JavaScript

Bring Bootstrap to life with our optional JavaScript plugins built on jQuery. Learn about each plugin, our data and programmatic API options, and more.

Individual or compiled

Plugins can be included individually (using Bootstrap's individual `*.js` files), or all at once using `bootstrap.js` or the minified `bootstrap.min.js` (don't include both).

Dependencies

Some plugins and CSS components depend on other plugins. If you include plugins individually, make sure to check for these dependencies in the docs. Also note that **all plugins depend on jQuery** (this means jQuery must be included **before** the plugin files). Consult our `package.json` to see which versions of jQuery are supported.

Our dropdowns, popovers and tooltips also depend on Popper.js.

Data attributes

Nearly all Bootstrap plugins can be enabled and configured through HTML alone with data attributes (our preferred way of using JavaScript functionality). Be sure to **only use one set of data attributes on a single element** (e.g., you cannot trigger a tooltip and modal from the same button.)

However, in some situations it may be desirable to disable this functionality. To disable the data attribute API, unbind all events on the document namespaced with `data-api` like so:

```
$(document).off('.data-api')
```

Alternatively, to target a specific plugin, just include the plugin's name as a namespace along with the data-api namespace like this:

```
$(document).off('.alert.data-api')
```

Events

Bootstrap provides custom events for most plugins' unique actions. Generally, these come in an infinitive and past participle form - where the infinitive (ex. `show`) is triggered at the start of an event, and its past participle form (ex. `shown`) is triggered on the completion of an action.

All infinitive events provide `preventDefault()` functionality. This provides the ability to stop the execution of an action before it starts. Returning false from an event handler will also automatically call `preventDefault()`.

```
$('#myModal').on('show.bs.modal', function (e) {
  if (!data) return e.preventDefault() // stops modal from being shown
})
```

Programmatic API

We also believe you should be able to use all Bootstrap plugins purely through the JavaScript API. All public APIs are single, chainable methods, and return the collection acted upon.

```
$('.btn.danger').button('toggle').addClass('fat')
```

All methods should accept an optional options object, a string which targets a particular method, or nothing (which initiates a plugin with default behavior):

```
$('#myModal').modal()                      // initialized with defaults
$('#myModal').modal({ keyboard: false })   // initialized with no keyboard
$('#myModal').modal('show')                // initializes and invokes show
immediately
```

Each plugin also exposes its raw constructor on a `Constructor` property: `$.fn.popover.Constructor`. If you'd like to get a particular plugin instance, retrieve it directly from an element: `$('[rel="popover"]').data('popover')`.

Asynchronous functions and transitions

All programmatic API methods are **asynchronous** and returns to the caller once the transition is started but **before it ends**.

In order to execute an action once the transition is complete, you can listen to the corresponding event.

```
$('#myCollapse').on('shown.bs.collapse', function (e) {
  // Action to execute once the collapsible area is expanded
})
```

In addition a method call on a **transitioning component will be ignored**.

```
$('#myCarousel').on('slid.bs.carousel', function (e) {
  $('#myCarousel').carousel('2') // Will slide to the slide 2 as soon as the
transition to slide 1 is finished
})

$('#myCarousel').carousel('1') // Will start sliding to the slide 1 and returns
to the caller
$('#myCarousel').carousel('2') // !! Will be ignored, as the transition to the
slide 1 is not finished !!
```

Default settings

You can change the default settings for a plugin by modifying the plugin's `Constructor.Default` object:

```
$.fn.modal.Constructor.Default.keyboard = false // changes default for the modal
plugin's `keyboard` option to false
```

No conflict

Sometimes it is necessary to use Bootstrap plugins with other UI frameworks. In these circumstances, namespace collisions can occasionally occur. If this happens, you may call `.noConflict` on the plugin you wish to revert the value of.

Copy

```
var bootstrapButton = $.fn.button.noConflict() // return $.fn.button to previously assigned value
$.fn.bootstrapBtn = bootstrapButton            // give $().bootstrapBtn the Bootstrap functionality
```

Version numbers

The version of each of Bootstrap's jQuery plugins can be accessed via the `VERSION` property of the plugin's constructor. For example, for the tooltip plugin:

Copy

```
$.fn.tooltip.Constructor.VERSION // => "4.0.0"
```

No special fallbacks when JavaScript is disabled

Bootstrap's plugins don't fall back particularly gracefully when JavaScript is disabled. If you care about the user experience in this case, use `<noscript>` to explain the situation (and how to re-enable JavaScript) to your users, and/or add your own custom fallbacks.

Third-party libraries

Bootstrap does not officially support third-party JavaScript libraries like Prototype or jQuery UI. Despite `.noConflict` and namespaced events, there may be compatibility problems that you need to fix on your own.

Util

All Bootstrap's JavaScript files depend on `util.js` and it has to be included alongside the other JavaScript files. If you're using the compiled (or minified) `bootstrap.js`, there is no need to include this—it's already there.

`util.js` includes utility functions and a basic helper for `transitionEnd` events as well as a CSS transition emulator. It's used by the other plugins to check for CSS transition support and to catch hanging transitions.