import $ from 'jquery'

/\*\*

\* Constants

\* ====================================================

\*/

const NAME = 'CardRefresh'

const DATA\_KEY = 'lte.cardrefresh'

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

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

const EVENT\_LOADED = `loaded${EVENT\_KEY}`

const EVENT\_OVERLAY\_ADDED = `overlay.added${EVENT\_KEY}`

const EVENT\_OVERLAY\_REMOVED = `overlay.removed${EVENT\_KEY}`

const CLASS\_NAME\_CARD = 'card'

const SELECTOR\_CARD = `.${CLASS\_NAME\_CARD}`

const SELECTOR\_DATA\_REFRESH = '[data-card-widget="card-refresh"]'

const Default = {

source: '',

sourceSelector: '',

params: {},

trigger: SELECTOR\_DATA\_REFRESH,

content: '.card-body',

loadInContent: true,

loadOnInit: true,

loadErrorTemplate: true,

responseType: '',

overlayTemplate: '<div class="overlay"><i class="fas fa-2x fa-sync-alt fa-spin"></i></div>',

errorTemplate: '<span class="text-danger"></span>',

onLoadStart() {},

onLoadDone(response) {

return response

},

onLoadFail(\_jqXHR, \_textStatus, \_errorThrown) {}

}

class CardRefresh {

constructor(element, settings) {

this.\_element = element

this.\_parent = element.parents(SELECTOR\_CARD).first()

this.\_settings = $.extend({}, Default, settings)

this.\_overlay = $(this.\_settings.overlayTemplate)

if (element.hasClass(CLASS\_NAME\_CARD)) {

this.\_parent = element

}

if (this.\_settings.source === '') {

throw new Error('Source url was not defined. Please specify a url in your CardRefresh source option.')

}

}

load() {

this.\_addOverlay()

this.\_settings.onLoadStart.call($(this))

$.get(this.\_settings.source, this.\_settings.params, response => {

if (this.\_settings.loadInContent) {

if (this.\_settings.sourceSelector !== '') {

response = $(response).find(this.\_settings.sourceSelector).html()

}

this.\_parent.find(this.\_settings.content).html(response)

}

this.\_settings.onLoadDone.call($(this), response)

this.\_removeOverlay()

}, this.\_settings.responseType !== '' && this.\_settings.responseType)

.fail((jqXHR, textStatus, errorThrown) => {

this.\_removeOverlay()

if (this.\_settings.loadErrorTemplate) {

const msg = $(this.\_settings.errorTemplate).text(errorThrown)

this.\_parent.find(this.\_settings.content).empty().append(msg)

}

this.\_settings.onLoadFail.call($(this), jqXHR, textStatus, errorThrown)

})

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

}

\_addOverlay() {

this.\_parent.append(this.\_overlay)

$(this.\_element).trigger($.Event(EVENT\_OVERLAY\_ADDED))

}

\_removeOverlay() {

this.\_parent.find(this.\_overlay).remove()

$(this.\_element).trigger($.Event(EVENT\_OVERLAY\_REMOVED))

}

// Private

\_init() {

$(this).find(this.\_settings.trigger).on('click', () => {

this.load()

})

if (this.\_settings.loadOnInit) {

this.load()

}

}

// 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 CardRefresh($(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()

}

})

}

}

/\*\*

\* Data API

\* ====================================================

\*/

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

if (event) {

event.preventDefault()

}

CardRefresh.\_jQueryInterface.call($(this), 'load')

})

$(() => {

$(SELECTOR\_DATA\_REFRESH).each(function () {

CardRefresh.\_jQueryInterface.call($(this))

})

})

/\*\*

\* jQuery API

\* ====================================================

\*/

$.fn[NAME] = CardRefresh.\_jQueryInterface

$.fn[NAME].Constructor = CardRefresh

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

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

return CardRefresh.\_jQueryInterface

}

export default CardRefresh