import $ from 'jquery'

const NAME = 'PushMenu'

const DATA\_KEY = 'lte.pushmenu'

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\_SHOWN = `shown${EVENT\_KEY}`

const SELECTOR\_TOGGLE\_BUTTON = '[data-widget="pushmenu"]'

const SELECTOR\_BODY = 'body'

const SELECTOR\_OVERLAY = '#sidebar-overlay'

const SELECTOR\_WRAPPER = '.wrapper'

const CLASS\_NAME\_COLLAPSED = 'sidebar-collapse'

const CLASS\_NAME\_OPEN = 'sidebar-open'

const CLASS\_NAME\_IS\_OPENING = 'sidebar-is-opening'

const CLASS\_NAME\_CLOSED = 'sidebar-closed'

const Default = {

autoCollapseSize: 992,

enableRemember: false,

noTransitionAfterReload: true,

animationSpeed: 300

}

class PushMenu {

constructor(element, options) {

this.\_element = element

this.\_options = options

if ($(SELECTOR\_OVERLAY).length === 0) {

this.\_addOverlay()

}

this.\_init()

}

// Public

expand() {

const $bodySelector = $(SELECTOR\_BODY)

if (this.\_options.autoCollapseSize && $(window).width() <= this.\_options.autoCollapseSize) {

$bodySelector.addClass(CLASS\_NAME\_OPEN)

}

$bodySelector.addClass(CLASS\_NAME\_IS\_OPENING).removeClass(`${CLASS\_NAME\_COLLAPSED} ${CLASS\_NAME\_CLOSED}`).delay(50).queue(function () {

$bodySelector.removeClass(CLASS\_NAME\_IS\_OPENING)

$(this).dequeue()

})

if (this.\_options.enableRemember) {

localStorage.setItem(`remember${EVENT\_KEY}`, CLASS\_NAME\_OPEN)

}

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

}

collapse() {

const $bodySelector = $(SELECTOR\_BODY)

if (this.\_options.autoCollapseSize && $(window).width() <= this.\_options.autoCollapseSize) {

$bodySelector.removeClass(CLASS\_NAME\_OPEN).addClass(CLASS\_NAME\_CLOSED)

}

$bodySelector.addClass(CLASS\_NAME\_COLLAPSED)

if (this.\_options.enableRemember) {

localStorage.setItem(`remember${EVENT\_KEY}`, CLASS\_NAME\_COLLAPSED)

}

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

setTimeout(() => {

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

}, this.\_options.animationSpeed)

}

toggle() {

if ($(SELECTOR\_BODY).hasClass(CLASS\_NAME\_COLLAPSED)) {

this.expand()

} else {

this.collapse()

}

}

autoCollapse(resize = false) {

if (!this.\_options.autoCollapseSize) {

return

}

const $bodySelector = $(SELECTOR\_BODY)

if ($(window).width() <= this.\_options.autoCollapseSize) {

if (!$bodySelector.hasClass(CLASS\_NAME\_OPEN)) {

this.collapse()

}

} else if (resize === true) {

if ($bodySelector.hasClass(CLASS\_NAME\_OPEN)) {

$bodySelector.removeClass(CLASS\_NAME\_OPEN)

} else if ($bodySelector.hasClass(CLASS\_NAME\_CLOSED)) {

this.expand()

}

}

}

remember() {

if (!this.\_options.enableRemember) {

return

}

const $body = $('body')

const toggleState = localStorage.getItem(`remember${EVENT\_KEY}`)

if (toggleState === CLASS\_NAME\_COLLAPSED) {

if (this.\_options.noTransitionAfterReload) {

$body.addClass('hold-transition').addClass(CLASS\_NAME\_COLLAPSED).delay(50).queue(function () {

$(this).removeClass('hold-transition')

$(this).dequeue()

})

} else {

$body.addClass(CLASS\_NAME\_COLLAPSED)

}

} else if (this.\_options.noTransitionAfterReload) {

$body.addClass('hold-transition').removeClass(CLASS\_NAME\_COLLAPSED).delay(50).queue(function () {

$(this).removeClass('hold-transition')

$(this).dequeue()

})

} else {

$body.removeClass(CLASS\_NAME\_COLLAPSED)

}

}

// Private

\_init() {

this.remember()

this.autoCollapse()

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

this.autoCollapse(true)

})

}

\_addOverlay() {

const overlay = $('<div />', {

id: 'sidebar-overlay'

})

overlay.on('click', () => {

this.collapse()

})

$(SELECTOR\_WRAPPER).append(overlay)

}

// 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 PushMenu($(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\_TOGGLE\_BUTTON, event => {

event.preventDefault()

let button = event.currentTarget

if ($(button).data('widget') !== 'pushmenu') {

button = $(button).closest(SELECTOR\_TOGGLE\_BUTTON)

}

PushMenu.\_jQueryInterface.call($(button), 'toggle')

})

$(window).on('load', () => {

PushMenu.\_jQueryInterface.call($(SELECTOR\_TOGGLE\_BUTTON))

})

$.fn[NAME] = PushMenu.\_jQueryInterface

$.fn[NAME].Constructor = PushMenu

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

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

return PushMenu.\_jQueryInterface

}

export default PushMenu

Footer