TimeCircles documentation

TimeCircles is a jQuery plugin that provides a nice looking way to either count down towards a certain time, or to count up from a certain time. The goal for TimeCircles is to provide a simple yet dynamic tool that makes it very easy to provide visitors an attractive countdown or timer.

This documentation will provide some examples of how to use TimeCircles. Usage of TimeCircles can be very simple, but for those willing to work a little harder can also provide more sophisticated functionality. The examples aim to provide a good basic idea of how various features can be used without overcomplicating things.

General use

The first thing to do is to include the javascript files for jQuery and TimeCircles, as well as the TimeCircles stylesheet. These should ideally be included in the head of your html file.

```HTML
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
<script type="text/javascript" src="TimeCircles.js"></script>
<link href="TimeCircles.css" rel="stylesheet">
```

When the neccesary files have been included, it's very simple to set up TimeCircles on your page, simply target the element you wish to use with jQuery, and execute the TimeCircles function on it. This will create TimeCircles inside the targeted element, counting up from 0 (when the page was opened)


```Javascript
$(".example").TimeCircles();
```

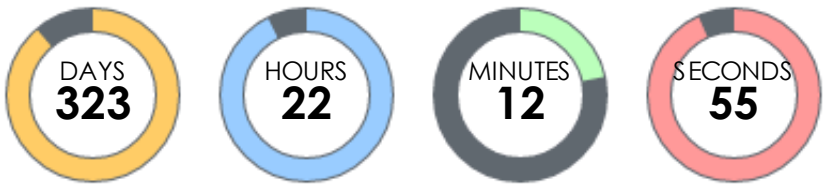
Important note: TimeCircles will automatically make it self the size of whatever element you place it in. If you do not

have a height set, it will attempt to determine a height based on the element's width. For the best results however, it's recommended to set both the width and height.

Setting your reference time

Of course, you might not want to start counting up from 0. Perhaps you're counting down the time until a wedding, or alternatively counting how long you've been with your girlfriend (or whatever else really). Really, TimeCircles is most useful if you're using it with some reference time and/or date.

Setting up your reference date and time is also fairly simple. The best way to do it is to simply include it in the html element you've set aside for TimeCircles. Create an attribute called `data-date` and provide a value in the format of `yyyy-mm-dd hh:mm:ss`



HTML

```
<div class="example" data-date="2014-01-01 00:00:00"></div>
```

Creating a stopwatch

It is also possible that you want to use TimeCircles to count down a specific amount of time, like 15 minutes. This works similarly as creating a reference time, however here the attribute `data-timer` is used, and the value is the time to count down from (*in seconds*).

TimeCircles documentation
General use
Setting your reference time
Creating a stopwatch
Options
Functions

HTML

```
<div class="example" data-timer="900"></div>
```

Options

It's nice that TimeCircles comes in yellow, green, blue, and red- but wouldn't it be even nicer if it came in the color theme of your own website? Alternatively, wouldn't it be great if you could change other aspects of the way it looks? Perhaps change the language of the text to whatever you want?

To customize TimeCircles to fit precisely what you're looking for you can use the options. In this section we will look into what options are available and what each option means.

start `(default: true)`

This option determines whether or not TimeCircles should start immediately. If for example you wish to create a stopwatch that starts when the users clicks a button, you'll want to set this to false.

```Javascript
$(".example").TimeCircles({start: false});
```

animation `(default: "smooth")`

The way the circles animate can be either a constant gradual rotating, slowly moving from one second to the other. Or it can be set to jump from one unit to the next, only changing the circle when the number within it also changes. The first type is called `"smooth"`, the second type is called `"ticks"`.

```Javascript
$(".example").TimeCircles({animation: "smooth"});
```

count_past_zero `(default: true)`

This option is only really useful for when counting down. What it does is either give you the option to stop the timer, or start counting up after you've hit the predefined date (or your stopwatch hits zero).

```Javascript
$(".example").TimeCircles({count_past_zero: false});
```

circle_bg_color `(default: "#60686F")`

This option determines the color of the background circle.

```Javascript
$(".example").TimeCircles({circle_bg_color: "#000000"});
```

use_background `(default: true)`

This options sets whether any background circle should be drawn at all. Disabling this option could be used in isolation, or you could use a background of your own to place behind TimeCircles.

```Javascript
$(".example").TimeCircles({use_background: false});
```

fg_width (default: 0.1)

This option sets the width of the foreground circle. The width is set relative to the size of the circle as a whole. A value of 0.1 means 10%, so if your TimeCircles are 100 pixels high, the foreground circle will be 10 percent of that (10 pixels).

```Javascript
$(".example").TimeCircles({fg_width: 0.05});
```

bg_width (default: 1.2)

This option sets the width of the backgroundground circle. The width of the background is set relative to the width of the foreground. A value of 1 means 100%, so a value of 1 would mean having a width equal to your foreground ring. Higher and you get wider, lower you get thinner.

```Javascript
$(".example").TimeCircles({bg_width: 0.5});
```

total_duration (default: "Auto")

This option can be set to change how much time will fill the largest visible circle. Normally this is the Days circle, but this can be any of the other circles depending on visible settings. Valid values for this variable are `"Auto"`, `"Years"`, `"Months"`, `"Days"`, `"Hours"`, `"Minutes"` or any numeric value (ie: 30 for thirty seconds).

```Javascript
$(".example").TimeCircles({total_duration: "Auto"});
```

direction (default: "Clockwise")

This option can be set to change the direction in which the circles fill up. Valid values for this are `"Clockwise"`, `"Counter-clockwise"` or `"Both"`.

```Javascript
$(".example").TimeCircles({direction: "Clockwise"});
```

start_angle (default: 0)

This option can be set to change the starting point from which the circles will fill up. This should be an integer value between 0 and 360. 0 is from the top, 90 from the right, 180 from the bottom and 270 from the left.

```Javascript
$(".example").TimeCircles({start_angle: 0});
```

time

The time option is actually a group of options that allows you to control the options of each time unit independently. As such, within time each unit of time has its own sub-category. These categories are: Days, Hours, Minutes, and Seconds. The options available within each category are as follows:

- **show**: Determines whether the time unit should be shown at all
- **text**: Determines the text shown below the time. Useful for use on non-English websites
- **color**: Determines the color of the foreground circle of the time unit

```Javascript
$(".example").TimeCircles({ time: {
    Days: { color: "#C0C8CF" },
    Hours: { color: "#C0C8CF" },
    Minutes: { color: "#C0C8CF" },
    Seconds: { color: "#C0C8CF" }
}});
```

Functions

Functions will allow you to interact with your TimeCircles as they're running. Generally speaking, this functionality is most often used by other developers who want their own javascript to interact with TimeCircles. However, if you're not a developer yourself, there are still a few functions that are quite simple to use and shouldn't be too hard to tackle.

Before we go into what each function does however, it should be pointed out how these functions can be used. Unlike quite a lot of other jQuery plugins, TimeCircles does not return a jQuery object after instantiating. Instead, it returns a TimeCircles object. This means that function chaining will work slightly differently than it does for other jQuery plugins. To find out more about how to chain other jQuery plugins and functions, have a look at the `end()` function.

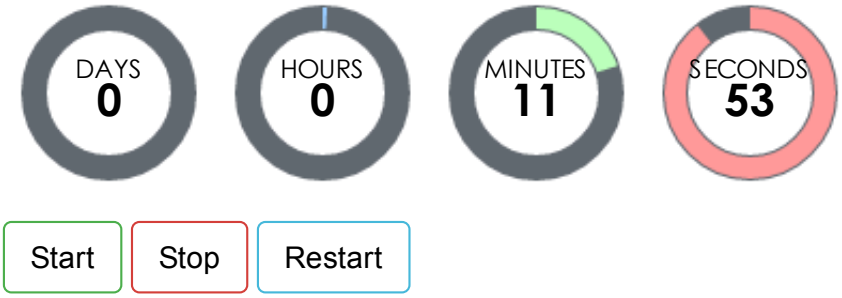
TimeCircles functions themselves (with the exception of the `end()` function) will return the TimeCircles object. This allows you to chain several functions into each other. IE: You could chain `start()` straight into `addEventListener(callback)`.

start() , stop() and restart()

These are the most basic functions provided. They allow you to temporarily stop TimeCircles or restart it. Start will unpause the timer when it's stopped, restart will restart it from it's original value. If you're using TimeCircles to count down to a certain point in the future, upon unpausing the countdown will jump ahead.

```HTML
<div class="example stopwatch" data-timer="900"></div>
```

```
<button class="btn btn-success start">Start</button>
<button class="btn btn-danger stop">Stop</button>
<button class="btn btn-info restart">Restart</button>
```



Javascript

```
$(".example.stopwatch").TimeCircles();
$(".start").click(function(){ $(".example.stopwatch").TimeCircles().start(); });
$(".stop").click(function(){ $(".example.stopwatch").TimeCircles().stop(); });
$(".restart").click(function(){ $(".example.stopwatch").TimeCircles().restart(); });
```

destroy()

If for some reason, you need to get rid of your TimeCircles, or you want to allow users remove them at the click of a button; you can do that with destroy.

Javascript

```
$(".example").TimeCircles().destroy();
```

rebuild()

Some options or variables are only initialized once, at the time of creating TimeCircles (For example, the width and height, or which circles are being shown). If you change these settings later on, you can have everything reinitialized anew by using `.rebuild()`

Javascript

```
$(".example").TimeCircles().rebuild();
```

getTime()

Retrieves the number seconds left (or since) the zero point. Values until zero are positive, values after zero are negative.

Javascript

```
$(".example").TimeCircles().getTime();
```

addListener `(callback, type = "visible")`

The most powerful interactions with TimeCircles can be achieved using listeners. Using listeners, you can make a ticking sound play every second, or you can make a sound whenever a minute passes. You could even use it to trigger some alarm or whole other javascript when the timer hits zero.

The type parameter allows you to listen to either only the events from the visible TimeCircles, or all TimeCircles. Correct values are `"visible"` or `"all"`. The default value is `"visible"`, but through using `"all"` you can listen to the circles you're hiding (if you're hiding any of course).

To add a listener, use the `addEventListener(callback)` function. Callback is a function you pass to the event listener. The callback will then be triggered for each event. Three parameters are passed to your callback function, namely:

- **unit**: The time unit in string format. So, "Days"/"Hours"/"Minutes"/"Seconds".
- **value**: The new value of the time unit that changed. I.e.: 15.
- **total**: This is the total time left (or elapsed) since the zero point.

end()

To allow you to chain TimeCircles to other jQuery functions, you can use the `end()` function. The end function returns the jQuery object and allows you to trigger jQuery function as desired.

```Javascript
$(".example").TimeCircles().end().fadeOut();
```