/\*! jQuery UI - v1.12.1 - 2018-12-06

\* http://jqueryui.com

\* Includes: widget.js, position.js, keycode.js, unique-id.js, widgets/autocomplete.js, widgets/menu.js

\* Copyright jQuery Foundation and other contributors; Licensed MIT \*/

(function( factory ) {

if ( typeof define === "function" && define.amd ) {

// AMD. Register as an anonymous module.

define([ "jquery" ], factory );

} else {

// Browser globals

factory( jQuery );

}

}(function( $ ) {

$.ui = $.ui || {};

var version = $.ui.version = "1.12.1";

/\*!

\* jQuery UI Widget 1.12.1

\* http://jqueryui.com

\*

\* Copyright jQuery Foundation and other contributors

\* Released under the MIT license.

\* http://jquery.org/license

\*/

//>>label: Widget

//>>group: Core

//>>description: Provides a factory for creating stateful widgets with a common API.

//>>docs: http://api.jqueryui.com/jQuery.widget/

//>>demos: http://jqueryui.com/widget/

var widgetUuid = 0;

var widgetSlice = Array.prototype.slice;

$.cleanData = ( function( orig ) {

return function( elems ) {

var events, elem, i;

for ( i = 0; ( elem = elems[ i ] ) != null; i++ ) {

try {

// Only trigger remove when necessary to save time

events = $.\_data( elem, "events" );

if ( events && events.remove ) {

$( elem ).triggerHandler( "remove" );

}

// Http://bugs.jquery.com/ticket/8235

} catch ( e ) {}

}

orig( elems );

};

} )( $.cleanData );

$.widget = function( name, base, prototype ) {

var existingConstructor, constructor, basePrototype;

// ProxiedPrototype allows the provided prototype to remain unmodified

// so that it can be used as a mixin for multiple widgets (#8876)

var proxiedPrototype = {};

var namespace = name.split( "." )[ 0 ];

name = name.split( "." )[ 1 ];

var fullName = namespace + "-" + name;

if ( !prototype ) {

prototype = base;

base = $.Widget;

}

if ( $.isArray( prototype ) ) {

prototype = $.extend.apply( null, [ {} ].concat( prototype ) );

}

// Create selector for plugin

$.expr[ ":" ][ fullName.toLowerCase() ] = function( elem ) {

return !!$.data( elem, fullName );

};

$[ namespace ] = $[ namespace ] || {};

existingConstructor = $[ namespace ][ name ];

constructor = $[ namespace ][ name ] = function( options, element ) {

// Allow instantiation without "new" keyword

if ( !this.\_createWidget ) {

return new constructor( options, element );

}

// Allow instantiation without initializing for simple inheritance

// must use "new" keyword (the code above always passes args)

if ( arguments.length ) {

this.\_createWidget( options, element );

}

};

// Extend with the existing constructor to carry over any static properties

$.extend( constructor, existingConstructor, {

version: prototype.version,

// Copy the object used to create the prototype in case we need to

// redefine the widget later

\_proto: $.extend( {}, prototype ),

// Track widgets that inherit from this widget in case this widget is

// redefined after a widget inherits from it

\_childConstructors: []

} );

basePrototype = new base();

// We need to make the options hash a property directly on the new instance

// otherwise we'll modify the options hash on the prototype that we're

// inheriting from

basePrototype.options = $.widget.extend( {}, basePrototype.options );

$.each( prototype, function( prop, value ) {

if ( !$.isFunction( value ) ) {

proxiedPrototype[ prop ] = value;

return;

}

proxiedPrototype[ prop ] = ( function() {

function \_super() {

return base.prototype[ prop ].apply( this, arguments );

}

function \_superApply( args ) {

return base.prototype[ prop ].apply( this, args );

}

return function() {

var \_\_super = this.\_super;

var \_\_superApply = this.\_superApply;

var returnValue;

this.\_super = \_super;

this.\_superApply = \_superApply;

returnValue = value.apply( this, arguments );

this.\_super = \_\_super;

this.\_superApply = \_\_superApply;

return returnValue;

};

} )();

} );

constructor.prototype = $.widget.extend( basePrototype, {

// TODO: remove support for widgetEventPrefix

// always use the name + a colon as the prefix, e.g., draggable:start

// don't prefix for widgets that aren't DOM-based

widgetEventPrefix: existingConstructor ? ( basePrototype.widgetEventPrefix || name ) : name

}, proxiedPrototype, {

constructor: constructor,

namespace: namespace,

widgetName: name,

widgetFullName: fullName

} );

// If this widget is being redefined then we need to find all widgets that

// are inheriting from it and redefine all of them so that they inherit from

// the new version of this widget. We're essentially trying to replace one

// level in the prototype chain.

if ( existingConstructor ) {

$.each( existingConstructor.\_childConstructors, function( i, child ) {

var childPrototype = child.prototype;

// Redefine the child widget using the same prototype that was

// originally used, but inherit from the new version of the base

$.widget( childPrototype.namespace + "." + childPrototype.widgetName, constructor,

child.\_proto );

} );

// Remove the list of existing child constructors from the old constructor

// so the old child constructors can be garbage collected

delete existingConstructor.\_childConstructors;

} else {

base.\_childConstructors.push( constructor );

}

$.widget.bridge( name, constructor );

return constructor;

};

$.widget.extend = function( target ) {

var input = widgetSlice.call( arguments, 1 );

var inputIndex = 0;

var inputLength = input.length;

var key;

var value;

for ( ; inputIndex < inputLength; inputIndex++ ) {

for ( key in input[ inputIndex ] ) {

value = input[ inputIndex ][ key ];

if ( input[ inputIndex ].hasOwnProperty( key ) && value !== undefined ) {

// Clone objects

if ( $.isPlainObject( value ) ) {

target[ key ] = $.isPlainObject( target[ key ] ) ?

$.widget.extend( {}, target[ key ], value ) :

// Don't extend strings, arrays, etc. with objects

$.widget.extend( {}, value );

// Copy everything else by reference

} else {

target[ key ] = value;

}

}

}

}

return target;

};

$.widget.bridge = function( name, object ) {

var fullName = object.prototype.widgetFullName || name;

$.fn[ name ] = function( options ) {

var isMethodCall = typeof options === "string";

var args = widgetSlice.call( arguments, 1 );

var returnValue = this;

if ( isMethodCall ) {

// If this is an empty collection, we need to have the instance method

// return undefined instead of the jQuery instance

if ( !this.length && options === "instance" ) {

returnValue = undefined;

} else {

this.each( function() {

var methodValue;

var instance = $.data( this, fullName );

if ( options === "instance" ) {

returnValue = instance;

return false;

}

if ( !instance ) {

return $.error( "cannot call methods on " + name +

" prior to initialization; " +

"attempted to call method '" + options + "'" );

}

if ( !$.isFunction( instance[ options ] ) || options.charAt( 0 ) === "\_" ) {

return $.error( "no such method '" + options + "' for " + name +

" widget instance" );

}

methodValue = instance[ options ].apply( instance, args );

if ( methodValue !== instance && methodValue !== undefined ) {

returnValue = methodValue && methodValue.jquery ?

returnValue.pushStack( methodValue.get() ) :

methodValue;

return false;

}

} );

}

} else {

// Allow multiple hashes to be passed on init

if ( args.length ) {

options = $.widget.extend.apply( null, [ options ].concat( args ) );

}

this.each( function() {

var instance = $.data( this, fullName );

if ( instance ) {

instance.option( options || {} );

if ( instance.\_init ) {

instance.\_init();

}

} else {

$.data( this, fullName, new object( options, this ) );

}

} );

}

return returnValue;

};

};

$.Widget = function( /\* options, element \*/ ) {};

$.Widget.\_childConstructors = [];

$.Widget.prototype = {

widgetName: "widget",

widgetEventPrefix: "",

defaultElement: "<div>",

options: {

classes: {},

disabled: false,

// Callbacks

create: null

},

\_createWidget: function( options, element ) {

element = $( element || this.defaultElement || this )[ 0 ];

this.element = $( element );

this.uuid = widgetUuid++;

this.eventNamespace = "." + this.widgetName + this.uuid;

this.bindings = $();

this.hoverable = $();

this.focusable = $();

this.classesElementLookup = {};

if ( element !== this ) {

$.data( element, this.widgetFullName, this );

this.\_on( true, this.element, {

remove: function( event ) {

if ( event.target === element ) {

this.destroy();

}

}

} );

this.document = $( element.style ?

// Element within the document

element.ownerDocument :

// Element is window or document

element.document || element );

this.window = $( this.document[ 0 ].defaultView || this.document[ 0 ].parentWindow );

}

this.options = $.widget.extend( {},

this.options,

this.\_getCreateOptions(),

options );

this.\_create();

if ( this.options.disabled ) {

this.\_setOptionDisabled( this.options.disabled );

}

this.\_trigger( "create", null, this.\_getCreateEventData() );

this.\_init();

},

\_getCreateOptions: function() {

return {};

},

\_getCreateEventData: $.noop,

\_create: $.noop,

\_init: $.noop,

destroy: function() {

var that = this;

this.\_destroy();

$.each( this.classesElementLookup, function( key, value ) {

that.\_removeClass( value, key );

} );

// We can probably remove the unbind calls in 2.0

// all event bindings should go through this.\_on()

this.element

.off( this.eventNamespace )

.removeData( this.widgetFullName );

this.widget()

.off( this.eventNamespace )

.removeAttr( "aria-disabled" );

// Clean up events and states

this.bindings.off( this.eventNamespace );

},

\_destroy: $.noop,

widget: function() {

return this.element;

},

option: function( key, value ) {

var options = key;

var parts;

var curOption;

var i;

if ( arguments.length === 0 ) {

// Don't return a reference to the internal hash

return $.widget.extend( {}, this.options );

}

if ( typeof key === "string" ) {

// Handle nested keys, e.g., "foo.bar" => { foo: { bar: \_\_\_ } }

options = {};

parts = key.split( "." );

key = parts.shift();

if ( parts.length ) {

curOption = options[ key ] = $.widget.extend( {}, this.options[ key ] );

for ( i = 0; i < parts.length - 1; i++ ) {

curOption[ parts[ i ] ] = curOption[ parts[ i ] ] || {};

curOption = curOption[ parts[ i ] ];

}

key = parts.pop();

if ( arguments.length === 1 ) {

return curOption[ key ] === undefined ? null : curOption[ key ];

}

curOption[ key ] = value;

} else {

if ( arguments.length === 1 ) {

return this.options[ key ] === undefined ? null : this.options[ key ];

}

options[ key ] = value;

}

}

this.\_setOptions( options );

return this;

},

\_setOptions: function( options ) {

var key;

for ( key in options ) {

this.\_setOption( key, options[ key ] );

}

return this;

},

\_setOption: function( key, value ) {

if ( key === "classes" ) {

this.\_setOptionClasses( value );

}

this.options[ key ] = value;

if ( key === "disabled" ) {

this.\_setOptionDisabled( value );

}

return this;

},

\_setOptionClasses: function( value ) {

var classKey, elements, currentElements;

for ( classKey in value ) {

currentElements = this.classesElementLookup[ classKey ];

if ( value[ classKey ] === this.options.classes[ classKey ] ||

!currentElements ||

!currentElements.length ) {

continue;

}

// We are doing this to create a new jQuery object because the \_removeClass() call

// on the next line is going to destroy the reference to the current elements being

// tracked. We need to save a copy of this collection so that we can add the new classes

// below.

elements = $( currentElements.get() );

this.\_removeClass( currentElements, classKey );

// We don't use \_addClass() here, because that uses this.options.classes

// for generating the string of classes. We want to use the value passed in from

// \_setOption(), this is the new value of the classes option which was passed to

// \_setOption(). We pass this value directly to \_classes().

elements.addClass( this.\_classes( {

element: elements,

keys: classKey,

classes: value,

add: true

} ) );

}

},

\_setOptionDisabled: function( value ) {

this.\_toggleClass( this.widget(), this.widgetFullName + "-disabled", null, !!value );

// If the widget is becoming disabled, then nothing is interactive

if ( value ) {

this.\_removeClass( this.hoverable, null, "ui-state-hover" );

this.\_removeClass( this.focusable, null, "ui-state-focus" );

}

},

enable: function() {

return this.\_setOptions( { disabled: false } );

},

disable: function() {

return this.\_setOptions( { disabled: true } );

},

\_classes: function( options ) {

var full = [];

var that = this;

options = $.extend( {

element: this.element,

classes: this.options.classes || {}

}, options );

function processClassString( classes, checkOption ) {

var current, i;

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

current = that.classesElementLookup[ classes[ i ] ] || $();

if ( options.add ) {

current = $( $.unique( current.get().concat( options.element.get() ) ) );

} else {

current = $( current.not( options.element ).get() );

}

that.classesElementLookup[ classes[ i ] ] = current;

full.push( classes[ i ] );

if ( checkOption && options.classes[ classes[ i ] ] ) {

full.push( options.classes[ classes[ i ] ] );

}

}

}

this.\_on( options.element, {

"remove": "\_untrackClassesElement"

} );

if ( options.keys ) {

processClassString( options.keys.match( /\S+/g ) || [], true );

}

if ( options.extra ) {

processClassString( options.extra.match( /\S+/g ) || [] );

}

return full.join( " " );

},

\_untrackClassesElement: function( event ) {

var that = this;

$.each( that.classesElementLookup, function( key, value ) {

if ( $.inArray( event.target, value ) !== -1 ) {

that.classesElementLookup[ key ] = $( value.not( event.target ).get() );

}

} );

},

\_removeClass: function( element, keys, extra ) {

return this.\_toggleClass( element, keys, extra, false );

},

\_addClass: function( element, keys, extra ) {

return this.\_toggleClass( element, keys, extra, true );

},

\_toggleClass: function( element, keys, extra, add ) {

add = ( typeof add === "boolean" ) ? add : extra;

var shift = ( typeof element === "string" || element === null ),

options = {

extra: shift ? keys : extra,

keys: shift ? element : keys,

element: shift ? this.element : element,

add: add

};

options.element.toggleClass( this.\_classes( options ), add );

return this;

},

\_on: function( suppressDisabledCheck, element, handlers ) {

var delegateElement;

var instance = this;

// No suppressDisabledCheck flag, shuffle arguments

if ( typeof suppressDisabledCheck !== "boolean" ) {

handlers = element;

element = suppressDisabledCheck;

suppressDisabledCheck = false;

}

// No element argument, shuffle and use this.element

if ( !handlers ) {

handlers = element;

element = this.element;

delegateElement = this.widget();

} else {

element = delegateElement = $( element );

this.bindings = this.bindings.add( element );

}

$.each( handlers, function( event, handler ) {

function handlerProxy() {

// Allow widgets to customize the disabled handling

// - disabled as an array instead of boolean

// - disabled class as method for disabling individual parts

if ( !suppressDisabledCheck &&

( instance.options.disabled === true ||

$( this ).hasClass( "ui-state-disabled" ) ) ) {

return;

}

return ( typeof handler === "string" ? instance[ handler ] : handler )

.apply( instance, arguments );

}

// Copy the guid so direct unbinding works

if ( typeof handler !== "string" ) {

handlerProxy.guid = handler.guid =

handler.guid || handlerProxy.guid || $.guid++;

}

var match = event.match( /^([\w:-]\*)\s\*(.\*)$/ );

var eventName = match[ 1 ] + instance.eventNamespace;

var selector = match[ 2 ];

if ( selector ) {

delegateElement.on( eventName, selector, handlerProxy );

} else {

element.on( eventName, handlerProxy );

}

} );

},

\_off: function( element, eventName ) {

eventName = ( eventName || "" ).split( " " ).join( this.eventNamespace + " " ) +

this.eventNamespace;

element.off( eventName ).off( eventName );

// Clear the stack to avoid memory leaks (#10056)

this.bindings = $( this.bindings.not( element ).get() );

this.focusable = $( this.focusable.not( element ).get() );

this.hoverable = $( this.hoverable.not( element ).get() );

},

\_delay: function( handler, delay ) {

function handlerProxy() {

return ( typeof handler === "string" ? instance[ handler ] : handler )

.apply( instance, arguments );

}

var instance = this;

return setTimeout( handlerProxy, delay || 0 );

},

\_hoverable: function( element ) {

this.hoverable = this.hoverable.add( element );

this.\_on( element, {

mouseenter: function( event ) {

this.\_addClass( $( event.currentTarget ), null, "ui-state-hover" );

},

mouseleave: function( event ) {

this.\_removeClass( $( event.currentTarget ), null, "ui-state-hover" );

}

} );

},

\_focusable: function( element ) {

this.focusable = this.focusable.add( element );

this.\_on( element, {

focusin: function( event ) {

this.\_addClass( $( event.currentTarget ), null, "ui-state-focus" );

},

focusout: function( event ) {

this.\_removeClass( $( event.currentTarget ), null, "ui-state-focus" );

}

} );

},

\_trigger: function( type, event, data ) {

var prop, orig;

var callback = this.options[ type ];

data = data || {};

event = $.Event( event );

event.type = ( type === this.widgetEventPrefix ?

type :

this.widgetEventPrefix + type ).toLowerCase();

// The original event may come from any element

// so we need to reset the target on the new event

event.target = this.element[ 0 ];

// Copy original event properties over to the new event

orig = event.originalEvent;

if ( orig ) {

for ( prop in orig ) {

if ( !( prop in event ) ) {

event[ prop ] = orig[ prop ];

}

}

}

this.element.trigger( event, data );

return !( $.isFunction( callback ) &&

callback.apply( this.element[ 0 ], [ event ].concat( data ) ) === false ||

event.isDefaultPrevented() );

}

};

$.each( { show: "fadeIn", hide: "fadeOut" }, function( method, defaultEffect ) {

$.Widget.prototype[ "\_" + method ] = function( element, options, callback ) {

if ( typeof options === "string" ) {

options = { effect: options };

}

var hasOptions;

var effectName = !options ?

method :

options === true || typeof options === "number" ?

defaultEffect :

options.effect || defaultEffect;

options = options || {};

if ( typeof options === "number" ) {

options = { duration: options };

}

hasOptions = !$.isEmptyObject( options );

options.complete = callback;

if ( options.delay ) {

element.delay( options.delay );

}

if ( hasOptions && $.effects && $.effects.effect[ effectName ] ) {

element[ method ]( options );

} else if ( effectName !== method && element[ effectName ] ) {

element[ effectName ]( options.duration, options.easing, callback );

} else {

element.queue( function( next ) {

$( this )[ method ]();

if ( callback ) {

callback.call( element[ 0 ] );

}

next();

} );

}

};

} );

var widget = $.widget;

/\*!

\* jQuery UI Position 1.12.1

\* http://jqueryui.com

\*

\* Copyright jQuery Foundation and other contributors

\* Released under the MIT license.

\* http://jquery.org/license

\*

\* http://api.jqueryui.com/position/

\*/

//>>label: Position

//>>group: Core

//>>description: Positions elements relative to other elements.

//>>docs: http://api.jqueryui.com/position/

//>>demos: http://jqueryui.com/position/

( function() {

var cachedScrollbarWidth,

max = Math.max,

abs = Math.abs,

rhorizontal = /left|center|right/,

rvertical = /top|center|bottom/,

roffset = /[\+\-]\d+(\.[\d]+)?%?/,

rposition = /^\w+/,

rpercent = /%$/,

\_position = $.fn.position;

function getOffsets( offsets, width, height ) {

return [

parseFloat( offsets[ 0 ] ) \* ( rpercent.test( offsets[ 0 ] ) ? width / 100 : 1 ),

parseFloat( offsets[ 1 ] ) \* ( rpercent.test( offsets[ 1 ] ) ? height / 100 : 1 )

];

}

function parseCss( element, property ) {

return parseInt( $.css( element, property ), 10 ) || 0;

}

function getDimensions( elem ) {

var raw = elem[ 0 ];

if ( raw.nodeType === 9 ) {

return {

width: elem.width(),

height: elem.height(),

offset: { top: 0, left: 0 }

};

}

if ( $.isWindow( raw ) ) {

return {

width: elem.width(),

height: elem.height(),

offset: { top: elem.scrollTop(), left: elem.scrollLeft() }

};

}

if ( raw.preventDefault ) {

return {

width: 0,

height: 0,

offset: { top: raw.pageY, left: raw.pageX }

};

}

return {

width: elem.outerWidth(),

height: elem.outerHeight(),

offset: elem.offset()

};

}

$.position = {

scrollbarWidth: function() {

if ( cachedScrollbarWidth !== undefined ) {

return cachedScrollbarWidth;

}

var w1, w2,

div = $( "<div " +

"style='display:block;position:absolute;width:50px;height:50px;overflow:hidden;'>" +

"<div style='height:100px;width:auto;'></div></div>" ),

innerDiv = div.children()[ 0 ];

$( "body" ).append( div );

w1 = innerDiv.offsetWidth;

div.css( "overflow", "scroll" );

w2 = innerDiv.offsetWidth;

if ( w1 === w2 ) {

w2 = div[ 0 ].clientWidth;

}

div.remove();

return ( cachedScrollbarWidth = w1 - w2 );

},

getScrollInfo: function( within ) {

var overflowX = within.isWindow || within.isDocument ? "" :

within.element.css( "overflow-x" ),

overflowY = within.isWindow || within.isDocument ? "" :

within.element.css( "overflow-y" ),

hasOverflowX = overflowX === "scroll" ||

( overflowX === "auto" && within.width < within.element[ 0 ].scrollWidth ),

hasOverflowY = overflowY === "scroll" ||

( overflowY === "auto" && within.height < within.element[ 0 ].scrollHeight );

return {

width: hasOverflowY ? $.position.scrollbarWidth() : 0,

height: hasOverflowX ? $.position.scrollbarWidth() : 0

};

},

getWithinInfo: function( element ) {

var withinElement = $( element || window ),

isWindow = $.isWindow( withinElement[ 0 ] ),

isDocument = !!withinElement[ 0 ] && withinElement[ 0 ].nodeType === 9,

hasOffset = !isWindow && !isDocument;

return {

element: withinElement,

isWindow: isWindow,

isDocument: isDocument,

offset: hasOffset ? $( element ).offset() : { left: 0, top: 0 },

scrollLeft: withinElement.scrollLeft(),

scrollTop: withinElement.scrollTop(),

width: withinElement.outerWidth(),

height: withinElement.outerHeight()

};

}

};

$.fn.position = function( options ) {

if ( !options || !options.of ) {

return \_position.apply( this, arguments );

}

// Make a copy, we don't want to modify arguments

options = $.extend( {}, options );

var atOffset, targetWidth, targetHeight, targetOffset, basePosition, dimensions,

target = $( options.of ),

within = $.position.getWithinInfo( options.within ),

scrollInfo = $.position.getScrollInfo( within ),

collision = ( options.collision || "flip" ).split( " " ),

offsets = {};

dimensions = getDimensions( target );

if ( target[ 0 ].preventDefault ) {

// Force left top to allow flipping

options.at = "left top";

}

targetWidth = dimensions.width;

targetHeight = dimensions.height;

targetOffset = dimensions.offset;

// Clone to reuse original targetOffset later

basePosition = $.extend( {}, targetOffset );

// Force my and at to have valid horizontal and vertical positions

// if a value is missing or invalid, it will be converted to center

$.each( [ "my", "at" ], function() {

var pos = ( options[ this ] || "" ).split( " " ),

horizontalOffset,

verticalOffset;

if ( pos.length === 1 ) {

pos = rhorizontal.test( pos[ 0 ] ) ?

pos.concat( [ "center" ] ) :

rvertical.test( pos[ 0 ] ) ?

[ "center" ].concat( pos ) :

[ "center", "center" ];

}

pos[ 0 ] = rhorizontal.test( pos[ 0 ] ) ? pos[ 0 ] : "center";

pos[ 1 ] = rvertical.test( pos[ 1 ] ) ? pos[ 1 ] : "center";

// Calculate offsets

horizontalOffset = roffset.exec( pos[ 0 ] );

verticalOffset = roffset.exec( pos[ 1 ] );

offsets[ this ] = [

horizontalOffset ? horizontalOffset[ 0 ] : 0,

verticalOffset ? verticalOffset[ 0 ] : 0

];

// Reduce to just the positions without the offsets

options[ this ] = [

rposition.exec( pos[ 0 ] )[ 0 ],

rposition.exec( pos[ 1 ] )[ 0 ]

];

} );

// Normalize collision option

if ( collision.length === 1 ) {

collision[ 1 ] = collision[ 0 ];

}

if ( options.at[ 0 ] === "right" ) {

basePosition.left += targetWidth;

} else if ( options.at[ 0 ] === "center" ) {

basePosition.left += targetWidth / 2;

}

if ( options.at[ 1 ] === "bottom" ) {

basePosition.top += targetHeight;

} else if ( options.at[ 1 ] === "center" ) {

basePosition.top += targetHeight / 2;

}

atOffset = getOffsets( offsets.at, targetWidth, targetHeight );

basePosition.left += atOffset[ 0 ];

basePosition.top += atOffset[ 1 ];

return this.each( function() {

var collisionPosition, using,

elem = $( this ),

elemWidth = elem.outerWidth(),

elemHeight = elem.outerHeight(),

marginLeft = parseCss( this, "marginLeft" ),

marginTop = parseCss( this, "marginTop" ),

collisionWidth = elemWidth + marginLeft + parseCss( this, "marginRight" ) +

scrollInfo.width,

collisionHeight = elemHeight + marginTop + parseCss( this, "marginBottom" ) +

scrollInfo.height,

position = $.extend( {}, basePosition ),

myOffset = getOffsets( offsets.my, elem.outerWidth(), elem.outerHeight() );

if ( options.my[ 0 ] === "right" ) {

position.left -= elemWidth;

} else if ( options.my[ 0 ] === "center" ) {

position.left -= elemWidth / 2;

}

if ( options.my[ 1 ] === "bottom" ) {

position.top -= elemHeight;

} else if ( options.my[ 1 ] === "center" ) {

position.top -= elemHeight / 2;

}

position.left += myOffset[ 0 ];

position.top += myOffset[ 1 ];

collisionPosition = {

marginLeft: marginLeft,

marginTop: marginTop

};

$.each( [ "left", "top" ], function( i, dir ) {

if ( $.ui.position[ collision[ i ] ] ) {

$.ui.position[ collision[ i ] ][ dir ]( position, {

targetWidth: targetWidth,

targetHeight: targetHeight,

elemWidth: elemWidth,

elemHeight: elemHeight,

collisionPosition: collisionPosition,

collisionWidth: collisionWidth,

collisionHeight: collisionHeight,

offset: [ atOffset[ 0 ] + myOffset[ 0 ], atOffset [ 1 ] + myOffset[ 1 ] ],

my: options.my,

at: options.at,

within: within,

elem: elem

} );

}

} );

if ( options.using ) {

// Adds feedback as second argument to using callback, if present

using = function( props ) {

var left = targetOffset.left - position.left,

right = left + targetWidth - elemWidth,

top = targetOffset.top - position.top,

bottom = top + targetHeight - elemHeight,

feedback = {

target: {

element: target,

left: targetOffset.left,

top: targetOffset.top,

width: targetWidth,

height: targetHeight

},

element: {

element: elem,

left: position.left,

top: position.top,

width: elemWidth,

height: elemHeight

},

horizontal: right < 0 ? "left" : left > 0 ? "right" : "center",

vertical: bottom < 0 ? "top" : top > 0 ? "bottom" : "middle"

};

if ( targetWidth < elemWidth && abs( left + right ) < targetWidth ) {

feedback.horizontal = "center";

}

if ( targetHeight < elemHeight && abs( top + bottom ) < targetHeight ) {

feedback.vertical = "middle";

}

if ( max( abs( left ), abs( right ) ) > max( abs( top ), abs( bottom ) ) ) {

feedback.important = "horizontal";

} else {

feedback.important = "vertical";

}

options.using.call( this, props, feedback );

};

}

elem.offset( $.extend( position, { using: using } ) );

} );

};

$.ui.position = {

fit: {

left: function( position, data ) {

var within = data.within,

withinOffset = within.isWindow ? within.scrollLeft : within.offset.left,

outerWidth = within.width,

collisionPosLeft = position.left - data.collisionPosition.marginLeft,

overLeft = withinOffset - collisionPosLeft,

overRight = collisionPosLeft + data.collisionWidth - outerWidth - withinOffset,

newOverRight;

// Element is wider than within

if ( data.collisionWidth > outerWidth ) {

// Element is initially over the left side of within

if ( overLeft > 0 && overRight <= 0 ) {

newOverRight = position.left + overLeft + data.collisionWidth - outerWidth -

withinOffset;

position.left += overLeft - newOverRight;

// Element is initially over right side of within

} else if ( overRight > 0 && overLeft <= 0 ) {

position.left = withinOffset;

// Element is initially over both left and right sides of within

} else {

if ( overLeft > overRight ) {

position.left = withinOffset + outerWidth - data.collisionWidth;

} else {

position.left = withinOffset;

}

}

// Too far left -> align with left edge

} else if ( overLeft > 0 ) {

position.left += overLeft;

// Too far right -> align with right edge

} else if ( overRight > 0 ) {

position.left -= overRight;

// Adjust based on position and margin

} else {

position.left = max( position.left - collisionPosLeft, position.left );

}

},

top: function( position, data ) {

var within = data.within,

withinOffset = within.isWindow ? within.scrollTop : within.offset.top,

outerHeight = data.within.height,

collisionPosTop = position.top - data.collisionPosition.marginTop,

overTop = withinOffset - collisionPosTop,

overBottom = collisionPosTop + data.collisionHeight - outerHeight - withinOffset,

newOverBottom;

// Element is taller than within

if ( data.collisionHeight > outerHeight ) {

// Element is initially over the top of within

if ( overTop > 0 && overBottom <= 0 ) {

newOverBottom = position.top + overTop + data.collisionHeight - outerHeight -

withinOffset;

position.top += overTop - newOverBottom;

// Element is initially over bottom of within

} else if ( overBottom > 0 && overTop <= 0 ) {

position.top = withinOffset;

// Element is initially over both top and bottom of within

} else {

if ( overTop > overBottom ) {

position.top = withinOffset + outerHeight - data.collisionHeight;

} else {

position.top = withinOffset;

}

}

// Too far up -> align with top

} else if ( overTop > 0 ) {

position.top += overTop;

// Too far down -> align with bottom edge

} else if ( overBottom > 0 ) {

position.top -= overBottom;

// Adjust based on position and margin

} else {

position.top = max( position.top - collisionPosTop, position.top );

}

}

},

flip: {

left: function( position, data ) {

var within = data.within,

withinOffset = within.offset.left + within.scrollLeft,

outerWidth = within.width,

offsetLeft = within.isWindow ? within.scrollLeft : within.offset.left,

collisionPosLeft = position.left - data.collisionPosition.marginLeft,

overLeft = collisionPosLeft - offsetLeft,

overRight = collisionPosLeft + data.collisionWidth - outerWidth - offsetLeft,

myOffset = data.my[ 0 ] === "left" ?

-data.elemWidth :

data.my[ 0 ] === "right" ?

data.elemWidth :

0,

atOffset = data.at[ 0 ] === "left" ?

data.targetWidth :

data.at[ 0 ] === "right" ?

-data.targetWidth :

0,

offset = -2 \* data.offset[ 0 ],

newOverRight,

newOverLeft;

if ( overLeft < 0 ) {

newOverRight = position.left + myOffset + atOffset + offset + data.collisionWidth -

outerWidth - withinOffset;

if ( newOverRight < 0 || newOverRight < abs( overLeft ) ) {

position.left += myOffset + atOffset + offset;

}

} else if ( overRight > 0 ) {

newOverLeft = position.left - data.collisionPosition.marginLeft + myOffset +

atOffset + offset - offsetLeft;

if ( newOverLeft > 0 || abs( newOverLeft ) < overRight ) {

position.left += myOffset + atOffset + offset;

}

}

},

top: function( position, data ) {

var within = data.within,

withinOffset = within.offset.top + within.scrollTop,

outerHeight = within.height,

offsetTop = within.isWindow ? within.scrollTop : within.offset.top,

collisionPosTop = position.top - data.collisionPosition.marginTop,

overTop = collisionPosTop - offsetTop,

overBottom = collisionPosTop + data.collisionHeight - outerHeight - offsetTop,

top = data.my[ 1 ] === "top",

myOffset = top ?

-data.elemHeight :

data.my[ 1 ] === "bottom" ?

data.elemHeight :

0,

atOffset = data.at[ 1 ] === "top" ?

data.targetHeight :

data.at[ 1 ] === "bottom" ?

-data.targetHeight :

0,

offset = -2 \* data.offset[ 1 ],

newOverTop,

newOverBottom;

if ( overTop < 0 ) {

newOverBottom = position.top + myOffset + atOffset + offset + data.collisionHeight -

outerHeight - withinOffset;

if ( newOverBottom < 0 || newOverBottom < abs( overTop ) ) {

position.top += myOffset + atOffset + offset;

}

} else if ( overBottom > 0 ) {

newOverTop = position.top - data.collisionPosition.marginTop + myOffset + atOffset +

offset - offsetTop;

if ( newOverTop > 0 || abs( newOverTop ) < overBottom ) {

position.top += myOffset + atOffset + offset;

}

}

}

},

flipfit: {

left: function() {

$.ui.position.flip.left.apply( this, arguments );

$.ui.position.fit.left.apply( this, arguments );

},

top: function() {

$.ui.position.flip.top.apply( this, arguments );

$.ui.position.fit.top.apply( this, arguments );

}

}

};

} )();

var position = $.ui.position;

/\*!

\* jQuery UI Keycode 1.12.1

\* http://jqueryui.com

\*

\* Copyright jQuery Foundation and other contributors

\* Released under the MIT license.

\* http://jquery.org/license

\*/

//>>label: Keycode

//>>group: Core

//>>description: Provide keycodes as keynames

//>>docs: http://api.jqueryui.com/jQuery.ui.keyCode/

var keycode = $.ui.keyCode = {

BACKSPACE: 8,

COMMA: 188,

DELETE: 46,

DOWN: 40,

END: 35,

ENTER: 13,

ESCAPE: 27,

HOME: 36,

LEFT: 37,

PAGE\_DOWN: 34,

PAGE\_UP: 33,

PERIOD: 190,

RIGHT: 39,

SPACE: 32,

TAB: 9,

UP: 38

};

/\*!

\* jQuery UI Unique ID 1.12.1

\* http://jqueryui.com

\*

\* Copyright jQuery Foundation and other contributors

\* Released under the MIT license.

\* http://jquery.org/license

\*/

//>>label: uniqueId

//>>group: Core

//>>description: Functions to generate and remove uniqueId's

//>>docs: http://api.jqueryui.com/uniqueId/

var uniqueId = $.fn.extend( {

uniqueId: ( function() {

var uuid = 0;

return function() {

return this.each( function() {

if ( !this.id ) {

this.id = "ui-id-" + ( ++uuid );

}

} );

};

} )(),

removeUniqueId: function() {

return this.each( function() {

if ( /^ui-id-\d+$/.test( this.id ) ) {

$( this ).removeAttr( "id" );

}

} );

}

} );

var safeActiveElement = $.ui.safeActiveElement = function( document ) {

var activeElement;

// Support: IE 9 only

// IE9 throws an "Unspecified error" accessing document.activeElement from an <iframe>

try {

activeElement = document.activeElement;

} catch ( error ) {

activeElement = document.body;

}

// Support: IE 9 - 11 only

// IE may return null instead of an element

// Interestingly, this only seems to occur when NOT in an iframe

if ( !activeElement ) {

activeElement = document.body;

}

// Support: IE 11 only

// IE11 returns a seemingly empty object in some cases when accessing

// document.activeElement from an <iframe>

if ( !activeElement.nodeName ) {

activeElement = document.body;

}

return activeElement;

};

/\*!

\* jQuery UI Menu 1.12.1

\* http://jqueryui.com

\*

\* Copyright jQuery Foundation and other contributors

\* Released under the MIT license.

\* http://jquery.org/license

\*/

//>>label: Menu

//>>group: Widgets

//>>description: Creates nestable menus.

//>>docs: http://api.jqueryui.com/menu/

//>>demos: http://jqueryui.com/menu/

//>>css.structure: ../../themes/base/core.css

//>>css.structure: ../../themes/base/menu.css

//>>css.theme: ../../themes/base/theme.css

var widgetsMenu = $.widget( "ui.menu", {

version: "1.12.1",

defaultElement: "<ul>",

delay: 300,

options: {

icons: {

submenu: "ui-icon-caret-1-e"

},

items: "> \*",

menus: "ul",

position: {

my: "left top",

at: "right top"

},

role: "menu",

// Callbacks

blur: null,

focus: null,

select: null

},

\_create: function() {

this.activeMenu = this.element;

// Flag used to prevent firing of the click handler

// as the event bubbles up through nested menus

this.mouseHandled = false;

this.element

.uniqueId()

.attr( {

role: this.options.role,

tabIndex: 0

} );

this.\_addClass( "ui-menu", "ui-widget ui-widget-content" );

this.\_on( {

// Prevent focus from sticking to links inside menu after clicking

// them (focus should always stay on UL during navigation).

"mousedown .ui-menu-item": function( event ) {

event.preventDefault();

},

"click .ui-menu-item": function( event ) {

var target = $( event.target );

var active = $( $.ui.safeActiveElement( this.document[ 0 ] ) );

if ( !this.mouseHandled && target.not( ".ui-state-disabled" ).length ) {

this.select( event );

// Only set the mouseHandled flag if the event will bubble, see #9469.

if ( !event.isPropagationStopped() ) {

this.mouseHandled = true;

}

// Open submenu on click

if ( target.has( ".ui-menu" ).length ) {

this.expand( event );

} else if ( !this.element.is( ":focus" ) &&

active.closest( ".ui-menu" ).length ) {

// Redirect focus to the menu

this.element.trigger( "focus", [ true ] );

// If the active item is on the top level, let it stay active.

// Otherwise, blur the active item since it is no longer visible.

if ( this.active && this.active.parents( ".ui-menu" ).length === 1 ) {

clearTimeout( this.timer );

}

}

}

},

"mouseenter .ui-menu-item": function( event ) {

// Ignore mouse events while typeahead is active, see #10458.

// Prevents focusing the wrong item when typeahead causes a scroll while the mouse

// is over an item in the menu

if ( this.previousFilter ) {

return;

}

var actualTarget = $( event.target ).closest( ".ui-menu-item" ),

target = $( event.currentTarget );

// Ignore bubbled events on parent items, see #11641

if ( actualTarget[ 0 ] !== target[ 0 ] ) {

return;

}

// Remove ui-state-active class from siblings of the newly focused menu item

// to avoid a jump caused by adjacent elements both having a class with a border

this.\_removeClass( target.siblings().children( ".ui-state-active" ),

null, "ui-state-active" );

this.focus( event, target );

},

mouseleave: "collapseAll",

"mouseleave .ui-menu": "collapseAll",

focus: function( event, keepActiveItem ) {

// If there's already an active item, keep it active

// If not, activate the first item

var item = this.active || this.element.find( this.options.items ).eq( 0 );

if ( !keepActiveItem ) {

this.focus( event, item );

}

},

blur: function( event ) {

this.\_delay( function() {

var notContained = !$.contains(

this.element[ 0 ],

$.ui.safeActiveElement( this.document[ 0 ] )

);

if ( notContained ) {

this.collapseAll( event );

}

} );

},

keydown: "\_keydown"

} );

this.refresh();

// Clicks outside of a menu collapse any open menus

this.\_on( this.document, {

click: function( event ) {

if ( this.\_closeOnDocumentClick( event ) ) {

this.collapseAll( event );

}

// Reset the mouseHandled flag

this.mouseHandled = false;

}

} );

},

\_destroy: function() {

var items = this.element.find( ".ui-menu-item" )

.removeAttr( "role aria-disabled" ),

submenus = items.children( ".ui-menu-item-wrapper" )

.removeUniqueId()

.removeAttr( "tabIndex role aria-haspopup" );

// Destroy (sub)menus

this.element

.removeAttr( "aria-activedescendant" )

.find( ".ui-menu" ).addBack()

.removeAttr( "role aria-labelledby aria-expanded aria-hidden aria-disabled " +

"tabIndex" )

.removeUniqueId()

.show();

submenus.children().each( function() {

var elem = $( this );

if ( elem.data( "ui-menu-submenu-caret" ) ) {

elem.remove();

}

} );

},

\_keydown: function( event ) {

var match, prev, character, skip,

preventDefault = true;

switch ( event.keyCode ) {

case $.ui.keyCode.PAGE\_UP:

this.previousPage( event );

break;

case $.ui.keyCode.PAGE\_DOWN:

this.nextPage( event );

break;

case $.ui.keyCode.HOME:

this.\_move( "first", "first", event );

break;

case $.ui.keyCode.END:

this.\_move( "last", "last", event );

break;

case $.ui.keyCode.UP:

this.previous( event );

break;

case $.ui.keyCode.DOWN:

this.next( event );

break;

case $.ui.keyCode.LEFT:

this.collapse( event );

break;

case $.ui.keyCode.RIGHT:

if ( this.active && !this.active.is( ".ui-state-disabled" ) ) {

this.expand( event );

}

break;

case $.ui.keyCode.ENTER:

case $.ui.keyCode.SPACE:

this.\_activate( event );

break;

case $.ui.keyCode.ESCAPE:

this.collapse( event );

break;

default:

preventDefault = false;

prev = this.previousFilter || "";

skip = false;

// Support number pad values

character = event.keyCode >= 96 && event.keyCode <= 105 ?

( event.keyCode - 96 ).toString() : String.fromCharCode( event.keyCode );

clearTimeout( this.filterTimer );

if ( character === prev ) {

skip = true;

} else {

character = prev + character;

}

match = this.\_filterMenuItems( character );

match = skip && match.index( this.active.next() ) !== -1 ?

this.active.nextAll( ".ui-menu-item" ) :

match;

// If no matches on the current filter, reset to the last character pressed

// to move down the menu to the first item that starts with that character

if ( !match.length ) {

character = String.fromCharCode( event.keyCode );

match = this.\_filterMenuItems( character );

}

if ( match.length ) {

this.focus( event, match );

this.previousFilter = character;

this.filterTimer = this.\_delay( function() {

delete this.previousFilter;

}, 1000 );

} else {

delete this.previousFilter;

}

}

if ( preventDefault ) {

event.preventDefault();

}

},

\_activate: function( event ) {

if ( this.active && !this.active.is( ".ui-state-disabled" ) ) {

if ( this.active.children( "[aria-haspopup='true']" ).length ) {

this.expand( event );

} else {

this.select( event );

}

}

},

refresh: function() {

var menus, items, newSubmenus, newItems, newWrappers,

that = this,

icon = this.options.icons.submenu,

submenus = this.element.find( this.options.menus );

this.\_toggleClass( "ui-menu-icons", null, !!this.element.find( ".ui-icon" ).length );

// Initialize nested menus

newSubmenus = submenus.filter( ":not(.ui-menu)" )

.hide()

.attr( {

role: this.options.role,

"aria-hidden": "true",

"aria-expanded": "false"

} )

.each( function() {

var menu = $( this ),

item = menu.prev(),

submenuCaret = $( "<span>" ).data( "ui-menu-submenu-caret", true );

that.\_addClass( submenuCaret, "ui-menu-icon", "ui-icon " + icon );

item

.attr( "aria-haspopup", "true" )

.prepend( submenuCaret );

menu.attr( "aria-labelledby", item.attr( "id" ) );

} );

this.\_addClass( newSubmenus, "ui-menu", "ui-widget ui-widget-content ui-front" );

menus = submenus.add( this.element );

items = menus.find( this.options.items );

// Initialize menu-items containing spaces and/or dashes only as dividers

items.not( ".ui-menu-item" ).each( function() {

var item = $( this );

if ( that.\_isDivider( item ) ) {

that.\_addClass( item, "ui-menu-divider", "ui-widget-content" );

}

} );

// Don't refresh list items that are already adapted

newItems = items.not( ".ui-menu-item, .ui-menu-divider" );

newWrappers = newItems.children()

.not( ".ui-menu" )

.uniqueId()

.attr( {

tabIndex: -1,

role: this.\_itemRole()

} );

this.\_addClass( newItems, "ui-menu-item" )

.\_addClass( newWrappers, "ui-menu-item-wrapper" );

// Add aria-disabled attribute to any disabled menu item

items.filter( ".ui-state-disabled" ).attr( "aria-disabled", "true" );

// If the active item has been removed, blur the menu

if ( this.active && !$.contains( this.element[ 0 ], this.active[ 0 ] ) ) {

this.blur();

}

},

\_itemRole: function() {

return {

menu: "menuitem",

listbox: "option"

}[ this.options.role ];

},

\_setOption: function( key, value ) {

if ( key === "icons" ) {

var icons = this.element.find( ".ui-menu-icon" );

this.\_removeClass( icons, null, this.options.icons.submenu )

.\_addClass( icons, null, value.submenu );

}

this.\_super( key, value );

},

\_setOptionDisabled: function( value ) {

this.\_super( value );

this.element.attr( "aria-disabled", String( value ) );

this.\_toggleClass( null, "ui-state-disabled", !!value );

},

focus: function( event, item ) {

var nested, focused, activeParent;

this.blur( event, event && event.type === "focus" );

this.\_scrollIntoView( item );

this.active = item.first();

focused = this.active.children( ".ui-menu-item-wrapper" );

this.\_addClass( focused, null, "ui-state-active" );

// Only update aria-activedescendant if there's a role

// otherwise we assume focus is managed elsewhere

if ( this.options.role ) {

this.element.attr( "aria-activedescendant", focused.attr( "id" ) );

}

// Highlight active parent menu item, if any

activeParent = this.active

.parent()

.closest( ".ui-menu-item" )

.children( ".ui-menu-item-wrapper" );

this.\_addClass( activeParent, null, "ui-state-active" );

if ( event && event.type === "keydown" ) {

this.\_close();

} else {

this.timer = this.\_delay( function() {

this.\_close();

}, this.delay );

}

nested = item.children( ".ui-menu" );

if ( nested.length && event && ( /^mouse/.test( event.type ) ) ) {

this.\_startOpening( nested );

}

this.activeMenu = item.parent();

this.\_trigger( "focus", event, { item: item } );

},

\_scrollIntoView: function( item ) {

var borderTop, paddingTop, offset, scroll, elementHeight, itemHeight;

if ( this.\_hasScroll() ) {

borderTop = parseFloat( $.css( this.activeMenu[ 0 ], "borderTopWidth" ) ) || 0;

paddingTop = parseFloat( $.css( this.activeMenu[ 0 ], "paddingTop" ) ) || 0;

offset = item.offset().top - this.activeMenu.offset().top - borderTop - paddingTop;

scroll = this.activeMenu.scrollTop();

elementHeight = this.activeMenu.height();

itemHeight = item.outerHeight();

if ( offset < 0 ) {

this.activeMenu.scrollTop( scroll + offset );

} else if ( offset + itemHeight > elementHeight ) {

this.activeMenu.scrollTop( scroll + offset - elementHeight + itemHeight );

}

}

},

blur: function( event, fromFocus ) {

if ( !fromFocus ) {

clearTimeout( this.timer );

}

if ( !this.active ) {

return;

}

this.\_removeClass( this.active.children( ".ui-menu-item-wrapper" ),

null, "ui-state-active" );

this.\_trigger( "blur", event, { item: this.active } );

this.active = null;

},

\_startOpening: function( submenu ) {

clearTimeout( this.timer );

// Don't open if already open fixes a Firefox bug that caused a .5 pixel

// shift in the submenu position when mousing over the caret icon

if ( submenu.attr( "aria-hidden" ) !== "true" ) {

return;

}

this.timer = this.\_delay( function() {

this.\_close();

this.\_open( submenu );

}, this.delay );

},

\_open: function( submenu ) {

var position = $.extend( {

of: this.active

}, this.options.position );

clearTimeout( this.timer );

this.element.find( ".ui-menu" ).not( submenu.parents( ".ui-menu" ) )

.hide()

.attr( "aria-hidden", "true" );

submenu

.show()

.removeAttr( "aria-hidden" )

.attr( "aria-expanded", "true" )

.position( position );

},

collapseAll: function( event, all ) {

clearTimeout( this.timer );

this.timer = this.\_delay( function() {

// If we were passed an event, look for the submenu that contains the event

var currentMenu = all ? this.element :

$( event && event.target ).closest( this.element.find( ".ui-menu" ) );

// If we found no valid submenu ancestor, use the main menu to close all

// sub menus anyway

if ( !currentMenu.length ) {

currentMenu = this.element;

}

this.\_close( currentMenu );

this.blur( event );

// Work around active item staying active after menu is blurred

this.\_removeClass( currentMenu.find( ".ui-state-active" ), null, "ui-state-active" );

this.activeMenu = currentMenu;

}, this.delay );

},

// With no arguments, closes the currently active menu - if nothing is active

// it closes all menus. If passed an argument, it will search for menus BELOW

\_close: function( startMenu ) {

if ( !startMenu ) {

startMenu = this.active ? this.active.parent() : this.element;

}

startMenu.find( ".ui-menu" )

.hide()

.attr( "aria-hidden", "true" )

.attr( "aria-expanded", "false" );

},

\_closeOnDocumentClick: function( event ) {

return !$( event.target ).closest( ".ui-menu" ).length;

},

\_isDivider: function( item ) {

// Match hyphen, em dash, en dash

return !/[^\-\u2014\u2013\s]/.test( item.text() );

},

collapse: function( event ) {

var newItem = this.active &&

this.active.parent().closest( ".ui-menu-item", this.element );

if ( newItem && newItem.length ) {

this.\_close();

this.focus( event, newItem );

}

},

expand: function( event ) {

var newItem = this.active &&

this.active

.children( ".ui-menu " )

.find( this.options.items )

.first();

if ( newItem && newItem.length ) {

this.\_open( newItem.parent() );

// Delay so Firefox will not hide activedescendant change in expanding submenu from AT

this.\_delay( function() {

this.focus( event, newItem );

} );

}

},

next: function( event ) {

this.\_move( "next", "first", event );

},

previous: function( event ) {

this.\_move( "prev", "last", event );

},

isFirstItem: function() {

return this.active && !this.active.prevAll( ".ui-menu-item" ).length;

},

isLastItem: function() {

return this.active && !this.active.nextAll( ".ui-menu-item" ).length;

},

\_move: function( direction, filter, event ) {

var next;

if ( this.active ) {

if ( direction === "first" || direction === "last" ) {

next = this.active

[ direction === "first" ? "prevAll" : "nextAll" ]( ".ui-menu-item" )

.eq( -1 );

} else {

next = this.active

[ direction + "All" ]( ".ui-menu-item" )

.eq( 0 );

}

}

if ( !next || !next.length || !this.active ) {

next = this.activeMenu.find( this.options.items )[ filter ]();

}

this.focus( event, next );

},

nextPage: function( event ) {

var item, base, height;

if ( !this.active ) {

this.next( event );

return;

}

if ( this.isLastItem() ) {

return;

}

if ( this.\_hasScroll() ) {

base = this.active.offset().top;

height = this.element.height();

this.active.nextAll( ".ui-menu-item" ).each( function() {

item = $( this );

return item.offset().top - base - height < 0;

} );

this.focus( event, item );

} else {

this.focus( event, this.activeMenu.find( this.options.items )

[ !this.active ? "first" : "last" ]() );

}

},

previousPage: function( event ) {

var item, base, height;

if ( !this.active ) {

this.next( event );

return;

}

if ( this.isFirstItem() ) {

return;

}

if ( this.\_hasScroll() ) {

base = this.active.offset().top;

height = this.element.height();

this.active.prevAll( ".ui-menu-item" ).each( function() {

item = $( this );

return item.offset().top - base + height > 0;

} );

this.focus( event, item );

} else {

this.focus( event, this.activeMenu.find( this.options.items ).first() );

}

},

\_hasScroll: function() {

return this.element.outerHeight() < this.element.prop( "scrollHeight" );

},

select: function( event ) {

// TODO: It should never be possible to not have an active item at this

// point, but the tests don't trigger mouseenter before click.

this.active = this.active || $( event.target ).closest( ".ui-menu-item" );

var ui = { item: this.active };

if ( !this.active.has( ".ui-menu" ).length ) {

this.collapseAll( event, true );

}

this.\_trigger( "select", event, ui );

},

\_filterMenuItems: function( character ) {

var escapedCharacter = character.replace( /[\-\[\]{}()\*+?.,\\\^$|#\s]/g, "\\$&" ),

regex = new RegExp( "^" + escapedCharacter, "i" );

return this.activeMenu

.find( this.options.items )

// Only match on items, not dividers or other content (#10571)

.filter( ".ui-menu-item" )

.filter( function() {

return regex.test(

$.trim( $( this ).children( ".ui-menu-item-wrapper" ).text() ) );

} );

}

} );

/\*!

\* jQuery UI Autocomplete 1.12.1

\* http://jqueryui.com

\*

\* Copyright jQuery Foundation and other contributors

\* Released under the MIT license.

\* http://jquery.org/license

\*/

//>>label: Autocomplete

//>>group: Widgets

//>>description: Lists suggested words as the user is typing.

//>>docs: http://api.jqueryui.com/autocomplete/

//>>demos: http://jqueryui.com/autocomplete/

//>>css.structure: ../../themes/base/core.css

//>>css.structure: ../../themes/base/autocomplete.css

//>>css.theme: ../../themes/base/theme.css

$.widget( "ui.autocomplete", {

version: "1.12.1",

defaultElement: "<input>",

options: {

appendTo: null,

autoFocus: false,

delay: 300,

minLength: 1,

position: {

my: "left top",

at: "left bottom",

collision: "none"

},

source: null,

// Callbacks

change: null,

close: null,

focus: null,

open: null,

response: null,

search: null,

select: null

},

requestIndex: 0,

pending: 0,

\_create: function() {

// Some browsers only repeat keydown events, not keypress events,

// so we use the suppressKeyPress flag to determine if we've already

// handled the keydown event. #7269

// Unfortunately the code for & in keypress is the same as the up arrow,

// so we use the suppressKeyPressRepeat flag to avoid handling keypress

// events when we know the keydown event was used to modify the

// search term. #7799

var suppressKeyPress, suppressKeyPressRepeat, suppressInput,

nodeName = this.element[ 0 ].nodeName.toLowerCase(),

isTextarea = nodeName === "textarea",

isInput = nodeName === "input";

// Textareas are always multi-line

// Inputs are always single-line, even if inside a contentEditable element

// IE also treats inputs as contentEditable

// All other element types are determined by whether or not they're contentEditable

this.isMultiLine = isTextarea || !isInput && this.\_isContentEditable( this.element );

this.valueMethod = this.element[ isTextarea || isInput ? "val" : "text" ];

this.isNewMenu = true;

this.\_addClass( "ui-autocomplete-input" );

this.element.attr( "autocomplete", "off" );

this.\_on( this.element, {

keydown: function( event ) {

if ( this.element.prop( "readOnly" ) ) {

suppressKeyPress = true;

suppressInput = true;

suppressKeyPressRepeat = true;

return;

}

suppressKeyPress = false;

suppressInput = false;

suppressKeyPressRepeat = false;

var keyCode = $.ui.keyCode;

switch ( event.keyCode ) {

case keyCode.PAGE\_UP:

suppressKeyPress = true;

this.\_move( "previousPage", event );

break;

case keyCode.PAGE\_DOWN:

suppressKeyPress = true;

this.\_move( "nextPage", event );

break;

case keyCode.UP:

suppressKeyPress = true;

this.\_keyEvent( "previous", event );

break;

case keyCode.DOWN:

suppressKeyPress = true;

this.\_keyEvent( "next", event );

break;

case keyCode.ENTER:

// when menu is open and has focus

if ( this.menu.active ) {

// #6055 - Opera still allows the keypress to occur

// which causes forms to submit

suppressKeyPress = true;

event.preventDefault();

this.menu.select( event );

}

break;

case keyCode.TAB:

if ( this.menu.active ) {

this.menu.select( event );

}

break;

case keyCode.ESCAPE:

if ( this.menu.element.is( ":visible" ) ) {

if ( !this.isMultiLine ) {

this.\_value( this.term );

}

this.close( event );

// Different browsers have different default behavior for escape

// Single press can mean undo or clear

// Double press in IE means clear the whole form

event.preventDefault();

}

break;

default:

suppressKeyPressRepeat = true;

// search timeout should be triggered before the input value is changed

this.\_searchTimeout( event );

break;

}

},

keypress: function( event ) {

if ( suppressKeyPress ) {

suppressKeyPress = false;

if ( !this.isMultiLine || this.menu.element.is( ":visible" ) ) {

event.preventDefault();

}

return;

}

if ( suppressKeyPressRepeat ) {

return;

}

// Replicate some key handlers to allow them to repeat in Firefox and Opera

var keyCode = $.ui.keyCode;

switch ( event.keyCode ) {

case keyCode.PAGE\_UP:

this.\_move( "previousPage", event );

break;

case keyCode.PAGE\_DOWN:

this.\_move( "nextPage", event );

break;

case keyCode.UP:

this.\_keyEvent( "previous", event );

break;

case keyCode.DOWN:

this.\_keyEvent( "next", event );

break;

}

},

input: function( event ) {

if ( suppressInput ) {

suppressInput = false;

event.preventDefault();

return;

}

this.\_searchTimeout( event );

},

focus: function() {

this.selectedItem = null;

this.previous = this.\_value();

},

blur: function( event ) {

if ( this.cancelBlur ) {

delete this.cancelBlur;

return;

}

clearTimeout( this.searching );

this.close( event );

this.\_change( event );

}

} );

this.\_initSource();

this.menu = $( "<ul>" )

.appendTo( this.\_appendTo() )

.menu( {

// disable ARIA support, the live region takes care of that

role: null

} )

.hide()

.menu( "instance" );

this.\_addClass( this.menu.element, "ui-autocomplete", "ui-front" );

this.\_on( this.menu.element, {

mousedown: function( event ) {

// prevent moving focus out of the text field

event.preventDefault();

// IE doesn't prevent moving focus even with event.preventDefault()

// so we set a flag to know when we should ignore the blur event

this.cancelBlur = true;

this.\_delay( function() {

delete this.cancelBlur;

// Support: IE 8 only

// Right clicking a menu item or selecting text from the menu items will

// result in focus moving out of the input. However, we've already received

// and ignored the blur event because of the cancelBlur flag set above. So

// we restore focus to ensure that the menu closes properly based on the user's

// next actions.

if ( this.element[ 0 ] !== $.ui.safeActiveElement( this.document[ 0 ] ) ) {

this.element.trigger( "focus" );

}

} );

},

menufocus: function( event, ui ) {

var label, item;

// support: Firefox

// Prevent accidental activation of menu items in Firefox (#7024 #9118)

if ( this.isNewMenu ) {

this.isNewMenu = false;

if ( event.originalEvent && /^mouse/.test( event.originalEvent.type ) ) {

this.menu.blur();

this.document.one( "mousemove", function() {

$( event.target ).trigger( event.originalEvent );

} );

return;

}

}

item = ui.item.data( "ui-autocomplete-item" );

if ( false !== this.\_trigger( "focus", event, { item: item } ) ) {

// use value to match what will end up in the input, if it was a key event

if ( event.originalEvent && /^key/.test( event.originalEvent.type ) ) {

this.\_value( item.value );

}

}

// Announce the value in the liveRegion

label = ui.item.attr( "aria-label" ) || item.value;

if ( label && $.trim( label ).length ) {

this.liveRegion.children().hide();

$( "<div>" ).text( label ).appendTo( this.liveRegion );

}

},

menuselect: function( event, ui ) {

var item = ui.item.data( "ui-autocomplete-item" ),

previous = this.previous;

// Only trigger when focus was lost (click on menu)

if ( this.element[ 0 ] !== $.ui.safeActiveElement( this.document[ 0 ] ) ) {

this.element.trigger( "focus" );

this.previous = previous;

// #6109 - IE triggers two focus events and the second

// is asynchronous, so we need to reset the previous

// term synchronously and asynchronously :-(

this.\_delay( function() {

this.previous = previous;

this.selectedItem = item;

} );

}

if ( false !== this.\_trigger( "select", event, { item: item } ) ) {

this.\_value( item.value );

}

// reset the term after the select event

// this allows custom select handling to work properly

this.term = this.\_value();

this.close( event );

this.selectedItem = item;

}

} );

this.liveRegion = $( "<div>", {

role: "status",

"aria-live": "assertive",

"aria-relevant": "additions"

} )

.appendTo( this.document[ 0 ].body );

this.\_addClass( this.liveRegion, null, "ui-helper-hidden-accessible" );

// Turning off autocomplete prevents the browser from remembering the

// value when navigating through history, so we re-enable autocomplete

// if the page is unloaded before the widget is destroyed. #7790

this.\_on( this.window, {

beforeunload: function() {

this.element.removeAttr( "autocomplete" );

}

} );

},

\_destroy: function() {

clearTimeout( this.searching );

this.element.removeAttr( "autocomplete" );

this.menu.element.remove();

this.liveRegion.remove();

},

\_setOption: function( key, value ) {

this.\_super( key, value );

if ( key === "source" ) {

this.\_initSource();

}

if ( key === "appendTo" ) {

this.menu.element.appendTo( this.\_appendTo() );

}

if ( key === "disabled" && value && this.xhr ) {

this.xhr.abort();

}

},

\_isEventTargetInWidget: function( event ) {

var menuElement = this.menu.element[ 0 ];

return event.target === this.element[ 0 ] ||

event.target === menuElement ||

$.contains( menuElement, event.target );

},

\_closeOnClickOutside: function( event ) {

if ( !this.\_isEventTargetInWidget( event ) ) {

this.close();

}

},

\_appendTo: function() {

var element = this.options.appendTo;

if ( element ) {

element = element.jquery || element.nodeType ?

$( element ) :

this.document.find( element ).eq( 0 );

}

if ( !element || !element[ 0 ] ) {

element = this.element.closest( ".ui-front, dialog" );

}

if ( !element.length ) {

element = this.document[ 0 ].body;

}

return element;

},

\_initSource: function() {

var array, url,

that = this;

if ( $.isArray( this.options.source ) ) {

array = this.options.source;

this.source = function( request, response ) {

response( $.ui.autocomplete.filter( array, request.term ) );

};

} else if ( typeof this.options.source === "string" ) {

url = this.options.source;

this.source = function( request, response ) {

if ( that.xhr ) {

that.xhr.abort();

}

that.xhr = $.ajax( {

url: url,

data: request,

dataType: "json",

success: function( data ) {

response( data );

},

error: function() {

response( [] );

}

} );

};

} else {

this.source = this.options.source;

}

},

\_searchTimeout: function( event ) {

clearTimeout( this.searching );

this.searching = this.\_delay( function() {

// Search if the value has changed, or if the user retypes the same value (see #7434)

var equalValues = this.term === this.\_value(),

menuVisible = this.menu.element.is( ":visible" ),

modifierKey = event.altKey || event.ctrlKey || event.metaKey || event.shiftKey;

if ( !equalValues || ( equalValues && !menuVisible && !modifierKey ) ) {

this.selectedItem = null;

this.search( null, event );

}

}, this.options.delay );

},

search: function( value, event ) {

value = value != null ? value : this.\_value();

// Always save the actual value, not the one passed as an argument

this.term = this.\_value();

if ( value.length < this.options.minLength ) {

return this.close( event );

}

if ( this.\_trigger( "search", event ) === false ) {

return;

}

return this.\_search( value );

},

\_search: function( value ) {

this.pending++;

this.\_addClass( "ui-autocomplete-loading" );

this.cancelSearch = false;

this.source( { term: value }, this.\_response() );

},

\_response: function() {

var index = ++this.requestIndex;

return $.proxy( function( content ) {

if ( index === this.requestIndex ) {

this.\_\_response( content );

}

this.pending--;

if ( !this.pending ) {

this.\_removeClass( "ui-autocomplete-loading" );

}

}, this );

},

\_\_response: function( content ) {

if ( content ) {

content = this.\_normalize( content );

}

this.\_trigger( "response", null, { content: content } );

if ( !this.options.disabled && content && content.length && !this.cancelSearch ) {

this.\_suggest( content );

this.\_trigger( "open" );

} else {

// use .\_close() instead of .close() so we don't cancel future searches

this.\_close();

}

},

close: function( event ) {

this.cancelSearch = true;

this.\_close( event );

},

\_close: function( event ) {

// Remove the handler that closes the menu on outside clicks

this.\_off( this.document, "mousedown" );

if ( this.menu.element.is( ":visible" ) ) {

this.menu.element.hide();

this.menu.blur();

this.isNewMenu = true;

this.\_trigger( "close", event );

}

},

\_change: function( event ) {

if ( this.previous !== this.\_value() ) {

this.\_trigger( "change", event, { item: this.selectedItem } );

}

},

\_normalize: function( items ) {

// assume all items have the right format when the first item is complete

if ( items.length && items[ 0 ].label && items[ 0 ].value ) {

return items;

}

return $.map( items, function( item ) {

if ( typeof item === "string" ) {

return {

label: item,

value: item

};

}

return $.extend( {}, item, {

label: item.label || item.value,

value: item.value || item.label

} );

} );

},

\_suggest: function( items ) {

var ul = this.menu.element.empty();

this.\_renderMenu( ul, items );

this.isNewMenu = true;

this.menu.refresh();

// Size and position menu

ul.show();

this.\_resizeMenu();

ul.position( $.extend( {

of: this.element

}, this.options.position ) );

if ( this.options.autoFocus ) {

this.menu.next();

}

// Listen for interactions outside of the widget (#6642)

this.\_on( this.document, {

mousedown: "\_closeOnClickOutside"

} );

},

\_resizeMenu: function() {

var ul = this.menu.element;

ul.outerWidth( Math.max(

// Firefox wraps long text (possibly a rounding bug)

// so we add 1px to avoid the wrapping (#7513)

ul.width( "" ).outerWidth() + 1,

this.element.outerWidth()

) );

},

\_renderMenu: function( ul, items ) {

var that = this;

$.each( items, function( index, item ) {

that.\_renderItemData( ul, item );

} );

},

\_renderItemData: function( ul, item ) {

return this.\_renderItem( ul, item ).data( "ui-autocomplete-item", item );

},

\_renderItem: function( ul, item ) {

return $( "<li>" )

.append( $( "<div>" ).text( item.label ) )

.appendTo( ul );

},

\_move: function( direction, event ) {

if ( !this.menu.element.is( ":visible" ) ) {

this.search( null, event );

return;

}

if ( this.menu.isFirstItem() && /^previous/.test( direction ) ||

this.menu.isLastItem() && /^next/.test( direction ) ) {

if ( !this.isMultiLine ) {

this.\_value( this.term );

}

this.menu.blur();

return;

}

this.menu[ direction ]( event );

},

widget: function() {

return this.menu.element;

},

\_value: function() {

return this.valueMethod.apply( this.element, arguments );

},

\_keyEvent: function( keyEvent, event ) {

if ( !this.isMultiLine || this.menu.element.is( ":visible" ) ) {

this.\_move( keyEvent, event );

// Prevents moving cursor to beginning/end of the text field in some browsers

event.preventDefault();

}

},

// Support: Chrome <=50

// We should be able to just use this.element.prop( "isContentEditable" )

// but hidden elements always report false in Chrome.

// https://code.google.com/p/chromium/issues/detail?id=313082

\_isContentEditable: function( element ) {

if ( !element.length ) {

return false;

}

var editable = element.prop( "contentEditable" );

if ( editable === "inherit" ) {

return this.\_isContentEditable( element.parent() );

}

return editable === "true";

}

} );

$.extend( $.ui.autocomplete, {

escapeRegex: function( value ) {

return value.replace( /[\-\[\]{}()\*+?.,\\\^$|#\s]/g, "\\$&" );

},

filter: function( array, term ) {

var matcher = new RegExp( $.ui.autocomplete.escapeRegex( term ), "i" );

return $.grep( array, function( value ) {

return matcher.test( value.label || value.value || value );

} );

}

} );

// Live region extension, adding a `messages` option

// NOTE: This is an experimental API. We are still investigating

// a full solution for string manipulation and internationalization.

$.widget( "ui.autocomplete", $.ui.autocomplete, {

options: {

messages: {

noResults: "No search results.",

results: function( amount ) {

return amount + ( amount > 1 ? " results are" : " result is" ) +

" available, use up and down arrow keys to navigate.";

}

}

},

\_\_response: function( content ) {

var message;

this.\_superApply( arguments );

if ( this.options.disabled || this.cancelSearch ) {

return;

}

if ( content && content.length ) {

message = this.options.messages.results( content.length );

} else {

message = this.options.messages.noResults;

}

this.liveRegion.children().hide();

$( "<div>" ).text( message ).appendTo( this.liveRegion );

}

} );

var widgetsAutocomplete = $.ui.autocomplete;

}));