▼ Table of contents■ Timers■ Class: Immediate
<pre>immediate.hasRef() immediate.ref() immediate.unref() immediate[Symbol.dispose]()</pre>
Class: Timeouttimeout.close() legacytimeout.hasRef()timeout.ref()
<pre>timeout.refresh() timeout.unref() timeout[Symbol.toPrimitive]() timeout[Symbol.dispose]() Scheduling timers</pre>
 setImmediate(callback[,args]) setInterval(callback[, delay[,args]]) setTimeout(callback[, delay[,args]]) Cancelling timers
 clearImmediate(immediate) clearInterval(timeout) clearTimeout(timeout) Timers Promises API timersPromises.setTimeout([delay[, value[, options]]])
<pre>timersPromises.setImmediate([value[, options]]) timersPromises.setInterval([delay[, value[, options]]]) timersPromises.scheduler.wait(delay[, options]) timersPromises.scheduler.yield()</pre>
Timers #
Source Code: lib/timers.js The timer module exposes a global API for scheduling functions to be called at some future period of time. Because the timer functions are globals, there is no need to call require('node:timers') to use the API. The timer functions within Node.js implement a similar API as the timers API provided by Web Browsers but use a
different internal implementation that is built around the Node.js Event Loop . Class: Immediate # This object is created internally and is returned from setImmediate() . It can be passed to clearImmediate() in order to cancel the scheduled actions.
By default, when an immediate is scheduled, the Node.js event loop will continue running as long as the immediate is active. The Immediate object returned by <pre>setImmediate()</pre> exports both immediate.ref() and immediate.unref() functions that can be used to control this default behavior. ##
Added in: v11.0.0 • Returns: <boolean> If true, the Immediate object will keep the Node.js event loop active. ##</boolean>
Added in: v9.7.0 • Returns: <immediate> a reference to immediate When called, requests that the Node.js event loop not exit so long as the Immediate is active. Calling immediate.ref() multiple times will have no effect.</immediate>
By default, all Immediate objects are "ref'ed", making it normally unnecessary to call immediate.ref() unless immediate.unref() had been called previously. ## Added in: v9.7.0
• Returns: <immediate> a reference to immediate When called, the active Immediate object will not require the Node.js event loop to remain active. If there is no other activity keeping the event loop running, the process may exit before the Immediate object's callback is invoked. Calling immediate.unref() multiple times will have no effect. ##</immediate>
 ► History Cancels the immediate. This is similar to calling clear Immediate(). Class: Timeout
This object is created internally and is returned from setTimeout () and setInterval (). It can be passed to either clearInterval () or clearInterval () in order to cancel the scheduled actions. By default, when a timer is scheduled using either setTimeout () or setInterval (), the Node.js event loop will continue running as long as the timer is active. Each of the Timeout objects returned by these functions export
both timeout.ref() and timeout.unref() functions that can be used to control this default behavior. timeout.close() # Added in: v0.9.1
Stability: 3 - Legacy: Use clearTimeout() instead. Cancels the timeout. ##
Added in: v11.0.0 • Returns: <boolean> If true, the Timeout object will keep the Node.js event loop active.</boolean>
<pre># Added in: v0.9.1 • Returns: <timeout> a reference to timeout When called, requests that the Node.js event loop not exit so long as the Timeout is active. Calling timeout.ref() multiple times will have no effect.</timeout></pre>
By default, all Timeout objects are "ref'ed", making it normally unnecessary to call timeout.ref() unless timeout.unref() had been called previously. timeout.refresh() # Added in: v10.2.0
• Returns: <
<pre>timeout.unref() Added in: v0.9.1 • Returns: <timeout> a reference to timeout When called, the active Timeout object will not require the Node.js event loop to remain active. If there is no other</timeout></pre>
activity keeping the event loop running, the process may exit before the Timeout object's callback is invoked. Calling timeout.unref() multiple times will have no effect. ## Added in: v14.9.0, v12.19.0
• Returns: <integer> a number that can be used to reference this timeout Coerce a Timeout to a primitive. The primitive can be used to clear the Timeout. The primitive can only be used in the same thread where the timeout was created. Therefore, to use it across worker threads it must first be passed to the correct thread. This allows enhanced compatibility with browser setTimeout() and setInterval() implementations.</integer>
timeout[Symbol.dispose]() ► History Cancels the timeout.
Scheduling timers A timer in Node.js is an internal construct that calls a given function after a certain period of time. When a timer's function is called varies depending on which method was used to create the timer and what other work the Node.js event loop is doing.
 setImmediate(callback[,args]) History callback <function> The function to call at the end of this turn of the Node.js Event Loop</function> args <any> Optional arguments to pass when the callback is called.</any>
• Returns: <immediate> for use with clearImmediate() Schedules the "immediate" execution of the callback after I/O events' callbacks. When multiple calls to setImmediate() are made, the callback functions are queued for execution in the order in which they are created. The entire callback queue is processed every event loop iteration. If an immediate timer is queued from inside an executing callback, that timer will not be triggered until the next event loop iteration.</immediate>
If callback is not a function, a TypeError will be thrown. This method has a custom variant for promises that is available using timersPromises.setImmediate() . setInterval(callback[, delay[,args]]) #
 History callback <<u>Function></u> The function to call when the timer elapses. delay <<u>number></u> The number of milliseconds to wait before calling the callback. Default: 1. args <<u>any></u> Optional arguments to pass when the callback is called.
 callback <function> The function to call when the timer elapses.</function> delay <number> The number of milliseconds to wait before calling the callback. Default: 1.</number> args <any> Optional arguments to pass when the callback is called.</any> Returns: <timeout> for use with clearInterval()</timeout> Schedules repeated execution of callback every delay milliseconds. When delay is larger than 2147483647 or less than 1 or NaN, the delay will be set to 1. Non-integer delays are truncated to an integer.
 callback <function> The function to call when the timer elapses.</function> delay <number> The number of milliseconds to wait before calling the callback. Default: 1.</number> args <any> Optional arguments to pass when the callback is called.</any> Returns: <timeout> for use with clearInterval()</timeout> Schedules repeated execution of callback every delay milliseconds. When delay is larger than 2147483647 or less than 1 or NaN, the delay will be set to 1. Non-integer delays are
 callback <function> The function to call when the timer elapses.</function> delay <number> The number of milliseconds to wait before calling the callback. Default: 1.</number> args <any> Optional arguments to pass when the callback is called.</any> Returns: <timeout> for use with clearInterval().</timeout> Schedules repeated execution of callback every delay milliseconds. When delay is larger than 2147483647 or less than 1 or NaN, the delay will be set to 1. Non-integer delays are truncated to an integer. If callback is not a function, a TypeError will be thrown. This method has a custom variant for promises that is available using timersPromises.setInterval(). setTimeout(callback[, delay[,args]])
 callback <function> The function to call when the timer elapses.</function> delay <number> The number of milliseconds to wait before callback. Default: 1.</number> args <any≥ arguments="" callback="" called.<="" is="" li="" optional="" pass="" the="" to="" when=""> Returns: <timeout> for use with clearInterval(.)</timeout> </any≥> Schedules repeated execution of callback every delay milliseconds. When delay is larger than 2147483647 or less than 1 or NaN, the delay will be set to 1. Non-integer delays are truncated to an integer. If callback is not a function, a Imperor will be thrown. This method has a custom variant for promises that is available using timersPromises. set Interval("). set Imperor interval (") ## History callback callback, delay number > The function to call when the timer elapses. delay number > The number of milliseconds to wait before calling the callback. Default: 1. args any> Optional arguments to pass when the callback is called. Returns: Imperor is callback in callback is called. Returns: Imperor or use with clearTimeout(") Schedules execution of a one-time callback after delay milliseconds. Node, is makes no guarantees about the exact timing of when callbacks will fire, nor of their ordering. The callback will be called as close as possible to the time specified.
 callback <function> The function to call when the timer elapses.</function> delay <number> The number of milliseconds to wait before calling the callback. Default: 1.</number> args <any> Optional arguments to pass when the callback is called.</any> Returns: <timeout> for use with clearInterval(").</timeout> Schedules repeated execution of callback every delay milliseconds. When delay is larger than 2147483647 or less than 1 or NaN, the delay will be set to 1. Non-integer delays are truncated to an integer. If callback is not a function, a IxpeError will be thrown. This method has a custom variant for promises that is available using timersPromises.setInterval("). setTimeout(callback[, delay[,args]]) # History callback setUnimersPromises. SetInterval("). Acallback SetUnimethodelay. The function to call when the timer elapses. delay Inimethodelay. Default: 1. args ">any>">any> Optional arguments to pass when the callback is called. Returns: Timeout> for use with clearTimeout() Schedules execution of a one-time callback after delay milliseconds. Node.js makes no guarantees about the exact timing of when callbacks will fire, nor of their ordering. The callback will be called as close as possible to the
 callback ≤Function≥ The function to call when the timer elapses. delay ≤number≥ The number of milliseconds to wait before calling the callback. Default: 1. args ≤any≥ Optional arguments to pass when the callback is called. Returns: <i>Timeout≥ for use with clear Interval().</i> Schedules repeated execution of callback every delay milliseconds. When delay is larger than 21474B3647 or less than 1 or NaN, the delay will be set to 1. Non-integer delays are truncated to an integer. If callback is not a function, a TypeError. will be thrown. This method has a custom variant for promises that is available using timersPromises. setInterval(). setTimeout(callback[, delay[,args]]) History callback ≤Function≥ The function to call when the timer elapses. delay ≤number≥ The number of milliseconds to wait before calling the callback. Default: 1. args ≤any≥ Optional arguments to pass when the callback is called. Returns: <i>Timeout≥ for use with clear Timeout()</i> Schedules execution of a one-time callback after delay milliseconds. The callback will likely not be invoked in precisely delay milliseconds. Node is makes no guarantees about the exact timing of when callbacks will fire, nor of their ordering. The callback will be called as close as possible to the time specified. When delay is larger than 2147483647 or less than 1 or NaN, the delay will be set to 1. Non-integer delays are truncated to an integer. If callback is not a function, a TypeError will be thrown. This method has a custom variant for promises that is available using timersPromises.setTimeout(). Cancelling timers The setImmediate(), setInterval(), and setTimeout() methods each return objects that represent the scheduled timers. These can be used to cancel the timer and p
 callback <u>Function</u> The function to call when the timer elapses. delay <u>snumbers</u> The number of milliseconds to wait before calling the callback. Default: 1. args <u>sany</u> Optional arguments to pass when the callback is called. Returns: <u>stimeoute</u> for use with <u>clearInterval()</u>. Schedules repeated execution of callback every delay milliseconds. When delay is larger than 2147483647 or less than 1 or NaN, the delay will be set to 1. Non-integer delays are truncated to an integer. If callback is not a function, a <u>TypeError</u> will be thrown. This method has a custom variant for promises that is available using <u>timersPromises.setInterval()</u>. setTimeout(callback[, delay[,args]]) # History callback <u>Functions</u> The function to call when the timer elapses. delay <u>snumbers</u> The number of milliseconds to wait before calling the callback. Default: 1. args <u>sany</u> Optional arguments to pass when the callback is called. Returns: <u>stimeoute</u> for use with <u>clearTimeout()</u> Schedules execution of a one-time callback after delay milliseconds. The callback will likely not be invoked in precisely delay milliseconds. Node is makes no guarantees about the exact timing of when callbacks will fire, nor of their ordering. The callback will be called as close as possible to the time specified. When delay is larger than 2147483647 or less than 1 or NaN, the delay will be set to 1. Non-integer delays are truncated to an integer. If callback is not a function, a <u>TypeError</u> will be thrown. This method has a custom variant for promises that is available using <u>timersPromises.setTimeout()</u>. Cancelling timers #
 callback <u> delay snumber</u> The number of milliseconds to wait before calling the callback. Default: 1. args <u> angs</u> entry pointed arguments to pass when the callback is called. Returns: <u> Timeout</u> for use with <u>clearInterval()</u>. Schedules repeated execution of callback every delay milliseconds. When delay is larger than 2147483647 or less than 1 or NaN, the delay will be set to 1. Non-integer delays are truncated to an integer. If callback is not a function, a TypeError will be thrown. This method has a custom variant for promises that is available using <u>timersPromises_setInterval()</u>. setTimeout{callback[, delay[,args]]} ## History callback sunction The function to call when the timer elapses. delay <u> delay snumber</u> The number of milliseconds to wait before calling the callback. Default: 1. args_asany Optional arguments to pass when the callback is called. Returns: <u> Timeout</u> for use with clear <u>Timeout</u>() Schedules execution of a one-time callback after delay milliseconds. The callback will likely not be invoked in precisely delay milliseconds. Node is makes no guarantees about the exact timing of when callbacks will fire, nor of their ordering. The callback will be called as close as possible to the time specified. When delay is larger than 2147483647 or less than 1 or NaN, the delay will be set to 1. Non-integer delays are truncated to an integer. If callback is not a function, a <u>TypeError</u> will be thrown. This method has a custom variant for promises that is available using <u>timersPromises setTimeout()</u>. Cancelling timers The setTimeout(), setTiterval(), and <u>setTimeout()</u> methods each return objects that represent the scheduled timers. These can be used to cancel
. callback <pre>function> The function to call when the timer elapses delay <number> The number of milliseconds to wait before calling the callback. Default: 1</number></pre>
. callback of more time. The function to call when the timer elapses. delay consider: The number of milliseconds to wait before calling the call thank. Default: 1.
e callback of Functions The function to call when the timer elapses. darlay on uniber? The number of milliseconds to with before calling the callback. Default: 1. args carrys Optional arguments to pass when the callback is called. Returns: Timeration for use with return to number of callback body leave milliseconds. When delay is larger than 2147483647 or less than 1 or NAN, the delay will be set to 1. Non-integer delays are translated to an integer. If rail thank is not a function, a TypeError will be thrown. This method has a custom variant for promises that is available using fine railback set interval(). sectTimeout(callback, [delay[,args]]) ## I likewy • callback Securitions The function to call when the timer elapses. • delay symmetry: The number of milliseconds to wait before calling the callback. Default: 1. •args carrys Optional arguments to pass when the callback is called. • Returns: Climanute for use with clear Timeout() Schedules execution of a one-time callback after delay milliseconds. The callback will filely not be invoked in precisely delay milliseconds. Node is makes no guarantees about the exact timing of when callback will file nor of their ordering. The callback will be called as close as possible to the cime specified. When dalay biarger than 2147488047 or less than 1 or NaN, the delay will be set to 1, Non-integer delays are truncated to an integer. If callback is not a function, a TypeError will be thrown. This method has a custom variant for promises that is available using timers Promises, set Timeout (). Cancelling timers The aptrimediate(), action on all (), and set Timeout (), methods each return objects that represent the scheduled timer. These can be used to cancel the timer and prevent is from triggering. contact (or extinued as a constant) contact (or extinued as a constant) contact (setTimeout; setTimeoutFormics) = require('node: timers/promises');
* call back described into The function to call when the timer elapses. * delay *saubers* The number of milliseconds to wais before calling the callback. Default: 1. *
callback dispertions. The function to call when the timer elapses. delay shallback Thenumber of millisconds to walchefore calling the callback. Default 1arays dange Collonal arguments to pass when the callback is called. Returns of Timeseuth of ruse with clear interval () Schedules repeated execution of callback every desay millisconds. When delay is birger than 214748847 or less than 1 or Man, the delay will be set to 1. Non-integer tidays are truncated to an integer. If callback is not a function a TypeEncopy will be thrown. This method has a costom workant for promises that is available using *Immostranticos, and Interval (). set Timeout(callback[, delay[,args]]) # Hedary callback of accidence to the function to call when the timer elapses. delay similarity the number of millises mode to wall before calling the call back. Default: 1aras cange Optional arguments to pass when the callback is called. Returns of immost one-time call back after delay milliseconds. The callback will likely not be invented in processly delay milliseconds. The callback will likely not be invented in processly delay milliseconds. The callback will likely not be invented in processly delay milliseconds. The callback will likely not be invented in processly delay milliseconds. Nodelis makes no guarantees about the exact timing of when callbacks will fire, nor of their ordering. The callback will be set to 1. Non-integer delays are truncated to an integer. If callback is not a function, a TypeEncor will be thrown. This method has a custom variant for promises that is available using TimersPromises.setTimeout(). Cancelling timers The setTimedate(), setTimeout(), and setTimeout() methods each return objects that represent the devaluable is not a function to legit immostate and accorded to timer When canceld, the returned Promises will be rejected with an "Abort Encord") consider of the immostate and accorded () accorded (encorded to the immostate and accorded); contile setTimeout () setTimeout Promises) require
e caliback sharecisor. The function to all between the filmer elabases. • delay sharecisor. The number of millibeanods to walk before tailing the caliback. Default: 1. • args samps. Optional arguments to pass when the caliback hospital. Shouldes repeated occurring of an interior record () Shouldes repeated occurring of an interior record () Shouldes repeated occurring of a laback recey as my milliperrecis. The interior of an interior occurring of the chrown. This method has a caseom variant for promises that is well believed to 1. Non-integer delays are transcaled to an interior of promises that is well believed to 1. Non-integer delays are transcaled by slangar than 2127480647 or less than 1 or flat, the solay will be set to 1. Non-integer delays are transcaled to 1. Non-integer del
. as these situations. The functions call who the interestings. • Bully gooders. The makes of millisconds own between the set black. • Bully gooders. The makes of millisconds own between the set black. • Bully gooders composed control of ast back completely politicated. • Bully set interest to resemble place the set of the s
e. act least structures. The functions call when the timer object. debty structures. The median milliactures is well the callback is called. Returnes "The capture for previous in contributions" () Standards resoluted wearboard and activate, error easy milliactures. When debty is larger the 2247400447 or less than 10 met. the debty will be set to 1. Non-integer debty are the 2247400447 or less than 10 met. the debty will be set to 1. Non-integer debty are the callback is made if crotions a passagrae, will be thrown. This method has a costom waithful for provises the its available unline interestings. Examines the callback is made if crotions a passagrae, will be thrown. This method has a costom waithful for an interesting of callback. Debtaffin 1. - Larger standard Optional arguments to assessment the callback is called. Returnes "Interest by contribution of milliocronists waith before calling the callback. Debtaffin 1. - Larger standard Optional arguments to assessment the callback is called. Returnes a standard prove with patient interesting and any influenceds. The callback will file any old arising a file and interesting of callback will be called as done as possible to deline application of a one time as it back, often on any milliocronist. The callback will file any old arising and interesting and the callback is called as done as possible to deline application of a one time as it back, often on any milliocronist. The callback is interest that a 224404464 or files than 1 or file. It debty will be called as done as possible to deline application of a one difference of a callback will be called as done as possible to deline as interesting and intere
color commons. The function could of the linear earses. color commons. The remains of millisecurits cover, between ingite conclused. Returns _injuriescent to use with _promiser seal. Returns _injuriescent to use with _promiser seal. Returns _injuriescent to use with _promiser seal. Returns _injuriescent to _promiser seal. Returns _promiser to _promise seal of the chief with the feeling will be excited. Returns _promiser to _promise seal of the chief will be excited to _promise seal. Returns _promiser to record the chief of _promise seal of _promise seal. Returns _promiser to record feeling _promiser seal of _promise seal. Returns _promiser _promiser seal of _promise seal. Returns _promiser _promiser _promiser seal. Returns _promiser _pr
- catalogue and the content of milliocontrib each tree calling the call back Debutals 1 - causes again, content the content of milliocontrib each batter calling the call back Debutals 1 - Partition of administration of an interview of the partition of a content of an interview of the partition of a content of an interview of the partition of a content of an interview of the partition of an interview of the calling the partition of an interview of the partition of the content of the content of the calling of the partition of the calling of the callin
- entitles counted less the the backets and electronic terror places delay contents in the cause of electronic and electronic terror places Section discontinuous of electronic and
contracts of the contract in the desire contract of the contribution of the contribution. Default 1. contract on growth and present of the contribution of the contribution. Name about proportion placetable and observable and assert a contract of the contribution of contracts to easy with a contract of the contribution of contracts. The makes in the contract of the contribution of
cease parameters because the control and electrical and electron control and electron contro
is not have a passaged for the control of the control. Is discovered to express the control of the control. Is discovered to express the control of the co
the companies of the common with contract and inclined and a cities and access to each access and the common of th
- Saltack industable Transition for the mode for collection of the certification of the cert
The contract of mathematical intermediate of the contract of mathematical intermediate and the contract of mathematical intermediate of mathematical intermediate in the contract of mathematical intermediate of math
services and action to be inches model as where the region is not action to be able in a construction of the construction of t
services and action to be between each assemble that appears in a sixty against the second action to be a sixty and action actions and actions of the second actions of the second actions actions and actions
electron dimining between the attention of the state and color of the state of the

console.log(Date.now()); })(); timersPromises.scheduler.wait(delay[, options]) # Added in: v17.3.0, v16.14.0
 Stability: 1 - Experimental delay number of milliseconds to wait before resolving the promise. options object
 ref <boolean> Set to false to indicate that the scheduled Timeout should not require the Node.js event loop to remain active. Default: true.</boolean> signal <abortsignal> An optional AbortSignal that can be used to cancel waiting.</abortsignal> Returns: <promise></promise> An experimental API defined by the Scheduling APIs draft specification being developed as a standard Web
Platform API. Calling timersPromises.scheduler.wait(delay, options) is equivalent to calling timersPromises.setTimeout(delay, undefined, options).
<pre>import { scheduler } from 'node:timers/promises'; await scheduler.wait(1000); // Wait one second before continuing timersPromises.scheduler.yield() #</pre>
Added in: v17.3.0, v16.14.0 Stability: 1 - Experimental • Returns: <promise></promise>
An experimental API defined by the <u>Scheduling APIs</u> draft specification being developed as a standard Web Platform API. Calling timersPromises.scheduler.yield() is equivalent to calling timersPromises.setImmediate() with no arguments.