**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8"/>

<link rel="stylesheet" type="text/css" href="./jquery.datetimepicker.css"/>

<style type="text/css">

.custom-date-style {

background-color: red !important;

}

.input{

}

.input-wide{

width: 500px;

}

</style>

</head>

<body>

<p><a href="http://xdsoft.net/jqplugins/datetimepicker/">Homepage</a></p>

<h3>DateTimePicker</h3>

<input type="text" value="" id="datetimepicker"/><br><br>

<h3>DateTimePickers selected by class</h3>

<input type="text" class="some\_class" value="" id="some\_class\_1"/>

<input type="text" class="some\_class" value="" id="some\_class\_2"/>

<h3>Mask DateTimePicker</h3>

<input type="text" value="" id="datetimepicker\_mask"/><br><br>

<h3>TimePicker</h3>

<input type="text" id="datetimepicker1"/><br><br>

<h3>DatePicker</h3>

<input type="text" id="datetimepicker2"/><br><br>

<h3>Inline DateTimePicker</h3>

<!--<div id="console" style="background-color:#fff;color:red">sdfdsfsdf</div>-->

<input type="text" id="datetimepicker3"/><input type="button" onclick="$('#datetimepicker3').datetimepicker({value:'2011/12/11 12:00'})" value="set inline value 2011/12/11 12:00"/><br><br>

<h3>Button Trigger</h3>

<input type="text" value="2013/12/03 18:00" id="datetimepicker4"/><input id="open" type="button" value="open"/><input id="close" type="button" value="close"/><input id="reset" type="button" value="reset"/>

<h3>TimePicker allows time</h3>

<input type="text" id="datetimepicker5"/><br><br>

<h3>Destroy DateTimePicker</h3>

<input type="text" id="datetimepicker6"/><input id="destroy" type="button" value="destroy"/>

<h3>Set options runtime DateTimePicker</h3>

<input type="text" id="datetimepicker7"/>

<p>If select day is Saturday, the minimum set 11:00, otherwise 8:00</p>

<h3>onGenerate</h3>

<input type="text" id="datetimepicker8"/>

<h3>disable all weekend</h3>

<input type="text" id="datetimepicker9"/>

<h3>Default date and time </h3>

<input type="text" id="default\_datetimepicker"/>

<h3>Show inline</h3>

<a href="javascript:void(0)" onclick="var si = document.getElementById('show\_inline').style; si.display = (si.display=='none')?'block':'none';return false; ">Show/Hide</a>

<div id="show\_inline" style="display:none">

<input type="text" id="datetimepicker10"/>

</div>

<h3>Disable Specific Dates</h3>

<p>Disable the dates 2 days from now.</p>

<input type="text" id="datetimepicker11"/>

<h3>Custom Date Styling</h3>

<p>Make the background of the date 2 days from now bright red.</p>

<input type="text" id="datetimepicker12"/>

<h3>Dark theme</h3>

<p>thank for this <a href="https://github.com/lampslave">https://github.com/lampslave</a></p>

<input type="text" id="datetimepicker\_dark"/>

<h3>Date time format and locale</h3>

<p></p>

<select id="datetimepicker\_format\_locale">

<option value="en">English</option>

<option value="de">German</option>

<option value="ru">Russian</option>

<option value="uk">Ukrainian</option>

<option value="fr">French</option>

<option value="es">Spanish</option>

</select>

<input type="text" value="D, l, M, F, Y-m-d H:i:s" id="datetimepicker\_format\_value"/>

<input type="button" value="applay =>" id="datetimepicker\_format\_change"/>

<input type="text" id="datetimepicker\_format" class="input input-wide"/>

</body>

<script src="./jquery.js"></script>

<script src="build/jquery.datetimepicker.full.js"></script>

<script>/\*

window.onerror = function(errorMsg) {

$('#console').html($('#console').html()+'<br>'+errorMsg)

}\*/

$.datetimepicker.setLocale('en');

$('#datetimepicker\_format').datetimepicker({value:'2015/04/15 05:03', format: $("#datetimepicker\_format\_value").val()});

console.log($('#datetimepicker\_format').datetimepicker('getValue'));

$("#datetimepicker\_format\_change").on("click", function(e){

$("#datetimepicker\_format").data('xdsoft\_datetimepicker').setOptions({format: $("#datetimepicker\_format\_value").val()});

});

$("#datetimepicker\_format\_locale").on("change", function(e){

$.datetimepicker.setLocale($(e.currentTarget).val());

});

$('#datetimepicker').datetimepicker({

dayOfWeekStart : 1,

lang:'en',

disabledDates:['1986/01/08','1986/01/09','1986/01/10'],

startDate: '1986/01/05'

});

$('#datetimepicker').datetimepicker({value:'2015/04/15 05:03',step:10});

$('.some\_class').datetimepicker();

$('#default\_datetimepicker').datetimepicker({

formatTime:'H:i',

formatDate:'d.m.Y',

//defaultDate:'8.12.1986', // it's my birthday

defaultDate:'+03.01.1970', // it's my birthday

defaultTime:'10:00',

timepickerScrollbar:false

});

$('#datetimepicker10').datetimepicker({

step:5,

inline:true

});

$('#datetimepicker\_mask').datetimepicker({

mask:'9999/19/39 29:59'

});

$('#datetimepicker1').datetimepicker({

datepicker:false,

format:'H:i',

step:5

});

$('#datetimepicker2').datetimepicker({

yearOffset:222,

lang:'ch',

timepicker:false,

format:'d/m/Y',

formatDate:'Y/m/d',

minDate:'-1970/01/02', // yesterday is minimum date

maxDate:'+1970/01/02' // and tommorow is maximum date calendar

});

$('#datetimepicker3').datetimepicker({

inline:true

});

$('#datetimepicker4').datetimepicker();

$('#open').click(function(){

$('#datetimepicker4').datetimepicker('show');

});

$('#close').click(function(){

$('#datetimepicker4').datetimepicker('hide');

});

$('#reset').click(function(){

$('#datetimepicker4').datetimepicker('reset');

});

$('#datetimepicker5').datetimepicker({

datepicker:false,

allowTimes:['12:00','13:00','15:00','17:00','17:05','17:20','19:00','20:00'],

step:5

});

$('#datetimepicker6').datetimepicker();

$('#destroy').click(function(){

if( $('#datetimepicker6').data('xdsoft\_datetimepicker') ){

$('#datetimepicker6').datetimepicker('destroy');

this.value = 'create';

}else{

$('#datetimepicker6').datetimepicker();

this.value = 'destroy';

}

});

var logic = function( currentDateTime ){

if (currentDateTime && currentDateTime.getDay() == 6){

this.setOptions({

minTime:'11:00'

});

}else

this.setOptions({

minTime:'8:00'

});

};

$('#datetimepicker7').datetimepicker({

onChangeDateTime:logic,

onShow:logic

});

$('#datetimepicker8').datetimepicker({

onGenerate:function( ct ){

$(this).find('.xdsoft\_date')

.toggleClass('xdsoft\_disabled');

},

minDate:'-1970/01/2',

maxDate:'+1970/01/2',

timepicker:false

});

$('#datetimepicker9').datetimepicker({

onGenerate:function( ct ){

$(this).find('.xdsoft\_date.xdsoft\_weekend')

.addClass('xdsoft\_disabled');

},

weekends:['01.01.2014','02.01.2014','03.01.2014','04.01.2014','05.01.2014','06.01.2014'],

timepicker:false

});

var dateToDisable = new Date();

dateToDisable.setDate(dateToDisable.getDate() + 2);

$('#datetimepicker11').datetimepicker({

beforeShowDay: function(date) {

if (date.getMonth() == dateToDisable.getMonth() && date.getDate() == dateToDisable.getDate()) {

return [false, ""]

}

return [true, ""];

}

});

$('#datetimepicker12').datetimepicker({

beforeShowDay: function(date) {

if (date.getMonth() == dateToDisable.getMonth() && date.getDate() == dateToDisable.getDate()) {

return [true, "custom-date-style"];

}

return [true, ""];

}

});

$('#datetimepicker\_dark').datetimepicker({theme:'dark'})

</script>

</html>

**CSS:**

.xdsoft\_datetimepicker {

box-shadow: 0 5px 15px -5px rgba(0, 0, 0, 0.506);

background: #fff;

border-bottom: 1px solid #bbb;

border-left: 1px solid #ccc;

border-right: 1px solid #ccc;

border-top: 1px solid #ccc;

color: #333;

font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;

padding: 8px;

padding-left: 0;

padding-top: 2px;

position: absolute;

z-index: 9999;

-moz-box-sizing: border-box;

box-sizing: border-box;

display: none;

}

.xdsoft\_datetimepicker.xdsoft\_rtl {

padding: 8px 0 8px 8px;

}

.xdsoft\_datetimepicker iframe {

position: absolute;

left: 0;

top: 0;

width: 75px;

height: 210px;

background: transparent;

border: none;

}

/\*For IE8 or lower\*/

.xdsoft\_datetimepicker button {

border: none !important;

}

.xdsoft\_noselect {

-webkit-touch-callout: none;

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

}

.xdsoft\_noselect::selection { background: transparent }

.xdsoft\_noselect::-moz-selection { background: transparent }

.xdsoft\_datetimepicker.xdsoft\_inline {

display: inline-block;

position: static;

box-shadow: none;

}

.xdsoft\_datetimepicker \* {

-moz-box-sizing: border-box;

box-sizing: border-box;

padding: 0;

margin: 0;

}

.xdsoft\_datetimepicker .xdsoft\_datepicker, .xdsoft\_datetimepicker .xdsoft\_timepicker {

display: none;

}

.xdsoft\_datetimepicker .xdsoft\_datepicker.active, .xdsoft\_datetimepicker .xdsoft\_timepicker.active {

display: block;

}

.xdsoft\_datetimepicker .xdsoft\_datepicker {

width: 224px;

float: left;

margin-left: 8px;

}

.xdsoft\_datetimepicker.xdsoft\_rtl .xdsoft\_datepicker {

float: right;

margin-right: 8px;

margin-left: 0;

}

.xdsoft\_datetimepicker.xdsoft\_showweeks .xdsoft\_datepicker {

width: 256px;

}

.xdsoft\_datetimepicker .xdsoft\_timepicker {

width: 58px;

float: left;

text-align: center;

margin-left: 8px;

margin-top: 0;

}

.xdsoft\_datetimepicker.xdsoft\_rtl .xdsoft\_timepicker {

float: right;

margin-right: 8px;

margin-left: 0;

}

.xdsoft\_datetimepicker .xdsoft\_datepicker.active+.xdsoft\_timepicker {

margin-top: 8px;

margin-bottom: 3px

}

.xdsoft\_datetimepicker .xdsoft\_monthpicker {

position: relative;

text-align: center;

}

.xdsoft\_datetimepicker .xdsoft\_label i,

.xdsoft\_datetimepicker .xdsoft\_prev,

.xdsoft\_datetimepicker .xdsoft\_next,

.xdsoft\_datetimepicker .xdsoft\_today\_button {

background-image: url(data:image/png;base64,);

}

.xdsoft\_datetimepicker .xdsoft\_label i {

opacity: 0.5;

background-position: -92px -19px;

display: inline-block;

width: 9px;

height: 20px;

vertical-align: middle;

}

.xdsoft\_datetimepicker .xdsoft\_prev {

float: left;

background-position: -20px 0;

}

.xdsoft\_datetimepicker .xdsoft\_today\_button {

float: left;

background-position: -70px 0;

margin-left: 5px;

}

.xdsoft\_datetimepicker .xdsoft\_next {

float: right;

background-position: 0 0;

}

.xdsoft\_datetimepicker .xdsoft\_next,

.xdsoft\_datetimepicker .xdsoft\_prev ,

.xdsoft\_datetimepicker .xdsoft\_today\_button {

background-color: transparent;

background-repeat: no-repeat;

border: 0 none;

cursor: pointer;

display: block;

height: 30px;

opacity: 0.5;

-ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=50)";

outline: medium none;

overflow: hidden;

padding: 0;

position: relative;

text-indent: 100%;

white-space: nowrap;

width: 20px;

min-width: 0;

}

.xdsoft\_datetimepicker .xdsoft\_timepicker .xdsoft\_prev,

.xdsoft\_datetimepicker .xdsoft\_timepicker .xdsoft\_next {

float: none;

background-position: -40px -15px;

height: 15px;

width: 30px;

display: block;

margin-left: 14px;

margin-top: 7px;

}

.xdsoft\_datetimepicker.xdsoft\_rtl .xdsoft\_timepicker .xdsoft\_prev,

.xdsoft\_datetimepicker.xdsoft\_rtl .xdsoft\_timepicker .xdsoft\_next {

float: none;

margin-left: 0;

margin-right: 14px;

}

.xdsoft\_datetimepicker .xdsoft\_timepicker .xdsoft\_prev {

background-position: -40px 0;

margin-bottom: 7px;

margin-top: 0;

}

.xdsoft\_datetimepicker .xdsoft\_timepicker .xdsoft\_time\_box {

height: 151px;

overflow: hidden;

border-bottom: 1px solid #ddd;

}

.xdsoft\_datetimepicker .xdsoft\_timepicker .xdsoft\_time\_box >div >div {

background: #f5f5f5;

border-top: 1px solid #ddd;

color: #666;

font-size: 12px;

text-align: center;

border-collapse: collapse;

cursor: pointer;

border-bottom-width: 0;

height: 25px;

line-height: 25px;

}

.xdsoft\_datetimepicker .xdsoft\_timepicker .xdsoft\_time\_box >div > div:first-child {

border-top-width: 0;

}

.xdsoft\_datetimepicker .xdsoft\_today\_button:hover,

.xdsoft\_datetimepicker .xdsoft\_next:hover,

.xdsoft\_datetimepicker .xdsoft\_prev:hover {

opacity: 1;

-ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=100)";

}

.xdsoft\_datetimepicker .xdsoft\_label {

display: inline;

position: relative;

z-index: 9999;

margin: 0;

padding: 5px 3px;

font-size: 14px;

line-height: 20px;

font-weight: bold;

background-color: #fff;

float: left;

width: 182px;

text-align: center;

cursor: pointer;

}

.xdsoft\_datetimepicker .xdsoft\_label:hover>span {

text-decoration: underline;

}

.xdsoft\_datetimepicker .xdsoft\_label:hover i {

opacity: 1.0;

}

.xdsoft\_datetimepicker .xdsoft\_label > .xdsoft\_select {

border: 1px solid #ccc;

position: absolute;

right: 0;

top: 30px;

z-index: 101;

display: none;

background: #fff;

max-height: 160px;

overflow-y: hidden;

}

.xdsoft\_datetimepicker .xdsoft\_label > .xdsoft\_select.xdsoft\_monthselect{ right: -7px }

.xdsoft\_datetimepicker .xdsoft\_label > .xdsoft\_select.xdsoft\_yearselect{ right: 2px }

.xdsoft\_datetimepicker .xdsoft\_label > .xdsoft\_select > div > .xdsoft\_option:hover {

color: #fff;

background: #ff8000;

}

.xdsoft\_datetimepicker .xdsoft\_label > .xdsoft\_select > div > .xdsoft\_option {

padding: 2px 10px 2px 5px;

text-decoration: none !important;

}

.xdsoft\_datetimepicker .xdsoft\_label > .xdsoft\_select > div > .xdsoft\_option.xdsoft\_current {

background: #33aaff;

box-shadow: #178fe5 0 1px 3px 0 inset;

color: #fff;

font-weight: 700;

}

.xdsoft\_datetimepicker .xdsoft\_month {

width: 100px;

text-align: right;

}

.xdsoft\_datetimepicker .xdsoft\_calendar {

clear: both;

}

.xdsoft\_datetimepicker .xdsoft\_year{

width: 48px;

margin-left: 5px;

}

.xdsoft\_datetimepicker .xdsoft\_calendar table {

border-collapse: collapse;

width: 100%;

}

.xdsoft\_datetimepicker .xdsoft\_calendar td > div {

padding-right: 5px;

}

.xdsoft\_datetimepicker .xdsoft\_calendar th {

height: 25px;

}

.xdsoft\_datetimepicker .xdsoft\_calendar td,.xdsoft\_datetimepicker .xdsoft\_calendar th {

width: 14.2857142%;

background: #f5f5f5;

border: 1px solid #ddd;

color: #666;

font-size: 12px;

text-align: right;

vertical-align: middle;

padding: 0;

border-collapse: collapse;

cursor: pointer;

height: 25px;

}

.xdsoft\_datetimepicker.xdsoft\_showweeks .xdsoft\_calendar td,.xdsoft\_datetimepicker.xdsoft\_showweeks .xdsoft\_calendar th {

width: 12.5%;

}

.xdsoft\_datetimepicker .xdsoft\_calendar th {

background: #f1f1f1;

}

.xdsoft\_datetimepicker .xdsoft\_calendar td.xdsoft\_today {

color: #33aaff;

}

.xdsoft\_datetimepicker .xdsoft\_calendar td.xdsoft\_highlighted\_default {

background: #ffe9d2;

box-shadow: #ffb871 0 1px 4px 0 inset;

color: #000;

}

.xdsoft\_datetimepicker .xdsoft\_calendar td.xdsoft\_highlighted\_mint {

background: #c1ffc9;

box-shadow: #00dd1c 0 1px 4px 0 inset;

color: #000;

}

.xdsoft\_datetimepicker .xdsoft\_calendar td.xdsoft\_default,

.xdsoft\_datetimepicker .xdsoft\_calendar td.xdsoft\_current,

.xdsoft\_datetimepicker .xdsoft\_timepicker .xdsoft\_time\_box >div >div.xdsoft\_current {

background: #33aaff;

box-shadow: #178fe5 0 1px 3px 0 inset;

color: #fff;

font-weight: 700;

}

.xdsoft\_datetimepicker .xdsoft\_calendar td.xdsoft\_other\_month,

.xdsoft\_datetimepicker .xdsoft\_calendar td.xdsoft\_disabled,

.xdsoft\_datetimepicker .xdsoft\_time\_box >div >div.xdsoft\_disabled {

opacity: 0.5;

-ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=50)";

cursor: default;

}

.xdsoft\_datetimepicker .xdsoft\_calendar td.xdsoft\_other\_month.xdsoft\_disabled {

opacity: 0.2;

-ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=20)";

}

.xdsoft\_datetimepicker .xdsoft\_calendar td:hover,

.xdsoft\_datetimepicker .xdsoft\_timepicker .xdsoft\_time\_box >div >div:hover {

color: #fff !important;

background: #ff8000 !important;

box-shadow: none !important;

}

.xdsoft\_datetimepicker .xdsoft\_calendar td.xdsoft\_current.xdsoft\_disabled:hover,

.xdsoft\_datetimepicker .xdsoft\_timepicker .xdsoft\_time\_box>div>div.xdsoft\_current.xdsoft\_disabled:hover {

background: #33aaff !important;

box-shadow: #178fe5 0 1px 3px 0 inset !important;

color: #fff !important;

}

.xdsoft\_datetimepicker .xdsoft\_calendar td.xdsoft\_disabled:hover,

.xdsoft\_datetimepicker .xdsoft\_timepicker .xdsoft\_time\_box >div >div.xdsoft\_disabled:hover {

color: inherit !important;

background: inherit !important;

box-shadow: inherit !important;

}

.xdsoft\_datetimepicker .xdsoft\_calendar th {

font-weight: 700;

text-align: center;

color: #999;

cursor: default;

}

.xdsoft\_datetimepicker .xdsoft\_copyright {

color: #ccc !important;

font-size: 10px;

clear: both;

float: none;

margin-left: 8px;

}

.xdsoft\_datetimepicker .xdsoft\_copyright a { color: #eee !important }

.xdsoft\_datetimepicker .xdsoft\_copyright a:hover { color: #aaa !important }

.xdsoft\_time\_box {

position: relative;

border: 1px solid #ccc;

}

.xdsoft\_scrollbar >.xdsoft\_scroller {

background: #ccc !important;

height: 20px;

border-radius: 3px;

}

.xdsoft\_scrollbar {

position: absolute;

width: 7px;

right: 0;

top: 0;

bottom: 0;

cursor: pointer;

}

.xdsoft\_datetimepicker.xdsoft\_rtl .xdsoft\_scrollbar {

left: 0;

right: auto;

}

.xdsoft\_scroller\_box {

position: relative;

}

.xdsoft\_datetimepicker.xdsoft\_dark {

box-shadow: 0 5px 15px -5px rgba(255, 255, 255, 0.506);

background: #000;

border-bottom: 1px solid #444;

border-left: 1px solid #333;

border-right: 1px solid #333;

border-top: 1px solid #333;

color: #ccc;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_timepicker .xdsoft\_time\_box {

border-bottom: 1px solid #222;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_timepicker .xdsoft\_time\_box >div >div {

background: #0a0a0a;

border-top: 1px solid #222;

color: #999;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_label {

background-color: #000;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_label > .xdsoft\_select {

border: 1px solid #333;

background: #000;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_label > .xdsoft\_select > div > .xdsoft\_option:hover {

color: #000;

background: #007fff;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_label > .xdsoft\_select > div > .xdsoft\_option.xdsoft\_current {

background: #cc5500;

box-shadow: #b03e00 0 1px 3px 0 inset;

color: #000;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_label i,

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_prev,

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_next,

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_today\_button {

background-image: url(data:image/png;base64,);

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_calendar td,

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_calendar th {

background: #0a0a0a;

border: 1px solid #222;

color: #999;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_calendar th {

background: #0e0e0e;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_calendar td.xdsoft\_today {

color: #cc5500;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_calendar td.xdsoft\_highlighted\_default {

background: #ffe9d2;

box-shadow: #ffb871 0 1px 4px 0 inset;

color:#000;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_calendar td.xdsoft\_highlighted\_mint {

background: #c1ffc9;

box-shadow: #00dd1c 0 1px 4px 0 inset;

color:#000;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_calendar td.xdsoft\_default,

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_calendar td.xdsoft\_current,

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_timepicker .xdsoft\_time\_box >div >div.xdsoft\_current {

background: #cc5500;

box-shadow: #b03e00 0 1px 3px 0 inset;

color: #000;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_calendar td:hover,

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_timepicker .xdsoft\_time\_box >div >div:hover {

color: #000 !important;

background: #007fff !important;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_calendar th {

color: #666;

}

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_copyright { color: #333 !important }

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_copyright a { color: #111 !important }

.xdsoft\_datetimepicker.xdsoft\_dark .xdsoft\_copyright a:hover { color: #555 !important }

.xdsoft\_dark .xdsoft\_time\_box {

border: 1px solid #333;

}

.xdsoft\_dark .xdsoft\_scrollbar >.xdsoft\_scroller {

background: #333 !important;

}

.xdsoft\_datetimepicker .xdsoft\_save\_selected {

display: block;

border: 1px solid #dddddd !important;

margin-top: 5px;

width: 100%;

color: #454551;

font-size: 13px;

}

.xdsoft\_datetimepicker .blue-gradient-button {

font-family: "museo-sans", "Book Antiqua", sans-serif;

font-size: 12px;

font-weight: 300;

color: #82878c;

height: 28px;

position: relative;

padding: 4px 17px 4px 33px;

border: 1px solid #d7d8da;

background: -moz-linear-gradient(top, #fff 0%, #f4f8fa 73%);

/\* FF3.6+ \*/

background: -webkit-gradient(linear, left top, left bottom, color-stop(0%, #fff), color-stop(73%, #f4f8fa));

/\* Chrome,Safari4+ \*/

background: -webkit-linear-gradient(top, #fff 0%, #f4f8fa 73%);

/\* Chrome10+,Safari5.1+ \*/

background: -o-linear-gradient(top, #fff 0%, #f4f8fa 73%);

/\* Opera 11.10+ \*/

background: -ms-linear-gradient(top, #fff 0%, #f4f8fa 73%);

/\* IE10+ \*/

background: linear-gradient(to bottom, #fff 0%, #f4f8fa 73%);

/\* W3C \*/

filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#fff', endColorstr='#f4f8fa',GradientType=0 );

/\* IE6-9 \*/

}

.xdsoft\_datetimepicker .blue-gradient-button:hover, .xdsoft\_datetimepicker .blue-gradient-button:focus, .xdsoft\_datetimepicker .blue-gradient-button:hover span, .xdsoft\_datetimepicker .blue-gradient-button:focus span {

color: #454551;

background: -moz-linear-gradient(top, #f4f8fa 0%, #FFF 73%);

/\* FF3.6+ \*/

background: -webkit-gradient(linear, left top, left bottom, color-stop(0%, #f4f8fa), color-stop(73%, #FFF));

/\* Chrome,Safari4+ \*/

background: -webkit-linear-gradient(top, #f4f8fa 0%, #FFF 73%);

/\* Chrome10+,Safari5.1+ \*/

background: -o-linear-gradient(top, #f4f8fa 0%, #FFF 73%);

/\* Opera 11.10+ \*/

background: -ms-linear-gradient(top, #f4f8fa 0%, #FFF 73%);

/\* IE10+ \*/

background: linear-gradient(to bottom, #f4f8fa 0%, #FFF 73%);

/\* W3C \*/

filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#f4f8fa', endColorstr='#FFF',GradientType=0 );

/\* IE6-9 \*/

}

**JS:**

/\*\*

\* @preserve jQuery DateTimePicker plugin v2.5.4

\* @homepage http://xdsoft.net/jqplugins/datetimepicker/

\* @author Chupurnov Valeriy (<chupurnov@gmail.com>)

\*/

/\*global DateFormatter, document,window,jQuery,setTimeout,clearTimeout,HighlightedDate,getCurrentValue\*/

;(function (factory) {

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

// AMD. Register as an anonymous module.

define(['jquery', 'jquery-mousewheel'], factory);

} else if (typeof exports === 'object') {

// Node/CommonJS style for Browserify

module.exports = factory;

} else {

// Browser globals

factory(jQuery);

}

}(function ($) {

'use strict';

var default\_options = {

i18n: {

ar: { // Arabic

months: [

"كانون الثاني", "شباط", "آذار", "نيسان", "مايو", "حزيران", "تموز", "آب", "أيلول", "تشرين الأول", "تشرين الثاني", "كانون الأول"

],

dayOfWeekShort: [

"ن", "ث", "ع", "خ", "ج", "س", "ح"

],

dayOfWeek: ["الأحد", "الاثنين", "الثلاثاء", "الأربعاء", "الخميس", "الجمعة", "السبت", "الأحد"]

},

ro: { // Romanian

months: [

"Ianuarie", "Februarie", "Martie", "Aprilie", "Mai", "Iunie", "Iulie", "August", "Septembrie", "Octombrie", "Noiembrie", "Decembrie"

],

dayOfWeekShort: [

"Du", "Lu", "Ma", "Mi", "Jo", "Vi", "Sâ"

],

dayOfWeek: ["Duminică", "Luni", "Marţi", "Miercuri", "Joi", "Vineri", "Sâmbătă"]

},

id: { // Indonesian

months: [

"Januari", "Februari", "Maret", "April", "Mei", "Juni", "Juli", "Agustus", "September", "Oktober", "November", "Desember"

],

dayOfWeekShort: [

"Min", "Sen", "Sel", "Rab", "Kam", "Jum", "Sab"

],

dayOfWeek: ["Minggu", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"]

},

is: { // Icelandic

months: [

"Janúar", "Febrúar", "Mars", "Apríl", "Maí", "Júní", "Júlí", "Ágúst", "September", "Október", "Nóvember", "Desember"

],

dayOfWeekShort: [

"Sun", "Mán", "Þrið", "Mið", "Fim", "Fös", "Lau"

],

dayOfWeek: ["Sunnudagur", "Mánudagur", "Þriðjudagur", "Miðvikudagur", "Fimmtudagur", "Föstudagur", "Laugardagur"]

},

bg: { // Bulgarian

months: [

"Януари", "Февруари", "Март", "Април", "Май", "Юни", "Юли", "Август", "Септември", "Октомври", "Ноември", "Декември"

],

dayOfWeekShort: [

"Нд", "Пн", "Вт", "Ср", "Чт", "Пт", "Сб"

],

dayOfWeek: ["Неделя", "Понеделник", "Вторник", "Сряда", "Четвъртък", "Петък", "Събота"]

},

fa: { // Persian/Farsi

months: [

'فروردین', 'اردیبهشت', 'خرداد', 'تیر', 'مرداد', 'شهریور', 'مهر', 'آبان', 'آذر', 'دی', 'بهمن', 'اسفند'

],

dayOfWeekShort: [

'یکشنبه', 'دوشنبه', 'سه شنبه', 'چهارشنبه', 'پنجشنبه', 'جمعه', 'شنبه'

],

dayOfWeek: ["یک‌شنبه", "دوشنبه", "سه‌شنبه", "چهارشنبه", "پنج‌شنبه", "جمعه", "شنبه", "یک‌شنبه"]

},

ru: { // Russian

months: [

'Январь', 'Февраль', 'Март', 'Апрель', 'Май', 'Июнь', 'Июль', 'Август', 'Сентябрь', 'Октябрь', 'Ноябрь', 'Декабрь'

],

dayOfWeekShort: [

"Вс", "Пн", "Вт", "Ср", "Чт", "Пт", "Сб"

],

dayOfWeek: ["Воскресенье", "Понедельник", "Вторник", "Среда", "Четверг", "Пятница", "Суббота"]

},

uk: { // Ukrainian

months: [

'Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень', 'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень'

],

dayOfWeekShort: [

"Ндл", "Пнд", "Втр", "Срд", "Чтв", "Птн", "Сбт"

],

dayOfWeek: ["Неділя", "Понеділок", "Вівторок", "Середа", "Четвер", "П'ятниця", "Субота"]

},

en: { // English

months: [

"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"

],

dayOfWeekShort: [

"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"

],

dayOfWeek: ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

},

el: { // Ελληνικά

months: [

"Ιανουάριος", "Φεβρουάριος", "Μάρτιος", "Απρίλιος", "Μάιος", "Ιούνιος", "Ιούλιος", "Αύγουστος", "Σεπτέμβριος", "Οκτώβριος", "Νοέμβριος", "Δεκέμβριος"

],

dayOfWeekShort: [

"Κυρ", "Δευ", "Τρι", "Τετ", "Πεμ", "Παρ", "Σαβ"

],

dayOfWeek: ["Κυριακή", "Δευτέρα", "Τρίτη", "Τετάρτη", "Πέμπτη", "Παρασκευή", "Σάββατο"]

},

de: { // German

months: [

'Januar', 'Februar', 'März', 'April', 'Mai', 'Juni', 'Juli', 'August', 'September', 'Oktober', 'November', 'Dezember'

],

dayOfWeekShort: [

"So", "Mo", "Di", "Mi", "Do", "Fr", "Sa"

],

dayOfWeek: ["Sonntag", "Montag", "Dienstag", "Mittwoch", "Donnerstag", "Freitag", "Samstag"]

},

nl: { // Dutch

months: [

"januari", "februari", "maart", "april", "mei", "juni", "juli", "augustus", "september", "oktober", "november", "december"

],

dayOfWeekShort: [

"zo", "ma", "di", "wo", "do", "vr", "za"

],

dayOfWeek: ["zondag", "maandag", "dinsdag", "woensdag", "donderdag", "vrijdag", "zaterdag"]

},

tr: { // Turkish

months: [

"Ocak", "Şubat", "Mart", "Nisan", "Mayıs", "Haziran", "Temmuz", "Ağustos", "Eylül", "Ekim", "Kasım", "Aralık"

],

dayOfWeekShort: [

"Paz", "Pts", "Sal", "Çar", "Per", "Cum", "Cts"

],

dayOfWeek: ["Pazar", "Pazartesi", "Salı", "Çarşamba", "Perşembe", "Cuma", "Cumartesi"]

},

fr: { //French

months: [

"Janvier", "Février", "Mars", "Avril", "Mai", "Juin", "Juillet", "Août", "Septembre", "Octobre", "Novembre", "Décembre"

],

dayOfWeekShort: [

"Dim", "Lun", "Mar", "Mer", "Jeu", "Ven", "Sam"

],

dayOfWeek: ["dimanche", "lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"]

},

es: { // Spanish

months: [

"Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"

],

dayOfWeekShort: [

"Dom", "Lun", "Mar", "Mié", "Jue", "Vie", "Sáb"

],

dayOfWeek: ["Domingo", "Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado"]

},

th: { // Thai

months: [

'มกราคม', 'กุมภาพันธ์', 'มีนาคม', 'เมษายน', 'พฤษภาคม', 'มิถุนายน', 'กรกฎาคม', 'สิงหาคม', 'กันยายน', 'ตุลาคม', 'พฤศจิกายน', 'ธันวาคม'

],

dayOfWeekShort: [

'อา.', 'จ.', 'อ.', 'พ.', 'พฤ.', 'ศ.', 'ส.'

],

dayOfWeek: ["อาทิตย์", "จันทร์", "อังคาร", "พุธ", "พฤหัส", "ศุกร์", "เสาร์", "อาทิตย์"]

},

pl: { // Polish

months: [

"styczeń", "luty", "marzec", "kwiecień", "maj", "czerwiec", "lipiec", "sierpień", "wrzesień", "październik", "listopad", "grudzień"

],

dayOfWeekShort: [

"nd", "pn", "wt", "śr", "cz", "pt", "sb"

],

dayOfWeek: ["niedziela", "poniedziałek", "wtorek", "środa", "czwartek", "piątek", "sobota"]

},

pt: { // Portuguese

months: [

"Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho", "Julho", "Agosto", "Setembro", "Outubro", "Novembro", "Dezembro"

],

dayOfWeekShort: [

"Dom", "Seg", "Ter", "Qua", "Qui", "Sex", "Sab"

],

dayOfWeek: ["Domingo", "Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sábado"]

},

ch: { // Simplified Chinese

months: [

"一月", "二月", "三月", "四月", "五月", "六月", "七月", "八月", "九月", "十月", "十一月", "十二月"

],

dayOfWeekShort: [

"日", "一", "二", "三", "四", "五", "六"

]

},

se: { // Swedish

months: [

"Januari", "Februari", "Mars", "April", "Maj", "Juni", "Juli", "Augusti", "September", "Oktober", "November", "December"

],

dayOfWeekShort: [

"Sön", "Mån", "Tis", "Ons", "Tor", "Fre", "Lör"

]

},

kr: { // Korean

months: [

"1월", "2월", "3월", "4월", "5월", "6월", "7월", "8월", "9월", "10월", "11월", "12월"

],

dayOfWeekShort: [

"일", "월", "화", "수", "목", "금", "토"

],

dayOfWeek: ["일요일", "월요일", "화요일", "수요일", "목요일", "금요일", "토요일"]

},

it: { // Italian

months: [

"Gennaio", "Febbraio", "Marzo", "Aprile", "Maggio", "Giugno", "Luglio", "Agosto", "Settembre", "Ottobre", "Novembre", "Dicembre"

],

dayOfWeekShort: [

"Dom", "Lun", "Mar", "Mer", "Gio", "Ven", "Sab"

],

dayOfWeek: ["Domenica", "Lunedì", "Martedì", "Mercoledì", "Giovedì", "Venerdì", "Sabato"]

},

da: { // Dansk

months: [

"January", "Februar", "Marts", "April", "Maj", "Juni", "July", "August", "September", "Oktober", "November", "December"

],

dayOfWeekShort: [

"Søn", "Man", "Tir", "Ons", "Tor", "Fre", "Lør"

],

dayOfWeek: ["søndag", "mandag", "tirsdag", "onsdag", "torsdag", "fredag", "lørdag"]

},

no: { // Norwegian

months: [

"Januar", "Februar", "Mars", "April", "Mai", "Juni", "Juli", "August", "September", "Oktober", "November", "Desember"

],

dayOfWeekShort: [

"Søn", "Man", "Tir", "Ons", "Tor", "Fre", "Lør"

],

dayOfWeek: ['Søndag', 'Mandag', 'Tirsdag', 'Onsdag', 'Torsdag', 'Fredag', 'Lørdag']

},

ja: { // Japanese

months: [

"1月", "2月", "3月", "4月", "5月", "6月", "7月", "8月", "9月", "10月", "11月", "12月"

],

dayOfWeekShort: [

"日", "月", "火", "水", "木", "金", "土"

],

dayOfWeek: ["日曜", "月曜", "火曜", "水曜", "木曜", "金曜", "土曜"]

},

vi: { // Vietnamese

months: [

"Tháng 1", "Tháng 2", "Tháng 3", "Tháng 4", "Tháng 5", "Tháng 6", "Tháng 7", "Tháng 8", "Tháng 9", "Tháng 10", "Tháng 11", "Tháng 12"

],

dayOfWeekShort: [

"CN", "T2", "T3", "T4", "T5", "T6", "T7"

],

dayOfWeek: ["Chủ nhật", "Thứ hai", "Thứ ba", "Thứ tư", "Thứ năm", "Thứ sáu", "Thứ bảy"]

},

sl: { // Slovenščina

months: [

"Januar", "Februar", "Marec", "April", "Maj", "Junij", "Julij", "Avgust", "September", "Oktober", "November", "December"

],

dayOfWeekShort: [

"Ned", "Pon", "Tor", "Sre", "Čet", "Pet", "Sob"

],

dayOfWeek: ["Nedelja", "Ponedeljek", "Torek", "Sreda", "Četrtek", "Petek", "Sobota"]

},

cs: { // Čeština

months: [

"Leden", "Únor", "Březen", "Duben", "Květen", "Červen", "Červenec", "Srpen", "Září", "Říjen", "Listopad", "Prosinec"

],

dayOfWeekShort: [

"Ne", "Po", "Út", "St", "Čt", "Pá", "So"

]

},

hu: { // Hungarian

months: [

"Január", "Február", "Március", "Április", "Május", "Június", "Július", "Augusztus", "Szeptember", "Október", "November", "December"

],

dayOfWeekShort: [

"Va", "Hé", "Ke", "Sze", "Cs", "Pé", "Szo"

],

dayOfWeek: ["vasárnap", "hétfő", "kedd", "szerda", "csütörtök", "péntek", "szombat"]

},

az: { //Azerbaijanian (Azeri)

months: [

"Yanvar", "Fevral", "Mart", "Aprel", "May", "Iyun", "Iyul", "Avqust", "Sentyabr", "Oktyabr", "Noyabr", "Dekabr"

],

dayOfWeekShort: [

"B", "Be", "Ça", "Ç", "Ca", "C", "Ş"

],

dayOfWeek: ["Bazar", "Bazar ertəsi", "Çərşənbə axşamı", "Çərşənbə", "Cümə axşamı", "Cümə", "Şənbə"]

},

bs: { //Bosanski

months: [

"Januar", "Februar", "Mart", "April", "Maj", "Jun", "Jul", "Avgust", "Septembar", "Oktobar", "Novembar", "Decembar"

],

dayOfWeekShort: [

"Ned", "Pon", "Uto", "Sri", "Čet", "Pet", "Sub"

],

dayOfWeek: ["Nedjelja","Ponedjeljak", "Utorak", "Srijeda", "Četvrtak", "Petak", "Subota"]

},

ca: { //Català

months: [

"Gener", "Febrer", "Març", "Abril", "Maig", "Juny", "Juliol", "Agost", "Setembre", "Octubre", "Novembre", "Desembre"

],

dayOfWeekShort: [

"Dg", "Dl", "Dt", "Dc", "Dj", "Dv", "Ds"

],

dayOfWeek: ["Diumenge", "Dilluns", "Dimarts", "Dimecres", "Dijous", "Divendres", "Dissabte"]

},

'en-GB': { //English (British)

months: [

"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"

],

dayOfWeekShort: [

"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"

],

dayOfWeek: ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

},

et: { //"Eesti"

months: [

"Jaanuar", "Veebruar", "Märts", "Aprill", "Mai", "Juuni", "Juuli", "August", "September", "Oktoober", "November", "Detsember"

],

dayOfWeekShort: [

"P", "E", "T", "K", "N", "R", "L"

],

dayOfWeek: ["Pühapäev", "Esmaspäev", "Teisipäev", "Kolmapäev", "Neljapäev", "Reede", "Laupäev"]

},

eu: { //Euskara

months: [

"Urtarrila", "Otsaila", "Martxoa", "Apirila", "Maiatza", "Ekaina", "Uztaila", "Abuztua", "Iraila", "Urria", "Azaroa", "Abendua"

],

dayOfWeekShort: [

"Ig.", "Al.", "Ar.", "Az.", "Og.", "Or.", "La."

],

dayOfWeek: ['Igandea', 'Astelehena', 'Asteartea', 'Asteazkena', 'Osteguna', 'Ostirala', 'Larunbata']

},

fi: { //Finnish (Suomi)

months: [

"Tammikuu", "Helmikuu", "Maaliskuu", "Huhtikuu", "Toukokuu", "Kesäkuu", "Heinäkuu", "Elokuu", "Syyskuu", "Lokakuu", "Marraskuu", "Joulukuu"

],

dayOfWeekShort: [

"Su", "Ma", "Ti", "Ke", "To", "Pe", "La"

],

dayOfWeek: ["sunnuntai", "maanantai", "tiistai", "keskiviikko", "torstai", "perjantai", "lauantai"]

},

gl: { //Galego

months: [

"Xan", "Feb", "Maz", "Abr", "Mai", "Xun", "Xul", "Ago", "Set", "Out", "Nov", "Dec"

],

dayOfWeekShort: [

"Dom", "Lun", "Mar", "Mer", "Xov", "Ven", "Sab"

],

dayOfWeek: ["Domingo", "Luns", "Martes", "Mércores", "Xoves", "Venres", "Sábado"]

},

hr: { //Hrvatski

months: [

"Siječanj", "Veljača", "Ožujak", "Travanj", "Svibanj", "Lipanj", "Srpanj", "Kolovoz", "Rujan", "Listopad", "Studeni", "Prosinac"

],

dayOfWeekShort: [

"Ned", "Pon", "Uto", "Sri", "Čet", "Pet", "Sub"

],

dayOfWeek: ["Nedjelja", "Ponedjeljak", "Utorak", "Srijeda", "Četvrtak", "Petak", "Subota"]

},

ko: { //Korean (한국어)

months: [

"1월", "2월", "3월", "4월", "5월", "6월", "7월", "8월", "9월", "10월", "11월", "12월"

],

dayOfWeekShort: [

"일", "월", "화", "수", "목", "금", "토"

],

dayOfWeek: ["일요일", "월요일", "화요일", "수요일", "목요일", "금요일", "토요일"]

},

lt: { //Lithuanian (lietuvių)

months: [

"Sausio", "Vasario", "Kovo", "Balandžio", "Gegužės", "Birželio", "Liepos", "Rugpjūčio", "Rugsėjo", "Spalio", "Lapkričio", "Gruodžio"

],

dayOfWeekShort: [

"Sek", "Pir", "Ant", "Tre", "Ket", "Pen", "Šeš"

],

dayOfWeek: ["Sekmadienis", "Pirmadienis", "Antradienis", "Trečiadienis", "Ketvirtadienis", "Penktadienis", "Šeštadienis"]

},

lv: { //Latvian (Latviešu)

months: [

"Janvāris", "Februāris", "Marts", "Aprīlis ", "Maijs", "Jūnijs", "Jūlijs", "Augusts", "Septembris", "Oktobris", "Novembris", "Decembris"

],

dayOfWeekShort: [

"Sv", "Pr", "Ot", "Tr", "Ct", "Pk", "St"

],

dayOfWeek: ["Svētdiena", "Pirmdiena", "Otrdiena", "Trešdiena", "Ceturtdiena", "Piektdiena", "Sestdiena"]

},

mk: { //Macedonian (Македонски)

months: [

"јануари", "февруари", "март", "април", "мај", "јуни", "јули", "август", "септември", "октомври", "ноември", "декември"

],

dayOfWeekShort: [

"нед", "пон", "вто", "сре", "чет", "пет", "саб"

],

dayOfWeek: ["Недела", "Понеделник", "Вторник", "Среда", "Четврток", "Петок", "Сабота"]

},

mn: { //Mongolian (Монгол)

months: [

"1-р сар", "2-р сар", "3-р сар", "4-р сар", "5-р сар", "6-р сар", "7-р сар", "8-р сар", "9-р сар", "10-р сар", "11-р сар", "12-р сар"

],

dayOfWeekShort: [

"Дав", "Мяг", "Лха", "Пүр", "Бсн", "Бям", "Ням"

],

dayOfWeek: ["Даваа", "Мягмар", "Лхагва", "Пүрэв", "Баасан", "Бямба", "Ням"]

},

'pt-BR': { //Português(Brasil)

months: [

"Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho", "Julho", "Agosto", "Setembro", "Outubro", "Novembro", "Dezembro"

],

dayOfWeekShort: [

"Dom", "Seg", "Ter", "Qua", "Qui", "Sex", "Sáb"

],

dayOfWeek: ["Domingo", "Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sábado"]

},

sk: { //Slovenčina

months: [

"Január", "Február", "Marec", "Apríl", "Máj", "Jún", "Júl", "August", "September", "Október", "November", "December"

],

dayOfWeekShort: [

"Ne", "Po", "Ut", "St", "Št", "Pi", "So"

],

dayOfWeek: ["Nedeľa", "Pondelok", "Utorok", "Streda", "Štvrtok", "Piatok", "Sobota"]

},

sq: { //Albanian (Shqip)

months: [

"Janar", "Shkurt", "Mars", "Prill", "Maj", "Qershor", "Korrik", "Gusht", "Shtator", "Tetor", "Nëntor", "Dhjetor"

],

dayOfWeekShort: [

"Die", "Hën", "Mar", "Mër", "Enj", "Pre", "Shtu"

],

dayOfWeek: ["E Diel", "E Hënë", "E Martē", "E Mërkurë", "E Enjte", "E Premte", "E Shtunë"]

},

'sr-YU': { //Serbian (Srpski)

months: [

"Januar", "Februar", "Mart", "April", "Maj", "Jun", "Jul", "Avgust", "Septembar", "Oktobar", "Novembar", "Decembar"

],

dayOfWeekShort: [

"Ned", "Pon", "Uto", "Sre", "čet", "Pet", "Sub"

],

dayOfWeek: ["Nedelja","Ponedeljak", "Utorak", "Sreda", "Četvrtak", "Petak", "Subota"]

},

sr: { //Serbian Cyrillic (Српски)

months: [

"јануар", "фебруар", "март", "април", "мај", "јун", "јул", "август", "септембар", "октобар", "новембар", "децембар"

],

dayOfWeekShort: [

"нед", "пон", "уто", "сре", "чет", "пет", "суб"

],

dayOfWeek: ["Недеља","Понедељак", "Уторак", "Среда", "Четвртак", "Петак", "Субота"]

},

sv: { //Svenska

months: [

"Januari", "Februari", "Mars", "April", "Maj", "Juni", "Juli", "Augusti", "September", "Oktober", "November", "December"

],

dayOfWeekShort: [

"Sön", "Mån", "Tis", "Ons", "Tor", "Fre", "Lör"

],

dayOfWeek: ["Söndag", "Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag", "Lördag"]

},

'zh-TW': { //Traditional Chinese (繁體中文)

months: [

"一月", "二月", "三月", "四月", "五月", "六月", "七月", "八月", "九月", "十月", "十一月", "十二月"

],

dayOfWeekShort: [

"日", "一", "二", "三", "四", "五", "六"

],

dayOfWeek: ["星期日", "星期一", "星期二", "星期三", "星期四", "星期五", "星期六"]

},

zh: { //Simplified Chinese (简体中文)

months: [

"一月", "二月", "三月", "四月", "五月", "六月", "七月", "八月", "九月", "十月", "十一月", "十二月"

],

dayOfWeekShort: [

"日", "一", "二", "三", "四", "五", "六"

],

dayOfWeek: ["星期日", "星期一", "星期二", "星期三", "星期四", "星期五", "星期六"]

},

he: { //Hebrew (עברית)

months: [

'ינואר', 'פברואר', 'מרץ', 'אפריל', 'מאי', 'יוני', 'יולי', 'אוגוסט', 'ספטמבר', 'אוקטובר', 'נובמבר', 'דצמבר'

],

dayOfWeekShort: [

'א\'', 'ב\'', 'ג\'', 'ד\'', 'ה\'', 'ו\'', 'שבת'

],

dayOfWeek: ["ראשון", "שני", "שלישי", "רביעי", "חמישי", "שישי", "שבת", "ראשון"]

},

hy: { // Armenian

months: [

"Հունվար", "Փետրվար", "Մարտ", "Ապրիլ", "Մայիս", "Հունիս", "Հուլիս", "Օգոստոս", "Սեպտեմբեր", "Հոկտեմբեր", "Նոյեմբեր", "Դեկտեմբեր"

],

dayOfWeekShort: [

"Կի", "Երկ", "Երք", "Չոր", "Հնգ", "Ուրբ", "Շբթ"

],

dayOfWeek: ["Կիրակի", "Երկուշաբթի", "Երեքշաբթի", "Չորեքշաբթի", "Հինգշաբթի", "Ուրբաթ", "Շաբաթ"]

},

kg: { // Kyrgyz

months: [

'Үчтүн айы', 'Бирдин айы', 'Жалган Куран', 'Чын Куран', 'Бугу', 'Кулжа', 'Теке', 'Баш Оона', 'Аяк Оона', 'Тогуздун айы', 'Жетинин айы', 'Бештин айы'

],

dayOfWeekShort: [

"Жек", "Дүй", "Шей", "Шар", "Бей", "Жум", "Ише"

],

dayOfWeek: [

"Жекшемб", "Дүйшөмб", "Шейшемб", "Шаршемб", "Бейшемби", "Жума", "Ишенб"

]

},

rm: { // Romansh

months: [

"Schaner", "Favrer", "Mars", "Avrigl", "Matg", "Zercladur", "Fanadur", "Avust", "Settember", "October", "November", "December"

],

dayOfWeekShort: [

"Du", "Gli", "Ma", "Me", "Gie", "Ve", "So"

],

dayOfWeek: [

"Dumengia", "Glindesdi", "Mardi", "Mesemna", "Gievgia", "Venderdi", "Sonda"

]

},

ka: { // Georgian

months: [

'იანვარი', 'თებერვალი', 'მარტი', 'აპრილი', 'მაისი', 'ივნისი', 'ივლისი', 'აგვისტო', 'სექტემბერი', 'ოქტომბერი', 'ნოემბერი', 'დეკემბერი'

],

dayOfWeekShort: [

"კვ", "ორშ", "სამშ", "ოთხ", "ხუთ", "პარ", "შაბ"

],

dayOfWeek: ["კვირა", "ორშაბათი", "სამშაბათი", "ოთხშაბათი", "ხუთშაბათი", "პარასკევი", "შაბათი"]

},

},

value: '',

rtl: false,

format: 'Y/m/d H:i',

formatTime: 'H:i',

formatDate: 'Y/m/d',

startDate: false, // new Date(), '1986/12/08', '-1970/01/05','-1970/01/05',

step: 60,

monthChangeSpinner: true,

closeOnDateSelect: false,

closeOnTimeSelect: true,

closeOnWithoutClick: true,

closeOnInputClick: true,

timepicker: true,

datepicker: true,

weeks: false,

defaultTime: false, // use formatTime format (ex. '10:00' for formatTime: 'H:i')

defaultDate: false, // use formatDate format (ex new Date() or '1986/12/08' or '-1970/01/05' or '-1970/01/05')

minDate: false,

maxDate: false,

minTime: false,

maxTime: false,

disabledMinTime: false,

disabledMaxTime: false,

allowTimes: [],

opened: false,

initTime: true,

inline: false,

theme: '',

onSelectDate: function () {},

onSelectTime: function () {},

onChangeMonth: function () {},

onGetWeekOfYear: function () {},

onChangeYear: function () {},

onChangeDateTime: function () {},

onShow: function () {},

onClose: function () {},

onGenerate: function () {},

withoutCopyright: true,

inverseButton: false,

hours12: false,

next: 'xdsoft\_next',

prev : 'xdsoft\_prev',

dayOfWeekStart: 0,

parentID: 'body',

timeHeightInTimePicker: 25,

timepickerScrollbar: true,

todayButton: true,

prevButton: true,

nextButton: true,

defaultSelect: true,

scrollMonth: true,

scrollTime: true,

scrollInput: true,

lazyInit: false,

mask: false,

validateOnBlur: true,

allowBlank: true,

yearStart: 1950,

yearEnd: 2050,

monthStart: 0,

monthEnd: 11,

style: '',

id: '',

fixed: false,

roundTime: 'round', // ceil, floor

className: '',

weekends: [],

highlightedDates: [],

highlightedPeriods: [],

allowDates : [],

allowDateRe : null,

disabledDates : [],

disabledWeekDays: [],

yearOffset: 0,

beforeShowDay: null,

enterLikeTab: true,

showApplyButton: false

};

var dateHelper = null,

globalLocaleDefault = 'en',

globalLocale = 'en';

var dateFormatterOptionsDefault = {

meridiem: ['AM', 'PM']

};

var initDateFormatter = function(){

var locale = default\_options.i18n[globalLocale],

opts = {

days: locale.dayOfWeek,

daysShort: locale.dayOfWeekShort,

months: locale.months,

monthsShort: $.map(locale.months, function(n){ return n.substring(0, 3) }),

};

dateHelper = new DateFormatter({

dateSettings: $.extend({}, dateFormatterOptionsDefault, opts)

});

};

// for locale settings

$.datetimepicker = {

setLocale: function(locale){

var newLocale = default\_options.i18n[locale]?locale:globalLocaleDefault;

if(globalLocale != newLocale){

globalLocale = newLocale;

// reinit date formatter

initDateFormatter();

}

},

setDateFormatter: function(dateFormatter) {

dateHelper = dateFormatter;

},

RFC\_2822: 'D, d M Y H:i:s O',

ATOM: 'Y-m-d\TH:i:sP',

ISO\_8601: 'Y-m-d\TH:i:sO',

RFC\_822: 'D, d M y H:i:s O',

RFC\_850: 'l, d-M-y H:i:s T',

RFC\_1036: 'D, d M y H:i:s O',

RFC\_1123: 'D, d M Y H:i:s O',

RSS: 'D, d M Y H:i:s O',

W3C: 'Y-m-d\TH:i:sP'

};

// first init date formatter

initDateFormatter();

// fix for ie8

if (!window.getComputedStyle) {

window.getComputedStyle = function (el, pseudo) {

this.el = el;

this.getPropertyValue = function (prop) {

var re = /(\-([a-z]){1})/g;

if (prop === 'float') {

prop = 'styleFloat';

}

if (re.test(prop)) {

prop = prop.replace(re, function (a, b, c) {

return c.toUpperCase();

});

}

return el.currentStyle[prop] || null;

};

return this;

};

}

if (!Array.prototype.indexOf) {

Array.prototype.indexOf = function (obj, start) {

var i, j;

for (i = (start || 0), j = this.length; i < j; i += 1) {

if (this[i] === obj) { return i; }

}

return -1;

};

}

Date.prototype.countDaysInMonth = function () {

return new Date(this.getFullYear(), this.getMonth() + 1, 0).getDate();

};

$.fn.xdsoftScroller = function (percent) {

return this.each(function () {

var timeboxparent = $(this),

pointerEventToXY = function (e) {

var out = {x: 0, y: 0},

touch;

if (e.type === 'touchstart' || e.type === 'touchmove' || e.type === 'touchend' || e.type === 'touchcancel') {

touch = e.originalEvent.touches[0] || e.originalEvent.changedTouches[0];

out.x = touch.clientX;

out.y = touch.clientY;

} else if (e.type === 'mousedown' || e.type === 'mouseup' || e.type === 'mousemove' || e.type === 'mouseover' || e.type === 'mouseout' || e.type === 'mouseenter' || e.type === 'mouseleave') {

out.x = e.clientX;

out.y = e.clientY;

}

return out;

},

timebox,

parentHeight,

height,

scrollbar,

scroller,

maximumOffset = 100,

start = false,

startY = 0,

startTop = 0,

h1 = 0,

touchStart = false,

startTopScroll = 0,

calcOffset = function () {};

if (percent === 'hide') {

timeboxparent.find('.xdsoft\_scrollbar').hide();

return;

}

if (!$(this).hasClass('xdsoft\_scroller\_box')) {

timebox = timeboxparent.children().eq(0);

parentHeight = timeboxparent[0].clientHeight;

height = timebox[0].offsetHeight;

scrollbar = $('<div class="xdsoft\_scrollbar"></div>');

scroller = $('<div class="xdsoft\_scroller"></div>');

scrollbar.append(scroller);

timeboxparent.addClass('xdsoft\_scroller\_box').append(scrollbar);

calcOffset = function calcOffset(event) {

var offset = pointerEventToXY(event).y - startY + startTopScroll;

if (offset < 0) {

offset = 0;

}

if (offset + scroller[0].offsetHeight > h1) {

offset = h1 - scroller[0].offsetHeight;

}

timeboxparent.trigger('scroll\_element.xdsoft\_scroller', [maximumOffset ? offset / maximumOffset : 0]);

};

scroller

.on('touchstart.xdsoft\_scroller mousedown.xdsoft\_scroller', function (event) {

if (!parentHeight) {

timeboxparent.trigger('resize\_scroll.xdsoft\_scroller', [percent]);

}

startY = pointerEventToXY(event).y;

startTopScroll = parseInt(scroller.css('margin-top'), 10);

h1 = scrollbar[0].offsetHeight;

if (event.type === 'mousedown' || event.type === 'touchstart') {

if (document) {

$(document.body).addClass('xdsoft\_noselect');

}

$([document.body, window]).on('touchend mouseup.xdsoft\_scroller', function arguments\_callee() {

$([document.body, window]).off('touchend mouseup.xdsoft\_scroller', arguments\_callee)

.off('mousemove.xdsoft\_scroller', calcOffset)

.removeClass('xdsoft\_noselect');

});

$(document.body).on('mousemove.xdsoft\_scroller', calcOffset);

} else {

touchStart = true;

event.stopPropagation();

event.preventDefault();

}

})

.on('touchmove', function (event) {

if (touchStart) {

event.preventDefault();

calcOffset(event);

}

})

.on('touchend touchcancel', function () {

touchStart = false;

startTopScroll = 0;

});

timeboxparent

.on('scroll\_element.xdsoft\_scroller', function (event, percentage) {

if (!parentHeight) {

timeboxparent.trigger('resize\_scroll.xdsoft\_scroller', [percentage, true]);

}

percentage = percentage > 1 ? 1 : (percentage < 0 || isNaN(percentage)) ? 0 : percentage;

scroller.css('margin-top', maximumOffset \* percentage);

setTimeout(function () {

timebox.css('marginTop', -parseInt((timebox[0].offsetHeight - parentHeight) \* percentage, 10));

}, 10);

})

.on('resize\_scroll.xdsoft\_scroller', function (event, percentage, noTriggerScroll) {

var percent, sh;

parentHeight = timeboxparent[0].clientHeight;

height = timebox[0].offsetHeight;

percent = parentHeight / height;

sh = percent \* scrollbar[0].offsetHeight;

if (percent > 1) {

scroller.hide();

} else {

scroller.show();

scroller.css('height', parseInt(sh > 10 ? sh : 10, 10));

maximumOffset = scrollbar[0].offsetHeight - scroller[0].offsetHeight;

if (noTriggerScroll !== true) {

timeboxparent.trigger('scroll\_element.xdsoft\_scroller', [percentage || Math.abs(parseInt(timebox.css('marginTop'), 10)) / (height - parentHeight)]);

}

}

});

timeboxparent.on('mousewheel', function (event) {

var top = Math.abs(parseInt(timebox.css('marginTop'), 10));

top = top - (event.deltaY \* 20);

if (top < 0) {

top = 0;

}

timeboxparent.trigger('scroll\_element.xdsoft\_scroller', [top / (height - parentHeight)]);

event.stopPropagation();

return false;

});

timeboxparent.on('touchstart', function (event) {

start = pointerEventToXY(event);

startTop = Math.abs(parseInt(timebox.css('marginTop'), 10));

});

timeboxparent.on('touchmove', function (event) {

if (start) {

event.preventDefault();

var coord = pointerEventToXY(event);

timeboxparent.trigger('scroll\_element.xdsoft\_scroller', [(startTop - (coord.y - start.y)) / (height - parentHeight)]);

}

});

timeboxparent.on('touchend touchcancel', function () {

start = false;

startTop = 0;

});

}

timeboxparent.trigger('resize\_scroll.xdsoft\_scroller', [percent]);

});

};

$.fn.datetimepicker = function (opt, opt2) {

var result = this,

KEY0 = 48,

KEY9 = 57,

\_KEY0 = 96,

\_KEY9 = 105,

CTRLKEY = 17,

DEL = 46,

ENTER = 13,

ESC = 27,

BACKSPACE = 8,

ARROWLEFT = 37,

ARROWUP = 38,

ARROWRIGHT = 39,

ARROWDOWN = 40,

TAB = 9,

F5 = 116,

AKEY = 65,

CKEY = 67,

VKEY = 86,

ZKEY = 90,

YKEY = 89,

ctrlDown = false,

options = ($.isPlainObject(opt) || !opt) ? $.extend(true, {}, default\_options, opt) : $.extend(true, {}, default\_options),

lazyInitTimer = 0,

createDateTimePicker,

destroyDateTimePicker,

lazyInit = function (input) {

input

.on('open.xdsoft focusin.xdsoft mousedown.xdsoft touchstart', function initOnActionCallback() {

if (input.is(':disabled') || input.data('xdsoft\_datetimepicker')) {

return;

}

clearTimeout(lazyInitTimer);

lazyInitTimer = setTimeout(function () {

if (!input.data('xdsoft\_datetimepicker')) {

createDateTimePicker(input);

}

input

.off('open.xdsoft focusin.xdsoft mousedown.xdsoft touchstart', initOnActionCallback)

.trigger('open.xdsoft');

}, 100);

});

};

createDateTimePicker = function (input) {

var datetimepicker = $('<div class="xdsoft\_datetimepicker xdsoft\_noselect"></div>'),

xdsoft\_copyright = $('<div class="xdsoft\_copyright"><a target="\_blank" href="http://xdsoft.net/jqplugins/datetimepicker/">xdsoft.net</a></div>'),

datepicker = $('<div class="xdsoft\_datepicker active"></div>'),

month\_picker = $('<div class="xdsoft\_monthpicker"><button type="button" class="xdsoft\_prev"></button><button type="button" class="xdsoft\_today\_button"></button>' +

'<div class="xdsoft\_label xdsoft\_month"><span></span><i></i></div>' +

'<div class="xdsoft\_label xdsoft\_year"><span></span><i></i></div>' +

'<button type="button" class="xdsoft\_next"></button></div>'),

calendar = $('<div class="xdsoft\_calendar"></div>'),

timepicker = $('<div class="xdsoft\_timepicker active"><button type="button" class="xdsoft\_prev"></button><div class="xdsoft\_time\_box"></div><button type="button" class="xdsoft\_next"></button></div>'),

timeboxparent = timepicker.find('.xdsoft\_time\_box').eq(0),

timebox = $('<div class="xdsoft\_time\_variant"></div>'),

applyButton = $('<button type="button" class="xdsoft\_save\_selected blue-gradient-button">Save Selected</button>'),

monthselect = $('<div class="xdsoft\_select xdsoft\_monthselect"><div></div></div>'),

yearselect = $('<div class="xdsoft\_select xdsoft\_yearselect"><div></div></div>'),

triggerAfterOpen = false,

XDSoft\_datetime,

xchangeTimer,

timerclick,

current\_time\_index,

setPos,

timer = 0,

\_xdsoft\_datetime,

forEachAncestorOf,

throttle;

if (options.id) {

datetimepicker.attr('id', options.id);

}

if (options.style) {

datetimepicker.attr('style', options.style);

}

if (options.weeks) {

datetimepicker.addClass('xdsoft\_showweeks');

}

if (options.rtl) {

datetimepicker.addClass('xdsoft\_rtl');

}

datetimepicker.addClass('xdsoft\_' + options.theme);

datetimepicker.addClass(options.className);

month\_picker

.find('.xdsoft\_month span')

.after(monthselect);

month\_picker

.find('.xdsoft\_year span')

.after(yearselect);

month\_picker

.find('.xdsoft\_month,.xdsoft\_year')

.on('touchstart mousedown.xdsoft', function (event) {

var select = $(this).find('.xdsoft\_select').eq(0),

val = 0,

top = 0,

visible = select.is(':visible'),

items,

i;

month\_picker

.find('.xdsoft\_select')

.hide();

if (\_xdsoft\_datetime.currentTime) {

val = \_xdsoft\_datetime.currentTime[$(this).hasClass('xdsoft\_month') ? 'getMonth' : 'getFullYear']();

}

select[visible ? 'hide' : 'show']();

for (items = select.find('div.xdsoft\_option'), i = 0; i < items.length; i += 1) {

if (items.eq(i).data('value') === val) {

break;

} else {

top += items[0].offsetHeight;

}

}

select.xdsoftScroller(top / (select.children()[0].offsetHeight - (select[0].clientHeight)));

event.stopPropagation();

return false;

});

month\_picker

.find('.xdsoft\_select')

.xdsoftScroller()

.on('touchstart mousedown.xdsoft', function (event) {

event.stopPropagation();

event.preventDefault();

})

.on('touchstart mousedown.xdsoft', '.xdsoft\_option', function () {

if (\_xdsoft\_datetime.currentTime === undefined || \_xdsoft\_datetime.currentTime === null) {

\_xdsoft\_datetime.currentTime = \_xdsoft\_datetime.now();

}

var year = \_xdsoft\_datetime.currentTime.getFullYear();

if (\_xdsoft\_datetime && \_xdsoft\_datetime.currentTime) {

\_xdsoft\_datetime.currentTime[$(this).parent().parent().hasClass('xdsoft\_monthselect') ? 'setMonth' : 'setFullYear']($(this).data('value'));

}

$(this).parent().parent().hide();

datetimepicker.trigger('xchange.xdsoft');

if (options.onChangeMonth && $.isFunction(options.onChangeMonth)) {

options.onChangeMonth.call(datetimepicker, \_xdsoft\_datetime.currentTime, datetimepicker.data('input'));

}

if (year !== \_xdsoft\_datetime.currentTime.getFullYear() && $.isFunction(options.onChangeYear)) {

options.onChangeYear.call(datetimepicker, \_xdsoft\_datetime.currentTime, datetimepicker.data('input'));

}

});

datetimepicker.getValue = function () {

return \_xdsoft\_datetime.getCurrentTime();

};

datetimepicker.setOptions = function (\_options) {

var highlightedDates = {};

options = $.extend(true, {}, options, \_options);

if (\_options.allowTimes && $.isArray(\_options.allowTimes) && \_options.allowTimes.length) {

options.allowTimes = $.extend(true, [], \_options.allowTimes);

}

if (\_options.weekends && $.isArray(\_options.weekends) && \_options.weekends.length) {

options.weekends = $.extend(true, [], \_options.weekends);

}

if (\_options.allowDates && $.isArray(\_options.allowDates) && \_options.allowDates.length) {

options.allowDates = $.extend(true, [], \_options.allowDates);

}

if (\_options.allowDateRe && Object.prototype.toString.call(\_options.allowDateRe)==="[object String]") {

options.allowDateRe = new RegExp(\_options.allowDateRe);

}

if (\_options.highlightedDates && $.isArray(\_options.highlightedDates) && \_options.highlightedDates.length) {

$.each(\_options.highlightedDates, function (index, value) {

var splitData = $.map(value.split(','), $.trim),

exDesc,

hDate = new HighlightedDate(dateHelper.parseDate(splitData[0], options.formatDate), splitData[1], splitData[2]), // date, desc, style

keyDate = dateHelper.formatDate(hDate.date, options.formatDate);

if (highlightedDates[keyDate] !== undefined) {

exDesc = highlightedDates[keyDate].desc;

if (exDesc && exDesc.length && hDate.desc && hDate.desc.length) {

highlightedDates[keyDate].desc = exDesc + "\n" + hDate.desc;

}

} else {

highlightedDates[keyDate] = hDate;

}

});

options.highlightedDates = $.extend(true, [], highlightedDates);

}

if (\_options.highlightedPeriods && $.isArray(\_options.highlightedPeriods) && \_options.highlightedPeriods.length) {

highlightedDates = $.extend(true, [], options.highlightedDates);

$.each(\_options.highlightedPeriods, function (index, value) {

var dateTest, // start date

dateEnd,

desc,

hDate,

keyDate,

exDesc,

style;

if ($.isArray(value)) {

dateTest = value[0];

dateEnd = value[1];

desc = value[2];

style = value[3];

}

else {

var splitData = $.map(value.split(','), $.trim);

dateTest = dateHelper.parseDate(splitData[0], options.formatDate);

dateEnd = dateHelper.parseDate(splitData[1], options.formatDate);

desc = splitData[2];

style = splitData[3];

}

while (dateTest <= dateEnd) {

hDate = new HighlightedDate(dateTest, desc, style);

keyDate = dateHelper.formatDate(dateTest, options.formatDate);

dateTest.setDate(dateTest.getDate() + 1);

if (highlightedDates[keyDate] !== undefined) {

exDesc = highlightedDates[keyDate].desc;

if (exDesc && exDesc.length && hDate.desc && hDate.desc.length) {

highlightedDates[keyDate].desc = exDesc + "\n" + hDate.desc;

}

} else {

highlightedDates[keyDate] = hDate;

}

}

});

options.highlightedDates = $.extend(true, [], highlightedDates);

}

if (\_options.disabledDates && $.isArray(\_options.disabledDates) && \_options.disabledDates.length) {

options.disabledDates = $.extend(true, [], \_options.disabledDates);

}

if (\_options.disabledWeekDays && $.isArray(\_options.disabledWeekDays) && \_options.disabledWeekDays.length) {

options.disabledWeekDays = $.extend(true, [], \_options.disabledWeekDays);

}

if ((options.open || options.opened) && (!options.inline)) {

input.trigger('open.xdsoft');

}

if (options.inline) {

triggerAfterOpen = true;

datetimepicker.addClass('xdsoft\_inline');

input.after(datetimepicker).hide();

}

if (options.inverseButton) {

options.next = 'xdsoft\_prev';

options.prev = 'xdsoft\_next';

}

if (options.datepicker) {

datepicker.addClass('active');

} else {

datepicker.removeClass('active');

}

if (options.timepicker) {

timepicker.addClass('active');

} else {

timepicker.removeClass('active');

}

if (options.value) {

\_xdsoft\_datetime.setCurrentTime(options.value);

if (input && input.val) {

input.val(\_xdsoft\_datetime.str);

}

}

if (isNaN(options.dayOfWeekStart)) {

options.dayOfWeekStart = 0;

} else {

options.dayOfWeekStart = parseInt(options.dayOfWeekStart, 10) % 7;

}

if (!options.timepickerScrollbar) {

timeboxparent.xdsoftScroller('hide');

}

if (options.minDate && /^[\+\-](.\*)$/.test(options.minDate)) {

options.minDate = dateHelper.formatDate(\_xdsoft\_datetime.strToDateTime(options.minDate), options.formatDate);

}

if (options.maxDate && /^[\+\-](.\*)$/.test(options.maxDate)) {

options.maxDate = dateHelper.formatDate(\_xdsoft\_datetime.strToDateTime(options.maxDate), options.formatDate);

}

applyButton.toggle(options.showApplyButton);

month\_picker

.find('.xdsoft\_today\_button')

.css('visibility', !options.todayButton ? 'hidden' : 'visible');

month\_picker

.find('.' + options.prev)

.css('visibility', !options.prevButton ? 'hidden' : 'visible');

month\_picker

.find('.' + options.next)

.css('visibility', !options.nextButton ? 'hidden' : 'visible');

setMask(options);

if (options.validateOnBlur) {

input

.off('blur.xdsoft')

.on('blur.xdsoft', function () {

if (options.allowBlank && (!$.trim($(this).val()).length || (typeof options.mask == "string" && $.trim($(this).val()) === options.mask.replace(/[0-9]/g, '\_')))) {

$(this).val(null);

datetimepicker.data('xdsoft\_datetime').empty();

} else {

var d = dateHelper.parseDate($(this).val(), options.format);

if (d) { // parseDate() may skip some invalid parts like date or time, so make it clear for user: show parsed date/time

$(this).val(dateHelper.formatDate(d, options.format));

} else {

var splittedHours = +([$(this).val()[0], $(this).val()[1]].join('')),

splittedMinutes = +([$(this).val()[2], $(this).val()[3]].join(''));

// parse the numbers as 0312 => 03:12

if (!options.datepicker && options.timepicker && splittedHours >= 0 && splittedHours < 24 && splittedMinutes >= 0 && splittedMinutes < 60) {

$(this).val([splittedHours, splittedMinutes].map(function (item) {

return item > 9 ? item : '0' + item;

}).join(':'));

} else {

$(this).val(dateHelper.formatDate(\_xdsoft\_datetime.now(), options.format));

}

}

datetimepicker.data('xdsoft\_datetime').setCurrentTime($(this).val());

}

datetimepicker.trigger('changedatetime.xdsoft');

datetimepicker.trigger('close.xdsoft');

});

}

options.dayOfWeekStartPrev = (options.dayOfWeekStart === 0) ? 6 : options.dayOfWeekStart - 1;

datetimepicker

.trigger('xchange.xdsoft')

.trigger('afterOpen.xdsoft');

};

datetimepicker

.data('options', options)

.on('touchstart mousedown.xdsoft', function (event) {

event.stopPropagation();

event.preventDefault();

yearselect.hide();

monthselect.hide();

return false;

});

//scroll\_element = timepicker.find('.xdsoft\_time\_box');

timeboxparent.append(timebox);

timeboxparent.xdsoftScroller();

datetimepicker.on('afterOpen.xdsoft', function () {

timeboxparent.xdsoftScroller();

});

datetimepicker

.append(datepicker)

.append(timepicker);

if (options.withoutCopyright !== true) {

datetimepicker

.append(xdsoft\_copyright);

}

datepicker

.append(month\_picker)

.append(calendar)

.append(applyButton);

$(options.parentID)

.append(datetimepicker);

XDSoft\_datetime = function () {

var \_this = this;

\_this.now = function (norecursion) {

var d = new Date(),

date,

time;

if (!norecursion && options.defaultDate) {

date = \_this.strToDateTime(options.defaultDate);

d.setFullYear(date.getFullYear());

d.setMonth(date.getMonth());

d.setDate(date.getDate());

}

if (options.yearOffset) {

d.setFullYear(d.getFullYear() + options.yearOffset);

}

if (!norecursion && options.defaultTime) {

time = \_this.strtotime(options.defaultTime);

d.setHours(time.getHours());

d.setMinutes(time.getMinutes());

}

return d;

};

\_this.isValidDate = function (d) {

if (Object.prototype.toString.call(d) !== "[object Date]") {

return false;

}

return !isNaN(d.getTime());

};

\_this.setCurrentTime = function (dTime, requireValidDate) {

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

\_this.currentTime = \_this.strToDateTime(dTime);

}

else if (\_this.isValidDate(dTime)) {

\_this.currentTime = dTime;

}

else if (!dTime && !requireValidDate && options.allowBlank) {

\_this.currentTime = null;

}

else {

\_this.currentTime = \_this.now();

}

datetimepicker.trigger('xchange.xdsoft');

};

\_this.empty = function () {

\_this.currentTime = null;

};

\_this.getCurrentTime = function (dTime) {

return \_this.currentTime;

};

\_this.nextMonth = function () {

if (\_this.currentTime === undefined || \_this.currentTime === null) {

\_this.currentTime = \_this.now();

}

var month = \_this.currentTime.getMonth() + 1,

year;

if (month === 12) {

\_this.currentTime.setFullYear(\_this.currentTime.getFullYear() + 1);

month = 0;

}

year = \_this.currentTime.getFullYear();

\_this.currentTime.setDate(

Math.min(

new Date(\_this.currentTime.getFullYear(), month + 1, 0).getDate(),

\_this.currentTime.getDate()

)

);

\_this.currentTime.setMonth(month);

if (options.onChangeMonth && $.isFunction(options.onChangeMonth)) {

options.onChangeMonth.call(datetimepicker, \_xdsoft\_datetime.currentTime, datetimepicker.data('input'));

}

if (year !== \_this.currentTime.getFullYear() && $.isFunction(options.onChangeYear)) {

options.onChangeYear.call(datetimepicker, \_xdsoft\_datetime.currentTime, datetimepicker.data('input'));

}

datetimepicker.trigger('xchange.xdsoft');

return month;

};

\_this.prevMonth = function () {

if (\_this.currentTime === undefined || \_this.currentTime === null) {

\_this.currentTime = \_this.now();

}

var month = \_this.currentTime.getMonth() - 1;

if (month === -1) {

\_this.currentTime.setFullYear(\_this.currentTime.getFullYear() - 1);

month = 11;

}

\_this.currentTime.setDate(

Math.min(

new Date(\_this.currentTime.getFullYear(), month + 1, 0).getDate(),

\_this.currentTime.getDate()

)

);

\_this.currentTime.setMonth(month);

if (options.onChangeMonth && $.isFunction(options.onChangeMonth)) {

options.onChangeMonth.call(datetimepicker, \_xdsoft\_datetime.currentTime, datetimepicker.data('input'));

}

datetimepicker.trigger('xchange.xdsoft');

return month;

};

\_this.getWeekOfYear = function (datetime) {

if (options.onGetWeekOfYear && $.isFunction(options.onGetWeekOfYear)) {

var week = options.onGetWeekOfYear.call(datetimepicker, datetime);

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

return week;

}

}

var onejan = new Date(datetime.getFullYear(), 0, 1);

//First week of the year is th one with the first Thursday according to ISO8601

if(onejan.getDay()!=4)

onejan.setMonth(0, 1 + ((4 - onejan.getDay()+ 7) % 7));

return Math.ceil((((datetime - onejan) / 86400000) + onejan.getDay() + 1) / 7);

};

\_this.strToDateTime = function (sDateTime) {

var tmpDate = [], timeOffset, currentTime;

if (sDateTime && sDateTime instanceof Date && \_this.isValidDate(sDateTime)) {

return sDateTime;

}

tmpDate = /^(\+|\-)(.\*)$/.exec(sDateTime);

if (tmpDate) {

tmpDate[2] = dateHelper.parseDate(tmpDate[2], options.formatDate);

}

if (tmpDate && tmpDate[2]) {

timeOffset = tmpDate[2].getTime() - (tmpDate[2].getTimezoneOffset()) \* 60000;

currentTime = new Date((\_this.now(true)).getTime() + parseInt(tmpDate[1] + '1', 10) \* timeOffset);

} else {

currentTime = sDateTime ? dateHelper.parseDate(sDateTime, options.format) : \_this.now();

}

if (!\_this.isValidDate(currentTime)) {

currentTime = \_this.now();

}

return currentTime;

};

\_this.strToDate = function (sDate) {

if (sDate && sDate instanceof Date && \_this.isValidDate(sDate)) {

return sDate;

}

var currentTime = sDate ? dateHelper.parseDate(sDate, options.formatDate) : \_this.now(true);

if (!\_this.isValidDate(currentTime)) {

currentTime = \_this.now(true);

}

return currentTime;

};

\_this.strtotime = function (sTime) {

if (sTime && sTime instanceof Date && \_this.isValidDate(sTime)) {

return sTime;

}

var currentTime = sTime ? dateHelper.parseDate(sTime, options.formatTime) : \_this.now(true);

if (!\_this.isValidDate(currentTime)) {

currentTime = \_this.now(true);

}

return currentTime;

};

\_this.str = function () {

return dateHelper.formatDate(\_this.currentTime, options.format);

};

\_this.currentTime = this.now();

};

\_xdsoft\_datetime = new XDSoft\_datetime();

applyButton.on('touchend click', function (e) {//pathbrite

e.preventDefault();

datetimepicker.data('changed', true);

\_xdsoft\_datetime.setCurrentTime(getCurrentValue());

input.val(\_xdsoft\_datetime.str());

datetimepicker.trigger('close.xdsoft');

});

month\_picker

.find('.xdsoft\_today\_button')

.on('touchend mousedown.xdsoft', function () {

datetimepicker.data('changed', true);

\_xdsoft\_datetime.setCurrentTime(0, true);

datetimepicker.trigger('afterOpen.xdsoft');

}).on('dblclick.xdsoft', function () {

var currentDate = \_xdsoft\_datetime.getCurrentTime(), minDate, maxDate;

currentDate = new Date(currentDate.getFullYear(), currentDate.getMonth(), currentDate.getDate());

minDate = \_xdsoft\_datetime.strToDate(options.minDate);

minDate = new Date(minDate.getFullYear(), minDate.getMonth(), minDate.getDate());

if (currentDate < minDate) {

return;

}

maxDate = \_xdsoft\_datetime.strToDate(options.maxDate);

maxDate = new Date(maxDate.getFullYear(), maxDate.getMonth(), maxDate.getDate());

if (currentDate > maxDate) {

return;

}

input.val(\_xdsoft\_datetime.str());

input.trigger('change');

datetimepicker.trigger('close.xdsoft');

});

month\_picker

.find('.xdsoft\_prev,.xdsoft\_next')

.on('touchend mousedown.xdsoft', function () {

var $this = $(this),

timer = 0,

stop = false;

(function arguments\_callee1(v) {

if ($this.hasClass(options.next)) {

\_xdsoft\_datetime.nextMonth();

} else if ($this.hasClass(options.prev)) {

\_xdsoft\_datetime.prevMonth();

}

if (options.monthChangeSpinner) {

if (!stop) {

timer = setTimeout(arguments\_callee1, v || 100);

}

}

}(500));

$([document.body, window]).on('touchend mouseup.xdsoft', function arguments\_callee2() {

clearTimeout(timer);

stop = true;

$([document.body, window]).off('touchend mouseup.xdsoft', arguments\_callee2);

});

});

timepicker

.find('.xdsoft\_prev,.xdsoft\_next')

.on('touchend mousedown.xdsoft', function () {

var $this = $(this),

timer = 0,

stop = false,

period = 110;

(function arguments\_callee4(v) {

var pheight = timeboxparent[0].clientHeight,

height = timebox[0].offsetHeight,

top = Math.abs(parseInt(timebox.css('marginTop'), 10));

if ($this.hasClass(options.next) && (height - pheight) - options.timeHeightInTimePicker >= top) {

timebox.css('marginTop', '-' + (top + options.timeHeightInTimePicker) + 'px');

} else if ($this.hasClass(options.prev) && top - options.timeHeightInTimePicker >= 0) {

timebox.css('marginTop', '-' + (top - options.timeHeightInTimePicker) + 'px');

}

/\*\*

\* Fixed bug:

\* When using css3 transition, it will cause a bug that you cannot scroll the timepicker list.

\* The reason is that the transition-duration time, if you set it to 0, all things fine, otherwise, this

\* would cause a bug when you use jquery.css method.

\* Let's say: \* { transition: all .5s ease; }

\* jquery timebox.css('marginTop') will return the original value which is before you clicking the next/prev button,

\* meanwhile the timebox[0].style.marginTop will return the right value which is after you clicking the

\* next/prev button.

\*

\* What we should do:

\* Replace timebox.css('marginTop') with timebox[0].style.marginTop.

\*/

timeboxparent.trigger('scroll\_element.xdsoft\_scroller', [Math.abs(parseInt(timebox[0].style.marginTop, 10) / (height - pheight))]);

period = (period > 10) ? 10 : period - 10;

if (!stop) {

timer = setTimeout(arguments\_callee4, v || period);

}

}(500));

$([document.body, window]).on('touchend mouseup.xdsoft', function arguments\_callee5() {

clearTimeout(timer);

stop = true;

$([document.body, window])

.off('touchend mouseup.xdsoft', arguments\_callee5);

});

});

xchangeTimer = 0;

// base handler - generating a calendar and timepicker

datetimepicker

.on('xchange.xdsoft', function (event) {

clearTimeout(xchangeTimer);

xchangeTimer = setTimeout(function () {

if (\_xdsoft\_datetime.currentTime === undefined || \_xdsoft\_datetime.currentTime === null) {

//In case blanks are allowed, delay construction until we have a valid date

if (options.allowBlank)

return;

\_xdsoft\_datetime.currentTime = \_xdsoft\_datetime.now();

}

var table = '',

start = new Date(\_xdsoft\_datetime.currentTime.getFullYear(), \_xdsoft\_datetime.currentTime.getMonth(), 1, 12, 0, 0),

i = 0,

j,

today = \_xdsoft\_datetime.now(),

maxDate = false,

minDate = false,

hDate,

day,

d,

y,

m,

w,

classes = [],

customDateSettings,

newRow = true,

time = '',

h = '',

line\_time,

description;

while (start.getDay() !== options.dayOfWeekStart) {

start.setDate(start.getDate() - 1);

}

table += '<table><thead><tr>';

if (options.weeks) {

table += '<th></th>';

}

for (j = 0; j < 7; j += 1) {

table += '<th>' + options.i18n[globalLocale].dayOfWeekShort[(j + options.dayOfWeekStart) % 7] + '</th>';

}

table += '</tr></thead>';

table += '<tbody>';

if (options.maxDate !== false) {

maxDate = \_xdsoft\_datetime.strToDate(options.maxDate);

maxDate = new Date(maxDate.getFullYear(), maxDate.getMonth(), maxDate.getDate(), 23, 59, 59, 999);

}

if (options.minDate !== false) {

minDate = \_xdsoft\_datetime.strToDate(options.minDate);

minDate = new Date(minDate.getFullYear(), minDate.getMonth(), minDate.getDate());

}

while (i < \_xdsoft\_datetime.currentTime.countDaysInMonth() || start.getDay() !== options.dayOfWeekStart || \_xdsoft\_datetime.currentTime.getMonth() === start.getMonth()) {

classes = [];

i += 1;

day = start.getDay();

d = start.getDate();

y = start.getFullYear();

m = start.getMonth();

w = \_xdsoft\_datetime.getWeekOfYear(start);

description = '';

classes.push('xdsoft\_date');

if (options.beforeShowDay && $.isFunction(options.beforeShowDay.call)) {

customDateSettings = options.beforeShowDay.call(datetimepicker, start);

} else {

customDateSettings = null;

}

if(options.allowDateRe && Object.prototype.toString.call(options.allowDateRe) === "[object RegExp]"){

if(!options.allowDateRe.test(dateHelper.formatDate(start, options.formatDate))){

classes.push('xdsoft\_disabled');

}

} else if(options.allowDates && options.allowDates.length>0){

if(options.allowDates.indexOf(dateHelper.formatDate(start, options.formatDate)) === -1){

classes.push('xdsoft\_disabled');

}

} else if ((maxDate !== false && start > maxDate) || (minDate !== false && start < minDate) || (customDateSettings && customDateSettings[0] === false)) {

classes.push('xdsoft\_disabled');

} else if (options.disabledDates.indexOf(dateHelper.formatDate(start, options.formatDate)) !== -1) {

classes.push('xdsoft\_disabled');

} else if (options.disabledWeekDays.indexOf(day) !== -1) {

classes.push('xdsoft\_disabled');

}else if (input.is('[readonly]')) {

classes.push('xdsoft\_disabled');

}

if (customDateSettings && customDateSettings[1] !== "") {

classes.push(customDateSettings[1]);

}

if (\_xdsoft\_datetime.currentTime.getMonth() !== m) {

classes.push('xdsoft\_other\_month');

}

if ((options.defaultSelect || datetimepicker.data('changed')) && dateHelper.formatDate(\_xdsoft\_datetime.currentTime, options.formatDate) === dateHelper.formatDate(start, options.formatDate)) {

classes.push('xdsoft\_current');

}

if (dateHelper.formatDate(today, options.formatDate) === dateHelper.formatDate(start, options.formatDate)) {

classes.push('xdsoft\_today');

}

if (start.getDay() === 0 || start.getDay() === 6 || options.weekends.indexOf(dateHelper.formatDate(start, options.formatDate)) !== -1) {

classes.push('xdsoft\_weekend');

}

if (options.highlightedDates[dateHelper.formatDate(start, options.formatDate)] !== undefined) {

hDate = options.highlightedDates[dateHelper.formatDate(start, options.formatDate)];

classes.push(hDate.style === undefined ? 'xdsoft\_highlighted\_default' : hDate.style);

description = hDate.desc === undefined ? '' : hDate.desc;

}

if (options.beforeShowDay && $.isFunction(options.beforeShowDay)) {

classes.push(options.beforeShowDay(start));

}

if (newRow) {

table += '<tr>';

newRow = false;

if (options.weeks) {

table += '<th>' + w + '</th>';

}

}

table += '<td data-date="' + d + '" data-month="' + m + '" data-year="' + y + '"' + ' class="xdsoft\_date xdsoft\_day\_of\_week' + start.getDay() + ' ' + classes.join(' ') + '" title="' + description + '">' +

'<div>' + d + '</div>' +

'</td>';

if (start.getDay() === options.dayOfWeekStartPrev) {

table += '</tr>';

newRow = true;

}

start.setDate(d + 1);

}

table += '</tbody></table>';

calendar.html(table);

month\_picker.find('.xdsoft\_label span').eq(0).text(options.i18n[globalLocale].months[\_xdsoft\_datetime.currentTime.getMonth()]);

month\_picker.find('.xdsoft\_label span').eq(1).text(\_xdsoft\_datetime.currentTime.getFullYear());

// generate timebox

time = '';

h = '';

m = '';

line\_time = function line\_time(h, m) {

var now = \_xdsoft\_datetime.now(), optionDateTime, current\_time,

isALlowTimesInit = options.allowTimes && $.isArray(options.allowTimes) && options.allowTimes.length;

now.setHours(h);

h = parseInt(now.getHours(), 10);

now.setMinutes(m);

m = parseInt(now.getMinutes(), 10);

optionDateTime = new Date(\_xdsoft\_datetime.currentTime);

optionDateTime.setHours(h);

optionDateTime.setMinutes(m);

classes = [];

if ((options.minDateTime !== false && options.minDateTime > optionDateTime) || (options.maxTime !== false && \_xdsoft\_datetime.strtotime(options.maxTime).getTime() < now.getTime()) || (options.minTime !== false && \_xdsoft\_datetime.strtotime(options.minTime).getTime() > now.getTime())) {

classes.push('xdsoft\_disabled');

} else if ((options.minDateTime !== false && options.minDateTime > optionDateTime) || ((options.disabledMinTime !== false && now.getTime() > \_xdsoft\_datetime.strtotime(options.disabledMinTime).getTime()) && (options.disabledMaxTime !== false && now.getTime() < \_xdsoft\_datetime.strtotime(options.disabledMaxTime).getTime()))) {

classes.push('xdsoft\_disabled');

} else if (input.is('[readonly]')) {

classes.push('xdsoft\_disabled');

}

current\_time = new Date(\_xdsoft\_datetime.currentTime);

current\_time.setHours(parseInt(\_xdsoft\_datetime.currentTime.getHours(), 10));

if (!isALlowTimesInit) {

current\_time.setMinutes(Math[options.roundTime](\_xdsoft\_datetime.currentTime.getMinutes() / options.step) \* options.step);

}

if ((options.initTime || options.defaultSelect || datetimepicker.data('changed')) && current\_time.getHours() === parseInt(h, 10) && ((!isALlowTimesInit && options.step > 59) || current\_time.getMinutes() === parseInt(m, 10))) {

if (options.defaultSelect || datetimepicker.data('changed')) {

classes.push('xdsoft\_current');

} else if (options.initTime) {

classes.push('xdsoft\_init\_time');

}

}

if (parseInt(today.getHours(), 10) === parseInt(h, 10) && parseInt(today.getMinutes(), 10) === parseInt(m, 10)) {

classes.push('xdsoft\_today');

}

time += '<div class="xdsoft\_time ' + classes.join(' ') + '" data-hour="' + h + '" data-minute="' + m + '">' + dateHelper.formatDate(now, options.formatTime) + '</div>';

};

if (!options.allowTimes || !$.isArray(options.allowTimes) || !options.allowTimes.length) {

for (i = 0, j = 0; i < (options.hours12 ? 12 : 24); i += 1) {

for (j = 0; j < 60; j += options.step) {

h = (i < 10 ? '0' : '') + i;

m = (j < 10 ? '0' : '') + j;

line\_time(h, m);

}

}

} else {

for (i = 0; i < options.allowTimes.length; i += 1) {

h = \_xdsoft\_datetime.strtotime(options.allowTimes[i]).getHours();

m = \_xdsoft\_datetime.strtotime(options.allowTimes[i]).getMinutes();

line\_time(h, m);

}

}

timebox.html(time);

opt = '';

i = 0;

for (i = parseInt(options.yearStart, 10) + options.yearOffset; i <= parseInt(options.yearEnd, 10) + options.yearOffset; i += 1) {

opt += '<div class="xdsoft\_option ' + (\_xdsoft\_datetime.currentTime.getFullYear() === i ? 'xdsoft\_current' : '') + '" data-value="' + i + '">' + i + '</div>';

}

yearselect.children().eq(0)

.html(opt);

for (i = parseInt(options.monthStart, 10), opt = ''; i <= parseInt(options.monthEnd, 10); i += 1) {

opt += '<div class="xdsoft\_option ' + (\_xdsoft\_datetime.currentTime.getMonth() === i ? 'xdsoft\_current' : '') + '" data-value="' + i + '">' + options.i18n[globalLocale].months[i] + '</div>';

}

monthselect.children().eq(0).html(opt);

$(datetimepicker)

.trigger('generate.xdsoft');

}, 10);

event.stopPropagation();

})

.on('afterOpen.xdsoft', function () {

if (options.timepicker) {

var classType, pheight, height, top;

if (timebox.find('.xdsoft\_current').length) {

classType = '.xdsoft\_current';

} else if (timebox.find('.xdsoft\_init\_time').length) {

classType = '.xdsoft\_init\_time';

}

if (classType) {

pheight = timeboxparent[0].clientHeight;

height = timebox[0].offsetHeight;

top = timebox.find(classType).index() \* options.timeHeightInTimePicker + 1;

if ((height - pheight) < top) {

top = height - pheight;

}

timeboxparent.trigger('scroll\_element.xdsoft\_scroller', [parseInt(top, 10) / (height - pheight)]);

} else {

timeboxparent.trigger('scroll\_element.xdsoft\_scroller', [0]);

}

}

});

timerclick = 0;

calendar

.on('touchend click.xdsoft', 'td', function (xdevent) {

xdevent.stopPropagation(); // Prevents closing of Pop-ups, Modals and Flyouts in Bootstrap

timerclick += 1;

var $this = $(this),

currentTime = \_xdsoft\_datetime.currentTime;

if (currentTime === undefined || currentTime === null) {

\_xdsoft\_datetime.currentTime = \_xdsoft\_datetime.now();

currentTime = \_xdsoft\_datetime.currentTime;

}

if ($this.hasClass('xdsoft\_disabled')) {

return false;

}

currentTime.setDate(1);

currentTime.setFullYear($this.data('year'));

currentTime.setMonth($this.data('month'));

currentTime.setDate($this.data('date'));

datetimepicker.trigger('select.xdsoft', [currentTime]);

input.val(\_xdsoft\_datetime.str());

if (options.onSelectDate && $.isFunction(options.onSelectDate)) {

options.onSelectDate.call(datetimepicker, \_xdsoft\_datetime.currentTime, datetimepicker.data('input'), xdevent);

}

datetimepicker.data('changed', true);

datetimepicker.trigger('xchange.xdsoft');

datetimepicker.trigger('changedatetime.xdsoft');

if ((timerclick > 1 || (options.closeOnDateSelect === true || (options.closeOnDateSelect === false && !options.timepicker))) && !options.inline) {

datetimepicker.trigger('close.xdsoft');

}

setTimeout(function () {

timerclick = 0;

}, 200);

});

timebox

.on('touchend click.xdsoft', 'div', function (xdevent) {

xdevent.stopPropagation();

var $this = $(this),

currentTime = \_xdsoft\_datetime.currentTime;

if (currentTime === undefined || currentTime === null) {

\_xdsoft\_datetime.currentTime = \_xdsoft\_datetime.now();

currentTime = \_xdsoft\_datetime.currentTime;

}

if ($this.hasClass('xdsoft\_disabled')) {

return false;

}

currentTime.setHours($this.data('hour'));

currentTime.setMinutes($this.data('minute'));

datetimepicker.trigger('select.xdsoft', [currentTime]);

datetimepicker.data('input').val(\_xdsoft\_datetime.str());

if (options.onSelectTime && $.isFunction(options.onSelectTime)) {

options.onSelectTime.call(datetimepicker, \_xdsoft\_datetime.currentTime, datetimepicker.data('input'), xdevent);

}

datetimepicker.data('changed', true);

datetimepicker.trigger('xchange.xdsoft');

datetimepicker.trigger('changedatetime.xdsoft');

if (options.inline !== true && options.closeOnTimeSelect === true) {

datetimepicker.trigger('close.xdsoft');

}

});

datepicker

.on('mousewheel.xdsoft', function (event) {

if (!options.scrollMonth) {

return true;

}

if (event.deltaY < 0) {

\_xdsoft\_datetime.nextMonth();

} else {

\_xdsoft\_datetime.prevMonth();

}

return false;

});

input

.on('mousewheel.xdsoft', function (event) {

if (!options.scrollInput) {

return true;

}

if (!options.datepicker && options.timepicker) {

current\_time\_index = timebox.find('.xdsoft\_current').length ? timebox.find('.xdsoft\_current').eq(0).index() : 0;

if (current\_time\_index + event.deltaY >= 0 && current\_time\_index + event.deltaY < timebox.children().length) {

current\_time\_index += event.deltaY;

}

if (timebox.children().eq(current\_time\_index).length) {

timebox.children().eq(current\_time\_index).trigger('mousedown');

}

return false;

}

if (options.datepicker && !options.timepicker) {

datepicker.trigger(event, [event.deltaY, event.deltaX, event.deltaY]);

if (input.val) {

input.val(\_xdsoft\_datetime.str());

}

datetimepicker.trigger('changedatetime.xdsoft');

return false;

}

});

datetimepicker

.on('changedatetime.xdsoft', function (event) {

if (options.onChangeDateTime && $.isFunction(options.onChangeDateTime)) {

var $input = datetimepicker.data('input');

options.onChangeDateTime.call(datetimepicker, \_xdsoft\_datetime.currentTime, $input, event);

delete options.value;

$input.trigger('change');

}

})

.on('generate.xdsoft', function () {

if (options.onGenerate && $.isFunction(options.onGenerate)) {

options.onGenerate.call(datetimepicker, \_xdsoft\_datetime.currentTime, datetimepicker.data('input'));

}

if (triggerAfterOpen) {

datetimepicker.trigger('afterOpen.xdsoft');

triggerAfterOpen = false;

}

})

.on('click.xdsoft', function (xdevent) {

xdevent.stopPropagation();

});

current\_time\_index = 0;

/\*\*

\* Runs the callback for each of the specified node's ancestors.

\*

\* Return FALSE from the callback to stop ascending.

\*

\* @param {DOMNode} node

\* @param {Function} callback

\* @returns {undefined}

\*/

forEachAncestorOf = function (node, callback) {

do {

node = node.parentNode;

if (callback(node) === false) {

break;

}

} while (node.nodeName !== 'HTML');

};

/\*\*

\* Sets the position of the picker.

\*

\* @returns {undefined}

\*/

setPos = function () {

var dateInputOffset,

dateInputElem,

verticalPosition,

left,

position,

datetimepickerElem,

dateInputHasFixedAncestor,

$dateInput,

windowWidth,

verticalAnchorEdge,

datetimepickerCss,

windowHeight,

windowScrollTop;

$dateInput = datetimepicker.data('input');

dateInputOffset = $dateInput.offset();

dateInputElem = $dateInput[0];

verticalAnchorEdge = 'top';

verticalPosition = (dateInputOffset.top + dateInputElem.offsetHeight) - 1;

left = dateInputOffset.left;

position = "absolute";

windowWidth = $(window).width();

windowHeight = $(window).height();

windowScrollTop = $(window).scrollTop();

if ((document.documentElement.clientWidth - dateInputOffset.left) < datepicker.parent().outerWidth(true)) {

var diff = datepicker.parent().outerWidth(true) - dateInputElem.offsetWidth;

left = left - diff;

}

if ($dateInput.parent().css('direction') === 'rtl') {

left -= (datetimepicker.outerWidth() - $dateInput.outerWidth());

}

if (options.fixed) {

verticalPosition -= windowScrollTop;

left -= $(window).scrollLeft();

position = "fixed";

} else {

dateInputHasFixedAncestor = false;

forEachAncestorOf(dateInputElem, function (ancestorNode) {

if (window.getComputedStyle(ancestorNode).getPropertyValue('position') === 'fixed') {

dateInputHasFixedAncestor = true;

return false;

}

});

if (dateInputHasFixedAncestor) {

position = 'fixed';

//If the picker won't fit entirely within the viewport then display it above the date input.

if (verticalPosition + datetimepicker.outerHeight() > windowHeight + windowScrollTop) {

verticalAnchorEdge = 'bottom';

verticalPosition = (windowHeight + windowScrollTop) - dateInputOffset.top;

} else {

verticalPosition -= windowScrollTop;

}

} else {

if (verticalPosition + dateInputElem.offsetHeight > windowHeight + windowScrollTop) {

verticalPosition = dateInputOffset.top - dateInputElem.offsetHeight + 1;

}

}

if (verticalPosition < 0) {

verticalPosition = 0;

}

if (left + dateInputElem.offsetWidth > windowWidth) {

left = windowWidth - dateInputElem.offsetWidth;

}

}

datetimepickerElem = datetimepicker[0];

forEachAncestorOf(datetimepickerElem, function (ancestorNode) {

var ancestorNodePosition;

ancestorNodePosition = window.getComputedStyle(ancestorNode).getPropertyValue('position');

if (ancestorNodePosition === 'relative' && windowWidth >= ancestorNode.offsetWidth) {

left = left - ((windowWidth - ancestorNode.offsetWidth) / 2);

return false;

}

});

datetimepickerCss = {

position: position,

left: left,

top: '', //Initialize to prevent previous values interfering with new ones.

bottom: '' //Initialize to prevent previous values interfering with new ones.

};

datetimepickerCss[verticalAnchorEdge] = verticalPosition;

datetimepicker.css(datetimepickerCss);

};

datetimepicker

.on('open.xdsoft', function (event) {

var onShow = true;

if (options.onShow && $.isFunction(options.onShow)) {

onShow = options.onShow.call(datetimepicker, \_xdsoft\_datetime.currentTime, datetimepicker.data('input'), event);

}

if (onShow !== false) {

datetimepicker.show();

setPos();

$(window)

.off('resize.xdsoft', setPos)

.on('resize.xdsoft', setPos);

if (options.closeOnWithoutClick) {

$([document.body, window]).on('touchstart mousedown.xdsoft', function arguments\_callee6() {

datetimepicker.trigger('close.xdsoft');

$([document.body, window]).off('touchstart mousedown.xdsoft', arguments\_callee6);

});

}

}

})

.on('close.xdsoft', function (event) {

var onClose = true;

month\_picker

.find('.xdsoft\_month,.xdsoft\_year')

.find('.xdsoft\_select')

.hide();

if (options.onClose && $.isFunction(options.onClose)) {

onClose = options.onClose.call(datetimepicker, \_xdsoft\_datetime.currentTime, datetimepicker.data('input'), event);

}

if (onClose !== false && !options.opened && !options.inline) {

datetimepicker.hide();

}

event.stopPropagation();

})

.on('toggle.xdsoft', function () {

if (datetimepicker.is(':visible')) {

datetimepicker.trigger('close.xdsoft');

} else {

datetimepicker.trigger('open.xdsoft');

}

})

.data('input', input);

timer = 0;

datetimepicker.data('xdsoft\_datetime', \_xdsoft\_datetime);

datetimepicker.setOptions(options);

function getCurrentValue() {

var ct = false, time;

if (options.startDate) {

ct = \_xdsoft\_datetime.strToDate(options.startDate);

} else {

ct = options.value || ((input && input.val && input.val()) ? input.val() : '');

if (ct) {

ct = \_xdsoft\_datetime.strToDateTime(ct);

} else if (options.defaultDate) {

ct = \_xdsoft\_datetime.strToDateTime(options.defaultDate);

if (options.defaultTime) {

time = \_xdsoft\_datetime.strtotime(options.defaultTime);

ct.setHours(time.getHours());

ct.setMinutes(time.getMinutes());

}

}

}

if (ct && \_xdsoft\_datetime.isValidDate(ct)) {

datetimepicker.data('changed', true);

} else {

ct = '';

}

return ct || 0;

}

function setMask(options) {

var isValidValue = function (mask, value) {

var reg = mask

.replace(/([\[\]\/\{\}\(\)\-\.\+]{1})/g, '\\$1')

.replace(/\_/g, '{digit+}')

.replace(/([0-9]{1})/g, '{digit$1}')

.replace(/\{digit([0-9]{1})\}/g, '[0-$1\_]{1}')

.replace(/\{digit[\+]\}/g, '[0-9\_]{1}');

return (new RegExp(reg)).test(value);

},

getCaretPos = function (input) {

try {

if (document.selection && document.selection.createRange) {

var range = document.selection.createRange();

return range.getBookmark().charCodeAt(2) - 2;

}

if (input.setSelectionRange) {

return input.selectionStart;

}

} catch (e) {

return 0;

}

},

setCaretPos = function (node, pos) {

node = (typeof node === "string" || node instanceof String) ? document.getElementById(node) : node;

if (!node) {

return false;

}

if (node.createTextRange) {

var textRange = node.createTextRange();

textRange.collapse(true);

textRange.moveEnd('character', pos);

textRange.moveStart('character', pos);

textRange.select();

return true;

}

if (node.setSelectionRange) {

node.setSelectionRange(pos, pos);

return true;

}

return false;

};

if(options.mask) {

input.off('keydown.xdsoft');

}

if (options.mask === true) {

if (typeof moment != 'undefined') {

options.mask = options.format

.replace(/Y{4}/g, '9999')

.replace(/Y{2}/g, '99')

.replace(/M{2}/g, '19')

.replace(/D{2}/g, '39')

.replace(/H{2}/g, '29')

.replace(/m{2}/g, '59')

.replace(/s{2}/g, '59');

} else {

options.mask = options.format

.replace(/Y/g, '9999')

.replace(/F/g, '9999')

.replace(/m/g, '19')

.replace(/d/g, '39')

.replace(/H/g, '29')

.replace(/i/g, '59')

.replace(/s/g, '59');

}

}

if ($.type(options.mask) === 'string') {

if (!isValidValue(options.mask, input.val())) {

input.val(options.mask.replace(/[0-9]/g, '\_'));

setCaretPos(input[0], 0);

}

input.on('keydown.xdsoft', function (event) {

var val = this.value,

key = event.which,

pos,

digit;

if (((key >= KEY0 && key <= KEY9) || (key >= \_KEY0 && key <= \_KEY9)) || (key === BACKSPACE || key === DEL)) {

pos = getCaretPos(this);

digit = (key !== BACKSPACE && key !== DEL) ? String.fromCharCode((\_KEY0 <= key && key <= \_KEY9) ? key - KEY0 : key) : '\_';

if ((key === BACKSPACE || key === DEL) && pos) {

pos -= 1;

digit = '\_';

}

while (/[^0-9\_]/.test(options.mask.substr(pos, 1)) && pos < options.mask.length && pos > 0) {

pos += (key === BACKSPACE || key === DEL) ? -1 : 1;

}

val = val.substr(0, pos) + digit + val.substr(pos + 1);

if ($.trim(val) === '') {

val = options.mask.replace(/[0-9]/g, '\_');

} else {

if (pos === options.mask.length) {

event.preventDefault();

return false;

}

}

pos += (key === BACKSPACE || key === DEL) ? 0 : 1;

while (/[^0-9\_]/.test(options.mask.substr(pos, 1)) && pos < options.mask.length && pos > 0) {

pos += (key === BACKSPACE || key === DEL) ? -1 : 1;

}

if (isValidValue(options.mask, val)) {

this.value = val;

setCaretPos(this, pos);

} else if ($.trim(val) === '') {

this.value = options.mask.replace(/[0-9]/g, '\_');

} else {

input.trigger('error\_input.xdsoft');

}

} else {

if (([AKEY, CKEY, VKEY, ZKEY, YKEY].indexOf(key) !== -1 && ctrlDown) || [ESC, ARROWUP, ARROWDOWN, ARROWLEFT, ARROWRIGHT, F5, CTRLKEY, TAB, ENTER].indexOf(key) !== -1) {

return true;

}

}

event.preventDefault();

return false;

});

}

}

\_xdsoft\_datetime.setCurrentTime(getCurrentValue());

input

.data('xdsoft\_datetimepicker', datetimepicker)

.on('open.xdsoft focusin.xdsoft mousedown.xdsoft touchstart', function () {

if (input.is(':disabled') || (input.data('xdsoft\_datetimepicker').is(':visible') && options.closeOnInputClick)) {

return;

}

clearTimeout(timer);

timer = setTimeout(function () {

if (input.is(':disabled')) {

return;

}

triggerAfterOpen = true;

\_xdsoft\_datetime.setCurrentTime(getCurrentValue(), true);

if(options.mask) {

setMask(options);

}

datetimepicker.trigger('open.xdsoft');

}, 100);

})

.on('keydown.xdsoft', function (event) {

var elementSelector,

key = event.which;

if ([ENTER].indexOf(key) !== -1 && options.enterLikeTab) {

elementSelector = $("input:visible,textarea:visible,button:visible,a:visible");

datetimepicker.trigger('close.xdsoft');

elementSelector.eq(elementSelector.index(this) + 1).focus();

return false;

}

if ([TAB].indexOf(key) !== -1) {

datetimepicker.trigger('close.xdsoft');

return true;

}

})

.on('blur.xdsoft', function () {

datetimepicker.trigger('close.xdsoft');

});

};

destroyDateTimePicker = function (input) {

var datetimepicker = input.data('xdsoft\_datetimepicker');

if (datetimepicker) {

datetimepicker.data('xdsoft\_datetime', null);

datetimepicker.remove();

input

.data('xdsoft\_datetimepicker', null)

.off('.xdsoft');

$(window).off('resize.xdsoft');

$([window, document.body]).off('mousedown.xdsoft touchstart');

if (input.unmousewheel) {

input.unmousewheel();

}

}

};

$(document)

.off('keydown.xdsoftctrl keyup.xdsoftctrl')

.on('keydown.xdsoftctrl', function (e) {

if (e.keyCode === CTRLKEY) {

ctrlDown = true;

}

})

.on('keyup.xdsoftctrl', function (e) {

if (e.keyCode === CTRLKEY) {

ctrlDown = false;

}

});

this.each(function () {

var datetimepicker = $(this).data('xdsoft\_datetimepicker'), $input;

if (datetimepicker) {

if ($.type(opt) === 'string') {

switch (opt) {

case 'show':

$(this).select().focus();

datetimepicker.trigger('open.xdsoft');

break;

case 'hide':

datetimepicker.trigger('close.xdsoft');

break;

case 'toggle':

datetimepicker.trigger('toggle.xdsoft');

break;

case 'destroy':

destroyDateTimePicker($(this));

break;

case 'reset':

this.value = this.defaultValue;

if (!this.value || !datetimepicker.data('xdsoft\_datetime').isValidDate(dateHelper.parseDate(this.value, options.format))) {

datetimepicker.data('changed', false);

}

datetimepicker.data('xdsoft\_datetime').setCurrentTime(this.value);

break;

case 'validate':

$input = datetimepicker.data('input');

$input.trigger('blur.xdsoft');

break;

default:

if (datetimepicker[opt] && $.isFunction(datetimepicker[opt])) {

result = datetimepicker[opt](opt2);

}

}

} else {

datetimepicker

.setOptions(opt);

}

return 0;

}

if ($.type(opt) !== 'string') {

if (!options.lazyInit || options.open || options.inline) {

createDateTimePicker($(this));

} else {

lazyInit($(this));

}

}

});

return result;

};

$.fn.datetimepicker.defaults = default\_options;

function HighlightedDate(date, desc, style) {

"use strict";

this.date = date;

this.desc = desc;

this.style = style;

}

}));