js\bootstrap.js

```
1  if (typeof jQuery === 'undefined') {
2      throw new Error('Bootstrap\'s JavaScript requires jQuery')
3    }
4
5    +function ($) {
6      'use strict';
7      var version = $.fn.jquery.split(' ')[0].split('.')
8      if ((version[0] < 2 && version[1] < 9) || (version[0] == 1 && version[1] == 9 &&
   version[2] < 1) || (version[0] > 2)) {
9        throw new Error('Bootstrap\'s JavaScript requires jQuery version 1.9.1 or higher,
   but lower than version 3')
10     }
11   }(jQuery);
12
13   /* ========================================================================
14    * Bootstrap: transition.js v3.3.6
15    * http://getbootstrap.com/javascript/#transitions
16    * ========================================================================
17    * Copyright 2011-2015 Twitter, Inc.
18    * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
19    * ======================================================================== */
20
21
22   +function ($) {
23     'use strict';
24
25     // CSS TRANSITION SUPPORT (Shoutout: http://www.modernizr.com/)
26     // ============================================================
27
28     function transitionEnd() {
29       var el = document.createElement('bootstrap')
30
31       var transEndEventNames = {
32         WebkitTransition : 'webkitTransitionEnd',
33         MozTransition    : 'transitionend',
34         OTransition      : 'oTransitionEnd otransitionend',
35         transition       : 'transitionend'
36       }
37
38       for (var name in transEndEventNames) {
39         if (el.style[name] !== undefined) {
40           return { end: transEndEventNames[name] }
41         }
42       }
43
44       return false // explicit for ie8 (  ._.)
45     }
46
47     // http://blog.alexmaccaw.com/css-transitions
48     $.fn.emulateTransitionEnd = function (duration) {
49       var called = false
50       var $el = this
51       $(this).one('bsTransitionEnd', function () { called = true })
52       var callback = function () { if (!called) $($el).trigger($.support.transition.end)
   }
53       setTimeout(callback, duration)
54       return this
```

```
  }

  $(function () {
    $.support.transition = transitionEnd()

    if (!$.support.transition) return

    $.event.special.bsTransitionEnd = {
      bindType: $.support.transition.end,
      delegateType: $.support.transition.end,
      handle: function (e) {
        if ($(e.target).is(this)) return e.handleObj.handler.apply(this, arguments)
      }
    }
  })

}(jQuery);

/* ========================================================================
 * Bootstrap: alert.js v3.3.6
 * http://getbootstrap.com/javascript/#alerts
 * ========================================================================
 * Copyright 2011-2015 Twitter, Inc.
 * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
 * ======================================================================== */


+function ($) {
  'use strict';

  // ALERT CLASS DEFINITION
  // ======================

  var dismiss = '[data-dismiss="alert"]'
  var Alert   = function (el) {
    $(el).on('click', dismiss, this.close)
  }

  Alert.VERSION = '3.3.6'

  Alert.TRANSITION_DURATION = 150

  Alert.prototype.close = function (e) {
    var $this    = $(this)
    var selector = $this.attr('data-target')

    if (!selector) {
      selector = $this.attr('href')
      selector = selector && selector.replace(/.*(?=#[^\s]*$)/, '') // strip for ie7
    }

    var $parent = $(selector)

    if (e) e.preventDefault()

    if (!$parent.length) {
      $parent = $this.closest('.alert')
    }

    $parent.trigger(e = $.Event('close.bs.alert'))
```

```
115
116       if (e.isDefaultPrevented()) return
117
118       $parent.removeClass('in')
119
120       function removeElement() {
121         // detach from parent, fire event then clean up data
122         $parent.detach().trigger('closed.bs.alert').remove()
123       }
124
125       $.support.transition && $parent.hasClass('fade') ?
126         $parent
127           .one('bsTransitionEnd', removeElement)
128           .emulateTransitionEnd(Alert.TRANSITION_DURATION) :
129         removeElement()
130     }
131
132
133     // ALERT PLUGIN DEFINITION
134     // =======================
135
136     function Plugin(option) {
137       return this.each(function () {
138         var $this = $(this)
139         var data  = $this.data('bs.alert')
140
141         if (!data) $this.data('bs.alert', (data = new Alert(this)))
142         if (typeof option == 'string') data[option].call($this)
143       })
144     }
145
146     var old = $.fn.alert
147
148     $.fn.alert             = Plugin
149     $.fn.alert.Constructor = Alert
150
151
152     // ALERT NO CONFLICT
153     // =================
154
155     $.fn.alert.noConflict = function () {
156       $.fn.alert = old
157       return this
158     }
159
160
161     // ALERT DATA-API
162     // ==============
163
164     $(document).on('click.bs.alert.data-api', dismiss, Alert.prototype.close)
165
166   }(jQuery);
167
168   /* ========================================================================
169    * Bootstrap: button.js v3.3.6
170    * http://getbootstrap.com/javascript/#buttons
171    * ========================================================================
172    * Copyright 2011-2015 Twitter, Inc.
173    * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
174    * ======================================================================== */
```

```
175
176
177   +function ($) {
178     'use strict';
179
180     // BUTTON PUBLIC CLASS DEFINITION
181     // ==============================
182
183     var Button = function (element, options) {
184       this.$element  = $(element)
185       this.options   = $.extend({}, Button.DEFAULTS, options)
186       this.isLoading = false
187     }
188
189     Button.VERSION  = '3.3.6'
190
191     Button.DEFAULTS = {
192       loadingText: 'loading...'
193     }
194
195     Button.prototype.setState = function (state) {
196       var d    = 'disabled'
197       var $el  = this.$element
198       var val  = $el.is('input') ? 'val' : 'html'
199       var data = $el.data()
200
201       state += 'Text'
202
203       if (data.resetText == null) $el.data('resetText', $el[val]())
204
205       // push to event loop to allow forms to submit
206       setTimeout($.proxy(function () {
207         $el[val](data[state] == null ? this.options[state] : data[state])
208
209         if (state == 'loadingText') {
210           this.isLoading = true
211           $el.addClass(d).attr(d, d)
212         } else if (this.isLoading) {
213           this.isLoading = false
214           $el.removeClass(d).removeAttr(d)
215         }
216       }, this), 0)
217     }
218
219     Button.prototype.toggle = function () {
220       var changed = true
221       var $parent = this.$element.closest('[data-toggle="buttons"]')
222
223       if ($parent.length) {
224         var $input = this.$element.find('input')
225         if ($input.prop('type') == 'radio') {
226           if ($input.prop('checked')) changed = false
227           $parent.find('.active').removeClass('active')
228           this.$element.addClass('active')
229         } else if ($input.prop('type') == 'checkbox') {
230           if (($input.prop('checked')) !== this.$element.hasClass('active')) changed =
    false
231           this.$element.toggleClass('active')
232         }
233         $input.prop('checked', this.$element.hasClass('active'))
```

```
234         if (changed) $input.trigger('change')
235       } else {
236         this.$element.attr('aria-pressed', !this.$element.hasClass('active'))
237         this.$element.toggleClass('active')
238       }
239     }
240
241
242     // BUTTON PLUGIN DEFINITION
243     // ========================
244
245     function Plugin(option) {
246       return this.each(function () {
247         var $this   = $(this)
248         var data    = $this.data('bs.button')
249         var options = typeof option == 'object' && option
250
251         if (!data) $this.data('bs.button', (data = new Button(this, options)))
252
253         if (option == 'toggle') data.toggle()
254         else if (option) data.setState(option)
255       })
256     }
257
258     var old = $.fn.button
259
260     $.fn.button             = Plugin
261     $.fn.button.Constructor = Button
262
263
264     // BUTTON NO CONFLICT
265     // ==================
266
267     $.fn.button.noConflict = function () {
268       $.fn.button = old
269       return this
270     }
271
272
273     // BUTTON DATA-API
274     // ===============
275
276     $(document)
277       .on('click.bs.button.data-api', '[data-toggle^="button"]', function (e) {
278         var $btn = $(e.target)
279         if (!$btn.hasClass('btn')) $btn = $btn.closest('.btn')
280         Plugin.call($btn, 'toggle')
281         if (!($(e.target).is('input[type="radio"]') ||
$(e.target).is('input[type="checkbox"]'))) e.preventDefault()
282       })
283       .on('focus.bs.button.data-api blur.bs.button.data-api', '[data-toggle^="button"]',
function (e) {
284         $(e.target).closest('.btn').toggleClass('focus', /^focus(in)?$/.test(e.type))
285       })
286
287   }(jQuery);
288
289   /* ========================================================================
290    * Bootstrap: carousel.js v3.3.6
291    * http://getbootstrap.com/javascript/#carousel
```

```
292   * ========================================================================
293   * Copyright 2011-2015 Twitter, Inc.
294   * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
295   * ======================================================================== */
296
297
298  +function ($) {
299    'use strict';
300
301    // CAROUSEL CLASS DEFINITION
302    // =========================
303
304    var Carousel = function (element, options) {
305      this.$element    = $(element)
306      this.$indicators = this.$element.find('.carousel-indicators')
307      this.options     = options
308      this.paused      = null
309      this.sliding     = null
310      this.interval    = null
311      this.$active     = null
312      this.$items      = null
313
314      this.options.keyboard && this.$element.on('keydown.bs.carousel', $.proxy(this.keydown, this))
315
316      this.options.pause == 'hover' && !('ontouchstart' in document.documentElement) && this.$element
317        .on('mouseenter.bs.carousel', $.proxy(this.pause, this))
318        .on('mouseleave.bs.carousel', $.proxy(this.cycle, this))
319    }
320
321    Carousel.VERSION  = '3.3.6'
322
323    Carousel.TRANSITION_DURATION = 600
324
325    Carousel.DEFAULTS = {
326      interval: 5000,
327      pause: 'hover',
328      wrap: true,
329      keyboard: true
330    }
331
332    Carousel.prototype.keydown = function (e) {
333      if (/input|textarea/i.test(e.target.tagName)) return
334      switch (e.which) {
335        case 37: this.prev(); break
336        case 39: this.next(); break
337        default: return
338      }
339
340      e.preventDefault()
341    }
342
343    Carousel.prototype.cycle = function (e) {
344      e || (this.paused = false)
345
346      this.interval && clearInterval(this.interval)
347
348      this.options.interval
349        && !this.paused
```

```
350         && (this.interval = setInterval($.proxy(this.next, this), this.options.interval))
351
352       return this
353     }
354
355     Carousel.prototype.getItemIndex = function (item) {
356       this.$items = item.parent().children('.item')
357       return this.$items.index(item || this.$active)
358     }
359
360     Carousel.prototype.getItemForDirection = function (direction, active) {
361       var activeIndex = this.getItemIndex(active)
362       var willWrap = (direction == 'prev' && activeIndex === 0)
363                   || (direction == 'next' && activeIndex == (this.$items.length - 1))
364       if (willWrap && !this.options.wrap) return active
365       var delta = direction == 'prev' ? -1 : 1
366       var itemIndex = (activeIndex + delta) % this.$items.length
367       return this.$items.eq(itemIndex)
368     }
369
370     Carousel.prototype.to = function (pos) {
371       var that        = this
372       var activeIndex = this.getItemIndex(this.$active =
    this.$element.find('.item.active'))
373
374       if (pos > (this.$items.length - 1) || pos < 0) return
375
376       if (this.sliding)       return this.$element.one('slid.bs.carousel', function () {
    that.to(pos) }) // yes, "slid"
377       if (activeIndex == pos) return this.pause().cycle()
378
379       return this.slide(pos > activeIndex ? 'next' : 'prev', this.$items.eq(pos))
380     }
381
382     Carousel.prototype.pause = function (e) {
383       e || (this.paused = true)
384
385       if (this.$element.find('.next, .prev').length && $.support.transition) {
386         this.$element.trigger($.support.transition.end)
387         this.cycle(true)
388       }
389
390       this.interval = clearInterval(this.interval)
391
392       return this
393     }
394
395     Carousel.prototype.next = function () {
396       if (this.sliding) return
397       return this.slide('next')
398     }
399
400     Carousel.prototype.prev = function () {
401       if (this.sliding) return
402       return this.slide('prev')
403     }
404
405     Carousel.prototype.slide = function (type, next) {
406       var $active   = this.$element.find('.item.active')
407       var $next     = next || this.getItemForDirection(type, $active)
```

```
408       var isCycling = this.interval
409       var direction = type == 'next' ? 'left' : 'right'
410       var that      = this
411
412       if ($next.hasClass('active')) return (this.sliding = false)
413
414       var relatedTarget = $next[0]
415       var slideEvent = $.Event('slide.bs.carousel', {
416         relatedTarget: relatedTarget,
417         direction: direction
418       })
419       this.$element.trigger(slideEvent)
420       if (slideEvent.isDefaultPrevented()) return
421
422       this.sliding = true
423
424       isCycling && this.pause()
425
426       if (this.$indicators.length) {
427         this.$indicators.find('.active').removeClass('active')
428         var $nextIndicator = $(this.$indicators.children()[this.getItemIndex($next)])
429         $nextIndicator && $nextIndicator.addClass('active')
430       }
431
432       var slidEvent = $.Event('slid.bs.carousel', { relatedTarget: relatedTarget,
    direction: direction }) // yes, "slid"
433       if ($.support.transition && this.$element.hasClass('slide')) {
434         $next.addClass(type)
435         $next[0].offsetWidth // force reflow
436         $active.addClass(direction)
437         $next.addClass(direction)
438         $active
439           .one('bsTransitionEnd', function () {
440             $next.removeClass([type, direction].join(' ')).addClass('active')
441             $active.removeClass(['active', direction].join(' '))
442             that.sliding = false
443             setTimeout(function () {
444               that.$element.trigger(slidEvent)
445             }, 0)
446           })
447           .emulateTransitionEnd(Carousel.TRANSITION_DURATION)
448       } else {
449         $active.removeClass('active')
450         $next.addClass('active')
451         this.sliding = false
452         this.$element.trigger(slidEvent)
453       }
454
455       isCycling && this.cycle()
456
457       return this
458     }
459
460
461     // CAROUSEL PLUGIN DEFINITION
462     // ==========================
463
464     function Plugin(option) {
465       return this.each(function () {
466         var $this   = $(this)
```

```
467         var data    = $this.data('bs.carousel')
468         var options = $.extend({}, Carousel.DEFAULTS, $this.data(), typeof option ==
    'object' && option)
469         var action  = typeof option == 'string' ? option : options.slide
470
471         if (!data) $this.data('bs.carousel', (data = new Carousel(this, options)))
472         if (typeof option == 'number') data.to(option)
473         else if (action) data[action]()
474         else if (options.interval) data.pause().cycle()
475       })
476     }
477
478     var old = $.fn.carousel
479
480     $.fn.carousel             = Plugin
481     $.fn.carousel.Constructor = Carousel
482
483
484     // CAROUSEL NO CONFLICT
485     // ====================
486
487     $.fn.carousel.noConflict = function () {
488       $.fn.carousel = old
489       return this
490     }
491
492
493     // CAROUSEL DATA-API
494     // =================
495
496     var clickHandler = function (e) {
497       var href
498       var $this   = $(this)
499       var $target = $($this.attr('data-target') || (href = $this.attr('href')) &&
    href.replace(/.*(?=#[^\s]+$)/, '')) // strip for ie7
500       if (!$target.hasClass('carousel')) return
501       var options = $.extend({}, $target.data(), $this.data())
502       var slideIndex = $this.attr('data-slide-to')
503       if (slideIndex) options.interval = false
504
505       Plugin.call($target, options)
506
507       if (slideIndex) {
508         $target.data('bs.carousel').to(slideIndex)
509       }
510
511       e.preventDefault()
512     }
513
514     $(document)
515       .on('click.bs.carousel.data-api', '[data-slide]', clickHandler)
516       .on('click.bs.carousel.data-api', '[data-slide-to]', clickHandler)
517
518     $(window).on('load', function () {
519       $('[data-ride="carousel"]').each(function () {
520         var $carousel = $(this)
521         Plugin.call($carousel, $carousel.data())
522       })
523     })
524
```

```
525   }(jQuery);
526
527   /* ========================================================================
528    * Bootstrap: collapse.js v3.3.6
529    * http://getbootstrap.com/javascript/#collapse
530    * ========================================================================
531    * Copyright 2011-2015 Twitter, Inc.
532    * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
533    * ======================================================================== */
534
535
536   +function ($) {
537     'use strict';
538
539     // COLLAPSE PUBLIC CLASS DEFINITION
540     // ================================
541
542     var Collapse = function (element, options) {
543       this.$element      = $(element)
544       this.options       = $.extend({}, Collapse.DEFAULTS, options)
545       this.$trigger      = $('[data-toggle="collapse"][href="#' + element.id + '"],' +
546                              '[data-toggle="collapse"][data-target="#' + element.id +
'"]')
547       this.transitioning = null
548
549       if (this.options.parent) {
550         this.$parent = this.getParent()
551       } else {
552         this.addAriaAndCollapsedClass(this.$element, this.$trigger)
553       }
554
555       if (this.options.toggle) this.toggle()
556     }
557
558     Collapse.VERSION  = '3.3.6'
559
560     Collapse.TRANSITION_DURATION = 350
561
562     Collapse.DEFAULTS = {
563       toggle: true
564     }
565
566     Collapse.prototype.dimension = function () {
567       var hasWidth = this.$element.hasClass('width')
568       return hasWidth ? 'width' : 'height'
569     }
570
571     Collapse.prototype.show = function () {
572       if (this.transitioning || this.$element.hasClass('in')) return
573
574       var activesData
575       var actives = this.$parent && this.$parent.children('.panel').children('.in,
.collapsing')
576
577       if (actives && actives.length) {
578         activesData = actives.data('bs.collapse')
579         if (activesData && activesData.transitioning) return
580       }
581
582       var startEvent = $.Event('show.bs.collapse')
```

```
583       this.$element.trigger(startEvent)
584       if (startEvent.isDefaultPrevented()) return
585
586       if (actives && actives.length) {
587         Plugin.call(actives, 'hide')
588         activesData || actives.data('bs.collapse', null)
589       }
590
591       var dimension = this.dimension()
592
593       this.$element
594         .removeClass('collapse')
595         .addClass('collapsing')[dimension](0)
596         .attr('aria-expanded', true)
597
598       this.$trigger
599         .removeClass('collapsed')
600         .attr('aria-expanded', true)
601
602       this.transitioning = 1
603
604       var complete = function () {
605         this.$element
606           .removeClass('collapsing')
607           .addClass('collapse in')[dimension]('')
608         this.transitioning = 0
609         this.$element
610           .trigger('shown.bs.collapse')
611       }
612
613       if (!$.support.transition) return complete.call(this)
614
615       var scrollSize = $.camelCase(['scroll', dimension].join('-'))
616
617       this.$element
618         .one('bsTransitionEnd', $.proxy(complete, this))
619         .emulateTransitionEnd(Collapse.TRANSITION_DURATION)[dimension](this.$element[0]
    [scrollSize])
620     }
621
622     Collapse.prototype.hide = function () {
623       if (this.transitioning || !this.$element.hasClass('in')) return
624
625       var startEvent = $.Event('hide.bs.collapse')
626       this.$element.trigger(startEvent)
627       if (startEvent.isDefaultPrevented()) return
628
629       var dimension = this.dimension()
630
631       this.$element[dimension](this.$element[dimension]())[0].offsetHeight
632
633       this.$element
634         .addClass('collapsing')
635         .removeClass('collapse in')
636         .attr('aria-expanded', false)
637
638       this.$trigger
639         .addClass('collapsed')
640         .attr('aria-expanded', false)
641
```

```
642       this.transitioning = 1
643
644       var complete = function () {
645         this.transitioning = 0
646         this.$element
647           .removeClass('collapsing')
648           .addClass('collapse')
649           .trigger('hidden.bs.collapse')
650       }
651
652       if (!$.support.transition) return complete.call(this)
653
654       this.$element
655         [dimension](0)
656         .one('bsTransitionEnd', $.proxy(complete, this))
657         .emulateTransitionEnd(Collapse.TRANSITION_DURATION)
658     }
659
660     Collapse.prototype.toggle = function () {
661       this[this.$element.hasClass('in') ? 'hide' : 'show']()
662     }
663
664     Collapse.prototype.getParent = function () {
665       return $(this.options.parent)
666         .find('[data-toggle="collapse"][data-parent="' + this.options.parent + '"]')
667         .each($.proxy(function (i, element) {
668           var $element = $(element)
669           this.addAriaAndCollapsedClass(getTargetFromTrigger($element), $element)
670         }, this))
671         .end()
672     }
673
674     Collapse.prototype.addAriaAndCollapsedClass = function ($element, $trigger) {
675       var isOpen = $element.hasClass('in')
676
677       $element.attr('aria-expanded', isOpen)
678       $trigger
679         .toggleClass('collapsed', !isOpen)
680         .attr('aria-expanded', isOpen)
681     }
682
683     function getTargetFromTrigger($trigger) {
684       var href
685       var target = $trigger.attr('data-target')
686         || (href = $trigger.attr('href')) && href.replace(/.*(?=#[^\s]+$)/, '') // strip
for ie7
687
688       return $(target)
689     }
690
691
692     // COLLAPSE PLUGIN DEFINITION
693     // ==========================
694
695     function Plugin(option) {
696       return this.each(function () {
697         var $this   = $(this)
698         var data    = $this.data('bs.collapse')
699         var options = $.extend({}, Collapse.DEFAULTS, $this.data(), typeof option ==
'object' && option)
```

```
700
701         if (!data && options.toggle && /show|hide/.test(option)) options.toggle = false
702         if (!data) $this.data('bs.collapse', (data = new Collapse(this, options)))
703         if (typeof option == 'string') data[option]()
704       })
705     }
706
707     var old = $.fn.collapse
708
709     $.fn.collapse             = Plugin
710     $.fn.collapse.Constructor = Collapse
711
712
713     // COLLAPSE NO CONFLICT
714     // ====================
715
716     $.fn.collapse.noConflict = function () {
717       $.fn.collapse = old
718       return this
719     }
720
721
722     // COLLAPSE DATA-API
723     // =================
724
725     $(document).on('click.bs.collapse.data-api', '[data-toggle="collapse"]', function (e)
    {
726       var $this   = $(this)
727
728       if (!$this.attr('data-target')) e.preventDefault()
729
730       var $target = getTargetFromTrigger($this)
731       var data    = $target.data('bs.collapse')
732       var option  = data ? 'toggle' : $this.data()
733
734       Plugin.call($target, option)
735     })
736
737   }(jQuery);
738
739   /* ========================================================================
740    * Bootstrap: dropdown.js v3.3.6
741    * http://getbootstrap.com/javascript/#dropdowns
742    * ========================================================================
743    * Copyright 2011-2015 Twitter, Inc.
744    * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
745    * ======================================================================== */
746
747
748   +function ($) {
749     'use strict';
750
751     // DROPDOWN CLASS DEFINITION
752     // =========================
753
754     var backdrop = '.dropdown-backdrop'
755     var toggle   = '[data-toggle="dropdown"]'
756     var Dropdown = function (element) {
757       $(element).on('click.bs.dropdown', this.toggle)
758     }
```

```
759
760     Dropdown.VERSION = '3.3.6'
761
762     function getParent($this) {
763       var selector = $this.attr('data-target')
764
765       if (!selector) {
766         selector = $this.attr('href')
767         selector = selector && /#[A-Za-z]/.test(selector) && selector.replace(/.*(?=#
[^\s]*$)/, '') // strip for ie7
768       }
769
770       var $parent = selector && $(selector)
771
772       return $parent && $parent.length ? $parent : $this.parent()
773     }
774
775     function clearMenus(e) {
776       if (e && e.which === 3) return
777       $(backdrop).remove()
778       $(toggle).each(function () {
779         var $this         = $(this)
780         var $parent       = getParent($this)
781         var relatedTarget = { relatedTarget: this }
782
783         if (!$parent.hasClass('open')) return
784
785         if (e && e.type == 'click' && /input|textarea/i.test(e.target.tagName) &&
$.contains($parent[0], e.target)) return
786
787         $parent.trigger(e = $.Event('hide.bs.dropdown', relatedTarget))
788
789         if (e.isDefaultPrevented()) return
790
791         $this.attr('aria-expanded', 'false')
792         $parent.removeClass('open').trigger($.Event('hidden.bs.dropdown', relatedTarget))
793       })
794     }
795
796     Dropdown.prototype.toggle = function (e) {
797       var $this = $(this)
798
799       if ($this.is('.disabled, :disabled')) return
800
801       var $parent  = getParent($this)
802       var isActive = $parent.hasClass('open')
803
804       clearMenus()
805
806       if (!isActive) {
807         if ('ontouchstart' in document.documentElement && !$parent.closest('.navbar-
nav').length) {
808           // if mobile we use a backdrop because click events don't delegate
809           $(document.createElement('div'))
810             .addClass('dropdown-backdrop')
811             .insertAfter($(this))
812             .on('click', clearMenus)
813         }
814
815         var relatedTarget = { relatedTarget: this }
816         $parent.trigger(e = $.Event('show.bs.dropdown', relatedTarget))
```

```
817
818         if (e.isDefaultPrevented()) return
819
820         $this
821           .trigger('focus')
822           .attr('aria-expanded', 'true')
823
824         $parent
825           .toggleClass('open')
826           .trigger($.Event('shown.bs.dropdown', relatedTarget))
827       }
828
829       return false
830     }
831
832     Dropdown.prototype.keydown = function (e) {
833       if (!/(38|40|27|32)/.test(e.which) || /input|textarea/i.test(e.target.tagName))
    return
834
835       var $this = $(this)
836
837       e.preventDefault()
838       e.stopPropagation()
839
840       if ($this.is('.disabled, :disabled')) return
841
842       var $parent  = getParent($this)
843       var isActive = $parent.hasClass('open')
844
845       if (!isActive && e.which != 27 || isActive && e.which == 27) {
846         if (e.which == 27) $parent.find(toggle).trigger('focus')
847         return $this.trigger('click')
848       }
849
850       var desc = ' li:not(.disabled):visible a'
851       var $items = $parent.find('.dropdown-menu' + desc)
852
853       if (!$items.length) return
854
855       var index = $items.index(e.target)
856
857       if (e.which == 38 && index > 0)                 index--         // up
858       if (e.which == 40 && index < $items.length - 1) index++         // down
859       if (!~index)                                    index = 0
860
861       $items.eq(index).trigger('focus')
862     }
863
864
865     // DROPDOWN PLUGIN DEFINITION
866     // ==========================
867
868     function Plugin(option) {
869       return this.each(function () {
870         var $this = $(this)
871         var data  = $this.data('bs.dropdown')
872
873         if (!data) $this.data('bs.dropdown', (data = new Dropdown(this)))
874         if (typeof option == 'string') data[option].call($this)
875       })
```

```
876     }
877
878     var old = $.fn.dropdown
879
880     $.fn.dropdown             = Plugin
881     $.fn.dropdown.Constructor = Dropdown
882
883
884     // DROPDOWN NO CONFLICT
885     // ====================
886
887     $.fn.dropdown.noConflict = function () {
888       $.fn.dropdown = old
889       return this
890     }
891
892
893     // APPLY TO STANDARD DROPDOWN ELEMENTS
894     // ===================================
895
896     $(document)
897       .on('click.bs.dropdown.data-api', clearMenus)
898       .on('click.bs.dropdown.data-api', '.dropdown form', function (e) {
    e.stopPropagation() })
899       .on('click.bs.dropdown.data-api', toggle, Dropdown.prototype.toggle)
900       .on('keydown.bs.dropdown.data-api', toggle, Dropdown.prototype.keydown)
901       .on('keydown.bs.dropdown.data-api', '.dropdown-menu', Dropdown.prototype.keydown)
902
903   }(jQuery);
904
905   /* ========================================================================
906    * Bootstrap: modal.js v3.3.6
907    * http://getbootstrap.com/javascript/#modals
908    * ========================================================================
909    * Copyright 2011-2015 Twitter, Inc.
910    * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
911    * ======================================================================== */
912
913
914   +function ($) {
915     'use strict';
916
917     // MODAL CLASS DEFINITION
918     // ======================
919
920     var Modal = function (element, options) {
921       this.options             = options
922       this.$body               = $(document.body)
923       this.$element            = $(element)
924       this.$dialog             = this.$element.find('.modal-dialog')
925       this.$backdrop           = null
926       this.isShown             = null
927       this.originalBodyPad     = null
928       this.scrollbarWidth      = 0
929       this.ignoreBackdropClick = false
930
931       if (this.options.remote) {
932         this.$element
933           .find('.modal-content')
934           .load(this.options.remote, $.proxy(function () {
```

```
935             this.$element.trigger('loaded.bs.modal')
936           }, this))
937       }
938     }
939
940     Modal.VERSION  = '3.3.6'
941
942     Modal.TRANSITION_DURATION = 300
943     Modal.BACKDROP_TRANSITION_DURATION = 150
944
945     Modal.DEFAULTS = {
946       backdrop: true,
947       keyboard: true,
948       show: true
949     }
950
951     Modal.prototype.toggle = function (_relatedTarget) {
952       return this.isShown ? this.hide() : this.show(_relatedTarget)
953     }
954
955     Modal.prototype.show = function (_relatedTarget) {
956       var that = this
957       var e    = $.Event('show.bs.modal', { relatedTarget: _relatedTarget })
958
959       this.$element.trigger(e)
960
961       if (this.isShown || e.isDefaultPrevented()) return
962
963       this.isShown = true
964
965       this.checkScrollbar()
966       this.setScrollbar()
967       this.$body.addClass('modal-open')
968
969       this.escape()
970       this.resize()
971
972       this.$element.on('click.dismiss.bs.modal', '[data-dismiss="modal"]',
    $.proxy(this.hide, this))
973
974       this.$dialog.on('mousedown.dismiss.bs.modal', function () {
975         that.$element.one('mouseup.dismiss.bs.modal', function (e) {
976           if ($(e.target).is(that.$element)) that.ignoreBackdropClick = true
977         })
978       })
979
980       this.backdrop(function () {
981         var transition = $.support.transition && that.$element.hasClass('fade')
982
983         if (!that.$element.parent().length) {
984           that.$element.appendTo(that.$body) // don't move modals dom position
985         }
986
987         that.$element
988           .show()
989           .scrollTop(0)
990
991         that.adjustDialog()
992
993         if (transition) {
```

```
994           that.$element[0].offsetWidth // force reflow
995         }
996
997         that.$element.addClass('in')
998
999         that.enforceFocus()
1000
1001        var e = $.Event('shown.bs.modal', { relatedTarget: _relatedTarget })
1002
1003        transition ?
1004          that.$dialog // wait for modal to slide in
1005            .one('bsTransitionEnd', function () {
1006              that.$element.trigger('focus').trigger(e)
1007            })
1008            .emulateTransitionEnd(Modal.TRANSITION_DURATION) :
1009          that.$element.trigger('focus').trigger(e)
1010      })
1011    }
1012
1013    Modal.prototype.hide = function (e) {
1014      if (e) e.preventDefault()
1015
1016      e = $.Event('hide.bs.modal')
1017
1018      this.$element.trigger(e)
1019
1020      if (!this.isShown || e.isDefaultPrevented()) return
1021
1022      this.isShown = false
1023
1024      this.escape()
1025      this.resize()
1026
1027      $(document).off('focusin.bs.modal')
1028
1029      this.$element
1030        .removeClass('in')
1031        .off('click.dismiss.bs.modal')
1032        .off('mouseup.dismiss.bs.modal')
1033
1034      this.$dialog.off('mousedown.dismiss.bs.modal')
1035
1036      $.support.transition && this.$element.hasClass('fade') ?
1037        this.$element
1038          .one('bsTransitionEnd', $.proxy(this.hideModal, this))
1039          .emulateTransitionEnd(Modal.TRANSITION_DURATION) :
1040        this.hideModal()
1041    }
1042
1043    Modal.prototype.enforceFocus = function () {
1044      $(document)
1045        .off('focusin.bs.modal') // guard against infinite focus loop
1046        .on('focusin.bs.modal', $.proxy(function (e) {
1047          if (this.$element[0] !== e.target && !this.$element.has(e.target).length) {
1048            this.$element.trigger('focus')
1049          }
1050        }, this))
1051    }
1052
1053    Modal.prototype.escape = function () {
```

```
1054        if (this.isShown && this.options.keyboard) {
1055          this.$element.on('keydown.dismiss.bs.modal', $.proxy(function (e) {
1056            e.which == 27 && this.hide()
1057          }, this))
1058        } else if (!this.isShown) {
1059          this.$element.off('keydown.dismiss.bs.modal')
1060        }
1061      }
1062
1063      Modal.prototype.resize = function () {
1064        if (this.isShown) {
1065          $(window).on('resize.bs.modal', $.proxy(this.handleUpdate, this))
1066        } else {
1067          $(window).off('resize.bs.modal')
1068        }
1069      }
1070
1071      Modal.prototype.hideModal = function () {
1072        var that = this
1073        this.$element.hide()
1074        this.backdrop(function () {
1075          that.$body.removeClass('modal-open')
1076          that.resetAdjustments()
1077          that.resetScrollbar()
1078          that.$element.trigger('hidden.bs.modal')
1079        })
1080      }
1081
1082      Modal.prototype.removeBackdrop = function () {
1083        this.$backdrop && this.$backdrop.remove()
1084        this.$backdrop = null
1085      }
1086
1087      Modal.prototype.backdrop = function (callback) {
1088        var that = this
1089        var animate = this.$element.hasClass('fade') ? 'fade' : ''
1090
1091        if (this.isShown && this.options.backdrop) {
1092          var doAnimate = $.support.transition && animate
1093
1094          this.$backdrop = $(document.createElement('div'))
1095            .addClass('modal-backdrop ' + animate)
1096            .appendTo(this.$body)
1097
1098          this.$element.on('click.dismiss.bs.modal', $.proxy(function (e) {
1099            if (this.ignoreBackdropClick) {
1100              this.ignoreBackdropClick = false
1101              return
1102            }
1103            if (e.target !== e.currentTarget) return
1104            this.options.backdrop == 'static'
1105              ? this.$element[0].focus()
1106              : this.hide()
1107          }, this))
1108
1109          if (doAnimate) this.$backdrop[0].offsetWidth // force reflow
1110
1111          this.$backdrop.addClass('in')
1112
1113          if (!callback) return
```

```
1114
1115          doAnimate ?
1116            this.$backdrop
1117              .one('bsTransitionEnd', callback)
1118              .emulateTransitionEnd(Modal.BACKDROP_TRANSITION_DURATION) :
1119            callback()
1120
1121        } else if (!this.isShown && this.$backdrop) {
1122          this.$backdrop.removeClass('in')
1123
1124          var callbackRemove = function () {
1125            that.removeBackdrop()
1126            callback && callback()
1127          }
1128          $.support.transition && this.$element.hasClass('fade') ?
1129            this.$backdrop
1130              .one('bsTransitionEnd', callbackRemove)
1131              .emulateTransitionEnd(Modal.BACKDROP_TRANSITION_DURATION) :
1132            callbackRemove()
1133
1134        } else if (callback) {
1135          callback()
1136        }
1137      }
1138
1139      // these following methods are used to handle overflowing modals
1140
1141      Modal.prototype.handleUpdate = function () {
1142        this.adjustDialog()
1143      }
1144
1145      Modal.prototype.adjustDialog = function () {
1146        var modalIsOverflowing = this.$element[0].scrollHeight >
     document.documentElement.clientHeight
1147
1148        this.$element.css({
1149          paddingLeft:  !this.bodyIsOverflowing && modalIsOverflowing ? this.scrollbarWidth
     : '',
1150          paddingRight: this.bodyIsOverflowing && !modalIsOverflowing ? this.scrollbarWidth
     : ''
1151        })
1152      }
1153
1154      Modal.prototype.resetAdjustments = function () {
1155        this.$element.css({
1156          paddingLeft: '',
1157          paddingRight: ''
1158        })
1159      }
1160
1161      Modal.prototype.checkScrollbar = function () {
1162        var fullWindowWidth = window.innerWidth
1163        if (!fullWindowWidth) { // workaround for missing window.innerWidth in IE8
1164          var documentElementRect = document.documentElement.getBoundingClientRect()
1165          fullWindowWidth = documentElementRect.right - Math.abs(documentElementRect.left)
1166        }
1167        this.bodyIsOverflowing = document.body.clientWidth < fullWindowWidth
1168        this.scrollbarWidth = this.measureScrollbar()
1169      }
1170
1171      Modal.prototype.setScrollbar = function () {
```

```
1172        var bodyPad = parseInt((this.$body.css('padding-right') || 0), 10)
1173        this.originalBodyPad = document.body.style.paddingRight || ''
1174        if (this.bodyIsOverflowing) this.$body.css('padding-right', bodyPad +
    this.scrollbarWidth)
1175      }
1176
1177      Modal.prototype.resetScrollbar = function () {
1178        this.$body.css('padding-right', this.originalBodyPad)
1179      }
1180
1181      Modal.prototype.measureScrollbar = function () { // thx walsh
1182        var scrollDiv = document.createElement('div')
1183        scrollDiv.className = 'modal-scrollbar-measure'
1184        this.$body.append(scrollDiv)
1185        var scrollbarWidth = scrollDiv.offsetWidth - scrollDiv.clientWidth
1186        this.$body[0].removeChild(scrollDiv)
1187        return scrollbarWidth
1188      }
1189
1190
1191      // MODAL PLUGIN DEFINITION
1192      // =======================
1193
1194      function Plugin(option, _relatedTarget) {
1195        return this.each(function () {
1196          var $this   = $(this)
1197          var data    = $this.data('bs.modal')
1198          var options = $.extend({}, Modal.DEFAULTS, $this.data(), typeof option ==
    'object' && option)
1199
1200          if (!data) $this.data('bs.modal', (data = new Modal(this, options)))
1201          if (typeof option == 'string') data[option](_relatedTarget)
1202          else if (options.show) data.show(_relatedTarget)
1203        })
1204      }
1205
1206      var old = $.fn.modal
1207
1208      $.fn.modal             = Plugin
1209      $.fn.modal.Constructor = Modal
1210
1211
1212      // MODAL NO CONFLICT
1213      // =================
1214
1215      $.fn.modal.noConflict = function () {
1216        $.fn.modal = old
1217        return this
1218      }
1219
1220
1221      // MODAL DATA-API
1222      // ==============
1223
1224      $(document).on('click.bs.modal.data-api', '[data-toggle="modal"]', function (e) {
1225        var $this   = $(this)
1226        var href    = $this.attr('href')
1227        var $target = $($this.attr('data-target') || (href && href.replace(/.*(?=#
    [^\s]+$)/, ''))) // strip for ie7
1228        var option  = $target.data('bs.modal') ? 'toggle' : $.extend({ remote:
    !/#/.test(href) && href }, $target.data(), $this.data())
```

```
1229
1230        if ($this.is('a')) e.preventDefault()
1231
1232        $target.one('show.bs.modal', function (showEvent) {
1233          if (showEvent.isDefaultPrevented()) return // only register focus restorer if modal will actually get shown
1234          $target.one('hidden.bs.modal', function () {
1235            $this.is(':visible') && $this.trigger('focus')
1236          })
1237        })
1238        Plugin.call($target, option, this)
1239      })
1240
1241    }(jQuery);
1242
1243    /* ========================================================================
1244     * Bootstrap: tooltip.js v3.3.6
1245     * http://getbootstrap.com/javascript/#tooltip
1246     * Inspired by the original jQuery.tipsy by Jason Frame
1247     * ========================================================================
1248     * Copyright 2011-2015 Twitter, Inc.
1249     * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
1250     * ======================================================================== */
1251
1252
1253    +function ($) {
1254      'use strict';
1255
1256      // TOOLTIP PUBLIC CLASS DEFINITION
1257      // ===============================
1258
1259      var Tooltip = function (element, options) {
1260        this.type       = null
1261        this.options    = null
1262        this.enabled    = null
1263        this.timeout    = null
1264        this.hoverState = null
1265        this.$element   = null
1266        this.inState    = null
1267
1268        this.init('tooltip', element, options)
1269      }
1270
1271      Tooltip.VERSION  = '3.3.6'
1272
1273      Tooltip.TRANSITION_DURATION = 150
1274
1275      Tooltip.DEFAULTS = {
1276        animation: true,
1277        placement: 'top',
1278        selector: false,
1279        template: '<div class="tooltip" role="tooltip"><div class="tooltip-arrow"></div><div class="tooltip-inner"></div></div>',
1280        trigger: 'hover focus',
1281        title: '',
1282        delay: 0,
1283        html: false,
1284        container: false,
1285        viewport: {
1286          selector: 'body',
```

```
1287          padding: 0
1288        }
1289      }
1290
1291      Tooltip.prototype.init = function (type, element, options) {
1292        this.enabled   = true
1293        this.type      = type
1294        this.$element  = $(element)
1295        this.options   = this.getOptions(options)
1296        this.$viewport = this.options.viewport && $($.isFunction(this.options.viewport) ? this.options.viewport.call(this, this.$element) : (this.options.viewport.selector || this.options.viewport))
1297        this.inState   = { click: false, hover: false, focus: false }
1298
1299        if (this.$element[0] instanceof document.constructor && !this.options.selector) {
1300          throw new Error('`selector` option must be specified when initializing ' + this.type + ' on the window.document object!')
1301        }
1302
1303        var triggers = this.options.trigger.split(' ')
1304
1305        for (var i = triggers.length; i--;) {
1306          var trigger = triggers[i]
1307
1308          if (trigger == 'click') {
1309            this.$element.on('click.' + this.type, this.options.selector, $.proxy(this.toggle, this))
1310          } else if (trigger != 'manual') {
1311            var eventIn  = trigger == 'hover' ? 'mouseenter' : 'focusin'
1312            var eventOut = trigger == 'hover' ? 'mouseleave' : 'focusout'
1313
1314            this.$element.on(eventIn  + '.' + this.type, this.options.selector, $.proxy(this.enter, this))
1315            this.$element.on(eventOut + '.' + this.type, this.options.selector, $.proxy(this.leave, this))
1316          }
1317        }
1318
1319        this.options.selector ?
1320          (this._options = $.extend({}, this.options, { trigger: 'manual', selector: '' })) :
1321          this.fixTitle()
1322      }
1323
1324      Tooltip.prototype.getDefaults = function () {
1325        return Tooltip.DEFAULTS
1326      }
1327
1328      Tooltip.prototype.getOptions = function (options) {
1329        options = $.extend({}, this.getDefaults(), this.$element.data(), options)
1330
1331        if (options.delay && typeof options.delay == 'number') {
1332          options.delay = {
1333            show: options.delay,
1334            hide: options.delay
1335          }
1336        }
1337
1338        return options
1339      }
1340
```

```
1341      Tooltip.prototype.getDelegateOptions = function () {
1342        var options  = {}
1343        var defaults = this.getDefaults()
1344
1345        this._options && $.each(this._options, function (key, value) {
1346          if (defaults[key] != value) options[key] = value
1347        })
1348
1349        return options
1350      }
1351
1352      Tooltip.prototype.enter = function (obj) {
1353        var self = obj instanceof this.constructor ?
1354          obj : $(obj.currentTarget).data('bs.' + this.type)
1355
1356        if (!self) {
1357          self = new this.constructor(obj.currentTarget, this.getDelegateOptions())
1358          $(obj.currentTarget).data('bs.' + this.type, self)
1359        }
1360
1361        if (obj instanceof $.Event) {
1362          self.inState[obj.type == 'focusin' ? 'focus' : 'hover'] = true
1363        }
1364
1365        if (self.tip().hasClass('in') || self.hoverState == 'in') {
1366          self.hoverState = 'in'
1367          return
1368        }
1369
1370        clearTimeout(self.timeout)
1371
1372        self.hoverState = 'in'
1373
1374        if (!self.options.delay || !self.options.delay.show) return self.show()
1375
1376        self.timeout = setTimeout(function () {
1377          if (self.hoverState == 'in') self.show()
1378        }, self.options.delay.show)
1379      }
1380
1381      Tooltip.prototype.isInStateTrue = function () {
1382        for (var key in this.inState) {
1383          if (this.inState[key]) return true
1384        }
1385
1386        return false
1387      }
1388
1389      Tooltip.prototype.leave = function (obj) {
1390        var self = obj instanceof this.constructor ?
1391          obj : $(obj.currentTarget).data('bs.' + this.type)
1392
1393        if (!self) {
1394          self = new this.constructor(obj.currentTarget, this.getDelegateOptions())
1395          $(obj.currentTarget).data('bs.' + this.type, self)
1396        }
1397
1398        if (obj instanceof $.Event) {
1399          self.inState[obj.type == 'focusout' ? 'focus' : 'hover'] = false
1400        }
```

```
1401
1402        if (self.isInStateTrue()) return
1403
1404        clearTimeout(self.timeout)
1405
1406        self.hoverState = 'out'
1407
1408        if (!self.options.delay || !self.options.delay.hide) return self.hide()
1409
1410        self.timeout = setTimeout(function () {
1411          if (self.hoverState == 'out') self.hide()
1412        }, self.options.delay.hide)
1413      }
1414
1415      Tooltip.prototype.show = function () {
1416        var e = $.Event('show.bs.' + this.type)
1417
1418        if (this.hasContent() && this.enabled) {
1419          this.$element.trigger(e)
1420
1421          var inDom = $.contains(this.$element[0].ownerDocument.documentElement,
    this.$element[0])
1422          if (e.isDefaultPrevented() || !inDom) return
1423          var that = this
1424
1425          var $tip = this.tip()
1426
1427          var tipId = this.getUID(this.type)
1428
1429          this.setContent()
1430          $tip.attr('id', tipId)
1431          this.$element.attr('aria-describedby', tipId)
1432
1433          if (this.options.animation) $tip.addClass('fade')
1434
1435          var placement = typeof this.options.placement == 'function' ?
1436            this.options.placement.call(this, $tip[0], this.$element[0]) :
1437            this.options.placement
1438
1439          var autoToken = /\s?auto?\s?/i
1440          var autoPlace = autoToken.test(placement)
1441          if (autoPlace) placement = placement.replace(autoToken, '') || 'top'
1442
1443          $tip
1444            .detach()
1445            .css({ top: 0, left: 0, display: 'block' })
1446            .addClass(placement)
1447            .data('bs.' + this.type, this)
1448
1449          this.options.container ? $tip.appendTo(this.options.container) :
    $tip.insertAfter(this.$element)
1450          this.$element.trigger('inserted.bs.' + this.type)
1451
1452          var pos          = this.getPosition()
1453          var actualWidth  = $tip[0].offsetWidth
1454          var actualHeight = $tip[0].offsetHeight
1455
1456          if (autoPlace) {
1457            var orgPlacement = placement
1458            var viewportDim = this.getPosition(this.$viewport)
```

```
1459
1460          placement = placement == 'bottom' && pos.bottom + actualHeight > viewportDim.bottom ? 'top'    :
1461                      placement == 'top'    && pos.top    - actualHeight < viewportDim.top    ? 'bottom' :
1462                      placement == 'right'  && pos.right  + actualWidth  > viewportDim.width  ? 'left'   :
1463                      placement == 'left'   && pos.left   - actualWidth  < viewportDim.left   ? 'right'  :
1464                      placement
1465
1466          $tip
1467            .removeClass(orgPlacement)
1468            .addClass(placement)
1469        }
1470
1471        var calculatedOffset = this.getCalculatedOffset(placement, pos, actualWidth, actualHeight)
1472
1473        this.applyPlacement(calculatedOffset, placement)
1474
1475        var complete = function () {
1476          var prevHoverState = that.hoverState
1477          that.$element.trigger('shown.bs.' + that.type)
1478          that.hoverState = null
1479
1480          if (prevHoverState == 'out') that.leave(that)
1481        }
1482
1483        $.support.transition && this.$tip.hasClass('fade') ?
1484          $tip
1485            .one('bsTransitionEnd', complete)
1486            .emulateTransitionEnd(Tooltip.TRANSITION_DURATION) :
1487          complete()
1488      }
1489    }
1490
1491    Tooltip.prototype.applyPlacement = function (offset, placement) {
1492      var $tip   = this.tip()
1493      var width  = $tip[0].offsetWidth
1494      var height = $tip[0].offsetHeight
1495
1496      // manually read margins because getBoundingClientRect includes difference
1497      var marginTop = parseInt($tip.css('margin-top'), 10)
1498      var marginLeft = parseInt($tip.css('margin-left'), 10)
1499
1500      // we must check for NaN for ie 8/9
1501      if (isNaN(marginTop))  marginTop  = 0
1502      if (isNaN(marginLeft)) marginLeft = 0
1503
1504      offset.top  += marginTop
1505      offset.left += marginLeft
1506
1507      // $.fn.offset doesn't round pixel values
1508      // so we use setOffset directly with our own function B-0
1509      $.offset.setOffset($tip[0], $.extend({
1510        using: function (props) {
1511          $tip.css({
1512            top: Math.round(props.top),
1513            left: Math.round(props.left)
1514          })
```

```
1515          }
1516        }, offset), 0)
1517
1518        $tip.addClass('in')
1519
1520        // check to see if placing tip in new offset caused the tip to resize itself
1521        var actualWidth  = $tip[0].offsetWidth
1522        var actualHeight = $tip[0].offsetHeight
1523
1524        if (placement == 'top' && actualHeight != height) {
1525          offset.top = offset.top + height - actualHeight
1526        }
1527
1528        var delta = this.getViewportAdjustedDelta(placement, offset, actualWidth,
    actualHeight)
1529
1530        if (delta.left) offset.left += delta.left
1531        else offset.top += delta.top
1532
1533        var isVertical          = /top|bottom/.test(placement)
1534        var arrowDelta          = isVertical ? delta.left * 2 - width + actualWidth :
    delta.top * 2 - height + actualHeight
1535        var arrowOffsetPosition = isVertical ? 'offsetWidth' : 'offsetHeight'
1536
1537        $tip.offset(offset)
1538        this.replaceArrow(arrowDelta, $tip[0][arrowOffsetPosition], isVertical)
1539      }
1540
1541      Tooltip.prototype.replaceArrow = function (delta, dimension, isVertical) {
1542        this.arrow()
1543          .css(isVertical ? 'left' : 'top', 50 * (1 - delta / dimension) + '%')
1544          .css(isVertical ? 'top' : 'left', '')
1545      }
1546
1547      Tooltip.prototype.setContent = function () {
1548        var $tip  = this.tip()
1549        var title = this.getTitle()
1550
1551        $tip.find('.tooltip-inner')[this.options.html ? 'html' : 'text'](title)
1552        $tip.removeClass('fade in top bottom left right')
1553      }
1554
1555      Tooltip.prototype.hide = function (callback) {
1556        var that = this
1557        var $tip = $(this.$tip)
1558        var e    = $.Event('hide.bs.' + this.type)
1559
1560        function complete() {
1561          if (that.hoverState != 'in') $tip.detach()
1562          that.$element
1563            .removeAttr('aria-describedby')
1564            .trigger('hidden.bs.' + that.type)
1565          callback && callback()
1566        }
1567
1568        this.$element.trigger(e)
1569
1570        if (e.isDefaultPrevented()) return
1571
1572        $tip.removeClass('in')
```

```
1573
1574        $.support.transition && $tip.hasClass('fade') ?
1575          $tip
1576            .one('bsTransitionEnd', complete)
1577            .emulateTransitionEnd(Tooltip.TRANSITION_DURATION) :
1578          complete()
1579
1580        this.hoverState = null
1581
1582        return this
1583      }
1584
1585      Tooltip.prototype.fixTitle = function () {
1586        var $e = this.$element
1587        if ($e.attr('title') || typeof $e.attr('data-original-title') != 'string') {
1588          $e.attr('data-original-title', $e.attr('title') || '').attr('title', '')
1589        }
1590      }
1591
1592      Tooltip.prototype.hasContent = function () {
1593        return this.getTitle()
1594      }
1595
1596      Tooltip.prototype.getPosition = function ($element) {
1597        $element   = $element || this.$element
1598
1599        var el     = $element[0]
1600        var isBody = el.tagName == 'BODY'
1601
1602        var elRect    = el.getBoundingClientRect()
1603        if (elRect.width == null) {
1604          // width and height are missing in IE8, so compute them manually; see
      https://github.com/twbs/bootstrap/issues/14093
1605          elRect = $.extend({}, elRect, { width: elRect.right - elRect.left, height:
      elRect.bottom - elRect.top })
1606        }
1607        var elOffset  = isBody ? { top: 0, left: 0 } : $element.offset()
1608        var scroll    = { scroll: isBody ? document.documentElement.scrollTop ||
      document.body.scrollTop : $element.scrollTop() }
1609        var outerDims = isBody ? { width: $(window).width(), height: $(window).height() } :
      null
1610
1611        return $.extend({}, elRect, scroll, outerDims, elOffset)
1612      }
1613
1614      Tooltip.prototype.getCalculatedOffset = function (placement, pos, actualWidth,
      actualHeight) {
1615        return placement == 'bottom' ? { top: pos.top + pos.height,   left: pos.left +
      pos.width / 2 - actualWidth / 2 } :
1616               placement == 'top'    ? { top: pos.top - actualHeight, left: pos.left +
      pos.width / 2 - actualWidth / 2 } :
1617               placement == 'left'   ? { top: pos.top + pos.height / 2 - actualHeight / 2,
      left: pos.left - actualWidth } :
1618            /* placement == 'right' */ { top: pos.top + pos.height / 2 - actualHeight / 2,
      left: pos.left + pos.width }
1619
1620      }
1621
1622      Tooltip.prototype.getViewportAdjustedDelta = function (placement, pos, actualWidth,
      actualHeight) {
1623        var delta = { top: 0, left: 0 }
1624        if (!this.$viewport) return delta
```

```
1625
1626        var viewportPadding = this.options.viewport && this.options.viewport.padding || 0
1627        var viewportDimensions = this.getPosition(this.$viewport)
1628
1629        if (/right|left/.test(placement)) {
1630          var topEdgeOffset    = pos.top - viewportPadding - viewportDimensions.scroll
1631          var bottomEdgeOffset = pos.top + viewportPadding - viewportDimensions.scroll +
      actualHeight
1632          if (topEdgeOffset < viewportDimensions.top) { // top overflow
1633            delta.top = viewportDimensions.top - topEdgeOffset
1634          } else if (bottomEdgeOffset > viewportDimensions.top + viewportDimensions.height)
      { // bottom overflow
1635            delta.top = viewportDimensions.top + viewportDimensions.height -
      bottomEdgeOffset
1636          }
1637        } else {
1638          var leftEdgeOffset  = pos.left - viewportPadding
1639          var rightEdgeOffset = pos.left + viewportPadding + actualWidth
1640          if (leftEdgeOffset < viewportDimensions.left) { // left overflow
1641            delta.left = viewportDimensions.left - leftEdgeOffset
1642          } else if (rightEdgeOffset > viewportDimensions.right) { // right overflow
1643            delta.left = viewportDimensions.left + viewportDimensions.width -
      rightEdgeOffset
1644          }
1645        }
1646
1647        return delta
1648      }
1649
1650      Tooltip.prototype.getTitle = function () {
1651        var title
1652        var $e = this.$element
1653        var o  = this.options
1654
1655        title = $e.attr('data-original-title')
1656          || (typeof o.title == 'function' ? o.title.call($e[0]) :  o.title)
1657
1658        return title
1659      }
1660
1661      Tooltip.prototype.getUID = function (prefix) {
1662        do prefix += ~~(Math.random() * 1000000)
1663        while (document.getElementById(prefix))
1664        return prefix
1665      }
1666
1667      Tooltip.prototype.tip = function () {
1668        if (!this.$tip) {
1669          this.$tip = $(this.options.template)
1670          if (this.$tip.length != 1) {
1671            throw new Error(this.type + ' `template` option must consist of exactly 1 top-
      level element!')
1672          }
1673        }
1674        return this.$tip
1675      }
1676
1677      Tooltip.prototype.arrow = function () {
1678        return (this.$arrow = this.$arrow || this.tip().find('.tooltip-arrow'))
1679      }
1680
```

```
1681      Tooltip.prototype.enable = function () {
1682        this.enabled = true
1683      }
1684
1685      Tooltip.prototype.disable = function () {
1686        this.enabled = false
1687      }
1688
1689      Tooltip.prototype.toggleEnabled = function () {
1690        this.enabled = !this.enabled
1691      }
1692
1693      Tooltip.prototype.toggle = function (e) {
1694        var self = this
1695        if (e) {
1696          self = $(e.currentTarget).data('bs.' + this.type)
1697          if (!self) {
1698            self = new this.constructor(e.currentTarget, this.getDelegateOptions())
1699            $(e.currentTarget).data('bs.' + this.type, self)
1700          }
1701        }
1702
1703        if (e) {
1704          self.inState.click = !self.inState.click
1705          if (self.isInStateTrue()) self.enter(self)
1706          else self.leave(self)
1707        } else {
1708          self.tip().hasClass('in') ? self.leave(self) : self.enter(self)
1709        }
1710      }
1711
1712      Tooltip.prototype.destroy = function () {
1713        var that = this
1714        clearTimeout(this.timeout)
1715        this.hide(function () {
1716          that.$element.off('.' + that.type).removeData('bs.' + that.type)
1717          if (that.$tip) {
1718            that.$tip.detach()
1719          }
1720          that.$tip = null
1721          that.$arrow = null
1722          that.$viewport = null
1723        })
1724      }
1725
1726
1727      // TOOLTIP PLUGIN DEFINITION
1728      // =========================
1729
1730      function Plugin(option) {
1731        return this.each(function () {
1732          var $this   = $(this)
1733          var data    = $this.data('bs.tooltip')
1734          var options = typeof option == 'object' && option
1735
1736          if (!data && /destroy|hide/.test(option)) return
1737          if (!data) $this.data('bs.tooltip', (data = new Tooltip(this, options)))
1738          if (typeof option == 'string') data[option]()
1739        })
1740      }
```

```
1741
1742      var old = $.fn.tooltip
1743
1744      $.fn.tooltip             = Plugin
1745      $.fn.tooltip.Constructor = Tooltip
1746
1747
1748      // TOOLTIP NO CONFLICT
1749      // ===================
1750
1751      $.fn.tooltip.noConflict = function () {
1752        $.fn.tooltip = old
1753        return this
1754      }
1755
1756    }(jQuery);
1757
1758    /* ========================================================================
1759     * Bootstrap: popover.js v3.3.6
1760     * http://getbootstrap.com/javascript/#popovers
1761     * ========================================================================
1762     * Copyright 2011-2015 Twitter, Inc.
1763     * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
1764     * ======================================================================== */
1765
1766
1767    +function ($) {
1768      'use strict';
1769
1770      // POPOVER PUBLIC CLASS DEFINITION
1771      // ===============================
1772
1773      var Popover = function (element, options) {
1774        this.init('popover', element, options)
1775      }
1776
1777      if (!$.fn.tooltip) throw new Error('Popover requires tooltip.js')
1778
1779      Popover.VERSION  = '3.3.6'
1780
1781      Popover.DEFAULTS = $.extend({}, $.fn.tooltip.Constructor.DEFAULTS, {
1782        placement: 'right',
1783        trigger: 'click',
1784        content: '',
1785        template: '<div class="popover" role="tooltip"><div class="arrow"></div><h3
     class="popover-title"></h3><div class="popover-content"></div></div>'
1786      })
1787
1788
1789      // NOTE: POPOVER EXTENDS tooltip.js
1790      // ================================
1791
1792      Popover.prototype = $.extend({}, $.fn.tooltip.Constructor.prototype)
1793
1794      Popover.prototype.constructor = Popover
1795
1796      Popover.prototype.getDefaults = function () {
1797        return Popover.DEFAULTS
1798      }
1799
```

```
1800      Popover.prototype.setContent = function () {
1801        var $tip    = this.tip()
1802        var title   = this.getTitle()
1803        var content = this.getContent()
1804
1805        $tip.find('.popover-title')[this.options.html ? 'html' : 'text'](title)
1806        $tip.find('.popover-content').children().detach().end()[ // we use append for html
    objects to maintain js events
1807          this.options.html ? (typeof content == 'string' ? 'html' : 'append') : 'text'
1808        ](content)
1809
1810        $tip.removeClass('fade top bottom left right in')
1811
1812        // IE8 doesn't accept hiding via the `:empty` pseudo selector, we have to do
1813        // this manually by checking the contents.
1814        if (!$tip.find('.popover-title').html()) $tip.find('.popover-title').hide()
1815      }
1816
1817      Popover.prototype.hasContent = function () {
1818        return this.getTitle() || this.getContent()
1819      }
1820
1821      Popover.prototype.getContent = function () {
1822        var $e = this.$element
1823        var o  = this.options
1824
1825        return $e.attr('data-content')
1826          || (typeof o.content == 'function' ?
1827                o.content.call($e[0]) :
1828                o.content)
1829      }
1830
1831      Popover.prototype.arrow = function () {
1832        return (this.$arrow = this.$arrow || this.tip().find('.arrow'))
1833      }
1834
1835
1836      // POPOVER PLUGIN DEFINITION
1837      // =========================
1838
1839      function Plugin(option) {
1840        return this.each(function () {
1841          var $this   = $(this)
1842          var data    = $this.data('bs.popover')
1843          var options = typeof option == 'object' && option
1844
1845          if (!data && /destroy|hide/.test(option)) return
1846          if (!data) $this.data('bs.popover', (data = new Popover(this, options)))
1847          if (typeof option == 'string') data[option]()
1848        })
1849      }
1850
1851      var old = $.fn.popover
1852
1853      $.fn.popover             = Plugin
1854      $.fn.popover.Constructor = Popover
1855
1856
1857      // POPOVER NO CONFLICT
1858      // ===================
```

```
1859
1860      $.fn.popover.noConflict = function () {
1861        $.fn.popover = old
1862        return this
1863      }
1864
1865    }(jQuery);
1866
1867    /* ========================================================================
1868     * Bootstrap: scrollspy.js v3.3.6
1869     * http://getbootstrap.com/javascript/#scrollspy
1870     * ========================================================================
1871     * Copyright 2011-2015 Twitter, Inc.
1872     * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
1873     * ======================================================================== */
1874
1875
1876    +function ($) {
1877      'use strict';
1878
1879      // SCROLLSPY CLASS DEFINITION
1880      // ==========================
1881
1882      function ScrollSpy(element, options) {
1883        this.$body          = $(document.body)
1884        this.$scrollElement = $(element).is(document.body) ? $(window) : $(element)
1885        this.options        = $.extend({}, ScrollSpy.DEFAULTS, options)
1886        this.selector       = (this.options.target || '') + ' .nav li > a'
1887        this.offsets        = []
1888        this.targets        = []
1889        this.activeTarget   = null
1890        this.scrollHeight   = 0
1891
1892        this.$scrollElement.on('scroll.bs.scrollspy', $.proxy(this.process, this))
1893        this.refresh()
1894        this.process()
1895      }
1896
1897      ScrollSpy.VERSION  = '3.3.6'
1898
1899      ScrollSpy.DEFAULTS = {
1900        offset: 10
1901      }
1902
1903      ScrollSpy.prototype.getScrollHeight = function () {
1904        return this.$scrollElement[0].scrollHeight || Math.max(this.$body[0].scrollHeight,
    document.documentElement.scrollHeight)
1905      }
1906
1907      ScrollSpy.prototype.refresh = function () {
1908        var that          = this
1909        var offsetMethod  = 'offset'
1910        var offsetBase    = 0
1911
1912        this.offsets      = []
1913        this.targets      = []
1914        this.scrollHeight = this.getScrollHeight()
1915
1916        if (!$.isWindow(this.$scrollElement[0])) {
1917          offsetMethod = 'position'
```

```
1918          offsetBase   = this.$scrollElement.scrollTop()
1919        }
1920
1921        this.$body
1922          .find(this.selector)
1923          .map(function () {
1924            var $el   = $(this)
1925            var href  = $el.data('target') || $el.attr('href')
1926            var $href = /^#./.test(href) && $(href)
1927
1928            return ($href
1929              && $href.length
1930              && $href.is(':visible')
1931              && [[$href[offsetMethod]().top + offsetBase, href]]) || null
1932          })
1933          .sort(function (a, b) { return a[0] - b[0] })
1934          .each(function () {
1935            that.offsets.push(this[0])
1936            that.targets.push(this[1])
1937          })
1938      }
1939
1940      ScrollSpy.prototype.process = function () {
1941        var scrollTop    = this.$scrollElement.scrollTop() + this.options.offset
1942        var scrollHeight = this.getScrollHeight()
1943        var maxScroll    = this.options.offset + scrollHeight -
  this.$scrollElement.height()
1944        var offsets      = this.offsets
1945        var targets      = this.targets
1946        var activeTarget = this.activeTarget
1947        var i
1948
1949        if (this.scrollHeight != scrollHeight) {
1950          this.refresh()
1951        }
1952
1953        if (scrollTop >= maxScroll) {
1954          return activeTarget != (i = targets[targets.length - 1]) && this.activate(i)
1955        }
1956
1957        if (activeTarget && scrollTop < offsets[0]) {
1958          this.activeTarget = null
1959          return this.clear()
1960        }
1961
1962        for (i = offsets.length; i--;) {
1963          activeTarget != targets[i]
1964            && scrollTop >= offsets[i]
1965            && (offsets[i + 1] === undefined || scrollTop < offsets[i + 1])
1966            && this.activate(targets[i])
1967        }
1968      }
1969
1970      ScrollSpy.prototype.activate = function (target) {
1971        this.activeTarget = target
1972
1973        this.clear()
1974
1975        var selector = this.selector +
1976          '[data-target="' + target + '"],' +
```

```
1977          this.selector + '[href="' + target + '"]'
1978
1979        var active = $(selector)
1980          .parents('li')
1981          .addClass('active')
1982
1983        if (active.parent('.dropdown-menu').length) {
1984          active = active
1985            .closest('li.dropdown')
1986            .addClass('active')
1987        }
1988
1989        active.trigger('activate.bs.scrollspy')
1990      }
1991
1992      ScrollSpy.prototype.clear = function () {
1993        $(this.selector)
1994          .parentsUntil(this.options.target, '.active')
1995          .removeClass('active')
1996      }
1997
1998
1999      // SCROLLSPY PLUGIN DEFINITION
2000      // ===========================
2001
2002      function Plugin(option) {
2003        return this.each(function () {
2004          var $this   = $(this)
2005          var data    = $this.data('bs.scrollspy')
2006          var options = typeof option == 'object' && option
2007
2008          if (!data) $this.data('bs.scrollspy', (data = new ScrollSpy(this, options)))
2009          if (typeof option == 'string') data[option]()
2010        })
2011      }
2012
2013      var old = $.fn.scrollspy
2014
2015      $.fn.scrollspy             = Plugin
2016      $.fn.scrollspy.Constructor = ScrollSpy
2017
2018
2019      // SCROLLSPY NO CONFLICT
2020      // =====================
2021
2022      $.fn.scrollspy.noConflict = function () {
2023        $.fn.scrollspy = old
2024        return this
2025      }
2026
2027
2028      // SCROLLSPY DATA-API
2029      // ==================
2030
2031      $(window).on('load.bs.scrollspy.data-api', function () {
2032        $('[data-spy="scroll"]').each(function () {
2033          var $spy = $(this)
2034          Plugin.call($spy, $spy.data())
2035        })
2036      })
```

```
}(jQuery);

/* ========================================================================
 * Bootstrap: tab.js v3.3.6
 * http://getbootstrap.com/javascript/#tabs
 * ========================================================================
 * Copyright 2011-2015 Twitter, Inc.
 * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
 * ======================================================================== */


+function ($) {
  'use strict';

  // TAB CLASS DEFINITION
  // ====================

  var Tab = function (element) {
    // jscs:disable requireDollarBeforejQueryAssignment
    this.element = $(element)
    // jscs:enable requireDollarBeforejQueryAssignment
  }

  Tab.VERSION = '3.3.6'

  Tab.TRANSITION_DURATION = 150

  Tab.prototype.show = function () {
    var $this    = this.element
    var $ul      = $this.closest('ul:not(.dropdown-menu)')
    var selector = $this.data('target')

    if (!selector) {
      selector = $this.attr('href')
      selector = selector && selector.replace(/.*(?=#[^\s]*$)/, '') // strip for ie7
    }

    if ($this.parent('li').hasClass('active')) return

    var $previous = $ul.find('.active:last a')
    var hideEvent = $.Event('hide.bs.tab', {
      relatedTarget: $this[0]
    })
    var showEvent = $.Event('show.bs.tab', {
      relatedTarget: $previous[0]
    })

    $previous.trigger(hideEvent)
    $this.trigger(showEvent)

    if (showEvent.isDefaultPrevented() || hideEvent.isDefaultPrevented()) return

    var $target = $(selector)

    this.activate($this.closest('li'), $ul)
    this.activate($target, $target.parent(), function () {
      $previous.trigger({
        type: 'hidden.bs.tab',
        relatedTarget: $this[0]
```

```
2097          })
2098          $this.trigger({
2099            type: 'shown.bs.tab',
2100            relatedTarget: $previous[0]
2101          })
2102        })
2103      }
2104
2105      Tab.prototype.activate = function (element, container, callback) {
2106        var $active    = container.find('> .active')
2107        var transition = callback
2108          && $.support.transition
2109          && ($active.length && $active.hasClass('fade') || !!container.find('>
    .fade').length)
2110
2111        function next() {
2112          $active
2113            .removeClass('active')
2114            .find('> .dropdown-menu > .active')
2115              .removeClass('active')
2116            .end()
2117            .find('[data-toggle="tab"]')
2118              .attr('aria-expanded', false)
2119
2120          element
2121            .addClass('active')
2122            .find('[data-toggle="tab"]')
2123              .attr('aria-expanded', true)
2124
2125          if (transition) {
2126            element[0].offsetWidth // reflow for transition
2127            element.addClass('in')
2128          } else {
2129            element.removeClass('fade')
2130          }
2131
2132          if (element.parent('.dropdown-menu').length) {
2133            element
2134              .closest('li.dropdown')
2135                .addClass('active')
2136              .end()
2137              .find('[data-toggle="tab"]')
2138                .attr('aria-expanded', true)
2139          }
2140
2141          callback && callback()
2142        }
2143
2144        $active.length && transition ?
2145          $active
2146            .one('bsTransitionEnd', next)
2147            .emulateTransitionEnd(Tab.TRANSITION_DURATION) :
2148          next()
2149
2150        $active.removeClass('in')
2151      }
2152
2153
2154      // TAB PLUGIN DEFINITION
2155      // =====================
```

```
2156
2157      function Plugin(option) {
2158        return this.each(function () {
2159          var $this = $(this)
2160          var data  = $this.data('bs.tab')
2161
2162          if (!data) $this.data('bs.tab', (data = new Tab(this)))
2163          if (typeof option == 'string') data[option]()
2164        })
2165      }
2166
2167      var old = $.fn.tab
2168
2169      $.fn.tab             = Plugin
2170      $.fn.tab.Constructor = Tab
2171
2172
2173      // TAB NO CONFLICT
2174      // ===============
2175
2176      $.fn.tab.noConflict = function () {
2177        $.fn.tab = old
2178        return this
2179      }
2180
2181
2182      // TAB DATA-API
2183      // ============
2184
2185      var clickHandler = function (e) {
2186        e.preventDefault()
2187        Plugin.call($(this), 'show')
2188      }
2189
2190      $(document)
2191        .on('click.bs.tab.data-api', '[data-toggle="tab"]', clickHandler)
2192        .on('click.bs.tab.data-api', '[data-toggle="pill"]', clickHandler)
2193
2194    }(jQuery);
2195
2196    +function ($) {
2197      'use strict';
2198
2199      // AFFIX CLASS DEFINITION
2200      // ======================
2201
2202      var Affix = function (element, options) {
2203        this.options = $.extend({}, Affix.DEFAULTS, options)
2204
2205        this.$target = $(this.options.target)
2206          .on('scroll.bs.affix.data-api', $.proxy(this.checkPosition, this))
2207          .on('click.bs.affix.data-api',  $.proxy(this.checkPositionWithEventLoop, this))
2208
2209        this.$element     = $(element)
2210        this.affixed      = null
2211        this.unpin        = null
2212        this.pinnedOffset = null
2213
2214        this.checkPosition()
2215      }
```

```
2216
2217      Affix.VERSION  = '3.3.6'
2218
2219      Affix.RESET    = 'affix affix-top affix-bottom'
2220
2221      Affix.DEFAULTS = {
2222        offset: 0,
2223        target: window
2224      }
2225
2226      Affix.prototype.getState = function (scrollHeight, height, offsetTop, offsetBottom) {
2227        var scrollTop    = this.$target.scrollTop()
2228        var position     = this.$element.offset()
2229        var targetHeight = this.$target.height()
2230
2231        if (offsetTop != null && this.affixed == 'top') return scrollTop < offsetTop ?
    'top' : false
2232
2233        if (this.affixed == 'bottom') {
2234          if (offsetTop != null) return (scrollTop + this.unpin <= position.top) ? false :
    'bottom'
2235          return (scrollTop + targetHeight <= scrollHeight - offsetBottom) ? false :
    'bottom'
2236        }
2237
2238        var initializing   = this.affixed == null
2239        var colliderTop    = initializing ? scrollTop : position.top
2240        var colliderHeight = initializing ? targetHeight : height
2241
2242        if (offsetTop != null && scrollTop <= offsetTop) return 'top'
2243        if (offsetBottom != null && (colliderTop + colliderHeight >= scrollHeight -
    offsetBottom)) return 'bottom'
2244
2245        return false
2246      }
2247
2248      Affix.prototype.getPinnedOffset = function () {
2249        if (this.pinnedOffset) return this.pinnedOffset
2250        this.$element.removeClass(Affix.RESET).addClass('affix')
2251        var scrollTop = this.$target.scrollTop()
2252        var position  = this.$element.offset()
2253        return (this.pinnedOffset = position.top - scrollTop)
2254      }
2255
2256      Affix.prototype.checkPositionWithEventLoop = function () {
2257        setTimeout($.proxy(this.checkPosition, this), 1)
2258      }
2259
2260      Affix.prototype.checkPosition = function () {
2261        if (!this.$element.is(':visible')) return
2262
2263        var height       = this.$element.height()
2264        var offset       = this.options.offset
2265        var offsetTop    = offset.top
2266        var offsetBottom = offset.bottom
2267        var scrollHeight = Math.max($(document).height(), $(document.body).height())
2268
2269        if (typeof offset != 'object')         offsetBottom = offsetTop = offset
2270        if (typeof offsetTop == 'function')    offsetTop    = offset.top(this.$element)
2271        if (typeof offsetBottom == 'function') offsetBottom = offset.bottom(this.$element)
2272
```

```
2273        var affix = this.getState(scrollHeight, height, offsetTop, offsetBottom)
2274
2275        if (this.affixed != affix) {
2276          if (this.unpin != null) this.$element.css('top', '')
2277
2278          var affixType = 'affix' + (affix ? '-' + affix : '')
2279          var e         = $.Event(affixType + '.bs.affix')
2280
2281          this.$element.trigger(e)
2282
2283          if (e.isDefaultPrevented()) return
2284
2285          this.affixed = affix
2286          this.unpin = affix == 'bottom' ? this.getPinnedOffset() : null
2287
2288          this.$element
2289            .removeClass(Affix.RESET)
2290            .addClass(affixType)
2291            .trigger(affixType.replace('affix', 'affixed') + '.bs.affix')
2292        }
2293
2294        if (affix == 'bottom') {
2295          this.$element.offset({
2296            top: scrollHeight - height - offsetBottom
2297          })
2298        }
2299      }
2300
2301
2302      // AFFIX PLUGIN DEFINITION
2303      // =======================
2304
2305      function Plugin(option) {
2306        return this.each(function () {
2307          var $this   = $(this)
2308          var data    = $this.data('bs.affix')
2309          var options = typeof option == 'object' && option
2310
2311          if (!data) $this.data('bs.affix', (data = new Affix(this, options)))
2312          if (typeof option == 'string') data[option]()
2313        })
2314      }
2315
2316      var old = $.fn.affix
2317
2318      $.fn.affix             = Plugin
2319      $.fn.affix.Constructor = Affix
2320
2321
2322      // AFFIX NO CONFLICT
2323      // =================
2324
2325      $.fn.affix.noConflict = function () {
2326        $.fn.affix = old
2327        return this
2328      }
2329
2330
2331      // AFFIX DATA-API
2332      // ==============
```

```
2333
2334      $(window).on('load', function () {
2335        $('[data-spy="affix"]').each(function () {
2336          var $spy = $(this)
2337          var data = $spy.data()
2338
2339          data.offset = data.offset || {}
2340
2341          if (data.offsetBottom != null) data.offset.bottom = data.offsetBottom
2342          if (data.offsetTop    != null) data.offset.top    = data.offsetTop
2343
2344          Plugin.call($spy, data)
2345        })
2346      })
2347
2348    }(jQuery);
2349
```