Toasts

Contents

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`.

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

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.

{{< example class="bg-light" >}}

```
{{< placeholder width="20" height="20" background="#007aff" class="rounded me-2" text="false
<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>

Hello, world! This is a toast message.
```

{{< /example >}}

{{< callout warning >}} Previously, our scripts dynamically added the `.hide` class to completely hide a toast (with `display:none`, rather than just with `opacity:0`). This is now not necessary anymore. However, for backwards compatibility, our script will continue to toggle the class (even though there is no practical need for it) until the next major version. {{< /callout >}}

Live example

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

```
<div class="toast-header">
  {{< placeholder width="20" height="20" background="#007aff" class="rounded me-2" text="fal
  <strong class="me-auto">Bootstrap</strong>
  <small>11 mins ago</small>
  <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></butto
</div>
<div class="toast-body">
  Hello, world! This is a toast message.
</div>
```

Show live toast

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

<div class="position-fixed bottom-0 end-0 p-3" style="z-index: 11">
  <div id="liveToast" class="toast" role="alert" aria-live="assertive" aria-atomic="true">
    <div class="toast-header">
```

```
      <img src="..." class="rounded me-2" alt="...">
      <strong class="me-auto">Bootstrap</strong>
      <small>11 mins ago</small>
      <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></b
    </div>
    <div class="toast-body">
      Hello, world! This is a toast message.
    </div>
  </div>
</div>
```

We use the following JavaScript to trigger our live toast demo:

```
var toastTrigger = document.getElementById('liveToastBtn')
var toastLiveExample = document.getElementById('liveToast')
if (toastTrigger) {
  toastTrigger.addEventListener('click', function () {
    var toast = new bootstrap.Toast(toastLiveExample)

    toast.show()
  })
}
```

Translucent

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

{{< example class="bg-dark" >}}

```
{{< placeholder width="20" height="20" background="#007aff" class="rounded me-2" text="false
<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="Close"></button>

Hello, world! This is a toast message.
```

{{< /example >}}

Stacking

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

{{< example class="bg-light" >}}

```
<div class="toast-header">
  {{< placeholder width="20" height="20" background="#007aff" class="rounded me-2" text="fal
  <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="Close"></butto
```

```
</div>
<div class="toast-body">
  See? Just like this.
</div>
```

```
<div class="toast-header">
  {{< placeholder width="20" height="20" background="#007aff" class="rounded me-2" text="fal
  <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="Close"></butto
</div>
<div class="toast-body">
  Heads up, toasts will stack automatically
</div>
```

{{< /example >}}

Custom content

Customize your toasts by removing sub-components, tweaking them with [utilities]({{< docsref "/utilities/api" >}}), 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]({{< docsref "/utilities/flex" >}}) to adjust the layout.

{{< example class="bg-light" >}}

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

```
<button type="button" class="btn-close me-2 m-auto" data-bs-dismiss="toast" aria-label="Clos
```

{{< /example >}}

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

{{< example class="bg-light" >}}

```
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">Close</butt
</div>
```

{{< /example >}}

Color schemes

Building on the above example, you can create different toast color schemes with our [color]({{< docsref "/utilities/colors" >}}) and [background]({{< docsref "/utilities/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`.

{{< example class="bg-light" >}}

```
<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-dismiss="toast"
```

{{< /example >}}

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`.

{{< example >}}

```
<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 class="toast">
  <div class="toast-header">
    {{< placeholder width="20" height="20" background="#007aff" class="rounded me-2" text="
    <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>
```

{{< /example >}}

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

{{< example class="bg-dark bd-example-toasts p-0" >}}

```
<!-- Then put toasts within -->
<div class="toast" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="toast-header">
    {{< placeholder width="20" height="20" background="#007aff" class="rounded me-2" text="f
    <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="Close"></but
  </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">
    {{< placeholder width="20" height="20" background="#007aff" class="rounded me-2" text="f
    <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="Close"></but
  </div>
  <div class="toast-body">
    Heads up, toasts will stack automatically
  </div>
</div>
```

{{< /example >}}

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

{{< example class="bg-dark bd-example-toasts d-flex" >}}

```
<div class="toast-header">
  {{< placeholder width="20" height="20" background="#007aff" class="rounded me-2" text="fal
  <strong class="me-auto">Bootstrap</strong>
  <small>11 mins ago</small>
  <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></butto
</div>
<div class="toast-body">
  Hello, world! This is a toast message.
</div>
```

{{< /example >}}

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]({{< docsref "/components/alerts" >}}) 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="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.

{{< example class="bg-light" >}}

```
{{< placeholder width="20" height="20" background="#007aff" class="rounded me-2" text="false
<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>

Hello, world! This is a toast message.
```

{{< /example >}}

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

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

Usage

Initialize toasts via JavaScript:

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

Triggers

{{% js-dismiss "toast" %}}

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

animation

boolean

true

Apply a CSS fade transition to the toast

autohide

boolean

true

Auto hide the toast

delay

number

5000

Delay hiding the toast (ms)

Methods

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

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 toast instance associated with a DOM element

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

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

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

Events

Event type

Description

show.bs.toast

This event fires immediately when the show 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 hide instance method has been called.

hidden.bs.toast

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...
})
```