UNIT-IV

In Bootstrap, the **Programmatic API** provides a way to initialize and interact with various Bootstrap components (like modals, dropdowns, popovers, etc.) through JavaScript. This is useful when you need to trigger or configure components dynamically, without relying solely on the data attributes in HTML. Let’s go through the key concepts and methods mentioned in your description with examples.

1. Basic Initialization with Constructor

Bootstrap plugins can be initialized with an optional configuration object. When you create a new instance of a plugin (like Modal, Popover, Dropdown), you can pass either no configuration (to use the defaults) or a custom configuration object.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Bootstrap Modal Example</title>
  <!-- Link to Bootstrap CSS (Latest Bootstrap 5 CDN version) -->
                                                              <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/boots
trap.min.css" rel="stylesheet">
</head>
<body>
```

```
  <!-- Button to trigger modal -->
  <button type="button" class="btn btn-primary" data-bs-toggle="modal"
data-bs-target="#myModal">
        Launch modal
  </button>

  <!-- Modal HTML -->
     <div class="modal" id="myModal" tabindex="-1"
aria-labelledby="myModalLabel" aria-hidden="true">
        <div class="modal-dialog">
        <div class="modal-content">
        <div class="modal-header">
        <h5 class="modal-title" id="myModalLabel">Modal
title</h5>
        <button type="button" class="btn-close"
data-bs-dismiss="modal" aria-label="Close"></button>
        </div>
        <div class="modal-body">
        Content goes here.
        </div>
        <div class="modal-footer">
        <button type="button" class="btn btn-secondary"
data-bs-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save
changes</button>
        </div>
        </div>
        </div>
  </div>
  <!-- JavaScript to initialize the modal -->
  <script>
       document.addEventListener('DOMContentLoaded', function (
) {
       // Get the modal element
       const myModalEl = document.querySelector('#myModal');

       // Initialize the modal with custom configuration (e.g., disable
keyboard interaction)
       const configObject = { keyboard: false }; // Disables closing the
modal with the Escape key
       const    modal1    =    new    bootstrap.Modal(myModalEl,
configObject);

       // Show the modal immediately after page load
       modal1.show();
       });
  </script>

   <!-- Link to Bootstrap JS (Latest Bootstrap 5 CDN version) and
Popper.js (required for Bootstrap) -->
                                                              <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popp
er.min.js"></script>
                                                              <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstra
p.bundle.min.js"></script>

</body>
```

```
</html>
```

2. Getting an Instance

You can retrieve an existing instance of a plugin that has been initialized using the getInstance method. If the plugin was not initialized, this method will return null.

Example:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Bootstrap Popover Example</title>

  <!-- Link to Bootstrap CSS (Latest Bootstrap 5 CDN version) -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

  <!-- Optional: You can add Popper.js separately but it's included in Bootstrap's bundle JS -->
</head>

<body>

  <!-- Popover HTML -->
  <button type="button" class="btn btn-secondary" id="myPopover"
data-bs-toggle="popover"              title="Popover              title"
data-bs-content="Popover content">
       Hover me
  </button>

  <!-- JavaScript to initialize popover -->
  <script>
       // Ensure that the DOM is fully loaded
       document.addEventListener('DOMContentLoaded', function () {
       const myPopoverEl = document.querySelector('#myPopover');

       // Initialize popover
       const popover = new bootstrap.Popover(myPopoverEl);

       // You can also programmatically show the popover if needed
       // popover.show(); // Uncomment this line if you want to show the
popover on page load

       // Retrieve the existing popover instance
       const instance = bootstrap.Popover.getInstance(myPopoverEl);

       if (instance) {
       console.log('Popover instance:', instance);
       } else {
       console.log('Popover instance not found.');
       }
       });
  </script>
```

```
<!-- Link to Bootstrap JS (Latest Bootstrap 5 CDN version) and Popper.js (required for Bootstrap) -->
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>
```

A modal is a dialog box or pop-up window that is typically used for user interaction, such as form submissions, alerts, or notifications. Bootstrap provides a modal plugin to easily integrate these pop-ups.

- **Positioning**: Modals are positioned over the rest of the content using position: fixed and z-index to overlay the modal on the screen.
- **Backdrop**: Clicking outside the modal (on the backdrop) typically closes it by default.
- **Single Modal**: Bootstrap supports only one modal at a time on the page.
- **HTML5 Autocomplete**: The autofocus attribute doesn't work with Bootstrap modals. A custom JavaScript approach is used to focus on specific elements when the modal is shown.

2. Basic Modal Example

The simplest modal example includes a title, content body, and footer with action buttons like "Close" and "Save changes". Here's a sample code:

```
<button type="button" class="btn btn-primary" data-bs-toggle="modal"
data-bs-target="#exampleModal">
  Launch demo modal
</button>

<div class="modal fade" id="exampleModal" tabindex="-1"
aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
        <div class="modal-content">
        <div class="modal-header">
        <h1 class="modal-title fs-5" id="exampleModalLabel">Modal
title</h1>
        <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
        </div>
        <div class="modal-body">
        Modal body content goes here.
        </div>
        <div class="modal-footer">
        <button type="button" class="btn btn-secondary"
data-bs-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save
changes</button>
        </div>
        </div>
  </div>
</div>
```

Explanation:

- **Button**: The button triggers the modal with data-bs-toggle="modal" and the target data-bs-target="#exampleModal".
- **Modal**: The modal is defined inside a <div class="modal"> element. It is hidden initially and can be shown when triggered.

Static Backdrop

By default, clicking outside the modal will close it. You can prevent this behavior by using a **static backdrop**. This is useful when you want to ensure that the modal is only closed by clicking a close button.

```
<button type="button" class="btn btn-primary" data-bs-toggle="modal"
data-bs-target="#staticBackdrop">

  Launch static backdrop modal

</button>



<div class="modal fade" id="staticBackdrop" data-bs-backdrop="static"
data-bs-keyboard="false" tabindex="-1"
aria-labelledby="staticBackdropLabel" aria-hidden="true">

  <div class="modal-dialog">

    <div class="modal-content">

      <div class="modal-header">
```

```
        <h1 class="modal-title fs-5" id="staticBackdropLabel">Modal title</h1>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        Content here.
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Understood</button>
      </div>
    </div>
  </div>
</div>
```

Scrolling Long Content

If the modal contains too much content for the user's viewport, the modal body will scroll independently of the page. You can enable this by adding .modal-dialog-scrollable to the modal's dialog container.

```
<div class="modal-dialog modal-dialog-scrollable">

  <div class="modal-content">

    <div class="modal-header">

      <h1 class="modal-title fs-5">Scrollable Modal</h1>

        <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>

    </div>

    <div class="modal-body">

      <!-- Long content goes here -->

      <p>Very long content that requires scrolling...</p>

    </div>

    <div class="modal-footer">

      <button type="button" class="btn btn-secondary"
data-bs-dismiss="modal">Close</button>

      <button type="button" class="btn btn-primary">Save
changes</button>

    </div>

  </div>
```

```
</div>
```

Vertically Centered Modal

To center the modal vertically in the viewport, add the .modal-dialog-centered class to the modal dialog.

```
<div class="modal-dialog modal-dialog-centered">

  <!-- Modal content here -->

</div>
```

- This centers the modal on the page, making it more visually appealing and accessible.

6. Tooltips and Popovers in Modals

You can include tooltips and popovers inside the modal body. When the modal is closed, any tooltips or popovers inside it are also automatically dismissed.

```
<div class="modal-body">

        <button class="btn btn-secondary" data-bs-toggle="popover"
title="Popover title" data-bs-content="Popover content is here.">Show
popover</button>

     <a href="#" data-bs-toggle="tooltip" title="Tooltip text">Tooltip
link</a>

</div>
```

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Modal with Popover and Tooltip</title>
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>


  <!-- Button trigger modal -->
  <button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#exampleModal">
    Launch Modal
  </button>
  <!-- Modal -->
      <div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="modal-header">
            <h1 class="modal-title fs-5" id="exampleModalLabel">Modal with Popover and Tooltip</h1>
          <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
        </div>
        <div class="modal-body">
          <!-- Button for popover -->
              <button class="btn btn-secondary" data-bs-toggle="popover" title="Popover title" data-bs-content="Popover content is here.">Show popover</button>
          <!-- Link for tooltip -->
            <a href="#" data-bs-toggle="tooltip" title="Tooltip text">Tooltip link</a>
        </div>
        <div class="modal-footer">
```

```
                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
              <button type="button" class="btn btn-primary">Save changes</button>
        </div>
      </div>
    </div>
  </div>

  <!-- Bootstrap JS, Popper.js, and jQuery -->
  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"></script>
  <script>
    // Enable tooltips and popovers
    var tooltipTriggerList = [].slice.call(document.querySelectorAll('[data-bs-toggle="tooltip"]'))
    tooltipTriggerList.map(function (tooltipTriggerEl) {
      return new bootstrap.Tooltip(tooltipTriggerEl)
    });


var popoverTriggerList = [].slice.call(document.querySelectorAll('[data-bs-toggle="popover"]'))
    popoverTriggerList.map(function (popoverTriggerEl) {
      return new bootstrap.Popover(popoverTriggerEl)
    });
  </script>


</body>
</html>
```

Dynamic Content in Modals

Sometimes, you may need to change the content of a modal based on user interaction. You can achieve this by using JavaScript and passing dynamic content into the modal using the **data-bs-* attributes.**

```
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#exampleModal" data-bs-whatever="@mdo">Open modal for @mdo</button>


<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
  <div class="modal-content">
   <div class="modal-header">
        <h1 class="modal-title fs-5" id="exampleModalLabel">New message</h1>
      <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
   </div>
   <div class="modal-body">
    <form>
     <div class="mb-3">
      <label for="recipient-name" class="col-form-label">Recipient:</label>
      <input type="text" class="form-control" id="recipient-name">
     </div>
    </form>
   </div>
   <div class="modal-footer">
      <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
      <button type="button" class="btn btn-primary">Send message</button>
      </div>
    </div>
  </div>
</div>

<script>
  const exampleModal = document.getElementById('exampleModal');
  exampleModal.addEventListener('show.bs.modal', event => {
    const button = event.relatedTarget;
    const recipient = button.getAttribute('data-bs-whatever');
    const modalTitle = exampleModal.querySelector('.modal-title');
    const modalBodyInput = exampleModal.querySelector('.modal-body input');

    modalTitle.textContent = `New message to ${recipient}`;
    modalBodyInput.value = recipient;
  });
</script>
```

Here, the modal dynamically updates its content based on which button was clicked using the data-bs-whatever attribute.

Bootstrap ScrollSpy: An Overview

ScrollSpy is a powerful feature in Bootstrap that automatically highlights navigation links as you scroll through a page. It works by detecting which part of the page is currently in view and adding an .active class to the corresponding navigation link. This is particularly useful when you have a navigation bar or list of links that should reflect the current section of the page as the user scrolls.

How ScrollSpy Works

1. **Target Elements:**
   - o It works with navigation components, list groups, and even simple anchor links. For example, when you scroll down the page, it will highlight the relevant section in the navbar.
2. **Initialization:**
   - o ScrollSpy requires two key elements:
     1. A scrollable container (can be <body> or a custom container with a defined height and overflow-y: scroll).
     2. A target, usually a navigation element or list group.
3. **Activation:**

- It works by adding the data-bs-spy="scroll" attribute to the scrollable container and specifying the target element with data-bs-target. For each anchor link within the navigation, the href must correspond to an element with a matching id. When that element scrolls into view, the anchor link will receive the .active class.

Examples

1. Basic Example with Navbar

In this example, as you scroll through different headings, the active link in the navbar changes to match the section that's in view.

```
<!doctype html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Scrollspy Navbar Example</title>

    <!-- Bootstrap CSS -->

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

  </head>

  <body>
  <!-- Navbar -->

    <nav id="navbar-example2" class="navbar bg-body-tertiary px-3
mb-3">

   <a class="navbar-brand" href="#">Navbar</a>

   <ul class="nav nav-pills">

    <li class="nav-item">

     <a class="nav-link" href="#scrollspyHeading1">First</a>

    </li>

    <li class="nav-item">

     <a class="nav-link" href="#scrollspyHeading2">Second</a>

    </li>

    <li class="nav-item dropdown">

      <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown"
href="#">Dropdown</a>

     <ul class="dropdown-menu">

                                   <li><a    class="dropdown-item"
href="#scrollspyHeading3">Third</a></li>

                                   <li><a    class="dropdown-item"
href="#scrollspyHeading4">Fourth</a></li>

                                   <li><a    class="dropdown-item"
href="#scrollspyHeading5">Fifth</a></li>
```

```
      </ul>
    </li>
  </ul>
</nav>


<!-- Scrollspy content -->
        <div  data-bs-spy="scroll"  data-bs-target="#navbar-example2"
data-bs-smooth-scroll="true" class="scrollspy-example bg-body-tertiary
p-3 rounded-2" tabindex="0">
  <h4 id="scrollspyHeading1">First heading</h4>
  <p>Some content...</p>
  <h4 id="scrollspyHeading2">Second heading</h4>
  <p>Some content...</p>
  <h4 id="scrollspyHeading3">Third heading</h4>
  <p>Some content...</p>
  <h4 id="scrollspyHeading4">Fourth heading</h4>
  <p>Some content...</p>
  <h4 id="scrollspyHeading5">Fifth heading</h4>
  <p>Some content...</p>
</div>
```

```
    <!-- Bootstrap JS, Popper.js -->
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"></script>
  </body>
</html>
```

Nested Nav Example

If you have nested navigation links, ScrollSpy can automatically highlight the parent link if any of its children is active.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Scrollspy Navbar Example</title>
    <!-- Bootstrap CSS -->
```

```
                                                                <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/boots
trap.min.css" rel="stylesheet">
  </head>
  <body>
   <div class="row">
  <div class="col-4">
            <nav    id="navbar-example3"    class="h-100    flex-column
align-items-stretch pe-4 border-end">
      <nav class="nav nav-pills flex-column">
        <a class="nav-link" href="#item-1">Item 1</a>
        <nav class="nav nav-pills flex-column">
          <a class="nav-link ms-3 my-1" href="#item-1-1">Item 1-1</a>
          <a class="nav-link ms-3 my-1" href="#item-1-2">Item 1-2</a>
        </nav>
        <a class="nav-link" href="#item-2">Item 2</a>
        <a class="nav-link" href="#item-3">Item 3</a>
        <nav class="nav nav-pills flex-column">
          <a class="nav-link ms-3 my-1" href="#item-3-1">Item 3-1</a>
          <a class="nav-link ms-3 my-1" href="#item-3-2">Item 3-2</a>
        </nav>
      </nav>
    </nav>
  </div>

  <div class="col-8">
            <div data-bs-spy="scroll" data-bs-target="#navbar-example3"
data-bs-smooth-scroll="true" class="scrollspy-example-2"
tabindex="0">
      <div id="item-1">
        <h4>Item 1</h4>
        <p>Some content...</p>
      </div>
      <div id="item-1-1">
        <h5>Item 1-1</h5>
        <p>Some content...</p>
      </div>
      <div id="item-1-2">
        <h5>Item 1-2</h5>
        <p>Some content...</p>
      </div>
      <div id="item-2">
        <h4>Item 2</h4>
        <p>Some content...</p>
      </div>
      <div id="item-3">
        <h4>Item 3</h4>
        <p>Some content...</p>
      </div>
      <div id="item-3-1">
        <h5>Item 3-1</h5>
        <p>Some content...</p>
      </div>
      <div id="item-3-2">
        <h5>Item 3-2</h5>
        <p>Some content...</p>
      </div>
    </div>
  </div>
</div>
```

```
    <!-- Bootstrap JS, Popper.js -->
                                                                    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popp
er.min.js"></script>
                                                                    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstra
p.min.js"></script>
  </body>
</html>
```

List Group Example

You can also apply ScrollSpy to a list group. As you scroll, the active item in the list will be highlighted.

```
<div class="row">
  <div class="col-4">
    <div id="list-example" class="list-group">
                              <a  class="list-group-item  list-group-item-action"
href="#list-item-1">Item 1</a>
                              <a  class="list-group-item  list-group-item-action"
href="#list-item-2">Item 2</a>
```

```
                <a class="list-group-item list-group-item-action"
href="#list-item-3">Item 3</a>

                <a class="list-group-item list-group-item-action"
href="#list-item-4">Item 4</a>

    </div>

  </div>

  <div class="col-8">

            <div data-bs-spy="scroll" data-bs-target="#list-example"
data-bs-smooth-scroll="true" class="scrollspy-example" tabindex="0">

      <h4 id="list-item-1">Item 1</h4>

      <p>Some content...</p>

      <h4 id="list-item-2">Item 2</h4>

      <p>Some content...</p>

      <h4 id="list-item-3">Item 3</h4>

      <p>Some content...</p>

      <h4 id="list-item-4">Item 4</h4>

      <p>Some content...</p>

    </div>

  </div>

</div>
```

Simple Anchors Example

You can apply ScrollSpy to simple anchor links as well, not just navigation components or lists.

```
<div class="row">
  <div class="col-4">
        <div id="simple-list-example" class="d-flex flex-column gap-2 simple-list-example-scrollspy text-center">
      <a class="p-1 rounded" href="#simple-list-item-1">Item 1</a>
      <a class="p-1 rounded" href="#simple-list-item-2">Item 2</a>
        <a class="p-1 rounded" href="#simple-list-item-3">Item 3</a>
      <a class="p-1 rounded" href="#simple-list-item-4">Item 4</a>
      <a class="p-1 rounded" href="#simple-list-item-5">Item 5</a>
    </div>
  </div>
  <div class="col-8">
        <div data-bs-spy="scroll" data-bs-target="#simple-list-example" data-bs-offset="0" data-bs-smooth-scroll="true" class="scrollspy-example" tabindex="0">
      <h4 id="simple-list-item-1">Item 1</h4>
      <p>Some content...</p>
      <h4 id="simple-list-item-2">Item 2</h4>
      <p>Some content...</p>
      <h4 id="simple-list-item-3">Item 3</h4>

      <p>Some content...</p>

      <h4 id="simple-list-item-4">Item 4</h4>

      <p>Some content...</p>

      <h4 id="simple-list-item-5">Item 5</h4>

      <p>Some content...</p>

    </div>

  </div>

</div>
```

Popovers in Bootstrap provide a way to display additional content in a small overlay that can appear when a user interacts with an element. Popovers are similar to tooltips, but they can contain more content such as HTML, images, etc. Here's a detailed explanation with examples on how to use popovers effectively.

Steps to Use Popovers

1. **Include Dependencies:** Popovers rely on Popper.js for positioning. You can include the required files using Bootstrap's CDN. Make sure you use bootstrap.bundle.min.js to include both Bootstrap and Popper in one file.

2. **Trigger Popovers:** Popovers can be triggered in various ways, including hover, click, or focus. You can specify this behavior using the data-bs-trigger attribute.

3. **Set Content:** Popovers can contain a title and content. The title is set with data-bs-title, and the content with data-bs-content.
4. **Direction of Popovers:** You can specify the placement (top, right, bottom, or left) using data-bs-placement.
5. **Customizing Popovers:** You can also customize the appearance of the popovers using CSS variables or a custom class.

---

1. Basic Example:

This is a simple popover triggered by clicking a button. You can specify both the title and content of the popover using data-bs-title and data-bs-content.

```
<!-- Button to trigger popover -->

<button type="button" class="btn btn-lg btn-danger"

        data-bs-toggle="popover"

        data-bs-title="Popover title"

        data-bs-content="And here's some amazing content. It's very engaging. Right?">

  Click to toggle popover

</button>
```

```
<!-- Required JavaScript -->

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>

<script>

  // Initialize popovers

  const popoverTriggerList = document.querySelectorAll('[data-bs-toggle="popover"]');

  const popoverList = [...popoverTriggerList].map(popoverTriggerEl => new bootstrap.Popover(popoverTriggerEl));

</script>
```

Popover Directions:

You can position the popover on the top, right, bottom, or left of the trigger element. The direction can be controlled using the data-bs-placement attribute.

```
<!-- Popover on Top -->

<button type="button" class="btn btn-secondary"

        data-bs-toggle="popover"

        data-bs-placement="top"

        data-bs-content="Top popover">
```

```
  Popover on top
</button>

<!-- Popover on Right -->
<button type="button" class="btn btn-secondary"
        data-bs-toggle="popover"
        data-bs-placement="right"
        data-bs-content="Right popover">
  Popover on right
</button>

<!-- Popover on Bottom -->
<button type="button" class="btn btn-secondary"
        data-bs-toggle="popover"
        data-bs-placement="bottom"
        data-bs-content="Bottom popover">
  Popover on bottom
</button>
```

```
<!-- Popover on Left -->

<button type="button" class="btn btn-secondary"

        data-bs-toggle="popover"

        data-bs-placement="left"

        data-bs-content="Left popover">

  Popover on left

</button>
```

Custom Container for Popovers:

Sometimes, you may want the popover to be rendered inside a specific container instead of the default position. This can be useful when you have styles that may conflict with popover rendering.

```
<!-- Button with a custom container -->

<button type="button" class="btn btn-secondary"

        data-bs-toggle="popover"

        data-bs-container="body"

        data-bs-placement="top"

        data-bs-content="Popover inside body">

  Popover with custom container

</button>
```

```
<!-- Initialize Popover with custom container -->

<script>

  const popover = new bootstrap.Popover('.btn', {

    container: 'body'

  });

</script>
```

Popovers and tooltips are both components in Bootstrap that display extra information to the user in a floating box, but they serve slightly different purposes and have distinct features. Here's a comparison of the two:

1. Purpose & Content:

- **Popover:**
  - **Purpose**: Popovers are used to display more detailed content or a more complex overlay than tooltips. They can contain not only text but also other HTML elements like links, images, buttons, forms, etc.
  - **Content**: Popovers are more flexible in terms of content. They can contain large amounts of content, including headers, body text, and other HTML elements. They are useful when you need more than a simple hint or description.

- **Example Use**: Describing a feature with multiple lines of text, showing a form inside a modal, or displaying a detailed explanation.

· **Tooltip:**

- **Purpose**: Tooltips are used to provide short, concise information when a user hovers over an element. Typically, tooltips are simple text descriptions or hints.
- **Content**: Tooltips are generally limited to short text and are meant for brief, contextual information.
- **Example Use**: Displaying a description of a button or providing additional context on form input fields.

2. Triggering Mechanism:

- **Popover:**
  - Can be triggered by various actions like **click**, **focus**, or **hover**. However, popovers often require more intentional user interaction (like a click) to reveal their content.
  - **Common triggers**: data-bs-trigger="click", data-bs-trigger="focus", data-bs-trigger="hover".
- **Tooltip:**
  - Tooltips are typically triggered by **hover** or **focus** events, making them more passive than popovers.
  - **Common triggers**: data-bs-trigger="hover", data-bs-trigger="focus".

3. Appearance and Behavior:

- **Popover:**
  - Popovers have **headers** and **body content**. They can also have **buttons**, **links**, or other interactive elements.

    - Popovers are **larger** in size compared to tooltips and often need more space to display their content.
    - They are more **customizable** in terms of appearance, and you can define custom classes, widths, and more.
- **Tooltip:**
    - Tooltips are **smaller** and simpler in design, usually displaying just a short text.
    - They are more **compact** and show up near the element on hover or focus.

4. Direction and Positioning:

- **Popover:**

    - Popovers can be placed in **four directions**: top, right, bottom, or left. You can control this via the data-bs-placement attribute.
    - Since they can contain more content, their positioning might be more complex, and it can be customized further using options like container, offset, and boundary.

- **Tooltip:**

    - Tooltips also allow placement in **four directions** (top, right, bottom, left) with the data-bs-placement attribute.
    - Tooltips are **less customizable** in terms of positioning compared to popovers, as they are typically more straightforward to use and position.

Overview of Bootstrap Tooltips

Bootstrap's **Tooltip** component provides brief, contextual information when users hover over or focus on an element. Tooltips are highly useful for enhancing user experience by displaying additional hints or descriptions without taking up too much space on the screen.

The tooltip component uses **Popper.js** to handle positioning, which is why you need to include Popper.js or use the **bootstrap.bundle.min.js** (which includes Popper.js).

Here’s a breakdown of the **key points** and **examples** for using tooltips in Bootstrap:

Enabling Tooltips:

You need to **initialize** tooltips before using them on the page. You can initialize all tooltips on the page with the following JavaScript code:

```
const tooltipTriggerList = 
document.querySelectorAll('[data-bs-toggle="tooltip"]')

const tooltipList = [...tooltipTriggerList].map(tooltipTriggerEl => new 
bootstrap.Tooltip(tooltipTriggerEl))
```

Tooltips on Links:

You can use tooltips with **anchor links** by adding the data-bs-toggle="tooltip" attribute and setting the tooltip content with data-bs-title.

```
<p class="muted">

        Placeholder text to demonstrate some <a href="#"
data-bs-toggle="tooltip" data-bs-title="Default tooltip">inline links</a>
with tooltips.

   This is just filler content, giving you an idea of how <a href="#"
data-bs-toggle="tooltip" data-bs-title="Another tooltip">tooltips</a>
would look on your site.

</p>
```

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Tooltip Example</title>



  <!-- Bootstrap CSS -->

                                                                      <link
href="https://stackpath.bootstrapcdn.com/bootstrap/5.1.0/css/bootstrap.
min.css" rel="stylesheet">
```

```
  <!-- Popper.js for Bootstrap tooltips -->
                                                              <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/2.10.2/umd/popper.
min.js"></script>


  <!-- Bootstrap JS -->
                                                              <script
src="https://stackpath.bootstrapcdn.com/bootstrap/5.1.0/js/bootstrap.min
.js"></script>


  <script>
    // Initialize tooltips
    document.addEventListener('DOMContentLoaded', function () {
                                const        tooltipTriggerList        =
document.querySelectorAll('[data-bs-toggle="tooltip"]');
      tooltipTriggerList.forEach(tooltipTriggerEl => {
        new bootstrap.Tooltip(tooltipTriggerEl);
      });
    });
  </script>
</head>
```

```
<body>

  <div class="container mt-5">

    <p class="muted">

      Placeholder text to demonstrate some

            <a href="#" data-bs-toggle="tooltip" data-bs-title="Default tooltip">inline links</a> with tooltips.

      This is just filler content, giving you an idea of how

            <a href="#" data-bs-toggle="tooltip" data-bs-title="Another tooltip">tooltips</a> would look on your site.

    </p>

  </div>

</body>

</html>
```

Custom Tooltips (Added in v5.2.0):

In **Bootstrap 5.2**, you can customize the appearance of tooltips using **CSS variables**. You can define a custom class and override the tooltip styles with local variables.

Example of Custom Tooltip:

```
<button type="button" class="btn btn-secondary"

        data-bs-toggle="tooltip" data-bs-placement="top"
        data-bs-custom-class="custom-tooltip"
        data-bs-title="This top tooltip is themed via CSS variables.">
  Custom tooltip
</button>
.custom-tooltip {
  --bs-tooltip-bg: #5c3d7b; /* Custom background color */
  --bs-tooltip-color: #ffffff; /* Custom text color */
}

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Custom Tooltip Example</title>

  <!-- Bootstrap CSS -->
                                                              <link
href="https://stackpath.bootstrapcdn.com/bootstrap/5.2.0/css/bootstrap.
min.css" rel="stylesheet">
```

```
  <!-- Popper.js for Bootstrap tooltips -->
                                                              <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/2.10.2/umd/popper.
min.js"></script>

  <!-- Bootstrap JS -->
                                                              <script
src="https://stackpath.bootstrapcdn.com/bootstrap/5.2.0/js/bootstrap.min
.js"></script>

  <style>
    /* Custom Tooltip styles using CSS variables */
    .custom-tooltip {
      --bs-tooltip-bg: #5c3d7b; /* Custom background color */
      --bs-tooltip-color: #ffffff; /* Custom text color */
    }
  </style>

  <script>
    // Initialize tooltips
    document.addEventListener('DOMContentLoaded', function () {
                                                    const          tooltipTriggerList          =
document.querySelectorAll('[data-bs-toggle="tooltip"]');
      tooltipTriggerList.forEach(tooltipTriggerEl => {
        new bootstrap.Tooltip(tooltipTriggerEl);
      });
    });
  </script>
</head>
<body>
  <div class="container mt-5">
    <!-- Button to trigger tooltip with custom appearance -->
    <button type="button" class="btn btn-secondary"
            data-bs-toggle="tooltip" data-bs-placement="top"
            data-bs-custom-class="custom-tooltip"
            data-bs-title="This top tooltip is themed via CSS variables.">
      Custom tooltip
    </button>
  </div>
</body>
```

```
</html>
```

Tooltips with Directions:

Tooltips can be displayed in **four directions**: top, right, bottom, or left. You can change the position using the data-bs-placement attribute.

HTML Example:

```
<button type="button" class="btn btn-secondary"
        data-bs-toggle="tooltip" data-bs-placement="top"
        data-bs-title="Tooltip on top">
  Tooltip on top
</button>
<button type="button" class="btn btn-secondary"
        data-bs-toggle="tooltip" data-bs-placement="right"
        data-bs-title="Tooltip on right">
  Tooltip on right
</button>
<button type="button" class="btn btn-secondary"
        data-bs-toggle="tooltip" data-bs-placement="bottom"
        data-bs-title="Tooltip on bottom">
  Tooltip on bottom
```

```
</button>

<button type="button" class="btn btn-secondary"

         data-bs-toggle="tooltip" data-bs-placement="left"

        data-bs-title="Tooltip on left">

  Tooltip on left

</button>
```

Tooltips with HTML Content:

You can include **HTML content** in tooltips by setting the data-bs-html="true" attribute.

HTML Example:

```
<button type="button" class="btn btn-secondary"

        data-bs-toggle="tooltip" data-bs-html="true"

        data-bs-title="<em>Tooltip</em> <u>with</u> <b>HTML</b>">

  Tooltip with HTML

</button>
```