

[Ctrl + /](#)

v5.0 ▾

[View on GitHub](#)

Toasts

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



Your new development career awaits. Check out the latest listings.

ads via Carbon

On this page

[Overview](#)[Examples](#)[Basic](#)[Live](#)[Translucent](#)[Stacking](#)[Custom content](#)[Color schemes](#)[Placement](#)[Accessibility](#)[Sass](#)[Variables](#)[Usage](#)[Options](#)[Methods](#)[show](#)[hide](#)[dispose](#)[getInstance](#)

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:

- Toasts are opt-in for performance reasons, so **you must initialize them yourself**.
- 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.

```
<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    
    <strong class="me-auto">Bootstrap</strong>
    <small>11 mins ago</small>
    <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"
  </div>
```

```
<div class="toast-body">
  Hello, world! This is a toast message.
</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

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

<div class="position-fixed bottom-0 end-0 p-3" style="z-index: 11">
  <div id="liveToast" class="toast hide" role="alert" aria-live="assertive" aria-ato
    <div class="toast-header">
      
      <strong class="me-auto">Bootstrap</strong>
      <small>11 mins ago</small>
      <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Cl
    </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.

```
<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
```

```

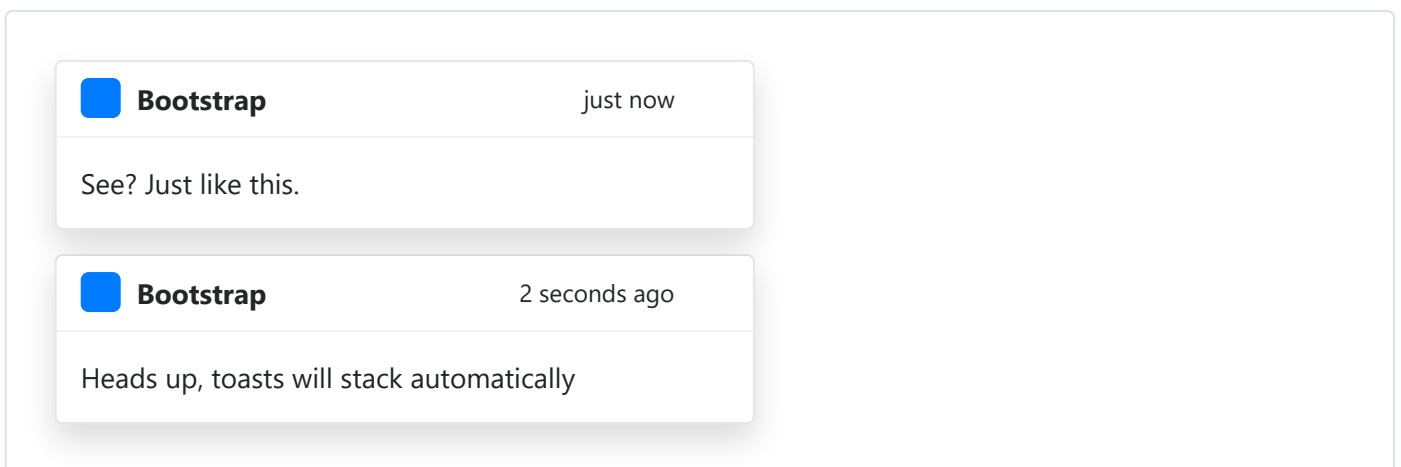

<strong class="me-auto">Bootstrap</strong>
<small class="text-muted">11 mins ago</small>
<button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Clos
</div>
<div class="toast-body">
  Hello, world! This is a toast message.
</div>
...

```



Stacking

You can stack toasts by wrapping them in a toast container, which will vertically add some spacing.



```

<div class="toast-container">
  <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
    <div class="toast-header">
      
      <strong class="me-auto">Bootstrap</strong>
      <small class="text-muted">just now</small>
      <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Cl
    </div>
    <div class="toast-body">
      See? Just like this.
    </div>
  </div>

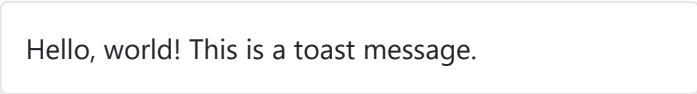
  <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
    <div class="toast-header">
      
      <strong class="me-auto">Bootstrap</strong>
      <small class="text-muted">2 seconds ago</small>
      <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Cl
    </div>
    <div class="toast-body">
      Heads up, toasts will stack automatically
    </div>
  </div>
</div>

```

```
</div>
</div>
...
```


Custom content

Customize your toasts by removing sub-components, tweaking them with [utilities](#), or by adding your own markup. Here we've created a simpler toast by removing the default `.toast-header`, adding a custom hide icon from [Bootstrap Icons](#), and using some [flexbox utilities](#) to adjust the layout.

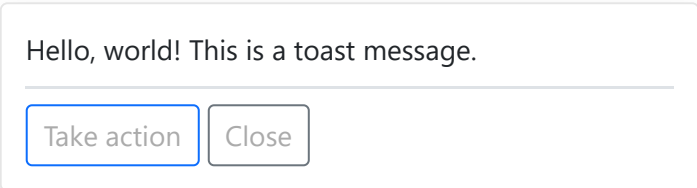


Hello, world! This is a toast message.

```
<div class="toast align-items-center" role="alert" aria-live="assertive" aria-atomic
  <div class="d-flex">
    <div class="toast-body">
      Hello, world! This is a toast message.
    </div>
    <button type="button" class="btn-close me-2 m-auto" data-bs-dismiss="toast" aria
  </div>
</div>
```



Alternatively, you can also add additional controls and components to toasts.



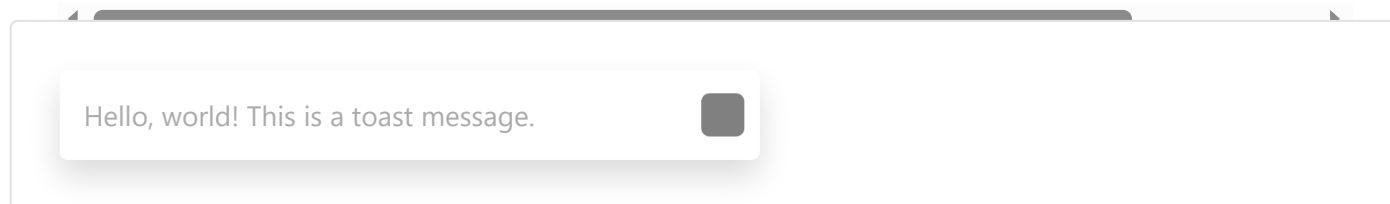
Hello, world! This is a toast message.

```
<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-body">
    Hello, world! This is a toast message.
    <div class="mt-2 pt-2 border-top">
      <button type="button" class="btn btn-primary btn-sm">Take action</button>
      <button type="button" class="btn btn-secondary btn-sm" data-bs-dismiss="toast"
    </div>
```

```
</div>
</div>
```

Color schemes

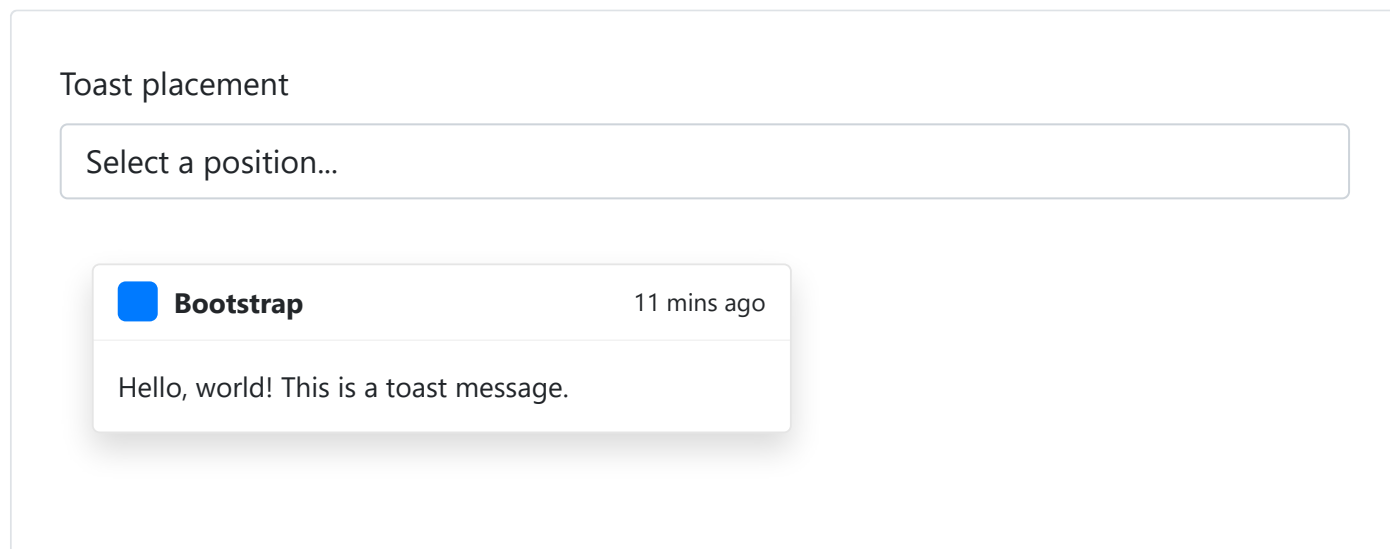
Building on the above example, you can create different toast color schemes with our [color](#) and [background](#) utilities. Here we've added `.bg-primary` and `.text-white` to the `.toast`, and then added `.btn-close-white` to our close button. For a crisp edge, we remove the default border with `.border-0`.



```
<div class="toast align-items-center text-white bg-primary border-0" role="alert" ar
  <div class="d-flex">
    <div class="toast-body">
      Hello, world! This is a toast message.
    </div>
    <button type="button" class="btn-close btn-close-white me-2 m-auto" data-bs-dism
  </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`.



```

<form>
  <div class="mb-3">
    <label for="selectToastPlacement">Toast placement</label>
    <select class="form-select mt-2" id="selectToastPlacement">
      <option value="" selected>Select a position...</option>
      <option value="top-0 start-0">Top left</option>
      <option value="top-0 start-50 translate-middle-x">Top center</option>
      <option value="top-0 end-0">Top right</option>
      <option value="top-50 start-0 translate-middle-y">Middle left</option>
      <option value="top-50 start-50 translate-middle">Middle center</option>
      <option value="top-50 end-0 translate-middle-y">Middle right</option>
      <option value="bottom-0 start-0">Bottom left</option>
      <option value="bottom-0 start-50 translate-middle-x">Bottom center</option>
      <option value="bottom-0 end-0">Bottom right</option>
    </select>
  </div>
</form>
<div aria-live="polite" aria-atomic="true" class="bg-dark position-relative bd-examp
  <div class="toast-container position-absolute p-3" id="toastPlacement">
    <div class="toast">
      <div class="toast-header">
        
        <strong class="me-auto">Bootstrap</strong>
        <small>11 mins ago</small>
      </div>
      <div class="toast-body">
        Hello, world! This is a toast message.
      </div>
    </div>
  </div>
</div>
</div>

```

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



Bootstrap

just now

See? Just like this.

**Bootstrap**

2 seconds ago

Heads up, toasts will stack automatically

```
<div aria-live="polite" aria-atomic="true" class="position-relative">
  <!-- Position it: -->
  <!-- - `.toast-container` for spacing between toasts -->
  <!-- - `.position-absolute`, `top-0` & `end-0` to position the toasts in the upper
  <!-- - `.p-3` to prevent the toasts from sticking to the edge of the container --
  <div class="toast-container position-absolute top-0 end-0 p-3">

    <!-- Then put toasts within -->
    <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
      <div class="toast-header">
        
        <strong class="me-auto">Bootstrap</strong>
        <small class="text-muted">just now</small>
        <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="
      </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="me-auto">Bootstrap</strong>
        <small class="text-muted">2 seconds ago</small>
        <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="
      </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.

**Bootstrap**

11 mins ago

Hello, world! This is a toast message.

```
<!-- Flexbox container for aligning the toasts -->
<div aria-live="polite" aria-atomic="true" class="d-flex justify-content-center align-items-center">

  <!-- Then put toasts within -->
  <div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
    <div class="toast-header">
      
      <strong class="me-auto">Bootstrap</strong>
      <small>11 mins ago</small>
      <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></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.

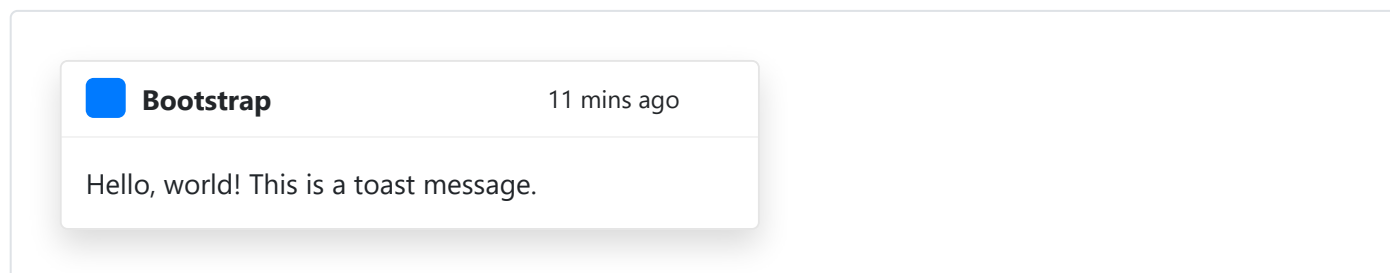
Note that the live region needs to be present in the markup *before* the toast is generated or updated. If you dynamically generate both at the same time and inject them into the page, they will generally not be announced by assistive technologies.

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.

```
<div class="toast" role="alert" aria-live="polite" aria-atomic="true" data-bs-delay=
  <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-bs-aut
  <div class="toast-header">
    
    <strong class="me-auto">Bootstrap</strong>
    <small>11 mins ago</small>
    <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Clos
  </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`.

Sass

Variables

```
$toast-max-width: 350px;
$toast-padding-x: .75rem;
```

```
$toast-padding-y: .5rem;
$toast-font-size: .875rem;
$toast-color: null;
$toast-background-color: rgba($white, .85);
$toast-border-width: 1px;
$toast-border-color: rgba(0, 0, 0, .1);
$toast-border-radius: $border-radius;
$toast-box-shadow: $box-shadow;
$toast-spacing: $container-padding-x;

$toast-header-color: $gray-600;
$toast-header-background-color: rgba($white, .85);
$toast-header-border-color: rgba(0, 0, 0, .05);
```

Usage

Initialize toasts via JavaScript:

```
var toastElList = [].slice.call(document.querySelectorAll('.toast'))
var toastList = toastElList.map(function (toastEl) {
  return new bootstrap.Toast(toastEl, option)
})
```

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=""`.

Name	Type	Default	Description
<code>animation</code>	boolean	<code>true</code>	Apply a CSS fade transition to the toast
<code>autohide</code>	boolean	<code>true</code>	Auto hide the toast
<code>delay</code>	number	<code>5000</code>	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.](#)

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.

```
toast.show()
```

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`.

```
toast.hide()
```

dispose

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

```
toast.dispose()
```

getInstance

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

```
var myToastEl = document.getElementById('myToastEl')  
var myToast = bootstrap.Toast.getInstance(myToastEl) // Returns a Bootstrap toast in
```

getOrCreateInstance

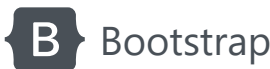
Static method which allows you to get the scrollspy instance associated with a DOM element, or create a new one in case it wasn't initialised

```
var myToastEl = document.getElementById('myToastEl')
var myToast = bootstrap.Toast.getOrCreateInstance(myToastEl) // Returns a Bootstrap
```

Events

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

```
var myToastEl = document.getElementById('myToast')
myToastEl.addEventListener('hidden.bs.toast', function () {
  // do something...
})
```



Designed and built with all the love in the world by the Bootstrap team with the help of our contributors.

Code licensed MIT, docs CC BY 3.0.

Currently v5.0.2.

Analytics by Fathom.

Links

[Home](#)

[Docs](#)

[Examples](#)

[Themes](#)

Guides

[Getting started](#)

[Starter template](#)

[Webpack](#)

[Parcel](#)

[Blog](#)

[Swag Store](#)

Projects

[Bootstrap 5](#)

[Bootstrap 4](#)

[Icons](#)

[RFS](#)

[npm starter](#)

Community

[Issues](#)

[Discussions](#)

[Corporate sponsors](#)

[Open Collective](#)

[Stack Overflow](#)