

```

/*!
 * SmartMenus jQuery Plugin - v1.0.1 - November 1, 2016
 * http://www.smartmenus.org/
 *
 * Copyright Vasil Dinkov, Vadikom Web Ltd.
 * http://vadikom.com
 *
 * Licensed MIT
 */

(function(factory) {
    if (typeof define === 'function' && define.amd) {
        // AMD
        define(['jquery'], factory);
    } else if (typeof module === 'object' && typeof module.exports === 'object') {
        // CommonJS
        module.exports = factory(require('jquery'));
    } else {
        // Global jQuery
        factory(jQuery);
    }
})(function($) {

    var menuTrees = [],
        IE = !window.createPopup, // detect it for the iframe shim
        mouse = false, // optimize for touch by default - we will detect for mouse input
        touchEvents = 'ontouchstart' in window, // we use this just to choose between touch and
pointer events, not for touch screen detection
        mouseDetectionEnabled = false,
        requestAnimationFrame = window.requestAnimationFrame || function(callback) { return
setTimeout(callback, 1000 / 60); },
        cancelAnimationFrame = window.cancelAnimationFrame || function(id) { clearTimeout(id);
};

    // Handle detection for mouse input (i.e. desktop browsers, tablets with a mouse, etc.)
    function initMouseDetection(disable) {
        var eNS = 'smartmenus_mouse';
        if (!mouseDetectionEnabled && !disable) {
            // if we get two consecutive mousmoves within 2 pixels from each other and
within 300ms, we assume a real mouse/cursor is present
            // in practice, this seems like impossible to trick unintentionally with a real mouse
and a pretty safe detection on touch devices (even with older browsers that do not support touch events)
            var firstTime = true,
                lastMove = null;
            $(document).bind(getEventsNS([
                ['mousemove', function(e) {
                    var thisMove = { x: e.pageX, y: e.pageY, timeStamp: new
Date().getTime() };

                    if (lastMove) {
                        var deltaX = Math.abs(lastMove.x - thisMove.x),
                            deltaY = Math.abs(lastMove.y - thisMove.y);
                        if ((deltaX > 0 || deltaY > 0) && deltaX <= 2 && deltaY <=
2 && thisMove.timeStamp - lastMove.timeStamp <= 300) {
                            mouse = true;
                            // if this is the first check after page load, check if
we are not over some item by chance and call the mouseenter handler if yes

```



order)

```
this.showTimeout = 0;
this.hideTimeout = 0;
this.scrollTimeout = 0;
this.clickActivated = false;
this.focusActivated = false;
this.zIndexInc = 0;
this.idInc = 0;
this.$firstLink = null; // we'll use these for some tests
this.$firstSub = null; // at runtime so we'll cache them
this.disabled = false;
this.$disableOverlay = null;
this.$touchScrollingSub = null;
this.cssTransforms3d = 'perspective' in elm.style || 'webkitPerspective' in elm.style;
this.wasCollapsible = false;
this.init();
```

};

```
$.extend($.SmartMenus, {
  hideAll: function() {
    $.each(menuTrees, function() {
      this.menuHideAll();
    });
  },
  destroy: function() {
    while (menuTrees.length) {
      menuTrees[0].destroy();
    }
    initMouseDetection(true);
  },
  prototype: {
```

```
    init: function(refresh) {
      var self = this;
```

```
      if (!refresh) {
        menuTrees.push(this);
```

```
        this.rootId = (new Date().getTime() + Math.random()) +
```

"".replace(/D/g, "");

```
        this.accessIdPrefix = 'sm-' + this.rootId + '-';
```

```
        if (this.$root.hasClass('sm-rtl')) {
          this.opts.rightToLeftSubMenus = true;
        }
      }
    },
```

```
    // init root (main menu)
    var eNS = '.smartmenus';
    this.$root
```

```
      .data('smartmenus', this)
      .attr('data-smartmenus-id', this.rootId)
      .dataSM('level', 1)
      .bind(getEventsNS([
        ['mouseover focusin', $.proxy(this.rootOver,
```

this]],

```
        ['mouseout focusout', $.proxy(this.rootOut, this)],
        ['keydown', $.proxy(this.rootKeyDown, this)]
```

```

    ], eNS))
    .delegate('a', getEventsNS([
        ['mouseenter', $.proxy(this.itemEnter, this)],
        ['mouseleave', $.proxy(this.itemLeave, this)],
        ['mousedown', $.proxy(this.itemDown, this)],
        ['focus', $.proxy(this.itemFocus, this)],
        ['blur', $.proxy(this.itemBlur, this)],
        ['click', $.proxy(this.itemClick, this)]
    ], eNS));

    // hide menus on tap or click outside the root UL
    eNS += this.rootId;
    if (this.opts.hideOnClick) {
        $(document).bind(getEventsNS([
            ['touchstart', $.proxy(this.docTouchStart, this)],
            ['touchmove', $.proxy(this.docTouchMove, this)],
            ['touchend', $.proxy(this.docTouchEnd, this)],
            // for Opera Mobile < 11.5, webOS browser, etc.
            ['click', $.proxy(this.docClick, this)]
        ], eNS));
    }
    // hide sub menus on resize
    $.proxy(this.winResize, this)].bind(getEventsNS(['resize orientationchange',
    $.proxy(this.winResize, this)]), eNS));

    if (this.opts.subIndicators) {
        this.$subArrow = $('<span/>').addClass('sub-arrow');
        if (this.opts.subIndicatorsText) {
            this.$subArrow.html(this.opts.subIndicatorsText);
        }
    }

    // make sure mouse detection is enabled
    initMouseDetection();
}

// init sub menus
this.$firstSub = this.$root.find('ul').each(function() { self.menuInit($(this));
}).eq(0);

this.$firstLink = this.$root.find('a').eq(0);

// find current item
if (this.opts.markCurrentItem) {
    var reDefaultDoc = /(index|default)\.[^#\?V]*/i,
        reHash = /#.*/,
        locHref = window.location.href.replace(reDefaultDoc, ""),
        locHrefNoHash = locHref.replace(reHash, "");
    this.$root.find('a').each(function() {
        var href = this.href.replace(reDefaultDoc, ""),
            $this = $(this);
        if (href == locHref || href == locHrefNoHash) {
            $this.addClass('current');
            if (self.opts.markCurrentTree) {

```

```

id]', 'ul').each(function() {
    $this.parentsUntil('[data-smartmenu-s-
    $(this).dataSM('parent-
a').addClass('current');
    });
}
});
}

// save initial state
this.wasCollapsible = this.isCollapsible();
},
destroy: function(refresh) {
    if (!refresh) {
        var eNS = '.smartmenus';
        this.$root
            .removeData('smartmenus')
            .removeAttr('data-smartmenu-id')
            .removeDataSM('level')
            .unbind(eNS)
            .undelegate(eNS);
        eNS += this.rootId;
        $(document).unbind(eNS);
        $(window).unbind(eNS);
        if (this.opts.subIndicators) {
            this.$subArrow = null;
        }
    }
    this.menuHideAll();
    var self = this;
    this.$root.find('ul').each(function() {
        var $this = $(this);
        if ($this.dataSM('scroll-arrows')) {
            $this.dataSM('scroll-arrows').remove();
        }
        if ($this.dataSM('shown-before')) {
            if (self.opts.subMenusMinWidth ||
self.opts.subMenusMaxWidth) {
                $this.css({ width: "", minWidth: "",
maxWidth: "" }).removeClass('sm-nowrap');
            }
            if ($this.dataSM('scroll-arrows')) {
                $this.dataSM('scroll-arrows').remove();
            }
            $this.css({ zIndex: "", top: "", left: "", marginLeft: "",
marginTop: "", display: "" });
        }
        if (($this.attr('id') || "").indexOf(self.accessIdPrefix) == 0) {
            $this.removeAttr('id');
        }
    })
    .removeDataSM('in-mega')
    .removeDataSM('shown-before')
    .removeDataSM('ie-shim')
    .removeDataSM('scroll-arrows')

```

```

        .removeDataSM('parent-a')
        .removeDataSM('level')
        .removeDataSM('beforefirstshowfired')
        .removeAttr('role')
        .removeAttr('aria-hidden')
        .removeAttr('aria-labelledby')
        .removeAttr('aria-expanded');
    this.$root.find('a.has-submenu').each(function() {
        var $this = $(this);
        if ($this.attr('id').indexOf(self.accessIdPrefix) == 0) {
            $this.removeAttr('id');
        }
    })
    .removeClass('has-submenu')
    .removeDataSM('sub')
    .removeAttr('aria-haspopup')
    .removeAttr('aria-controls')
    .removeAttr('aria-expanded')
    .closest('li').removeDataSM('sub');
    if (this.opts.subIndicators) {
        this.$root.find('span.sub-arrow').remove();
    }
    if (this.opts.markCurrentItem) {
        this.$root.find('a.current').removeClass('current');
    }
    if (!refresh) {
        this.$root = null;
        this.$firstLink = null;
        this.$firstSub = null;
        if (this.$disableOverlay) {
            this.$disableOverlay.remove();
            this.$disableOverlay = null;
        }
        menuTrees.splice($.inArray(this, menuTrees), 1);
    }
},
disable: function(noOverlay) {
    if (!this.disabled) {
        this.menuHideAll();
        // display overlay over the menu to prevent interaction
        if (!noOverlay && !this.opts.isPopup && this.$root.is(':visible')) {
            var pos = this.$root.offset();
            this.$disableOverlay = $('<div class="sm-jquery-disable-
overlay"/>').css({
                position: 'absolute',
                top: pos.top,
                left: pos.left,
                width: this.$root.outerWidth(),
                height: this.$root.outerHeight(),
                zIndex: this.getStartZIndex(true),
                opacity: 0
            }).appendTo(document.body);
        }
        this.disabled = true;
    }
},

```

```

docClick: function(e) {
    if (this.$touchScrollingSub) {
        this.$touchScrollingSub = null;
        return;
    }
    // hide on any click outside the menu or on a menu link
    if (this.visibleSubMenus.length && !$_.contains(this.$root[0], e.target) ||
$(e.target).is('a')) {
        this.menuHideAll();
    }
},
docTouchEnd: function(e) {
    if (!this.lastTouch) {
        return;
    }
    if (this.visibleSubMenus.length && (this.lastTouch.x2 === undefined ||
this.lastTouch.x1 == this.lastTouch.x2) && (this.lastTouch.y2 === undefined || this.lastTouch.y1 ==
this.lastTouch.y2) && (!this.lastTouch.target || !$_.contains(this.$root[0], this.lastTouch.target))) {
        if (this.hideTimeout) {
            clearTimeout(this.hideTimeout);
            this.hideTimeout = 0;
        }
        // hide with a delay to prevent triggering accidental unwanted
click on some page element
        var self = this;
        this.hideTimeout = setTimeout(function() { self.menuHideAll(); },
350);
    }
    this.lastTouch = null;
},
docTouchMove: function(e) {
    if (!this.lastTouch) {
        return;
    }
    var touchPoint = e.originalEvent.touches[0];
    this.lastTouch.x2 = touchPoint.pageX;
    this.lastTouch.y2 = touchPoint.pageY;
},
docTouchStart: function(e) {
    var touchPoint = e.originalEvent.touches[0];
    this.lastTouch = { x1: touchPoint.pageX, y1: touchPoint.pageY, target:
touchPoint.target };
},
enable: function() {
    if (this.disabled) {
        if (this.$disableOverlay) {
            this.$disableOverlay.remove();
            this.$disableOverlay = null;
        }
        this.disabled = false;
    }
},
getClosestMenu: function(elm) {
    var $closestMenu = $(elm).closest('ul');
    while ($closestMenu.dataSM('in-mega')) {
        $closestMenu = $closestMenu.parent().closest('ul');
    }
}

```

```

    }
    return $closestMenu[0] || null;
  },
  getHeight: function($elm) {
    return this.getOffset($elm, true);
  },
  // returns precise width/height float values
  getOffset: function($elm, height) {
    var old;
    if ($elm.css('display') == 'none') {
      old = { position: $elm[0].style.position, visibility:
$elm[0].style.visibility };
      $elm.css({ position: 'absolute', visibility: 'hidden' }).show();
    }
    var box = $elm[0].getBoundingClientRect() &&
$elm[0].getBoundingClientRect(),
    val = box && (height ? box.height || box.bottom - box.top :
box.width || box.right - box.left);
    if (!val && val !== 0) {
      val = height ? $elm[0].offsetHeight : $elm[0].offsetWidth;
    }
    if (old) {
      $elm.hide().css(old);
    }
    return val;
  },
  getStartZIndex: function(root) {
    var zIndex = parseInt(this[root ? '$root' : '$firstSub'].css('z-index'));
    if (!root && isNaN(zIndex)) {
      zIndex = parseInt(this.$root.css('z-index'));
    }
    return !isNaN(zIndex) ? zIndex : 1;
  },
  getTouchPoint: function(e) {
    return e.touches && e.touches[0] || e.changedTouches &&
e.changedTouches[0] || e;
  },
  getViewport: function(height) {
    var name = height ? 'Height' : 'Width',
        val = document.documentElement['client' + name],
        val2 = window['inner' + name];
    if (val2) {
      val = Math.min(val, val2);
    }
    return val;
  },
  getViewportHeight: function() {
    return this.getViewport(true);
  },
  getViewportWidth: function() {
    return this.getViewport();
  },
  getWidth: function($elm) {
    return this.getOffset($elm);
  },
  handleEvents: function() {

```

```

        return !this.disabled && this.isCSSOn();
    },
    handleItemEvents: function($a) {
        return this.handleEvents() && !this.isLinkInMegaMenu($a);
    },
    isCollapsible: function() {
        return this.$firstSub.css('position') == 'static';
    },
    isCSSOn: function() {
        return this.$firstLink.css('display') == 'block';
    },
    isFixed: function() {
        var isFixed = this.$root.css('position') == 'fixed';
        if (!isFixed) {
            this.$root.parentsUntil('body').each(function() {
                if ($(this).css('position') == 'fixed') {
                    isFixed = true;
                    return false;
                }
            });
        }
        return isFixed;
    },
    isLinkInMegaMenu: function($a) {
        return $(this.getClosestMenu($a[0])).hasClass('mega-menu');
    },
    isTouchMode: function() {
        return !mouse || this.opts.noMouseOver || this.isCollapsible();
    },
    itemActivate: function($a, focus) {
        var $ul = $a.closest('ul'),
            level = $ul.dataSM('level');
        // if for some reason the parent item is not activated (e.g. this is an API
        call to activate the item), activate all parent items first
        if (level > 1 && (!this.activatedItems[level - 2] || this.activatedItems[level -
        2][0] != $ul.dataSM('parent-a')[0])) {
            var self = this;
            $($ul.parentsUntil('[data-smartmenus-id]',
            'ul').get().reverse()).add($ul).each(function() {
                self.itemActivate($(this).dataSM('parent-a'));
            });
        }
        // hide any visible deeper level sub menus
        if (!this.isCollapsible() || focus) {
            this.menuHideSubMenus(!this.activatedItems[level - 1] ||
            this.activatedItems[level - 1][0] != $a[0] ? level - 1 : level);
        }
        // save new active item for this level
        this.activatedItems[level - 1] = $a;
        if (this.$root.triggerHandler('activate.smart', $a[0]) === false) {
            return;
        }
        // show the sub menu if this item has one
        var $sub = $a.dataSM('sub');
        if ($sub && (this.isTouchMode() || (!this.opts.showOnClick ||
        this.clickActivated))) {

```

```

        this.menuShow($sub);
    }
},
itemBlur: function(e) {
    var $a = $(e.currentTarget);
    if (!this.handleItemEvents($a)) {
        return;
    }
    this.$root.triggerHandler('blur.smap', $a[0]);
},
itemClick: function(e) {
    var $a = $(e.currentTarget);
    if (!this.handleItemEvents($a)) {
        return;
    }
    if (this.$touchScrollingSub && this.$touchScrollingSub[0] ==
$a.closest('ul')[0]) {
        this.$touchScrollingSub = null;
        e.stopPropagation();
        return false;
    }
    if (this.$root.triggerHandler('click.smap', $a[0]) === false) {
        return false;
    }
    var subArrowClicked = $(e.target).is('span.sub-arrow'),
        $sub = $a.dataSM('sub'),
        firstLevelSub = $sub ? $sub.dataSM('level') == 2 : false;
    // if the sub is not visible
    if ($sub && !$sub.is(':visible')) {
        if (this.opts.showOnClick && firstLevelSub) {
            this.clickActivated = true;
        }
        // try to activate the item and show the sub
        this.itemActivate($a);
        // if "itemActivate" showed the sub, prevent the click so that the
link is not loaded
        // if it couldn't show it, then the sub menus are disabled with an
important declaration (e.g. via mobile styles) so let the link get loaded
        if ($sub.is(':visible')) {
            this.focusActivated = true;
            return false;
        }
    } else if (this.isCollapsible() && subArrowClicked) {
        this.itemActivate($a);
        this.menuHide($sub);
        return false;
    }
    if (this.opts.showOnClick && firstLevelSub || $a.hasClass('disabled') ||
this.$root.triggerHandler('select.smap', $a[0]) === false) {
        return false;
    }
},
itemDown: function(e) {
    var $a = $(e.currentTarget);
    if (!this.handleItemEvents($a)) {
        return;
    }

```

```

    }
    $a.dataSM('mousedown', true);
  },
  itemEnter: function(e) {
    var $a = $(e.currentTarget);
    if (!this.handleItemEvents($a)) {
      return;
    }
    if (!this.isTouchMode()) {
      if (this.showTimeout) {
        clearTimeout(this.showTimeout);
        this.showTimeout = 0;
      }
      var self = this;
      this.showTimeout = setTimeout(function() { self.itemActivate($a);
    }, this.opts.showOnClick && $a.closest('ul').dataSM('level') == 1 ? 1 : this.opts.showTimeout);
  },
  this.$root.triggerHandler('mouseenter.smap', $a[0]);
},
itemFocus: function(e) {
  var $a = $(e.currentTarget);
  if (!this.handleItemEvents($a)) {
    return;
  }
  // fix (the mousedown check): in some browsers a tap/click produces
consecutive focus + click events so we don't need to activate the item on focus
  if (this.focusActivated && (!this.isTouchMode() ||
!$a.dataSM('mousedown')) && (!this.activatedItems.length || this.activatedItems[this.activatedItems.length -
1][0] != $a[0])) {
    this.itemActivate($a, true);
  }
  this.$root.triggerHandler('focus.smap', $a[0]);
},
itemLeave: function(e) {
  var $a = $(e.currentTarget);
  if (!this.handleItemEvents($a)) {
    return;
  }
  if (!this.isTouchMode()) {
    $a[0].blur();
    if (this.showTimeout) {
      clearTimeout(this.showTimeout);
      this.showTimeout = 0;
    }
  }
  $a.removeDataSM('mousedown');
  this.$root.triggerHandler('mouseleave.smap', $a[0]);
},
menuHide: function($sub) {
  if (this.$root.triggerHandler('beforehide.smap', $sub[0]) === false) {
    return;
  }
  $sub.stop(true, true);
  if ($sub.css('display') != 'none') {
    var complete = function() {
      // unset z-index

```

```

        $sub.css('z-index', "");
    };
    // if sub is collapsible (mobile view)
    if (this.isCollapsible()) {
        if (this.opts.collapsibleHideFunction) {
            this.opts.collapsibleHideFunction.call(this, $sub,
complete);

        } else {
            $sub.hide(this.opts.collapsibleHideDuration,
complete);

        }
    } else {
        if (this.opts.hideFunction) {
            this.opts.hideFunction.call(this, $sub, complete);
        } else {
            $sub.hide(this.opts.hideDuration, complete);
        }
    }
    // remove IE iframe shim
    if ($sub.dataSM('ie-shim')) {
        $sub.dataSM('ie-shim').remove().css({'-webkit-
transform': "", transform: "" });
    }
    // deactivate scrolling if it is activated for this sub
    if ($sub.dataSM('scroll')) {
        this.menuScrollStop($sub);
        $sub.css({'touch-action': "", '-ms-touch-action': "", '-webkit-
transform': "", transform: "" })
    }
    .unbind('.smartmenus_scroll').removeDataSM('scroll').dataSM('scroll-arrows').hide();
    }
    // unhighlight parent item + accessibility
    $sub.dataSM('parent-a').removeClass('highlighted').attr('aria-
expanded', 'false');

    $sub.attr({
        'aria-expanded': 'false',
        'aria-hidden': 'true'
    });
    var level = $sub.dataSM('level');
    this.activatedItems.splice(level - 1, 1);
    this.visibleSubMenus.splice($.inArray($sub,
this.visibleSubMenus), 1);

    this.$root.triggerHandler('hide.smapr', $sub[0]);
    }
},
menuHideAll: function() {
    if (this.showTimeout) {
        clearTimeout(this.showTimeout);
        this.showTimeout = 0;
    }
    // hide all subs
    // if it's a popup, this.visibleSubMenus[0] is the root UL
    var level = this.opts.isPopup ? 1 : 0;
    for (var i = this.visibleSubMenus.length - 1; i >= level; i--) {
        this.menuHide(this.visibleSubMenus[i]);
    }
}

```

```

// hide root if it's popup
if (this.opts.isPopup) {
    this.$root.stop(true, true);
    if (this.$root.is(':visible')) {
        if (this.opts.hideFunction) {
            this.opts.hideFunction.call(this, this.$root);
        } else {
            this.$root.hide(this.opts.hideDuration);
        }
        // remove IE iframe shim
        if (this.$root.dataSM('ie-shim')) {
            this.$root.dataSM('ie-shim').remove();
        }
    }
}
this.activatedItems = [];
this.visibleSubMenus = [];
this.clickActivated = false;
this.focusActivated = false;
// reset z-index increment
this.zIndexInc = 0;
this.$root.triggerHandler('hideAll.smap');
},
menuHideSubMenus: function(level) {
    for (var i = this.activatedItems.length - 1; i >= level; i--) {
        var $sub = this.activatedItems[i].dataSM('sub');
        if ($sub) {
            this.menuHide($sub);
        }
    }
},
menuIframeShim: function($ul) {
    // create iframe shim for the menu
    if (IE && this.opts.overlapControlsInIE && !$ul.dataSM('ie-shim')) {
        $ul.dataSM('ie-shim', $('<iframe/>').attr({ src: 'javascript:0',
            .css({ position: 'absolute', top: 'auto', left: '0', opacity: 0,
        }
    );
}
},
menuInit: function($ul) {
    if (!$ul.dataSM('in-mega')) {
        // mark UL's in mega drop downs (if any) so we can neglect them
        if ($ul.hasClass('mega-menu')) {
            $ul.find('ul').dataSM('in-mega', true);
        }
        // get level (much faster than, for example, using parentsUntil)
        var level = 2,
            par = $ul[0];
        while ((par = par.parentNode.parentNode) != this.$root[0]) {
            level++;
        }
        // cache stuff for quick access
        var $a = $ul.prevAll('a').eq(-1);
        // if the link is nested (e.g. in a heading)

```

```

        if (!$a.length) {
            $a = $ul.prevAll().find('a').eq(-1);
        }
        $a.addClass('has-submenu').dataSM('sub', $ul);
        $ul.dataSM('parent-a', $a)
            .dataSM('level', level)
            .parent().dataSM('sub', $ul);
        // accessibility
        var ald = $a.attr('id') || this.accessIdPrefix + (++this.idInc),
            ulld = $ul.attr('id') || this.accessIdPrefix + (++this.idInc);
        $a.attr({
            id: ald,
            'aria-haspopup': 'true',
            'aria-controls': ulld,
            'aria-expanded': 'false'
        });
        $ul.attr({
            id: ulld,
            'role': 'group',
            'aria-hidden': 'true',
            'aria-labelledby': ald,
            'aria-expanded': 'false'
        });
        // add sub indicator to parent item
        if (this.opts.subIndicators) {
            $a[this.opts.subIndicatorsPos](this.$subArrow.clone());
        }
    }
},
menuPosition: function($sub) {
    var $a = $sub.dataSM('parent-a'),
        $li = $a.closest('li'),
        $ul = $li.parent(),
        level = $sub.dataSM('level'),
        subW = this.getWidth($sub),
        subH = this.getHeight($sub),
        itemOffset = $a.offset(),
        itemX = itemOffset.left,
        itemY = itemOffset.top,
        itemW = this.getWidth($a),
        itemH = this.getHeight($a),
        $win = $(window),
        winX = $win.scrollLeft(),
        winY = $win.scrollTop(),
        winW = this.getViewPortWidth(),
        winH = this.getViewPortHeight(),
        horizontalParent = $ul.parent().is('[data-sm-horizontal-sub]') ||
        level == 2 && !$ul.hasClass('sm-vertical'),
        rightToLeft = this.opts.rightToLeftSubMenus && !$li.is('[data-sm-reverse]') || !this.opts.rightToLeftSubMenus && $li.is('[data-sm-reverse]'),
        subOffsetX = level == 2 ? this.opts.mainMenuSubOffsetX :
        this.opts.subMenusSubOffsetX,
        subOffsetY = level == 2 ? this.opts.mainMenuSubOffsetY :
        this.opts.subMenusSubOffsetY,
        x, y;
    if (horizontalParent) {

```

```

        x = rightToLeft ? itemW - subW - subOffsetX : subOffsetX;
        y = this.opts.bottomToTopSubMenus ? -subH - subOffsetY :
itemH + subOffsetY;
    } else {
        x = rightToLeft ? subOffsetX - subW : itemW - subOffsetX;
        y = this.opts.bottomToTopSubMenus ? itemH - subOffsetY -
subH : subOffsetY;
    }
    if (this.opts.keepInViewport) {
        var absX = itemX + x,
            absY = itemY + y;
        if (rightToLeft && absX < winX) {
            x = horizontalParent ? winX - absX + x : itemW -
subOffsetX;
        } else if (!rightToLeft && absX + subW > winX + winW) {
            x = horizontalParent ? winX + winW - subW - absX + x :
subOffsetX - subW;
        }
        if (!horizontalParent) {
            if (subH < winH && absY + subH > winY + winH) {
                y += winY + winH - subH - absY;
            } else if (subH >= winH || absY < winY) {
                y += winY - absY;
            }
        }
        // do we need scrolling?
        // 0.49 used for better precision when dealing with float values
        if (horizontalParent && (absY + subH > winY + winH + 0.49 ||
absY < winY) || !horizontalParent && subH > winH + 0.49) {
            var self = this;
            if (!$sub.dataSM('scroll-arrows')) {
                $sub.dataSM('scroll-arrows', $('<span
class="scroll-up"><span class="scroll-up-arrow"></span></span>')[0], $('<span class="scroll-
down"><span class="scroll-down-arrow"></span></span>')[0]);
                .bind({
                    mouseenter: function() {
                        $sub.dataSM('scroll').up
                        self.menuScroll($sub);
                    },
                    mouseleave: function(e) {
                        self.menuScrollStop($sub);
                        self.menuScrollOut($sub, e);
                    }
                },
                'mousewheel DOMMouseScroll':
function(e) { e.preventDefault(); }
            })
            .insertAfter($sub)
        );
    }
    // bind scroll events and save scroll data for this sub
    var eNS = 'smartmenus_scroll';
    $sub.dataSM('scroll', {
        y: this.cssTransforms3d ? 0 : y - itemH,

```

```

        step: 1,
        // cache stuff for faster recalcs later
        itemH: itemH,
        subH: subH,
        arrowDownH:
this.getHeight($sub.dataSM('scroll-arrows').eq(1))
    })
    .bind(getEventsNS([
self.menuScrollOver($sub, e); ]],
self.menuScrollOut($sub, e); ]],
function(e) { self.menuScrollMousewheel($sub, e); }
    ], eNS))
marginLeft: x + (parseInt($sub.css('border-left-width')) || 0), width: subW - (parseInt($sub.css('border-left-
width')) || 0) - (parseInt($sub.css('border-right-width')) || 0), zIndex: $sub.css('z-index') }
    .eq(horizontalParent &&
this.opts.bottomToTopSubMenus ? 0 : 1).show();
    // when a menu tree is fixed positioned we allow scrolling
via touch too
    // since there is no other way to access such long sub
menus if no mouse is present
    if (this.isFixed()) {
        $sub.css({ 'touch-action': 'none', '-ms-touch-
action': 'none' })
        .bind(getEventsNS([
touchmove touchend': 'pointerdown pointermove pointerup MSPointerDown MSPointerMove
MSPointerUp', function(e) {
    self.menuScrollTouch($sub, e);
    }
    ], eNS));
    }
}
}
    $sub.css({ top: 'auto', left: '0', marginLeft: x, marginTop: y - itemH });
// IE iframe shim
this.menuIframeShim($sub);
if ($sub.dataSM('ie-shim')) {
    $sub.dataSM('ie-shim').css({ zIndex: $sub.css('z-index'), width:
subW, height: subH, marginLeft: x, marginTop: y - itemH });
}
},
menuScroll: function($sub, once, step) {
    var data = $sub.dataSM('scroll'),
        $arrows = $sub.dataSM('scroll-arrows'),
        end = data.up ? data.upEnd : data.downEnd,
        diff;
    if (!once && data.momentum) {
        data.momentum *= 0.92;
        diff = data.momentum;
        if (diff < 0.5) {
            this.menuScrollStop($sub);

```

```

        return;
    } else {
        diff = step || (once || !this.opts.scrollAccelerate ?
this.opts.scrollStep : Math.floor(data.step));
    }
    // hide any visible deeper level sub menus
    var level = $sub.dataSM('level');
    if (this.activatedItems[level - 1] && this.activatedItems[level -
1].dataSM('sub') && this.activatedItems[level - 1].dataSM('sub').is(':visible')) {
        this.menuHideSubMenus(level - 1);
    }
    data.y = data.up && end <= data.y || !data.up && end >= data.y ? data.y :
(Math.abs(end - data.y) > diff ? data.y + (data.up ? diff : -diff) : end);
    $sub.add($sub.dataSM('ie-shim')).css(this.cssTransforms3d ? { '-webkit-
transform': 'translate3d(0, ' + data.y + 'px, 0)', transform: 'translate3d(0, ' + data.y + 'px, 0)' } : { marginTop:
data.y });

    // show opposite arrow if appropriate
    if (mouse && (data.up && data.y > data.downEnd || !data.up && data.y <
data.upEnd)) {
        $arrows.eq(data.up ? 1 : 0).show();
    }
    // if we've reached the end
    if (data.y == end) {
        if (mouse) {
            $arrows.eq(data.up ? 0 : 1).hide();
        }
        this.menuScrollStop($sub);
    } else if (!once) {
        if (this.opts.scrollAccelerate && data.step < this.opts.scrollStep) {
            data.step += 0.2;
        }
        var self = this;
        this.scrollTimeout = requestAnimationFrame(function() {
self.menuScroll($sub); });
    }
},
menuScrollMousewheel: function($sub, e) {
    if (this.getClosestMenu(e.target) == $sub[0]) {
        e = e.originalEvent;
        var up = (e.wheelDelta || -e.detail) > 0;
        if ($sub.dataSM('scroll-arrows').eq(up ? 0 : 1).is(':visible')) {
            $sub.dataSM('scroll').up = up;
            this.menuScroll($sub, true);
        }
    }
},
e.preventDefault();
},
menuScrollOut: function($sub, e) {
    if (mouse) {
        if (!/^scroll-(up|down)/.test((e.relatedTarget || "").className) &&
($sub[0] != e.relatedTarget && !$sub.contains($sub[0], e.relatedTarget) ||
this.getClosestMenu(e.relatedTarget) != $sub[0])) {
            $sub.dataSM('scroll-arrows').css('visibility', 'hidden');
        }
    }
}
}

```

```

    },
    menuScrollOver: function($sub, e) {
        if (mouse) {
            if (!/^scroll-(up|down)/.test(e.target.className) &&
this.getClosestMenu(e.target) == $sub[0]) {
                this.menuScrollRefreshData($sub);
                var data = $sub.dataSM('scroll'),
                    upEnd = $(window).scrollTop() -
$sub.dataSM('parent-a').offset().top - data.itemH;
                $sub.dataSM('scroll-arrows').eq(0).css('margin-top',
upEnd).end()
                    .eq(1).css('margin-top', upEnd +
this.getViewportHeight() - data.arrowDownH).end()
                    .css('visibility', 'visible');
            }
        }
    },
    menuScrollRefreshData: function($sub) {
        var data = $sub.dataSM('scroll'),
            upEnd = $(window).scrollTop() - $sub.dataSM('parent-
a').offset().top - data.itemH;
        if (this.cssTransforms3d) {
            upEnd = -(parseFloat($sub.css('margin-top')) - upEnd);
        }
        $.extend(data, {
            upEnd: upEnd,
            downEnd: upEnd + this.getViewportHeight() - data.subH
        });
    },
    menuScrollStop: function($sub) {
        if (this.scrollTimeout) {
            cancelAnimationFrame(this.scrollTimeout);
            this.scrollTimeout = 0;
            $sub.dataSM('scroll').step = 1;
            return true;
        }
    },
    menuScrollTouch: function($sub, e) {
        e = e.originalEvent;
        if (isTouchEvent(e)) {
            var touchPoint = this.getTouchPoint(e);
            // neglect event if we touched a visible deeper level sub menu
            if (this.getClosestMenu(touchPoint.target) == $sub[0]) {
                var data = $sub.dataSM('scroll');
                if (/^(start|down)$/.test(e.type)) {
                    if (this.menuScrollStop($sub)) {
                        // if we were scrolling, just stop and don't
activate any link on the first touch
                    }
                    e.preventDefault();
                    this.$touchScrollingSub = $sub;
                } else {
                    this.$touchScrollingSub = null;
                }
            }
            // update scroll data since the user might have
zoomed, etc.
            this.menuScrollRefreshData($sub);
        }
    }
}

```

```

// extend it with the touch properties
$.extend(data, {
    touchStartY: touchPoint.pageY,
    touchStartTime: e.timeStamp
});
} else if (/move$/i.test(e.type)) {
    var prevY = data.touchY !== undefined ?
        data.touchY : data.touchStartY;
    touchPoint.pageY) {
        up) {
            touchPoint.pageY,
            e.timeStamp
            Math.abs(touchPoint.pageY - prevY));
            Math.pow(Math.abs(touchPoint.pageY - data.touchStartY) / (e.timeStamp - data.touchStartTime), 2) * 15)
            {
                this.menuScrollStop($sub);
                this.menuScroll($sub);
                e.preventDefault();
            }
            delete data.touchY;
        }
    }
}
},
menuShow: function($sub) {
    if (!$sub.dataSM('beforefirstshowfired')) {
        $sub.dataSM('beforefirstshowfired', true);
        if (this.$root.triggerHandler('beforefirstshow.smap', $sub[0]) ===
false) {
            return;
        }
    }
    if (this.$root.triggerHandler('before show.smap', $sub[0]) === false) {
        return;
    }
}
}

```

```

    $sub.dataSM('shown-before', true)
        .stop(true, true);
    if (!$sub.is(':visible')) {
        // highlight parent item
        var $a = $sub.dataSM('parent-a');
        if (this.opts.keepHighlighted || this.isCollapsible()) {
            $a.addClass('highlighted');
        }
        if (this.isCollapsible()) {
            $sub.removeClass('sm-nowrap').css({ zIndex: "", width:
'auto', minWidth: "", maxWidth: "", top: "", left: "", marginLeft: "", marginTop: "" });
        } else {
            // set z-index
            $sub.css('z-index', this.zIndexInc = (this.zIndexInc ||
this.getStartZIndex()) + 1);
            // min/max-width fix - no way to rely purely on CSS as all
            // UL's are nested
            if (this.opts.subMenusMinWidth ||
this.opts.subMenusMaxWidth) {
                $sub.css({ width: 'auto', minWidth: "", maxWidth:
"" }).addClass('sm-nowrap');
                if (this.opts.subMenusMinWidth) {
                    $sub.css('min-width',
this.opts.subMenusMinWidth);
                }
                if (this.opts.subMenusMaxWidth) {
                    var noMaxWidth = this.getWidth($sub);
                    $sub.css('max-width',
this.opts.subMenusMaxWidth);
                    if (noMaxWidth > this.getWidth($sub)) {
                        $sub.removeClass('sm-
nowrap').css('width', this.opts.subMenusMaxWidth);
                    }
                }
            }
            this.menuPosition($sub);
            // insert IE iframe shim
            if ($sub.dataSM('ie-shim')) {
                $sub.dataSM('ie-shim').insertBefore($sub);
            }
        }
        var complete = function() {
            // fix: "overflow: hidden;" is not reset on animation
            // complete in jQuery < 1.9.0 in Chrome when global "box-sizing: border-box;" is used
            $sub.css('overflow', "");
        };
        // if sub is collapsible (mobile view)
        if (this.isCollapsible()) {
            if (this.opts.collapsibleShowFunction) {
                this.opts.collapsibleShowFunction.call(this,
$sub, complete);
            } else {
                $sub.show(this.opts.collapsibleShowDuration,
complete);
            }
        }
    } else {

```

```

        if (this.opts.showFunction) {
            this.opts.showFunction.call(this, $sub,
complete);
        } else {
            $sub.show(this.opts.showDuration, complete);
        }
    }
    // accessibility
    $a.attr('aria-expanded', 'true');
    $sub.attr({
        'aria-expanded': 'true',
        'aria-hidden': 'false'
    });
    // store sub menu in visible array
    this.visibleSubMenus.push($sub);
    this.$root.triggerHandler('show.smapi', $sub[0]);
}
},
popupHide: function(noHideTimeout) {
    if (this.hideTimeout) {
        clearTimeout(this.hideTimeout);
        this.hideTimeout = 0;
    }
    var self = this;
    this.hideTimeout = setTimeout(function() {
        self.menuHideAll();
    }, noHideTimeout ? 1 : this.opts.hideTimeout);
},
popupShow: function(left, top) {
    if (!this.opts.isPopup) {
        alert('SmartMenus jQuery Error:\n\nIf you want to show this
menu via the "popupShow" method, set the isPopup:true option. ');
        return;
    }
    if (this.hideTimeout) {
        clearTimeout(this.hideTimeout);
        this.hideTimeout = 0;
    }
    this.$root.dataSM('shown-before', true)
        .stop(true, true);
    if (!this.$root.is(':visible')) {
        this.$root.css({ left: left, top: top });
        // IE iframe shim
        this.menuIframeShim(this.$root);
        if (this.$root.dataSM('ie-shim')) {
            this.$root.dataSM('ie-shim').css({ zIndex:
this.$root.css('z-index'), width: this.getWidth(this.$root), height: this.getHeight(this.$root), left: left, top: top
}).insertBefore(this.$root);
        }
        // show menu
        var self = this,
            complete = function() {
                self.$root.css('overflow', '');
            };
        if (this.opts.showFunction) {
            this.opts.showFunction.call(this, this.$root, complete);

```

```

        } else {
            this.$root.show(this.opts.showDuration, complete);
        }
        this.visibleSubMenus[0] = this.$root;
    }
},
refresh: function() {
    this.destroy(true);
    this.init(true);
},
rootKeyDown: function(e) {
    if (!this.handleEvents()) {
        return;
    }
    switch (e.keyCode) {
        case 27: // reset on Esc
            var $activeTopItem = this.activatedItems[0];
            if ($activeTopItem) {
                this.menuHideAll();
                $activeTopItem[0].focus();
                var $sub = $activeTopItem.dataSM('sub');
                if ($sub) {
                    this.menuHide($sub);
                }
            }
            break;
        case 32: // activate item's sub on Space
            var $target = $(e.target);
            if ($target.is('a') && this.handleItemEvents($target)) {
                var $sub = $target.dataSM('sub');
                if ($sub && !$sub.is(':visible')) {
                    this.itemClick({ currentTarget: e.target
                    e.preventDefault();
                }
            }
            break;
    }
},
rootOut: function(e) {
    if (!this.handleEvents() || this.isTouchMode() || e.target == this.$root[0]) {
        return;
    }
    if (this.hideTimeout) {
        clearTimeout(this.hideTimeout);
        this.hideTimeout = 0;
    }
    if (!this.opts.showOnClick || !this.opts.hideOnClick) {
        var self = this;
        this.hideTimeout = setTimeout(function() { self.menuHideAll(); },
this.opts.hideTimeout);
    }
},
rootOver: function(e) {
    if (!this.handleEvents() || this.isTouchMode() || e.target == this.$root[0]) {
        return;
    }

```

```

        }
        if (this.hideTimeout) {
            clearTimeout(this.hideTimeout);
            this.hideTimeout = 0;
        }
    },
    winResize: function(e) {
        if (!this.handleEvents()) {
            // we still need to resize the disable overlay if it's visible
            if (this.$disableOverlay) {
                var pos = this.$root.offset();
                this.$disableOverlay.css({
                    top: pos.top,
                    left: pos.left,
                    width: this.$root.outerWidth(),
                    height: this.$root.outerHeight()
                });
            }
            return;
        }
        // hide sub menus on resize - on mobile do it only on orientation change
        if (!('onorientationchange' in window) || e.type == 'orientationchange') {
            var isCollapsible = this.isCollapsible();
            // if it was collapsible before resize and still is, don't do it
            if (!(this.wasCollapsible && isCollapsible)) {
                if (this.activatedItems.length) {
                    this.activatedItems[this.activatedItems.length -
1][0].blur();
                }
                this.menuHideAll();
            }
            this.wasCollapsible = isCollapsible;
        }
    }
});

$.fn.dataSM = function(key, val) {
    if (val) {
        return this.data(key + '_smartmenus', val);
    }
    return this.data(key + '_smartmenus');
};

$.fn.removeDataSM = function(key) {
    return this.removeData(key + '_smartmenus');
};

$.fn.smartmenus = function(options) {
    if (typeof options == 'string') {
        var args = arguments,
            method = options;
        Array.prototype.shift.call(args);
        return this.each(function() {
            var smartmenus = $(this).data('smartmenus');
            if (smartmenus && smartmenus[method]) {

```

```

        smartmenus[method].apply(smartmenus, args);
    }
    });
}
// [data-sm-options] attribute on the root UL
var dataOpts = this.data('sm-options') || null;
if (dataOpts) {
    try {
        dataOpts = eval('(' + dataOpts + ');');
    } catch(e) {
        dataOpts = null;
        alert("ERROR\n\nSmartMenus jQuery init:\nInvalid "data-sm-options"
attribute value syntax.");
    }
}
return this.each(function() {
    new $.SmartMenus(this, $.extend({}, $.fn.smartmenus.defaults, options,
dataOpts));
});
};

// default settings
$.fn.smartmenus.defaults = {
    isPopup: false, // is this a popup menu (can be shown via the
popupShow/popupHide methods) or a permanent menu bar
    mainMenuSubOffsetX: 0, // pixels offset from default position
    mainMenuSubOffsetY: 0, // pixels offset from default position
    subMenusSubOffsetX: 0, // pixels offset from default position
    subMenusSubOffsetY: 0, // pixels offset from default position
    subMenusMinWidth: '10em', // min-width for the sub menus (any CSS
unit) - if set, the fixed width set in CSS will be ignored
    subMenusMaxWidth: '20em', // max-width for the sub menus (any
CSS unit) - if set, the fixed width set in CSS will be ignored
    subIndicators: true, // create sub menu indicators - creates a SPAN
and inserts it in the A
    subIndicatorsPos: 'prepend', // position of the SPAN relative to the menu item
content ('prepend', 'append')
    subIndicatorsText: '+', // [optionally] add text in the SPAN (e.g. '+') (you
may want to check the CSS for the sub indicators too)
    scrollStep: 30, // pixels step when scrolling long sub menus that
do not fit in the viewport height
    scrollAccelerate: true, // accelerate scrolling or use a fixed step
    showTimeout: 250, // timeout before showing the sub menus
    hideTimeout: 500, // timeout before hiding the sub menus
    showDuration: 0, // duration for show animation - set to 0 for no
animation - matters only if showFunction:null
    showFunction: null, // custom function to use when showing a sub
menu (the default is the jQuery 'show')
    // don't forget to call complete() at the end of
whatever you do
    // e.g.: function($ul, complete) { $ul.fadeIn(250,
complete); }
    hideDuration: 0, // duration for hide animation - set to 0 for no
animation - matters only if hideFunction:null
    hideFunction: function($ul, complete) { $ul.fadeOut(200, complete); }, //
custom function to use when hiding a sub menu (the default is the jQuery 'hide')

```

```

// don't forget to call complete() at the end of
whatever you do // e.g.: function($ul, complete) { $ul.fadeOut(250,
complete); }
    collapsibleShowDuration:0, // duration for show animation for collapsible sub
menus - matters only if collapsibleShowFunction:null
    collapsibleShowFunction:function($ul, complete) { $ul.slideDown(200, complete); },
// custom function to use when showing a collapsible sub menu
// (i.e. when mobile styles are used to make the
sub menus collapsible)
    collapsibleHideDuration:0, // duration for hide animation for collapsible sub
menus - matters only if collapsibleHideFunction:null
    collapsibleHideFunction:function($ul, complete) { $ul.slideUp(200, complete); }, //
custom function to use when hiding a collapsible sub menu
// (i.e. when mobile styles are used to make the
sub menus collapsible)
    showOnClick: false, // show the first-level sub menus onclick instead
of onmouseover (i.e. mimic desktop app menus) (matters only for mouse input)
    hideOnClick: true, // hide the sub menus on click/tap anywhere on
the page
    noMouseOver: false, // disable sub menu activation onmouseover
(i.e. behave like in touch mode - use just mouse clicks) (matters only for mouse input)
    keepInViewport: true, // reposition the sub menus if needed to
make sure they always appear inside the viewport
    keepHighlighted: true, // keep all ancestor items of the current sub
menu highlighted (adds the 'highlighted' class to the A's)
    markCurrentItem: false, // automatically add the 'current' class to the A
element of the item linking to the current URL
    markCurrentTree: true, // add the 'current' class also to the A elements
of all ancestor items of the current item
    rightToLeftSubMenus: false, // right to left display of the sub menus (check
the CSS for the sub indicators' position)
    bottomToTopSubMenus: false, // bottom to top display of the sub
menus
    overlapControlsInIE: true // make sure sub menus appear on top of
special OS controls in IE (i.e. SELECT, OBJECT, EMBED, etc.)
};

return $;
});

```