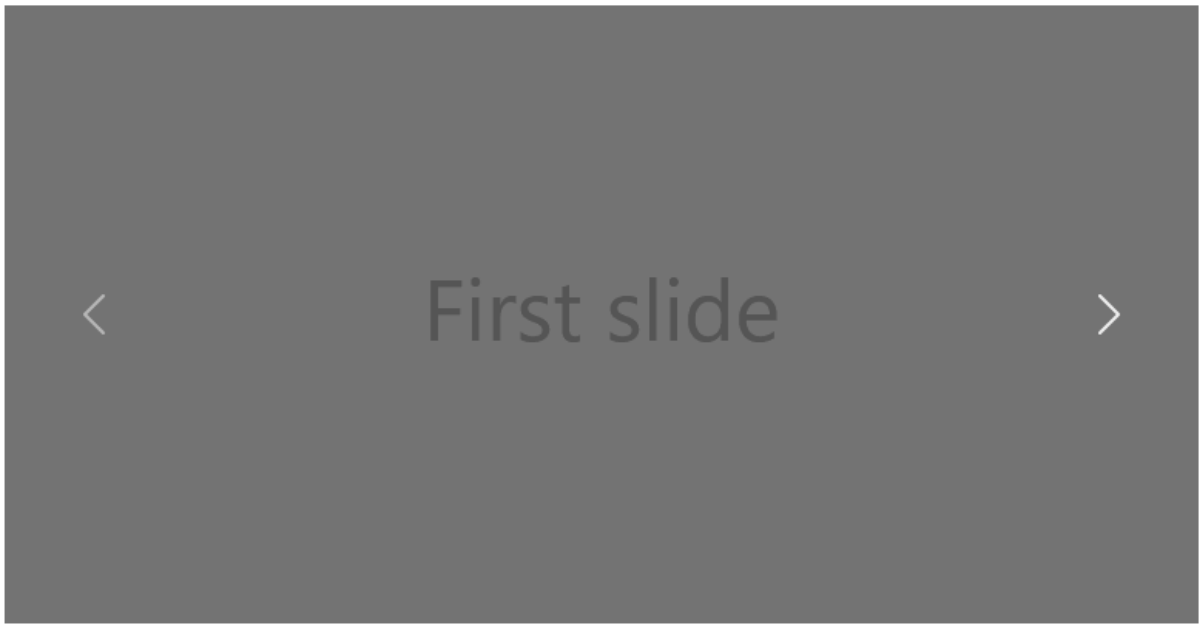
A **carousel** is a slideshow component used to display multiple items like images or slides of text in a cyclic manner, and it can be built using HTML, CSS, and JavaScript.



Basic Carousel

A simple carousel includes a set of images and allows users to cycle through them using navigation buttons (previous/next). Here's a basic example:

```
<div id="carouselExample" class="carousel slide">
  <div class="carousel-inner">
        <div class="carousel-item active">
        <img src="image1.jpg" class="d-block w-100" alt="Image 1">
        </div>
        <div class="carousel-item">
        <img src="image2.jpg" class="d-block w-100" alt="Image 2">
        </div>
```

```
        <div class="carousel-item">
        <img src="image3.jpg" class="d-block w-100" alt="Image 3">
        </div>
  </div>
  <button class="carousel-control-prev" type="button"
data-bs-target="#carouselExample" data-bs-slide="prev">
        <span class="carousel-control-prev-icon"
aria-hidden="true"></span>
        <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button"
data-bs-target="#carouselExample" data-bs-slide="next">
        <span class="carousel-control-next-icon"
aria-hidden="true"></span>
        <span class="visually-hidden">Next</span>
  </button>
</div>
```

carousel-inner: Wraps all the carousel items.
carousel-item: Each individual slide.
.active: Defines which slide is shown first.
carousel-control-prev and carousel-control-next: Buttons to move backward or forward.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Side-by-Side Images with Navigation</title>
  <!-- Bootstrap CSS -->
  <link
href="https://stackpath.bootstrapcdn.com/bootstrap/5.3.0-alpha1/css/boo
tstrap.min.css" rel="stylesheet">
  <style>
        .image-container {
        position: relative;
        }
        .carousel-btns {
        position: absolute;
        top: 50%;
        transform: translateY(-50%);
        z-index: 10;
        width: 100%;
        }
        .prev-btn {
        left: 0;
        }
        .next-btn {
        right: 0;
        }
  </style>
</head>
<body>

  <div class="container mt-4">
        <div class="row">
```

```
        <div class="col-md-4">
        <div class="image-container">
        <img id="image1" src="1.jpg" class="img-fluid" alt="Image 1">
        <div class="carousel-btns">
        <button class="btn btn-primary prev-btn"
onclick="previousImage()">←</button>
        <button class="btn btn-primary next-btn"
onclick="nextImage()">→</button>
        </div>
        </div>
        </div>
        <div class="col-md-4">
        <div class="image-container">
        <img id="image2" src="2.jpg" class="img-fluid" alt="Image 2">
        </div>
        </div>
        <div class="col-md-4">
        <div class="image-container">
        <img id="image3" src="1.jpg" class="img-fluid" alt="Image 3">
        </div>
        </div>
        </div>
  </div>

  <!-- Bootstrap JS and dependencies -->
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popp
er.min.js"></script>
```

```
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/5.3.0-alpha1/js/bootst
rap.min.js"></script>

  <script>
        let currentImageIndex = 0;
        const images = ['image1', 'image2', 'image3'];

        function updateImages() {
        // Hide all images first
        images.forEach((id, index) => {
        document.getElementById(id).style.display = 'none';
        });

        // Show the current image

document.getElementById(images[currentImageIndex]).style.display =
'block';
        }

        function previousImage() {
        currentImageIndex = (currentImageIndex - 1 + images.length) %
images.length;
        updateImages();
        }

        function nextImage() {
        currentImageIndex = (currentImageIndex + 1) % images.length;
        updateImages();
        }
```

```
        // Initialize
        updateImages();
  </script>

</body>
</html>
```

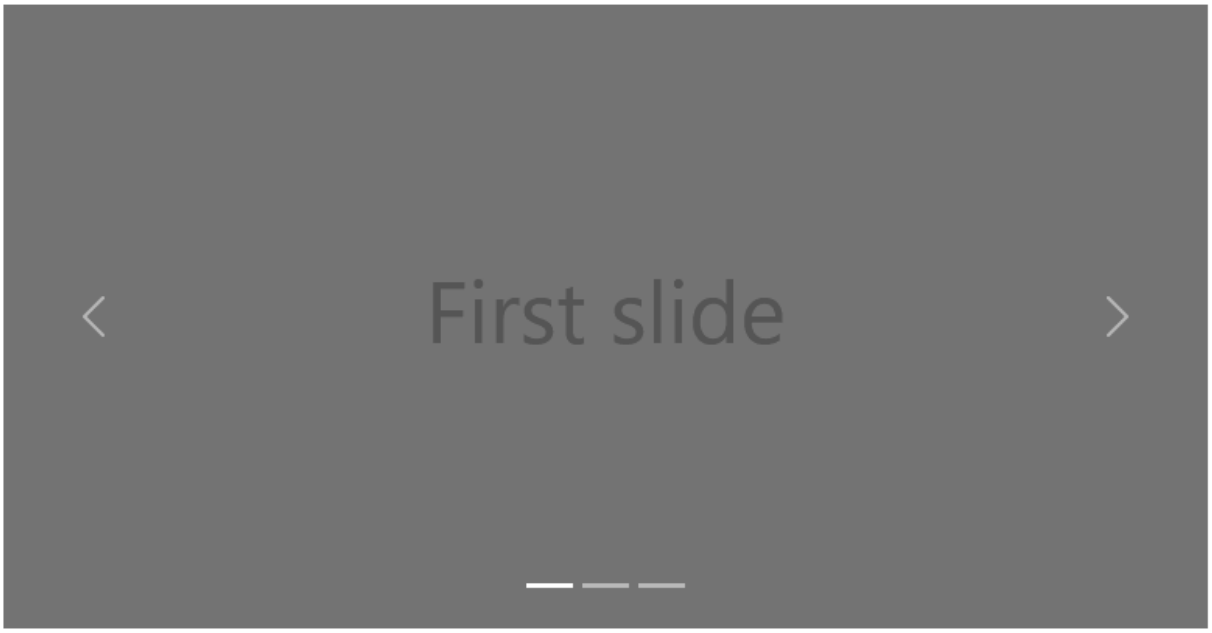
Indicators

You can add **indicators** (small circles or dots) below the carousel to let users directly jump to a particular slide.

```
<div id="carouselExampleIndicators" class="carousel slide">
  <div class="carousel-indicators">
        <button type="button"
data-bs-target="#carouselExampleIndicators" data-bs-slide-to="0"
class="active"></button>
        <button type="button"
data-bs-target="#carouselExampleIndicators"
data-bs-slide-to="1"></button>
        <button type="button"
data-bs-target="#carouselExampleIndicators"
data-bs-slide-to="2"></button>
  </div>
  <div class="carousel-inner">
        <div class="carousel-item active">
        <img src="image1.jpg" class="d-block w-100" alt="Image 1">
        </div>
        <div class="carousel-item">
        <img src="image2.jpg" class="d-block w-100" alt="Image 2">
```

```
        </div>
        <div class="carousel-item">
        <img src="image3.jpg" class="d-block w-100" alt="Image 3">
        </div>
  </div>
  <button class="carousel-control-prev" type="button"
data-bs-target="#carouselExampleIndicators" data-bs-slide="prev">
        <span class="carousel-control-prev-icon"
aria-hidden="true"></span>
  </button>
  <button class="carousel-control-next" type="button"
data-bs-target="#carouselExampleIndicators" data-bs-slide="next">
        <span class="carousel-control-next-icon"
aria-hidden="true"></span>
  </button>
</div>
```

- **carousel-indicators**: Contains the buttons to switch between slides.
- **data-bs-slide-to**: Specifies which slide to go to.



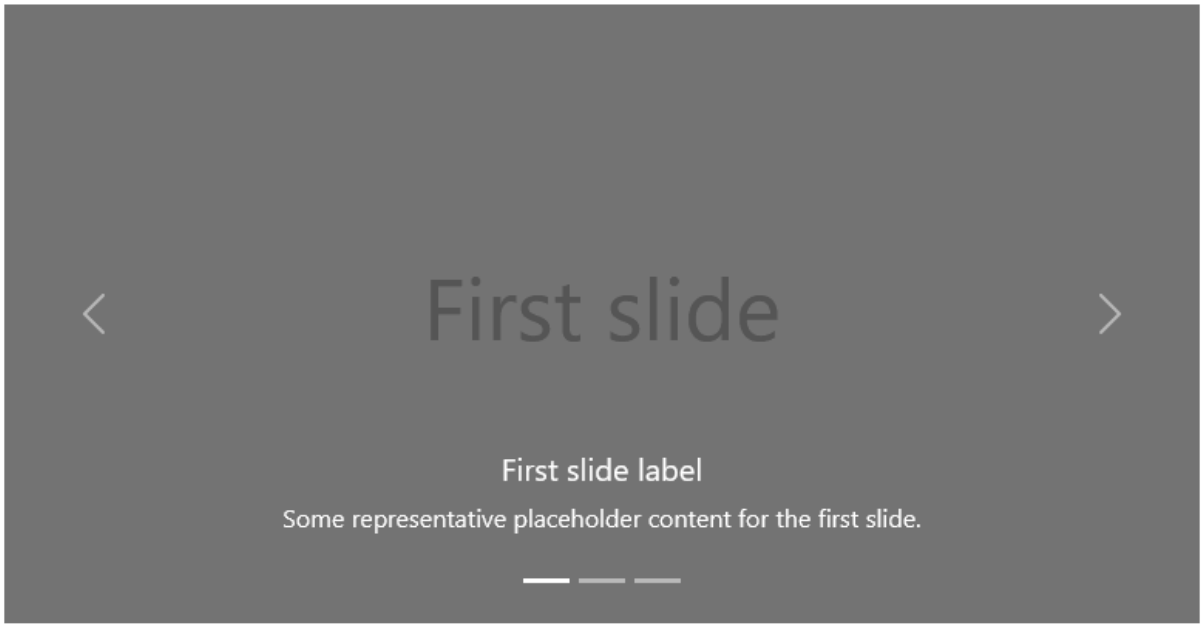
Captions

Carousels can also display **captions** (titles and descriptions) for each slide.

```
<div id="carouselExampleCaptions" class="carousel slide">
  <div class="carousel-inner">
        <div class="carousel-item active">
        <img src="image1.jpg" class="d-block w-100" alt="Image 1">
        <div class="carousel-caption d-none d-md-block">
        <h5>First slide label</h5>
        <p>Some representative placeholder content for the first slide.</p>
        </div>
        </div>
        <div class="carousel-item">
        <img src="image2.jpg" class="d-block w-100" alt="Image 2">
```

```
        <div class="carousel-caption d-none d-md-block">
        <h5>Second slide label</h5>
        <p>Some representative placeholder content for the second
slide.</p>
        </div>
        </div>
  </div>
</div>
```

- **carousel-caption**: Contains the title and description for each slide.
- **d-none d-md-block**: Hides captions on smaller screens (mobile devices) and displays them on medium-sized screens and above.

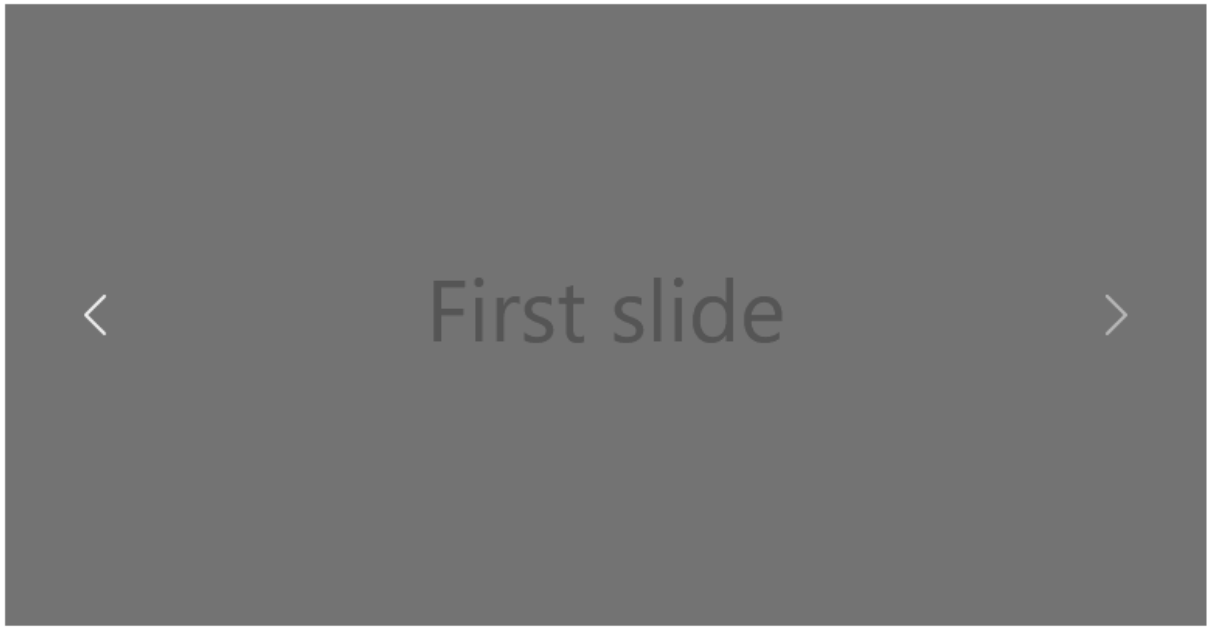


Crossfade Transition

To make the transition between slides a smooth fade effect rather than sliding, you can add the class .carousel-fade.

```
<div id="carouselExampleFade" class="carousel slide carousel-fade">
  <div class="carousel-inner">
        <div class="carousel-item active">
        <img src="image1.jpg" class="d-block w-100" alt="Image 1">
        </div>
        <div class="carousel-item">
        <img src="image2.jpg" class="d-block w-100" alt="Image 2">
        </div>
  </div>
</div>
```

- **carousel-fade**: Enables crossfade transition instead of the default sliding transition.



Typehead using Bootstrap:

To implement a typeahead (autocomplete) feature using Bootstrap, you can combine it with the **Bootstrap 5** framework and **jQuery** or **Bootstrap's own utility classes**. However, Bootstrap itself doesn't include a specific "typeahead" component out of the box, so you can use **Bootstrap's form components** along with JavaScript libraries like **Typeahead.js** (from Twitter's Bootstrap team) or implement it using Bootstrap's input group and JavaScript.

Here's a basic implementation using **Bootstrap 5** and **Typeahead.js** to create a typeahead feature.

1. Basic HTML Structure with Bootstrap 5

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>Typeahead using Bootstrap</title>

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/boots
trap.min.css" rel="stylesheet">

    <script
src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

```
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap5-typeahead@0.3.0/dist/boots
trap5-typeahead.bundle.min.js"></script>
</head>
<body>
    <div class="container my-5">
    <h2>Typeahead Input</h2>
    <div class="form-group">
    <label for="search">Search:</label>
    <input type="text" class="form-control" id="search"
placeholder="Start typing...">
    </div>
    </div>

    <script>
    $(document).ready(function(){
    var data = ["Apple", "Banana", "Cherry", "Grape", "Lemon",
"Orange", "Pineapple", "Strawberry"];

    $('#search').typeahead({
        source: data
```

```
    });
    });
    </script>
</body>
</html>
```

Explanation:

- **HTML**: A simple form with an input field is created inside a Bootstrap container.
- **Bootstrap CSS**: We include Bootstrap's CSS via a CDN for styling the form and input.
- **jQuery**: We include jQuery for handling DOM manipulation.
- **Typeahead.js**: This JavaScript library is included via a CDN and used to add the typeahead functionality to the input field.
- **JavaScript**: The data array is a list of options to be shown as suggestions. When the user starts typing in the input field, Typeahead will match the input against this list.

Typehead Functionalities:

Typeahead (also known as Autocomplete) is a powerful feature that suggests possible matches for a user's input based on what they type. The functionality of typeahead can be customized to provide a more user-friendly experience. Below are the key features and functionalities of Typeahead, including options and customization:

Key Typeahead Functionalities

1. **Basic Autocomplete**:
    - **Search Suggestions**: As the user types in the input field, the typeahead function suggests possible matches from a predefined list of data.
    - **Filtering Data**: The list of suggestions is filtered based on the user's input and is updated in real-time.

2. Source:

- The **source** property is the data that is used for autocompletion. It can be a simple static array or a dynamic list fetched from an API.

Example:

```
var data = ["Apple", "Banana", "Cherry", "Date", "Grape"];
```

3. Min Length:

- **Minimum Characters**: You can specify a minimum number of characters the user needs to type before suggestions appear. This helps avoid unnecessary API calls or excessive filtering for small inputs.

Example:

```
$('#input').typeahead({
    minLength: 2, // Only start showing results after 2 characters
    source: data
});
```

4. Highlighting:

- Typeahead often highlights the matching portion of the suggestions, providing visual feedback about which part of the suggestion matched the user's input.

Example:

```
$('#input').typeahead({

    highlight: true, // Highlight matched portion

    source: data

});
```

5.Remote Data / AJAX:

- Instead of using a static list of items, you can configure typeahead to fetch suggestions dynamically from a server (via an API or remote data source).

Example:

```
$('#input').typeahead({

    source: function(query, process) {

    return $.get('/api/suggestions', { query: query }, function(data) {

    return process(data);

    });
```

```
    }
});
```

6. Item Template:

- You can customize how each item in the suggestion list is rendered. This could be useful for adding custom HTML or additional information about the items.

Example:

```
$('#input').typeahead({
    source: data,
    item: function(item) {
    return '<div class="item">' + item + '</div>';
    }
});
```

7. **Multiple Datasets**:

- You can include multiple datasets for autocompletion, allowing the user to choose between different categories or types of suggestions.

Example:

```
var fruits = ["Apple", "Banana", "Cherry"];
var vegetables = ["Carrot", "Broccoli", "Spinach"];
```

```
$('#input').typeahead({
    source: [fruits, vegetables]
});
```

8. **Selectable Item**:

- When the user selects a suggestion from the dropdown, you can configure what happens next, like setting the selected value to the input field or triggering another action.

Example:

```
$('#input').typeahead({
    source: data
}).on('typeahead:selected', function(event, selectedItem) {
    console.log('You selected: ' + selectedItem);
});
```

9. **Template for Suggestions**:

- You can provide custom HTML templates for displaying the suggestions, which is helpful if you want to show more information (e.g., images, descriptions) in the dropdown.

Example:

```
$('#input').typeahead({
    source: data,
    template: function(item) {
    return '<div><strong>' + item + '</strong> - Description</div>';
    }
});
```

10. **Pagination**:

- If your dataset is large, you can implement pagination to load more items as the user scrolls through the suggestions, making the user experience smoother.

Example:

```
$('#input').typeahead({
    source: function(query, process) {
    return $.get('/api/suggestions', { query: query }, function(data) {
    return process(data.items);
    });
    },
    pagination: true
});
```

11. **Custom Datasets with Separate Views**:

- You can manage multiple datasets and assign each dataset a different view or rendering style. This is useful when you want to display results in different categories.

Example:

```
var fruits = ['Apple', 'Banana', 'Orange'];

var vegetables = ['Carrot', 'Broccoli', 'Spinach'];


$('#input').typeahead({
    datasets: [
    {
    name: 'fruits',
    source: fruits
    },
    {
    name: 'vegetables',
    source: vegetables
    }
    ]
});
```

12. **Custom Dropdown Styling**:

- You can also customize the dropdown's appearance, such as its width, borders, or even how the items are arranged.