Any+Time™ DatePicker/TimePicker AJAX Calendar Widget

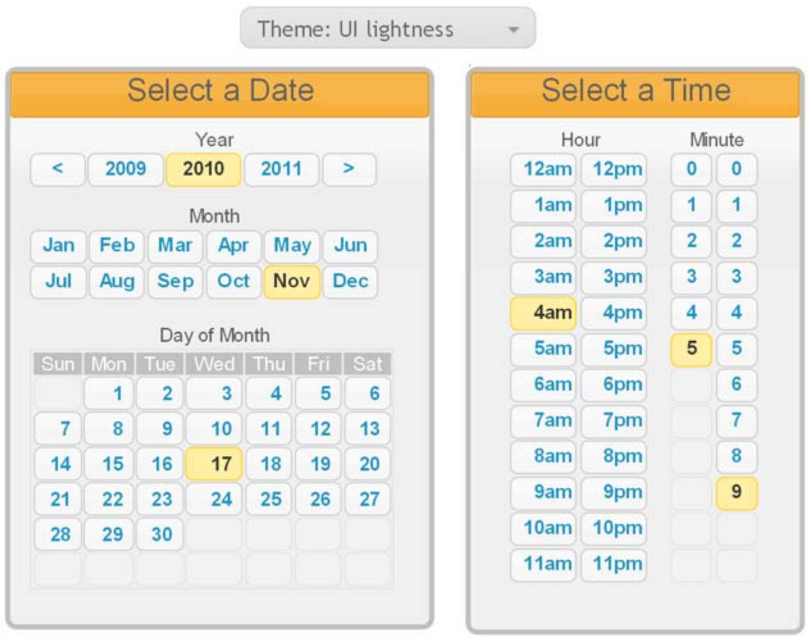
The **Any+Time™ JavaScript Library** includes a highly-customizable, jQuery-compatible datepicker/ timepicker (calendar/ clock widget) and a powerful Date/String parse/format utility.

> der Mercedes, ach was, der Rolls Royce unter den Datepicker-Plugins
> *(the Mercedes, no, the Rolls Royce of Datepicker Plugins)*

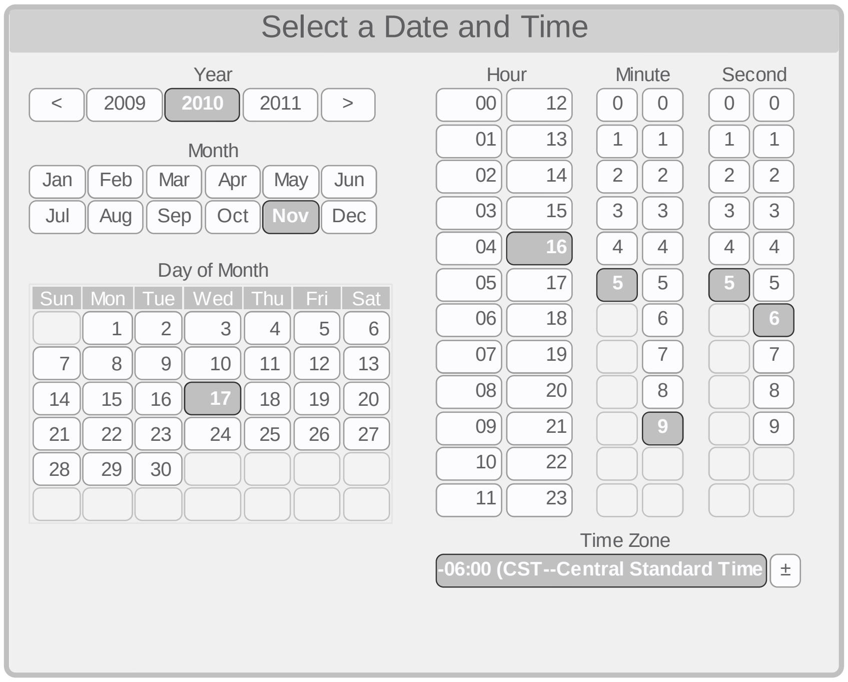
— drweb

Sexy is
OVER-RATED.

Let's try



Above: example datepicker and timepicker, with selectable themes (using jQuery UI Theme Switcher). **Below:** Date/time picker using default style, hidden field, a 24-hour clock and both era and timezone selection, followed by the single JavaScript statement to create it! Click the buttons on any of these widgets to change values. Examples of pop-up pickers and other variations follow.



```
<input type="text" id="DateTimeDemo" />
<script type="text/javascript">
  $("#DateTimeDemo").AnyTime_picker(
      { format: "%Y-%m-%d %H:%i:%s %E %#",
        formatUtcOffset: "%: (%@)",
        hideInput: true,
        placement: "inline" } );
</script>
```

Rip out your old date/time pickers, and drop in **Any+Time**™ this instant! Hurry, before you lose another frustrated user!

the best date/time picker out there

— *Peter Drinnan, OPCMF*

USER-FRIENDLY.

It seems like there's a million calendar and clock widgets out there, and when you look past the eye-candy, they all have one thing in common: they're **tedious** at best. At worst, **counter-intuitive**.

Sure, a picker that uses sliders or spinners, or looks like an analog clock face might be *cute*, but did you ever notice how long it takes to position one to the correct hour and minute? Or how about choosing a month that's more than a season away? Worse yet, entering your birth year on a datepicker that forces you to click... backwards... repeatedly... one... year... at... a... time?

Does your favorite "pretty" picker even work if your user has a keyboard, but not a mouse? Does it scale larger if the user increases the text size on your page? If not, ask your legal department how they'd feel about an accessibility lawsuit!

Enough already!

Any+Time™ is different. More powerful, yes, but more importantly, designed with speed and ease-of-use in mind. And not only can it create a date/time picker with advanced features and options not found in other calendar/clock widgets, it also allows you to format dates and times the way **you** want them. Or your database wants them. Or, better yet, the way your **users** want them.

And you can still **make it sexy**, with plenty of styling options. **`;)`**

For starters, take a look at these DATE/TIME ALTERNATIVES:

- 12-hour or 24-hour clock
- custom date/time format (countless possibilities, including JSON and XML)
- date-only, time-only, or specific fields!
- date/time range limits
- era-selection (BCE/CE, BC/AD, etc.)
- start week on any day (Sunday, Monday, etc.)
- custom base for 2-digit years (1900, 2000, etc.)
- UTC offsets and time zones

Then peep these STYLING CHOICES:

- custom labels/languages
- custom CSS styles
- jQuery UI Theming
- jQuery UI Theme Switcher
- jQuery ThemeRoller
- pop-up or always-present picker
- visible or hidden field

It's also PROGRAMMER-FRIENDLY:

- easy to implement
- easy AJAX validation
- easy Date/String conversion, including JSON and XML
- create multiple pickers at once
- easy removal
- easy to extend

And let's not forget those USABILITY FEATURES:

- single-click value selection
- double-click select-and-dismiss
- WAI-ARIA 1.0 keyboard accessibility
- em-based relative-size

A single JavaScript statement is all you need to add a date and/or time picker to any `<input>` field! Srsly. See the example code? It's for reals, yo.

Use ***AnyTime.Converter*** to parse a String into a Date, or convert a Date to a String. Many format options are supported—in fact, most of the fields specified by the MySQL *DATE_FORMAT()* function!

Any+Time™ uses the free *jQuery JavaScript Library* as a foundation for robust performance. An older version (2.x) of this library, based on Prototype, is still available also.

Any+Time™ has been tested compatible with Chrome 4.1, Firefox 3.6, Internet Explorer 8.0, Opera 10.51 and Safari 4.0, and should work with any version of ECMA-262 (JavaScript, JScript, ECMAScript, etc.) and HTML/XHTML supported by *jQuery*.

Any+Time™ follows WIA-ARIA Authoring Practices 1.0 for Date Picker keyboard interaction as closely as possible, to maximize accessibility without a mouse. Use `Tab` to navigate between the date and time sections, and arrows to navigate between time-selection buttons.

Any+Time™ is **$ FREE $** under the Creative Commons BY-NC-SA 3.0 License. **Tip:** site owners can avoid the need for a commercial license by not charging users to access any page that uses the library; site developers can avoid the need for a commercial license by not charging clients to add the library to their site or to modify the library (code that invokes the library functionality and CSS styles that override the default appearance are not considered modifications as long as the original source files are not modified in any way). A good rule of thumb is: "If nobody has to pay to use **Any+Time**™, then nobody has to pay to use **Any+Time**™!" **If you need a commercial license (or aren't sure), please contact the author** for terms and conditions tailored to your needs.

Instructions

Follow these easy steps to use the **Any+Time™ JavaScript Library** on your website!

1. Download

Any+Time™ consists of a JavaScript source file with a CSS stylesheet file. Both files are formatted to be human-readable, and they contain extensive comments to help you understand and modify them. Right-click on either link to save the file:

extremely easy to use

— *TechnoGadge*

- anytime.js - readable JavaScript source
- anytime.css - readable CSS stylesheet

You may wish to use the compressed (unformatted, no comments) versions instead, to improve download speed for your users:

- anytimec.js - compressed JavaScript source
- anytimec.css - compressed CSS stylesheet

A copy of *jQuery* is also required:

- jquery-1.4.2.min.js - production (compressed) jQuery source

If you want to display locale-specific time zone labels, or allow users to select different UTC offsets with the timepicker, also get the Basic Time Zone Support file (and modify it to meet your needs):

- anytimetz.js - readable JavaScript source for Basic Time Zone Support

PROTOTYPE LIBRARY USERS:

Version 2.x uses the Prototype JavaScript Framework (prototype.js) instead of jQuery. This older version is still available:

- anytime.prototype.js - Prototype-compatible, readable JavaScript source
- anytime.prototype.css - Prototype-compatible, readable CSS stylesheet

Many of the features and behaviors described on this page are different or unavailable in the older version (check the source file comments for more accurate instructions and information).

2. Save and Include

Save a copy of the JavaScript source and CSS stylesheet files on your web server, **remove the last line from the JavaScript file**, and reference both files in your HTML page. For example, if you install jquery.js, anytime.js and anytime.css in the document root directory, then add the following lines to the `<head>` section of the HTML page:

```
<link rel="stylesheet" type="text/css" href="/anytime.css" />
<script type="text/javascript" src="/jquery.js"></script>
<script type="text/javascript" src="/anytime.js"></script>
```

If you downloaded Basic Time Zone Support, be sure to include it last:

```
<script type="text/javascript" src="/anytimetz.js"></script>
```

For proper formatting, the `<link>` element must appear **before** the `<script>` elements!

3. Create HTML Input Fields

Create your date and/or time field as a simple `<input type="text">` element with a unique *id* attribute.

Here are examples of a date-only field that uses a verbose format, and a time-only field with a Spanish label:

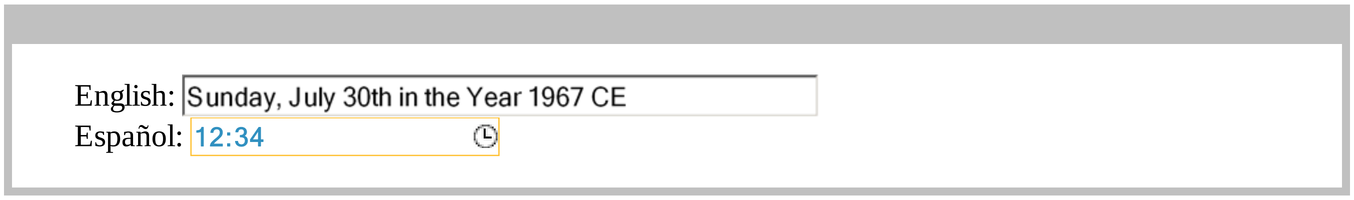
```
English: <input type="text" id="field1" size="50"
    value="Sunday, July 30th in the Year 1967 CE" /><br/>
Español: <input type="text" id="field2" value="12:34" />
```

4. Add JavaScript (and optional CSS)

Call ***AnyTime.picker()*** in your JavaScript code, passing it the *id* of the input element and any desired *options*. Or, use jQuery methods (such as ***$()*** or ***$.find()***) to select one-or-more elements and invoke ***.AnyTime_picker()*** on the result, passing the desired *options*. For example, the code to add pickers to the preceding example fields could be:

```
<script type="text/javascript">
  AnyTime.picker( "field1",
      { format: "%W, %M %D in the Year %z %E", firstDOW: 1 } );
  $("#field2").AnyTime_picker(
      { format: "%H:%i", labelTitle: "Hora",
        labelHour: "Hora", labelMinute: "Minuto" } );
</script>
```

Want a live demonstration? Click one of the following text fields to display the corresponding popup picker! For the first field, try choosing a year in the very distant past.



English: Sunday, July 30th in the Year 1967 CE
Español: 12:34

The first field specifies that the week begins with Monday.

The second field demonstrates a custom style, including a clock pseudo-button for the input field, achieved by the following CSS:

```
<style type="text/css">
  #field2 { background-image:url("clock.png");
    background-position:right center; background-repeat:no-repeat;
    border:1px solid #FFC030;color:#3090C0;font-weight:bold}
  #AnyTime--field2 {background-color:#EFEFEF;border:1px solid #CCC}
  #AnyTime--field2 * {font-weight:bold}
  #AnyTime--field2 .AnyTime-btn {background-color:#F9F9FC;
    border:1px solid #CCC;color:#3090C0}
  #AnyTime--field2 .AnyTime-cur-btn {background-color:#FCF9F6;
      border:1px solid #FFC030;color:#FFC030}
  #AnyTime--field2 .AnyTime-focus-btn {border-style:dotted}
  #AnyTime--field2 .AnyTime-lbl {color:black}
  #AnyTime--field2 .AnyTime-hdr {background-color:#FFC030;color:white}
</style>
```

Refer to the CSS stylesheet for additional details and instructions on custom styles.

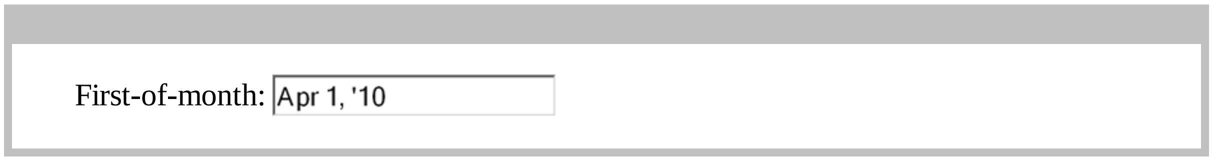
Be sure to create the picker *after* the text field has been added to the page, either by placing your `<script>` element after the `<input>` element, or using jQuery's `$(document).ready()` function.

Options and Format Specifiers

When creating a picker with *AnyTime.picker()* or *$.AnyTime_picker()*, the following members may be specified as part of the *options* argument. Options that are also supported by *AnyTime.Converter* are denoted by a dagger (†) symbol:

ajaxOptions
: Options to pass to jQuery's `$.ajax()` method whenever the user dismisses a popup picker or selects a value in an inline picker. The input's *name* (or *id*) and value are passed to the server (appended to *ajaxOptions.data*, if present), and the "*success*" handler sets the input's value to the responseText. Therefore, the text returned by the server must be valid for the input's date/time *format*, and the server must either echo or correct the value chosen by the user. For example, the server for the following AJAX-enabled picker always changes the day-of-the-month to `1` after the component is dismissed, no matter what day the user actually selects:

```
First-of-month: <input type="text" id="AjaxDemo" value="Apr 1, '10"/>
<script type="text/javascript">
  AnyTime.picker( "AjaxDemo",
      { ajaxOptions: { url: "ajaxdemo.php" },
        baseYear: 2000,
        earliest: new Date(2000,0,1,0,0,0),
        format: "%b %e, '%y"
        latest: new Date(2099,11,31,23,59,59)
      } );
</script>
```


First-of-month: Apr 1, '10

If *ajaxOptions.success* is specified, it is used *instead of* the default "success" behavior. Refer to the jQuery documentation for information about that library's Ajax options.

askEra
: If true, buttons to select the era (BCE/CE) are shown on the year selector popup, even if the *format* specifier does not include the era. If false, buttons to select the era are NOT shown, even if the format specifier includes the era. Normally, era buttons are only shown if the format string specifies the era.

askSecond
: If false, buttons for number-of-seconds are not shown on the year selector popup, even if the *format* specifier includes seconds. Normally, the buttons are shown if the format string specifies seconds.

baseYear†
: the number to add to two-digit years if the `"%y"` *format* specifier is used. By default, the MySQL convention that two-digit years are in the range 1970 to 2069 is used. The most common alternatives are 1900 and 2000. When using this option, you should also specify the *earliest* and *latest* options to the first and last dates in the century, respectively. Refer to the *ajaxOptions* example.

dayAbbreviations†
: An array of day abbreviations to replace `Sun`, `Mon`, etc. **Note:** if a different first day-of-week is specified by option *firstDOW*, this array should nonetheless start with the desired abbreviation for `Sunday`.

dayNames†
: An array of day names to replace `Sunday`, `Monday`, etc. **Note:** if a different first day-of-week is specified by option *firstDOW*, this array should nonetheless start with the desired name for `Sunday`.

earliest
: String or Date object representing the earliest date/time that a user can select. If a String is specified, it is expected to match the *format* specifier. For best results if the field is only used to specify a date, be sure to set the time to `00:00:00`. Refer to the *ajaxOptions* and extending examples.

eraAbbreviations†
: An array of era abbreviations to replace `BCE` and `CE`. The most common replacements are the obsolete: `BC` and `AD`.

firstDOW
: a value from 0 (Sunday) to 6 (Saturday) stating which day should appear at the beginning of the week. The default is `0` (Sunday). The most common substitution is `1` (Monday). **Note:** if custom arrays are specified for *dayAbbreviations* and *dayNames*, they should nonetheless begin with the desired value for `Sunday`. Refer to the earlier popup examples.

format†
: string specifying the date/time format. The following format specifiers are recognized:

specifier	**meaning**
`%a`	Abbreviated weekday name (`Sun`...`Sat`)
`%B`	Abbreviation for Before Common Era (if year<1)*
`%b`	Abbreviated month name (`Jan`...`Dec`)
`%C`	Abbreviation for Common Era (if year>=1)*
`%c`	Month, numeric (`1`..`12`)
`%D`	Day of the month with English suffix (`1st`, `2nd`, ...)
`%d`	Day of the month, numeric (`00`...`31`)
`%E`	Era abbreviation*
`%e`	Day of the month, numeric (`0`...`31`)
`%H`	Hour (`00`...`23`)
`%h`	Hour (`01`...`12`)
`%I`	Hour (`01`...`12`)
`%i`	Minutes, numeric (`00`...`59`)
`%k`	Hour (`0`...`23`)
`%l`	Hour (`1`...`12`)
`%M`	Month name (`January`...`December`)
`%m`	Month, numeric (`01`...`12`)
`%p`	`AM` or `PM`
`%r`	Time, 12-hour (`hh:mm:ss` followed by `AM` or `PM`)
`%S`	Seconds (`00`...`59`)
`%s`	Seconds (`00`...`59`)
`%T`	Time, 24-hour (`hh:mm:ss`)
`%W`	Weekday name (`Sunday`...`Saturday`)
`%w`	Day of the week (`0`=Sunday...`6`=Saturday)
`%Y`	Year, numeric, four digits (possibly signed)
`%y`	Year, numeric, two digits (possibly signed)
`%Z`	Year, numeric, four digits (no sign)*
`%z`	Year, numeric, variable length (no sign)*
`%#`	Signed UTC offset in minutes*
`%+`	Signed UTC offset in `%h%i` format*
`%-`	Signed UTC offset in `%l%i` format*
`%:`	Signed UTC offset in `%h:%i` format*
`%;`	Signed UTC offset in `%l:%i` format*
`%@`	UTC offset time zone label*
`%%`	A literal `%` character

The default format is `"%Y-%m-%d %T"`.

* **Note:** except for those delimited by an asterisk in the table above, these are the same format specifiers used by the MySQL database *DATE_FORMAT()* function. The default format is the one used for MySQL *DATETIME* and *TIMESTAMP* data types.

Any other character in the format string appears literally in the value. Any other sequence of percent sign ("%") followed by a character is reserved for future use, except for the following MySQL specifiers not implemented due to lack of support in JavaScript: `%f` (microseconds); `%j` (day-of-year); `%U`, `%u`, `%V` and `%v` (week-of-year); and `%X` and `%x` (year-for-week). Do not use format specifiers that are reserved or not implemented.

Specifiers and literal characters can be combined into more complex formats, such as JSON and XML.

formatUtcOffset

string specifying the format of the UTC offset choices displayed in the picker. Although all specifiers used by the *format* option are recognized, only those pertaining to UTC offsets (namely `%#`, `%+`, `%-`, `%:`, `%;` and `%@`) should be used. By default, the picker will extrapolate a format by scanning the *format* option for appropriate specifiers and their surrounding characters. Refer to the date/time picker near the beginning of this page for an example.

hideInput

if true, the `<input>` is "hidden" (the picker appears in its place). This actually sets the border, height, margin, padding and width of the field as small as possible, so it can still get focus. Refer to the date/time picker near the beginning of this page for an example. **Note:** if you try to hide the field using traditional techniques (such as setting `display:none`), the picker will not behave correctly. This option should only be used with *placement*`:"inline"`; otherwise, a popup will only appear (seemingly from nowhere) if the user tabs to the hidden field.

labelDayOfMonth

HTML to replace the `Day of Month` label

labelDismiss

HTML to replace the dismiss popup button's `X` label

labelHour

HTML to replace the `Hour` label. Refer to the earlier popup examples.

labelMinute

HTML to replace the `Minute` label. Refer to the earlier popup examples.

labelMonth

HTML to replace the `Month` label

labelSecond

HTML to replace the `Second` label

labelTimeZone

HTML to replace the `Time Zone` label

labelTitle

HTML for the title of the picker. If not specified, the picker automatically selects a title based on the *format* specifier fields. Refer to the earlier popup examples.

labelYear

HTML to replace the `Year` label

latest

String or Date object representing the latest date/time that a user can select. If a String is specified, it is expected to match the *format* specifier. For best results if the field is only used to specify a date, be sure to set the time to 23:59:59. Refer to the *ajaxOptions* and extending examples.

monthAbbreviations†

An array of month abbreviations to replace `Jan`, `Feb`, etc. **Note:** do not use an HTML entity reference (such as `&auml;`) in a month name or abbreviation; embed the actual character (such as ä) instead. Be careful to save your source files under the correct encoding, or the character may not display correctly in all browsers; this often happens when a character code from *UTF-8* is saved with *ISO-8859-1* encoding (or vice-versa).

monthNames†

An array of month names to replace `January`, `February`, etc.

placement

One of the following strings:

`"popup"`

the picker appears above its input when the input receives focus, and disappears when it is dismissed. This is the default behavior.

`"inline"`

the picker follows the `<input>` and remains visible at all times. When choosing this placement, you might prefer to hide the input field using the *hideInput* option (the correct value will still be submitted with the form). Refer to the date/time picker near the beginning of this page for an example.

† denotes options supported by *AnyTime.Converter*.

UTC Offset/Time Zone Customization

Time zone determination is extremely complicated, and ECMA-262 (the JavaScript standard) provides very little native support. Although **Any+Time**™ is a significant improvement, some features may require additional modification to meet your needs.

Default Functionality

By default, `AnyTime.Converter` and any date/time picker created by **Any+Time**™ can parse and/or format offsets from Coordinated Universal Time (UTC) as minutes (`%#`) or hours-and-minutes (`%+`, `%-`, `%:` and `%;`).

UTC offsets can also be represented as time zone labels using the `%@` specifier. By default, this uses the format "`UTC±%h:%m`", where "`UTC`" and "`:`" are literal characters, "`±`" is either a plus or minus sign (for before or after UTC), "`%h`" is the two-digit offset full-hours and "`%m`" is the two-digit offset remaining-minutes. For example, "`UTC+05:30`" represents five (5) hours and thirty (30) minutes ahead of Coordinated Universal Time.

Offset Selection and Time Zone Labels

If it is necessary to change the UTC offset using a date/time picker, or a list of locale-specific, human-friendly time zone labels are needed, then a member named *AnyTime.utcLabel* must be added to the library. This can easily be accomplished by including the anytimetz.js file and modifying it as needed, usually by removing unwanted UTC offsets and/or altering the labels provided.

AnyTime.utcLabel is an Array of Arrays. The primary array is indexed by available UTC offsets **in minutes** (not hours-and-minutes). Plus-sign (+) must **not** be used for positive minutes. Each sub-array contains one or more Strings; each String is a label for a possible time zone corresponding to the UTC offset. For example, the file includes the definition:

```
AnyTime.utcLabel[330]=[
    'IST--Indian Standard Time'
    ,'SLT--Sri Lanka Time'
    ];
```

which means that "`IST--Indian Standard Time`" and "`SLT--Sri Lanka Time`" are the two possible labels for 330 minutes (or 5 hours, 30 minutes) before UTC. The first label in a sub-array is the default label for that UTC offset, so IST will always be selected when *formatUtcOffset* contains the `%@` specifier but *format* does not.

Any label can be altered as desired (for example, you may want to show only abbreviations, or only long names). Any unwanted label can be removed from the sub-array. The sub-array for any unwanted UTC offset can be eliminated entirely, in which case the offset will not be offered by the picker. This can be useful, for example, if you only want to allow time zones for a limited geographic area (such as a single nation or continent).

Offset Conversion

Normally, *AnyTime.Converter* assumes local time when it parses a String or formats a Date. Conversion between local time and other UTC offsets is possible using two options:

utcFormatOffsetImposed
: offset from UTC, **in minutes**, to specify when formatting a Date object. This can be used to convert a local time to a different UTC offset. Refer to the example in the next section.

utcParseOffsetAssumed
: offset from UTC, **in minutes**, to assume when parsing a String object. This can be used to convert a String created in a different UTC offset to local time. **Note:** if the format string contains a UTC offset specifier (`%#`, `%+`, `%-`, `%:`, `%;` or `%@`), then the UTC offset specified in the String is used instead of *utcParseOffsetAssumed*.

Explanations of more obscure options for UTC offset manipulation appear in the source file.

Limitations

Unlike days and months, there are not separate labels for time zone abbreviations and long names. anytimetz.js can be modified to contain only one or the other, but the library does not provide the ability to select between abbreviations or long names using different format specifiers, in part because many abbreviations are ambiguous. Separate labels might be added to a future version, but no such work is currently underway.

There is no automatic detection of daylight savings time (AKA summer time), due to lack of support in JavaScript and the time-prohibitive complexity of attempting such support in code (alternate time zones are inconsistent from location-to-location and year-to-year, and relevant time zone data is updated many times per year)! If you are concerned that users will not know whether to select Standard or Daylight/Summer Time for a particular date, you can eliminate the Daylight/Summer members from the array, and remove the word "Standard" from the Standard labels. For example, instead of:

```
AnyTime.utcLabel[-300]=[
  'EST--Eastern Standard Time (North America)' ];
AnyTime.utcLabel[-240]=[
  'EDT--Eastern Daylight Time (North America)' ];
```

modify the array to contain:

```
AnyTime.utcLabel[-300]=[
    'Eastern Time (North America)' ];
```

and only use the label (`%@`) specifier (do not use `%#`, `%+`, `%-`, `%:` or `%;` because the UTC offset will be misrepresented). This is an effective solution for most cases, which do not require conversion between local time and different time zones.

Convert Dates to/from Strings (including JSON and XML!)

AnyTime.Converter can be used independently. The following example converts a string in the default date/time format into a Date object, then converts the Date into a JSON string (with distinct members) and an XML string (using the XML Schema dateTime data type and Coordinated Universal Time):

```
var defaultConv = new AnyTime.Converter();
var date = defaultConv.parse("1990-01-06 15:30:00");

var jsonConv = new AnyTime.Converter({format:
  '{"year":"%Y","month":"%m","day":"%d","hour":"%H","minute":"%i","second":"%s"}'});

var xmlConv = new AnyTime.Converter({utcFormatOffsetImposed: 0,
  format:"<"+"date>%Y-%m-%dT%H:%i:%s%:<"+"/date>"});

alert( "JSON:\n" + jsonConv.format(date) + "\n\nXML:\n" + xmlConv.format(date) );
```

AnyTime.Converter accepts the following *options*, which are the same as for *AnyTime.picker()* and *jQuery.AnyTime_picker()*: *baseYear*, *dayAbbreviations*, *dayNames*, *eraAbbreviations*, *format*, *monthAbbreviations* and *monthNames*.

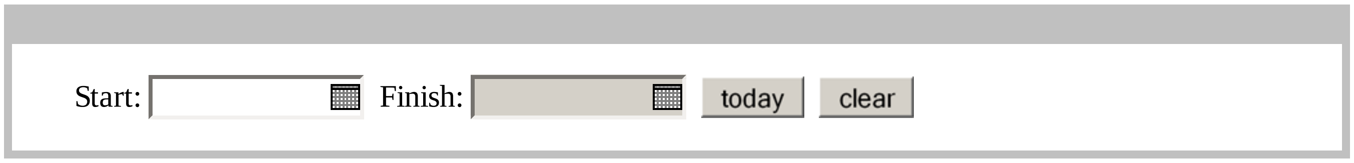
AnyTime.Converter supports all of the same format field specifiers as *AnyTime.picker()* and *jQuery.AnyTime_picker()*.

Check the JavaScript source code for additional details and instructions.

Extending Any+Time™ Functionality

In the following example, *AnyTime.Converter* and jQuery work together to provide date-range selection. The value for the second ("*Finish*") field must be at least one day after the date in the first ("*Start*") field (thanks to the earliest option), but no more than 90 days later (thanks to the latest option). This example also demonstrates a button that sets the first field to the current date, a button to clear the fields, and calendar pseudo-buttons using CSS background properties.

```
<style type="text/css">
  #rangeDemoStart, #rangeDemoFinish {
    background-image:url("calendar.png");
    background-position:right center;
    background-repeat:no-repeat; }
</style>
Start: <input type="text" id="rangeDemoStart" size="14" />
Finish: <input type="text" id="rangeDemoFinish" size="14" disabled="disabled"/>
<input type="button" id="rangeDemoToday" value="today" />
<input type="button" id="rangeDemoClear" value="clear" />
<script type="text/javascript">
  var oneDay = 24*60*60*1000;
  var rangeDemoFormat = "%e-%b-%Y";
  var rangeDemoConv = new AnyTime.Converter({format:rangeDemoFormat});
  $("#rangeDemoToday").click( function(e) {
        $("#rangeDemoStart").val(rangeDemoConv.format(new Date())).change(); } );
    $("#rangeDemoClear").click( function(e) {
        $("#rangeDemoStart").val("").change(); } );
    $("#rangeDemoStart").AnyTime_picker({format:rangeDemoFormat});
    $("#rangeDemoStart").change( function(e) { try {
        var fromDay = rangeDemoConv.parse($("#rangeDemoStart").val()).getTime();
        var dayLater = new Date(fromDay+oneDay);
        dayLater.setHours(0,0,0,0);
        var ninetyDaysLater = new Date(fromDay+(90*oneDay));
        ninetyDaysLater.setHours(23,59,59,999);
        $("#rangeDemoFinish").
            AnyTime_noPicker().
            removeAttr("disabled").
            val(rangeDemoConv.format(dayLater)).
            AnyTime_picker(
                { earliest: dayLater,
                  format: rangeDemoFormat,
                  latest: ninetyDaysLater
                } );
        } catch(e){ $("#rangeDemoFinish").val("").attr("disabled","disabled"); } } );
  </script>
```



A few people have requested the ability to type a value directly into the `<input>` field. Such behavior would be detrimental: not only does the picker make it easy to select the appropriate date/time, it also protects the user from entering a value in the wrong format! In addition, navigation keys (such as arrows) are used by the picker to select values, and would behave inappropriately for data entry. Therefore, although it is possible to hack the picker to permit typing in the field (simply remove the call to `event.preventDefault()` at the end of the `key` method), **doing so is <u>not</u> recommended**! **Note:** in most browsers, it *is* possible to paste a value into the input using CTRL-V; if the pasted value is not in the proper format, the current time is used (subject to any range limits).

Troubleshooting Tips

Following are some of the most common issues, and how to avoid or solve them. Even if you do not experience any problems, these are good rules to follow!

Display/Layout Issues

Most display/layout problems can be avoided by keeping the browser out of quirks mode. Common symptoms include groups of buttons appearing in the wrong location, especially in older versions of IE. For best results, be sure to include an appropriate `<!DOCTYPE>` declaration as the first line of your HTML page; for example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd">
```

Any legitimate HTML or XHTML `<!DOCTYPE>` can be used, as long as it is appropriate for the source file.

Note that an `<input>` field must be visible when its picker is created, in order to calculate the correct size and location (if the field is hidden, the picker might be too small to contain all of its buttons). To create a picker for a field that is initially hidden (for example, because it is on a collapsed accordian panel or "minimized" popup overlay), wait until the first time the field is made visible before calling *AnyTime.picker()* or *$.AnyTime_picker()*. For example, you can attach an initial `focus()` handler to the field that instantiates the picker:

```
<input id="HiddenFieldDemo">
<script type="text/javascript">
  $('#HiddenFieldDemo').focus( function() {
    $('#HiddenFieldDemo').unbind('focus').AnyTime_picker(); } );
</script>
```

When using a jQuery UI theme, be sure to include the **Any+Time**™ library stylesheet also. For best results, include the library stylesheet **before** the theme stylesheet. If you add the theme styles to the library stylesheet, place them at the end of the stylesheet for best results.

Also be aware that many jQueryUI themes use a *background* image for the *ui-widget-content* class that is shorter than a typical picker, causing the widget to have a "two-tone" background. If you do not like the appearance, modify the property set in the "Component containers" section of your theme stylesheet to:

- choose a different background color that blends more smoothly with the top of the image;
- remove the *background* `url("…")` property for a solid background color;
- make the `background repeat` in both directions; or
- substitute a taller image.

Minor variations in Microsoft Internet Explorer 6 and 7 are to be expected due to their broken box model, but nothing should appear too out-of-the-ordinary.

Some display problems are related to initialization conflicts with other libraries, especially .NET. These can be resolved using the `setTimeout()` function to delay picker creation until other initializations are finished; for example:

```
setTimeout( function() { AnyTime.picker("field1"); }, 1000 );
```

In some cases, it may be necessary to increase the timeout or use more complicated techniques, such as waiting to create the picker when the field first receives focus (as previously shown).

Be careful to place all `<link rel="stylesheet">` and `<style>` elements **before** `<script>` elements, or WebKit-based browsers (Apple Safari and Google Chrome) might not format the picker correctly (symptoms include extremely-tall pickers and misplaced time buttons).

All names, IDs and classes in HTML, JavaScript and CSS created by **Any+Time**™ contain the phrase "*AnyTime*". To avoid problems related to naming conflicts, do not use this sequence of characters in any variables, elements or class names.

Behavior Issues

If the picker does not appear at all, be sure you are trying create it *after* the text field has been added to the page, either by placing your `<script>` element *after* the `<input>` element, or using jQuery's `$(document).ready()` function.

Every `<input>` field for which a picker is to be created must have a unique *id* attribute, even if the picker is created by passing a non-ID selector to *jQuery.AnyTime_picker()*. An exception might be thrown if the ID is missing.

Use ***AnyTime.picker()*** instead of ***jQuery.AnyTime_picker()*** if the ID for a field contains characters that are misinterpreted by jQuery. For example, JSF (Java Server Faces) has been known to use the colon (`":"`) character in generated IDs, which is used by jQuery for pseudo-selectors.

If a picker shows today's date/time instead of the value in the associated `<input>` field, check that the format of the value matches the *format* specifier exactly. The value cannot be interpreted if it does not match perfectly!

Do not use an HTML entity reference (such as `&auml;`) in a string passed to the *monthAbbreviations* option; embed the actual character (such as ä) instead. Be careful to save your source files under the correct encoding, or the character may not display correctly in all browsers; this often happens when a character code from *UTF-8* is saved with *ISO-8859-1* encoding (or vice-versa).

When using *placement*`:"inline"`, XHTML and a day-of-the-month *format* specifier (`"%D"`, `"%d"` or `"%e"`), the `<input>` may only appear where a `<table>` element is permitted (for example, NOT within a `<p>` element). This is because the picker uses a `<table>` element to arrange the day-of-the-month (calendar) buttons. Failure to follow this advice may result in a JScript "unknown error" from Internet Explorer.

When specifying the earliest and/or latest option, be certain to include a time value, even if the user is only able to select the date. The time for an "earliest" date should be `00:00:00`, and the time for a "latest" date should be `23:59:59` (refer to the *ajaxOptions* example). This is because the Date objects used by the picker reflect exact moments in time, regardless of which fields are specified by the *format*. Failure to set the time could result in incorrect enforcement if, for example, the page is loaded at 23:59 one day, but the field not changed until 00:01 the day after!

Only use the *hideInput* option to hide the `<input>` associated with a picker. Traditional techniques (such as setting `display:none`) will cause the picker to behave incorrectly.

Some older versions of jQuery are incompatible with **Any+Time**™. JavaScript errors, often involving `null` or `undefined` objects, may result. After installing an updated version of jQuery, be sure to clear the browser cache so the new version is loaded.

All names, IDs and classes in HTML, JavaScript and CSS created by **Any+Time**™ contain the phrase "*AnyTime*". To avoid problems related to naming conflicts, do not use this sequence of characters in any variables, elements or class names.

Validation Issues

It should go without saying that **Any+Time**™ only works in browsers with JavaScript enabled. Any server-side form processing should validate every value it receives, in case JavaScript was disabled or otherwise unavailable when the form was submitted.

Remember that two-digit years (`"%y"`) are susceptible to the Y2K problem! For best results, use four-digit or variable-length years (`"%Y"`, `"%Z"` or `"%z"`) instead. The *baseYear* option can also be helpful in situations where `"%y"` is required. When using *baseYear*, you should also specify the *earliest* and *latest* options to the first and last dates in the century, respectively.

Other Issues

To reduce memory leaks, always call ***AnyTime.noPicker()*** (or the *.AnyTime_noPicker()* extension to jQuery) to remove the date/time picker from an `<input>` field before removing the field, for example:

```
AnyTime.noPicker("field1");
$("#field1").remove();
$("#field2").AnyTime_noPicker().remove();
```

This is especially necessary before adding a picker to a field with the same ID as a previously-removed field that also had a picker, because **Any+Time**™ will not create more than one picker per ID.

The JavaScript source files include an intrusive `alert()` call on the last line, to discourage hot-linking to this server. When you download a source file, be sure to remove the last line to eliminate annoying messages when your HTML page is loaded!

Any+Time™ follows WIA-ARIA Authoring Practices 1.0 for Date Picker keyboard interaction as closely as possible, to maximize accessibility without a mouse. However, if a user reports difficulty changing a date/time value using the picker (for example, due to problems with a "screen reader" or other assistive technology), ask them to disable JavaScript and carefully type the value into the input field. Again, be sure to validate the input when it is received by the server.

If you experience any other problems, please contact the author.

Interface Reference

Any+Time™ provides the following public objects and methods. Check the source code for additional methods that are not intended for general use, but potentially-helpful to more advanced web developers.

`(Object) new AnyTime.Converter( Object` *options* `)`
: Creates an object for parsing Strings into Dates and formatting Dates as Strings, using the specified *options*.

`(String) AnyTime.Converter.format( Date date )`
: Returns a String representing the specified Date.

`(Date) AnyTime.Converter.parse( String string )`
: Returns a Date represented by the specifed string.

`(void) AnyTime.noPicker( String input_id )`
: Removes the picker associated with the `<input>` having the specified ID, and cleans up the memory used by the widget.

`(void) AnyTime.picker( String input_id, Object` *options* `)`
: Creates a date and/or time picker for the `<input>` having the specified ID, according to the specified *options*.

`(jQuery) jQuery.AnyTime_noPicker()`
: Removes the pickers associated with each of the elements selected by jQuery. Refer to the extending example.

`(jQuery) jQuery.AnyTime_picker( Object` *options* `)`
: Creates a date and/or time picker for each of the elements selected by jQuery according to the specified *options*.

Share Your Success!

If you find **Any+Time**™ useful, please tell your colleagues and promote the library in your favorite forums, blogs and other social networks!

Both positive feedback and constructive criticism are also appreciated (please contact the author).