**Nano Timers API Documentation**

*Version: 0.1.1d*

Timer Library that includes countdown timers and stopwatches.

**Summary**

* Structures
  + [Time](#tsblwopsin25)
* Enumerables
  + [ETimerState](#w6zm3bx9i8oi)
* Classes
  + [CountdownTimer](#lpp7fty7m4jb)
  + [Stopwatch](#7w3tbfi8i5ym)

**Time**

* Summary
  + Structure that contains minutes and seconds.
* Namespace
  + NanoTimers
* Public Members
  + minutes
  + seconds

**ETimerState**

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

**CountdownTimer**

* Summary
  + Timer class that counts from a determined time to 0 and makes an event call to a function upon expiring.
* Remarks
  + Requires use of AddComponent<T> to make instances on game objects.
* Namespace
  + NanoTimers
* Parent Class
  + UnityEngine.MonoBehaviour
* Fields
  + minutes\_
    - Summary
      * The number of minutes currently on this timer.
  + seconds\_
    - Summary
      * The number of seconds currently on this timer.
  + state\_
    - Summary
      * The current state of this timer.
  + originTimeState\_
    - Summary
      * The initial number of minutes and seconds that were given to this timer upon its creation.
  + callback\_
    - Summary
      * Reference to a method that will be invoked when this timer expires.
  + coroutine\_
    - Summary
      * Reference to the Coroutine that can be stopped and started.
  + timerText\_
    - Summary
      * Reference to the Text object where the timer’s minutes and seconds are displayed.
  + isInitialized\_
    - Summary
      * Determines if this timer has been properly initialized and prevents it from being started with null values.
  + pausedTimer\_
    - Summary
      * Reference to a [CountdownTimer](#lpp7fty7m4jb) that determines how long this timer is paused for.
* Public Methods
  + Initialize
    - Summary
      * Initializes the timer with default values and references.
    - Remarks
      * Timer will not function properly unless this method is called with the correct parameters before StartTimer is called.
    - Params
      * minutes
      * seconds
      * timerText
      * callback
  + StartTimer
    - Summary
      * Starts the timer’s countdown sequence.
    - Remarks
      * Will return immediately if timer has not been properly initialized.
  + Pause
    - Summary
      * Pauses the timer indefinitely until StartTimer is called again.
    - Remarks
      * Will return immediately if timer has not been properly initialized.
  + PauseForSeconds
    - Summary
      * Pauses the timer for a determined amount of time.
    - Remarks
      * Creates another [CountdownTimer](#lpp7fty7m4jb), with no UI, on this game object and is recycled when possible.
      * Will return immediately if timer has not been properly initialized.
    - Params
      * seconds
  + AddTime
    - Summary
      * Adds time to the timer.
    - Remarks
      * Seconds will roll over into minutes if greater 59.
    - Parameters
      * minutes
      * seconds
  + SubTime
    - Summary
      * Subtracts time from the timer.
    - Remarks
      * Minutes roll off when seconds become less than 0.
    - Params
      * minutes
      * seconds
  + SetTime
    - Summary
      * Sets the timer’s minutes and seconds to this specific amount.
    - Remarks
      * Seconds roll over into minutes if greater than 59.
    - Params
      * minutes
      * seconds
  + GetTime
    - Summary
      * Returns the timer’s current time.
    - Return
      * [NanoTimers.Time](#tsblwopsin25)
  + GetTimerState
    - Summary
      * Returns the timer’s current state.
    - Return
      * [ETimerState](#w6zm3bx9i8oi)
  + Reset
    - Summary
      * Resets the timer to the original minutes and seconds it was created with and updates its display reference.
    - Remarks
      * Pauses and does not restart the countdown. StartTimer must be called again.
* Private Methods
  + UpdateUI\_
    - Summary
      * Updates the visual text display with the current time.
    - Remarks
      * Will return immediately if no Text object exists.
  + Tick\_
    - Summary
      * Primary update method that uses an infinite for-loop in a Unity Coroutine to subtract 1 second every second. Calls whatever function is referenced in callback\_ when 0 is reached.
    - Return
      * IEnumerator

**Stopwatch**

* Summary
  + Timer class that counts from 0 to infinity in minutes and seconds.
* Remarks
  + Does not make event calls.
  + Requires use of AddComponent<T> to make instances on game objects.
* Namespace
  + NanoTimers
* Parent Class
  + UnityEngine.MonoBehaviour
* Fields
  + minutes\_
    - Summary
      * The number of minutes currently on this timer.
  + seconds\_
    - Summary
      * The number of seconds currently on this timer.
  + state\_
    - Summary
      * The current state of this timer.
  + coroutine\_
    - Summary
      * Reference to the Coroutine that can be stopped and started.
  + timerText\_
    - Summary
      * Reference to the Text object where the timer’s minutes and seconds are displayed.
  + pausedTimer\_
    - Summary
      * Reference to a [CountdownTimer](#lpp7fty7m4jb) that determines how long this timer is paused for.
* Public Methods
  + Initialize
    - Summary
      * Initializes the timer with default values and references.
    - Remarks
      * Not required to be called for the timer to function properly but necessary if this timer is to update a Text object.
    - Params
      * timerText
  + StartTimer
    - Summary
      * Starts the timer’s stopwatch sequence.
  + Pause
    - Summary
      * Pauses the timer indefinitely until StartTimer is called again.
    - Remarks
      * Will return immediately if timer has not been previously started.
  + PauseForSeconds
    - Summary
      * Pauses the timer for a determined amount of time.
    - Remarks
      * Creates another [CountdownTimer](#lpp7fty7m4jb), with no UI, on this game object.
      * Is recycled when possible.
      * Will return immediately if timer has not been previously started.
    - Params
      * seconds
  + GetTime
    - Summary
      * Returns the timer’s current time.
    - Return
      * [NanoTimers.Time](#tsblwopsin25)
  + GetTimerState
    - Summary
      * Returns the timer’s current state.
    - Return
      * [ETimerState](#w6zm3bx9i8oi)
  + Reset
    - Summary
      * Stops the timer and resets minutes and seconds to 0.
    - Remarks
      * Does not restart. StartTimer must be called again.
* Private Methods
  + UpdateUI\_
    - Summary
      * Updates the visual text display with the current time.
    - Remarks
      * Will return immediately if no Text object exists.
  + Tick\_
    - Summary
      * Primary update method that uses an infinite for-loop in a Unity Coroutine to subtract 1 second every second.
    - Return
      * IEnumerator