Overview

Things to know when using the tooltip plugin:

- Tooltips rely on the 3rd party library Popper for positioning. You must include [popper.min.js]({{< param "cdn.popper" >}}) before bootstrap.js or use `bootstrap.bundle.min.js` / `bootstrap.bundle.js` which contains Popper in order for tooltips to work!
- Tooltips are opt-in for performance reasons, so **you must initialize them yourself**.
- Tooltips with zero-length titles are never displayed.
- Specify `container: 'body'` to avoid rendering problems in more complex components (like our input groups, button groups, etc).
- Triggering tooltips on hidden elements will not work.
- Tooltips for `.disabled` or `disabled` elements must be triggered on a wrapper element.
- When triggered from hyperlinks that span multiple lines, tooltips will be centered. Use `white-space: nowrap;` on your `<a>`s to avoid this behavior.
- Tooltips must be hidden before their corresponding elements have been removed from the DOM.
- Tooltips can be triggered thanks to an element inside a shadow DOM.

{{< callout info >}} {{< partial "callout-info-sanitizer.md" >}} {{< /callout >}}

{{< callout info >}} {{< partial "callout-info-prefersreducedmotion.md" >}} {{< /callout >}}

Got all that? Great, let's see how they work with some examples.

Example: Enable tooltips everywhere

One way to initialize all tooltips on a page would be to select them by their `data-bs-toggle` attribute:

```
var tooltipTriggerList = [].slice.call(document.querySelectorAll('[data-bs-
toggle="tooltip"]'))
var tooltipList = tooltipTriggerList.map(function (tooltipTriggerEl) {
  return new bootstrap.Tooltip(tooltipTriggerEl)
})
```

Examples

Hover over the links below to see tooltips:

Placeholder text to demonstrate some inline links with tooltips. This is now just filler, no killer. Content placed here just to mimic the presence of real text. And all that just to give you an idea of how tooltips would look when used in real-world situations. So hopefully you've now seen how these tooltips on links can work in practice, once you use them on your own site or project.

Hover over the buttons below to see the four tooltips directions: top, right, bottom, and left. Directions are mirrored when using Bootstrap in RTL.

Tooltip on top | Tooltip on right | Tooltip on bottom | Tooltip on left | Tooltip with HTML

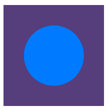
```
<button type="button" class="btn btn-secondary" data-bs-toggle="tooltip" data-bs-
placement="top" title="Tooltip on top">
  Tooltip on top
</button>
```

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

And with custom HTML added:

```
<button type="button" class="btn btn-secondary" data-bs-toggle="tooltip" data-bs-html="true" title="<em>Tooltip</em> <u>with</u> <b>HTML</b>">
  Tooltip with HTML
</button>
```

With an SVG:



Sass

Variables

{{< scss-docs name="tooltip-variables" file="scss/_variables.scss" >}}

Usage

The tooltip plugin generates content and markup on demand, and by default places tooltips after their trigger element.

Trigger the tooltip via JavaScript:

```
var exampleEl = document.getElementById('example')
var tooltip = new bootstrap.Tooltip(exampleEl, options)
```

{{< callout warning >}}

Overflow `auto` and `scroll`

Tooltip position attempts to automatically change when a **parent container** has `overflow: auto` or `overflow: scroll` like our `.table-responsive`, but still keeps the original placement's positioning. To resolve this, set the `boundary` option (for the flip modifier using the `popperConfig` option) to any HTMLElement to override the default value, `'clippingParents'`, such as `document.body`:

```
var exampleEl = document.getElementById('example')
var tooltip = new bootstrap.Tooltip(exampleEl, {
  boundary: document.body // or document.querySelector('#boundary')
})
```

{{< /callout >}}

Markup

The required markup for a tooltip is only a `data` attribute and `title` on the HTML element you wish to have a tooltip. The generated markup of a tooltip is rather simple, though it does require a position (by default, set to `top` by the plugin).

{{< callout warning >}}

Making tooltips work for keyboard and assistive technology users

You should only add tooltips to HTML elements that are traditionally keyboard-focusable and interactive (such as links or form controls). Although arbitrary HTML elements (such as `<span>`s) can be made focusable by adding the `tabindex="0"` attribute, this will add potentially annoying and confusing tab stops on non-interactive elements for keyboard users, and most assistive technologies currently do not announce the tooltip in this situation. Additionally, do not rely solely on `hover` as the trigger for your tooltip, as this will make your tooltips impossible to trigger for keyboard users. {{< /callout >}}

```
<!-- HTML to write -->
<a href="#" data-bs-toggle="tooltip" title="Some tooltip text!">Hover over me</a>

<!-- Generated markup by the plugin -->
<div class="tooltip bs-tooltip-top" role="tooltip">
  <div class="tooltip-arrow"></div>
  <div class="tooltip-inner">
    Some tooltip text!
  </div>
</div>
```

Disabled elements

Elements with the `disabled` attribute aren't interactive, meaning users cannot focus, hover, or click them to trigger a tooltip (or popover). As a workaround, you'll want to trigger the tooltip from a wrapper `<div>` or `<span>`, ideally made keyboard-focusable using `tabindex="0"`.

{{< example >}} Disabled button {{< /example >}}

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-bs-`, as in `data-bs-animation=""`. Make sure to change the case type of the option name from camelCase to kebab-case when passing the options via data attributes. For example, instead of using `data-bs-customClass="beautifier"`, use `data-bs-custom-class="beautifier"`.

{{< callout warning >}} Note that for security reasons the `sanitize`, `sanitizeFn`, and `allowList` options cannot be supplied using data attributes. {{< /callout >}}

Name	Type	Default	Description
`animation`	boolean	`true`	Apply a CSS fade transition to the tooltip
`container`	string | element | false	`false`	Appends the tooltip to a specific element. Example: `container: 'body'` . This option is particularly useful in that it allows you to position the tooltip in the flow of the document near the triggering element - which will prevent the tooltip from floating away from the triggering element during a window resize.
`delay`	number | object	`0`	Delay showing and hiding the tooltip (ms) - does not apply to manual trigger type If a number is supplied, delay is applied to both hide/show Object structure is: `delay: { "show": 500, "hide": 100 }`
`html`	boolean	`false`	Allow HTML in the tooltip. If true, HTML tags in the tooltip's `title` will be rendered in the tooltip. If false, `innerText` property will be used to insert content into the DOM. Use text if you're worried about XSS attacks.
`placement`	string | function	`'top'`	How to position the tooltip - auto | top | bottom | left | right. When `auto` is specified, it will dynamically reorient the tooltip. When a function is used to determine the placement, it is called with the tooltip DOM node as its first argument and the triggering element DOM node as its second. The `this` context is set to the tooltip instance.
`selector`	string | false	`false`	If a selector is provided, tooltip objects will be delegated to the specified targets. In practice, this is used to also apply tooltips to dynamically added DOM elements (`jQuery.on` support). See this and an informative example.

`template`	string	`'<div class="tooltip" role="tooltip"><div class="tooltip-arrow"></div><div class="tooltip-inner"></div></div>'`	Base HTML to use when creating the tooltip. The tooltip's `title` will be injected into the `.tooltip-inner`. `.tooltip-arrow` will become the tooltip's arrow. The outermost wrapper element should have the `.tooltip` class and `role="tooltip"`.
`title`	string | element | function	`''`	Default title value if `title` attribute isn't present. If a function is given, it will be called with its `this` reference set to the element that the tooltip is attached to.
`trigger`	string	`'hover focus'`	How tooltip is triggered - click | hover | focus | manual. You may pass multiple triggers; separate them with a space. `'manual'` indicates that the tooltip will be triggered programmatically via the `.show()`, `.hide()` and `.toggle()` methods; this value cannot be combined with any other trigger. `'hover'` on its own will result in tooltips that cannot be triggered via the keyboard, and should only be used if alternative methods for conveying the same information for keyboard users is present.
`fallbackPlacements`	array	`['top', 'right', 'bottom', 'left']`	Define fallback placements by providing a list of placements in array (in order of preference). For more information refer to Popper's behavior docs
`boundary`	string | element	`'clippingParents'`	Overflow constraint boundary of the tooltip (applies only to Popper's preventOverflow modifier). By default it's `'clippingParents'` and can accept an HTMLElement reference (via JavaScript only). For more information refer to Popper's detectOverflow docs.
`customClass`	string | function	`''`	Add classes to the tooltip when it is shown. Note that these classes will be

			added in addition to any classes specified in the template. To add multiple classes, separate them with spaces: `'class-1 class-2'`. You can also pass a function that should return a single string containing additional class names.
`sanitize`	boolean	`true`	Enable or disable the sanitization. If activated `'template'` and `'title'` options will be sanitized. See the }}">sanitizer section in our JavaScript documentation.
`allowList`	object	}}">Default value	Object which contains allowed attributes and tags
`sanitizeFn`	null | function	`null`	Here you can supply your own sanitize function. This can be useful if you prefer to use a dedicated library to perform sanitization.
`offset`	array | string | function	`[0, 0]`	Offset of the tooltip relative to its target. You can pass a string in data attributes with comma separated values like: `data-bs-offset="10,20"` When a function is used to determine the offset, it is called with an object containing the popper placement, the reference, and popper rects as its first argument. The triggering element DOM node is passed as the second argument. The function must return an array with two numbers: `[skidding, distance]`. For more information refer to Popper's offset docs.
`popperConfig`	null | object | function	`null`	To change Bootstrap's default Popper config, see Popper's configuration. When a function is used to create the Popper configuration, it's called with an object that contains the Bootstrap's default Popper configuration. It helps you use and merge the default with your own configuration. The function must return a configuration object for Popper.

{{< callout info >}}

Data attributes for individual tooltips

Options for individual tooltips can alternatively be specified through the use of data attributes, as explained above. {{< /callout >}}

Using function with `popperConfig`

```
var tooltip = new bootstrap.Tooltip(element, {
  popperConfig: function (defaultBsPopperConfig) {
    // var newPopperConfig = {...}
    // use defaultBsPopperConfig if needed...
    // return newPopperConfig
  }
})
```

Methods

{{< callout danger >}} {{< partial "callout-danger-async-methods.md" >}} {{< /callout >}}

show

Reveals an element's tooltip. **Returns to the caller before the tooltip has actually been shown** (i.e. before the `shown.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip. Tooltips with zero-length titles are never displayed.

```
tooltip.show()
```

hide

Hides an element's tooltip. **Returns to the caller before the tooltip has actually been hidden** (i.e. before the `hidden.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip.

```
tooltip.hide()
```

toggle

Toggles an element's tooltip. **Returns to the caller before the tooltip has actually been shown or hidden** (i.e. before the `shown.bs.tooltip` or `hidden.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip.

```
tooltip.toggle()
```

dispose

Hides and destroys an element's tooltip (Removes stored data on the DOM element). Tooltips that use delegation (which are created using the `selector` option) cannot be individually destroyed on descendant trigger elements.

```
tooltip.dispose()
```

enable

Gives an element's tooltip the ability to be shown. **Tooltips are enabled by default.**

```
tooltip.enable()
```

disable

Removes the ability for an element's tooltip to be shown. The tooltip will only be able to be shown if it is re-enabled.

```
tooltip.disable()
```

toggleEnabled

Toggles the ability for an element's tooltip to be shown or hidden.

```
tooltip.toggleEnabled()
```

update

Updates the position of an element's tooltip.

```
tooltip.update()
```

getInstance

Static method which allows you to get the tooltip instance associated with a DOM element

```
var exampleTriggerEl = document.getElementById('example')
var tooltip = bootstrap.Tooltip.getInstance(exampleTriggerEl) // Returns a Bootstrap
tooltip instance
```

getOrCreateInstance

Static method which allows you to get the tooltip instance associated with a DOM element, or create a new one in case it wasn't initialized

```
var exampleTriggerEl = document.getElementById('example')
var tooltip = bootstrap.Tooltip.getOrCreateInstance(exampleTriggerEl) // Returns a
Bootstrap tooltip instance
```

Events

Event type	Description
`show.bs.tooltip`	This event fires immediately when the `show` instance method is called.
`shown.bs.tooltip`	This event is fired when the tooltip has been made visible to the user (will wait for CSS transitions to complete).
`hide.bs.tooltip`	This event is fired immediately when the `hide` instance method has been called.
`hidden.bs.tooltip`	This event is fired when the tooltip has finished being hidden from the user (will

	wait for CSS transitions to complete).
`inserted.bs.tooltip`	This event is fired after the `show.bs.tooltip` event when the tooltip template has been added to the DOM.

```
var myTooltipEl = document.getElementById('myTooltip')
var tooltip = new bootstrap.Tooltip(myTooltipEl)

myTooltipEl.addEventListener('hidden.bs.tooltip', function () {
  // do something...
})

tooltip.hide()
```