process

> *Extensions to process object.*

Process: Main, Renderer

Electron's `process` object is extended from the Node.js `process` object. It adds the following events, properties, and methods:

Sandbox

In sandboxed renderers the `process` object contains only a subset of the APIs:

- `crash()`
- `hang()`
- `getCreationTime()`
- `getHeapStatistics()`
- `getBlinkMemoryInfo()`
- `getProcessMemoryInfo()`
- `getSystemMemoryInfo()`
- `getSystemVersion()`
- `getCPUUsage()`
- `getIOCounters()`
- `uptime()`
- `argv`
- `execPath`
- `env`
- `pid`
- `arch`
- `platform`
- `sandboxed`
- `contextIsolated`
- `type`
- `version`
- `versions`
- `mas`
- `windowsStore`
- `contextId`

Events

Event: 'loaded'

Emitted when Electron has loaded its internal initialization script and is beginning to load the web page or the main script.

Properties

`process.defaultApp` *Readonly*

A `boolean`. When app is started by being passed as parameter to the default app, this property is `true` in the main process, otherwise it is `undefined`.

`process.isMainFrame` *Readonly*

A `boolean`, `true` when the current renderer context is the "main" renderer frame. If you want the ID of the current frame you should use `webFrame.routingId`.

`process.mas` *Readonly*

A `boolean`. For Mac App Store build, this property is `true`, for other builds it is `undefined`.

`process.noAsar`

A `boolean` that controls ASAR support inside your application. Setting this to `true` will disable the support for `asar` archives in Node's built-in modules.

`process.noDeprecation`

A `boolean` that controls whether or not deprecation warnings are printed to `stderr`. Setting this to `true` will silence deprecation warnings. This property is used instead of the `--no-deprecation` command line flag.

`process.resourcesPath` *Readonly*

A `string` representing the path to the resources directory.

`process.sandboxed` *Readonly*

A `boolean`. When the renderer process is sandboxed, this property is `true`, otherwise it is `undefined`.

`process.contextIsolated` *Readonly*

A `boolean` that indicates whether the current renderer context has `contextIsolation` enabled. It is `undefined` in the main process.

`process.throwDeprecation`

A `boolean` that controls whether or not deprecation warnings will be thrown as exceptions. Setting this to `true` will throw errors for deprecations. This property is used instead of the `--throw-deprecation` command line flag.

`process.traceDeprecation`

A `boolean` that controls whether or not deprecations printed to `stderr` include their stack trace. Setting this to `true` will print stack traces for deprecations. This property is instead of the `--trace-deprecation` command line flag.

`process.traceProcessWarnings`

A `boolean` that controls whether or not process warnings printed to `stderr` include their stack trace. Setting this to `true` will print stack traces for process warnings (including deprecations). This property is instead of the `--trace-warnings` command line flag.

`process.type` *Readonly*

A `string` representing the current process's type, can be:

- `browser` - The main process
- `renderer` - A renderer process
- `worker` - In a web worker

`process.versions.chrome` *Readonly*

A `string` representing Chrome's version string.

`process.versions.electron` *Readonly*

A `string` representing Electron's version string.

`process.windowsStore` *Readonly*

A `boolean`. If the app is running as a Windows Store app (appx), this property is `true`, for otherwise it is `undefined`.

`process.contextId` *Readonly*

A `string` (optional) representing a globally unique ID of the current JavaScript context. Each frame has its own JavaScript context. When contextIsolation is enabled, the isolated world also has a separate JavaScript context. This property is only available in the renderer process.

Methods

The `process` object has the following methods:

`process.crash()`

Causes the main thread of the current process crash.

`process.getCreationTime()`

Returns `number | null` - The number of milliseconds since epoch, or `null` if the information is unavailable

Indicates the creation time of the application. The time is represented as number of milliseconds since epoch. It returns null if it is unable to get the process creation time.

`process.getCPUUsage()`

Returns `CPUUsage`

`process.getIOCounters()` *Windows Linux*

Returns `IOCounters`

`process.getHeapStatistics()`

Returns `Object`:

- `totalHeapSize` Integer
- `totalHeapSizeExecutable` Integer

- `totalPhysicalSize` Integer
- `totalAvailableSize` Integer
- `usedHeapSize` Integer
- `heapSizeLimit` Integer
- `mallocedMemory` Integer
- `peakMallocedMemory` Integer
- `doesZapGarbage` boolean

Returns an object with V8 heap statistics. Note that all statistics are reported in Kilobytes.

`process.getBlinkMemoryInfo()`

Returns `Object`:

- `allocated` Integer - Size of all allocated objects in Kilobytes.
- `total` Integer - Total allocated space in Kilobytes.

Returns an object with Blink memory information. It can be useful for debugging rendering / DOM related memory issues. Note that all values are reported in Kilobytes.

`process.getProcessMemoryInfo()`

Returns `Promise<ProcessMemoryInfo>` - Resolves with a ProcessMemoryInfo

Returns an object giving memory usage statistics about the current process. Note that all statistics are reported in Kilobytes. This api should be called after app ready.

Chromium does not provide `residentSet` value for macOS. This is because macOS performs in-memory compression of pages that haven't been recently used. As a result the resident set size value is not what one would expect. `private` memory is more representative of the actual pre-compression memory usage of the process on macOS.

`process.getSystemMemoryInfo()`

Returns `Object`:

- `total` Integer - The total amount of physical memory in Kilobytes available to the system.
- `free` Integer - The total amount of memory not being used by applications or disk cache.
- `swapTotal` Integer *Windows Linux* - The total amount of swap memory in Kilobytes available to the system.
- `swapFree` Integer *Windows Linux* - The free amount of swap memory in Kilobytes available to the system.

Returns an object giving memory usage statistics about the entire system. Note that all statistics are reported in Kilobytes.

`process.getSystemVersion()`

Returns `string` - The version of the host operating system.

Example:

```
const version = process.getSystemVersion()
console.log(version)
// On macOS -> '10.13.6'
```

```
// On Windows -> '10.0.17763'
// On Linux -> '4.15.0-45-generic'
```

Note: It returns the actual operating system version instead of kernel version on macOS unlike `os.release()`.

`process.takeHeapSnapshot(filePath)`

- `filePath` string - Path to the output file.

Returns `boolean` - Indicates whether the snapshot has been created successfully.

Takes a V8 heap snapshot and saves it to `filePath`.

`process.hang()`

Causes the main thread of the current process hang.

`process.setFdLimit(maxDescriptors)` *macOS Linux*

- `maxDescriptors` Integer

Sets the file descriptor soft limit to `maxDescriptors` or the OS hard limit, whichever is lower for the current process.