(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) :

(global = typeof globalThis !== 'undefined' ? globalThis : global || self, factory(global.bootstrap = {}, global.jQuery, global.Popper));

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

function \_interopDefaultLegacy (e) { return e && typeof e === 'object' && 'default' in e ? e : { 'default': e }; }

var $\_\_default = /\*#\_\_PURE\_\_\*/\_interopDefaultLegacy($);

var Popper\_\_default = /\*#\_\_PURE\_\_\*/\_interopDefaultLegacy(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;

\_setPrototypeOf(subClass, superClass);

}

function \_setPrototypeOf(o, p) {

\_setPrototypeOf = Object.setPrototypeOf || function \_setPrototypeOf(o, p) {

o.\_\_proto\_\_ = p;

return o;

};

return \_setPrototypeOf(o, p);

}

/\*\*

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

\* Bootstrap (v4.6.1): util.js

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

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

\*/

/\*\*

\* Private TransitionEnd Helpers

\*/

var TRANSITION\_END = 'transitionend';

var MAX\_UID = 1000000;

var MILLISECONDS\_MULTIPLIER = 1000; // Shoutout AngusCroll (https://goo.gl/pxwQGp)

function toType(obj) {

if (obj === null || typeof obj === 'undefined') {

return "" + obj;

}

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

}

function getSpecialTransitionEndEvent() {

return {

bindType: TRANSITION\_END,

delegateType: TRANSITION\_END,

handle: function handle(event) {

if ($\_\_default["default"](event.target).is(this)) {

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

}

return undefined;

}

};

}

function transitionEndEmulator(duration) {

var \_this = this;

var called = false;

$\_\_default["default"](this).one(Util.TRANSITION\_END, function () {

called = true;

});

setTimeout(function () {

if (!called) {

Util.triggerTransitionEnd(\_this);

}

}, duration);

return this;

}

function setTransitionEndSupport() {

$\_\_default["default"].fn.emulateTransitionEnd = transitionEndEmulator;

$\_\_default["default"].event.special[Util.TRANSITION\_END] = getSpecialTransitionEndEvent();

}

/\*\*

\* 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 === '#') {

var hrefAttr = element.getAttribute('href');

selector = hrefAttr && hrefAttr !== '#' ? hrefAttr.trim() : '';

}

try {

return document.querySelector(selector) ? selector : null;

} catch (\_) {

return null;

}

},

getTransitionDurationFromElement: function getTransitionDurationFromElement(element) {

if (!element) {

return 0;

} // Get transition-duration of the element

var transitionDuration = $\_\_default["default"](element).css('transition-duration');

var transitionDelay = $\_\_default["default"](element).css('transition-delay');

var floatTransitionDuration = parseFloat(transitionDuration);

var floatTransitionDelay = parseFloat(transitionDelay); // Return 0 if element or transition duration is not found

if (!floatTransitionDuration && !floatTransitionDelay) {

return 0;

} // If multiple durations are defined, take the first

transitionDuration = transitionDuration.split(',')[0];

transitionDelay = transitionDelay.split(',')[0];

return (parseFloat(transitionDuration) + parseFloat(transitionDelay)) \* MILLISECONDS\_MULTIPLIER;

},

reflow: function reflow(element) {

return element.offsetHeight;

},

triggerTransitionEnd: function triggerTransitionEnd(element) {

$\_\_default["default"](element).trigger(TRANSITION\_END);

},

supportsTransitionEnd: function supportsTransitionEnd() {

return Boolean(TRANSITION\_END);

},

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 + "\"."));

}

}

}

},

findShadowRoot: function findShadowRoot(element) {

if (!document.documentElement.attachShadow) {

return null;

} // Can find the shadow root otherwise it'll return the document

if (typeof element.getRootNode === 'function') {

var root = element.getRootNode();

return root instanceof ShadowRoot ? root : null;

}

if (element instanceof ShadowRoot) {

return element;

} // when we don't find a shadow root

if (!element.parentNode) {

return null;

}

return Util.findShadowRoot(element.parentNode);

},

jQueryDetection: function jQueryDetection() {

if (typeof $\_\_default["default"] === 'undefined') {

throw new TypeError('Bootstrap\'s JavaScript requires jQuery. jQuery must be included before Bootstrap\'s JavaScript.');

}

var version = $\_\_default["default"].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');

}

}

};

Util.jQueryDetection();

setTransitionEndSupport();

/\*\*

\* Constants

\*/

var NAME$a = 'alert';

var VERSION$a = '4.6.1';

var DATA\_KEY$a = 'bs.alert';

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

var DATA\_API\_KEY$7 = '.data-api';

var JQUERY\_NO\_CONFLICT$a = $\_\_default["default"].fn[NAME$a];

var CLASS\_NAME\_ALERT = 'alert';

var CLASS\_NAME\_FADE$5 = 'fade';

var CLASS\_NAME\_SHOW$7 = 'show';

var EVENT\_CLOSE = "close" + EVENT\_KEY$a;

var EVENT\_CLOSED = "closed" + EVENT\_KEY$a;

var EVENT\_CLICK\_DATA\_API$6 = "click" + EVENT\_KEY$a + DATA\_API\_KEY$7;

var SELECTOR\_DISMISS = '[data-dismiss="alert"]';

/\*\*

\* Class definition

\*/

var Alert = /\*#\_\_PURE\_\_\*/function () {

function Alert(element) {

this.\_element = element;

} // Getters

var \_proto = Alert.prototype;

// Public

\_proto.close = function close(element) {

var rootElement = this.\_element;

if (element) {

rootElement = this.\_getRootElement(element);

}

var customEvent = this.\_triggerCloseEvent(rootElement);

if (customEvent.isDefaultPrevented()) {

return;

}

this.\_removeElement(rootElement);

};

\_proto.dispose = function dispose() {

$\_\_default["default"].removeData(this.\_element, DATA\_KEY$a);

this.\_element = null;

} // Private

;

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

var selector = Util.getSelectorFromElement(element);

var parent = false;

if (selector) {

parent = document.querySelector(selector);

}

if (!parent) {

parent = $\_\_default["default"](element).closest("." + CLASS\_NAME\_ALERT)[0];

}

return parent;

};

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

var closeEvent = $\_\_default["default"].Event(EVENT\_CLOSE);

$\_\_default["default"](element).trigger(closeEvent);

return closeEvent;

};

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

var \_this = this;

$\_\_default["default"](element).removeClass(CLASS\_NAME\_SHOW$7);

if (!$\_\_default["default"](element).hasClass(CLASS\_NAME\_FADE$5)) {

this.\_destroyElement(element);

return;

}

var transitionDuration = Util.getTransitionDurationFromElement(element);

$\_\_default["default"](element).one(Util.TRANSITION\_END, function (event) {

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

}).emulateTransitionEnd(transitionDuration);

};

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

$\_\_default["default"](element).detach().trigger(EVENT\_CLOSED).remove();

} // Static

;

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

return this.each(function () {

var $element = $\_\_default["default"](this);

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

if (!data) {

data = new Alert(this);

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

}

}]);

return Alert;

}();

/\*\*

\* Data API implementation

\*/

$\_\_default["default"](document).on(EVENT\_CLICK\_DATA\_API$6, SELECTOR\_DISMISS, Alert.\_handleDismiss(new Alert()));

/\*\*

\* jQuery

\*/

$\_\_default["default"].fn[NAME$a] = Alert.\_jQueryInterface;

$\_\_default["default"].fn[NAME$a].Constructor = Alert;

$\_\_default["default"].fn[NAME$a].noConflict = function () {

$\_\_default["default"].fn[NAME$a] = JQUERY\_NO\_CONFLICT$a;

return Alert.\_jQueryInterface;

};

/\*\*

\* Constants

\*/

var NAME$9 = 'button';

var VERSION$9 = '4.6.1';

var DATA\_KEY$9 = 'bs.button';

var EVENT\_KEY$9 = "." + DATA\_KEY$9;

var DATA\_API\_KEY$6 = '.data-api';

var JQUERY\_NO\_CONFLICT$9 = $\_\_default["default"].fn[NAME$9];

var CLASS\_NAME\_ACTIVE$3 = 'active';

var CLASS\_NAME\_BUTTON = 'btn';

var CLASS\_NAME\_FOCUS = 'focus';

var EVENT\_CLICK\_DATA\_API$5 = "click" + EVENT\_KEY$9 + DATA\_API\_KEY$6;

var EVENT\_FOCUS\_BLUR\_DATA\_API = "focus" + EVENT\_KEY$9 + DATA\_API\_KEY$6 + " " + ("blur" + EVENT\_KEY$9 + DATA\_API\_KEY$6);

var EVENT\_LOAD\_DATA\_API$2 = "load" + EVENT\_KEY$9 + DATA\_API\_KEY$6;

var SELECTOR\_DATA\_TOGGLE\_CARROT = '[data-toggle^="button"]';

var SELECTOR\_DATA\_TOGGLES = '[data-toggle="buttons"]';

var SELECTOR\_DATA\_TOGGLE$4 = '[data-toggle="button"]';

var SELECTOR\_DATA\_TOGGLES\_BUTTONS = '[data-toggle="buttons"] .btn';

var SELECTOR\_INPUT = 'input:not([type="hidden"])';

var SELECTOR\_ACTIVE$2 = '.active';

var SELECTOR\_BUTTON = '.btn';

/\*\*

\* Class definition

\*/

var Button = /\*#\_\_PURE\_\_\*/function () {

function Button(element) {

this.\_element = element;

this.shouldAvoidTriggerChange = false;

} // Getters

var \_proto = Button.prototype;

// Public

\_proto.toggle = function toggle() {

var triggerChangeEvent = true;

var addAriaPressed = true;

var rootElement = $\_\_default["default"](this.\_element).closest(SELECTOR\_DATA\_TOGGLES)[0];

if (rootElement) {

var input = this.\_element.querySelector(SELECTOR\_INPUT);

if (input) {

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

if (input.checked && this.\_element.classList.contains(CLASS\_NAME\_ACTIVE$3)) {

triggerChangeEvent = false;

} else {

var activeElement = rootElement.querySelector(SELECTOR\_ACTIVE$2);

if (activeElement) {

$\_\_default["default"](activeElement).removeClass(CLASS\_NAME\_ACTIVE$3);

}

}

}

if (triggerChangeEvent) {

// if it's not a radio button or checkbox don't add a pointless/invalid checked property to the input

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

input.checked = !this.\_element.classList.contains(CLASS\_NAME\_ACTIVE$3);

}

if (!this.shouldAvoidTriggerChange) {

$\_\_default["default"](input).trigger('change');

}

}

input.focus();

addAriaPressed = false;

}

}

if (!(this.\_element.hasAttribute('disabled') || this.\_element.classList.contains('disabled'))) {

if (addAriaPressed) {

this.\_element.setAttribute('aria-pressed', !this.\_element.classList.contains(CLASS\_NAME\_ACTIVE$3));

}

if (triggerChangeEvent) {

$\_\_default["default"](this.\_element).toggleClass(CLASS\_NAME\_ACTIVE$3);

}

}

};

\_proto.dispose = function dispose() {

$\_\_default["default"].removeData(this.\_element, DATA\_KEY$9);

this.\_element = null;

} // Static

;

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

return this.each(function () {

var $element = $\_\_default["default"](this);

var data = $element.data(DATA\_KEY$9);

if (!data) {

data = new Button(this);

$element.data(DATA\_KEY$9, data);

}

data.shouldAvoidTriggerChange = avoidTriggerChange;

if (config === 'toggle') {

data[config]();

}

});

};

\_createClass(Button, null, [{

key: "VERSION",

get: function get() {

return VERSION$9;

}

}]);

return Button;

}();

/\*\*

\* Data API implementation

\*/

$\_\_default["default"](document).on(EVENT\_CLICK\_DATA\_API$5, SELECTOR\_DATA\_TOGGLE\_CARROT, function (event) {

var button = event.target;

var initialButton = button;

if (!$\_\_default["default"](button).hasClass(CLASS\_NAME\_BUTTON)) {

button = $\_\_default["default"](button).closest(SELECTOR\_BUTTON)[0];

}

if (!button || button.hasAttribute('disabled') || button.classList.contains('disabled')) {

event.preventDefault(); // work around Firefox bug #1540995

} else {

var inputBtn = button.querySelector(SELECTOR\_INPUT);

if (inputBtn && (inputBtn.hasAttribute('disabled') || inputBtn.classList.contains('disabled'))) {

event.preventDefault(); // work around Firefox bug #1540995

return;

}

if (initialButton.tagName === 'INPUT' || button.tagName !== 'LABEL') {

Button.\_jQueryInterface.call($\_\_default["default"](button), 'toggle', initialButton.tagName === 'INPUT');

}

}

}).on(EVENT\_FOCUS\_BLUR\_DATA\_API, SELECTOR\_DATA\_TOGGLE\_CARROT, function (event) {

var button = $\_\_default["default"](event.target).closest(SELECTOR\_BUTTON)[0];

$\_\_default["default"](button).toggleClass(CLASS\_NAME\_FOCUS, /^focus(in)?$/.test(event.type));

});

$\_\_default["default"](window).on(EVENT\_LOAD\_DATA\_API$2, function () {

// ensure correct active class is set to match the controls' actual values/states

// find all checkboxes/readio buttons inside data-toggle groups

var buttons = [].slice.call(document.querySelectorAll(SELECTOR\_DATA\_TOGGLES\_BUTTONS));

for (var i = 0, len = buttons.length; i < len; i++) {

var button = buttons[i];

var input = button.querySelector(SELECTOR\_INPUT);

if (input.checked || input.hasAttribute('checked')) {

button.classList.add(CLASS\_NAME\_ACTIVE$3);

} else {

button.classList.remove(CLASS\_NAME\_ACTIVE$3);

}

} // find all button toggles

buttons = [].slice.call(document.querySelectorAll(SELECTOR\_DATA\_TOGGLE$4));

for (var \_i = 0, \_len = buttons.length; \_i < \_len; \_i++) {

var \_button = buttons[\_i];

if (\_button.getAttribute('aria-pressed') === 'true') {

\_button.classList.add(CLASS\_NAME\_ACTIVE$3);

} else {

\_button.classList.remove(CLASS\_NAME\_ACTIVE$3);

}

}

});

/\*\*

\* jQuery

\*/

$\_\_default["default"].fn[NAME$9] = Button.\_jQueryInterface;

$\_\_default["default"].fn[NAME$9].Constructor = Button;

$\_\_default["default"].fn[NAME$9].noConflict = function () {

$\_\_default["default"].fn[NAME$9] = JQUERY\_NO\_CONFLICT$9;

return Button.\_jQueryInterface;

};

/\*\*

\* Constants

\*/

var NAME$8 = 'carousel';

var VERSION$8 = '4.6.1';

var DATA\_KEY$8 = 'bs.carousel';

var EVENT\_KEY$8 = "." + DATA\_KEY$8;

var DATA\_API\_KEY$5 = '.data-api';

var JQUERY\_NO\_CONFLICT$8 = $\_\_default["default"].fn[NAME$8];

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 SWIPE\_THRESHOLD = 40;

var CLASS\_NAME\_CAROUSEL = 'carousel';

var CLASS\_NAME\_ACTIVE$2 = 'active';

var CLASS\_NAME\_SLIDE = 'slide';

var CLASS\_NAME\_RIGHT = 'carousel-item-right';

var CLASS\_NAME\_LEFT = 'carousel-item-left';

var CLASS\_NAME\_NEXT = 'carousel-item-next';

var CLASS\_NAME\_PREV = 'carousel-item-prev';

var CLASS\_NAME\_POINTER\_EVENT = 'pointer-event';

var DIRECTION\_NEXT = 'next';

var DIRECTION\_PREV = 'prev';

var DIRECTION\_LEFT = 'left';

var DIRECTION\_RIGHT = 'right';

var EVENT\_SLIDE = "slide" + EVENT\_KEY$8;

var EVENT\_SLID = "slid" + EVENT\_KEY$8;

var EVENT\_KEYDOWN = "keydown" + EVENT\_KEY$8;

var EVENT\_MOUSEENTER = "mouseenter" + EVENT\_KEY$8;

var EVENT\_MOUSELEAVE = "mouseleave" + EVENT\_KEY$8;

var EVENT\_TOUCHSTART = "touchstart" + EVENT\_KEY$8;

var EVENT\_TOUCHMOVE = "touchmove" + EVENT\_KEY$8;

var EVENT\_TOUCHEND = "touchend" + EVENT\_KEY$8;

var EVENT\_POINTERDOWN = "pointerdown" + EVENT\_KEY$8;

var EVENT\_POINTERUP = "pointerup" + EVENT\_KEY$8;

var EVENT\_DRAG\_START = "dragstart" + EVENT\_KEY$8;

var EVENT\_LOAD\_DATA\_API$1 = "load" + EVENT\_KEY$8 + DATA\_API\_KEY$5;

var EVENT\_CLICK\_DATA\_API$4 = "click" + EVENT\_KEY$8 + DATA\_API\_KEY$5;

var SELECTOR\_ACTIVE$1 = '.active';

var SELECTOR\_ACTIVE\_ITEM = '.active.carousel-item';

var SELECTOR\_ITEM = '.carousel-item';

var SELECTOR\_ITEM\_IMG = '.carousel-item img';

var SELECTOR\_NEXT\_PREV = '.carousel-item-next, .carousel-item-prev';

var SELECTOR\_INDICATORS = '.carousel-indicators';

var SELECTOR\_DATA\_SLIDE = '[data-slide], [data-slide-to]';

var SELECTOR\_DATA\_RIDE = '[data-ride="carousel"]';

var Default$7 = {

interval: 5000,

keyboard: true,

slide: false,

pause: 'hover',

wrap: true,

touch: true

};

var DefaultType$7 = {

interval: '(number|boolean)',

keyboard: 'boolean',

slide: '(boolean|string)',

pause: '(string|boolean)',

wrap: 'boolean',

touch: 'boolean'

};

var PointerType = {

TOUCH: 'touch',

PEN: 'pen'

};

/\*\*

\* 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.touchStartX = 0;

this.touchDeltaX = 0;

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

this.\_element = element;

this.\_indicatorsElement = this.\_element.querySelector(SELECTOR\_INDICATORS);

this.\_touchSupported = 'ontouchstart' in document.documentElement || navigator.maxTouchPoints > 0;

this.\_pointerEvent = Boolean(window.PointerEvent || window.MSPointerEvent);

this.\_addEventListeners();

} // Getters

var \_proto = Carousel.prototype;

// Public

\_proto.next = function next() {

if (!this.\_isSliding) {

this.\_slide(DIRECTION\_NEXT);

}

};

\_proto.nextWhenVisible = function nextWhenVisible() {

var $element = $\_\_default["default"](this.\_element); // Don't call next when the page isn't visible

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

if (!document.hidden && $element.is(':visible') && $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 (this.\_element.querySelector(SELECTOR\_NEXT\_PREV)) {

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.\_updateInterval();

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

}

};

\_proto.to = function to(index) {

var \_this = this;

this.\_activeElement = this.\_element.querySelector(SELECTOR\_ACTIVE\_ITEM);

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

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

return;

}

if (this.\_isSliding) {

$\_\_default["default"](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() {

$\_\_default["default"](this.\_element).off(EVENT\_KEY$8);

$\_\_default["default"].removeData(this.\_element, DATA\_KEY$8);

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$7, config);

Util.typeCheckConfig(NAME$8, config, DefaultType$7);

return config;

};

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

var absDeltax = Math.abs(this.touchDeltaX);

if (absDeltax <= SWIPE\_THRESHOLD) {

return;

}

var direction = absDeltax / this.touchDeltaX;

this.touchDeltaX = 0; // swipe left

if (direction > 0) {

this.prev();

} // swipe right

if (direction < 0) {

this.next();

}

};

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

var \_this2 = this;

if (this.\_config.keyboard) {

$\_\_default["default"](this.\_element).on(EVENT\_KEYDOWN, function (event) {

return \_this2.\_keydown(event);

});

}

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

$\_\_default["default"](this.\_element).on(EVENT\_MOUSEENTER, function (event) {

return \_this2.pause(event);

}).on(EVENT\_MOUSELEAVE, function (event) {

return \_this2.cycle(event);

});

}

if (this.\_config.touch) {

this.\_addTouchEventListeners();

}

};

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

var \_this3 = this;

if (!this.\_touchSupported) {

return;

}

var start = function start(event) {

if (\_this3.\_pointerEvent && PointerType[event.originalEvent.pointerType.toUpperCase()]) {

\_this3.touchStartX = event.originalEvent.clientX;

} else if (!\_this3.\_pointerEvent) {

\_this3.touchStartX = event.originalEvent.touches[0].clientX;

}

};

var move = function move(event) {

// ensure swiping with one touch and not pinching

\_this3.touchDeltaX = event.originalEvent.touches && event.originalEvent.touches.length > 1 ? 0 : event.originalEvent.touches[0].clientX - \_this3.touchStartX;

};

var end = function end(event) {

if (\_this3.\_pointerEvent && PointerType[event.originalEvent.pointerType.toUpperCase()]) {

\_this3.touchDeltaX = event.originalEvent.clientX - \_this3.touchStartX;

}

\_this3.\_handleSwipe();

if (\_this3.\_config.pause === 'hover') {

// 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

\_this3.pause();

if (\_this3.touchTimeout) {

clearTimeout(\_this3.touchTimeout);

}

\_this3.touchTimeout = setTimeout(function (event) {

return \_this3.cycle(event);

}, TOUCHEVENT\_COMPAT\_WAIT + \_this3.\_config.interval);

}

};

$\_\_default["default"](this.\_element.querySelectorAll(SELECTOR\_ITEM\_IMG)).on(EVENT\_DRAG\_START, function (e) {

return e.preventDefault();

});

if (this.\_pointerEvent) {

$\_\_default["default"](this.\_element).on(EVENT\_POINTERDOWN, function (event) {

return start(event);

});

$\_\_default["default"](this.\_element).on(EVENT\_POINTERUP, function (event) {

return end(event);

});

this.\_element.classList.add(CLASS\_NAME\_POINTER\_EVENT);

} else {

$\_\_default["default"](this.\_element).on(EVENT\_TOUCHSTART, function (event) {

return start(event);

});

$\_\_default["default"](this.\_element).on(EVENT\_TOUCHMOVE, function (event) {

return move(event);

});

$\_\_default["default"](this.\_element).on(EVENT\_TOUCHEND, function (event) {

return end(event);

});

}

};

\_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;

}

};

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

this.\_items = element && element.parentNode ? [].slice.call(element.parentNode.querySelectorAll(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(this.\_element.querySelector(SELECTOR\_ACTIVE\_ITEM));

var slideEvent = $\_\_default["default"].Event(EVENT\_SLIDE, {

relatedTarget: relatedTarget,

direction: eventDirectionName,

from: fromIndex,

to: targetIndex

});

$\_\_default["default"](this.\_element).trigger(slideEvent);

return slideEvent;

};

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

if (this.\_indicatorsElement) {

var indicators = [].slice.call(this.\_indicatorsElement.querySelectorAll(SELECTOR\_ACTIVE$1));

$\_\_default["default"](indicators).removeClass(CLASS\_NAME\_ACTIVE$2);

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

if (nextIndicator) {

$\_\_default["default"](nextIndicator).addClass(CLASS\_NAME\_ACTIVE$2);

}

}

};

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

var element = this.\_activeElement || this.\_element.querySelector(SELECTOR\_ACTIVE\_ITEM);

if (!element) {

return;

}

var elementInterval = parseInt(element.getAttribute('data-interval'), 10);

if (elementInterval) {

this.\_config.defaultInterval = this.\_config.defaultInterval || this.\_config.interval;

this.\_config.interval = elementInterval;

} else {

this.\_config.interval = this.\_config.defaultInterval || this.\_config.interval;

}

};

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

var \_this4 = this;

var activeElement = this.\_element.querySelector(SELECTOR\_ACTIVE\_ITEM);

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 = CLASS\_NAME\_LEFT;

orderClassName = CLASS\_NAME\_NEXT;

eventDirectionName = DIRECTION\_LEFT;

} else {

directionalClassName = CLASS\_NAME\_RIGHT;

orderClassName = CLASS\_NAME\_PREV;

eventDirectionName = DIRECTION\_RIGHT;

}

if (nextElement && $\_\_default["default"](nextElement).hasClass(CLASS\_NAME\_ACTIVE$2)) {

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);

this.\_activeElement = nextElement;

var slidEvent = $\_\_default["default"].Event(EVENT\_SLID, {

relatedTarget: nextElement,

direction: eventDirectionName,

from: activeElementIndex,

to: nextElementIndex

});

if ($\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_SLIDE)) {

$\_\_default["default"](nextElement).addClass(orderClassName);

Util.reflow(nextElement);

$\_\_default["default"](activeElement).addClass(directionalClassName);

$\_\_default["default"](nextElement).addClass(directionalClassName);

var transitionDuration = Util.getTransitionDurationFromElement(activeElement);

$\_\_default["default"](activeElement).one(Util.TRANSITION\_END, function () {

$\_\_default["default"](nextElement).removeClass(directionalClassName + " " + orderClassName).addClass(CLASS\_NAME\_ACTIVE$2);

$\_\_default["default"](activeElement).removeClass(CLASS\_NAME\_ACTIVE$2 + " " + orderClassName + " " + directionalClassName);

\_this4.\_isSliding = false;

setTimeout(function () {

return $\_\_default["default"](\_this4.\_element).trigger(slidEvent);

}, 0);

}).emulateTransitionEnd(transitionDuration);

} else {

$\_\_default["default"](activeElement).removeClass(CLASS\_NAME\_ACTIVE$2);

$\_\_default["default"](nextElement).addClass(CLASS\_NAME\_ACTIVE$2);

this.\_isSliding = false;

$\_\_default["default"](this.\_element).trigger(slidEvent);

}

if (isCycling) {

this.cycle();

}

} // Static

;

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

return this.each(function () {

var data = $\_\_default["default"](this).data(DATA\_KEY$8);

var \_config = \_extends({}, Default$7, $\_\_default["default"](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);

$\_\_default["default"](this).data(DATA\_KEY$8, 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 && \_config.ride) {

data.pause();

data.cycle();

}

});

};

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

var selector = Util.getSelectorFromElement(this);

if (!selector) {

return;

}

var target = $\_\_default["default"](selector)[0];

if (!target || !$\_\_default["default"](target).hasClass(CLASS\_NAME\_CAROUSEL)) {

return;

}

var config = \_extends({}, $\_\_default["default"](target).data(), $\_\_default["default"](this).data());

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

if (slideIndex) {

config.interval = false;

}

Carousel.\_jQueryInterface.call($\_\_default["default"](target), config);

if (slideIndex) {

$\_\_default["default"](target).data(DATA\_KEY$8).to(slideIndex);

}

event.preventDefault();

};

\_createClass(Carousel, null, [{

key: "VERSION",

get: function get() {

return VERSION$8;

}

}, {

key: "Default",

get: function get() {

return Default$7;

}

}]);

return Carousel;

}();

/\*\*

\* Data API implementation

\*/

$\_\_default["default"](document).on(EVENT\_CLICK\_DATA\_API$4, SELECTOR\_DATA\_SLIDE, Carousel.\_dataApiClickHandler);

$\_\_default["default"](window).on(EVENT\_LOAD\_DATA\_API$1, function () {

var carousels = [].slice.call(document.querySelectorAll(SELECTOR\_DATA\_RIDE));

for (var i = 0, len = carousels.length; i < len; i++) {

var $carousel = $\_\_default["default"](carousels[i]);

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

}

});

/\*\*

\* jQuery

\*/

$\_\_default["default"].fn[NAME$8] = Carousel.\_jQueryInterface;

$\_\_default["default"].fn[NAME$8].Constructor = Carousel;

$\_\_default["default"].fn[NAME$8].noConflict = function () {

$\_\_default["default"].fn[NAME$8] = JQUERY\_NO\_CONFLICT$8;

return Carousel.\_jQueryInterface;

};

/\*\*

\* Constants

\*/

var NAME$7 = 'collapse';

var VERSION$7 = '4.6.1';

var DATA\_KEY$7 = 'bs.collapse';

var EVENT\_KEY$7 = "." + DATA\_KEY$7;

var DATA\_API\_KEY$4 = '.data-api';

var JQUERY\_NO\_CONFLICT$7 = $\_\_default["default"].fn[NAME$7];

var CLASS\_NAME\_SHOW$6 = 'show';

var CLASS\_NAME\_COLLAPSE = 'collapse';

var CLASS\_NAME\_COLLAPSING = 'collapsing';

var CLASS\_NAME\_COLLAPSED = 'collapsed';

var DIMENSION\_WIDTH = 'width';

var DIMENSION\_HEIGHT = 'height';

var EVENT\_SHOW$4 = "show" + EVENT\_KEY$7;

var EVENT\_SHOWN$4 = "shown" + EVENT\_KEY$7;

var EVENT\_HIDE$4 = "hide" + EVENT\_KEY$7;

var EVENT\_HIDDEN$4 = "hidden" + EVENT\_KEY$7;

var EVENT\_CLICK\_DATA\_API$3 = "click" + EVENT\_KEY$7 + DATA\_API\_KEY$4;

var SELECTOR\_ACTIVES = '.show, .collapsing';

var SELECTOR\_DATA\_TOGGLE$3 = '[data-toggle="collapse"]';

var Default$6 = {

toggle: true,

parent: ''

};

var DefaultType$6 = {

toggle: 'boolean',

parent: '(string|element)'

};

/\*\*

\* Class definition

\*/

var Collapse = /\*#\_\_PURE\_\_\*/function () {

function Collapse(element, config) {

this.\_isTransitioning = false;

this.\_element = element;

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

this.\_triggerArray = [].slice.call(document.querySelectorAll("[data-toggle=\"collapse\"][href=\"#" + element.id + "\"]," + ("[data-toggle=\"collapse\"][data-target=\"#" + element.id + "\"]")));

var toggleList = [].slice.call(document.querySelectorAll(SELECTOR\_DATA\_TOGGLE$3));

for (var i = 0, len = toggleList.length; i < len; i++) {

var elem = toggleList[i];

var selector = Util.getSelectorFromElement(elem);

var filterElement = [].slice.call(document.querySelectorAll(selector)).filter(function (foundElem) {

return foundElem === element;

});

if (selector !== null && filterElement.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 ($\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_SHOW$6)) {

this.hide();

} else {

this.show();

}

};

\_proto.show = function show() {

var \_this = this;

if (this.\_isTransitioning || $\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_SHOW$6)) {

return;

}

var actives;

var activesData;

if (this.\_parent) {

actives = [].slice.call(this.\_parent.querySelectorAll(SELECTOR\_ACTIVES)).filter(function (elem) {

if (typeof \_this.\_config.parent === 'string') {

return elem.getAttribute('data-parent') === \_this.\_config.parent;

}

return elem.classList.contains(CLASS\_NAME\_COLLAPSE);

});

if (actives.length === 0) {

actives = null;

}

}

if (actives) {

activesData = $\_\_default["default"](actives).not(this.\_selector).data(DATA\_KEY$7);

if (activesData && activesData.\_isTransitioning) {

return;

}

}

var startEvent = $\_\_default["default"].Event(EVENT\_SHOW$4);

$\_\_default["default"](this.\_element).trigger(startEvent);

if (startEvent.isDefaultPrevented()) {

return;

}

if (actives) {

Collapse.\_jQueryInterface.call($\_\_default["default"](actives).not(this.\_selector), 'hide');

if (!activesData) {

$\_\_default["default"](actives).data(DATA\_KEY$7, null);

}

}

var dimension = this.\_getDimension();

$\_\_default["default"](this.\_element).removeClass(CLASS\_NAME\_COLLAPSE).addClass(CLASS\_NAME\_COLLAPSING);

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

if (this.\_triggerArray.length) {

$\_\_default["default"](this.\_triggerArray).removeClass(CLASS\_NAME\_COLLAPSED).attr('aria-expanded', true);

}

this.setTransitioning(true);

var complete = function complete() {

$\_\_default["default"](\_this.\_element).removeClass(CLASS\_NAME\_COLLAPSING).addClass(CLASS\_NAME\_COLLAPSE + " " + CLASS\_NAME\_SHOW$6);

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

\_this.setTransitioning(false);

$\_\_default["default"](\_this.\_element).trigger(EVENT\_SHOWN$4);

};

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

var scrollSize = "scroll" + capitalizedDimension;

var transitionDuration = Util.getTransitionDurationFromElement(this.\_element);

$\_\_default["default"](this.\_element).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(transitionDuration);

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

};

\_proto.hide = function hide() {

var \_this2 = this;

if (this.\_isTransitioning || !$\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_SHOW$6)) {

return;

}

var startEvent = $\_\_default["default"].Event(EVENT\_HIDE$4);

$\_\_default["default"](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);

$\_\_default["default"](this.\_element).addClass(CLASS\_NAME\_COLLAPSING).removeClass(CLASS\_NAME\_COLLAPSE + " " + CLASS\_NAME\_SHOW$6);

var triggerArrayLength = this.\_triggerArray.length;

if (triggerArrayLength > 0) {

for (var i = 0; i < triggerArrayLength; i++) {

var trigger = this.\_triggerArray[i];

var selector = Util.getSelectorFromElement(trigger);

if (selector !== null) {

var $elem = $\_\_default["default"]([].slice.call(document.querySelectorAll(selector)));

if (!$elem.hasClass(CLASS\_NAME\_SHOW$6)) {

$\_\_default["default"](trigger).addClass(CLASS\_NAME\_COLLAPSED).attr('aria-expanded', false);

}

}

}

}

this.setTransitioning(true);

var complete = function complete() {

\_this2.setTransitioning(false);

$\_\_default["default"](\_this2.\_element).removeClass(CLASS\_NAME\_COLLAPSING).addClass(CLASS\_NAME\_COLLAPSE).trigger(EVENT\_HIDDEN$4);

};

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

var transitionDuration = Util.getTransitionDurationFromElement(this.\_element);

$\_\_default["default"](this.\_element).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(transitionDuration);

};

\_proto.setTransitioning = function setTransitioning(isTransitioning) {

this.\_isTransitioning = isTransitioning;

};

\_proto.dispose = function dispose() {

$\_\_default["default"].removeData(this.\_element, DATA\_KEY$7);

this.\_config = null;

this.\_parent = null;

this.\_element = null;

this.\_triggerArray = null;

this.\_isTransitioning = null;

} // Private

;

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

config = \_extends({}, Default$6, config);

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

Util.typeCheckConfig(NAME$7, config, DefaultType$6);

return config;

};

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

var hasWidth = $\_\_default["default"](this.\_element).hasClass(DIMENSION\_WIDTH);

return hasWidth ? DIMENSION\_WIDTH : DIMENSION\_HEIGHT;

};

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

var \_this3 = this;

var parent;

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 = document.querySelector(this.\_config.parent);

}

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

var children = [].slice.call(parent.querySelectorAll(selector));

$\_\_default["default"](children).each(function (i, element) {

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

});

return parent;

};

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

var isOpen = $\_\_default["default"](element).hasClass(CLASS\_NAME\_SHOW$6);

if (triggerArray.length) {

$\_\_default["default"](triggerArray).toggleClass(CLASS\_NAME\_COLLAPSED, !isOpen).attr('aria-expanded', isOpen);

}

} // Static

;

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

var selector = Util.getSelectorFromElement(element);

return selector ? document.querySelector(selector) : null;

};

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

return this.each(function () {

var $element = $\_\_default["default"](this);

var data = $element.data(DATA\_KEY$7);

var \_config = \_extends({}, Default$6, $element.data(), typeof config === 'object' && config ? config : {});

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

\_config.toggle = false;

}

if (!data) {

data = new Collapse(this, \_config);

$element.data(DATA\_KEY$7, 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$7;

}

}, {

key: "Default",

get: function get() {

return Default$6;

}

}]);

return Collapse;

}();

/\*\*

\* Data API implementation

\*/

$\_\_default["default"](document).on(EVENT\_CLICK\_DATA\_API$3, SELECTOR\_DATA\_TOGGLE$3, 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 = $\_\_default["default"](this);

var selector = Util.getSelectorFromElement(this);

var selectors = [].slice.call(document.querySelectorAll(selector));

$\_\_default["default"](selectors).each(function () {

var $target = $\_\_default["default"](this);

var data = $target.data(DATA\_KEY$7);

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

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

});

});

/\*\*

\* jQuery

\*/

$\_\_default["default"].fn[NAME$7] = Collapse.\_jQueryInterface;

$\_\_default["default"].fn[NAME$7].Constructor = Collapse;

$\_\_default["default"].fn[NAME$7].noConflict = function () {

$\_\_default["default"].fn[NAME$7] = JQUERY\_NO\_CONFLICT$7;

return Collapse.\_jQueryInterface;

};

/\*\*

\* Constants

\*/

var NAME$6 = 'dropdown';

var VERSION$6 = '4.6.1';

var DATA\_KEY$6 = 'bs.dropdown';

var EVENT\_KEY$6 = "." + DATA\_KEY$6;

var DATA\_API\_KEY$3 = '.data-api';

var JQUERY\_NO\_CONFLICT$6 = $\_\_default["default"].fn[NAME$6];

var ESCAPE\_KEYCODE$1 = 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$1);

var CLASS\_NAME\_DISABLED$1 = 'disabled';

var CLASS\_NAME\_SHOW$5 = 'show';

var CLASS\_NAME\_DROPUP = 'dropup';

var CLASS\_NAME\_DROPRIGHT = 'dropright';

var CLASS\_NAME\_DROPLEFT = 'dropleft';

var CLASS\_NAME\_MENURIGHT = 'dropdown-menu-right';

var CLASS\_NAME\_POSITION\_STATIC = 'position-static';

var EVENT\_HIDE$3 = "hide" + EVENT\_KEY$6;

var EVENT\_HIDDEN$3 = "hidden" + EVENT\_KEY$6;

var EVENT\_SHOW$3 = "show" + EVENT\_KEY$6;

var EVENT\_SHOWN$3 = "shown" + EVENT\_KEY$6;

var EVENT\_CLICK = "click" + EVENT\_KEY$6;

var EVENT\_CLICK\_DATA\_API$2 = "click" + EVENT\_KEY$6 + DATA\_API\_KEY$3;

var EVENT\_KEYDOWN\_DATA\_API = "keydown" + EVENT\_KEY$6 + DATA\_API\_KEY$3;

var EVENT\_KEYUP\_DATA\_API = "keyup" + EVENT\_KEY$6 + DATA\_API\_KEY$3;

var SELECTOR\_DATA\_TOGGLE$2 = '[data-toggle="dropdown"]';

var SELECTOR\_FORM\_CHILD = '.dropdown form';

var SELECTOR\_MENU = '.dropdown-menu';

var SELECTOR\_NAVBAR\_NAV = '.navbar-nav';

var SELECTOR\_VISIBLE\_ITEMS = '.dropdown-menu .dropdown-item:not(.disabled):not(:disabled)';

var PLACEMENT\_TOP = 'top-start';

var PLACEMENT\_TOPEND = 'top-end';

var PLACEMENT\_BOTTOM = 'bottom-start';

var PLACEMENT\_BOTTOMEND = 'bottom-end';

var PLACEMENT\_RIGHT = 'right-start';

var PLACEMENT\_LEFT = 'left-start';

var Default$5 = {

offset: 0,

flip: true,

boundary: 'scrollParent',

reference: 'toggle',

display: 'dynamic',

popperConfig: null

};

var DefaultType$5 = {

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

flip: 'boolean',

boundary: '(string|element)',

reference: '(string|element)',

display: 'string',

popperConfig: '(null|object)'

};

/\*\*

\* 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 || $\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_DISABLED$1)) {

return;

}

var isActive = $\_\_default["default"](this.\_menu).hasClass(CLASS\_NAME\_SHOW$5);

Dropdown.\_clearMenus();

if (isActive) {

return;

}

this.show(true);

};

\_proto.show = function show(usePopper) {

if (usePopper === void 0) {

usePopper = false;

}

if (this.\_element.disabled || $\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_DISABLED$1) || $\_\_default["default"](this.\_menu).hasClass(CLASS\_NAME\_SHOW$5)) {

return;

}

var relatedTarget = {

relatedTarget: this.\_element

};

var showEvent = $\_\_default["default"].Event(EVENT\_SHOW$3, relatedTarget);

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

$\_\_default["default"](parent).trigger(showEvent);

if (showEvent.isDefaultPrevented()) {

return;

} // Totally disable Popper for Dropdowns in Navbar

if (!this.\_inNavbar && usePopper) {

// Check for Popper dependency

if (typeof Popper\_\_default["default"] === 'undefined') {

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

}

var referenceElement = this.\_element;

if (this.\_config.reference === 'parent') {

referenceElement = parent;

} else if (Util.isElement(this.\_config.reference)) {

referenceElement = this.\_config.reference; // Check if it's jQuery element

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

referenceElement = this.\_config.reference[0];

}

} // 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') {

$\_\_default["default"](parent).addClass(CLASS\_NAME\_POSITION\_STATIC);

}

this.\_popper = new Popper\_\_default["default"](referenceElement, 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 && $\_\_default["default"](parent).closest(SELECTOR\_NAVBAR\_NAV).length === 0) {

$\_\_default["default"](document.body).children().on('mouseover', null, $\_\_default["default"].noop);

}

this.\_element.focus();

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

$\_\_default["default"](this.\_menu).toggleClass(CLASS\_NAME\_SHOW$5);

$\_\_default["default"](parent).toggleClass(CLASS\_NAME\_SHOW$5).trigger($\_\_default["default"].Event(EVENT\_SHOWN$3, relatedTarget));

};

\_proto.hide = function hide() {

if (this.\_element.disabled || $\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_DISABLED$1) || !$\_\_default["default"](this.\_menu).hasClass(CLASS\_NAME\_SHOW$5)) {

return;

}

var relatedTarget = {

relatedTarget: this.\_element

};

var hideEvent = $\_\_default["default"].Event(EVENT\_HIDE$3, relatedTarget);

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

$\_\_default["default"](parent).trigger(hideEvent);

if (hideEvent.isDefaultPrevented()) {

return;

}

if (this.\_popper) {

this.\_popper.destroy();

}

$\_\_default["default"](this.\_menu).toggleClass(CLASS\_NAME\_SHOW$5);

$\_\_default["default"](parent).toggleClass(CLASS\_NAME\_SHOW$5).trigger($\_\_default["default"].Event(EVENT\_HIDDEN$3, relatedTarget));

};

\_proto.dispose = function dispose() {

$\_\_default["default"].removeData(this.\_element, DATA\_KEY$6);

$\_\_default["default"](this.\_element).off(EVENT\_KEY$6);

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;

$\_\_default["default"](this.\_element).on(EVENT\_CLICK, function (event) {

event.preventDefault();

event.stopPropagation();

\_this.toggle();

});

};

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

config = \_extends({}, this.constructor.Default, $\_\_default["default"](this.\_element).data(), config);

Util.typeCheckConfig(NAME$6, config, this.constructor.DefaultType);

return config;

};

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

if (!this.\_menu) {

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

if (parent) {

this.\_menu = parent.querySelector(SELECTOR\_MENU);

}

}

return this.\_menu;

};

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

var $parentDropdown = $\_\_default["default"](this.\_element.parentNode);

var placement = PLACEMENT\_BOTTOM; // Handle dropup

if ($parentDropdown.hasClass(CLASS\_NAME\_DROPUP)) {

placement = $\_\_default["default"](this.\_menu).hasClass(CLASS\_NAME\_MENURIGHT) ? PLACEMENT\_TOPEND : PLACEMENT\_TOP;

} else if ($parentDropdown.hasClass(CLASS\_NAME\_DROPRIGHT)) {

placement = PLACEMENT\_RIGHT;

} else if ($parentDropdown.hasClass(CLASS\_NAME\_DROPLEFT)) {

placement = PLACEMENT\_LEFT;

} else if ($\_\_default["default"](this.\_menu).hasClass(CLASS\_NAME\_MENURIGHT)) {

placement = PLACEMENT\_BOTTOMEND;

}

return placement;

};

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

return $\_\_default["default"](this.\_element).closest('.navbar').length > 0;

};

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

var \_this2 = this;

var offset = {};

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

offset.fn = function (data) {

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

return data;

};

} else {

offset.offset = this.\_config.offset;

}

return offset;

};

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

var popperConfig = {

placement: this.\_getPlacement(),

modifiers: {

offset: this.\_getOffset(),

flip: {

enabled: this.\_config.flip

},

preventOverflow: {

boundariesElement: this.\_config.boundary

}

}

}; // Disable Popper if we have a static display

if (this.\_config.display === 'static') {

popperConfig.modifiers.applyStyle = {

enabled: false

};

}

return \_extends({}, popperConfig, this.\_config.popperConfig);

} // Static

;

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

return this.each(function () {

var data = $\_\_default["default"](this).data(DATA\_KEY$6);

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

if (!data) {

data = new Dropdown(this, \_config);

$\_\_default["default"](this).data(DATA\_KEY$6, 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 = [].slice.call(document.querySelectorAll(SELECTOR\_DATA\_TOGGLE$2));

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

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

var context = $\_\_default["default"](toggles[i]).data(DATA\_KEY$6);

var relatedTarget = {

relatedTarget: toggles[i]

};

if (event && event.type === 'click') {

relatedTarget.clickEvent = event;

}

if (!context) {

continue;

}

var dropdownMenu = context.\_menu;

if (!$\_\_default["default"](parent).hasClass(CLASS\_NAME\_SHOW$5)) {

continue;

}

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

continue;

}

var hideEvent = $\_\_default["default"].Event(EVENT\_HIDE$3, relatedTarget);

$\_\_default["default"](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) {

$\_\_default["default"](document.body).children().off('mouseover', null, $\_\_default["default"].noop);

}

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

if (context.\_popper) {

context.\_popper.destroy();

}

$\_\_default["default"](dropdownMenu).removeClass(CLASS\_NAME\_SHOW$5);

$\_\_default["default"](parent).removeClass(CLASS\_NAME\_SHOW$5).trigger($\_\_default["default"].Event(EVENT\_HIDDEN$3, relatedTarget));

}

};

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

var parent;

var selector = Util.getSelectorFromElement(element);

if (selector) {

parent = document.querySelector(selector);

}

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$1 && (event.which !== ARROW\_DOWN\_KEYCODE && event.which !== ARROW\_UP\_KEYCODE || $\_\_default["default"](event.target).closest(SELECTOR\_MENU).length) : !REGEXP\_KEYDOWN.test(event.which)) {

return;

}

if (this.disabled || $\_\_default["default"](this).hasClass(CLASS\_NAME\_DISABLED$1)) {

return;

}

var parent = Dropdown.\_getParentFromElement(this);

var isActive = $\_\_default["default"](parent).hasClass(CLASS\_NAME\_SHOW$5);

if (!isActive && event.which === ESCAPE\_KEYCODE$1) {

return;

}

event.preventDefault();

event.stopPropagation();

if (!isActive || event.which === ESCAPE\_KEYCODE$1 || event.which === SPACE\_KEYCODE) {

if (event.which === ESCAPE\_KEYCODE$1) {

$\_\_default["default"](parent.querySelector(SELECTOR\_DATA\_TOGGLE$2)).trigger('focus');

}

$\_\_default["default"](this).trigger('click');

return;

}

var items = [].slice.call(parent.querySelectorAll(SELECTOR\_VISIBLE\_ITEMS)).filter(function (item) {

return $\_\_default["default"](item).is(':visible');

});

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$6;

}

}, {

key: "Default",

get: function get() {

return Default$5;

}

}, {

key: "DefaultType",

get: function get() {

return DefaultType$5;

}

}]);

return Dropdown;

}();

/\*\*

\* Data API implementation

\*/

$\_\_default["default"](document).on(EVENT\_KEYDOWN\_DATA\_API, SELECTOR\_DATA\_TOGGLE$2, Dropdown.\_dataApiKeydownHandler).on(EVENT\_KEYDOWN\_DATA\_API, SELECTOR\_MENU, Dropdown.\_dataApiKeydownHandler).on(EVENT\_CLICK\_DATA\_API$2 + " " + EVENT\_KEYUP\_DATA\_API, Dropdown.\_clearMenus).on(EVENT\_CLICK\_DATA\_API$2, SELECTOR\_DATA\_TOGGLE$2, function (event) {

event.preventDefault();

event.stopPropagation();

Dropdown.\_jQueryInterface.call($\_\_default["default"](this), 'toggle');

}).on(EVENT\_CLICK\_DATA\_API$2, SELECTOR\_FORM\_CHILD, function (e) {

e.stopPropagation();

});

/\*\*

\* jQuery

\*/

$\_\_default["default"].fn[NAME$6] = Dropdown.\_jQueryInterface;

$\_\_default["default"].fn[NAME$6].Constructor = Dropdown;

$\_\_default["default"].fn[NAME$6].noConflict = function () {

$\_\_default["default"].fn[NAME$6] = JQUERY\_NO\_CONFLICT$6;

return Dropdown.\_jQueryInterface;

};

/\*\*

\* Constants

\*/

var NAME$5 = 'modal';

var VERSION$5 = '4.6.1';

var DATA\_KEY$5 = 'bs.modal';

var EVENT\_KEY$5 = "." + DATA\_KEY$5;

var DATA\_API\_KEY$2 = '.data-api';

var JQUERY\_NO\_CONFLICT$5 = $\_\_default["default"].fn[NAME$5];

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

var CLASS\_NAME\_SCROLLABLE = 'modal-dialog-scrollable';

var CLASS\_NAME\_SCROLLBAR\_MEASURER = 'modal-scrollbar-measure';

var CLASS\_NAME\_BACKDROP = 'modal-backdrop';

var CLASS\_NAME\_OPEN = 'modal-open';

var CLASS\_NAME\_FADE$4 = 'fade';

var CLASS\_NAME\_SHOW$4 = 'show';

var CLASS\_NAME\_STATIC = 'modal-static';

var EVENT\_HIDE$2 = "hide" + EVENT\_KEY$5;

var EVENT\_HIDE\_PREVENTED = "hidePrevented" + EVENT\_KEY$5;

var EVENT\_HIDDEN$2 = "hidden" + EVENT\_KEY$5;

var EVENT\_SHOW$2 = "show" + EVENT\_KEY$5;

var EVENT\_SHOWN$2 = "shown" + EVENT\_KEY$5;

var EVENT\_FOCUSIN = "focusin" + EVENT\_KEY$5;

var EVENT\_RESIZE = "resize" + EVENT\_KEY$5;

var EVENT\_CLICK\_DISMISS$1 = "click.dismiss" + EVENT\_KEY$5;

var EVENT\_KEYDOWN\_DISMISS = "keydown.dismiss" + EVENT\_KEY$5;

var EVENT\_MOUSEUP\_DISMISS = "mouseup.dismiss" + EVENT\_KEY$5;

var EVENT\_MOUSEDOWN\_DISMISS = "mousedown.dismiss" + EVENT\_KEY$5;

var EVENT\_CLICK\_DATA\_API$1 = "click" + EVENT\_KEY$5 + DATA\_API\_KEY$2;

var SELECTOR\_DIALOG = '.modal-dialog';

var SELECTOR\_MODAL\_BODY = '.modal-body';

var SELECTOR\_DATA\_TOGGLE$1 = '[data-toggle="modal"]';

var SELECTOR\_DATA\_DISMISS$1 = '[data-dismiss="modal"]';

var SELECTOR\_FIXED\_CONTENT = '.fixed-top, .fixed-bottom, .is-fixed, .sticky-top';

var SELECTOR\_STICKY\_CONTENT = '.sticky-top';

var Default$4 = {

backdrop: true,

keyboard: true,

focus: true,

show: true

};

var DefaultType$4 = {

backdrop: '(boolean|string)',

keyboard: 'boolean',

focus: 'boolean',

show: 'boolean'

};

/\*\*

\* Class definition

\*/

var Modal = /\*#\_\_PURE\_\_\*/function () {

function Modal(element, config) {

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

this.\_element = element;

this.\_dialog = element.querySelector(SELECTOR\_DIALOG);

this.\_backdrop = null;

this.\_isShown = false;

this.\_isBodyOverflowing = false;

this.\_ignoreBackdropClick = false;

this.\_isTransitioning = false;

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.\_isShown || this.\_isTransitioning) {

return;

}

var showEvent = $\_\_default["default"].Event(EVENT\_SHOW$2, {

relatedTarget: relatedTarget

});

$\_\_default["default"](this.\_element).trigger(showEvent);

if (showEvent.isDefaultPrevented()) {

return;

}

this.\_isShown = true;

if ($\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_FADE$4)) {

this.\_isTransitioning = true;

}

this.\_checkScrollbar();

this.\_setScrollbar();

this.\_adjustDialog();

this.\_setEscapeEvent();

this.\_setResizeEvent();

$\_\_default["default"](this.\_element).on(EVENT\_CLICK\_DISMISS$1, SELECTOR\_DATA\_DISMISS$1, function (event) {

return \_this.hide(event);

});

$\_\_default["default"](this.\_dialog).on(EVENT\_MOUSEDOWN\_DISMISS, function () {

$\_\_default["default"](\_this.\_element).one(EVENT\_MOUSEUP\_DISMISS, function (event) {

if ($\_\_default["default"](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.\_isShown || this.\_isTransitioning) {

return;

}

var hideEvent = $\_\_default["default"].Event(EVENT\_HIDE$2);

$\_\_default["default"](this.\_element).trigger(hideEvent);

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

return;

}

this.\_isShown = false;

var transition = $\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_FADE$4);

if (transition) {

this.\_isTransitioning = true;

}

this.\_setEscapeEvent();

this.\_setResizeEvent();

$\_\_default["default"](document).off(EVENT\_FOCUSIN);

$\_\_default["default"](this.\_element).removeClass(CLASS\_NAME\_SHOW$4);

$\_\_default["default"](this.\_element).off(EVENT\_CLICK\_DISMISS$1);

$\_\_default["default"](this.\_dialog).off(EVENT\_MOUSEDOWN\_DISMISS);

if (transition) {

var transitionDuration = Util.getTransitionDurationFromElement(this.\_element);

$\_\_default["default"](this.\_element).one(Util.TRANSITION\_END, function (event) {

return \_this2.\_hideModal(event);

}).emulateTransitionEnd(transitionDuration);

} else {

this.\_hideModal();

}

};

\_proto.dispose = function dispose() {

[window, this.\_element, this.\_dialog].forEach(function (htmlElement) {

return $\_\_default["default"](htmlElement).off(EVENT\_KEY$5);

});

/\*\*

\* `document` has 2 events `EVENT\_FOCUSIN` and `EVENT\_CLICK\_DATA\_API`

\* Do not move `document` in `htmlElements` array

\* It will remove `EVENT\_CLICK\_DATA\_API` event that should remain

\*/

$\_\_default["default"](document).off(EVENT\_FOCUSIN);

$\_\_default["default"].removeData(this.\_element, DATA\_KEY$5);

this.\_config = null;

this.\_element = null;

this.\_dialog = null;

this.\_backdrop = null;

this.\_isShown = null;

this.\_isBodyOverflowing = null;

this.\_ignoreBackdropClick = null;

this.\_isTransitioning = null;

this.\_scrollbarWidth = null;

};

\_proto.handleUpdate = function handleUpdate() {

this.\_adjustDialog();

} // Private

;

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

config = \_extends({}, Default$4, config);

Util.typeCheckConfig(NAME$5, config, DefaultType$4);

return config;

};

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

var \_this3 = this;

var hideEventPrevented = $\_\_default["default"].Event(EVENT\_HIDE\_PREVENTED);

$\_\_default["default"](this.\_element).trigger(hideEventPrevented);

if (hideEventPrevented.isDefaultPrevented()) {

return;

}

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

if (!isModalOverflowing) {

this.\_element.style.overflowY = 'hidden';

}

this.\_element.classList.add(CLASS\_NAME\_STATIC);

var modalTransitionDuration = Util.getTransitionDurationFromElement(this.\_dialog);

$\_\_default["default"](this.\_element).off(Util.TRANSITION\_END);

$\_\_default["default"](this.\_element).one(Util.TRANSITION\_END, function () {

\_this3.\_element.classList.remove(CLASS\_NAME\_STATIC);

if (!isModalOverflowing) {

$\_\_default["default"](\_this3.\_element).one(Util.TRANSITION\_END, function () {

\_this3.\_element.style.overflowY = '';

}).emulateTransitionEnd(\_this3.\_element, modalTransitionDuration);

}

}).emulateTransitionEnd(modalTransitionDuration);

this.\_element.focus();

};

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

var \_this4 = this;

var transition = $\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_FADE$4);

var modalBody = this.\_dialog ? this.\_dialog.querySelector(SELECTOR\_MODAL\_BODY) : null;

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.setAttribute('aria-modal', true);

this.\_element.setAttribute('role', 'dialog');

if ($\_\_default["default"](this.\_dialog).hasClass(CLASS\_NAME\_SCROLLABLE) && modalBody) {

modalBody.scrollTop = 0;

} else {

this.\_element.scrollTop = 0;

}

if (transition) {

Util.reflow(this.\_element);

}

$\_\_default["default"](this.\_element).addClass(CLASS\_NAME\_SHOW$4);

if (this.\_config.focus) {

this.\_enforceFocus();

}

var shownEvent = $\_\_default["default"].Event(EVENT\_SHOWN$2, {

relatedTarget: relatedTarget

});

var transitionComplete = function transitionComplete() {

if (\_this4.\_config.focus) {

\_this4.\_element.focus();

}

\_this4.\_isTransitioning = false;

$\_\_default["default"](\_this4.\_element).trigger(shownEvent);

};

if (transition) {

var transitionDuration = Util.getTransitionDurationFromElement(this.\_dialog);

$\_\_default["default"](this.\_dialog).one(Util.TRANSITION\_END, transitionComplete).emulateTransitionEnd(transitionDuration);

} else {

transitionComplete();

}

};

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

var \_this5 = this;

$\_\_default["default"](document).off(EVENT\_FOCUSIN) // Guard against infinite focus loop

.on(EVENT\_FOCUSIN, function (event) {

if (document !== event.target && \_this5.\_element !== event.target && $\_\_default["default"](\_this5.\_element).has(event.target).length === 0) {

\_this5.\_element.focus();

}

});

};

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

var \_this6 = this;

if (this.\_isShown) {

$\_\_default["default"](this.\_element).on(EVENT\_KEYDOWN\_DISMISS, function (event) {

if (\_this6.\_config.keyboard && event.which === ESCAPE\_KEYCODE) {

event.preventDefault();

\_this6.hide();

} else if (!\_this6.\_config.keyboard && event.which === ESCAPE\_KEYCODE) {

\_this6.\_triggerBackdropTransition();

}

});

} else if (!this.\_isShown) {

$\_\_default["default"](this.\_element).off(EVENT\_KEYDOWN\_DISMISS);

}

};

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

var \_this7 = this;

if (this.\_isShown) {

$\_\_default["default"](window).on(EVENT\_RESIZE, function (event) {

return \_this7.handleUpdate(event);

});

} else {

$\_\_default["default"](window).off(EVENT\_RESIZE);

}

};

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

var \_this8 = this;

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

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

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

this.\_element.removeAttribute('role');

this.\_isTransitioning = false;

this.\_showBackdrop(function () {

$\_\_default["default"](document.body).removeClass(CLASS\_NAME\_OPEN);

\_this8.\_resetAdjustments();

\_this8.\_resetScrollbar();

$\_\_default["default"](\_this8.\_element).trigger(EVENT\_HIDDEN$2);

});

};

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

if (this.\_backdrop) {

$\_\_default["default"](this.\_backdrop).remove();

this.\_backdrop = null;

}

};

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

var \_this9 = this;

var animate = $\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_FADE$4) ? CLASS\_NAME\_FADE$4 : '';

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

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

this.\_backdrop.className = CLASS\_NAME\_BACKDROP;

if (animate) {

this.\_backdrop.classList.add(animate);

}

$\_\_default["default"](this.\_backdrop).appendTo(document.body);

$\_\_default["default"](this.\_element).on(EVENT\_CLICK\_DISMISS$1, function (event) {

if (\_this9.\_ignoreBackdropClick) {

\_this9.\_ignoreBackdropClick = false;

return;

}

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

return;

}

if (\_this9.\_config.backdrop === 'static') {

\_this9.\_triggerBackdropTransition();

} else {

\_this9.hide();

}

});

if (animate) {

Util.reflow(this.\_backdrop);

}

$\_\_default["default"](this.\_backdrop).addClass(CLASS\_NAME\_SHOW$4);

if (!callback) {

return;

}

if (!animate) {

callback();

return;

}

var backdropTransitionDuration = Util.getTransitionDurationFromElement(this.\_backdrop);

$\_\_default["default"](this.\_backdrop).one(Util.TRANSITION\_END, callback).emulateTransitionEnd(backdropTransitionDuration);

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

$\_\_default["default"](this.\_backdrop).removeClass(CLASS\_NAME\_SHOW$4);

var callbackRemove = function callbackRemove() {

\_this9.\_removeBackdrop();

if (callback) {

callback();

}

};

if ($\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_FADE$4)) {

var \_backdropTransitionDuration = Util.getTransitionDurationFromElement(this.\_backdrop);

$\_\_default["default"](this.\_backdrop).one(Util.TRANSITION\_END, callbackRemove).emulateTransitionEnd(\_backdropTransitionDuration);

} 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 = Math.round(rect.left + rect.right) < window.innerWidth;

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

};

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

var \_this10 = 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

var fixedContent = [].slice.call(document.querySelectorAll(SELECTOR\_FIXED\_CONTENT));

var stickyContent = [].slice.call(document.querySelectorAll(SELECTOR\_STICKY\_CONTENT)); // Adjust fixed content padding

$\_\_default["default"](fixedContent).each(function (index, element) {

var actualPadding = element.style.paddingRight;

var calculatedPadding = $\_\_default["default"](element).css('padding-right');

$\_\_default["default"](element).data('padding-right', actualPadding).css('padding-right', parseFloat(calculatedPadding) + \_this10.\_scrollbarWidth + "px");

}); // Adjust sticky content margin

$\_\_default["default"](stickyContent).each(function (index, element) {

var actualMargin = element.style.marginRight;

var calculatedMargin = $\_\_default["default"](element).css('margin-right');

$\_\_default["default"](element).data('margin-right', actualMargin).css('margin-right', parseFloat(calculatedMargin) - \_this10.\_scrollbarWidth + "px");

}); // Adjust body padding

var actualPadding = document.body.style.paddingRight;

var calculatedPadding = $\_\_default["default"](document.body).css('padding-right');

$\_\_default["default"](document.body).data('padding-right', actualPadding).css('padding-right', parseFloat(calculatedPadding) + this.\_scrollbarWidth + "px");

}

$\_\_default["default"](document.body).addClass(CLASS\_NAME\_OPEN);

};

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

// Restore fixed content padding

var fixedContent = [].slice.call(document.querySelectorAll(SELECTOR\_FIXED\_CONTENT));

$\_\_default["default"](fixedContent).each(function (index, element) {

var padding = $\_\_default["default"](element).data('padding-right');

$\_\_default["default"](element).removeData('padding-right');

element.style.paddingRight = padding ? padding : '';

}); // Restore sticky content

var elements = [].slice.call(document.querySelectorAll("" + SELECTOR\_STICKY\_CONTENT));

$\_\_default["default"](elements).each(function (index, element) {

var margin = $\_\_default["default"](element).data('margin-right');

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

$\_\_default["default"](element).css('margin-right', margin).removeData('margin-right');

}

}); // Restore body padding

var padding = $\_\_default["default"](document.body).data('padding-right');

$\_\_default["default"](document.body).removeData('padding-right');

document.body.style.paddingRight = padding ? padding : '';

};

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

// thx d.walsh

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

scrollDiv.className = CLASS\_NAME\_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 = $\_\_default["default"](this).data(DATA\_KEY$5);

var \_config = \_extends({}, Default$4, $\_\_default["default"](this).data(), typeof config === 'object' && config ? config : {});

if (!data) {

data = new Modal(this, \_config);

$\_\_default["default"](this).data(DATA\_KEY$5, 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$5;

}

}, {

key: "Default",

get: function get() {

return Default$4;

}

}]);

return Modal;

}();

/\*\*

\* Data API implementation

\*/

$\_\_default["default"](document).on(EVENT\_CLICK\_DATA\_API$1, SELECTOR\_DATA\_TOGGLE$1, function (event) {

var \_this11 = this;

var target;

var selector = Util.getSelectorFromElement(this);

if (selector) {

target = document.querySelector(selector);

}

var config = $\_\_default["default"](target).data(DATA\_KEY$5) ? 'toggle' : \_extends({}, $\_\_default["default"](target).data(), $\_\_default["default"](this).data());

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

event.preventDefault();

}

var $target = $\_\_default["default"](target).one(EVENT\_SHOW$2, function (showEvent) {

if (showEvent.isDefaultPrevented()) {

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

return;

}

$target.one(EVENT\_HIDDEN$2, function () {

if ($\_\_default["default"](\_this11).is(':visible')) {

\_this11.focus();

}

});

});

Modal.\_jQueryInterface.call($\_\_default["default"](target), config, this);

});

/\*\*

\* jQuery

\*/

$\_\_default["default"].fn[NAME$5] = Modal.\_jQueryInterface;

$\_\_default["default"].fn[NAME$5].Constructor = Modal;

$\_\_default["default"].fn[NAME$5].noConflict = function () {

$\_\_default["default"].fn[NAME$5] = JQUERY\_NO\_CONFLICT$5;

return Modal.\_jQueryInterface;

};

/\*\*

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

\* Bootstrap (v4.6.1): tools/sanitizer.js

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

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

\*/

var uriAttrs = ['background', 'cite', 'href', 'itemtype', 'longdesc', 'poster', 'src', 'xlink:href'];

var ARIA\_ATTRIBUTE\_PATTERN = /^aria-[\w-]\*$/i;

var DefaultWhitelist = {

// Global attributes allowed on any supplied element below.

'\*': ['class', 'dir', 'id', 'lang', 'role', ARIA\_ATTRIBUTE\_PATTERN],

a: ['target', 'href', 'title', 'rel'],

area: [],

b: [],

br: [],

col: [],

code: [],

div: [],

em: [],

hr: [],

h1: [],

h2: [],

h3: [],

h4: [],

h5: [],

h6: [],

i: [],

img: ['src', 'srcset', 'alt', 'title', 'width', 'height'],

li: [],

ol: [],

p: [],

pre: [],

s: [],

small: [],

span: [],

sub: [],

sup: [],

strong: [],

u: [],

ul: []

};

/\*\*

\* A pattern that recognizes a commonly useful subset of URLs that are safe.

\*

\* Shoutout to Angular https://github.com/angular/angular/blob/12.2.x/packages/core/src/sanitization/url\_sanitizer.ts

\*/

var SAFE\_URL\_PATTERN = /^(?:(?:https?|mailto|ftp|tel|file|sms):|[^#&/:?]\*(?:[#/?]|$))/i;

/\*\*

\* A pattern that matches safe data URLs. Only matches image, video and audio types.

\*

\* Shoutout to Angular https://github.com/angular/angular/blob/12.2.x/packages/core/src/sanitization/url\_sanitizer.ts

\*/

var DATA\_URL\_PATTERN = /^data:(?:image\/(?:bmp|gif|jpeg|jpg|png|tiff|webp)|video\/(?:mpeg|mp4|ogg|webm)|audio\/(?:mp3|oga|ogg|opus));base64,[\d+/a-z]+=\*$/i;

function allowedAttribute(attr, allowedAttributeList) {

var attrName = attr.nodeName.toLowerCase();

if (allowedAttributeList.indexOf(attrName) !== -1) {

if (uriAttrs.indexOf(attrName) !== -1) {

return Boolean(SAFE\_URL\_PATTERN.test(attr.nodeValue) || DATA\_URL\_PATTERN.test(attr.nodeValue));

}

return true;

}

var regExp = allowedAttributeList.filter(function (attrRegex) {

return attrRegex instanceof RegExp;

}); // Check if a regular expression validates the attribute.

for (var i = 0, len = regExp.length; i < len; i++) {

if (regExp[i].test(attrName)) {

return true;

}

}

return false;

}

function sanitizeHtml(unsafeHtml, whiteList, sanitizeFn) {

if (unsafeHtml.length === 0) {

return unsafeHtml;

}

if (sanitizeFn && typeof sanitizeFn === 'function') {

return sanitizeFn(unsafeHtml);

}

var domParser = new window.DOMParser();

var createdDocument = domParser.parseFromString(unsafeHtml, 'text/html');

var whitelistKeys = Object.keys(whiteList);

var elements = [].slice.call(createdDocument.body.querySelectorAll('\*'));

var \_loop = function \_loop(i, len) {

var el = elements[i];

var elName = el.nodeName.toLowerCase();

if (whitelistKeys.indexOf(el.nodeName.toLowerCase()) === -1) {

el.parentNode.removeChild(el);

return "continue";

}

var attributeList = [].slice.call(el.attributes); // eslint-disable-next-line unicorn/prefer-spread

var whitelistedAttributes = [].concat(whiteList['\*'] || [], whiteList[elName] || []);

attributeList.forEach(function (attr) {

if (!allowedAttribute(attr, whitelistedAttributes)) {

el.removeAttribute(attr.nodeName);

}

});

};

for (var i = 0, len = elements.length; i < len; i++) {

var \_ret = \_loop(i);

if (\_ret === "continue") continue;

}

return createdDocument.body.innerHTML;

}

/\*\*

\* Constants

\*/

var NAME$4 = 'tooltip';

var VERSION$4 = '4.6.1';

var DATA\_KEY$4 = 'bs.tooltip';

var EVENT\_KEY$4 = "." + DATA\_KEY$4;

var JQUERY\_NO\_CONFLICT$4 = $\_\_default["default"].fn[NAME$4];

var CLASS\_PREFIX$1 = 'bs-tooltip';

var BSCLS\_PREFIX\_REGEX$1 = new RegExp("(^|\\s)" + CLASS\_PREFIX$1 + "\\S+", 'g');

var DISALLOWED\_ATTRIBUTES = ['sanitize', 'whiteList', 'sanitizeFn'];

var CLASS\_NAME\_FADE$3 = 'fade';

var CLASS\_NAME\_SHOW$3 = 'show';

var HOVER\_STATE\_SHOW = 'show';

var HOVER\_STATE\_OUT = 'out';

var SELECTOR\_TOOLTIP\_INNER = '.tooltip-inner';

var SELECTOR\_ARROW = '.arrow';

var TRIGGER\_HOVER = 'hover';

var TRIGGER\_FOCUS = 'focus';

var TRIGGER\_CLICK = 'click';

var TRIGGER\_MANUAL = 'manual';

var AttachmentMap = {

AUTO: 'auto',

TOP: 'top',

RIGHT: 'right',

BOTTOM: 'bottom',

LEFT: 'left'

};

var Default$3 = {

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',

customClass: '',

sanitize: true,

sanitizeFn: null,

whiteList: DefaultWhitelist,

popperConfig: null

};

var DefaultType$3 = {

animation: 'boolean',

template: 'string',

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

trigger: 'string',

delay: '(number|object)',

html: 'boolean',

selector: '(string|boolean)',

placement: '(string|function)',

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

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

fallbackPlacement: '(string|array)',

boundary: '(string|element)',

customClass: '(string|function)',

sanitize: 'boolean',

sanitizeFn: '(null|function)',

whiteList: 'object',

popperConfig: '(null|object)'

};

var Event$1 = {

HIDE: "hide" + EVENT\_KEY$4,

HIDDEN: "hidden" + EVENT\_KEY$4,

SHOW: "show" + EVENT\_KEY$4,

SHOWN: "shown" + EVENT\_KEY$4,

INSERTED: "inserted" + EVENT\_KEY$4,

CLICK: "click" + EVENT\_KEY$4,

FOCUSIN: "focusin" + EVENT\_KEY$4,

FOCUSOUT: "focusout" + EVENT\_KEY$4,

MOUSEENTER: "mouseenter" + EVENT\_KEY$4,

MOUSELEAVE: "mouseleave" + EVENT\_KEY$4

};

/\*\*

\* Class definition

\*/

var Tooltip = /\*#\_\_PURE\_\_\*/function () {

function Tooltip(element, config) {

if (typeof Popper\_\_default["default"] === 'undefined') {

throw new TypeError('Bootstrap\'s tooltips require Popper (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 = $\_\_default["default"](event.currentTarget).data(dataKey);

if (!context) {

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

$\_\_default["default"](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 ($\_\_default["default"](this.getTipElement()).hasClass(CLASS\_NAME\_SHOW$3)) {

this.\_leave(null, this);

return;

}

this.\_enter(null, this);

}

};

\_proto.dispose = function dispose() {

clearTimeout(this.\_timeout);

$\_\_default["default"].removeData(this.element, this.constructor.DATA\_KEY);

$\_\_default["default"](this.element).off(this.constructor.EVENT\_KEY);

$\_\_default["default"](this.element).closest('.modal').off('hide.bs.modal', this.\_hideModalHandler);

if (this.tip) {

$\_\_default["default"](this.tip).remove();

}

this.\_isEnabled = null;

this.\_timeout = null;

this.\_hoverState = null;

this.\_activeTrigger = null;

if (this.\_popper) {

this.\_popper.destroy();

}

this.\_popper = null;

this.element = null;

this.config = null;

this.tip = null;

};

\_proto.show = function show() {

var \_this = this;

if ($\_\_default["default"](this.element).css('display') === 'none') {

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

}

var showEvent = $\_\_default["default"].Event(this.constructor.Event.SHOW);

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

$\_\_default["default"](this.element).trigger(showEvent);

var shadowRoot = Util.findShadowRoot(this.element);

var isInTheDom = $\_\_default["default"].contains(shadowRoot !== null ? shadowRoot : 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) {

$\_\_default["default"](tip).addClass(CLASS\_NAME\_FADE$3);

}

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.\_getContainer();

$\_\_default["default"](tip).data(this.constructor.DATA\_KEY, this);

if (!$\_\_default["default"].contains(this.element.ownerDocument.documentElement, this.tip)) {

$\_\_default["default"](tip).appendTo(container);

}

$\_\_default["default"](this.element).trigger(this.constructor.Event.INSERTED);

this.\_popper = new Popper\_\_default["default"](this.element, tip, this.\_getPopperConfig(attachment));

$\_\_default["default"](tip).addClass(CLASS\_NAME\_SHOW$3);

$\_\_default["default"](tip).addClass(this.config.customClass); // 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) {

$\_\_default["default"](document.body).children().on('mouseover', null, $\_\_default["default"].noop);

}

var complete = function complete() {

if (\_this.config.animation) {

\_this.\_fixTransition();

}

var prevHoverState = \_this.\_hoverState;

\_this.\_hoverState = null;

$\_\_default["default"](\_this.element).trigger(\_this.constructor.Event.SHOWN);

if (prevHoverState === HOVER\_STATE\_OUT) {

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

}

};

if ($\_\_default["default"](this.tip).hasClass(CLASS\_NAME\_FADE$3)) {

var transitionDuration = Util.getTransitionDurationFromElement(this.tip);

$\_\_default["default"](this.tip).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(transitionDuration);

} else {

complete();

}

}

};

\_proto.hide = function hide(callback) {

var \_this2 = this;

var tip = this.getTipElement();

var hideEvent = $\_\_default["default"].Event(this.constructor.Event.HIDE);

var complete = function complete() {

if (\_this2.\_hoverState !== HOVER\_STATE\_SHOW && tip.parentNode) {

tip.parentNode.removeChild(tip);

}

\_this2.\_cleanTipClass();

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

$\_\_default["default"](\_this2.element).trigger(\_this2.constructor.Event.HIDDEN);

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

\_this2.\_popper.destroy();

}

if (callback) {

callback();

}

};

$\_\_default["default"](this.element).trigger(hideEvent);

if (hideEvent.isDefaultPrevented()) {

return;

}

$\_\_default["default"](tip).removeClass(CLASS\_NAME\_SHOW$3); // If this is a touch-enabled device we remove the extra

// empty mouseover listeners we added for iOS support

if ('ontouchstart' in document.documentElement) {

$\_\_default["default"](document.body).children().off('mouseover', null, $\_\_default["default"].noop);

}

this.\_activeTrigger[TRIGGER\_CLICK] = false;

this.\_activeTrigger[TRIGGER\_FOCUS] = false;

this.\_activeTrigger[TRIGGER\_HOVER] = false;

if ($\_\_default["default"](this.tip).hasClass(CLASS\_NAME\_FADE$3)) {

var transitionDuration = Util.getTransitionDurationFromElement(tip);

$\_\_default["default"](tip).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(transitionDuration);

} 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) {

$\_\_default["default"](this.getTipElement()).addClass(CLASS\_PREFIX$1 + "-" + attachment);

};

\_proto.getTipElement = function getTipElement() {

this.tip = this.tip || $\_\_default["default"](this.config.template)[0];

return this.tip;

};

\_proto.setContent = function setContent() {

var tip = this.getTipElement();

this.setElementContent($\_\_default["default"](tip.querySelectorAll(SELECTOR\_TOOLTIP\_INNER)), this.getTitle());

$\_\_default["default"](tip).removeClass(CLASS\_NAME\_FADE$3 + " " + CLASS\_NAME\_SHOW$3);

};

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

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

// Content is a DOM node or a jQuery

if (this.config.html) {

if (!$\_\_default["default"](content).parent().is($element)) {

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

}

} else {

$element.text($\_\_default["default"](content).text());

}

return;

}

if (this.config.html) {

if (this.config.sanitize) {

content = sanitizeHtml(content, this.config.whiteList, this.config.sanitizeFn);

}

$element.html(content);

} else {

$element.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.\_getPopperConfig = function \_getPopperConfig(attachment) {

var \_this3 = this;

var defaultBsConfig = {

placement: attachment,

modifiers: {

offset: this.\_getOffset(),

flip: {

behavior: this.config.fallbackPlacement

},

arrow: {

element: SELECTOR\_ARROW

},

preventOverflow: {

boundariesElement: this.config.boundary

}

},

onCreate: function onCreate(data) {

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

\_this3.\_handlePopperPlacementChange(data);

}

},

onUpdate: function onUpdate(data) {

return \_this3.\_handlePopperPlacementChange(data);

}

};

return \_extends({}, defaultBsConfig, this.config.popperConfig);

};

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

var \_this4 = this;

var offset = {};

if (typeof this.config.offset === 'function') {

offset.fn = function (data) {

data.offsets = \_extends({}, data.offsets, \_this4.config.offset(data.offsets, \_this4.element));

return data;

};

} else {

offset.offset = this.config.offset;

}

return offset;

};

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

if (this.config.container === false) {

return document.body;

}

if (Util.isElement(this.config.container)) {

return $\_\_default["default"](this.config.container);

}

return $\_\_default["default"](document).find(this.config.container);

};

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

return AttachmentMap[placement.toUpperCase()];

};

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

var \_this5 = this;

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

triggers.forEach(function (trigger) {

if (trigger === 'click') {

$\_\_default["default"](\_this5.element).on(\_this5.constructor.Event.CLICK, \_this5.config.selector, function (event) {

return \_this5.toggle(event);

});

} else if (trigger !== TRIGGER\_MANUAL) {

var eventIn = trigger === TRIGGER\_HOVER ? \_this5.constructor.Event.MOUSEENTER : \_this5.constructor.Event.FOCUSIN;

var eventOut = trigger === TRIGGER\_HOVER ? \_this5.constructor.Event.MOUSELEAVE : \_this5.constructor.Event.FOCUSOUT;

$\_\_default["default"](\_this5.element).on(eventIn, \_this5.config.selector, function (event) {

return \_this5.\_enter(event);

}).on(eventOut, \_this5.config.selector, function (event) {

return \_this5.\_leave(event);

});

}

});

this.\_hideModalHandler = function () {

if (\_this5.element) {

\_this5.hide();

}

};

$\_\_default["default"](this.element).closest('.modal').on('hide.bs.modal', this.\_hideModalHandler);

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 || $\_\_default["default"](event.currentTarget).data(dataKey);

if (!context) {

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

$\_\_default["default"](event.currentTarget).data(dataKey, context);

}

if (event) {

context.\_activeTrigger[event.type === 'focusin' ? TRIGGER\_FOCUS : TRIGGER\_HOVER] = true;

}

if ($\_\_default["default"](context.getTipElement()).hasClass(CLASS\_NAME\_SHOW$3) || context.\_hoverState === HOVER\_STATE\_SHOW) {

context.\_hoverState = HOVER\_STATE\_SHOW;

return;

}

clearTimeout(context.\_timeout);

context.\_hoverState = HOVER\_STATE\_SHOW;

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

context.show();

return;

}

context.\_timeout = setTimeout(function () {

if (context.\_hoverState === HOVER\_STATE\_SHOW) {

context.show();

}

}, context.config.delay.show);

};

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

var dataKey = this.constructor.DATA\_KEY;

context = context || $\_\_default["default"](event.currentTarget).data(dataKey);

if (!context) {

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

$\_\_default["default"](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 = HOVER\_STATE\_OUT;

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

context.hide();

return;

}

context.\_timeout = setTimeout(function () {

if (context.\_hoverState === HOVER\_STATE\_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) {

var dataAttributes = $\_\_default["default"](this.element).data();

Object.keys(dataAttributes).forEach(function (dataAttr) {

if (DISALLOWED\_ATTRIBUTES.indexOf(dataAttr) !== -1) {

delete dataAttributes[dataAttr];

}

});

config = \_extends({}, this.constructor.Default, dataAttributes, typeof config === 'object' && config ? 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$4, config, this.constructor.DefaultType);

if (config.sanitize) {

config.template = sanitizeHtml(config.template, config.whiteList, config.sanitizeFn);

}

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 = $\_\_default["default"](this.getTipElement());

var tabClass = $tip.attr('class').match(BSCLS\_PREFIX\_REGEX$1);

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

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

}

};

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

this.tip = popperData.instance.popper;

this.\_cleanTipClass();

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

};

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

var tip = this.getTipElement();

var initConfigAnimation = this.config.animation;

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

return;

}

$\_\_default["default"](tip).removeClass(CLASS\_NAME\_FADE$3);

this.config.animation = false;

this.hide();

this.show();

this.config.animation = initConfigAnimation;

} // Static

;

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

return this.each(function () {

var $element = $\_\_default["default"](this);

var data = $element.data(DATA\_KEY$4);

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

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

return;

}

if (!data) {

data = new Tooltip(this, \_config);

$element.data(DATA\_KEY$4, 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$4;

}

}, {

key: "Default",

get: function get() {

return Default$3;

}

}, {

key: "NAME",

get: function get() {

return NAME$4;

}

}, {

key: "DATA\_KEY",

get: function get() {

return DATA\_KEY$4;

}

}, {

key: "Event",

get: function get() {

return Event$1;

}

}, {

key: "EVENT\_KEY",

get: function get() {

return EVENT\_KEY$4;

}

}, {

key: "DefaultType",

get: function get() {

return DefaultType$3;

}

}]);

return Tooltip;

}();

/\*\*

\* jQuery

\*/

$\_\_default["default"].fn[NAME$4] = Tooltip.\_jQueryInterface;

$\_\_default["default"].fn[NAME$4].Constructor = Tooltip;

$\_\_default["default"].fn[NAME$4].noConflict = function () {

$\_\_default["default"].fn[NAME$4] = JQUERY\_NO\_CONFLICT$4;

return Tooltip.\_jQueryInterface;

};

/\*\*

\* Constants

\*/

var NAME$3 = 'popover';

var VERSION$3 = '4.6.1';

var DATA\_KEY$3 = 'bs.popover';

var EVENT\_KEY$3 = "." + DATA\_KEY$3;

var JQUERY\_NO\_CONFLICT$3 = $\_\_default["default"].fn[NAME$3];

var CLASS\_PREFIX = 'bs-popover';

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

var CLASS\_NAME\_FADE$2 = 'fade';

var CLASS\_NAME\_SHOW$2 = 'show';

var SELECTOR\_TITLE = '.popover-header';

var SELECTOR\_CONTENT = '.popover-body';

var Default$2 = \_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$2 = \_extends({}, Tooltip.DefaultType, {

content: '(string|element|function)'

});

var Event = {

HIDE: "hide" + EVENT\_KEY$3,

HIDDEN: "hidden" + EVENT\_KEY$3,

SHOW: "show" + EVENT\_KEY$3,

SHOWN: "shown" + EVENT\_KEY$3,

INSERTED: "inserted" + EVENT\_KEY$3,

CLICK: "click" + EVENT\_KEY$3,

FOCUSIN: "focusin" + EVENT\_KEY$3,

FOCUSOUT: "focusout" + EVENT\_KEY$3,

MOUSEENTER: "mouseenter" + EVENT\_KEY$3,

MOUSELEAVE: "mouseleave" + EVENT\_KEY$3

};

/\*\*

\* 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) {

$\_\_default["default"](this.getTipElement()).addClass(CLASS\_PREFIX + "-" + attachment);

};

\_proto.getTipElement = function getTipElement() {

this.tip = this.tip || $\_\_default["default"](this.config.template)[0];

return this.tip;

};

\_proto.setContent = function setContent() {

var $tip = $\_\_default["default"](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(CLASS\_NAME\_FADE$2 + " " + CLASS\_NAME\_SHOW$2);

} // Private

;

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

return this.element.getAttribute('data-content') || this.config.content;

};

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

var $tip = $\_\_default["default"](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 = $\_\_default["default"](this).data(DATA\_KEY$3);

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

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

return;

}

if (!data) {

data = new Popover(this, \_config);

$\_\_default["default"](this).data(DATA\_KEY$3, data);

}

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

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

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

}

data[config]();

}

});

};

\_createClass(Popover, null, [{

key: "VERSION",

get: // Getters

function get() {

return VERSION$3;

}

}, {

key: "Default",

get: function get() {

return Default$2;

}

}, {

key: "NAME",

get: function get() {

return NAME$3;

}

}, {

key: "DATA\_KEY",

get: function get() {

return DATA\_KEY$3;

}

}, {

key: "Event",

get: function get() {

return Event;

}

}, {

key: "EVENT\_KEY",

get: function get() {

return EVENT\_KEY$3;

}

}, {

key: "DefaultType",

get: function get() {

return DefaultType$2;

}

}]);

return Popover;

}(Tooltip);

/\*\*

\* jQuery

\*/

$\_\_default["default"].fn[NAME$3] = Popover.\_jQueryInterface;

$\_\_default["default"].fn[NAME$3].Constructor = Popover;

$\_\_default["default"].fn[NAME$3].noConflict = function () {

$\_\_default["default"].fn[NAME$3] = JQUERY\_NO\_CONFLICT$3;

return Popover.\_jQueryInterface;

};

/\*\*

\* Constants

\*/

var NAME$2 = 'scrollspy';

var VERSION$2 = '4.6.1';

var DATA\_KEY$2 = 'bs.scrollspy';

var EVENT\_KEY$2 = "." + DATA\_KEY$2;

var DATA\_API\_KEY$1 = '.data-api';

var JQUERY\_NO\_CONFLICT$2 = $\_\_default["default"].fn[NAME$2];

var CLASS\_NAME\_DROPDOWN\_ITEM = 'dropdown-item';

var CLASS\_NAME\_ACTIVE$1 = 'active';

var EVENT\_ACTIVATE = "activate" + EVENT\_KEY$2;

var EVENT\_SCROLL = "scroll" + EVENT\_KEY$2;

var EVENT\_LOAD\_DATA\_API = "load" + EVENT\_KEY$2 + DATA\_API\_KEY$1;

var METHOD\_OFFSET = 'offset';

var METHOD\_POSITION = 'position';

var SELECTOR\_DATA\_SPY = '[data-spy="scroll"]';

var SELECTOR\_NAV\_LIST\_GROUP$1 = '.nav, .list-group';

var SELECTOR\_NAV\_LINKS = '.nav-link';

var SELECTOR\_NAV\_ITEMS = '.nav-item';

var SELECTOR\_LIST\_ITEMS = '.list-group-item';

var SELECTOR\_DROPDOWN$1 = '.dropdown';

var SELECTOR\_DROPDOWN\_ITEMS = '.dropdown-item';

var SELECTOR\_DROPDOWN\_TOGGLE$1 = '.dropdown-toggle';

var Default$1 = {

offset: 10,

method: 'auto',

target: ''

};

var DefaultType$1 = {

offset: 'number',

method: 'string',

target: '(string|element)'

};

/\*\*

\* 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;

$\_\_default["default"](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 ? METHOD\_OFFSET : METHOD\_POSITION;

var offsetMethod = this.\_config.method === 'auto' ? autoMethod : this.\_config.method;

var offsetBase = offsetMethod === METHOD\_POSITION ? this.\_getScrollTop() : 0;

this.\_offsets = [];

this.\_targets = [];

this.\_scrollHeight = this.\_getScrollHeight();

var targets = [].slice.call(document.querySelectorAll(this.\_selector));

targets.map(function (element) {

var target;

var targetSelector = Util.getSelectorFromElement(element);

if (targetSelector) {

target = document.querySelector(targetSelector);

}

if (target) {

var targetBCR = target.getBoundingClientRect();

if (targetBCR.width || targetBCR.height) {

// TODO (fat): remove sketch reliance on jQuery position/offset

return [$\_\_default["default"](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() {

$\_\_default["default"].removeData(this.\_element, DATA\_KEY$2);

$\_\_default["default"](this.\_scrollElement).off(EVENT\_KEY$2);

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$1, typeof config === 'object' && config ? config : {});

if (typeof config.target !== 'string' && Util.isElement(config.target)) {

var id = $\_\_default["default"](config.target).attr('id');

if (!id) {

id = Util.getUID(NAME$2);

$\_\_default["default"](config.target).attr('id', id);

}

config.target = "#" + id;

}

Util.typeCheckConfig(NAME$2, config, DefaultType$1);

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(',').map(function (selector) {

return selector + "[data-target=\"" + target + "\"]," + selector + "[href=\"" + target + "\"]";

});

var $link = $\_\_default["default"]([].slice.call(document.querySelectorAll(queries.join(','))));

if ($link.hasClass(CLASS\_NAME\_DROPDOWN\_ITEM)) {

$link.closest(SELECTOR\_DROPDOWN$1).find(SELECTOR\_DROPDOWN\_TOGGLE$1).addClass(CLASS\_NAME\_ACTIVE$1);

$link.addClass(CLASS\_NAME\_ACTIVE$1);

} else {

// Set triggered link as active

$link.addClass(CLASS\_NAME\_ACTIVE$1); // 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$1).prev(SELECTOR\_NAV\_LINKS + ", " + SELECTOR\_LIST\_ITEMS).addClass(CLASS\_NAME\_ACTIVE$1); // Handle special case when .nav-link is inside .nav-item

$link.parents(SELECTOR\_NAV\_LIST\_GROUP$1).prev(SELECTOR\_NAV\_ITEMS).children(SELECTOR\_NAV\_LINKS).addClass(CLASS\_NAME\_ACTIVE$1);

}

$\_\_default["default"](this.\_scrollElement).trigger(EVENT\_ACTIVATE, {

relatedTarget: target

});

};

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

[].slice.call(document.querySelectorAll(this.\_selector)).filter(function (node) {

return node.classList.contains(CLASS\_NAME\_ACTIVE$1);

}).forEach(function (node) {

return node.classList.remove(CLASS\_NAME\_ACTIVE$1);

});

} // Static

;

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

return this.each(function () {

var data = $\_\_default["default"](this).data(DATA\_KEY$2);

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

if (!data) {

data = new ScrollSpy(this, \_config);

$\_\_default["default"](this).data(DATA\_KEY$2, 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$2;

}

}, {

key: "Default",

get: function get() {

return Default$1;

}

}]);

return ScrollSpy;

}();

/\*\*

\* Data API implementation

\*/

$\_\_default["default"](window).on(EVENT\_LOAD\_DATA\_API, function () {

var scrollSpys = [].slice.call(document.querySelectorAll(SELECTOR\_DATA\_SPY));

var scrollSpysLength = scrollSpys.length;

for (var i = scrollSpysLength; i--;) {

var $spy = $\_\_default["default"](scrollSpys[i]);

ScrollSpy.\_jQueryInterface.call($spy, $spy.data());

}

});

/\*\*

\* jQuery

\*/

$\_\_default["default"].fn[NAME$2] = ScrollSpy.\_jQueryInterface;

$\_\_default["default"].fn[NAME$2].Constructor = ScrollSpy;

$\_\_default["default"].fn[NAME$2].noConflict = function () {

$\_\_default["default"].fn[NAME$2] = JQUERY\_NO\_CONFLICT$2;

return ScrollSpy.\_jQueryInterface;

};

/\*\*

\* Constants

\*/

var NAME$1 = 'tab';

var VERSION$1 = '4.6.1';

var DATA\_KEY$1 = 'bs.tab';

var EVENT\_KEY$1 = "." + DATA\_KEY$1;

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

var JQUERY\_NO\_CONFLICT$1 = $\_\_default["default"].fn[NAME$1];

var CLASS\_NAME\_DROPDOWN\_MENU = 'dropdown-menu';

var CLASS\_NAME\_ACTIVE = 'active';

var CLASS\_NAME\_DISABLED = 'disabled';

var CLASS\_NAME\_FADE$1 = 'fade';

var CLASS\_NAME\_SHOW$1 = 'show';

var EVENT\_HIDE$1 = "hide" + EVENT\_KEY$1;

var EVENT\_HIDDEN$1 = "hidden" + EVENT\_KEY$1;

var EVENT\_SHOW$1 = "show" + EVENT\_KEY$1;

var EVENT\_SHOWN$1 = "shown" + EVENT\_KEY$1;

var EVENT\_CLICK\_DATA\_API = "click" + EVENT\_KEY$1 + DATA\_API\_KEY;

var SELECTOR\_DROPDOWN = '.dropdown';

var SELECTOR\_NAV\_LIST\_GROUP = '.nav, .list-group';

var SELECTOR\_ACTIVE = '.active';

var SELECTOR\_ACTIVE\_UL = '> li > .active';

var SELECTOR\_DATA\_TOGGLE = '[data-toggle="tab"], [data-toggle="pill"], [data-toggle="list"]';

var SELECTOR\_DROPDOWN\_TOGGLE = '.dropdown-toggle';

var SELECTOR\_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 && $\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_ACTIVE) || $\_\_default["default"](this.\_element).hasClass(CLASS\_NAME\_DISABLED)) {

return;

}

var target;

var previous;

var listElement = $\_\_default["default"](this.\_element).closest(SELECTOR\_NAV\_LIST\_GROUP)[0];

var selector = Util.getSelectorFromElement(this.\_element);

if (listElement) {

var itemSelector = listElement.nodeName === 'UL' || listElement.nodeName === 'OL' ? SELECTOR\_ACTIVE\_UL : SELECTOR\_ACTIVE;

previous = $\_\_default["default"].makeArray($\_\_default["default"](listElement).find(itemSelector));

previous = previous[previous.length - 1];

}

var hideEvent = $\_\_default["default"].Event(EVENT\_HIDE$1, {

relatedTarget: this.\_element

});

var showEvent = $\_\_default["default"].Event(EVENT\_SHOW$1, {

relatedTarget: previous

});

if (previous) {

$\_\_default["default"](previous).trigger(hideEvent);

}

$\_\_default["default"](this.\_element).trigger(showEvent);

if (showEvent.isDefaultPrevented() || hideEvent.isDefaultPrevented()) {

return;

}

if (selector) {

target = document.querySelector(selector);

}

this.\_activate(this.\_element, listElement);

var complete = function complete() {

var hiddenEvent = $\_\_default["default"].Event(EVENT\_HIDDEN$1, {

relatedTarget: \_this.\_element

});

var shownEvent = $\_\_default["default"].Event(EVENT\_SHOWN$1, {

relatedTarget: previous

});

$\_\_default["default"](previous).trigger(hiddenEvent);

$\_\_default["default"](\_this.\_element).trigger(shownEvent);

};

if (target) {

this.\_activate(target, target.parentNode, complete);

} else {

complete();

}

};

\_proto.dispose = function dispose() {

$\_\_default["default"].removeData(this.\_element, DATA\_KEY$1);

this.\_element = null;

} // Private

;

\_proto.\_activate = function \_activate(element, container, callback) {

var \_this2 = this;

var activeElements = container && (container.nodeName === 'UL' || container.nodeName === 'OL') ? $\_\_default["default"](container).find(SELECTOR\_ACTIVE\_UL) : $\_\_default["default"](container).children(SELECTOR\_ACTIVE);

var active = activeElements[0];

var isTransitioning = callback && active && $\_\_default["default"](active).hasClass(CLASS\_NAME\_FADE$1);

var complete = function complete() {

return \_this2.\_transitionComplete(element, active, callback);

};

if (active && isTransitioning) {

var transitionDuration = Util.getTransitionDurationFromElement(active);

$\_\_default["default"](active).removeClass(CLASS\_NAME\_SHOW$1).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(transitionDuration);

} else {

complete();

}

};

\_proto.\_transitionComplete = function \_transitionComplete(element, active, callback) {

if (active) {

$\_\_default["default"](active).removeClass(CLASS\_NAME\_ACTIVE);

var dropdownChild = $\_\_default["default"](active.parentNode).find(SELECTOR\_DROPDOWN\_ACTIVE\_CHILD)[0];

if (dropdownChild) {

$\_\_default["default"](dropdownChild).removeClass(CLASS\_NAME\_ACTIVE);

}

if (active.getAttribute('role') === 'tab') {

active.setAttribute('aria-selected', false);

}

}

$\_\_default["default"](element).addClass(CLASS\_NAME\_ACTIVE);

if (element.getAttribute('role') === 'tab') {

element.setAttribute('aria-selected', true);

}

Util.reflow(element);

if (element.classList.contains(CLASS\_NAME\_FADE$1)) {

element.classList.add(CLASS\_NAME\_SHOW$1);

}

var parent = element.parentNode;

if (parent && parent.nodeName === 'LI') {

parent = parent.parentNode;

}

if (parent && $\_\_default["default"](parent).hasClass(CLASS\_NAME\_DROPDOWN\_MENU)) {

var dropdownElement = $\_\_default["default"](element).closest(SELECTOR\_DROPDOWN)[0];

if (dropdownElement) {

var dropdownToggleList = [].slice.call(dropdownElement.querySelectorAll(SELECTOR\_DROPDOWN\_TOGGLE));

$\_\_default["default"](dropdownToggleList).addClass(CLASS\_NAME\_ACTIVE);

}

element.setAttribute('aria-expanded', true);

}

if (callback) {

callback();

}

} // Static

;

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

return this.each(function () {

var $this = $\_\_default["default"](this);

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

if (!data) {

data = new Tab(this);

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

}

}]);

return Tab;

}();

/\*\*

\* Data API implementation

\*/

$\_\_default["default"](document).on(EVENT\_CLICK\_DATA\_API, SELECTOR\_DATA\_TOGGLE, function (event) {

event.preventDefault();

Tab.\_jQueryInterface.call($\_\_default["default"](this), 'show');

});

/\*\*

\* jQuery

\*/

$\_\_default["default"].fn[NAME$1] = Tab.\_jQueryInterface;

$\_\_default["default"].fn[NAME$1].Constructor = Tab;

$\_\_default["default"].fn[NAME$1].noConflict = function () {

$\_\_default["default"].fn[NAME$1] = JQUERY\_NO\_CONFLICT$1;

return Tab.\_jQueryInterface;

};

/\*\*

\* Constants

\*/

var NAME = 'toast';

var VERSION = '4.6.1';

var DATA\_KEY = 'bs.toast';

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

var JQUERY\_NO\_CONFLICT = $\_\_default["default"].fn[NAME];

var CLASS\_NAME\_FADE = 'fade';

var CLASS\_NAME\_HIDE = 'hide';

var CLASS\_NAME\_SHOW = 'show';

var CLASS\_NAME\_SHOWING = 'showing';

var EVENT\_CLICK\_DISMISS = "click.dismiss" + EVENT\_KEY;

var EVENT\_HIDE = "hide" + EVENT\_KEY;

var EVENT\_HIDDEN = "hidden" + EVENT\_KEY;

var EVENT\_SHOW = "show" + EVENT\_KEY;

var EVENT\_SHOWN = "shown" + EVENT\_KEY;

var SELECTOR\_DATA\_DISMISS = '[data-dismiss="toast"]';

var Default = {

animation: true,

autohide: true,

delay: 500

};

var DefaultType = {

animation: 'boolean',

autohide: 'boolean',

delay: 'number'

};

/\*\*

\* Class definition

\*/

var Toast = /\*#\_\_PURE\_\_\*/function () {

function Toast(element, config) {

this.\_element = element;

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

this.\_timeout = null;

this.\_setListeners();

} // Getters

var \_proto = Toast.prototype;

// Public

\_proto.show = function show() {

var \_this = this;

var showEvent = $\_\_default["default"].Event(EVENT\_SHOW);

$\_\_default["default"](this.\_element).trigger(showEvent);

if (showEvent.isDefaultPrevented()) {

return;

}

this.\_clearTimeout();

if (this.\_config.animation) {

this.\_element.classList.add(CLASS\_NAME\_FADE);

}

var complete = function complete() {

\_this.\_element.classList.remove(CLASS\_NAME\_SHOWING);

\_this.\_element.classList.add(CLASS\_NAME\_SHOW);

$\_\_default["default"](\_this.\_element).trigger(EVENT\_SHOWN);

if (\_this.\_config.autohide) {

\_this.\_timeout = setTimeout(function () {

\_this.hide();

}, \_this.\_config.delay);

}

};

this.\_element.classList.remove(CLASS\_NAME\_HIDE);

Util.reflow(this.\_element);

this.\_element.classList.add(CLASS\_NAME\_SHOWING);

if (this.\_config.animation) {

var transitionDuration = Util.getTransitionDurationFromElement(this.\_element);

$\_\_default["default"](this.\_element).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(transitionDuration);

} else {

complete();

}

};

\_proto.hide = function hide() {

if (!this.\_element.classList.contains(CLASS\_NAME\_SHOW)) {

return;

}

var hideEvent = $\_\_default["default"].Event(EVENT\_HIDE);

$\_\_default["default"](this.\_element).trigger(hideEvent);

if (hideEvent.isDefaultPrevented()) {

return;

}

this.\_close();

};

\_proto.dispose = function dispose() {

this.\_clearTimeout();

if (this.\_element.classList.contains(CLASS\_NAME\_SHOW)) {

this.\_element.classList.remove(CLASS\_NAME\_SHOW);

}

$\_\_default["default"](this.\_element).off(EVENT\_CLICK\_DISMISS);

$\_\_default["default"].removeData(this.\_element, DATA\_KEY);

this.\_element = null;

this.\_config = null;

} // Private

;

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

config = \_extends({}, Default, $\_\_default["default"](this.\_element).data(), typeof config === 'object' && config ? config : {});

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

return config;

};

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

var \_this2 = this;

$\_\_default["default"](this.\_element).on(EVENT\_CLICK\_DISMISS, SELECTOR\_DATA\_DISMISS, function () {

return \_this2.hide();

});

};

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

var \_this3 = this;

var complete = function complete() {

\_this3.\_element.classList.add(CLASS\_NAME\_HIDE);

$\_\_default["default"](\_this3.\_element).trigger(EVENT\_HIDDEN);

};

this.\_element.classList.remove(CLASS\_NAME\_SHOW);

if (this.\_config.animation) {

var transitionDuration = Util.getTransitionDurationFromElement(this.\_element);

$\_\_default["default"](this.\_element).one(Util.TRANSITION\_END, complete).emulateTransitionEnd(transitionDuration);

} else {

complete();

}

};

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

clearTimeout(this.\_timeout);

this.\_timeout = null;

} // Static

;

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

return this.each(function () {

var $element = $\_\_default["default"](this);

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

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

if (!data) {

data = new Toast(this, \_config);

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

}

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

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

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

}

data[config](this);

}

});

};

\_createClass(Toast, null, [{

key: "VERSION",

get: function get() {

return VERSION;

}

}, {

key: "DefaultType",

get: function get() {

return DefaultType;

}

}, {

key: "Default",

get: function get() {

return Default;

}

}]);

return Toast;

}();

/\*\*

\* jQuery

\*/

$\_\_default["default"].fn[NAME] = Toast.\_jQueryInterface;

$\_\_default["default"].fn[NAME].Constructor = Toast;

$\_\_default["default"].fn[NAME].noConflict = function () {

$\_\_default["default"].fn[NAME] = JQUERY\_NO\_CONFLICT;

return Toast.\_jQueryInterface;

};

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.Toast = Toast;

exports.Tooltip = Tooltip;

exports.Util = Util;

Object.defineProperty(exports, '\_\_esModule', { value: true });

}));