="end"
name="pos"

```

```
kendoRadioButton
type="radio"
[(ngModel)]="scrollButtonsPosition"
value="end"
/>
<kendo-label for="end" class="k-checkbox-label" text="End">
</kendo-label>
<input
  id="split"
  name="pos"
  kendoRadioButton
  type="radio"
  [(ngModel)]="scrollButtonsPosition"
  value="split"
/>
<kendo-label
  for="split"
  class="k-checkbox-label"
  text="Split"
></kendo-label>
</div>

<kendo-toolbar
  [overflow]="{
    mode: 'scroll',
    scrollButtonsPosition: scrollButtonsPosition
  }"
  [style.width.%]="#toolbarWidth"
>
  <kendo-toolbar-butongroup>
    <kendo-toolbar-button
      [toggleable]="true"
      [svgIcon]="boldIcon"
      (click)="onItemClick('Bold button clicked')"
    >
    </kendo-toolbar-button>
    <kendo-toolbar-button
      [toggleable]="true"
      [svgIcon]="italicIcon"
      (click)="onItemClick('Italic button clicked')"
    >
    </kendo-toolbar-button>
    <kendo-toolbar-button
      [toggleable]="true"
      [svgIcon]="underlineIcon"
      (click)="onItemClick('Underline button clicked')"
    >
    </kendo-toolbar-button>
  </kendo-toolbar-butongroup>
  <kendo-toolbar-separator></kendo-toolbar-separator>
  <kendo-toolbar-butongroup selection="single">
```

```
<kendo-toolbar-button  
[toggleable]="true"  
[svgIcon]="alignLeftIcon"  
text="Left"  
(click)="onItemClick('Align left button clicked')"  
>  
</kendo-toolbar-button>  
<kendo-toolbar-button  
[toggleable]="true"  
[svgIcon]="alignCenterIcon"  
text="Center"  
(click)="onItemClick('Align center button clicked')"  
>  
</kendo-toolbar-button>  
<kendo-toolbar-button  
[toggleable]="true"  
[svgIcon]="alignRightIcon"  
text="Right"  
(click)="onItemClick('Align right button clicked')"  
>  
</kendo-toolbar-button>  
<kendo-toolbar-button  
[toggleable]="true"  
[svgIcon]="alignJustifyIcon"  
text="Justify"  
(click)="onItemClick('Align justify button clicked')"  
>  
</kendo-toolbar-button>  
</kendo-toolbar-buttongroup>  
<kendo-toolbar-separator></kendo-toolbar-separator>  
<kendo-toolbar-splitbutton  
[svgIcon]="clipboardIcon"  
text="Paste"  
[data]="splitButtonData"  
(buttonClick)="onItemClick('Paste splitbutton clicked')"  
>  
</kendo-toolbar-splitbutton>  
<kendo-toolbar-dropdownbutton  
[svgIcon]="gearIcon"  
text="Set default paste"  
[data]="dropdownButtonData"  
>  
</kendo-toolbar-dropdownbutton>  
</kendo-toolbar>  
  
<p>{{ clickMessage }}</p>  
,  
})  
export class AppComponent {  
  public scrollButtonsPosition: ToolbarScrollButtonsPosition = "start";
```

```
public clickMessage: string;
public toolbarWidth = 60;
public undoIcon: SVGIcon = undoIcon;
public redoIcon: SVGIcon = redoIcon;
public boldIcon: SVGIcon = boldIcon;
public italicIcon: SVGIcon = italicIcon;
public underlineIcon: SVGIcon = underlineIcon;
public alignLeftIcon: SVGIcon = alignLeftIcon;
public alignCenterIcon: SVGIcon = alignCenterIcon;
public alignRightIcon: SVGIcon = alignRightIcon;
public alignJustifyIcon: SVGIcon = alignJustifyIcon;
public clipboardIcon: SVGIcon = clipboardIcon;
public gearIcon: SVGIcon = gearIcon;

public splitButtonData: Array<{ [key: string]: unknown }> = [
  {
    text: "Keep Text Only",
    svgIcon: clipboardTextIcon,
    click: (): void =>
      this.onItemClick("Keep Text Only splitbutton button clicked"),
  },
  {
    text: "Paste as HTML",
    svgIcon: clipboardCodeIcon,
    click: (): void =>
      this.onItemClick("Paste as HTML splitbutton button clicked"),
  },
  {
    text: "Paste Markdown",
    svgIcon: clipboardMarkdownIcon,
    click: (): void =>
      this.onItemClick("Paste Markdown splitbutton button clicked"),
  },
];

public dropdownButtonData: Array<{ [key: string]: unknown }> = [
  {
    text: "Keep Text Only",
    svgIcon: clipboardTextIcon,
    click: (): void =>
      this.onItemClick("Keep Text Only dropdownbutton button clicked"),
  },
  {
    text: "Paste as HTML",
    svgIcon: clipboardCodeIcon,
    click: (): void =>
      this.onItemClick("Paste as HTML dropdownbutton button clicked"),
  },
  {
    text: "Paste Markdown",
```

```
    svgIcon: clipboardMarkdownIcon,
    click: () =>
      this.onItemClick("Paste Markdown dropdownbutton button
clicked"),
  },
];

public onItemClick(message: string): void {
  console.log(message);
  this.clickMessage = message;
}

```

Collapse Code ^

## Customizing Scroll Buttons Icons

You can modify the icons used for the scroll buttons in the ToolBar. To replace the default icons with the ones that better fit your application's requirement, utilize the `prevSVGButtonIcon` and `nextSVGButtonIcon` properties.

In case you have configured the application to [use font icons](#), use the `prevButtonIcon` and `nextButtonIcon` properties.

The following example demonstrates how to change the default scroll buttons icons.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import {
  SVGIcon,
  alignCenterIcon,
  alignJustifyIcon,
  alignLeftIcon,
  alignRightIcon,
  arrowLeftIcon,
  arrowRightIcon,
  boldIcon,
  clipboardCodeIcon,
  clipboardIcon,
  clipboardMarkdownIcon,
  clipboardTextIcon,
```

```
gearIcon,
italicIcon,
redoIcon,
underlineIcon,
undoIcon,
} from "@progress/kendo-svg-icons";

import {
KENDO_TOOLBAR,
ToolbarOverflowSettings,
} from "@progress/kendo-angular-toolbar";

import { FormsModule } from "@angular/forms";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_LABELS } from "@progress/kendo-angular-label";

@Component({
selector: "my-app",
standalone: true,
imports: [FormsModule, KENDO_TOOLBAR, KENDO_INPUTS, KENDO_LABELS],
template: `
<kendo-label [for]="width" text="Set toolbar width"></kendo-label>
<kendo-slider
#width
[(ngModel)]="toolbarWidth"
style="width: 100%; display: block;"
[showButtons]="false"
[min]="'0'"
[max]="'100'"
[largeStep]="'1'"
tickPlacement="none"
></kendo-slider>

<kendo-toolbar
[overflow]="'scrollModeSettings'"
[style.width.%]="'toolbarWidth'"
>
<kendo-toolbar-buttongroup>
<kendo-toolbar-button
[toggleable]="'true'"
[svgIcon]="'boldIcon'"
(click)="onItemClick('Bold button clicked')"
>
</kendo-toolbar-button>
<kendo-toolbar-button
[toggleable]="'true'"
[svgIcon]="'italicIcon'"
(click)="onItemClick('Italic button clicked')"
>
</kendo-toolbar-button>
<kendo-toolbar-button
```

```
[toggleable]="true"
[svgIcon]="underlineIcon"
(click)="onItemClick('Underline button clicked')"
>
</kendo-toolbar-button>
</kendo-toolbar-buttongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-buttongroup selection="single">
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignLeftIcon"
    text="Left"
    (click)="onItemClick('Align left button clicked')"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignCenterIcon"
    text="Center"
    (click)="onItemClick('Align center button clicked')"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignRightIcon"
    text="Right"
    (click)="onItemClick('Align right button clicked')"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignJustifyIcon"
    text="Justify"
    (click)="onItemClick('Align justify button clicked')"
  >
  </kendo-toolbar-button>
</kendo-toolbar-buttongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-splitbutton
  [svgIcon]="clipboardIcon"
  text="Paste"
  [data]="splitButtonData"
  (buttonClick)="onItemClick('Paste splitbutton clicked')"
>
</kendo-toolbar-splitbutton>
<kendo-toolbar-dropdownbutton
  [svgIcon]="gearIcon"
  text="Set default paste"
  [data]="dropdownButtonData"
>
</kendo-toolbar-dropdownbutton>
```

```
</kendo-toolbar>

    <p>{{ clickMessage }}</p>
  ,
})
export class AppComponent {
  public scrollModeSettings: ToolbarOverflowSettings = {
    mode: "scroll",
    prevSVGButtonIcon: arrowLeftIcon,
    nextSVGButtonIcon: arrowRightIcon,
  };

  public clickMessage: string;
  public toolbarWidth = 60;
  public undoIcon: SVGIcon = undoIcon;
  public redoIcon: SVGIcon = redoIcon;
  public boldIcon: SVGIcon = boldIcon;
  public italicIcon: SVGIcon = italicIcon;
  public underlineIcon: SVGIcon = underlineIcon;
  public alignLeftIcon: SVGIcon = alignLeftIcon;
  public alignCenterIcon: SVGIcon = alignCenterIcon;
  public alignRightIcon: SVGIcon = alignRightIcon;
  public alignJustifyIcon: SVGIcon = alignJustifyIcon;
  public clipboardIcon: SVGIcon = clipboardIcon;
  public gearIcon: SVGIcon = gearIcon;

  public splitButtonData: Array<{ [key: string]: unknown }> = [
    {
      text: "Keep Text Only",
      svgIcon: clipboardTextIcon,
      click: (): void =>
        this.onItemClick("Keep Text Only splitbutton button clicked"),
    },
    {
      text: "Paste as HTML",
      svgIcon: clipboardCodeIcon,
      click: (): void =>
        this.onItemClick("Paste as HTML splitbutton button clicked"),
    },
    {
      text: "Paste Markdown",
      svgIcon: clipboardMarkdownIcon,
      click: (): void =>
        this.onItemClick("Paste Markdown splitbutton button clicked"),
    },
  ];
  public dropdownButtonData: Array<{ [key: string]: unknown }> = [
    {
      text: "Keep Text Only",
      svgIcon: clipboardTextIcon,
```

```
    click: () =>
      this.onItemClick("Keep Text Only dropdownbutton button
    clicked"),
    },
  {
    text: "Paste as HTML",
    svgIcon: clipboardCodeIcon,
    click: () =>
      this.onItemClick("Paste as HTML dropdownbutton button clicked"),
  },
  {
    text: "Paste Markdown",
    svgIcon: clipboardMarkdownIcon,
    click: () =>
      this.onItemClick("Paste Markdown dropdownbutton button
    clicked"),
  },
];
}

public onItemClick(message: string): void {
  console.log(message);
  this.clickMessage = message;
}

```

Collasne Code ^

## Section Mode

The section mode allows you to display overflowing ToolBar tools in a popup element while also maintaining the components used for their rendering and their initial layout. To enable this behavior, set the value of the `overflow` property to `section`.

The following example demonstrates how to render overflowing ToolBar tools in a popup using their initial components and layout.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import {
  SVGIcon,
  alignCenterIcon,
```

```
alignJustifyIcon,
alignLeftIcon,
alignRightIcon,
boldIcon,
clipboardCodeIcon,
clipboardIcon,
clipboardMarkdownIcon,
clipboardTextIcon,
gearIcon,
italicIcon,
redoIcon,
underlineIcon,
undoIcon,
} from "@progress/kendo-svg-icons";

import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";

import { FormsModule } from "@angular/forms";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_LABELS } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [FormsModule, KENDO_TOOLBAR, KENDO_INPUTS, KENDO_LABELS],
  template: `
    <kendo-label [for]="width" text="Set toolbar width"></kendo-label>
    <kendo-slider
      #width
      [(ngModel)]="toolbarWidth"
      style="width: 100%; display: block;"
      [showButtons]="false"
      [min]="0"
      [max]="100"
      [largeStep]="1"
      tickPlacement="none"
    ></kendo-slider>

    <kendo-toolbar overflow="section" [style.width.%]="#toolbarWidth">
      <kendo-toolbar-butongroup>
        <kendo-toolbar-button
          [svgIcon]="undoIcon"
          (click)="onItemClick('Undo button clicked')"
        >
        </kendo-toolbar-button>
        <kendo-toolbar-button
          [svgIcon]="redoIcon"
          (click)="onItemClick('Redo button clicked')"
        >
        </kendo-toolbar-button>
      </kendo-toolbar-butongroup>
    </kendo-toolbar>
  
```

```
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-butongroup>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="boldIcon"
    (click)="onItemClick('Bold button clicked')"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="italicIcon"
    (click)="onItemClick('Italic button clicked')"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="underlineIcon"
    (click)="onItemClick('Underline button clicked')"
  >
  </kendo-toolbar-button>
</kendo-toolbar-butongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-butongroup selection="single">
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignLeftIcon"
    text="Left"
    (click)="onItemClick('Align left button clicked')"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignCenterIcon"
    text="Center"
    (click)="onItemClick('Align center button clicked')"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignRightIcon"
    text="Right"
    (click)="onItemClick('Align right button clicked')"
  >
  </kendo-toolbar-button>
  <kendo-toolbar-button
    [toggleable]="true"
    [svgIcon]="alignJustifyIcon"
    text="Justify"
    (click)="onItemClick('Align justify button clicked')"
  >
  </kendo-toolbar-button>
```

```
</kendo-toolbar-buttongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-splitbutton
  [svgIcon]="clipboardIcon"
  text="Paste"
  [data]="splitButtonData"
  (buttonClick)="onItemClick('Paste splitbutton clicked')"
>
</kendo-toolbar-splitbutton>
<kendo-toolbar-dropdownbutton
  [svgIcon]="gearIcon"
  text="Set default paste"
  [data]="dropdownButtonData"
>
</kendo-toolbar-dropdownbutton>
</kendo-toolbar>

  <p>{{ clickMessage }}</p>
``,
})
export class AppComponent {
  public clickMessage: string;
  public toolbarWidth = 60;
  public undoIcon: SVGIcon = undoIcon;
  public redoIcon: SVGIcon = redoIcon;
  public boldIcon: SVGIcon = boldIcon;
  public italicIcon: SVGIcon = italicIcon;
  public underlineIcon: SVGIcon = underlineIcon;
  public alignLeftIcon: SVGIcon = alignLeftIcon;
  public alignCenterIcon: SVGIcon = alignCenterIcon;
  public alignRightIcon: SVGIcon = alignRightIcon;
  public alignJustifyIcon: SVGIcon = alignJustifyIcon;
  public clipboardIcon: SVGIcon = clipboardIcon;
  public gearIcon: SVGIcon = gearIcon;

  public splitButtonData: Array<{ [key: string]: unknown }> = [
    {
      text: "Keep Text Only",
      svgIcon: clipboardTextIcon,
      click: (): void =>
        this.onItemClick("Keep Text Only splitbutton button clicked"),
    },
    {
      text: "Paste as HTML",
      svgIcon: clipboardCodeIcon,
      click: (): void =>
        this.onItemClick("Paste as HTML splitbutton button clicked"),
    },
    {
      text: "Paste Markdown",
      svgIcon: clipboardMarkdownIcon,
    }
  ];
}
```

```
    click: () =>
      this.onItemClick("Paste Markdown splitbutton button clicked"),
    },
];

public dropdownButtonData: Array<{ [key: string]: unknown }> = [
  {
    text: "Keep Text Only",
    svgIcon: clipboardTextIcon,
    click: () =>
      this.onItemClick("Keep Text Only dropdownbutton button clicked"),
  },
  {
    text: "Paste as HTML",
    svgIcon: clipboardCodeIcon,
    click: () =>
      this.onItemClick("Paste as HTML dropdownbutton button clicked"),
  },
  {
    text: "Paste Markdown",
    svgIcon: clipboardMarkdownIcon,
    click: () =>
      this.onItemClick("Paste Markdown dropdownbutton button clicked"),
  },
];

public onItemClick(message: string): void {
  console.log(message);
  this.clickMessage = message;
}
```

Collanse Code ^

## Menu Mode

The menu mode of the ToolBar displays tools that don't fit within the component's width in a popup, which opens when a designated button is clicked. To enable this behavior, set the value of the `overflow` property to `menu` or `true`.

The following example demonstrates how to display overflowing ToolBar tools in a menu popup element.

 EXAMPLE

 VIEW SOURCE



```
import { Component } from "@angular/core";
import {
  SVGIcon,
  alignCenterIcon,
  alignJustifyIcon,
  alignLeftIcon,
  alignRightIcon,
  boldIcon,
  clipboardCodeIcon,
  clipboardIcon,
  clipboardMarkdownIcon,
  clipboardTextIcon,
  gearIcon,
  italicIcon,
  redoIcon,
  underlineIcon,
  undoIcon,
} from "@progress/kendo-svg-icons";

import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";

import { FormsModule } from "@angular/forms";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_LABELS } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [FormsModule, KENDO_TOOLBAR, KENDO_INPUTS, KENDO_LABELS],
  template: `
    <kendo-label [for]="width" text="Set toolbar width"></kendo-label>
    <kendo-slider
      #width
      [(ngModel)]="toolbarWidth"
      style="width: 100%; display: block;"
      [showButtons]="false"
      [min]="0"
      [max]="100"
      [largeStep]="1"
      tickPlacement="none"
    ></kendo-slider>

    <kendo-toolbar overflow="menu" [style.width.%]="toolbarWidth">
      <kendo-toolbar-butongroup>
        <kendo-toolbar-button
          [svgIcon]="undoIcon"
          (click)="onItemClick('Undo button clicked')"
        >

```

```
</kendo-toolbar-button>
<kendo-toolbar-button
    [svgIcon] = "redoIcon"
    (click) = "onItemClick('Redo button clicked')"
>
</kendo-toolbar-button>
</kendo-toolbar-buttongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-buttongroup>
    <kendo-toolbar-button
        [toggleable] = "true"
        [svgIcon] = "boldIcon"
        (click) = "onItemClick('Bold button clicked')"
    >
    </kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable] = "true"
        [svgIcon] = "italicIcon"
        (click) = "onItemClick('Italic button clicked')"
    >
    </kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable] = "true"
        [svgIcon] = "underlineIcon"
        (click) = "onItemClick('Underline button clicked')"
    >
    </kendo-toolbar-button>
</kendo-toolbar-buttongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-buttongroup selection="single">
    <kendo-toolbar-button
        [toggleable] = "true"
        [svgIcon] = "alignLeftIcon"
        text = "Left"
        (click) = "onItemClick('Align left button clicked')"
    >
    </kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable] = "true"
        [svgIcon] = "alignCenterIcon"
        text = "Center"
        (click) = "onItemClick('Align center button clicked')"
    >
    </kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable] = "true"
        [svgIcon] = "alignRightIcon"
        text = "Right"
        (click) = "onItemClick('Align right button clicked')"
    >
    </kendo-toolbar-button>
```

```
        <kendo-toolbar-button  
          [toggleable]="true"  
          [svgIcon]="alignJustifyIcon"  
          text="Justify"  
          (click)="onItemClick('Align justify button clicked')"  
        >  
        </kendo-toolbar-button>  
      </kendo-toolbar-buttongroup>  
      <kendo-toolbar-separator></kendo-toolbar-separator>  
      <kendo-toolbar-splitbutton  
        [svgIcon]="clipboardIcon"  
        text="Paste"  
        [data]="splitButtonData"  
        (buttonClick)="onItemClick('Paste splitbutton clicked')"  
      >  
      </kendo-toolbar-splitbutton>  
      <kendo-toolbar-dropdownbutton  
        [svgIcon]="gearIcon"  
        text="Set default paste"  
        [data]="dropdownButtonData"  
      >  
      </kendo-toolbar-dropdownbutton>  
    </kendo-toolbar>  
  
    <p>{{ clickMessage }}</p>  
,
```

```
})  
export class AppComponent {  
  public clickMessage: string;  
  public toolbarWidth = 60;  
  public undoIcon: SVGIcon = undoIcon;  
  public redoIcon: SVGIcon = redoIcon;  
  public boldIcon: SVGIcon = boldIcon;  
  public italicIcon: SVGIcon = italicIcon;  
  public underlineIcon: SVGIcon = underlineIcon;  
  public alignLeftIcon: SVGIcon = alignLeftIcon;  
  public alignCenterIcon: SVGIcon = alignCenterIcon;  
  public alignRightIcon: SVGIcon = alignRightIcon;  
  public alignJustifyIcon: SVGIcon = alignJustifyIcon;  
  public clipboardIcon: SVGIcon = clipboardIcon;  
  public gearIcon: SVGIcon = gearIcon;  
  
  public splitButtonData: Array<{ [key: string]: unknown }> = [  
    {  
      text: "Keep Text Only",  
      svgIcon: clipboardTextIcon,  
      click: (): void =>  
        this.onItemClick("Keep Text Only splitbutton button clicked"),  
    },  
    {  
      text: "Paste as HTML",  
    },  
  ]  
}
```

```

        svgIcon: clipboardCodeIcon,
        click: () =>
          this.onItemClick("Paste as HTML splitbutton button clicked"),
      },
      {
        text: "Paste Markdown",
        svgIcon: clipboardMarkdownIcon,
        click: () =>
          this.onItemClick("Paste Markdown splitbutton button clicked"),
      },
    ];
}

public dropdownButtonData: Array<{ [key: string]: unknown }> = [
  {
    text: "Keep Text Only",
    svgIcon: clipboardTextIcon,
    click: () =>
      this.onItemClick("Keep Text Only dropdownbutton button clicked"),
  },
  {
    text: "Paste as HTML",
    svgIcon: clipboardCodeIcon,
    click: () =>
      this.onItemClick("Paste as HTML dropdownbutton button clicked"),
  },
  {
    text: "Paste Markdown",
    svgIcon: clipboardMarkdownIcon,
    click: () =>
      this.onItemClick("Paste Markdown dropdownbutton button clicked"),
  },
];
}

public onItemClick(message: string): void {
  console.log(message);
  this.clickMessage = message;
}
}

```

[Collapse Code ^](#)

## Customizing Tools Rendering

You can set the ToolBar tools to display their text or icon either in the ToolBar or in its overflow popup by using the `showText` or `showIcon` properties.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮

```
import { Component } from "@angular/core";
import {
  SVGIcon,
  alignCenterIcon,
  alignJustifyIcon,
  alignLeftIcon,
  alignRightIcon,
  boldIcon,
  clipboardCodeIcon,
  clipboardIcon,
  clipboardMarkdownIcon,
  clipboardTextIcon,
  gearIcon,
  italicIcon,
  redoIcon,
  underlineIcon,
  undoIcon,
} from "@progress/kendo-svg-icons";

import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";

import { FormsModule } from "@angular/forms";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_LABELS } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [FormsModule, KENDO_TOOLBAR, KENDO_INPUTS, KENDO_LABELS],
  template: `
    <kendo-label [for]="width" text="Set toolbar width"></kendo-label>
    <kendo-slider
      #width
      [(ngModel)]="toolbarWidth"
      style="width: 100%; display: block;"
      [showButtons]="false"
      [min]="0"
      [max]="100"
      [largeStep]="1"
      tickPlacement="none"
    ></kendo-slider>

    <kendo-toolbar overflow="menu" [style.width.%]="#toolbarWidth">
      <kendo-toolbar-buttongroup>
```

```
<kendo-toolbar-button
    text="Undo"
    showText="overflow"
    [svgIcon]="undoIcon"
    (click)="onItemClick('Undo button clicked')"
>
</kendo-toolbar-button>
<kendo-toolbar-button
    text="Redo"
    showText="overflow"
    [svgIcon]="redoIcon"
    (click)="onItemClick('Redo button clicked')"
>
</kendo-toolbar-button>
</kendo-toolbar-butongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-butongroup>
    <kendo-toolbar-button
        [toggleable]="true"
        text="Bold"
        showText="overflow"
        [svgIcon]="boldIcon"
        (click)="onItemClick('Bold button clicked')"
>
</kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable]="true"
        text="Italic"
        showText="overflow"
        [svgIcon]="italicIcon"
        (click)="onItemClick('Italic button clicked')"
>
</kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable]="true"
        text="Underline"
        showText="overflow"
        [svgIcon]="underlineIcon"
        (click)="onItemClick('Underline button clicked')"
>
</kendo-toolbar-button>
</kendo-toolbar-butongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-butongroup selection="single">
    <kendo-toolbar-button
        [toggleable]="true"
        [svgIcon]="alignLeftIcon"
        text="Left"
        showText="overflow"
        (click)="onItemClick('Align left button clicked')"
>
```

```
</kendo-toolbar-button>
<kendo-toolbar-button
  [toggleable]="true"
  [svgIcon]="alignCenterIcon"
  text="Center"
  showText="overflow"
  (click)="onItemClick('Align center button clicked')"
>
</kendo-toolbar-button>
<kendo-toolbar-button
  [toggleable]="true"
  [svgIcon]="alignRightIcon"
  text="Right"
  showText="overflow"
  (click)="onItemClick('Align right button clicked')"
>
</kendo-toolbar-button>
<kendo-toolbar-button
  [toggleable]="true"
  [svgIcon]="alignJustifyIcon"
  text="Justify"
  showText="overflow"
  (click)="onItemClick('Align justify button clicked')"
>
</kendo-toolbar-button>
</kendo-toolbar-buttongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-splitbutton
  [svgIcon]="clipboardIcon"
  showText="overflow"
  showIcon="toolbar"
  text="Paste"
  [data]="splitButtonData"
  (buttonClick)="onItemClick('Paste splitbutton clicked')"
>
</kendo-toolbar-splitbutton>
<kendo-toolbar-dropdownbutton
  [svgIcon]="gearIcon"
  showIcon="toolbar"
  text="Set default paste"
  showText="overflow"
  [data]="dropdownButtonData"
>
</kendo-toolbar-dropdownbutton>
</kendo-toolbar>

  <p>{{ clickMessage }}</p>
,
})
export class AppComponent {
  public clickMessage: string;
```

```
public toolbarWidth = 60;
public undoIcon: SVGIcon = undoIcon;
public redoIcon: SVGIcon = redoIcon;
public boldIcon: SVGIcon = boldIcon;
public italicIcon: SVGIcon = italicIcon;
public underlineIcon: SVGIcon = underlineIcon;
public alignLeftIcon: SVGIcon = alignLeftIcon;
public alignCenterIcon: SVGIcon = alignCenterIcon;
public alignRightIcon: SVGIcon = alignRightIcon;
public alignJustifyIcon: SVGIcon = alignJustifyIcon;
public clipboardIcon: SVGIcon = clipboardIcon;
public gearIcon: SVGIcon = gearIcon;

public splitButtonData: Array<{ [key: string]: unknown }> = [
  {
    text: "Keep Text Only",
    svgIcon: clipboardTextIcon,
    click: () =>
      this.onItemClick("Keep Text Only splitbutton button clicked"),
  },
  {
    text: "Paste as HTML",
    svgIcon: clipboardCodeIcon,
    click: () =>
      this.onItemClick("Paste as HTML splitbutton button clicked"),
  },
  {
    text: "Paste Markdown",
    svgIcon: clipboardMarkdownIcon,
    click: () =>
      this.onItemClick("Paste Markdown splitbutton button clicked"),
  },
];

public dropdownButtonData: Array<{ [key: string]: unknown }> = [
  {
    text: "Keep Text Only",
    svgIcon: clipboardTextIcon,
    click: () =>
      this.onItemClick("Keep Text Only dropdownbutton button clicked"),
  },
  {
    text: "Paste as HTML",
    svgIcon: clipboardCodeIcon,
    click: () =>
      this.onItemClick("Paste as HTML dropdownbutton button clicked"),
  },
  {
    text: "Paste Markdown",
    svgIcon: clipboardMarkdownIcon,
```

```
    click: () =>
      this.onItemClick("Paste Markdown dropdownbutton button
clicked"),
    },
];

public onItemClick(message: string): void {
  console.log(message);
  this.clickMessage = message;
}

```

[Collapse Code ▾](#)

## Customizing the Popup

You can customize the overflow button popup by setting the `popupSettings` property of the Toolbar.

The `popupSettings` input allows you to configure the following options:

- `animate`—Controls the popup animation. By default, the open and close animations are enabled.
- `popupClass`—Specifies a list of CSS classes that are used to style the popup.

The following example demonstrates how to turn off the default open and close animations and assign a CSS class to the popup to customize its height.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component, ViewEncapsulation } from "@angular/core";
import {
  SVGIcon,
  alignCenterIcon,
  alignJustifyIcon,
  alignLeftIcon,
  alignRightIcon,
  boldIcon,
  clipboardCodeIcon,
  clipboardIcon,
  clipboardMarkdownIcon,
  clipboardTextIcon,
```

```
gearIcon,
italicIcon,
redoIcon,
underlineIcon,
undoIcon,
} from "@progress/kendo-svg-icons";

import { KENDO_TOOLBAR } from "@progress/kendo-angular-toolbar";

import { FormsModule } from "@angular/forms";
import { KENDO_INPUTS } from "@progress/kendo-angular-inputs";
import { KENDO_LABELS } from "@progress/kendo-angular-label";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [FormsModule, KENDO_TOOLBAR, KENDO_INPUTS, KENDO_LABELS],
  styles: [
    `
      .my-class {
        height: 120px;
      }
    `,
  ],
  encapsulation: ViewEncapsulation.None,
  template: `
    <kendo-label [for]="width" text="Set toolbar width"></kendo-label>
    <kendo-slider
      #width
      [(ngModel)]="toolbarWidth"
      style="width: 100%; display: block;"
      [showButtons]="false"
      [min]="'0'"
      [max]="'100'"
      [largeStep]="'1'"
      tickPlacement="none"
    ></kendo-slider>

    <kendo-toolbar
      overflow="menu"
      [style.width.%]="'toolbarWidth'"
      [popupSettings]="popupSettings"
    >
      <kendo-toolbar-butongroup>
        <kendo-toolbar-button
          [svgIcon]="'undoIcon'"
          (click)="onItemClick('Undo button clicked')"
        >
        </kendo-toolbar-button>
        <kendo-toolbar-button
          [svgIcon]="'redoIcon'"
        >
        </kendo-toolbar-button>
    </kendo-toolbar>
  `
})
```

```
        (click)="onItemClick('Redo button clicked')"
    >
</kendo-toolbar-button>
</kendo-toolbar-buttongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-buttongroup>
    <kendo-toolbar-button
        [toggleable]="true"
        [svgIcon]="boldIcon"
        (click)="onItemClick('Bold button clicked')"
    >
</kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable]="true"
        [svgIcon]="italicIcon"
        (click)="onItemClick('Italic button clicked')"
    >
</kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable]="true"
        [svgIcon]="underlineIcon"
        (click)="onItemClick('Underline button clicked')"
    >
</kendo-toolbar-button>
</kendo-toolbar-buttongroup>
<kendo-toolbar-separator></kendo-toolbar-separator>
<kendo-toolbar-buttongroup selection="single">
    <kendo-toolbar-button
        [toggleable]="true"
        [svgIcon]="alignLeftIcon"
        text="Left"
        (click)="onItemClick('Align left button clicked')"
    >
</kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable]="true"
        [svgIcon]="alignCenterIcon"
        text="Center"
        (click)="onItemClick('Align center button clicked')"
    >
</kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable]="true"
        [svgIcon]="alignRightIcon"
        text="Right"
        (click)="onItemClick('Align right button clicked')"
    >
</kendo-toolbar-button>
    <kendo-toolbar-button
        [toggleable]="true"
        [svgIcon]="alignJustifyIcon"
```

```
        text="Justify"
        (click)="onItemClick('Align justify button clicked')"
    >
</kendo-toolbar-button>
</kendo-toolbar-buttongroup>
<kendo-toolbar-splitbutton
    [svgIcon]="clipboardIcon"
    text="Paste"
    [data]="splitButtonData"
    (buttonClick)="onItemClick('Paste splitbutton clicked')"
>
</kendo-toolbar-splitbutton>
<kendo-toolbar-dropdownbutton
    [svgIcon]="gearIcon"
    text="Set default paste"
    [data]="dropdownButtonData"
>
</kendo-toolbar-dropdownbutton>
</kendo-toolbar>

<p>{{ clickMessage }}</p>
``,
})
export class AppComponent {
    public clickMessage: string;
    public toolbarWidth = 50;
    public undoIcon: SVGIcon = undoIcon;
    public redoIcon: SVGIcon = redoIcon;
    public boldIcon: SVGIcon = boldIcon;
    public italicIcon: SVGIcon = italicIcon;
    public underlineIcon: SVGIcon = underlineIcon;
    public alignLeftIcon: SVGIcon = alignLeftIcon;
    public alignCenterIcon: SVGIcon = alignCenterIcon;
    public alignRightIcon: SVGIcon = alignRightIcon;
    public alignJustifyIcon: SVGIcon = alignJustifyIcon;
    public clipboardIcon: SVGIcon = clipboardIcon;
    public gearIcon: SVGIcon = gearIcon;

    public popupSettings = { animate: true, popupClass: "my-class" };

    public splitButtonData: Array<{ [key: string]: unknown }> = [
        {
            text: "Keep Text Only",
            svgIcon: clipboardTextIcon,
            click: (): void =>
                this.onItemClick("Keep Text Only splitbutton button clicked"),
        },
        {
            text: "Paste as HTML",
            svgIcon: clipboardCodeIcon,
            click: (): void =>
```

```
        this.onItemClick("Paste as HTML splitbutton button clicked"),
    },
    {
      text: "Paste Markdown",
      svgIcon: clipboardMarkdownIcon,
      click: (): void =>
        this.onItemClick("Paste Markdown splitbutton button clicked"),
    },
];
}

public dropdownButtonData: Array<{ [key: string]: unknown }> = [
{
  text: "Keep Text Only",
  svgIcon: clipboardTextIcon,
  click: (): void =>
    this.onItemClick("Keep Text Only dropdownbutton button
clicked"),
},
{
  text: "Paste as HTML",
  svgIcon: clipboardCodeIcon,
  click: (): void =>
    this.onItemClick("Paste as HTML dropdownbutton button clicked"),
},
{
  text: "Paste Markdown",
  svgIcon: clipboardMarkdownIcon,
  click: (): void =>
    this.onItemClick("Paste Markdown dropdownbutton button
clicked"),
},
];
}

public onItemClick(message: string): void {
  console.log(message);
  this.clickMessage = message;
}

```

Collanse Code ^

## Suggested Links

- [API Reference of the ToolBar](#)
- [ToolBar Appearance](#)

# Getting Started with the Kendo UI for Angular Tooltips

This guide provides the information you need to start using the Kendo UI for Angular Tooltips—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#).

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_TOOLTIPS } from "@progress/kendo-angular-tooltip";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";

@Component({
  standalone: true,
  selector: "my-app",
  imports: [KENDO_TOOLTIPS, KENDO_BUTTONS],
```

```
template: `<div kendoTooltip>
  <button kendoButton title="Saves the current
document">Save</button>
</div>
` ,
})
export class AppComponent {}
```

Collanse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Tooltips package:

1. Run the following command.

```
ng add @progress/kendo-angular-tooltip
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-tooltip` package as a dependency to the `package.json` file.
- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the required components, directives and utility arrays:

The utility arrays are available starting from **v16.6.0**. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

- To add all components from the Tooltips package, import the `KENDO_TOOLTIPS` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_TOOLTIPS } from '@progress/kendo-angular-tooltip';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_TOOLTIPS]
})
```

TS

- To add individual Tooltips components, import the corresponding utility arrays in your standalone component. See the list of [available utility arrays](#).

For example if you only need the Popover component, import `KENDO_POPOVER`.

```
import { Component } from '@angular/core';
import { KENDO_POPOVER } from '@progress/kendo-angular-tooltip';

@Component({
  standalone: true,
```

TS

```
        selector: 'my-app',
        imports: [KENDO_POPOVER]
    })
```

# Using the Components

1. After successfully installing the Tooltips package and importing the required components and directives, add the corresponding tags of the components you need in the `app.component.html`. For example, if you need the Tooltip component, add the following code:

```
<div kendoTooltip>
    <button kendoButton title="Saves the current
document">Save</button>
</div>
```

HTML

2. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

3. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular Tooltip component on the page.

# Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

# Next Steps

- [Tooltips Dependencies & Standalone Utilities](#)
- [Tooltip Overview](#)
- [Popover Overview](#)
- [Tooltips API Documentation](#)

# Learning Resources

- [Tooltips Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Angular Tooltips Overview

The Kendo UI for Angular Tooltips represent popups with information that is related to a single or multiple UI elements and which is displayed when an UI element is focused or hovered over.

The Tooltips are built from the ground up and specifically for Angular, so that you get high-performance tooltip controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



**Popover**

Component with rich popup content



**Tooltip**

Component that displays informative text in a popup

## Angular Tooltips Example

EXAMPLE

VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_TOOLTIP } from "@progress/kendo-angular-tooltip";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_TOOLTIP, KENDO_BUTTONS],
  template: `
    <div class="example-wrapper">
      <p class="instructions">
```

Hover over the yellow dots or click the 'Summary' button.

```
</p>
<div kendoTooltip filter="span" id="agglomerations">
    <span href="#" title="Canton - 26,300,000" id="canton"></span>
    <span href="#" title="Jakarta - 25,800,000" id="jakarta"></span>
    <span href="#" title="Mexico City - 23,500,000" id="mexico">
</span>
    <span href="#" title="Delhi - 23,500,000" id="delhi"></span>
    <span href="#" title="Karachi - 22,100,000" id="karachi"></span>
    <span href="#" title="New York - 21,500,000" id="newyork">
</span>
    <span href="#" title="Sao Paulo - 21,300,000" id="saopaulo">
</span>
    <span href="#" title="Mumbai/Bombay - 21,100,000" id="bombay">
</span>
    <span href="#" title="Los Angeles - 17,100,000" id="losangeles">
</span>
    <span href="#" title="Osaka - 16,800,000" id="osaka"></span>
    <span href="#" title="Moscow - 16,200,000" id="moscow"></span>
</div>

<button
    kendoPopoverAnchor
    #anchor="kendoPopoverAnchor"
    [popover]="myPopover"
    showOn="click"
    class="summary-btn"
    kendoButton
    themeColor="primary"
    >
    Summary
</button>

<kendo-popover #myPopover [width]="250" [height]="210"
position="top">
    <ng-template kendoPopoverBodyTemplate>
        <em>Top 3 biggest agglomerations:</em>
        <hr />
        <table>
            <tr>
                <th>Rank</th>
                <th>City</th>
                <th>Population</th>
            </tr>
            <tr>
                <td>1.</td>
                <td>Canton</td>
                <td>26,300,000</td>
            </tr>
            <tr>
                <td>2.</td>
```

```
        <td>Jakarta</td>
        <td>25,800,000</td>
    </tr>
    <tr>
        <td>3.</td>
        <td>Mexico City</td>
        <td>23,500,000</td>
    </tr>
</table>
<div class="actions">
    <button (click)="anchor.hide()" kendoButton
themeColor="primary">
        Close
    </button>
</div>
</ng-template>
</kendo-popover>
</div>
,
styles: [
    #agglomerations {
        position: relative;
        width: 692px;
        height: 480px;
        margin: 0 auto;
        background: url("assets/tooltips/world-map.jpg") no-repeat 0 0;
    }

    #agglomerations > span {
        cursor: pointer;
        position: absolute;
        display: block;
        width: 12px;
        height: 12px;
        background-color: #fff600;
        border-radius: 30px;
        border: 0;
        box-shadow: 0 0 0 1px rgba(0, 0, 0, 0.5);
        transition: box-shadow 0.3s;
        z-index: 10;
    }

    #agglomerations > span:hover {
        box-shadow: 0 0 0 15px rgba(0, 0, 0, 0.5);
    }

    .instructions {
        text-align: center;
        font-size: 16px;
    }
]
```

```
#canton {
    top: 226px;
    left: 501px;
}
#jakarta {
    top: 266px;
    left: 494px;
}
#mexico {
    top: 227px;
    left: 182px;
}
#delhi {
    top: 214px;
    left: 448px;
}
#karachi {
    top: 222px;
    left: 431px;
}
#newyork {
    top: 188px;
    left: 214px;
}
#saopaolo {
    top: 304px;
    left: 248px;
}
#bombay {
    top: 233px;
    left: 438px;
}
#losangeles {
    top: 202px;
    left: 148px;
}
#osaka {
    top: 201px;
    left: 535px;
}
#moscow {
    top: 153px;
    left: 402px;
}
.summary-btn {
    left: calc(50% - 40px);
    top: -63px;
}
table {
    width: 100%;
```

```
        }
      th {
        color: #528aae;
      }
      .actions {
        display: flex;
        justify-content: center;
        margin-top: 10px;
      }
    ,
  ],
})
export class AnnComponent {}
```

Collapse Code ^

# Angular Tooltips Key Features

The Kendo UI for Angular Tooltips package delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section.

## Anchor Elements

Both the Tooltip and Popover allow you to specify which elements (anchors) will render the popup element. [Read more about the anchor elements functionality of the Tooltip...](#)

## Programmatic Control

The Kendo UI for Angular Tooltips package provides options for consuming its components' API programmatically. [Read more about the programmatic consumption of the Tooltip API...](#)

## Positioning

The Kendo UI for Angular Tooltips come with built-in logic to help use the available space around the components and determine where to display them according to the anchor element. [Read more about the positioning of the Popover...](#)

# Templates

The Kendo UI for Angular Tooltips components enable you to provide a custom template for their content. The popover goes one step further by allowing a richer content to be added. [Read more about the content template feature of the Popover...](#)

## Accessibility

The Tooltips are accessible for screen readers and supports WAI-ARIA attributes. [Read more about the accessibility of the Tooltip...](#)

## Keyboard Navigation

The Tooltips support a number of keyboard shortcuts which allows users to accomplish various commands. [Read more about the keyboard navigation of the Tooltip...](#)

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular Tooltip is part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of the Kendo UI for Angular Tooltip, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual

developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).

- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Tooltips](#)
- [API Reference of the Tooltips](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Popover Overview

The Popover represents a popup with rich content that is related to a single or multiple UI elements. It could be displayed when the target element is hovered, clicked or focused.

The following example demonstrates the Kendo UI for Angular Popover in action.

## EXAMPLE

</> VIEW SOURCE



```
</span>
    </ng-template>
  </kendo-calendar>
</div>

<kendo-popover
  #myPopover
  [width]="250"
  [height]="300"
  [callout]="true"
  [offset]="0"
  [templateData]="getContextData"
>
  <ng-template kendoPopoverTitleTemplate let-anchor let-data="data">
    @if (!data) {
      <div class="centered-wrapper">
        <span>No events for this day</span>
      </div>
    } @else {
      <div class="centered-wrapper">
        <span class="title">{{ data?.title }}</span>
      </div>
    }
  </ng-template>

  <ng-template kendoPopoverBodyTemplate let-anchor let-data="data">
    @if (!data) {
      <div class="centered-wrapper">
        <kendo-svg-icon [icon]="inboxIcon"></kendo-svg-icon>
      </div>
    } @if (data) {
      <div>
        @for (event of data.events; track event) {
          <ng-container # eventData>
            <div
              [ngStyle]="{
                'background-color': event.colorCoding,
                color: event.color
              }"
              class="event-wrapper"
            >
              <span>{{ event?.name }}</span>
              <span>{{ event?.start }}</span>
            </div>
          </ng-container>
        } @if (data.url) {
          <div class="centered-wrapper margin-top--20">
            <button kendoButton fillMode="flat" themeColor="primary">
              See more...
            </button>
          </div>
        }
      </div>
    }
  </ng-template>
</kendo-popover>
```

```

        }
      </div>
    }
  </ng-template>
</kendo-popover>
`,
styleUrls: ['./styles.css'],
encapsulation: ViewEncapsulation.None,
})
export class AppComponent {
  public data = eventData;
  public value: Date = new Date(2021, 11, 12);
  public inboxIcon: SVGIcon = inboxIcon;

  public getContextData = (anchor: HTMLElement): IEventsData => {
    if (anchor.title === "Friday, December 10, 2021") {
      return this.data[0];
    }

    if (anchor.title === "Tuesday, December 21, 2021") {
      return this.data[1];
    }
  };

  public isConference(date: Date): string {
    return date.toJSON().includes("2021-12-09")
      ? "conference day1"
      : date.toJSON().includes("2021-12-20")
      ? "conference day2"
      : "";
  }
}

```

[Collapse Code ^](#)

## Angular Popover Key Features

The Kendo UI for Angular Popover delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests ongoing efforts to improve the performance and add more value to the existing Popover as well as develop new features to it.

- **Single Anchor Element**—A quick way to display the Popover for a single anchor element.

- **Multiple Anchor Elements**—Define a container, which holds multiple anchor elements for the same Popover instance.
- **Programmatic Control**—Take advantage of the Popover's provided options for consuming its API programmatically.
- **Positioning**—The Popover allows the option to set the position in which it is opened in regards to its target element.
- **Templates**—Provide custom content for the Popover's header, body and actions sections.
- **Animations**—Enable animations that will be applied on the opening of the Popover.

## Support and Learning Resources

- [Popover Homepage](#)
- [Getting Started with the Kendo UI for Angular Tooltips](#)
- [API Reference of the Popover](#)
- [API Reference of the PopoverAnchor](#)
- [API Reference of the PopoverContainer](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Popover Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)



# Kendo UI for Angular Tooltip Overview

The Tooltip represents a popup with information that is related to a UI element and which is displayed when this UI element is focused or hovered over.

The following example demonstrates the Kendo UI for Angular Tooltip directive in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_TOOLTIPS } from "@progress/kendo-angular-tooltip";

@Component({
  // eslint-disable-next-line @angular-eslint/component-selector
  selector: "my-app",
  standalone: true,
  imports: [KENDO_TOOLTIPS],
  template: `
    <div kendoTooltip filter="span" id="agglomerations">
      <span href="#" title="Canton - 26,300,000" id="canton"></span>
      <span href="#" title="Jakarta - 25,800,000" id="jakarta"></span>
      <span href="#" title="Mexico City - 23,500,000" id="mexico">
        </span>
      <span href="#" title="Delhi - 23,500,000" id="delhi"></span>
      <span href="#" title="Karachi - 22,100,000" id="karachi"></span>
      <span href="#" title="New York - 21,500,000" id="newyork"></span>
      <span href="#" title="Sao Paulo - 21,300,000" id="saopaulo">
        </span>
      <span href="#" title="Mumbai/Bombay - 21,100,000" id="bombay"></span>
      <span href="#" title="Los Angeles - 17,100,000" id="losangeles">
        </span>
      <span href="#" title="Osaka - 16,800,000" id="osaka"></span>
      <span href="#" title="Moscow - 16,200,000" id="moscow"></span>
    </div>
  `,
}
```

```
  styles: [  
    #agglomerations {  
      position: relative;  
      width: 692px;  
      height: 480px;  
      margin: 0 auto;  
      background: url("^assets/tooltips/world-map.jpg") no-repeat 0 0;  
    }  
  
    #agglomerations > span {  
      cursor: pointer;  
      position: absolute;  
      display: block;  
      width: 12px;  
      height: 12px;  
      background-color: #fff600;  
      border-radius: 30px;  
      border: 0;  
      box-shadow: 0 0 0 1px rgba(0, 0, 0, 0.5);  
      transition: box-shadow 0.3s;  
      z-index: 10;  
    }  
  
    #agglomerations > span:hover {  
      box-shadow: 0 0 0 15px rgba(0, 0, 0, 0.5);  
    }  
  
    #canton {  
      top: 226px;  
      left: 501px;  
    }  
    #jakarta {  
      top: 266px;  
      left: 494px;  
    }  
    #mexico {  
      top: 227px;  
      left: 182px;  
    }  
    #delhi {  
      top: 214px;  
      left: 448px;  
    }  
    #karachi {  
      top: 222px;  
      left: 431px;  
    }  
    #newyork {  
      top: 188px;  
      left: 214px;  
    }  
  ]
```

```
        }
      #saopaolo {
        top: 304px;
        left: 248px;
      }
      #bombay {
        top: 233px;
        left: 438px;
      }
      #losangeles {
        top: 202px;
        left: 148px;
      }
      #osaka {
        top: 201px;
        left: 535px;
      }
      #moscow {
        top: 153px;
        left: 402px;
      }
    },
  ],
})
export class AppComponent {}
```

[Collapse Code ^](#)

## Angular Tooltip Key Features

The Kendo UI for Angular Tooltip directive delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests ongoing efforts to improve the performance and add more value to the existing Tooltip directive as well as develop new features to it.

## Anchor Elements

The Tooltip allows you to specify which elements (anchors) will render a tooltip. [Read more about the anchor elements functionality of the Tooltip...](#)

# Programmatic Opening

The Tooltip provides options for consuming its API programmatically. [Read more about the programmatic consumption of the Tooltip API...](#)

## Closable Tooltip

The Tooltip provides an option for closing its content with a button. [Read more about the closable Tooltip feature...](#)

## Positioning

The Tooltip allows you to change the position in which it is opened. [Read more about the positioning feature of the Tooltip...](#)

## Templates

The Tooltip enables you to provide a custom template for its content. [Read more about the content template feature of the Tooltip...](#)

# Support and Learning Resources

- [Tooltip Homepage](#)
- [Getting Started with the Kendo UI for Angular Tooltips](#)
- [API Reference of the Tooltip](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Tooltip Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Typography

This guide provides the information you need to start using the Kendo UI for Angular Typography—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#)

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_TYPOGRAPHY } from "@progress/kendo-angular-typography";

@Component({
  standalone: true,
  imports: [KENDO_TYPOGRAPHY],
  selector: "my-app",
  template: `
```

```
<div kendoTypography variant="h1">Heading 1</div>
<div kendoTypography variant="h2">Heading 2</div>
<div kendoTypography variant="h3">Heading 3</div>
<div kendoTypography variant="h4">Heading 4</div>
<div kendoTypography variant="h5">Heading 5</div>
<div kendoTypography variant="h6">Heading 6</div>
<div kendoTypography variant="p">I am a paragraph</div>

<code kendoTypography variant="code">code</code>

<pre kendoTypography variant="pre">
  <code kendoTypography variant="code">{{code}}</code>
</pre>
``,
})
export class AppComponent {
  public code =
    const handleChange = ({ target }) => {
      const { name, value } = target;
      const newData = Object.assign({}, data, { [name]: value });
      setData(newData);
    };
}
}

Collanse Code ^
```

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

# Installing the Component

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Typography package:

## 1. Run the following command:

```
ng add @progress/kendo-angular-typography
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-typography` package as a dependency to the `package.json` file.
- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the `KENDO_TYPOGRAPHY` utility array in your standalone component to enable the entire feature set of the Typography:

The utility array is available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

```
import { Component } from '@angular/core';
import { KENDO_TYPOGRAPHY } from '@progress/kendo-angular-typography';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_TYPOGRAPHY]
})
```

TS

# Using the Directive

1. After successfully installing the Typography package, add the following code:

```
<div kendoTypography="h1">Heading 1</div>
```

HTML

2. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

3. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular Typography directive on the page.

## Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

## Next Steps

- [Typography Dependencies & Standalone Utilities](#)
- [Typography Options](#)
- [API Reference of the Typography](#)

## Learning Resources

- [Typography Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)

- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Typography Options

The Kendo UI for Angular Typography directive provides multiple options for styling UI elements. You can apply an appearance variant to the element, configure the font size and weight and the alignment and transformation of the text, and also change the theme color.

## Element Variants

The Typography provides default appearance classes for the `<h1>` to `<h6>` and `<p>` elements. You can set these by using the `variant` property.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_TYPOGRAPHY } from "@progress/kendo-angular-typography";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_TYPOGRAPHY],
  template: `
    <div kendoTypography variant="h1">Heading 1</div>
    <div kendoTypography variant="h2">Heading 2</div>
    <div kendoTypography variant="h3">Heading 3</div>
    <div kendoTypography variant="h4">Heading 4</div>
    <div kendoTypography variant="h5">Heading 5</div>
    <div kendoTypography variant="h6">Heading 6</div>
    <div kendoTypography variant="p">I am a paragraph</div>

    <code kendoTypography variant="code">code</code>

    <pre kendoTypography variant="pre">
      <code kendoTypography variant="code">{{code}}</code>
    </pre>
  `,
})
```

```
)  
export class AppComponent {  
  public code =`  
    const handleChange = ({ target }) => {  
      const { name, value } = target;  
      const newData = Object.assign({}, data, { [name]: value });  
      setData(newData);  
    };  
}  
`
```

Collanse Code ^

# Font Size

To set the font size of the element, use the `fontSize` property.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";  
  
import { KENDO_TYPOGRAPHY } from "@progress/kendo-angular-typography";  
  
@Component({  
  selector: "my-app",  
  standalone: true,  
  imports: [KENDO_TYPOGRAPHY],  
  template:  
    <div kendoTypography fontSize="xs">extra small</div>  
    <div kendoTypography fontSize="sm">small</div>  
    <div kendoTypography fontSize="md">medium</div>  
    <div kendoTypography fontSize="lg">large</div>  
    <div kendoTypography fontSize="xl">extra large</div>  
  ,  
})  
export class AppComponent {}
```

Collanse Code ^

# Font Weight

To set the font weight of the element, use the `fontWeight` property.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import { KENDO_TYPOGRAPHY } from "@progress/kendo-angular-typography";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_TYPOGRAPHY],
  template: `
    <div kendoTypography fontWeight="light">light</div>
    <div kendoTypography fontWeight="normal">normal</div>
    <div kendoTypography fontWeight="bold">bold</div>
  `,
})
export class AppComponent {}
```

[Collapse Code ^](#)

## Text Alignment

To set the text alignment of the element, use the `textAlign` property.

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import { KENDO_TYPOGRAPHY } from "@progress/kendo-angular-typography";

@Component({
  selector: "my-app",
  standalone: true,
```

```
imports: [KENDO_TYPOGRAPHY],  
template:  
  <div kendoTypography textAlign="left">left</div>  
  <div kendoTypography textAlign="center">center</div>  
  <div kendoTypography textAlign="right">right</div>  
,  
})  
export class AppComponent {}
```

Collanse Code ^

# Text Transformation

To set the text transformation of the element, use the `textTransform` property.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";  
  
import { KENDO_TYPOGRAPHY } from "@progress/kendo-angular-typography";  
  
@Component({  
  selector: "my-app",  
  standalone: true,  
  imports: [KENDO_TYPOGRAPHY],  
  template:  
    <div kendoTypography textTransform="lowercase">lowercase</div>  
    <div kendoTypography textTransform="uppercase">uppercase</div>  
    <div kendoTypography textTransform="capitalize">capitalize</div>  
,  
})  
export class AppComponent {}
```

Collanse Code ^

# Theme Color

To set the color of the element, use the `themeColor` property.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component } from "@angular/core";
import { KENDO_TYPOGRAPHY } from "@progress/kendo-angular-typography";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_TYPOGRAPHY],
  template: `
    <div kendoTypography themeColor="inherit">inherit</div>
    <div kendoTypography themeColor="primary">primary</div>
    <div kendoTypography themeColor="secondary">secondary</div>
    <div kendoTypography themeColor="tertiary">tertiary</div>
    <div kendoTypography themeColor="info">info</div>
    <div kendoTypography themeColor="success">success</div>
    <div kendoTypography themeColor="warning">warning</div>
    <div kendoTypography themeColor="error">error</div>
    <div kendoTypography themeColor="dark">dark</div>
    <div kendoTypography themeColor="light">light</div>
    <div kendoTypography themeColor="inverse">inverse</div>
  `,
})
export class AppComponent {}
```

 Collapce Code ^ 

## Suggested Links

- [API Reference of the Typography Directive](#)
- [API Index of the Typography](#)

# Angular Typography Overview

The Kendo UI for Angular Typography directive allows you to improve the appearance of any UI element by using the available Kendo UI theme classes.

The provided functionality is suitable for users who want to unify the appearance of identical UI elements or to benefit from the built-in Kendo UI theme CSS classes.

## Angular Typography Example

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_TYPOGRAPHY } from "@progress/kendo-angular-typography";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [KENDO_TYPOGRAPHY],
  template: `
    <h1>Variants:</h1>

    <div kendoTypography variant="h1">Heading 1</div>
    <div kendoTypography variant="h2">Heading 2</div>
    <div kendoTypography variant="h3">Heading 3</div>
    <div kendoTypography variant="h4">Heading 4</div>
    <div kendoTypography variant="h5">Heading 5</div>
    <div kendoTypography variant="h6">Heading 6</div>
    <div kendoTypography variant="p">I am a paragraph</div>

    <code kendoTypography variant="code">code</code>

    <pre kendoTypography variant="pre">
```

```
        <code kendoTypography variant="code">{ {code} } </code>
    </pre>

    <hr />

    <h1>Font Size:</h1>

    <div kendoTypography fontSize="xs">extra small</div>
    <div kendoTypography fontSize="sm">small</div>
    <div kendoTypography fontSize="md">medium</div>
    <div kendoTypography fontSize="lg">large</div>
    <div kendoTypography fontSize="xl">extra large</div>

    <hr />

    <h1>Font Weights:</h1>

    <div kendoTypography fontWeight="light">light</div>
    <div kendoTypography fontWeight="normal">normal</div>
    <div kendoTypography fontWeight="bold">bold</div>
    <hr />

    <h1>Text Align:</h1>
    <div kendoTypography textAlign="left">left</div>
    <div kendoTypography textAlign="center">center</div>
    <div kendoTypography textAlign="right">right</div>
    <hr />

    <h1>Text Transform:</h1>

    <div kendoTypography textTransform="lowercase">lowercase</div>
    <div kendoTypography textTransform="uppercase">uppercase</div>
    <div kendoTypography textTransform="capitalize">capitalize</div>
    <hr />

    <h1>Theme Color:</h1>
    <div kendoTypography themeColor="inherit">inherit</div>
    <div kendoTypography themeColor="primary">primary</div>
    <div kendoTypography themeColor="secondary">secondary</div>
    <div kendoTypography themeColor="tertiary">tertiary</div>
    <div kendoTypography themeColor="info">info</div>
    <div kendoTypography themeColor="success">success</div>
    <div kendoTypography themeColor="warning">warning</div>
    <div kendoTypography themeColor="error">error</div>
    <div kendoTypography themeColor="dark">dark</div>
    <div kendoTypography themeColor="light">light</div>
    <div kendoTypography themeColor="inverse">inverse</div>
    ,
}

export class AppComponent {
    public code = `
```

```
const handleChange = ({ target }) => {
  const { name, value } = target;
  const newData = Object.assign({}, data, { [name]: value });
  setData(newData);
}`;
}
```

Collapse Code ^

# Angular Typography Key Features

The Kendo UI for Angular Typography directive delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests ongoing efforts to improve the performance and add more value to the existing Typography directive as well as develop new features to it.

## Styling Options

The Typography delivers default appearance classes for HTML elements. That allows you to set an element variant, configure the font size and weight, and the alignment and transformation of text, as well as change the theme color. [Read more about the options of the Typography...](#)

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular Typography is part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library —no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

# Support Options

For any questions about the use of the Kendo UI for Angular Typography, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

# Support and Learning Resources

- [Typography Homepage](#)
- [Getting Started with the Kendo UI for Angular Typography](#)
- [API Reference of the Typography](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Typography Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular FileSelect Overview

The Kendo UI for Angular FileSelect component helps users select single or multiple files from their local file systems. The component is especially useful when you want full control over the process of creating the server requests and sent forms. It is a richer version of an `<input type='file'>` element and supports model binding, templates, forms, and more.

The following example demonstrates the FileSelect in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";

@Component({
  selector: "my-app",
  template: `<kendo-fileselect></kendo-fileselect>`,
})
export class AppComponent {}
```

[Collapse Code ^](#)

# Key Features

- [Forms support](#)—You can use the FileSelect both in template-driven and reactive Angular forms.
- [Model binding](#)—The FileSelect provides settings for binding its value to the model and render initial files by using the `ngModel` Angular directive.
- [Disabled FileSelect](#)—You can use the configuration options of the FileSelect to disable the component so that users are not able to interact with it.
- [File restrictions](#)—The FileSelect also delivers a set of file restriction rules, which refer to the size and extension of the selected files.
- [External dropzone](#)—The FileSelect provides options to the users for dragging and dropping files from their files systems to a dedicated user interface by using the External DropZone Component or by adding the External DropZone Directive.
- [Templates](#)—You can either fully customize the FileSelect file list by using the File template, or customize part of its elements by utilizing the File Info template.
- [Globalization](#)—All Kendo UI for Angular Uploads provide globalization options.
- [Accessibility](#)—The FileSelect is accessible for screen readers and supports WAI-ARIA attributes.
- [Keyboard navigation](#)—The FileSelect supports a number of keyboard shortcuts for processing various commands.

# Support and Learning Resources

- [FileSelect Homepage](#)
- [Getting Started with the Kendo UI for Angular Uploads](#)
- [API Reference of the FileSelect](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [FileSelect Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Getting Started with the Kendo UI for Angular Uploads

This guide provides the information you need to start using the Kendo UI for Angular Uploads—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#).

The standalone components in Angular streamline development by removing the need for [NgModules](#), reducing complexity, and enhancing component reuse and modularity. This approach simplifies dependency management, making applications more maintainable and scalable.

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_UPLOADS } from "@progress/kendo-angular-upload";
import { HttpClientModule } from "@angular/common/http";

@Component({
  standalone: true,
  selector: "my-app",
  imports: [KENDO_UPLOADS, HttpClientModule],
```

```
template: `<kendo-fileselect></kendo-fileselect>`,  
})  
export class AppComponent {}
```

Collanse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Uploads package:

1. Run the following command.

```
ng add @progress/kendo-angular-upload
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-upload` package as a dependency to the `package.json` file.
- Add all required peer dependencies to the `package.json` file.
- Register the Kendo UI Default theme in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the required components, directives and utility arrays:

The utility arrays are available starting from `v16.6.0`. If you use an older version of the package, please follow the approach from the [Using Kendo Angular Components with NgModules](#) article.

- To add all components from the Uploads package, import the `KENDO_UPLOADS` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_UPLOADS } from '@progress/kendo-angular-upload';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_UPLOADS]
})
```

TS

- To add individual Uploads components, import the corresponding utility arrays in your standalone component. See the list of [available utility arrays](#).

For example if you only need the FileSelect component, import `KENDO_FILESELECT`.

```
import { Component } from '@angular/core';
import { KENDO_FILESELECT } from '@progress/kendo-angular-upload';

@Component({
  standalone: true,
```

TS

```
        selector: 'my-app',
        imports: [KENDO_FILESELECT]
    })
```

# Using the Components

1. After successfully installing the `Uploads` package and importing the desired modules, add the corresponding tags of the components you need in the `app.component.html`. For example, if you need the `FileSelect` component, add the following code:

```
<kendo-fileselect> </kendo-fileselect>
```

HTML

2. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

3. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular `FileSelect` component on the page.

# Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

# Next Steps

- [Uploads Dependencies & Standalone Utilities](#)

- [FileSelect Overview](#)
- [Upload Overview](#)
- [Globalization](#)
- [Uploads API Documentation](#)

# Learning Resources

- [Uploads Overview](#)
- [Get Started with Kendo UI for Angular](#)
- [Using Kendo UI for Angular with Angular CLI](#)
- [Styling Overview](#)
- [Theme Builder](#)
- [Customizing the Themes](#)
- [Activating Your License Key](#)
- [Video Courses](#)

# Angular Uploads Overview

The Kendo UI for Angular Uploads help users select files from their file systems and send them to dedicated server handlers configured to receive them.

The Uploads are built from the ground up and specifically for Angular, so that you get high-performance upload controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



**FileSelect**

Component for selecting and listing user files



**Upload**

Component for selecting and uploading user files to a server

## Angular Uploads Example

EXAMPLE

VIEW SOURCE

⋮

```
import { Component, Injectable } from "@angular/core";
import {
  HttpInterceptor,
  HttpRequest,
  HttpHandler,
  HttpEvent,
  HttpProgressEvent,
  HttpEventType,
  HttpResponse,
} from "@angular/common/http";
import { Observable, of, concat } from "rxjs";
import { delay } from "rxjs/operators";
```

```

@Component({
  selector: "my-app",
  template: `<my-uploads></my-uploads>`,
})
export class AppComponent {}

/*
  Mocked backend service.
  For further details, check
  https://angular.io/guide/http#writing-an-interceptor
*/

@Injectable()
export class UploadInterceptor implements HttpInterceptor {
  intercept(
    req: HttpRequest<unknown>,
    next: HttpHandler
  ): Observable<HttpEvent<unknown>> {
    if (req.url === "saveUrl") {
      const events: Observable<HttpEvent<unknown>>[] = [0, 30, 60,
100].map(
        (x) =>
        of(<HttpProgressEvent>{
          type: HttpEventType.UploadProgress,
          loaded: x,
          total: 100,
        }).pipe(delay(1000))
      );
      const success = of(new HttpResponse({ status: 200
})).pipe(delay(1000));
      events.push(success);

      return concat(...events);
    }

    if (req.url === "removeUrl") {
      return of(new HttpResponse({ status: 200 }));
    }

    return next.handle(req);
  }
}

```

[Collapse Code ^](#)

# Angular Uploads Key Features

Each Kendo UI for Angular Uploads component delivers a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Uploads library as well as develop new features and controls to it.

## Disabled Uploads

You can choose to render the Uploads in their disabled state so that, if need be present, users will not be able to interact with them. [Read more about the Disabled FileSelect...](#)

## External Dropzone

With the Uploads you can enable users to drag and drop files from their file systems to a dedicated user interface by using the drag-and-drop tool or by creating custom elements with a drag-and-drop functionality. [Read more about implementing the external dropzone FileSelect feature...](#)

## Forms Support

The Uploads provide support both for the asynchronous template-driven Angular forms and the predominantly synchronous reactive Angular forms. This feature allows you to draw on the logic set either in the template, or in the component or typescript code. [Read more about the forms support of the FileSelect...](#)

## Model Binding

The Uploads allow you to bind their value to the model and render initial files by using the `ngModel` Angular directive. [Read more about implementing model binding in the FileSelect...](#)

## Restrictions

The Uploads provide approaches for restricting the selected files based on predefined rules such as minimum or maximum file size, and file extension. [Read more about adding file restrictions to the FileSelect...](#)

## Templates

You can customize the way the Uploads display their file lists by utilizing the available file templates. [Read more about customizing the file list of the FileSelect...](#)

## Globalization

The Kendo UI for Angular Uploads support globalization to ensure that each Uploads component can fit well in any application, no matter what languages and locales need to be supported. Additionally, the Uploads support rendering in a right-to-left (RTL) direction. [Read more about Uploads globalization...](#)

## Accessibility

The Uploads are accessible for screen readers and support WAI-ARIA attributes. [Read more about accessibility support of the FileSelect...](#)

## Keyboard Navigation

The Uploads support a number of keyboard shortcuts which allow users to accomplish various commands. [Read more about the keyboard navigation of the FileSelect...](#)

# Trial Version of Kendo UI for Angular

The Kendo UI for Angular Uploads are part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library —no restrictions! What's more, you are eligible for full technical support during your

trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

# Support Options

For any questions about the use of Kendo UI for Angular Uploads, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.
- [Kendo UI for Angular feedback portal](#) and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

# Suggested Links

- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)

# Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Uploads](#)
- [API Reference of the Uploads](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Upload Overview

The Kendo UI for Angular Upload component helps users to send files from their file systems to dedicated server handlers which are configured to receive them. Once the files have been selected they are send to the server through a request.

The following example demonstrates the Upload in action.

 EXAMPLE

 VIEW SOURCE

⋮

```
import { Component, Injectable } from "@angular/core";
import {
  HttpInterceptor,
  HttpRequest,
  HttpHandler,
  HttpEvent,
  HttpProgressEvent,
  HttpEventType,
  HttpResponse,
} from "@angular/common/http";
import { Observable, of, concat } from "rxjs";
import { delay } from "rxjs/operators";

@Component({
  selector: "my-app",
  template: `<my-upload></my-upload>`,
})
export class AppComponent {}

/*
  Mocked backend service.
  For further details, check
  https://angular.io/guide/http#writing-an-interceptor
*/

@Injectable()
export class UploadInterceptor implements HttpInterceptor {
```

```

intercept(
  req: HttpRequest<unknown>,
  next: HttpHandler
): Observable<HttpEvent<unknown>> {
  if (req.url === "saveUrl") {
    const events: Observable<HttpEvent<unknown>>[] = [0, 30, 60,
100].map(
      (x) =>
      of(<HttpProgressEvent>{
        type: HttpEventType.UploadProgress,
        loaded: x,
        total: 100,
      }).pipe(delay(1000))
    );
  }

  const success = of(new HttpResponse({ status: 200
})).pipe(delay(1000));
  events.push(success);

  return concat(...events);
}

if (req.url === "removeUrl") {
  return of(new HttpResponse({ status: 200 }));
}

return next.handle(req);
}
}

```

[Collapse Code ^](#)

# Key Features

- **File processing**—The Upload delivers a number of options when processing the files for upload such as uploading single and multiple files, uploading files in chunks and batches of files, and more.
- **Batch upload**—The batch upload feature of the Upload component enables you to show files, which were selected at once, as a single file-list item and upload them in a single POST request.
- **Chunk upload**—The chunk upload feature of the Upload component enables you to split files into chunks and send them asynchronously through multiple requests to the server

- **Action buttons**—The Upload enables you to render action buttons and customize their layout.
- **Forms support**—You can use the Upload both in template-driven and reactive Angular forms.
- **Model binding**—The Upload provides settings for binding its value to the model and render initial files by using the `ngModel` Angular directive.
- **Disabled Upload**—You can use the configuration options of the Upload to disable the component so that users are not able to interact with it.
- **File restrictions**—The Upload also delivers a set of file restriction rules, which refer to the size and extension of the files for upload.
- **External dropzone**—The Upload provides options to the users for dragging and dropping files from their files systems to a dedicated user interface by using the External DropZone Component or by adding the External DropZone Directive.
- **Templates**—You can either fully customize the Upload file list by using the File template, or customize part of its elements by utilizing the File Info template.
- **Credentials and additional data**—The Upload allows you to decide whether to send cookies and headers for cross-site requests or additional data during the upload or removal of files.
- **Globalization**—All Kendo UI for Angular Uploads provide globalization options.
- **Accessibility**—The Upload is accessible for screen readers and supports WAI-ARIA attributes.
- **Keyboard navigation**—The FileSelect supports a number of keyboard shortcuts for processing various commands.

## Support and Learning Resources

- [Upload Homepage](#)
- [Getting Started with the Kendo UI for Angular Uploads](#)
- [API Reference of the Upload](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Upload Forum](#)

- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

# Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)

# Kendo UI for Angular Drag and Drop Overview

The Kendo UI for Angular Drag and Drop functionality helps to facilitate moving items within an Angular application.

The provided functionality allows the user to define any HTML element or Angular component as either drag or drop target. Customization of the interaction between targets could be achieved by utilizing a wide range of exposed properties and events.

The following example demonstrates the Drag and Drop functionality in action.

 EXAMPLE

 VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_UTILS } from "@progress/kendo-angular-utils";
import { arrowsMoveIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
  imports: [CommonModule, KENDO_BUTTONS, KENDO_UTILS],
  template:
    <div class="container">
      @for (dropTarget of dropTargets; track dropTarget) {
        <div
          kendoDropTarget
          (onDragEnter)="handleDragEnter(dropTarget)"
          (onDrop)="handleDrop(dropTarget)"
          (onDragLeave)="handleDragLeave()"
          [ngClass]="{
            'my-drop': true,
            entered: enteredBox === dropTarget && enteredBox !==
            currentBox
          }"
        >
      
```

```

        }
    >
    @if (currentBox === dropTarget) {
        <button
            kendoButton
            kendoDragTarget
            themeColor="primary"
            [svgIcon]="moveIcon"
        >
            Drag Me!
        </button>
    }
    {{ currentBox !== dropTarget ? "Drop Here" : "" }}
</div>
}
,
styleUrls: ['./styles.css'],
})
export class AppComponent {
    public moveIcon: SVGIcon = arrowsMoveIcon;
    public dropTargets = ["A", "B", "C", "D"];
    public currentBox = "A";
    public enteredBox = "A";

    public handleDragEnter(id: string): void {
        this.enteredBox = id;
    }

    public handleDragLeave(): void {
        this.enteredBox = "";
    }

    public handleDrop(id: string): void {
        this.currentBox = id;
    }
}

```

[Collapse Code ▲](#)

# Key Features

- **Drag Target**—Define any HTML element or an Angular component as a drag target.
- **Drop Target**—Define any HTML element or an Angular component as a drop target.
- **Drag Target Container**—Defines multiple HTML elements or Angular components within a container as drag targets.

- **Drop Target Container**—Defines multiple HTML elements or Angular components within a container as drop targets.
- **Auto scroll**—Automatically scroll the nearest scrollable container when the dragged component is close to the edges.
- **Drag Hint**—Render a copy of the original element (or a custom element) as hint during dragging.
- **Drag Handle**—Allow dragging only through interaction with a specified `drag handle` while still using the parent element for drop detection.
- **Hold to drag**—Enable dragging after holding the drag element for a specific period of time.
- **Axis lock**—Force dragging only on the horizontal or vertical axis.
- **Minimum distance**—Start dragging only after the pointer covers a predefined distance.

## Support and Learning Resources

- [Drag and Drop Homepage](#)
- [Getting Started with the Kendo UI for Angular Utilities](#)
- [API Reference of the DragTarget](#)
- [API Reference of the DropTarget](#)
- [API Reference of the DragHandle](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Drag and Drop Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)

- Kendo UI for Angular Roadmap

# Getting Started with the Kendo UI for Angular Utilities

This guide provides the information you need to start using the Kendo UI for Angular Utilities—it includes instructions about the recommended installation approach, the code for running the project, and links to additional resources.

As of version [17.0.0](#), Angular makes [standalone component](#) enabled by default. If you use [NgModules](#), refer to these articles:

- [Using Kendo Angular Components with NgModules](#)
- [Using Hybrid Configuration](#)

After the completion of this guide, you will be able to achieve an end result as demonstrated in the following example.

EXAMPLE

VIEW SOURCE

⋮



```
import { Component } from "@angular/core";
import { KENDO_BUTTON } from "@progress/kendo-angular-buttons";
import { KENDO_UTILS } from "@progress/kendo-angular-utils";
import { arrowsMoveIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  standalone: true,
  imports: [KENDO_UTILS, KENDO_BUTTON],
  selector: "my-app",
  template: `
    <div class="example-wrapper">
      <button
        kendoButton
        kendoDragTarget
        themeColor="primary"
        [svgIcon]="moveIcon"
      >
    </div>
  `
})
```

```
        Drag Me!
      </button>
    </div>
  ,
})
export class AppComponent {
  public moveIcon: SVGIcon = arrowsMoveIcon;
}
```

Collapse Code ^

# Setting Up Your Angular Project

Before you start with the installation of any Kendo UI for Angular control, ensure that you have a running Angular project. The prerequisites to accomplish the installation of the components are always the same regardless of the Kendo UI for Angular package you want to use, and are fully described in the [section on setting up your Angular project](#).

## Installing the Components

The following command demonstrates an efficient, automated method for adding packages using the [Angular CLI](#) through the `ng-add` command. This approach saves time and effort by executing a series of commands in a single step, which otherwise need to be run individually. Refer to the [Manual Setup](#) for more details.

To add the Kendo UI for Angular Utils package:

1. Run the following command.

```
ng add @progress/kendo-angular-utils
```

SH

As a result, the `ng-add` command will perform the following actions:

- Add the `@progress/kendo-angular-utils` package as a dependency to the `package.json` file.

- Add all required **peer dependencies** to the `package.json` file.
- Register the **Kendo UI Default theme** in the `angular.json` file.
- Trigger `npm install` to install the theme and all peer packages that are added.

## 2. Import the required components and directives:

- To add all components from the **Utils** package, import the `KENDO_UTILS` utility array in your standalone component.

```
import { Component } from '@angular/core';
import { KENDO_UTILS } from '@progress/kendo-angular-utils';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_UTILS]
})
```

TS

- To add individual Utility components, import the corresponding utility arrays in your standalone component. See the list of [available utility arrays](#).

For example if you only need the Drag and Drop functionality, import `KENDO_DRAGANDDROP`.

```
import { Component } from '@angular/core';
import { KENDO_DRAGANDDROP } from '@progress/kendo-angular-
utils';

@Component({
  standalone: true,
  selector: 'my-app',
  imports: [KENDO_DRAGANDDROP]
})
```

TS

# Using the Components

1. After successfully installing the `Utils` package and importing the required components and directives, add the corresponding directives you need in the `app.component.html`. For example, if you need the `DragTarget` directive, add the following code:

```
<div kendoDragTarget>
    <span class="k-icon k-font-icon k-i-drag-and-drop"></span>
    Drag me!
</div>
```

HTML

2. Build and serve the application by running the following command in the root folder.

```
ng serve
```

SH

3. Point your browser to <http://localhost:4200> to see the Kendo UI for Angular `DragTarget` on the page.

## Activating Your License Key

As of December 2020, using any of the UI components from the Kendo UI for Angular library requires either a commercial license key or an active trial license key. If your application does not contain a Kendo UI license file, [activate your license key](#).

## Next Steps

- [Utilities Dependencies & Standalone Utilities](#)
- [Drag and Drop Overview](#)
- [Utilities API Documentation](#)

## Learning Resources

- Utilities Overview
- Get Started with Kendo UI for Angular
- Using Kendo UI for Angular with Angular CLI
- Styling Overview
- Theme Builder
- Customizing the Themes
- Activating Your License Key
- Video Courses

# Angular Utilities Overview

The Kendo UI for Angular Utilities deliver a set of helper tools that can be used with Angular components or regular HTML elements to enhance their performance and functionality. They currently include the Drag and Drop and Adaptive Settings functionalities.

The Utilities are built from the ground up and specifically for Angular, so that you get high-performance controls which integrate tightly with your application and with the rest of the Kendo UI for Angular components.



## Drag and Drop

A set of Angular directives providing drag and drop functionality.



## Adaptive Settings

A set of configuration settings that allow customizing the adaptive behavior of the Kendo UI for Angular components.

# Angular Utilities Example

[EXAMPLE](#)[VIEW SOURCE](#)

⋮



```
import { Component } from "@angular/core";
import { CommonModule } from "@angular/common";
import { KENDO_BUTTONS } from "@progress/kendo-angular-buttons";
import { KENDO_UTILS } from "@progress/kendo-angular-utils";
import { arrowsMoveIcon, SVGIcon } from "@progress/kendo-svg-icons";

@Component({
  selector: "my-app",
  standalone: true,
```

```
imports: [CommonModule, KENDO_BUTTONS, KENDO_UTILS],
template: `
<div class="example-wrapper">
  <div class="example-col">
    <h5>Drag the Button around</h5>
    <div class="container">
      @for (dropTarget of dropTargets; track dropTarget) {
        <div
          kendoDropTarget
          (onDragEnter)="handleDragEnter(dropTarget)"
          (onDrop)="handleDrop(dropTarget)"
          (onDragLeave)="handleDragLeave()"
          [ngClass]="{
            'my-drop': true,
            entered: enteredBox === dropTarget && enteredBox !==
currentBox
          }"
        >
          @if (currentBox === dropTarget) {
            <button
              kendoButton
              kendoDragTarget
              themeColor="primary"
              [svgIcon]="moveIcon"
            >
              Drag Me!
            </button>
          }
          {{ currentBox !== dropTarget ? "Drop Here" : "" }}
        </div>
      }
    </div>
  </div>
</div>
`,
styleUrls: ["./styles.css"],
})
export class AppComponent {
  public moveIcon: SVGIcon = arrowsMoveIcon;
  public dropTargets = ["A", "B"];
  public currentBox = "A";
  public enteredBox = "A";

  public handleDragEnter(id: string): void {
    this.enteredBox = id;
  }

  public handleDragLeave(): void {
    this.enteredBox = "";
  }
}
```

```
public handleDrop(id: string): void {
    this.currentBox = id;
}
```

Collanse Code ^

# Angular Utilities Key Features

The Kendo UI for Angular Utilities deliver a range of handy and developer-friendly features whose number and further development are not limited by the list in this section. The Kendo UI for Angular team constantly invests efforts to improve the performance and add more value to the existing Utilities library as well as develop new features and controls to it.

## Configuring a Drag Target

You can configure any HTML element or Angular component as a drag target by using the [DragTarget](#) directive for single elements or the extended [DragTargetContainer](#) directive for multiple elements within a container. This enables the user to change the position of the elements by dragging them.

## Configuring a Drop Target

You can configure any HTML element or Angular component to handle drag targets that are dropped on it by using the [DropTarget](#) directive for single elements or the extended [DropTargetContainer](#) directive for multiple elements within a container.

The directives emit a number of events when a valid drag target interacts with a drop target. This enables you to execute custom logic and fine tune the behavior.

## Auto Scroll and Axis Lock

The Drag and Drop allows you to automatically scroll the nearest scrollable parent element while also providing an option to restrict the drag either to the horizontal

or the vertical axis. Read more about the [auto scroll](#) and [axis lock](#) features.

## Drag Hint

The Drag and Drop utility allows you to dynamically create and customize an element that can be used for the purpose of dragging instead of the drag target element itself. [Read more about the drag hint element...](#)

## Adaptive Settings

The Adaptive Settings utility enables you to customize the adaptive behavior of the Kendo UI for Angular components by providing configurable options for the default adaptive breakpoints. [Read more about the adaptive settings configuration...](#)

## Trial Version of Kendo UI for Angular

The Kendo UI for Angular Utility controls are part of the Kendo UI for Angular library. Kendo UI for Angular offers a 30-day trial with a full-featured version of the library—no restrictions! What's more, you are eligible for full technical support during your trial period in case you have any questions. Sign up for a [free 30-day trial](#) of Kendo UI for Angular.

## Support Options

For any questions about the use of Kendo UI for Angular Utilities, or any of our other components, there are [several support options available](#):

- Kendo UI license holders and anyone in an active trial can take advantage of the Kendo UI for Angular outstanding customer support delivered by the actual developers who built the library. To submit a support ticket, use the [Kendo UI support system](#).
- [Kendo UI for Angular forums](#) are part of the free support you can get from the community and from the Kendo UI team on all kinds of general issues.

- Kendo UI for Angular feedback portal and [Kendo UI for Angular roadmap](#) provide information on the features in discussion and also the planned ones for release.
- Kendo UI for Angular uses [GitHub Issues](#) as its bug tracker and you can submit any related reports there. Also, check out the [closed list](#).
- Of course, the Kendo UI for Angular team is active on [StackOverflow](#) as well and you can find a wealth of questions and answers there.
- Need something unique that is tailor-made for your project? Progress offers its [Progress Services](#) group that can work with you to create any customized solution that you might need.

## Support and Learning Resources

- [Getting Started with the Kendo UI for Angular Utilities](#)
- [API Reference of the Utilities](#)
- [Getting Started with Kendo UI for Angular \(Online Guide\)](#)
- [Getting Started with Kendo UI for Angular \(Video Tutorial\)](#)
- [Video Courses](#)
- [Kendo UI for Angular Forum](#)
- [Before You Start: All Things Angular \(Telerik Blog Post\)](#)
- [Knowledge Base](#)

## Additional Resources

- [Angular Blogs](#)
- [Angular Videos](#)
- [Kendo UI for Angular Roadmap](#)