**Nano Timers API Documentation**

*Version: 0.1.2beta*

*Date: 2017-08-09*

**Summary:**

Timer Library that includes countdown timers and stopwatches.

**Contents:**

Structures

* [Time](#Time)

Enumerables

* [ETimerState](#ETimerState)

Classes

* [Timer](#Timer)
* [CountdownTimer](#CountdownTimer)
* [Stopwatch](#Stopwatch)

**Time**

* **Summary**
* Container for minutes, seconds, and milliseconds
* **Remarks**
* “Time” is a structure that already exists in the UnityEngine namespace.
* **Namespace**
* NanoTimers
* **Members**
* **Public**
* minutes, int
* seconds, int
* millis, int
* **Constructor**
* **Params**
* minutes, int
* seconds, int
* millis, int

**ETimerState**

* **Summary**
* States that define a timer.
* **Namespace**
* NanoTimers
* **Members**
* Active
* Paused
* Expired
* New

**Timer**

* **Summary**
* Framework for a basic timer.
* **Namespace**
* NanoTimers
* **Modifiers**
* Abstract
* **Parent Class**
* MonoBehaviour
* **Fields**
* **Protected**
* m\_minutes, int
* **Summary**
* The number of minutes currently on this timer.
* m\_seconds, int
* **Summary**
* The number of seconds currently on this timer.
* m\_millis, int
* **Summary**
* The number of milliseconds currently on this timer.
* m\_showMillis, bool
* **Summary**
* Will this timer also display milliseconds?
* m\_state, [ETimerState](#ETimerState)
* **Summary**
* The current state of this timer.
* m\_timerText, Text
* **Summary**
* Reference to the UI Text object where the timer’s minutes and seconds are displayed.
* m\_pausedTimer, [CountdownTimer](#CountdownTimer)
* **Summary**
* Internal timer that determines for how long this timer should be paused.
* m\_originalState, [NanoTimers.Time](#Time)
* **Summary**
* Initial time this timer was initialized to. Used for resets.
* m\_isInitialized, bool
* **Summary**
* Determines if this timer has been properly initialized and prevents it from being started with uninitialized values.
* **Properties**
* State, [ETimerState](#ETimerState)
* **Summary**
* Only returns this timer’s current state.
* Time, [NanoTimers.Time](#Time)
* **Summary**
* Only returns this timer’s current time.
* **Constructor**
* Unity Default
* **Methods**
* **Public**
* StartTimer, void
* **Summary**
* Enables this timer’s active state if initialized.
* Pause, void
* **Summary**
* Pauses this timer indefinitely.
* PauseForSeconds, void
* **Summary**
* Pauses the timer for a determined number of seconds.
* **Remarks**
* Creates another [CountdownTimer](#CountdownTimer) on this game object that has no display. Is recycled when possible.
* **Params**
* seconds, int
* millis, int
* AddTime, void
* **Summary**
* Adds time to this timer.
* **Remarks**
* Will format time if given minutes and/or seconds are greater than or equal to 60, and milliseconds if greater than or equal to 1000.
* **Params**
* time, [NanoTimers.Time](#Time)
* SubTime, void
* **Summary**
* Subtracts time from this timer.
* **Params**
* time, [NanoTimers.Time](#Time)
* SetTime, void
* **Summary**
* Sets this timer to the given input.
* **Remarks**
* Will format time if given minutes and/or seconds are greater than or equal to 60, and milliseconds if greater than or equal to 1000.
* **Params**
* time, [NanoTimers.Time](#Time)
* Reset, void
* **Summary**
* Resets this timer to the original time it was created with.
* **Remarks**
* Does not reset the timer.
* SetShowMillis, void
* **Summary**
* Enables millisecond display.
* **Params**
* isShown, bool
* **Protected**
* UpdateUI, void
* **Summary**
* Updates visual text with this timer’s current time.
* **Remarks**
* Doesn't execute if there is no text reference in m\_timerText to update.
* **Static Methods**
* ConvertToMillis, int
* **Summary**
* Converts a given time into pure milliseconds.
* **Params**
* time, [NanoTimers.Time](#Time)

**CountdownTimer**

* **Summary**
* Timer class that counts from a determined time to 0 and makes an event call to a function upon expiring.
* **Namespace**
* NanoTimers
* **Parent Class**
* [Timer](#Timer)
* **Fields**
* **Private**
* m\_callback, Action
* **Summary**
* Reference to a method that will be invoked when this timer expires.
* **Constructor**
* Unity Default
* **Methods**
* **Public**
* Initialize, void
* **Summary**
* Initializes this time with default values and references.
* **Remarks**
* "Null" may be passed for timerText if no GUI is desired.
* **Params**
* time, [NanoTimers.Time](#Time)
* timerText, Text
* callback, Action
* **Private**
* FixedUpdate, void
* **Summary**
* Determines the number of milliseconds to subtract from this timer every update.

**Stopwatch**

* **Summary**
* Timer class that counts form 0 to infinity.
* **Remarks**
* Does not make event calls.
* **Namespace**
* NanoTimers
* **Parent Class**
* [Timer](#Timer)
* **Methods**
* **Public**
* Initialize, void
* **Summary**
* Initializes the timer with default values and references
* **Params**
* timerText, Text
* **Private**
* FixedUpdate, void
* **Summary**
* Determines the number of milliseconds to add to this timer every update.