/\*!

\* Bootstrap v4.0.0 (https://getbootstrap.com)

\* Copyright 2011-2018 The Bootstrap Authors (https://github.com/twbs/bootstrap/graphs/contributors)

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\*/

(function (global, factory) {

typeof exports === 'object' && typeof module !== 'undefined' ? factory(exports, require('jquery'), require('popper.js')) :

typeof define === 'function' && define.amd ? define(['exports', 'jquery', 'popper.js'], factory) :

(factory((global.bootstrap = {}),global.jQuery,global.Popper));

}(this, (function (exports,$,Popper) { 'use strict';

$ = $ && $.hasOwnProperty('default') ? $['default'] : $;

Popper = Popper && Popper.hasOwnProperty('default') ? Popper['default'] : Popper;

function \_defineProperties(target, props) {

for (var i = 0; i < props.length; i++) {

var descriptor = props[i];

descriptor.enumerable = descriptor.enumerable || false;

descriptor.configurable = true;

if ("value" in descriptor) descriptor.writable = true;

Object.defineProperty(target, descriptor.key, descriptor);

}

}

function \_createClass(Constructor, protoProps, staticProps) {

if (protoProps) \_defineProperties(Constructor.prototype, protoProps);

if (staticProps) \_defineProperties(Constructor, staticProps);

return Constructor;

}

function \_extends() {

\_extends = Object.assign || function (target) {

for (var i = 1; i < arguments.length; i++) {

var source = arguments[i];

for (var key in source) {

if (Object.prototype.hasOwnProperty.call(source, key)) {

target[key] = source[key];

}

}

}

return target;

};

return \_extends.apply(this, arguments);

}

function \_inheritsLoose(subClass, superClass) {

subClass.prototype = Object.create(superClass.prototype);

subClass.prototype.constructor = subClass;

subClass.\_\_proto\_\_ = superClass;

}

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0): util.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

var Util = function ($$$1) {

/\*\*

\* ------------------------------------------------------------------------

\* Private TransitionEnd Helpers

\* ------------------------------------------------------------------------

\*/

var transition = false;

var MAX\_UID = 1000000; // Shoutout AngusCroll (https://goo.gl/pxwQGp)

function toType(obj) {

return {}.toString.call(obj).match(/\s([a-zA-Z]+)/)[1].toLowerCase();

}

function getSpecialTransitionEndEvent() {

return {

bindType: transition.end,

delegateType: transition.end,

handle: function handle(event) {

if ($$$1(event.target).is(this)) {

return event.handleObj.handler.apply(this, arguments); // eslint-disable-line prefer-rest-params

}

return undefined; // eslint-disable-line no-undefined

}

};

}

function transitionEndTest() {

if (typeof window !== 'undefined' && window.QUnit) {

return false;

}

return {

end: 'transitionend'

};

}

function transitionEndEmulator(duration) {

var \_this = this;

var called = false;

$$$1(this).one(Util.TRANSITION\_END, function () {

called = true;

});

setTimeout(function () {

if (!called) {

Util.triggerTransitionEnd(\_this);

}

}, duration);

return this;

}

function setTransitionEndSupport() {

transition = transitionEndTest();

$$$1.fn.emulateTransitionEnd = transitionEndEmulator;

if (Util.supportsTransitionEnd()) {

$$$1.event.special[Util.TRANSITION\_END] = getSpecialTransitionEndEvent();

}

}

function escapeId(selector) {

// We escape IDs in case of special selectors (selector = '#myId:something')

// $.escapeSelector does not exist in jQuery < 3

selector = typeof $$$1.escapeSelector === 'function' ? $$$1.escapeSelector(selector).substr(1) : selector.replace(/(:|\.|\[|\]|,|=|@)/g, '\\$1');

return selector;

}

/\*\*

\* --------------------------------------------------------------------------

\* Public Util Api

\* --------------------------------------------------------------------------

\*/

var Util = {

TRANSITION\_END: 'bsTransitionEnd',

getUID: function getUID(prefix) {

do {

// eslint-disable-next-line no-bitwise

prefix += ~~(Math.random() \* MAX\_UID); // "~~" acts like a faster Math.floor() here

} while (document.getElementById(prefix));

return prefix;

},

getSelectorFromElement: function getSelectorFromElement(element) {

var selector = element.getAttribute('data-target');

if (!selector || selector === '#') {

selector = element.getAttribute('href') || '';

} // If it's an ID

if (selector.charAt(0) === '#') {

selector = escapeId(selector);

}

try {

var $selector = $$$1(document).find(selector);

return $selector.length > 0 ? selector : null;

} catch (err) {

return null;

}

},

reflow: function reflow(element) {

return element.offsetHeight;

},

triggerTransitionEnd: function triggerTransitionEnd(element) {

$$$1(element).trigger(transition.end);

},

supportsTransitionEnd: function supportsTransitionEnd() {

return Boolean(transition);

},

isElement: function isElement(obj) {

return (obj[0] || obj).nodeType;

},

typeCheckConfig: function typeCheckConfig(componentName, config, configTypes) {

for (var property in configTypes) {

if (Object.prototype.hasOwnProperty.call(configTypes, property)) {

var expectedTypes = configTypes[property];

var value = config[property];

var valueType = value && Util.isElement(value) ? 'element' : toType(value);

if (!new RegExp(expectedTypes).test(valueType)) {

throw new Error(componentName.toUpperCase() + ": " + ("Option \"" + property + "\" provided type \"" + valueType + "\" ") + ("but expected type \"" + expectedTypes + "\"."));

}

}

}

}

};

setTransitionEndSupport();

return Util;

}($);

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0): alert.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

var Alert = function ($$$1) {

/\*\*

\* ------------------------------------------------------------------------

\* Constants

\* ------------------------------------------------------------------------

\*/

var NAME = 'alert';

var VERSION = '4.0.0';

var DATA\_KEY = 'bs.alert';

var EVENT\_KEY = "." + DATA\_KEY;

var DATA\_API\_KEY = '.data-api';

var JQUERY\_NO\_CONFLICT = $$$1.fn[NAME];

var TRANSITION\_DURATION = 150;

var Selector = {

DISMISS: '[data-dismiss="alert"]'

};

var Event = {

CLOSE: "close" + EVENT\_KEY,

CLOSED: "closed" + EVENT\_KEY,

CLICK\_DATA\_API: "click" + EVENT\_KEY + DATA\_API\_KEY

};

var ClassName = {

ALERT: 'alert',

FADE: 'fade',

SHOW: 'show'

/\*\*

\* ------------------------------------------------------------------------

\* Class Definition

\* ------------------------------------------------------------------------

\*/

};

var Alert =

/\*#\_\_PURE\_\_\*/

function () {

function Alert(element) {

this.\_element = element;

} // Getters

var \_proto = Alert.prototype;

// Public

\_proto.close = function close(element) {

element = element || this.\_element;

var rootElement = this.\_getRootElement(element);

var customEvent = this.\_triggerCloseEvent(rootElement);

if (customEvent.isDefaultPrevented()) {

return;

}

this.\_removeElement(rootElement);

};

\_proto.dispose = function dispose() {

$$$1.removeData(this.\_element, DATA\_KEY);

this.\_element = null;

}; // Private

\_proto.\_getRootElement = function \_getRootElement(element) {

var selector = Util.getSelectorFromElement(element);

var parent = false;

if (selector) {

parent = $$$1(selector)[0];

}

if (!parent) {

parent = $$$1(element).closest("." + ClassName.ALERT)[0];

}

return parent;

};

\_proto.\_triggerCloseEvent = function \_triggerCloseEvent(element) {

var closeEvent = $$$1.Event(Event.CLOSE);

$$$1(element).trigger(closeEvent);

return closeEvent;

};

\_proto.\_removeElement = function \_removeElement(element) {

var \_this = this;

$$$1(element).removeClass(ClassName.SHOW);

if (!Util.supportsTransitionEnd() || !$$$1(element).hasClass(ClassName.FADE)) {

this.\_destroyElement(element);

return;

}

$$$1(element).one(Util.TRANSITION\_END, function (event) {

return \_this.\_destroyElement(element, event);

}).emulateTransitionEnd(TRANSITION\_DURATION);

};

\_proto.\_destroyElement = function \_destroyElement(element) {

$$$1(element).detach().trigger(Event.CLOSED).remove();

}; // Static

Alert.\_jQueryInterface = function \_jQueryInterface(config) {

return this.each(function () {

var $element = $$$1(this);

var data = $element.data(DATA\_KEY);

if (!data) {

data = new Alert(this);

$element.data(DATA\_KEY, data);

}

if (config === 'close') {

data[config](this);

}

});

};

Alert.\_handleDismiss = function \_handleDismiss(alertInstance) {

return function (event) {

if (event) {

event.preventDefault();

}

alertInstance.close(this);

};

};

\_createClass(Alert, null, [{

key: "VERSION",

get: function get() {

return VERSION;

}

}]);

return Alert;

}();

/\*\*

\* ------------------------------------------------------------------------

\* Data Api implementation

\* ------------------------------------------------------------------------

\*/

$$$1(document).on(Event.CLICK\_DATA\_API, Selector.DISMISS, Alert.\_handleDismiss(new Alert()));

/\*\*

\* ------------------------------------------------------------------------

\* jQuery

\* ------------------------------------------------------------------------

\*/

$$$1.fn[NAME] = Alert.\_jQueryInterface;

$$$1.fn[NAME].Constructor = Alert;

$$$1.fn[NAME].noConflict = function () {

$$$1.fn[NAME] = JQUERY\_NO\_CONFLICT;

return Alert.\_jQueryInterface;

};

return Alert;

}($);

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0): button.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

var Button = function ($$$1) {

/\*\*

\* ------------------------------------------------------------------------

\* Constants

\* ------------------------------------------------------------------------

\*/

var NAME = 'button';

var VERSION = '4.0.0';

var DATA\_KEY = 'bs.button';

var EVENT\_KEY = "." + DATA\_KEY;

var DATA\_API\_KEY = '.data-api';

var JQUERY\_NO\_CONFLICT = $$$1.fn[NAME];

var ClassName = {

ACTIVE: 'active',

BUTTON: 'btn',

FOCUS: 'focus'

};

var Selector = {

DATA\_TOGGLE\_CARROT: '[data-toggle^="button"]',

DATA\_TOGGLE: '[data-toggle="buttons"]',

INPUT: 'input',

ACTIVE: '.active',

BUTTON: '.btn'

};

var Event = {

CLICK\_DATA\_API: "click" + EVENT\_KEY + DATA\_API\_KEY,

FOCUS\_BLUR\_DATA\_API: "focus" + EVENT\_KEY + DATA\_API\_KEY + " " + ("blur" + EVENT\_KEY + DATA\_API\_KEY)

/\*\*

\* ------------------------------------------------------------------------

\* Class Definition

\* ------------------------------------------------------------------------

\*/

};

var Button =

/\*#\_\_PURE\_\_\*/

function () {

function Button(element) {

this.\_element = element;

} // Getters

var \_proto = Button.prototype;

// Public

\_proto.toggle = function toggle() {

var triggerChangeEvent = true;

var addAriaPressed = true;

var rootElement = $$$1(this.\_element).closest(Selector.DATA\_TOGGLE)[0];

if (rootElement) {

var input = $$$1(this.\_element).find(Selector.INPUT)[0];

if (input) {

if (input.type === 'radio') {

if (input.checked && $$$1(this.\_element).hasClass(ClassName.ACTIVE)) {

triggerChangeEvent = false;

} else {

var activeElement = $$$1(rootElement).find(Selector.ACTIVE)[0];

if (activeElement) {

$$$1(activeElement).removeClass(ClassName.ACTIVE);

}

}

}

if (triggerChangeEvent) {

if (input.hasAttribute('disabled') || rootElement.hasAttribute('disabled') || input.classList.contains('disabled') || rootElement.classList.contains('disabled')) {

return;

}

input.checked = !$$$1(this.\_element).hasClass(ClassName.ACTIVE);

$$$1(input).trigger('change');

}

input.focus();

addAriaPressed = false;

}

}

if (addAriaPressed) {

this.\_element.setAttribute('aria-pressed', !$$$1(this.\_element).hasClass(ClassName.ACTIVE));

}

if (triggerChangeEvent) {

$$$1(this.\_element).toggleClass(ClassName.ACTIVE);

}

};

\_proto.dispose = function dispose() {

$$$1.removeData(this.\_element, DATA\_KEY);

this.\_element = null;

}; // Static

Button.\_jQueryInterface = function \_jQueryInterface(config) {

return this.each(function () {

var data = $$$1(this).data(DATA\_KEY);

if (!data) {

data = new Button(this);

$$$1(this).data(DATA\_KEY, data);

}

if (config === 'toggle') {

data[config]();

}

});

};

\_createClass(Button, null, [{

key: "VERSION",

get: function get() {

return VERSION;

}

}]);

return Button;

}();

/\*\*

\* ------------------------------------------------------------------------

\* Data Api implementation

\* ------------------------------------------------------------------------

\*/

$$$1(document).on(Event.CLICK\_DATA\_API, Selector.DATA\_TOGGLE\_CARROT, function (event) {

event.preventDefault();

var button = event.target;

if (!$$$1(button).hasClass(ClassName.BUTTON)) {

button = $$$1(button).closest(Selector.BUTTON);

}

Button.\_jQueryInterface.call($$$1(button), 'toggle');

}).on(Event.FOCUS\_BLUR\_DATA\_API, Selector.DATA\_TOGGLE\_CARROT, function (event) {

var button = $$$1(event.target).closest(Selector.BUTTON)[0];

$$$1(button).toggleClass(ClassName.FOCUS, /^focus(in)?$/.test(event.type));

});

/\*\*

\* ------------------------------------------------------------------------

\* jQuery

\* ------------------------------------------------------------------------

\*/

$$$1.fn[NAME] = Button.\_jQueryInterface;

$$$1.fn[NAME].Constructor = Button;

$$$1.fn[NAME].noConflict = function () {

$$$1.fn[NAME] = JQUERY\_NO\_CONFLICT;

return Button.\_jQueryInterface;

};

return Button;

}($);

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0): carousel.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

var Carousel = function ($$$1) {

/\*\*

\* ------------------------------------------------------------------------

\* Constants

\* ------------------------------------------------------------------------

\*/

var NAME = 'carousel';

var VERSION = '4.0.0';

var DATA\_KEY = 'bs.carousel';

var EVENT\_KEY = "." + DATA\_KEY;

var DATA\_API\_KEY = '.data-api';

var JQUERY\_NO\_CONFLICT = $$$1.fn[NAME];

var TRANSITION\_DURATION = 600;

var ARROW\_LEFT\_KEYCODE = 37; // KeyboardEvent.which value for left arrow key

var ARROW\_RIGHT\_KEYCODE = 39; // KeyboardEvent.which value for right arrow key

var TOUCHEVENT\_COMPAT\_WAIT = 500; // Time for mouse compat events to fire after touch

var Default = {

interval: 5000,

keyboard: true,

slide: false,

pause: 'hover',

wrap: true

};

var DefaultType = {

interval: '(number|boolean)',

keyboard: 'boolean',

slide: '(boolean|string)',

pause: '(string|boolean)',

wrap: 'boolean'

};

var Direction = {

NEXT: 'next',

PREV: 'prev',

LEFT: 'left',

RIGHT: 'right'

};

var Event = {

SLIDE: "slide" + EVENT\_KEY,

SLID: "slid" + EVENT\_KEY,

KEYDOWN: "keydown" + EVENT\_KEY,

MOUSEENTER: "mouseenter" + EVENT\_KEY,

MOUSELEAVE: "mouseleave" + EVENT\_KEY,

TOUCHEND: "touchend" + EVENT\_KEY,

LOAD\_DATA\_API: "load" + EVENT\_KEY + DATA\_API\_KEY,

CLICK\_DATA\_API: "click" + EVENT\_KEY + DATA\_API\_KEY

};

var ClassName = {

CAROUSEL: 'carousel',

ACTIVE: 'active',

SLIDE: 'slide',

RIGHT: 'carousel-item-right',

LEFT: 'carousel-item-left',

NEXT: 'carousel-item-next',

PREV: 'carousel-item-prev',

ITEM: 'carousel-item'

};

var Selector = {

ACTIVE: '.active',

ACTIVE\_ITEM: '.active.carousel-item',

ITEM: '.carousel-item',

NEXT\_PREV: '.carousel-item-next, .carousel-item-prev',

INDICATORS: '.carousel-indicators',

DATA\_SLIDE: '[data-slide], [data-slide-to]',

DATA\_RIDE: '[data-ride="carousel"]'

/\*\*

\* ------------------------------------------------------------------------

\* Class Definition

\* ------------------------------------------------------------------------

\*/

};

var Carousel =

/\*#\_\_PURE\_\_\*/

function () {

function Carousel(element, config) {

this.\_items = null;

this.\_interval = null;

this.\_activeElement = null;

this.\_isPaused = false;

this.\_isSliding = false;

this.touchTimeout = null;

this.\_config = this.\_getConfig(config);

this.\_element = $$$1(element)[0];

this.\_indicatorsElement = $$$1(this.\_element).find(Selector.INDICATORS)[0];

this.\_addEventListeners();

} // Getters

var \_proto = Carousel.prototype;

// Public

\_proto.next = function next() {

if (!this.\_isSliding) {

this.\_slide(Direction.NEXT);

}

};

\_proto.nextWhenVisible = function nextWhenVisible() {

// Don't call next when the page isn't visible

// or the carousel or its parent isn't visible

if (!document.hidden && $$$1(this.\_element).is(':visible') && $$$1(this.\_element).css('visibility') !== 'hidden') {

this.next();

}

};

\_proto.prev = function prev() {

if (!this.\_isSliding) {

this.\_slide(Direction.PREV);

}

};

\_proto.pause = function pause(event) {

if (!event) {

this.\_isPaused = true;

}

if ($$$1(this.\_element).find(Selector.NEXT\_PREV)[0] && Util.supportsTransitionEnd()) {

Util.triggerTransitionEnd(this.\_element);

this.cycle(true);

}

clearInterval(this.\_interval);

this.\_interval = null;

};

\_proto.cycle = function cycle(event) {

if (!event) {

this.\_isPaused = false;

}

if (this.\_interval) {

clearInterval(this.\_interval);

this.\_interval = null;

}

if (this.\_config.interval && !this.\_isPaused) {

this.\_interval = setInterval((document.visibilityState ? this.nextWhenVisible : this.next).bind(this), this.\_config.interval);

}

};

\_proto.to = function to(index) {

var \_this = this;

this.\_activeElement = $$$1(this.\_element).find(Selector.ACTIVE\_ITEM)[0];

var activeIndex = this.\_getItemIndex(this.\_activeElement);

if (index > this.\_items.length - 1 || index < 0) {

return;

}

if (this.\_isSliding) {

$$$1(this.\_element).one(Event.SLID, function () {

return \_this.to(index);

});

return;

}

if (activeIndex === index) {

this.pause();

this.cycle();

return;

}

var direction = index > activeIndex ? Direction.NEXT : Direction.PREV;

this.\_slide(direction, this.\_items[index]);

};

\_proto.dispose = function dispose() {

$$$1(this.\_element).off(EVENT\_KEY);

$$$1.removeData(this.\_element, DATA\_KEY);

this.\_items = null;

this.\_config = null;

this.\_element = null;

this.\_interval = null;

this.\_isPaused = null;

this.\_isSliding = null;

this.\_activeElement = null;

this.\_indicatorsElement = null;

}; // Private

\_proto.\_getConfig = function \_getConfig(config) {

config = \_extends({}, Default, config);

Util.typeCheckConfig(NAME, config, DefaultType);

return config;

};

\_proto.\_addEventListeners = function \_addEventListeners() {

var \_this2 = this;

if (this.\_config.keyboard) {

$$$1(this.\_element).on(Event.KEYDOWN, function (event) {

return \_this2.\_keydown(event);

});

}

if (this.\_config.pause === 'hover') {

$$$1(this.\_element).on(Event.MOUSEENTER, function (event) {

return \_this2.pause(event);

}).on(Event.MOUSELEAVE, function (event) {

return \_this2.cycle(event);

});

if ('ontouchstart' in document.documentElement) {

// If it's a touch-enabled device, mouseenter/leave are fired as

// part of the mouse compatibility events on first tap - the carousel

// would stop cycling until user tapped out of it;

// here, we listen for touchend, explicitly pause the carousel

// (as if it's the second time we tap on it, mouseenter compat event

// is NOT fired) and after a timeout (to allow for mouse compatibility

// events to fire) we explicitly restart cycling

$$$1(this.\_element).on(Event.TOUCHEND, function () {

\_this2.pause();

if (\_this2.touchTimeout) {

clearTimeout(\_this2.touchTimeout);

}

\_this2.touchTimeout = setTimeout(function (event) {

return \_this2.cycle(event);

}, TOUCHEVENT\_COMPAT\_WAIT + \_this2.\_config.interval);

});

}

}

};

\_proto.\_keydown = function \_keydown(event) {

if (/input|textarea/i.test(event.target.tagName)) {

return;

}

switch (event.which) {

case ARROW\_LEFT\_KEYCODE:

event.preventDefault();

this.prev();

break;

case ARROW\_RIGHT\_KEYCODE:

event.preventDefault();

this.next();

break;

default:

}

};

\_proto.\_getItemIndex = function \_getItemIndex(element) {

this.\_items = $$$1.makeArray($$$1(element).parent().find(Selector.ITEM));

return this.\_items.indexOf(element);

};

\_proto.\_getItemByDirection = function \_getItemByDirection(direction, activeElement) {

var isNextDirection = direction === Direction.NEXT;

var isPrevDirection = direction === Direction.PREV;

var activeIndex = this.\_getItemIndex(activeElement);

var lastItemIndex = this.\_items.length - 1;

var isGoingToWrap = isPrevDirection && activeIndex === 0 || isNextDirection && activeIndex === lastItemIndex;

if (isGoingToWrap && !this.\_config.wrap) {

return activeElement;

}

var delta = direction === Direction.PREV ? -1 : 1;

var itemIndex = (activeIndex + delta) % this.\_items.length;

return itemIndex === -1 ? this.\_items[this.\_items.length - 1] : this.\_items[itemIndex];

};

\_proto.\_triggerSlideEvent = function \_triggerSlideEvent(relatedTarget, eventDirectionName) {

var targetIndex = this.\_getItemIndex(relatedTarget);

var fromIndex = this.\_getItemIndex($$$1(this.\_element).find(Selector.ACTIVE\_ITEM)[0]);

var slideEvent = $$$1.Event(Event.SLIDE, {

relatedTarget: relatedTarget,

direction: eventDirectionName,

from: fromIndex,

to: targetIndex

});

$$$1(this.\_element).trigger(slideEvent);

return slideEvent;

};

\_proto.\_setActiveIndicatorElement = function \_setActiveIndicatorElement(element) {

if (this.\_indicatorsElement) {

$$$1(this.\_indicatorsElement).find(Selector.ACTIVE).removeClass(ClassName.ACTIVE);

var nextIndicator = this.\_indicatorsElement.children[this.\_getItemIndex(element)];

if (nextIndicator) {

$$$1(nextIndicator).addClass(ClassName.ACTIVE);

}

}

};

\_proto.\_slide = function \_slide(direction, element) {

var \_this3 = this;

var activeElement = $$$1(this.\_element).find(Selector.ACTIVE\_ITEM)[0];

var activeElementIndex = this.\_getItemIndex(activeElement);

var nextElement = element || activeElement && this.\_getItemByDirection(direction, activeElement);

var nextElementIndex = this.\_getItemIndex(nextElement);

var isCycling = Boolean(this.\_interval);

var directionalClassName;

var orderClassName;

var eventDirectionName;

if (direction === Direction.NEXT) {

directionalClassName = ClassName.LEFT;

orderClassName = ClassName.NEXT;

eventDirectionName = Direction.LEFT;

} else {

directionalClassName = ClassName.RIGHT;

orderClassName = ClassName.PREV;

eventDirectionName = Direction.RIGHT;

}

if (nextElement && $$$1(nextElement).hasClass(ClassName.ACTIVE)) {

this.\_isSliding = false;

return;

}

var slideEvent = this.\_triggerSlideEvent(nextElement, eventDirectionName);

if (slideEvent.isDefaultPrevented()) {

return;

}

if (!activeElement || !nextElement) {

// Some weirdness is happening, so we bail

return;

}

this.\_isSliding = true;

if (isCycling) {

this.pause();

}

this.\_setActiveIndicatorElement(nextElement);

var slidEvent = $$$1.Event(Event.SLID, {

relatedTarget: nextElement,

direction: eventDirectionName,

from: activeElementIndex,

to: nextElementIndex

});

if (Util.supportsTransitionEnd() && $$$1(this.\_element).hasClass(ClassName.SLIDE)) {

$$$1(nextElement).addClass(orderClassName);

Util.reflow(nextElement);

$$$1(activeElement).addClass(directionalClassName);

$$$1(nextElement).addClass(directionalClassName);

$$$1(activeElement).one(Util.TRANSITION\_END, function () {

$$$1(nextElement).removeClass(directionalClassName + " " + orderClassName).addClass(ClassName.ACTIVE);

$$$1(activeElement).removeClass(ClassName.ACTIVE + " " + orderClassName + " " + directionalClassName);

\_this3.\_isSliding = false;

setTimeout(function () {

return $$$1(\_this3.\_element).trigger(slidEvent);

}, 0);

}).emulateTransitionEnd(TRANSITION\_DURATION);

} else {

$$$1(activeElement).removeClass(ClassName.ACTIVE);

$$$1(nextElement).addClass(ClassName.ACTIVE);

this.\_isSliding = false;

$$$1(this.\_element).trigger(slidEvent);

}

if (isCycling) {

this.cycle();

}

}; // Static

Carousel.\_jQueryInterface = function \_jQueryInterface(config) {

return this.each(function () {

var data = $$$1(this).data(DATA\_KEY);

var \_config = \_extends({}, Default, $$$1(this).data());

if (typeof config === 'object') {

\_config = \_extends({}, \_config, config);

}

var action = typeof config === 'string' ? config : \_config.slide;

if (!data) {

data = new Carousel(this, \_config);

$$$1(this).data(DATA\_KEY, data);

}

if (typeof config === 'number') {

data.to(config);

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

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

throw new TypeError("No method named \"" + action + "\"");

}

data[action]();

} else if (\_config.interval) {

data.pause();

data.cycle();

}

});

};

Carousel.\_dataApiClickHandler = function \_dataApiClickHandler(event) {

var selector = Util.getSelectorFromElement(this);

if (!selector) {

return;

}

var target = $$$1(selector)[0];

if (!target || !$$$1(target).hasClass(ClassName.CAROUSEL)) {

return;

}

var config = \_extends({}, $$$1(target).data(), $$$1(this).data());

var slideIndex = this.getAttribute('data-slide-to');

if (slideIndex) {

config.interval = false;

}

Carousel.\_jQueryInterface.call($$$1(target), config);

if (slideIndex) {

$$$1(target).data(DATA\_KEY).to(slideIndex);

}

event.preventDefault();

};

\_createClass(Carousel, null, [{

key: "VERSION",

get: function get() {

return VERSION;

}

}, {

key: "Default",

get: function get() {

return Default;

}

}]);

return Carousel;

}();

/\*\*

\* ------------------------------------------------------------------------

\* Data Api implementation

\* ------------------------------------------------------------------------

\*/

$$$1(document).on(Event.CLICK\_DATA\_API, Selector.DATA\_SLIDE, Carousel.\_dataApiClickHandler);

$$$1(window).on(Event.LOAD\_DATA\_API, function () {

$$$1(Selector.DATA\_RIDE).each(function () {

var $carousel = $$$1(this);

Carousel.\_jQueryInterface.call($carousel, $carousel.data());

});

});

/\*\*

\* ------------------------------------------------------------------------

\* jQuery

\* ------------------------------------------------------------------------

\*/

$$$1.fn[NAME] = Carousel.\_jQueryInterface;

$$$1.fn[NAME].Constructor = Carousel;

$$$1.fn[NAME].noConflict = function () {

$$$1.fn[NAME] = JQUERY\_NO\_CONFLICT;

return Carousel.\_jQueryInterface;

};

return Carousel;

}($);

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0): collapse.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

var Collapse = function ($$$1) {

/\*\*

\* ------------------------------------------------------------------------

\* Constants

\* ------------------------------------------------------------------------

\*/

var NAME = 'collapse';

var VERSION = '4.0.0';

var DATA\_KEY = 'bs.collapse';

var EVENT\_KEY = "." + DATA\_KEY;

var DATA\_API\_KEY = '.data-api';

var JQUERY\_NO\_CONFLICT = $$$1.fn[NAME];

var TRANSITION\_DURATION = 600;

var Default = {

toggle: true,

parent: ''

};

var DefaultType = {

toggle: 'boolean',

parent: '(string|element)'

};

var Event = {

SHOW: "show" + EVENT\_KEY,

SHOWN: "shown" + EVENT\_KEY,

HIDE: "hide" + EVENT\_KEY,

HIDDEN: "hidden" + EVENT\_KEY,

CLICK\_DATA\_API: "click" + EVENT\_KEY + DATA\_API\_KEY

};

var ClassName = {

SHOW: 'show',

COLLAPSE: 'collapse',

COLLAPSING: 'collapsing',

COLLAPSED: 'collapsed'

};

var Dimension = {

WIDTH: 'width',

HEIGHT: 'height'

};

var Selector = {

ACTIVES: '.show, .collapsing',

DATA\_TOGGLE: '[data-toggle="collapse"]'

/\*\*

\* ------------------------------------------------------------------------

\* Class Definition

\* ------------------------------------------------------------------------

\*/

};

var Collapse =

/\*#\_\_PURE\_\_\*/

function () {

function Collapse(element, config) {

this.\_isTransitioning = false;

this.\_element = element;

this.\_config = this.\_getConfig(config);

this.\_triggerArray = $$$1.makeArray($$$1("[data-toggle=\"collapse\"][href=\"#" + element.id + "\"]," + ("[data-toggle=\"collapse\"][data-target=\"#" + element.id + "\"]")));

var tabToggles = $$$1(Selector.DATA\_TOGGLE);

for (var i = 0; i < tabToggles.length; i++) {

var elem = tabToggles[i];

var selector = Util.getSelectorFromElement(elem);

if (selector !== null && $$$1(selector).filter(element).length > 0) {

this.\_selector = selector;

this.\_triggerArray.push(elem);

}

}

this.\_parent = this.\_config.parent ? this.\_getParent() : null;

if (!this.\_config.parent) {

this.\_addAriaAndCollapsedClass(this.\_element, this.\_triggerArray);

}

if (this.\_config.toggle) {

this.toggle();

}

} // Getters

var \_proto = Collapse.prototype;

// Public

\_proto.toggle = function toggle() {

if ($$$1(this.\_element).hasClass(ClassName.SHOW)) {

this.hide();

} else {

this.show();

}

};

\_proto.show = function show() {

var \_this = this;

if (this.\_isTransitioning || $$$1(this.\_element).hasClass(ClassName.SHOW)) {

return;

}

var actives;

var activesData;

if (this.\_parent) {

actives = $$$1.makeArray($$$1(this.\_parent).find(Selector.ACTIVES).filter("[data-parent=\"" + this.\_config.parent + "\"]"));

if (actives.length === 0) {

actives = null;

}

}

if (actives) {

activesData = $$$1(actives).not(this.\_selector).data(DATA\_KEY);

if (activesData && activesData.\_isTransitioning) {

return;

}

}

var startEvent = $$$1.Event(Event.SHOW);

$$$1(this.\_element).trigger(startEvent);

if (startEvent.isDefaultPrevented()) {

return;

}

if (actives) {

Collapse.\_jQueryInterface.call($$$1(actives).not(this.\_selector), 'hide');

if (!activesData) {

$$$1(actives).data(DATA\_KEY, null);

}

}

var dimension = this.\_getDimension();

$$$1(this.\_element).removeClass(ClassName.COLLAPSE).addClass(ClassName.COLLAPSING);

this.\_element.style[dimension] = 0;

if (this.\_triggerArray.length > 0) {

$$$1(this.\_triggerArray).removeClass(ClassName.COLLAPSED).attr('aria-expanded', true);

}

this.setTransitioning(true);

var complete = function complete() {

$$$1(\_this.\_element).removeClass(ClassName.COLLAPSING).addClass(ClassName.COLLAPSE).addClass(ClassName.SHOW);

\_this.\_element.style[dimension] = '';

\_this.setTransitioning(false);

$$$1(\_this.\_element).trigger(Event.SHOWN);

};

if (!Util.supportsTransitionEnd()) {

complete();

return;

}

var capitalizedDimension = dimension[0].toUpperCase() + dimension.slice(1);

var scrollSize = "scroll" + capitalizedDimension;

$$$1(this.\_element).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(TRANSITION\_DURATION);

this.\_element.style[dimension] = this.\_element[scrollSize] + "px";

};

\_proto.hide = function hide() {

var \_this2 = this;

if (this.\_isTransitioning || !$$$1(this.\_element).hasClass(ClassName.SHOW)) {

return;

}

var startEvent = $$$1.Event(Event.HIDE);

$$$1(this.\_element).trigger(startEvent);

if (startEvent.isDefaultPrevented()) {

return;

}

var dimension = this.\_getDimension();

this.\_element.style[dimension] = this.\_element.getBoundingClientRect()[dimension] + "px";

Util.reflow(this.\_element);

$$$1(this.\_element).addClass(ClassName.COLLAPSING).removeClass(ClassName.COLLAPSE).removeClass(ClassName.SHOW);

if (this.\_triggerArray.length > 0) {

for (var i = 0; i < this.\_triggerArray.length; i++) {

var trigger = this.\_triggerArray[i];

var selector = Util.getSelectorFromElement(trigger);

if (selector !== null) {

var $elem = $$$1(selector);

if (!$elem.hasClass(ClassName.SHOW)) {

$$$1(trigger).addClass(ClassName.COLLAPSED).attr('aria-expanded', false);

}

}

}

}

this.setTransitioning(true);

var complete = function complete() {

\_this2.setTransitioning(false);

$$$1(\_this2.\_element).removeClass(ClassName.COLLAPSING).addClass(ClassName.COLLAPSE).trigger(Event.HIDDEN);

};

this.\_element.style[dimension] = '';

if (!Util.supportsTransitionEnd()) {

complete();

return;

}

$$$1(this.\_element).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(TRANSITION\_DURATION);

};

\_proto.setTransitioning = function setTransitioning(isTransitioning) {

this.\_isTransitioning = isTransitioning;

};

\_proto.dispose = function dispose() {

$$$1.removeData(this.\_element, DATA\_KEY);

this.\_config = null;

this.\_parent = null;

this.\_element = null;

this.\_triggerArray = null;

this.\_isTransitioning = null;

}; // Private

\_proto.\_getConfig = function \_getConfig(config) {

config = \_extends({}, Default, config);

config.toggle = Boolean(config.toggle); // Coerce string values

Util.typeCheckConfig(NAME, config, DefaultType);

return config;

};

\_proto.\_getDimension = function \_getDimension() {

var hasWidth = $$$1(this.\_element).hasClass(Dimension.WIDTH);

return hasWidth ? Dimension.WIDTH : Dimension.HEIGHT;

};

\_proto.\_getParent = function \_getParent() {

var \_this3 = this;

var parent = null;

if (Util.isElement(this.\_config.parent)) {

parent = this.\_config.parent; // It's a jQuery object

if (typeof this.\_config.parent.jquery !== 'undefined') {

parent = this.\_config.parent[0];

}

} else {

parent = $$$1(this.\_config.parent)[0];

}

var selector = "[data-toggle=\"collapse\"][data-parent=\"" + this.\_config.parent + "\"]";

$$$1(parent).find(selector).each(function (i, element) {

\_this3.\_addAriaAndCollapsedClass(Collapse.\_getTargetFromElement(element), [element]);

});

return parent;

};

\_proto.\_addAriaAndCollapsedClass = function \_addAriaAndCollapsedClass(element, triggerArray) {

if (element) {

var isOpen = $$$1(element).hasClass(ClassName.SHOW);

if (triggerArray.length > 0) {

$$$1(triggerArray).toggleClass(ClassName.COLLAPSED, !isOpen).attr('aria-expanded', isOpen);

}

}

}; // Static

Collapse.\_getTargetFromElement = function \_getTargetFromElement(element) {

var selector = Util.getSelectorFromElement(element);

return selector ? $$$1(selector)[0] : null;

};

Collapse.\_jQueryInterface = function \_jQueryInterface(config) {

return this.each(function () {

var $this = $$$1(this);

var data = $this.data(DATA\_KEY);

var \_config = \_extends({}, Default, $this.data(), typeof config === 'object' && config);

if (!data && \_config.toggle && /show|hide/.test(config)) {

\_config.toggle = false;

}

if (!data) {

data = new Collapse(this, \_config);

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

}

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

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

throw new TypeError("No method named \"" + config + "\"");

}

data[config]();

}

});

};

\_createClass(Collapse, null, [{

key: "VERSION",

get: function get() {

return VERSION;

}

}, {

key: "Default",

get: function get() {

return Default;

}

}]);

return Collapse;

}();

/\*\*

\* ------------------------------------------------------------------------

\* Data Api implementation

\* ------------------------------------------------------------------------

\*/

$$$1(document).on(Event.CLICK\_DATA\_API, Selector.DATA\_TOGGLE, function (event) {

// preventDefault only for <a> elements (which change the URL) not inside the collapsible element

if (event.currentTarget.tagName === 'A') {

event.preventDefault();

}

var $trigger = $$$1(this);

var selector = Util.getSelectorFromElement(this);

$$$1(selector).each(function () {

var $target = $$$1(this);

var data = $target.data(DATA\_KEY);

var config = data ? 'toggle' : $trigger.data();

Collapse.\_jQueryInterface.call($target, config);

});

});

/\*\*

\* ------------------------------------------------------------------------

\* jQuery

\* ------------------------------------------------------------------------

\*/

$$$1.fn[NAME] = Collapse.\_jQueryInterface;

$$$1.fn[NAME].Constructor = Collapse;

$$$1.fn[NAME].noConflict = function () {

$$$1.fn[NAME] = JQUERY\_NO\_CONFLICT;

return Collapse.\_jQueryInterface;

};

return Collapse;

}($);

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0): dropdown.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

var Dropdown = function ($$$1) {

/\*\*

\* ------------------------------------------------------------------------

\* Constants

\* ------------------------------------------------------------------------

\*/

var NAME = 'dropdown';

var VERSION = '4.0.0';

var DATA\_KEY = 'bs.dropdown';

var EVENT\_KEY = "." + DATA\_KEY;

var DATA\_API\_KEY = '.data-api';

var JQUERY\_NO\_CONFLICT = $$$1.fn[NAME];

var ESCAPE\_KEYCODE = 27; // KeyboardEvent.which value for Escape (Esc) key

var SPACE\_KEYCODE = 32; // KeyboardEvent.which value for space key

var TAB\_KEYCODE = 9; // KeyboardEvent.which value for tab key

var ARROW\_UP\_KEYCODE = 38; // KeyboardEvent.which value for up arrow key

var ARROW\_DOWN\_KEYCODE = 40; // KeyboardEvent.which value for down arrow key

var RIGHT\_MOUSE\_BUTTON\_WHICH = 3; // MouseEvent.which value for the right button (assuming a right-handed mouse)

var REGEXP\_KEYDOWN = new RegExp(ARROW\_UP\_KEYCODE + "|" + ARROW\_DOWN\_KEYCODE + "|" + ESCAPE\_KEYCODE);

var Event = {

HIDE: "hide" + EVENT\_KEY,

HIDDEN: "hidden" + EVENT\_KEY,

SHOW: "show" + EVENT\_KEY,

SHOWN: "shown" + EVENT\_KEY,

CLICK: "click" + EVENT\_KEY,

CLICK\_DATA\_API: "click" + EVENT\_KEY + DATA\_API\_KEY,

KEYDOWN\_DATA\_API: "keydown" + EVENT\_KEY + DATA\_API\_KEY,

KEYUP\_DATA\_API: "keyup" + EVENT\_KEY + DATA\_API\_KEY

};

var ClassName = {

DISABLED: 'disabled',

SHOW: 'show',

DROPUP: 'dropup',

DROPRIGHT: 'dropright',

DROPLEFT: 'dropleft',

MENURIGHT: 'dropdown-menu-right',

MENULEFT: 'dropdown-menu-left',

POSITION\_STATIC: 'position-static'

};

var Selector = {

DATA\_TOGGLE: '[data-toggle="dropdown"]',

FORM\_CHILD: '.dropdown form',

MENU: '.dropdown-menu',

NAVBAR\_NAV: '.navbar-nav',

VISIBLE\_ITEMS: '.dropdown-menu .dropdown-item:not(.disabled)'

};

var AttachmentMap = {

TOP: 'top-start',

TOPEND: 'top-end',

BOTTOM: 'bottom-start',

BOTTOMEND: 'bottom-end',

RIGHT: 'right-start',

RIGHTEND: 'right-end',

LEFT: 'left-start',

LEFTEND: 'left-end'

};

var Default = {

offset: 0,

flip: true,

boundary: 'scrollParent'

};

var DefaultType = {

offset: '(number|string|function)',

flip: 'boolean',

boundary: '(string|element)'

/\*\*

\* ------------------------------------------------------------------------

\* Class Definition

\* ------------------------------------------------------------------------

\*/

};

var Dropdown =

/\*#\_\_PURE\_\_\*/

function () {

function Dropdown(element, config) {

this.\_element = element;

this.\_popper = null;

this.\_config = this.\_getConfig(config);

this.\_menu = this.\_getMenuElement();

this.\_inNavbar = this.\_detectNavbar();

this.\_addEventListeners();

} // Getters

var \_proto = Dropdown.prototype;

// Public

\_proto.toggle = function toggle() {

if (this.\_element.disabled || $$$1(this.\_element).hasClass(ClassName.DISABLED)) {

return;

}

var parent = Dropdown.\_getParentFromElement(this.\_element);

var isActive = $$$1(this.\_menu).hasClass(ClassName.SHOW);

Dropdown.\_clearMenus();

if (isActive) {

return;

}

var relatedTarget = {

relatedTarget: this.\_element

};

var showEvent = $$$1.Event(Event.SHOW, relatedTarget);

$$$1(parent).trigger(showEvent);

if (showEvent.isDefaultPrevented()) {

return;

} // Disable totally Popper.js for Dropdown in Navbar

if (!this.\_inNavbar) {

/\*\*

\* Check for Popper dependency

\* Popper - https://popper.js.org

\*/

if (typeof Popper === 'undefined') {

throw new TypeError('Bootstrap dropdown require Popper.js (https://popper.js.org)');

}

var element = this.\_element; // For dropup with alignment we use the parent as popper container

if ($$$1(parent).hasClass(ClassName.DROPUP)) {

if ($$$1(this.\_menu).hasClass(ClassName.MENULEFT) || $$$1(this.\_menu).hasClass(ClassName.MENURIGHT)) {

element = parent;

}

} // If boundary is not `scrollParent`, then set position to `static`

// to allow the menu to "escape" the scroll parent's boundaries

// https://github.com/twbs/bootstrap/issues/24251

if (this.\_config.boundary !== 'scrollParent') {

$$$1(parent).addClass(ClassName.POSITION\_STATIC);

}

this.\_popper = new Popper(element, this.\_menu, this.\_getPopperConfig());

} // If this is a touch-enabled device we add extra

// empty mouseover listeners to the body's immediate children;

// only needed because of broken event delegation on iOS

// https://www.quirksmode.org/blog/archives/2014/02/mouse\_event\_bub.html

if ('ontouchstart' in document.documentElement && $$$1(parent).closest(Selector.NAVBAR\_NAV).length === 0) {

$$$1('body').children().on('mouseover', null, $$$1.noop);

}

this.\_element.focus();

this.\_element.setAttribute('aria-expanded', true);

$$$1(this.\_menu).toggleClass(ClassName.SHOW);

$$$1(parent).toggleClass(ClassName.SHOW).trigger($$$1.Event(Event.SHOWN, relatedTarget));

};

\_proto.dispose = function dispose() {

$$$1.removeData(this.\_element, DATA\_KEY);

$$$1(this.\_element).off(EVENT\_KEY);

this.\_element = null;

this.\_menu = null;

if (this.\_popper !== null) {

this.\_popper.destroy();

this.\_popper = null;

}

};

\_proto.update = function update() {

this.\_inNavbar = this.\_detectNavbar();

if (this.\_popper !== null) {

this.\_popper.scheduleUpdate();

}

}; // Private

\_proto.\_addEventListeners = function \_addEventListeners() {

var \_this = this;

$$$1(this.\_element).on(Event.CLICK, function (event) {

event.preventDefault();

event.stopPropagation();

\_this.toggle();

});

};

\_proto.\_getConfig = function \_getConfig(config) {

config = \_extends({}, this.constructor.Default, $$$1(this.\_element).data(), config);

Util.typeCheckConfig(NAME, config, this.constructor.DefaultType);

return config;

};

\_proto.\_getMenuElement = function \_getMenuElement() {

if (!this.\_menu) {

var parent = Dropdown.\_getParentFromElement(this.\_element);

this.\_menu = $$$1(parent).find(Selector.MENU)[0];

}

return this.\_menu;

};

\_proto.\_getPlacement = function \_getPlacement() {

var $parentDropdown = $$$1(this.\_element).parent();

var placement = AttachmentMap.BOTTOM; // Handle dropup

if ($parentDropdown.hasClass(ClassName.DROPUP)) {

placement = AttachmentMap.TOP;

if ($$$1(this.\_menu).hasClass(ClassName.MENURIGHT)) {

placement = AttachmentMap.TOPEND;

}

} else if ($parentDropdown.hasClass(ClassName.DROPRIGHT)) {

placement = AttachmentMap.RIGHT;

} else if ($parentDropdown.hasClass(ClassName.DROPLEFT)) {

placement = AttachmentMap.LEFT;

} else if ($$$1(this.\_menu).hasClass(ClassName.MENURIGHT)) {

placement = AttachmentMap.BOTTOMEND;

}

return placement;

};

\_proto.\_detectNavbar = function \_detectNavbar() {

return $$$1(this.\_element).closest('.navbar').length > 0;

};

\_proto.\_getPopperConfig = function \_getPopperConfig() {

var \_this2 = this;

var offsetConf = {};

if (typeof this.\_config.offset === 'function') {

offsetConf.fn = function (data) {

data.offsets = \_extends({}, data.offsets, \_this2.\_config.offset(data.offsets) || {});

return data;

};

} else {

offsetConf.offset = this.\_config.offset;

}

var popperConfig = {

placement: this.\_getPlacement(),

modifiers: {

offset: offsetConf,

flip: {

enabled: this.\_config.flip

},

preventOverflow: {

boundariesElement: this.\_config.boundary

}

}

};

return popperConfig;

}; // Static

Dropdown.\_jQueryInterface = function \_jQueryInterface(config) {

return this.each(function () {

var data = $$$1(this).data(DATA\_KEY);

var \_config = typeof config === 'object' ? config : null;

if (!data) {

data = new Dropdown(this, \_config);

$$$1(this).data(DATA\_KEY, data);

}

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

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

throw new TypeError("No method named \"" + config + "\"");

}

data[config]();

}

});

};

Dropdown.\_clearMenus = function \_clearMenus(event) {

if (event && (event.which === RIGHT\_MOUSE\_BUTTON\_WHICH || event.type === 'keyup' && event.which !== TAB\_KEYCODE)) {

return;

}

var toggles = $$$1.makeArray($$$1(Selector.DATA\_TOGGLE));

for (var i = 0; i < toggles.length; i++) {

var parent = Dropdown.\_getParentFromElement(toggles[i]);

var context = $$$1(toggles[i]).data(DATA\_KEY);

var relatedTarget = {

relatedTarget: toggles[i]

};

if (!context) {

continue;

}

var dropdownMenu = context.\_menu;

if (!$$$1(parent).hasClass(ClassName.SHOW)) {

continue;

}

if (event && (event.type === 'click' && /input|textarea/i.test(event.target.tagName) || event.type === 'keyup' && event.which === TAB\_KEYCODE) && $$$1.contains(parent, event.target)) {

continue;

}

var hideEvent = $$$1.Event(Event.HIDE, relatedTarget);

$$$1(parent).trigger(hideEvent);

if (hideEvent.isDefaultPrevented()) {

continue;

} // If this is a touch-enabled device we remove the extra

// empty mouseover listeners we added for iOS support

if ('ontouchstart' in document.documentElement) {

$$$1('body').children().off('mouseover', null, $$$1.noop);

}

toggles[i].setAttribute('aria-expanded', 'false');

$$$1(dropdownMenu).removeClass(ClassName.SHOW);

$$$1(parent).removeClass(ClassName.SHOW).trigger($$$1.Event(Event.HIDDEN, relatedTarget));

}

};

Dropdown.\_getParentFromElement = function \_getParentFromElement(element) {

var parent;

var selector = Util.getSelectorFromElement(element);

if (selector) {

parent = $$$1(selector)[0];

}

return parent || element.parentNode;

}; // eslint-disable-next-line complexity

Dropdown.\_dataApiKeydownHandler = function \_dataApiKeydownHandler(event) {

// If not input/textarea:

// - And not a key in REGEXP\_KEYDOWN => not a dropdown command

// If input/textarea:

// - If space key => not a dropdown command

// - If key is other than escape

// - If key is not up or down => not a dropdown command

// - If trigger inside the menu => not a dropdown command

if (/input|textarea/i.test(event.target.tagName) ? event.which === SPACE\_KEYCODE || event.which !== ESCAPE\_KEYCODE && (event.which !== ARROW\_DOWN\_KEYCODE && event.which !== ARROW\_UP\_KEYCODE || $$$1(event.target).closest(Selector.MENU).length) : !REGEXP\_KEYDOWN.test(event.which)) {

return;

}

event.preventDefault();

event.stopPropagation();

if (this.disabled || $$$1(this).hasClass(ClassName.DISABLED)) {

return;

}

var parent = Dropdown.\_getParentFromElement(this);

var isActive = $$$1(parent).hasClass(ClassName.SHOW);

if (!isActive && (event.which !== ESCAPE\_KEYCODE || event.which !== SPACE\_KEYCODE) || isActive && (event.which === ESCAPE\_KEYCODE || event.which === SPACE\_KEYCODE)) {

if (event.which === ESCAPE\_KEYCODE) {

var toggle = $$$1(parent).find(Selector.DATA\_TOGGLE)[0];

$$$1(toggle).trigger('focus');

}

$$$1(this).trigger('click');

return;

}

var items = $$$1(parent).find(Selector.VISIBLE\_ITEMS).get();

if (items.length === 0) {

return;

}

var index = items.indexOf(event.target);

if (event.which === ARROW\_UP\_KEYCODE && index > 0) {

// Up

index--;

}

if (event.which === ARROW\_DOWN\_KEYCODE && index < items.length - 1) {

// Down

index++;

}

if (index < 0) {

index = 0;

}

items[index].focus();

};

\_createClass(Dropdown, null, [{

key: "VERSION",

get: function get() {

return VERSION;

}

}, {

key: "Default",

get: function get() {

return Default;

}

}, {

key: "DefaultType",

get: function get() {

return DefaultType;

}

}]);

return Dropdown;

}();

/\*\*

\* ------------------------------------------------------------------------

\* Data Api implementation

\* ------------------------------------------------------------------------

\*/

$$$1(document).on(Event.KEYDOWN\_DATA\_API, Selector.DATA\_TOGGLE, Dropdown.\_dataApiKeydownHandler).on(Event.KEYDOWN\_DATA\_API, Selector.MENU, Dropdown.\_dataApiKeydownHandler).on(Event.CLICK\_DATA\_API + " " + Event.KEYUP\_DATA\_API, Dropdown.\_clearMenus).on(Event.CLICK\_DATA\_API, Selector.DATA\_TOGGLE, function (event) {

event.preventDefault();

event.stopPropagation();

Dropdown.\_jQueryInterface.call($$$1(this), 'toggle');

}).on(Event.CLICK\_DATA\_API, Selector.FORM\_CHILD, function (e) {

e.stopPropagation();

});

/\*\*

\* ------------------------------------------------------------------------

\* jQuery

\* ------------------------------------------------------------------------

\*/

$$$1.fn[NAME] = Dropdown.\_jQueryInterface;

$$$1.fn[NAME].Constructor = Dropdown;

$$$1.fn[NAME].noConflict = function () {

$$$1.fn[NAME] = JQUERY\_NO\_CONFLICT;

return Dropdown.\_jQueryInterface;

};

return Dropdown;

}($, Popper);

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0): modal.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

var Modal = function ($$$1) {

/\*\*

\* ------------------------------------------------------------------------

\* Constants

\* ------------------------------------------------------------------------

\*/

var NAME = 'modal';

var VERSION = '4.0.0';

var DATA\_KEY = 'bs.modal';

var EVENT\_KEY = "." + DATA\_KEY;

var DATA\_API\_KEY = '.data-api';

var JQUERY\_NO\_CONFLICT = $$$1.fn[NAME];

var TRANSITION\_DURATION = 300;

var BACKDROP\_TRANSITION\_DURATION = 150;

var ESCAPE\_KEYCODE = 27; // KeyboardEvent.which value for Escape (Esc) key

var Default = {

backdrop: true,

keyboard: true,

focus: true,

show: true

};

var DefaultType = {

backdrop: '(boolean|string)',

keyboard: 'boolean',

focus: 'boolean',

show: 'boolean'

};

var Event = {

HIDE: "hide" + EVENT\_KEY,

HIDDEN: "hidden" + EVENT\_KEY,

SHOW: "show" + EVENT\_KEY,

SHOWN: "shown" + EVENT\_KEY,

FOCUSIN: "focusin" + EVENT\_KEY,

RESIZE: "resize" + EVENT\_KEY,

CLICK\_DISMISS: "click.dismiss" + EVENT\_KEY,

KEYDOWN\_DISMISS: "keydown.dismiss" + EVENT\_KEY,

MOUSEUP\_DISMISS: "mouseup.dismiss" + EVENT\_KEY,

MOUSEDOWN\_DISMISS: "mousedown.dismiss" + EVENT\_KEY,

CLICK\_DATA\_API: "click" + EVENT\_KEY + DATA\_API\_KEY

};

var ClassName = {

SCROLLBAR\_MEASURER: 'modal-scrollbar-measure',

BACKDROP: 'modal-backdrop',

OPEN: 'modal-open',

FADE: 'fade',

SHOW: 'show'

};

var Selector = {

DIALOG: '.modal-dialog',

DATA\_TOGGLE: '[data-toggle="modal"]',

DATA\_DISMISS: '[data-dismiss="modal"]',

FIXED\_CONTENT: '.fixed-top, .fixed-bottom, .is-fixed, .sticky-top',

STICKY\_CONTENT: '.sticky-top',

NAVBAR\_TOGGLER: '.navbar-toggler'

/\*\*

\* ------------------------------------------------------------------------

\* Class Definition

\* ------------------------------------------------------------------------

\*/

};

var Modal =

/\*#\_\_PURE\_\_\*/

function () {

function Modal(element, config) {

this.\_config = this.\_getConfig(config);

this.\_element = element;

this.\_dialog = $$$1(element).find(Selector.DIALOG)[0];

this.\_backdrop = null;

this.\_isShown = false;

this.\_isBodyOverflowing = false;

this.\_ignoreBackdropClick = false;

this.\_originalBodyPadding = 0;

this.\_scrollbarWidth = 0;

} // Getters

var \_proto = Modal.prototype;

// Public

\_proto.toggle = function toggle(relatedTarget) {

return this.\_isShown ? this.hide() : this.show(relatedTarget);

};

\_proto.show = function show(relatedTarget) {

var \_this = this;

if (this.\_isTransitioning || this.\_isShown) {

return;

}

if (Util.supportsTransitionEnd() && $$$1(this.\_element).hasClass(ClassName.FADE)) {

this.\_isTransitioning = true;

}

var showEvent = $$$1.Event(Event.SHOW, {

relatedTarget: relatedTarget

});

$$$1(this.\_element).trigger(showEvent);

if (this.\_isShown || showEvent.isDefaultPrevented()) {

return;

}

this.\_isShown = true;

this.\_checkScrollbar();

this.\_setScrollbar();

this.\_adjustDialog();

$$$1(document.body).addClass(ClassName.OPEN);

this.\_setEscapeEvent();

this.\_setResizeEvent();

$$$1(this.\_element).on(Event.CLICK\_DISMISS, Selector.DATA\_DISMISS, function (event) {

return \_this.hide(event);

});

$$$1(this.\_dialog).on(Event.MOUSEDOWN\_DISMISS, function () {

$$$1(\_this.\_element).one(Event.MOUSEUP\_DISMISS, function (event) {

if ($$$1(event.target).is(\_this.\_element)) {

\_this.\_ignoreBackdropClick = true;

}

});

});

this.\_showBackdrop(function () {

return \_this.\_showElement(relatedTarget);

});

};

\_proto.hide = function hide(event) {

var \_this2 = this;

if (event) {

event.preventDefault();

}

if (this.\_isTransitioning || !this.\_isShown) {

return;

}

var hideEvent = $$$1.Event(Event.HIDE);

$$$1(this.\_element).trigger(hideEvent);

if (!this.\_isShown || hideEvent.isDefaultPrevented()) {

return;

}

this.\_isShown = false;

var transition = Util.supportsTransitionEnd() && $$$1(this.\_element).hasClass(ClassName.FADE);

if (transition) {

this.\_isTransitioning = true;

}

this.\_setEscapeEvent();

this.\_setResizeEvent();

$$$1(document).off(Event.FOCUSIN);

$$$1(this.\_element).removeClass(ClassName.SHOW);

$$$1(this.\_element).off(Event.CLICK\_DISMISS);

$$$1(this.\_dialog).off(Event.MOUSEDOWN\_DISMISS);

if (transition) {

$$$1(this.\_element).one(Util.TRANSITION\_END, function (event) {

return \_this2.\_hideModal(event);

}).emulateTransitionEnd(TRANSITION\_DURATION);

} else {

this.\_hideModal();

}

};

\_proto.dispose = function dispose() {

$$$1.removeData(this.\_element, DATA\_KEY);

$$$1(window, document, this.\_element, this.\_backdrop).off(EVENT\_KEY);

this.\_config = null;

this.\_element = null;

this.\_dialog = null;

this.\_backdrop = null;

this.\_isShown = null;

this.\_isBodyOverflowing = null;

this.\_ignoreBackdropClick = null;

this.\_scrollbarWidth = null;

};

\_proto.handleUpdate = function handleUpdate() {

this.\_adjustDialog();

}; // Private

\_proto.\_getConfig = function \_getConfig(config) {

config = \_extends({}, Default, config);

Util.typeCheckConfig(NAME, config, DefaultType);

return config;

};

\_proto.\_showElement = function \_showElement(relatedTarget) {

var \_this3 = this;

var transition = Util.supportsTransitionEnd() && $$$1(this.\_element).hasClass(ClassName.FADE);

if (!this.\_element.parentNode || this.\_element.parentNode.nodeType !== Node.ELEMENT\_NODE) {

// Don't move modal's DOM position

document.body.appendChild(this.\_element);

}

this.\_element.style.display = 'block';

this.\_element.removeAttribute('aria-hidden');

this.\_element.scrollTop = 0;

if (transition) {

Util.reflow(this.\_element);

}

$$$1(this.\_element).addClass(ClassName.SHOW);

if (this.\_config.focus) {

this.\_enforceFocus();

}

var shownEvent = $$$1.Event(Event.SHOWN, {

relatedTarget: relatedTarget

});

var transitionComplete = function transitionComplete() {

if (\_this3.\_config.focus) {

\_this3.\_element.focus();

}

\_this3.\_isTransitioning = false;

$$$1(\_this3.\_element).trigger(shownEvent);

};

if (transition) {

$$$1(this.\_dialog).one(Util.TRANSITION\_END, transitionComplete).emulateTransitionEnd(TRANSITION\_DURATION);

} else {

transitionComplete();

}

};

\_proto.\_enforceFocus = function \_enforceFocus() {

var \_this4 = this;

$$$1(document).off(Event.FOCUSIN) // Guard against infinite focus loop

.on(Event.FOCUSIN, function (event) {

if (document !== event.target && \_this4.\_element !== event.target && $$$1(\_this4.\_element).has(event.target).length === 0) {

\_this4.\_element.focus();

}

});

};

\_proto.\_setEscapeEvent = function \_setEscapeEvent() {

var \_this5 = this;

if (this.\_isShown && this.\_config.keyboard) {

$$$1(this.\_element).on(Event.KEYDOWN\_DISMISS, function (event) {

if (event.which === ESCAPE\_KEYCODE) {

event.preventDefault();

\_this5.hide();

}

});

} else if (!this.\_isShown) {

$$$1(this.\_element).off(Event.KEYDOWN\_DISMISS);

}

};

\_proto.\_setResizeEvent = function \_setResizeEvent() {

var \_this6 = this;

if (this.\_isShown) {

$$$1(window).on(Event.RESIZE, function (event) {

return \_this6.handleUpdate(event);

});

} else {

$$$1(window).off(Event.RESIZE);

}

};

\_proto.\_hideModal = function \_hideModal() {

var \_this7 = this;

this.\_element.style.display = 'none';

this.\_element.setAttribute('aria-hidden', true);

this.\_isTransitioning = false;

this.\_showBackdrop(function () {

$$$1(document.body).removeClass(ClassName.OPEN);

\_this7.\_resetAdjustments();

\_this7.\_resetScrollbar();

$$$1(\_this7.\_element).trigger(Event.HIDDEN);

});

};

\_proto.\_removeBackdrop = function \_removeBackdrop() {

if (this.\_backdrop) {

$$$1(this.\_backdrop).remove();

this.\_backdrop = null;

}

};

\_proto.\_showBackdrop = function \_showBackdrop(callback) {

var \_this8 = this;

var animate = $$$1(this.\_element).hasClass(ClassName.FADE) ? ClassName.FADE : '';

if (this.\_isShown && this.\_config.backdrop) {

var doAnimate = Util.supportsTransitionEnd() && animate;

this.\_backdrop = document.createElement('div');

this.\_backdrop.className = ClassName.BACKDROP;

if (animate) {

$$$1(this.\_backdrop).addClass(animate);

}

$$$1(this.\_backdrop).appendTo(document.body);

$$$1(this.\_element).on(Event.CLICK\_DISMISS, function (event) {

if (\_this8.\_ignoreBackdropClick) {

\_this8.\_ignoreBackdropClick = false;

return;

}

if (event.target !== event.currentTarget) {

return;

}

if (\_this8.\_config.backdrop === 'static') {

\_this8.\_element.focus();

} else {

\_this8.hide();

}

});

if (doAnimate) {

Util.reflow(this.\_backdrop);

}

$$$1(this.\_backdrop).addClass(ClassName.SHOW);

if (!callback) {

return;

}

if (!doAnimate) {

callback();

return;

}

$$$1(this.\_backdrop).one(Util.TRANSITION\_END, callback).emulateTransitionEnd(BACKDROP\_TRANSITION\_DURATION);

} else if (!this.\_isShown && this.\_backdrop) {

$$$1(this.\_backdrop).removeClass(ClassName.SHOW);

var callbackRemove = function callbackRemove() {

\_this8.\_removeBackdrop();

if (callback) {

callback();

}

};

if (Util.supportsTransitionEnd() && $$$1(this.\_element).hasClass(ClassName.FADE)) {

$$$1(this.\_backdrop).one(Util.TRANSITION\_END, callbackRemove).emulateTransitionEnd(BACKDROP\_TRANSITION\_DURATION);

} else {

callbackRemove();

}

} else if (callback) {

callback();

}

}; // ----------------------------------------------------------------------

// the following methods are used to handle overflowing modals

// todo (fat): these should probably be refactored out of modal.js

// ----------------------------------------------------------------------

\_proto.\_adjustDialog = function \_adjustDialog() {

var isModalOverflowing = this.\_element.scrollHeight > document.documentElement.clientHeight;

if (!this.\_isBodyOverflowing && isModalOverflowing) {

this.\_element.style.paddingLeft = this.\_scrollbarWidth + "px";

}

if (this.\_isBodyOverflowing && !isModalOverflowing) {

this.\_element.style.paddingRight = this.\_scrollbarWidth + "px";

}

};

\_proto.\_resetAdjustments = function \_resetAdjustments() {

this.\_element.style.paddingLeft = '';

this.\_element.style.paddingRight = '';

};

\_proto.\_checkScrollbar = function \_checkScrollbar() {

var rect = document.body.getBoundingClientRect();

this.\_isBodyOverflowing = rect.left + rect.right < window.innerWidth;

this.\_scrollbarWidth = this.\_getScrollbarWidth();

};

\_proto.\_setScrollbar = function \_setScrollbar() {

var \_this9 = this;

if (this.\_isBodyOverflowing) {

// Note: DOMNode.style.paddingRight returns the actual value or '' if not set

// while $(DOMNode).css('padding-right') returns the calculated value or 0 if not set

// Adjust fixed content padding

$$$1(Selector.FIXED\_CONTENT).each(function (index, element) {

var actualPadding = $$$1(element)[0].style.paddingRight;

var calculatedPadding = $$$1(element).css('padding-right');

$$$1(element).data('padding-right', actualPadding).css('padding-right', parseFloat(calculatedPadding) + \_this9.\_scrollbarWidth + "px");

}); // Adjust sticky content margin

$$$1(Selector.STICKY\_CONTENT).each(function (index, element) {

var actualMargin = $$$1(element)[0].style.marginRight;

var calculatedMargin = $$$1(element).css('margin-right');

$$$1(element).data('margin-right', actualMargin).css('margin-right', parseFloat(calculatedMargin) - \_this9.\_scrollbarWidth + "px");

}); // Adjust navbar-toggler margin

$$$1(Selector.NAVBAR\_TOGGLER).each(function (index, element) {

var actualMargin = $$$1(element)[0].style.marginRight;

var calculatedMargin = $$$1(element).css('margin-right');

$$$1(element).data('margin-right', actualMargin).css('margin-right', parseFloat(calculatedMargin) + \_this9.\_scrollbarWidth + "px");

}); // Adjust body padding

var actualPadding = document.body.style.paddingRight;

var calculatedPadding = $$$1('body').css('padding-right');

$$$1('body').data('padding-right', actualPadding).css('padding-right', parseFloat(calculatedPadding) + this.\_scrollbarWidth + "px");

}

};

\_proto.\_resetScrollbar = function \_resetScrollbar() {

// Restore fixed content padding

$$$1(Selector.FIXED\_CONTENT).each(function (index, element) {

var padding = $$$1(element).data('padding-right');

if (typeof padding !== 'undefined') {

$$$1(element).css('padding-right', padding).removeData('padding-right');

}

}); // Restore sticky content and navbar-toggler margin

$$$1(Selector.STICKY\_CONTENT + ", " + Selector.NAVBAR\_TOGGLER).each(function (index, element) {

var margin = $$$1(element).data('margin-right');

if (typeof margin !== 'undefined') {

$$$1(element).css('margin-right', margin).removeData('margin-right');

}

}); // Restore body padding

var padding = $$$1('body').data('padding-right');

if (typeof padding !== 'undefined') {

$$$1('body').css('padding-right', padding).removeData('padding-right');

}

};

\_proto.\_getScrollbarWidth = function \_getScrollbarWidth() {

// thx d.walsh

var scrollDiv = document.createElement('div');

scrollDiv.className = ClassName.SCROLLBAR\_MEASURER;

document.body.appendChild(scrollDiv);

var scrollbarWidth = scrollDiv.getBoundingClientRect().width - scrollDiv.clientWidth;

document.body.removeChild(scrollDiv);

return scrollbarWidth;

}; // Static

Modal.\_jQueryInterface = function \_jQueryInterface(config, relatedTarget) {

return this.each(function () {

var data = $$$1(this).data(DATA\_KEY);

var \_config = \_extends({}, Modal.Default, $$$1(this).data(), typeof config === 'object' && config);

if (!data) {

data = new Modal(this, \_config);

$$$1(this).data(DATA\_KEY, data);

}

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

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

throw new TypeError("No method named \"" + config + "\"");

}

data[config](relatedTarget);

} else if (\_config.show) {

data.show(relatedTarget);

}

});

};

\_createClass(Modal, null, [{

key: "VERSION",

get: function get() {

return VERSION;

}

}, {

key: "Default",

get: function get() {

return Default;

}

}]);

return Modal;

}();

/\*\*

\* ------------------------------------------------------------------------

\* Data Api implementation

\* ------------------------------------------------------------------------

\*/

$$$1(document).on(Event.CLICK\_DATA\_API, Selector.DATA\_TOGGLE, function (event) {

var \_this10 = this;

var target;

var selector = Util.getSelectorFromElement(this);

if (selector) {

target = $$$1(selector)[0];

}

var config = $$$1(target).data(DATA\_KEY) ? 'toggle' : \_extends({}, $$$1(target).data(), $$$1(this).data());

if (this.tagName === 'A' || this.tagName === 'AREA') {

event.preventDefault();

}

var $target = $$$1(target).one(Event.SHOW, function (showEvent) {

if (showEvent.isDefaultPrevented()) {

// Only register focus restorer if modal will actually get shown

return;

}

$target.one(Event.HIDDEN, function () {

if ($$$1(\_this10).is(':visible')) {

\_this10.focus();

}

});

});

Modal.\_jQueryInterface.call($$$1(target), config, this);

});

/\*\*

\* ------------------------------------------------------------------------

\* jQuery

\* ------------------------------------------------------------------------

\*/

$$$1.fn[NAME] = Modal.\_jQueryInterface;

$$$1.fn[NAME].Constructor = Modal;

$$$1.fn[NAME].noConflict = function () {

$$$1.fn[NAME] = JQUERY\_NO\_CONFLICT;

return Modal.\_jQueryInterface;

};

return Modal;

}($);

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0): tooltip.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

var Tooltip = function ($$$1) {

/\*\*

\* ------------------------------------------------------------------------

\* Constants

\* ------------------------------------------------------------------------

\*/

var NAME = 'tooltip';

var VERSION = '4.0.0';

var DATA\_KEY = 'bs.tooltip';

var EVENT\_KEY = "." + DATA\_KEY;

var JQUERY\_NO\_CONFLICT = $$$1.fn[NAME];

var TRANSITION\_DURATION = 150;

var CLASS\_PREFIX = 'bs-tooltip';

var BSCLS\_PREFIX\_REGEX = new RegExp("(^|\\s)" + CLASS\_PREFIX + "\\S+", 'g');

var DefaultType = {

animation: 'boolean',

template: 'string',

title: '(string|element|function)',

trigger: 'string',

delay: '(number|object)',

html: 'boolean',

selector: '(string|boolean)',

placement: '(string|function)',

offset: '(number|string)',

container: '(string|element|boolean)',

fallbackPlacement: '(string|array)',

boundary: '(string|element)'

};

var AttachmentMap = {

AUTO: 'auto',

TOP: 'top',

RIGHT: 'right',

BOTTOM: 'bottom',

LEFT: 'left'

};

var Default = {

animation: true,

template: '<div class="tooltip" role="tooltip">' + '<div class="arrow"></div>' + '<div class="tooltip-inner"></div></div>',

trigger: 'hover focus',

title: '',

delay: 0,

html: false,

selector: false,

placement: 'top',

offset: 0,

container: false,

fallbackPlacement: 'flip',

boundary: 'scrollParent'

};

var HoverState = {

SHOW: 'show',

OUT: 'out'

};

var Event = {

HIDE: "hide" + EVENT\_KEY,

HIDDEN: "hidden" + EVENT\_KEY,

SHOW: "show" + EVENT\_KEY,

SHOWN: "shown" + EVENT\_KEY,

INSERTED: "inserted" + EVENT\_KEY,

CLICK: "click" + EVENT\_KEY,

FOCUSIN: "focusin" + EVENT\_KEY,

FOCUSOUT: "focusout" + EVENT\_KEY,

MOUSEENTER: "mouseenter" + EVENT\_KEY,

MOUSELEAVE: "mouseleave" + EVENT\_KEY

};

var ClassName = {

FADE: 'fade',

SHOW: 'show'

};

var Selector = {

TOOLTIP: '.tooltip',

TOOLTIP\_INNER: '.tooltip-inner',

ARROW: '.arrow'

};

var Trigger = {

HOVER: 'hover',

FOCUS: 'focus',

CLICK: 'click',

MANUAL: 'manual'

/\*\*

\* ------------------------------------------------------------------------

\* Class Definition

\* ------------------------------------------------------------------------

\*/

};

var Tooltip =

/\*#\_\_PURE\_\_\*/

function () {

function Tooltip(element, config) {

/\*\*

\* Check for Popper dependency

\* Popper - https://popper.js.org

\*/

if (typeof Popper === 'undefined') {

throw new TypeError('Bootstrap tooltips require Popper.js (https://popper.js.org)');

} // private

this.\_isEnabled = true;

this.\_timeout = 0;

this.\_hoverState = '';

this.\_activeTrigger = {};

this.\_popper = null; // Protected

this.element = element;

this.config = this.\_getConfig(config);

this.tip = null;

this.\_setListeners();

} // Getters

var \_proto = Tooltip.prototype;

// Public

\_proto.enable = function enable() {

this.\_isEnabled = true;

};

\_proto.disable = function disable() {

this.\_isEnabled = false;

};

\_proto.toggleEnabled = function toggleEnabled() {

this.\_isEnabled = !this.\_isEnabled;

};

\_proto.toggle = function toggle(event) {

if (!this.\_isEnabled) {

return;

}

if (event) {

var dataKey = this.constructor.DATA\_KEY;

var context = $$$1(event.currentTarget).data(dataKey);

if (!context) {

context = new this.constructor(event.currentTarget, this.\_getDelegateConfig());

$$$1(event.currentTarget).data(dataKey, context);

}

context.\_activeTrigger.click = !context.\_activeTrigger.click;

if (context.\_isWithActiveTrigger()) {

context.\_enter(null, context);

} else {

context.\_leave(null, context);

}

} else {

if ($$$1(this.getTipElement()).hasClass(ClassName.SHOW)) {

this.\_leave(null, this);

return;

}

this.\_enter(null, this);

}

};

\_proto.dispose = function dispose() {

clearTimeout(this.\_timeout);

$$$1.removeData(this.element, this.constructor.DATA\_KEY);

$$$1(this.element).off(this.constructor.EVENT\_KEY);

$$$1(this.element).closest('.modal').off('hide.bs.modal');

if (this.tip) {

$$$1(this.tip).remove();

}

this.\_isEnabled = null;

this.\_timeout = null;

this.\_hoverState = null;

this.\_activeTrigger = null;

if (this.\_popper !== null) {

this.\_popper.destroy();

}

this.\_popper = null;

this.element = null;

this.config = null;

this.tip = null;

};

\_proto.show = function show() {

var \_this = this;

if ($$$1(this.element).css('display') === 'none') {

throw new Error('Please use show on visible elements');

}

var showEvent = $$$1.Event(this.constructor.Event.SHOW);

if (this.isWithContent() && this.\_isEnabled) {

$$$1(this.element).trigger(showEvent);

var isInTheDom = $$$1.contains(this.element.ownerDocument.documentElement, this.element);

if (showEvent.isDefaultPrevented() || !isInTheDom) {

return;

}

var tip = this.getTipElement();

var tipId = Util.getUID(this.constructor.NAME);

tip.setAttribute('id', tipId);

this.element.setAttribute('aria-describedby', tipId);

this.setContent();

if (this.config.animation) {

$$$1(tip).addClass(ClassName.FADE);

}

var placement = typeof this.config.placement === 'function' ? this.config.placement.call(this, tip, this.element) : this.config.placement;

var attachment = this.\_getAttachment(placement);

this.addAttachmentClass(attachment);

var container = this.config.container === false ? document.body : $$$1(this.config.container);

$$$1(tip).data(this.constructor.DATA\_KEY, this);

if (!$$$1.contains(this.element.ownerDocument.documentElement, this.tip)) {

$$$1(tip).appendTo(container);

}

$$$1(this.element).trigger(this.constructor.Event.INSERTED);

this.\_popper = new Popper(this.element, tip, {

placement: attachment,

modifiers: {

offset: {

offset: this.config.offset

},

flip: {

behavior: this.config.fallbackPlacement

},

arrow: {

element: Selector.ARROW

},

preventOverflow: {

boundariesElement: this.config.boundary

}

},

onCreate: function onCreate(data) {

if (data.originalPlacement !== data.placement) {

\_this.\_handlePopperPlacementChange(data);

}

},

onUpdate: function onUpdate(data) {

\_this.\_handlePopperPlacementChange(data);

}

});

$$$1(tip).addClass(ClassName.SHOW); // If this is a touch-enabled device we add extra

// empty mouseover listeners to the body's immediate children;

// only needed because of broken event delegation on iOS

// https://www.quirksmode.org/blog/archives/2014/02/mouse\_event\_bub.html

if ('ontouchstart' in document.documentElement) {

$$$1('body').children().on('mouseover', null, $$$1.noop);

}

var complete = function complete() {

if (\_this.config.animation) {

\_this.\_fixTransition();

}

var prevHoverState = \_this.\_hoverState;

\_this.\_hoverState = null;

$$$1(\_this.element).trigger(\_this.constructor.Event.SHOWN);

if (prevHoverState === HoverState.OUT) {

\_this.\_leave(null, \_this);

}

};

if (Util.supportsTransitionEnd() && $$$1(this.tip).hasClass(ClassName.FADE)) {

$$$1(this.tip).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(Tooltip.\_TRANSITION\_DURATION);

} else {

complete();

}

}

};

\_proto.hide = function hide(callback) {

var \_this2 = this;

var tip = this.getTipElement();

var hideEvent = $$$1.Event(this.constructor.Event.HIDE);

var complete = function complete() {

if (\_this2.\_hoverState !== HoverState.SHOW && tip.parentNode) {

tip.parentNode.removeChild(tip);

}

\_this2.\_cleanTipClass();

\_this2.element.removeAttribute('aria-describedby');

$$$1(\_this2.element).trigger(\_this2.constructor.Event.HIDDEN);

if (\_this2.\_popper !== null) {

\_this2.\_popper.destroy();

}

if (callback) {

callback();

}

};

$$$1(this.element).trigger(hideEvent);

if (hideEvent.isDefaultPrevented()) {

return;

}

$$$1(tip).removeClass(ClassName.SHOW); // If this is a touch-enabled device we remove the extra

// empty mouseover listeners we added for iOS support

if ('ontouchstart' in document.documentElement) {

$$$1('body').children().off('mouseover', null, $$$1.noop);

}

this.\_activeTrigger[Trigger.CLICK] = false;

this.\_activeTrigger[Trigger.FOCUS] = false;

this.\_activeTrigger[Trigger.HOVER] = false;

if (Util.supportsTransitionEnd() && $$$1(this.tip).hasClass(ClassName.FADE)) {

$$$1(tip).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(TRANSITION\_DURATION);

} else {

complete();

}

this.\_hoverState = '';

};

\_proto.update = function update() {

if (this.\_popper !== null) {

this.\_popper.scheduleUpdate();

}

}; // Protected

\_proto.isWithContent = function isWithContent() {

return Boolean(this.getTitle());

};

\_proto.addAttachmentClass = function addAttachmentClass(attachment) {

$$$1(this.getTipElement()).addClass(CLASS\_PREFIX + "-" + attachment);

};

\_proto.getTipElement = function getTipElement() {

this.tip = this.tip || $$$1(this.config.template)[0];

return this.tip;

};

\_proto.setContent = function setContent() {

var $tip = $$$1(this.getTipElement());

this.setElementContent($tip.find(Selector.TOOLTIP\_INNER), this.getTitle());

$tip.removeClass(ClassName.FADE + " " + ClassName.SHOW);

};

\_proto.setElementContent = function setElementContent($element, content) {

var html = this.config.html;

if (typeof content === 'object' && (content.nodeType || content.jquery)) {

// Content is a DOM node or a jQuery

if (html) {

if (!$$$1(content).parent().is($element)) {

$element.empty().append(content);

}

} else {

$element.text($$$1(content).text());

}

} else {

$element[html ? 'html' : 'text'](content);

}

};

\_proto.getTitle = function getTitle() {

var title = this.element.getAttribute('data-original-title');

if (!title) {

title = typeof this.config.title === 'function' ? this.config.title.call(this.element) : this.config.title;

}

return title;

}; // Private

\_proto.\_getAttachment = function \_getAttachment(placement) {

return AttachmentMap[placement.toUpperCase()];

};

\_proto.\_setListeners = function \_setListeners() {

var \_this3 = this;

var triggers = this.config.trigger.split(' ');

triggers.forEach(function (trigger) {

if (trigger === 'click') {

$$$1(\_this3.element).on(\_this3.constructor.Event.CLICK, \_this3.config.selector, function (event) {

return \_this3.toggle(event);

});

} else if (trigger !== Trigger.MANUAL) {

var eventIn = trigger === Trigger.HOVER ? \_this3.constructor.Event.MOUSEENTER : \_this3.constructor.Event.FOCUSIN;

var eventOut = trigger === Trigger.HOVER ? \_this3.constructor.Event.MOUSELEAVE : \_this3.constructor.Event.FOCUSOUT;

$$$1(\_this3.element).on(eventIn, \_this3.config.selector, function (event) {

return \_this3.\_enter(event);

}).on(eventOut, \_this3.config.selector, function (event) {

return \_this3.\_leave(event);

});

}

$$$1(\_this3.element).closest('.modal').on('hide.bs.modal', function () {

return \_this3.hide();

});

});

if (this.config.selector) {

this.config = \_extends({}, this.config, {

trigger: 'manual',

selector: ''

});

} else {

this.\_fixTitle();

}

};

\_proto.\_fixTitle = function \_fixTitle() {

var titleType = typeof this.element.getAttribute('data-original-title');

if (this.element.getAttribute('title') || titleType !== 'string') {

this.element.setAttribute('data-original-title', this.element.getAttribute('title') || '');

this.element.setAttribute('title', '');

}

};

\_proto.\_enter = function \_enter(event, context) {

var dataKey = this.constructor.DATA\_KEY;

context = context || $$$1(event.currentTarget).data(dataKey);

if (!context) {

context = new this.constructor(event.currentTarget, this.\_getDelegateConfig());

$$$1(event.currentTarget).data(dataKey, context);

}

if (event) {

context.\_activeTrigger[event.type === 'focusin' ? Trigger.FOCUS : Trigger.HOVER] = true;

}

if ($$$1(context.getTipElement()).hasClass(ClassName.SHOW) || context.\_hoverState === HoverState.SHOW) {

context.\_hoverState = HoverState.SHOW;

return;

}

clearTimeout(context.\_timeout);

context.\_hoverState = HoverState.SHOW;

if (!context.config.delay || !context.config.delay.show) {

context.show();

return;

}

context.\_timeout = setTimeout(function () {

if (context.\_hoverState === HoverState.SHOW) {

context.show();

}

}, context.config.delay.show);

};

\_proto.\_leave = function \_leave(event, context) {

var dataKey = this.constructor.DATA\_KEY;

context = context || $$$1(event.currentTarget).data(dataKey);

if (!context) {

context = new this.constructor(event.currentTarget, this.\_getDelegateConfig());

$$$1(event.currentTarget).data(dataKey, context);

}

if (event) {

context.\_activeTrigger[event.type === 'focusout' ? Trigger.FOCUS : Trigger.HOVER] = false;

}

if (context.\_isWithActiveTrigger()) {

return;

}

clearTimeout(context.\_timeout);

context.\_hoverState = HoverState.OUT;

if (!context.config.delay || !context.config.delay.hide) {

context.hide();

return;

}

context.\_timeout = setTimeout(function () {

if (context.\_hoverState === HoverState.OUT) {

context.hide();

}

}, context.config.delay.hide);

};

\_proto.\_isWithActiveTrigger = function \_isWithActiveTrigger() {

for (var trigger in this.\_activeTrigger) {

if (this.\_activeTrigger[trigger]) {

return true;

}

}

return false;

};

\_proto.\_getConfig = function \_getConfig(config) {

config = \_extends({}, this.constructor.Default, $$$1(this.element).data(), config);

if (typeof config.delay === 'number') {

config.delay = {

show: config.delay,

hide: config.delay

};

}

if (typeof config.title === 'number') {

config.title = config.title.toString();

}

if (typeof config.content === 'number') {

config.content = config.content.toString();

}

Util.typeCheckConfig(NAME, config, this.constructor.DefaultType);

return config;

};

\_proto.\_getDelegateConfig = function \_getDelegateConfig() {

var config = {};

if (this.config) {

for (var key in this.config) {

if (this.constructor.Default[key] !== this.config[key]) {

config[key] = this.config[key];

}

}

}

return config;

};

\_proto.\_cleanTipClass = function \_cleanTipClass() {

var $tip = $$$1(this.getTipElement());

var tabClass = $tip.attr('class').match(BSCLS\_PREFIX\_REGEX);

if (tabClass !== null && tabClass.length > 0) {

$tip.removeClass(tabClass.join(''));

}

};

\_proto.\_handlePopperPlacementChange = function \_handlePopperPlacementChange(data) {

this.\_cleanTipClass();

this.addAttachmentClass(this.\_getAttachment(data.placement));

};

\_proto.\_fixTransition = function \_fixTransition() {

var tip = this.getTipElement();

var initConfigAnimation = this.config.animation;

if (tip.getAttribute('x-placement') !== null) {

return;

}

$$$1(tip).removeClass(ClassName.FADE);

this.config.animation = false;

this.hide();

this.show();

this.config.animation = initConfigAnimation;

}; // Static

Tooltip.\_jQueryInterface = function \_jQueryInterface(config) {

return this.each(function () {

var data = $$$1(this).data(DATA\_KEY);

var \_config = typeof config === 'object' && config;

if (!data && /dispose|hide/.test(config)) {

return;

}

if (!data) {

data = new Tooltip(this, \_config);

$$$1(this).data(DATA\_KEY, data);

}

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

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

throw new TypeError("No method named \"" + config + "\"");

}

data[config]();

}

});

};

\_createClass(Tooltip, null, [{

key: "VERSION",

get: function get() {

return VERSION;

}

}, {

key: "Default",

get: function get() {

return Default;

}

}, {

key: "NAME",

get: function get() {

return NAME;

}

}, {

key: "DATA\_KEY",

get: function get() {

return DATA\_KEY;

}

}, {

key: "Event",

get: function get() {

return Event;

}

}, {

key: "EVENT\_KEY",

get: function get() {

return EVENT\_KEY;

}

}, {

key: "DefaultType",

get: function get() {

return DefaultType;

}

}]);

return Tooltip;

}();

/\*\*

\* ------------------------------------------------------------------------

\* jQuery

\* ------------------------------------------------------------------------

\*/

$$$1.fn[NAME] = Tooltip.\_jQueryInterface;

$$$1.fn[NAME].Constructor = Tooltip;

$$$1.fn[NAME].noConflict = function () {

$$$1.fn[NAME] = JQUERY\_NO\_CONFLICT;

return Tooltip.\_jQueryInterface;

};

return Tooltip;

}($, Popper);

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0): popover.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

var Popover = function ($$$1) {

/\*\*

\* ------------------------------------------------------------------------

\* Constants

\* ------------------------------------------------------------------------

\*/

var NAME = 'popover';

var VERSION = '4.0.0';

var DATA\_KEY = 'bs.popover';

var EVENT\_KEY = "." + DATA\_KEY;

var JQUERY\_NO\_CONFLICT = $$$1.fn[NAME];

var CLASS\_PREFIX = 'bs-popover';

var BSCLS\_PREFIX\_REGEX = new RegExp("(^|\\s)" + CLASS\_PREFIX + "\\S+", 'g');

var Default = \_extends({}, Tooltip.Default, {

placement: 'right',

trigger: 'click',

content: '',

template: '<div class="popover" role="tooltip">' + '<div class="arrow"></div>' + '<h3 class="popover-header"></h3>' + '<div class="popover-body"></div></div>'

});

var DefaultType = \_extends({}, Tooltip.DefaultType, {

content: '(string|element|function)'

});

var ClassName = {

FADE: 'fade',

SHOW: 'show'

};

var Selector = {

TITLE: '.popover-header',

CONTENT: '.popover-body'

};

var Event = {

HIDE: "hide" + EVENT\_KEY,

HIDDEN: "hidden" + EVENT\_KEY,

SHOW: "show" + EVENT\_KEY,

SHOWN: "shown" + EVENT\_KEY,

INSERTED: "inserted" + EVENT\_KEY,

CLICK: "click" + EVENT\_KEY,

FOCUSIN: "focusin" + EVENT\_KEY,

FOCUSOUT: "focusout" + EVENT\_KEY,

MOUSEENTER: "mouseenter" + EVENT\_KEY,

MOUSELEAVE: "mouseleave" + EVENT\_KEY

/\*\*

\* ------------------------------------------------------------------------

\* Class Definition

\* ------------------------------------------------------------------------

\*/

};

var Popover =

/\*#\_\_PURE\_\_\*/

function (\_Tooltip) {

\_inheritsLoose(Popover, \_Tooltip);

function Popover() {

return \_Tooltip.apply(this, arguments) || this;

}

var \_proto = Popover.prototype;

// Overrides

\_proto.isWithContent = function isWithContent() {

return this.getTitle() || this.\_getContent();

};

\_proto.addAttachmentClass = function addAttachmentClass(attachment) {

$$$1(this.getTipElement()).addClass(CLASS\_PREFIX + "-" + attachment);

};

\_proto.getTipElement = function getTipElement() {

this.tip = this.tip || $$$1(this.config.template)[0];

return this.tip;

};

\_proto.setContent = function setContent() {

var $tip = $$$1(this.getTipElement()); // We use append for html objects to maintain js events

this.setElementContent($tip.find(Selector.TITLE), this.getTitle());

var content = this.\_getContent();

if (typeof content === 'function') {

content = content.call(this.element);

}

this.setElementContent($tip.find(Selector.CONTENT), content);

$tip.removeClass(ClassName.FADE + " " + ClassName.SHOW);

}; // Private

\_proto.\_getContent = function \_getContent() {

return this.element.getAttribute('data-content') || this.config.content;

};

\_proto.\_cleanTipClass = function \_cleanTipClass() {

var $tip = $$$1(this.getTipElement());

var tabClass = $tip.attr('class').match(BSCLS\_PREFIX\_REGEX);

if (tabClass !== null && tabClass.length > 0) {

$tip.removeClass(tabClass.join(''));

}

}; // Static

Popover.\_jQueryInterface = function \_jQueryInterface(config) {

return this.each(function () {

var data = $$$1(this).data(DATA\_KEY);

var \_config = typeof config === 'object' ? config : null;

if (!data && /destroy|hide/.test(config)) {

return;

}

if (!data) {

data = new Popover(this, \_config);

$$$1(this).data(DATA\_KEY, data);

}

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

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

throw new TypeError("No method named \"" + config + "\"");

}

data[config]();

}

});

};

\_createClass(Popover, null, [{

key: "VERSION",

// Getters

get: function get() {

return VERSION;

}

}, {

key: "Default",

get: function get() {

return Default;

}

}, {

key: "NAME",

get: function get() {

return NAME;

}

}, {

key: "DATA\_KEY",

get: function get() {

return DATA\_KEY;

}

}, {

key: "Event",

get: function get() {

return Event;

}

}, {

key: "EVENT\_KEY",

get: function get() {

return EVENT\_KEY;

}

}, {

key: "DefaultType",

get: function get() {

return DefaultType;

}

}]);

return Popover;

}(Tooltip);

/\*\*

\* ------------------------------------------------------------------------

\* jQuery

\* ------------------------------------------------------------------------

\*/

$$$1.fn[NAME] = Popover.\_jQueryInterface;

$$$1.fn[NAME].Constructor = Popover;

$$$1.fn[NAME].noConflict = function () {

$$$1.fn[NAME] = JQUERY\_NO\_CONFLICT;

return Popover.\_jQueryInterface;

};

return Popover;

}($);

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0): scrollspy.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

var ScrollSpy = function ($$$1) {

/\*\*

\* ------------------------------------------------------------------------

\* Constants

\* ------------------------------------------------------------------------

\*/

var NAME = 'scrollspy';

var VERSION = '4.0.0';

var DATA\_KEY = 'bs.scrollspy';

var EVENT\_KEY = "." + DATA\_KEY;

var DATA\_API\_KEY = '.data-api';

var JQUERY\_NO\_CONFLICT = $$$1.fn[NAME];

var Default = {

offset: 10,

method: 'auto',

target: ''

};

var DefaultType = {

offset: 'number',

method: 'string',

target: '(string|element)'

};

var Event = {

ACTIVATE: "activate" + EVENT\_KEY,

SCROLL: "scroll" + EVENT\_KEY,

LOAD\_DATA\_API: "load" + EVENT\_KEY + DATA\_API\_KEY

};

var ClassName = {

DROPDOWN\_ITEM: 'dropdown-item',

DROPDOWN\_MENU: 'dropdown-menu',

ACTIVE: 'active'

};

var Selector = {

DATA\_SPY: '[data-spy="scroll"]',

ACTIVE: '.active',

NAV\_LIST\_GROUP: '.nav, .list-group',

NAV\_LINKS: '.nav-link',

NAV\_ITEMS: '.nav-item',

LIST\_ITEMS: '.list-group-item',

DROPDOWN: '.dropdown',

DROPDOWN\_ITEMS: '.dropdown-item',

DROPDOWN\_TOGGLE: '.dropdown-toggle'

};

var OffsetMethod = {

OFFSET: 'offset',

POSITION: 'position'

/\*\*

\* ------------------------------------------------------------------------

\* Class Definition

\* ------------------------------------------------------------------------

\*/

};

var ScrollSpy =

/\*#\_\_PURE\_\_\*/

function () {

function ScrollSpy(element, config) {

var \_this = this;

this.\_element = element;

this.\_scrollElement = element.tagName === 'BODY' ? window : element;

this.\_config = this.\_getConfig(config);

this.\_selector = this.\_config.target + " " + Selector.NAV\_LINKS + "," + (this.\_config.target + " " + Selector.LIST\_ITEMS + ",") + (this.\_config.target + " " + Selector.DROPDOWN\_ITEMS);

this.\_offsets = [];

this.\_targets = [];

this.\_activeTarget = null;

this.\_scrollHeight = 0;

$$$1(this.\_scrollElement).on(Event.SCROLL, function (event) {

return \_this.\_process(event);

});

this.refresh();

this.\_process();

} // Getters

var \_proto = ScrollSpy.prototype;

// Public

\_proto.refresh = function refresh() {

var \_this2 = this;

var autoMethod = this.\_scrollElement === this.\_scrollElement.window ? OffsetMethod.OFFSET : OffsetMethod.POSITION;

var offsetMethod = this.\_config.method === 'auto' ? autoMethod : this.\_config.method;

var offsetBase = offsetMethod === OffsetMethod.POSITION ? this.\_getScrollTop() : 0;

this.\_offsets = [];

this.\_targets = [];

this.\_scrollHeight = this.\_getScrollHeight();

var targets = $$$1.makeArray($$$1(this.\_selector));

targets.map(function (element) {

var target;

var targetSelector = Util.getSelectorFromElement(element);

if (targetSelector) {

target = $$$1(targetSelector)[0];

}

if (target) {

var targetBCR = target.getBoundingClientRect();

if (targetBCR.width || targetBCR.height) {

// TODO (fat): remove sketch reliance on jQuery position/offset

return [$$$1(target)[offsetMethod]().top + offsetBase, targetSelector];

}

}

return null;

}).filter(function (item) {

return item;

}).sort(function (a, b) {

return a[0] - b[0];

}).forEach(function (item) {

\_this2.\_offsets.push(item[0]);

\_this2.\_targets.push(item[1]);

});

};

\_proto.dispose = function dispose() {

$$$1.removeData(this.\_element, DATA\_KEY);

$$$1(this.\_scrollElement).off(EVENT\_KEY);

this.\_element = null;

this.\_scrollElement = null;

this.\_config = null;

this.\_selector = null;

this.\_offsets = null;

this.\_targets = null;

this.\_activeTarget = null;

this.\_scrollHeight = null;

}; // Private

\_proto.\_getConfig = function \_getConfig(config) {

config = \_extends({}, Default, config);

if (typeof config.target !== 'string') {

var id = $$$1(config.target).attr('id');

if (!id) {

id = Util.getUID(NAME);

$$$1(config.target).attr('id', id);

}

config.target = "#" + id;

}

Util.typeCheckConfig(NAME, config, DefaultType);

return config;

};

\_proto.\_getScrollTop = function \_getScrollTop() {

return this.\_scrollElement === window ? this.\_scrollElement.pageYOffset : this.\_scrollElement.scrollTop;

};

\_proto.\_getScrollHeight = function \_getScrollHeight() {

return this.\_scrollElement.scrollHeight || Math.max(document.body.scrollHeight, document.documentElement.scrollHeight);

};

\_proto.\_getOffsetHeight = function \_getOffsetHeight() {

return this.\_scrollElement === window ? window.innerHeight : this.\_scrollElement.getBoundingClientRect().height;

};

\_proto.\_process = function \_process() {

var scrollTop = this.\_getScrollTop() + this.\_config.offset;

var scrollHeight = this.\_getScrollHeight();

var maxScroll = this.\_config.offset + scrollHeight - this.\_getOffsetHeight();

if (this.\_scrollHeight !== scrollHeight) {

this.refresh();

}

if (scrollTop >= maxScroll) {

var target = this.\_targets[this.\_targets.length - 1];

if (this.\_activeTarget !== target) {

this.\_activate(target);

}

return;

}

if (this.\_activeTarget && scrollTop < this.\_offsets[0] && this.\_offsets[0] > 0) {

this.\_activeTarget = null;

this.\_clear();

return;

}

for (var i = this.\_offsets.length; i--;) {

var isActiveTarget = this.\_activeTarget !== this.\_targets[i] && scrollTop >= this.\_offsets[i] && (typeof this.\_offsets[i + 1] === 'undefined' || scrollTop < this.\_offsets[i + 1]);

if (isActiveTarget) {

this.\_activate(this.\_targets[i]);

}

}

};

\_proto.\_activate = function \_activate(target) {

this.\_activeTarget = target;

this.\_clear();

var queries = this.\_selector.split(','); // eslint-disable-next-line arrow-body-style

queries = queries.map(function (selector) {

return selector + "[data-target=\"" + target + "\"]," + (selector + "[href=\"" + target + "\"]");

});

var $link = $$$1(queries.join(','));

if ($link.hasClass(ClassName.DROPDOWN\_ITEM)) {

$link.closest(Selector.DROPDOWN).find(Selector.DROPDOWN\_TOGGLE).addClass(ClassName.ACTIVE);

$link.addClass(ClassName.ACTIVE);

} else {

// Set triggered link as active

$link.addClass(ClassName.ACTIVE); // Set triggered links parents as active

// With both <ul> and <nav> markup a parent is the previous sibling of any nav ancestor

$link.parents(Selector.NAV\_LIST\_GROUP).prev(Selector.NAV\_LINKS + ", " + Selector.LIST\_ITEMS).addClass(ClassName.ACTIVE); // Handle special case when .nav-link is inside .nav-item

$link.parents(Selector.NAV\_LIST\_GROUP).prev(Selector.NAV\_ITEMS).children(Selector.NAV\_LINKS).addClass(ClassName.ACTIVE);

}

$$$1(this.\_scrollElement).trigger(Event.ACTIVATE, {

relatedTarget: target

});

};

\_proto.\_clear = function \_clear() {

$$$1(this.\_selector).filter(Selector.ACTIVE).removeClass(ClassName.ACTIVE);

}; // Static

ScrollSpy.\_jQueryInterface = function \_jQueryInterface(config) {

return this.each(function () {

var data = $$$1(this).data(DATA\_KEY);

var \_config = typeof config === 'object' && config;

if (!data) {

data = new ScrollSpy(this, \_config);

$$$1(this).data(DATA\_KEY, data);

}

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

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

throw new TypeError("No method named \"" + config + "\"");

}

data[config]();

}

});

};

\_createClass(ScrollSpy, null, [{

key: "VERSION",

get: function get() {

return VERSION;

}

}, {

key: "Default",

get: function get() {

return Default;

}

}]);

return ScrollSpy;

}();

/\*\*

\* ------------------------------------------------------------------------

\* Data Api implementation

\* ------------------------------------------------------------------------

\*/

$$$1(window).on(Event.LOAD\_DATA\_API, function () {

var scrollSpys = $$$1.makeArray($$$1(Selector.DATA\_SPY));

for (var i = scrollSpys.length; i--;) {

var $spy = $$$1(scrollSpys[i]);

ScrollSpy.\_jQueryInterface.call($spy, $spy.data());

}

});

/\*\*

\* ------------------------------------------------------------------------

\* jQuery

\* ------------------------------------------------------------------------

\*/

$$$1.fn[NAME] = ScrollSpy.\_jQueryInterface;

$$$1.fn[NAME].Constructor = ScrollSpy;

$$$1.fn[NAME].noConflict = function () {

$$$1.fn[NAME] = JQUERY\_NO\_CONFLICT;

return ScrollSpy.\_jQueryInterface;

};

return ScrollSpy;

}($);

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0): tab.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

var Tab = function ($$$1) {

/\*\*

\* ------------------------------------------------------------------------

\* Constants

\* ------------------------------------------------------------------------

\*/

var NAME = 'tab';

var VERSION = '4.0.0';

var DATA\_KEY = 'bs.tab';

var EVENT\_KEY = "." + DATA\_KEY;

var DATA\_API\_KEY = '.data-api';

var JQUERY\_NO\_CONFLICT = $$$1.fn[NAME];

var TRANSITION\_DURATION = 150;

var Event = {

HIDE: "hide" + EVENT\_KEY,

HIDDEN: "hidden" + EVENT\_KEY,

SHOW: "show" + EVENT\_KEY,

SHOWN: "shown" + EVENT\_KEY,

CLICK\_DATA\_API: "click" + EVENT\_KEY + DATA\_API\_KEY

};

var ClassName = {

DROPDOWN\_MENU: 'dropdown-menu',

ACTIVE: 'active',

DISABLED: 'disabled',

FADE: 'fade',

SHOW: 'show'

};

var Selector = {

DROPDOWN: '.dropdown',

NAV\_LIST\_GROUP: '.nav, .list-group',

ACTIVE: '.active',

ACTIVE\_UL: '> li > .active',

DATA\_TOGGLE: '[data-toggle="tab"], [data-toggle="pill"], [data-toggle="list"]',

DROPDOWN\_TOGGLE: '.dropdown-toggle',

DROPDOWN\_ACTIVE\_CHILD: '> .dropdown-menu .active'

/\*\*

\* ------------------------------------------------------------------------

\* Class Definition

\* ------------------------------------------------------------------------

\*/

};

var Tab =

/\*#\_\_PURE\_\_\*/

function () {

function Tab(element) {

this.\_element = element;

} // Getters

var \_proto = Tab.prototype;

// Public

\_proto.show = function show() {

var \_this = this;

if (this.\_element.parentNode && this.\_element.parentNode.nodeType === Node.ELEMENT\_NODE && $$$1(this.\_element).hasClass(ClassName.ACTIVE) || $$$1(this.\_element).hasClass(ClassName.DISABLED)) {

return;

}

var target;

var previous;

var listElement = $$$1(this.\_element).closest(Selector.NAV\_LIST\_GROUP)[0];

var selector = Util.getSelectorFromElement(this.\_element);

if (listElement) {

var itemSelector = listElement.nodeName === 'UL' ? Selector.ACTIVE\_UL : Selector.ACTIVE;

previous = $$$1.makeArray($$$1(listElement).find(itemSelector));

previous = previous[previous.length - 1];

}

var hideEvent = $$$1.Event(Event.HIDE, {

relatedTarget: this.\_element

});

var showEvent = $$$1.Event(Event.SHOW, {

relatedTarget: previous

});

if (previous) {

$$$1(previous).trigger(hideEvent);

}

$$$1(this.\_element).trigger(showEvent);

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

return;

}

if (selector) {

target = $$$1(selector)[0];

}

this.\_activate(this.\_element, listElement);

var complete = function complete() {

var hiddenEvent = $$$1.Event(Event.HIDDEN, {

relatedTarget: \_this.\_element

});

var shownEvent = $$$1.Event(Event.SHOWN, {

relatedTarget: previous

});

$$$1(previous).trigger(hiddenEvent);

$$$1(\_this.\_element).trigger(shownEvent);

};

if (target) {

this.\_activate(target, target.parentNode, complete);

} else {

complete();

}

};

\_proto.dispose = function dispose() {

$$$1.removeData(this.\_element, DATA\_KEY);

this.\_element = null;

}; // Private

\_proto.\_activate = function \_activate(element, container, callback) {

var \_this2 = this;

var activeElements;

if (container.nodeName === 'UL') {

activeElements = $$$1(container).find(Selector.ACTIVE\_UL);

} else {

activeElements = $$$1(container).children(Selector.ACTIVE);

}

var active = activeElements[0];

var isTransitioning = callback && Util.supportsTransitionEnd() && active && $$$1(active).hasClass(ClassName.FADE);

var complete = function complete() {

return \_this2.\_transitionComplete(element, active, callback);

};

if (active && isTransitioning) {

$$$1(active).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(TRANSITION\_DURATION);

} else {

complete();

}

};

\_proto.\_transitionComplete = function \_transitionComplete(element, active, callback) {

if (active) {

$$$1(active).removeClass(ClassName.SHOW + " " + ClassName.ACTIVE);

var dropdownChild = $$$1(active.parentNode).find(Selector.DROPDOWN\_ACTIVE\_CHILD)[0];

if (dropdownChild) {

$$$1(dropdownChild).removeClass(ClassName.ACTIVE);

}

if (active.getAttribute('role') === 'tab') {

active.setAttribute('aria-selected', false);

}

}

$$$1(element).addClass(ClassName.ACTIVE);

if (element.getAttribute('role') === 'tab') {

element.setAttribute('aria-selected', true);

}

Util.reflow(element);

$$$1(element).addClass(ClassName.SHOW);

if (element.parentNode && $$$1(element.parentNode).hasClass(ClassName.DROPDOWN\_MENU)) {

var dropdownElement = $$$1(element).closest(Selector.DROPDOWN)[0];

if (dropdownElement) {

$$$1(dropdownElement).find(Selector.DROPDOWN\_TOGGLE).addClass(ClassName.ACTIVE);

}

element.setAttribute('aria-expanded', true);

}

if (callback) {

callback();

}

}; // Static

Tab.\_jQueryInterface = function \_jQueryInterface(config) {

return this.each(function () {

var $this = $$$1(this);

var data = $this.data(DATA\_KEY);

if (!data) {

data = new Tab(this);

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

}

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

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

throw new TypeError("No method named \"" + config + "\"");

}

data[config]();

}

});

};

\_createClass(Tab, null, [{

key: "VERSION",

get: function get() {

return VERSION;

}

}]);

return Tab;

}();

/\*\*

\* ------------------------------------------------------------------------

\* Data Api implementation

\* ------------------------------------------------------------------------

\*/

$$$1(document).on(Event.CLICK\_DATA\_API, Selector.DATA\_TOGGLE, function (event) {

event.preventDefault();

Tab.\_jQueryInterface.call($$$1(this), 'show');

});

/\*\*

\* ------------------------------------------------------------------------

\* jQuery

\* ------------------------------------------------------------------------

\*/

$$$1.fn[NAME] = Tab.\_jQueryInterface;

$$$1.fn[NAME].Constructor = Tab;

$$$1.fn[NAME].noConflict = function () {

$$$1.fn[NAME] = JQUERY\_NO\_CONFLICT;

return Tab.\_jQueryInterface;

};

return Tab;

}($);

/\*\*

\* --------------------------------------------------------------------------

\* Bootstrap (v4.0.0-alpha.6): index.js

\* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

\* --------------------------------------------------------------------------

\*/

(function ($$$1) {

if (typeof $$$1 === 'undefined') {

throw new TypeError('Bootstrap\'s JavaScript requires jQuery. jQuery must be included before Bootstrap\'s JavaScript.');

}

var version = $$$1.fn.jquery.split(' ')[0].split('.');

var minMajor = 1;

var ltMajor = 2;

var minMinor = 9;

var minPatch = 1;

var maxMajor = 4;

if (version[0] < ltMajor && version[1] < minMinor || version[0] === minMajor && version[1] === minMinor && version[2] < minPatch || version[0] >= maxMajor) {

throw new Error('Bootstrap\'s JavaScript requires at least jQuery v1.9.1 but less than v4.0.0');

}

})($);

exports.Util = Util;

exports.Alert = Alert;

exports.Button = Button;

exports.Carousel = Carousel;

exports.Collapse = Collapse;

exports.Dropdown = Dropdown;

exports.Modal = Modal;

exports.Popover = Popover;

exports.Scrollspy = ScrollSpy;

exports.Tab = Tab;

exports.Tooltip = Tooltip;

Object.defineProperty(exports, '\_\_esModule', { value: true });

})));

//# sourceMappingURL=bootstrap.js.map