import $ from 'jquery'

const NAME = 'ControlSidebar'

const DATA\_KEY = 'lte.controlsidebar'

const EVENT\_KEY = `.${DATA\_KEY}`

const JQUERY\_NO\_CONFLICT = $.fn[NAME]

const EVENT\_COLLAPSED = `collapsed${EVENT\_KEY}`

const EVENT\_COLLAPSED\_DONE = `collapsed-done${EVENT\_KEY}`

const EVENT\_EXPANDED = `expanded${EVENT\_KEY}`

const SELECTOR\_CONTROL\_SIDEBAR = '.control-sidebar'

const SELECTOR\_CONTROL\_SIDEBAR\_CONTENT = '.control-sidebar-content'

const SELECTOR\_DATA\_TOGGLE = '[data-widget="control-sidebar"]'

const SELECTOR\_HEADER = '.main-header'

const SELECTOR\_FOOTER = '.main-footer'

const CLASS\_NAME\_CONTROL\_SIDEBAR\_ANIMATE = 'control-sidebar-animate'

const CLASS\_NAME\_CONTROL\_SIDEBAR\_OPEN = 'control-sidebar-open'

const CLASS\_NAME\_CONTROL\_SIDEBAR\_SLIDE = 'control-sidebar-slide-open'

const CLASS\_NAME\_LAYOUT\_FIXED = 'layout-fixed'

const CLASS\_NAME\_NAVBAR\_FIXED = 'layout-navbar-fixed'

const CLASS\_NAME\_NAVBAR\_SM\_FIXED = 'layout-sm-navbar-fixed'

const CLASS\_NAME\_NAVBAR\_MD\_FIXED = 'layout-md-navbar-fixed'

const CLASS\_NAME\_NAVBAR\_LG\_FIXED = 'layout-lg-navbar-fixed'

const CLASS\_NAME\_NAVBAR\_XL\_FIXED = 'layout-xl-navbar-fixed'

const CLASS\_NAME\_FOOTER\_FIXED = 'layout-footer-fixed'

const CLASS\_NAME\_FOOTER\_SM\_FIXED = 'layout-sm-footer-fixed'

const CLASS\_NAME\_FOOTER\_MD\_FIXED = 'layout-md-footer-fixed'

const CLASS\_NAME\_FOOTER\_LG\_FIXED = 'layout-lg-footer-fixed'

const CLASS\_NAME\_FOOTER\_XL\_FIXED = 'layout-xl-footer-fixed'

const Default = {

controlsidebarSlide: true,

scrollbarTheme: 'os-theme-light',

scrollbarAutoHide: 'l',

target: SELECTOR\_CONTROL\_SIDEBAR,

animationSpeed: 300

}

class ControlSidebar {

constructor(element, config) {

this.\_element = element

this.\_config = config

}

collapse() {

const $body = $('body')

const $html = $('html')

// Show the control sidebar

if (this.\_config.controlsidebarSlide) {

$html.addClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_ANIMATE)

$body.removeClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_SLIDE).delay(300).queue(function () {

$(SELECTOR\_CONTROL\_SIDEBAR).hide()

$html.removeClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_ANIMATE)

$(this).dequeue()

})

} else {

$body.removeClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_OPEN)

}

$(this.\_element).trigger($.Event(EVENT\_COLLAPSED))

setTimeout(() => {

$(this.\_element).trigger($.Event(EVENT\_COLLAPSED\_DONE))

}, this.\_config.animationSpeed)

}

show(toggle = false) {

const $body = $('body')

const $html = $('html')

if (toggle) {

$(SELECTOR\_CONTROL\_SIDEBAR).hide()

}

// Collapse the control sidebar

if (this.\_config.controlsidebarSlide) {

$html.addClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_ANIMATE)

$(this.\_config.target).show().delay(10).queue(function () {

$body.addClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_SLIDE).delay(300).queue(function () {

$html.removeClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_ANIMATE)

$(this).dequeue()

})

$(this).dequeue()

})

} else {

$body.addClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_OPEN)

}

this.\_fixHeight()

this.\_fixScrollHeight()

$(this.\_element).trigger($.Event(EVENT\_EXPANDED))

}

toggle() {

const $body = $('body')

const { target } = this.\_config

const notVisible = !$(target).is(':visible')

const shouldClose = ($body.hasClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_OPEN) ||

$body.hasClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_SLIDE))

const shouldToggle = notVisible && ($body.hasClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_OPEN) ||

$body.hasClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_SLIDE))

if (notVisible || shouldToggle) {

// Open the control sidebar

this.show(notVisible)

} else if (shouldClose) {

// Close the control sidebar

this.collapse()

}

}

// Private

\_init() {

const $body = $('body')

const shouldNotHideAll = $body.hasClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_OPEN) ||

$body.hasClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_SLIDE)

if (shouldNotHideAll) {

$(SELECTOR\_CONTROL\_SIDEBAR).not(this.\_config.target).hide()

$(this.\_config.target).css('display', 'block')

} else {

$(SELECTOR\_CONTROL\_SIDEBAR).hide()

}

this.\_fixHeight()

this.\_fixScrollHeight()

$(window).resize(() => {

this.\_fixHeight()

this.\_fixScrollHeight()

})

$(window).scroll(() => {

const $body = $('body')

const shouldFixHeight = $body.hasClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_OPEN) ||

$body.hasClass(CLASS\_NAME\_CONTROL\_SIDEBAR\_SLIDE)

if (shouldFixHeight) {

this.\_fixScrollHeight()

}

})

}

\_isNavbarFixed() {

const $body = $('body')

return (

$body.hasClass(CLASS\_NAME\_NAVBAR\_FIXED) ||

$body.hasClass(CLASS\_NAME\_NAVBAR\_SM\_FIXED) ||

$body.hasClass(CLASS\_NAME\_NAVBAR\_MD\_FIXED) ||

$body.hasClass(CLASS\_NAME\_NAVBAR\_LG\_FIXED) ||

$body.hasClass(CLASS\_NAME\_NAVBAR\_XL\_FIXED)

)

}

\_isFooterFixed() {

const $body = $('body')

return (

$body.hasClass(CLASS\_NAME\_FOOTER\_FIXED) ||

$body.hasClass(CLASS\_NAME\_FOOTER\_SM\_FIXED) ||

$body.hasClass(CLASS\_NAME\_FOOTER\_MD\_FIXED) ||

$body.hasClass(CLASS\_NAME\_FOOTER\_LG\_FIXED) ||

$body.hasClass(CLASS\_NAME\_FOOTER\_XL\_FIXED)

)

}

\_fixScrollHeight() {

const $body = $('body')

const $controlSidebar = $(this.\_config.target)

if (!$body.hasClass(CLASS\_NAME\_LAYOUT\_FIXED)) {

return

}

const heights = {

scroll: $(document).height(),

window: $(window).height(),

header: $(SELECTOR\_HEADER).outerHeight(),

footer: $(SELECTOR\_FOOTER).outerHeight()

}

const positions = {

bottom: Math.abs((heights.window + $(window).scrollTop()) - heights.scroll),

top: $(window).scrollTop()

}

const navbarFixed = this.\_isNavbarFixed() && $(SELECTOR\_HEADER).css('position') === 'fixed'

const footerFixed = this.\_isFooterFixed() && $(SELECTOR\_FOOTER).css('position') === 'fixed'

const $controlsidebarContent = $(`${this.\_config.target}, ${this.\_config.target} ${SELECTOR\_CONTROL\_SIDEBAR\_CONTENT}`)

if (positions.top === 0 && positions.bottom === 0) {

$controlSidebar.css({

bottom: heights.footer,

top: heights.header

})

$controlsidebarContent.css('height', heights.window - (heights.header + heights.footer))

} else if (positions.bottom <= heights.footer) {

if (footerFixed === false) {

const top = heights.header - positions.top

$controlSidebar.css('bottom', heights.footer - positions.bottom).css('top', top >= 0 ? top : 0)

$controlsidebarContent.css('height', heights.window - (heights.footer - positions.bottom))

} else {

$controlSidebar.css('bottom', heights.footer)

}

} else if (positions.top <= heights.header) {

if (navbarFixed === false) {

$controlSidebar.css('top', heights.header - positions.top)

$controlsidebarContent.css('height', heights.window - (heights.header - positions.top))

} else {

$controlSidebar.css('top', heights.header)

}

} else if (navbarFixed === false) {

$controlSidebar.css('top', 0)

$controlsidebarContent.css('height', heights.window)

} else {

$controlSidebar.css('top', heights.header)

}

if (footerFixed && navbarFixed) {

$controlsidebarContent.css('height', '100%')

$controlSidebar.css('height', '')

} else if (footerFixed || navbarFixed) {

$controlsidebarContent.css('height', '100%')

$controlsidebarContent.css('height', '')

}

}

\_fixHeight() {

const $body = $('body')

const $controlSidebar = $(`${this.\_config.target} ${SELECTOR\_CONTROL\_SIDEBAR\_CONTENT}`)

if (!$body.hasClass(CLASS\_NAME\_LAYOUT\_FIXED)) {

$controlSidebar.attr('style', '')

return

}

const heights = {

window: $(window).height(),

header: $(SELECTOR\_HEADER).outerHeight(),

footer: $(SELECTOR\_FOOTER).outerHeight()

}

let sidebarHeight = heights.window - heights.header

if (this.\_isFooterFixed() && $(SELECTOR\_FOOTER).css('position') === 'fixed') {

sidebarHeight = heights.window - heights.header - heights.footer

}

$controlSidebar.css('height', sidebarHeight)

if (typeof $.fn.overlayScrollbars !== 'undefined') {

$controlSidebar.overlayScrollbars({

className: this.\_config.scrollbarTheme,

sizeAutoCapable: true,

scrollbars: {

autoHide: this.\_config.scrollbarAutoHide,

clickScrolling: true

}

})

}

}

// Static

static \_jQueryInterface(config) {

return this.each(function () {

let data = $(this).data(DATA\_KEY)

const \_config = $.extend({}, Default, typeof config === 'object' ? config : $(this).data())

if (!data) {

data = new ControlSidebar($(this), \_config)

$(this).data(DATA\_KEY, data)

data.\_init()

} else if (typeof config === 'string') {

if (typeof data[config] === 'undefined') {

throw new TypeError(`No method named "${config}"`)

}

data[config]()

} else if (typeof config === 'undefined') {

data.\_init()

}

})

}

}

$(document).on('click', SELECTOR\_DATA\_TOGGLE, function (event) {

event.preventDefault()

ControlSidebar.\_jQueryInterface.call($(this), 'toggle')

})

$(document).ready(() => {

ControlSidebar.\_jQueryInterface.call($(SELECTOR\_DATA\_TOGGLE), '\_init')

})

$.fn[NAME] = ControlSidebar.\_jQueryInterface

$.fn[NAME].Constructor = ControlSidebar

$.fn[NAME].noConflict = function () {

$.fn[NAME] = JQUERY\_NO\_CONFLICT

return ControlSidebar.\_jQueryInterface

}

export default ControlSidebar

Footer