

Search...

[View on GitHub](#)[Getting started](#)[Layout](#)[Content](#)[Components](#)[Alerts](#)[Badge](#)[Breadcrumb](#)[Buttons](#)[Button group](#)[Card](#)[Carousel](#)[Collapse](#)[Dropdowns](#)[Forms](#)[Input group](#)[Jumbotron](#)[List group](#)[Media object](#)[Modal](#)[Navs](#)[Navbar](#)[Pagination](#)[Popovers](#)[Progress](#)[Scrollspy](#)[Spinners](#)[Toasts](#)[Tooltips](#)[Utilities](#)[Extend](#)[Migration](#)[About](#)

Toasts

Push notifications to your visitors with a toast, a lightweight and easily customizable alert message.



Get 10 Free Images From
Adobe Stock. Start Now.
ads via Carbon

Toasts are lightweight notifications designed to mimic the push notifications that have been popularized by mobile and desktop operating systems. They're built with flexbox, so they're easy to align and position.

Overview

Things to know when using the toast plugin:

- If you're building our JavaScript from source, it [requires util.js](#).
- Toasts are opt-in for performance reasons, so **you must initialize them yourself**.
- **Please note that you are responsible for positioning toasts.**
- Toasts will automatically hide if you do not specify `autohide: false`.

The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the [reduced motion section of our accessibility documentation](#).

Examples

Basic

To encourage extensible and predictable toasts, we recommend a header and body. Toast headers use `display: flex`, allowing easy alignment of content thanks to our margin and flexbox utilities.

Toasts are as flexible as you need and have very little required markup. At a minimum, we require a single element to contain your "toasted" content and strongly encourage a dismiss button.

**Bootstrap**

11 mins ago ✕

Hello, world! This is a toast message.

[Copy](#)

```

<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    
    <strong class="mr-auto">Bootstrap</strong>
    <small>11 mins ago</small>
    <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="toast-body">
    Hello, world! This is a toast message.
  </div>
</div>

```

Live

Click the button below to show a toast (positioned with our utilities in the lower right corner) that has been hidden by default with `.hide`.

Show live toast

Copy

```


<button type="button" class="btn btn-primary" id="liveToastBtn">Show live toast</button>

<div class="position-fixed bottom-0 right-0 p-3" style="z-index: 5; right: 0; bottom: 0;">
  <div id="liveToast" class="toast hide" role="alert" aria-live="assertive" aria-atomic="true" data-delay="2000">
    <div class="toast-header">
      
      <strong class="mr-auto">Bootstrap</strong>
      <small>11 mins ago</small>
      <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
    <div class="toast-body">
      Hello, world! This is a toast message.
    </div>
  </div>
</div>

```

Translucent

Toasts are slightly translucent to blend in with what's below them.

 **Bootstrap** 11 mins ago ✕

Hello, world! This is a toast message.

Copy

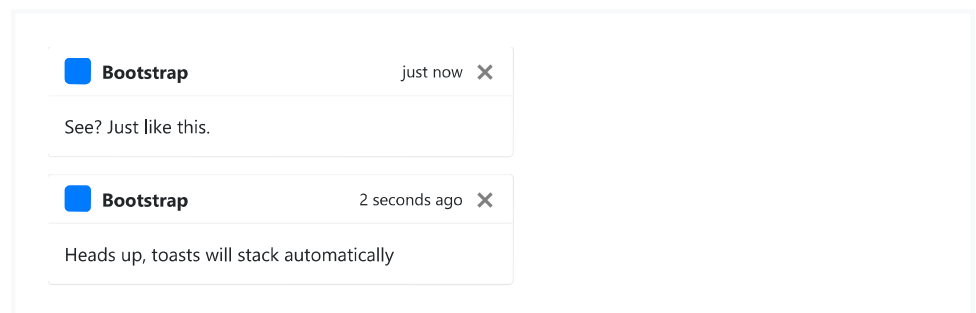
```

<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    
    <strong class="mr-auto">Bootstrap</strong>
    <small class="text-muted">11 mins ago</small>
    <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="toast-body">
    Hello, world! This is a toast message.
  </div>
</div>

```

Stacking

When you have multiple toasts, we default to vertically stacking them in a readable manner.



Copy

```

<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    
    <strong class="mr-auto">Bootstrap</strong>
    <small class="text-muted">just now</small>
    <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="toast-body">
    See? Just like this.
  </div>
</div>

```

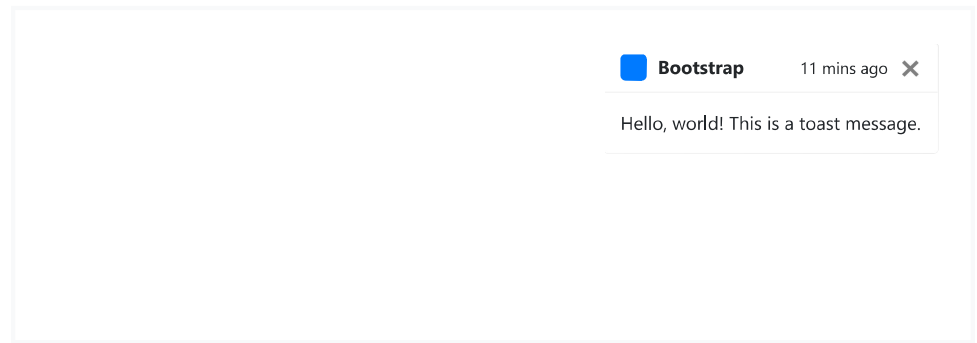
```

<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    
    <strong class="mr-auto">Bootstrap</strong>
    <small class="text-muted">2 seconds ago</small>
    <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="toast-body">
    Heads up, toasts will stack automatically
  </div>
</div>

```

Placement

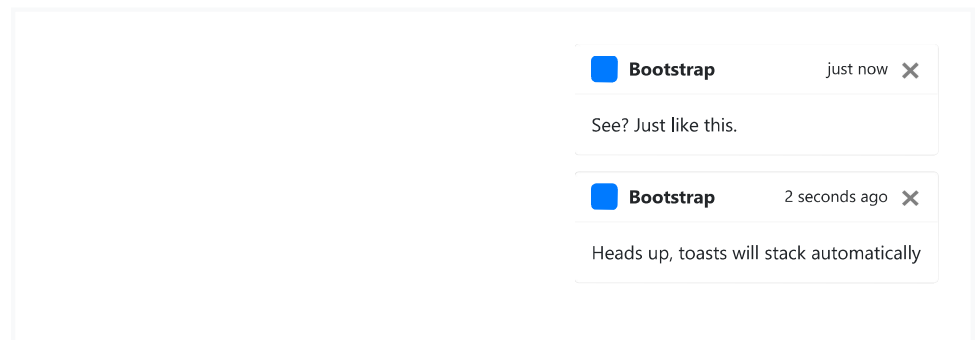
Place toasts with custom CSS as you need them. The top right is often used for notifications, as is the top middle. If you're only ever going to show one toast at a time, put the positioning styles right on the `.toast`.



Copy

```
<div aria-live="polite" aria-atomic="true" style="position: relative; min-height: 200px;">
  <div class="toast" style="position: absolute; top: 0; right: 0;">
    <div class="toast-header">
      
      <strong class="mr-auto">Bootstrap</strong>
      <small>11 mins ago</small>
      <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
    <div class="toast-body">
      Hello, world! This is a toast message.
    </div>
  </div>
</div>
```

For systems that generate more notifications, consider using a wrapping element so they can easily stack.



Copy

```

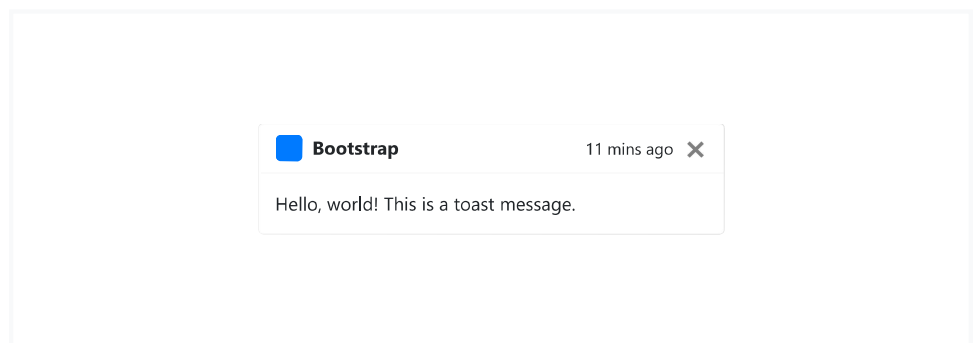
<div aria-live="polite" aria-atomic="true" style="position: relative; min-height: 200px;">
  <!-- Position it -->
  <div style="position: absolute; top: 0; right: 0;">

    <!-- Then put toasts within -->
    <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
      <div class="toast-header">
        
        <strong class="mr-auto">Bootstrap</strong>
        <small class="text-muted">just now</small>
        <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="toast-body">
        See? Just like this.
      </div>
    </div>

    <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
      <div class="toast-header">
        
        <strong class="mr-auto">Bootstrap</strong>
        <small class="text-muted">2 seconds ago</small>
        <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="toast-body">
        Heads up, toasts will stack automatically
      </div>
    </div>
  </div>
</div>

```

You can also get fancy with flexbox utilities to align toasts horizontally and/or vertically.



Copy

```

<!-- Flexbox container for aligning the toasts -->
<div aria-live="polite" aria-atomic="true" class="d-flex justify-content-center align-items-center" style="height: 200px;">

  <!-- Then put toasts within -->
  <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
    <div class="toast-header">
      
      <strong class="mr-auto">Bootstrap</strong>
      <small>11 mins ago</small>
      <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-label="Close">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
    <div class="toast-body">
      Hello, world! This is a toast message.
    </div>
  </div>
</div>

```

Accessibility

Toasts are intended to be small interruptions to your visitors or users, so to help those with screen readers and similar assistive technologies, you should wrap your toasts in an [aria-live region](#). Changes to live regions (such as injecting/updating a toast component) are automatically announced by screen readers without needing to move the user's focus or otherwise interrupt the user. Additionally, include `aria-atomic="true"` to ensure that the entire toast is always announced as a single (atomic) unit, rather than just announcing what was changed (which could lead to problems if you only update part of the toast's content, or if displaying the same toast content at a later point in time). If the information needed is important for the process, e.g. for a list of errors in a form, then use the [alert component](#) instead of toast.

Note that the live region needs to be present in the markup *before* the toast is generated or updated. If you dynamically generate both at the same time and inject them into the page, they will generally not be announced by assistive technologies.

You also need to adapt the `role` and `aria-live` level depending on the content. If it's an important message like an error, use `role="alert" aria-live="assertive"`, otherwise use `role="status" aria-live="polite"` attributes.

As the content you're displaying changes, be sure to update the [delay timeout](#) so that users have enough time to read the toast.

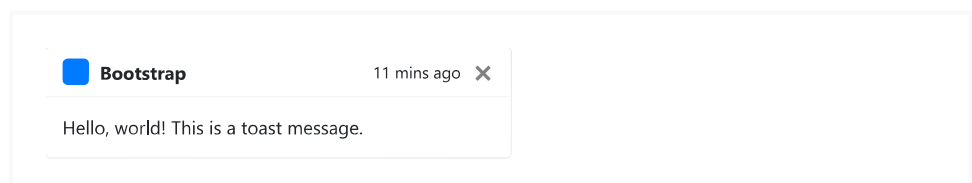
Copy

```

<div class="toast" role="alert" aria-live="polite" aria-atomic="true" data-delay="10000">
  <div role="alert" aria-live="assertive" aria-atomic="true">...</div>
</div>

```

When using `autohide: false`, you must add a close button to allow users to dismiss the toast.



```
<div role="alert" aria-live="assertive" aria-atomic="true" class="toast" data-
  autohide="false">
  <div class="toast-header">
    
    <strong class="mr-auto">Bootstrap</strong>
    <small>11 mins ago</small>
    <button type="button" class="ml-2 mb-1 close" data-dismiss="toast" aria-
      label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="toast-body">
    Hello, world! This is a toast message.
  </div>
</div>
```

While technically it's possible to add focusable/actionable controls (such as additional buttons or links) in your toast, you should avoid doing this for autohiding toasts. Even if you give the toast a long [delay timeout](#), keyboard and assistive technology users may find it difficult to reach the toast in time to take action (since toasts don't receive focus when they are displayed). If you absolutely must have further controls, we recommend using a toast with `autohide: false`.

JavaScript behavior

Usage

Initialize toasts via JavaScript:

```
$('.toast').toast(option)
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-animation=""`.

Name	Type	Default	Description
animation	boolean	true	Apply a CSS fade transition to the toast
autohide	boolean	true	Auto hide the toast
delay	number	500	Delay hiding the toast (ms)

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

[See our JavaScript documentation for more information.](#)

`$.toast(options)`

Attaches a toast handler to an element collection.

`.toast('show')`

Reveals an element's toast. **Returns to the caller before the toast has actually been shown** (i.e. before the `shown.bs.toast` event occurs). You have to manually call this method, instead your toast won't show.

```
$('#element').toast('show')
```

[Copy](#)

`.toast('hide')`

Hides an element's toast. **Returns to the caller before the toast has actually been hidden** (i.e. before the `hidden.bs.toast` event occurs). You have to manually call this method if you made `autohide` to `false`.

```
$('#element').toast('hide')
```

[Copy](#)

`.toast('dispose')`

Hides an element's toast. Your toast will remain on the DOM but won't show anymore.

```
$('#element').toast('dispose')
```

[Copy](#)

Events

Event Type	Description
show.bs.toast	This event fires immediately when the <code>show</code> instance method is called.
shown.bs.toast	This event is fired when the toast has been made visible to the user.
hide.bs.toast	This event is fired immediately when the <code>hide</code> instance method has been called.
hidden.bs.toast	This event is fired when the toast has finished being hidden from the user.

```
$('#myToast').on('hidden.bs.toast', function () {  
  // do something...  
})
```

[Copy](#)