Breaking Changes

Breaking changes will be documented here, and deprecation warnings added to JS code where possible, at least one major version before the change is made.

Types of Breaking Changes

This document uses the following convention to categorize breaking changes:

- **API Changed:** An API was changed in such a way that code that has not been updated is guaranteed to throw an exception.
- **Behavior Changed:** The behavior of Electron has changed, but not in such a way that an exception will necessarily be thrown.
- **Default Changed:** Code depending on the old default may break, not necessarily throwing an exception. The old behavior can be restored by explicitly specifying the value.
- **Deprecated:** An API was marked as deprecated. The API will continue to function, but will emit a deprecation warning, and will be removed in a future release.
- **Removed:** An API or feature was removed, and is no longer supported by Electron.

Planned Breaking API Changes (20.0)

Default Changed: renderers without `nodeIntegration: true` are sandboxed by default

Previously, renderers that specified a preload script defaulted to being unsandboxed. This meant that by default, preload scripts had access to Node.js. In Electron 20, this default has changed. Beginning in Electron 20, renderers will be sandboxed by default, unless `nodeIntegration: true` or `sandbox: false` is specified.

If your preload scripts do not depend on Node, no action is needed. If your preload scripts *do* depend on Node, either refactor them to remove Node usage from the renderer, or explicitly specify `sandbox: false` for the relevant renderers.

Removed: `skipTaskbar` on Linux

On X11, `skipTaskbar` sends a `_NET_WM_STATE_SKIP_TASKBAR` message to the X11 window manager. There is not a direct equivalent for Wayland, and the known workarounds have unacceptable tradeoffs (e.g. Window.is_skip_taskbar in GNOME requires unsafe mode), so Electron is unable to support this feature on Linux.

Planned Breaking API Changes (19.0)

None

Planned Breaking API Changes (18.0)

Removed: `nativeWindowOpen`

Prior to Electron 15, `window.open` was by default shimmed to use `BrowserWindowProxy`. This meant that `window.open('about:blank')` did not work to open synchronously scriptable child windows, among other incompatibilities. Since Electron 15, `nativeWindowOpen` has been enabled by default.

See the documentation for window.open in Electron for more details.

Planned Breaking API Changes (17.0)

Removed: `desktopCapturer.getSources` in the renderer

The `desktopCapturer.getSources` API is now only available in the main process. This has been changed in order to improve the default security of Electron apps.

If you need this functionality, it can be replaced as follows:

```
// Main process
const { ipcMain, desktopCapturer } = require('electron')

ipcMain.handle(
  'DESKTOP_CAPTURER_GET_SOURCES',
  (event, opts) => desktopCapturer.getSources(opts)
)
```

```
// Renderer process
const { ipcRenderer } = require('electron')

const desktopCapturer = {
  getSources: (opts) => ipcRenderer.invoke('DESKTOP_CAPTURER_GET_SOURCES', opts)
}
```

However, you should consider further restricting the information returned to the renderer; for instance, displaying a source selector to the user and only returning the selected source.

Deprecated: `nativeWindowOpen`

Prior to Electron 15, `window.open` was by default shimmed to use `BrowserWindowProxy`. This meant that `window.open('about:blank')` did not work to open synchronously scriptable child windows, among other incompatibilities. Since Electron 15, `nativeWindowOpen` has been enabled by default.

See the documentation for window.open in Electron for more details.

Planned Breaking API Changes (16.0)

Behavior Changed: `crashReporter` implementation switched to Crashpad on Linux

The underlying implementation of the `crashReporter` API on Linux has changed from Breakpad to Crashpad, bringing it in line with Windows and Mac. As a result of this, child processes are now automatically monitored, and calling `process.crashReporter.start` in Node child processes is no longer needed (and is not advisable, as it will start a second instance of the Crashpad reporter).

There are also some subtle changes to how annotations will be reported on Linux, including that long values will no longer be split between annotations appended with `__1`, `__2` and so on, and instead will be truncated at the (new, longer) annotation value limit.

Deprecated: `desktopCapturer.getSources` in the renderer

Usage of the `desktopCapturer.getSources` API in the renderer has been deprecated and will be removed. This change improves the default security of Electron apps.

See here for details on how to replace this API in your app.

Planned Breaking API Changes (15.0)

Default Changed: `nativeWindowOpen` defaults to `true`

Prior to Electron 15, `window.open` was by default shimmed to use `BrowserWindowProxy`. This meant that `window.open('about:blank')` did not work to open synchronously scriptable child windows, among other incompatibilities. `nativeWindowOpen` is no longer experimental, and is now the default.

See the documentation for window.open in Electron for more details.

Planned Breaking API Changes (14.0)

Removed: `remote` module

The `remote` module was deprecated in Electron 12, and will be removed in Electron 14. It is replaced by the `@electron/remote` module.

```
// Deprecated in Electron 12:
const { BrowserWindow } = require('electron').remote
```

```
// Replace with:
const { BrowserWindow } = require('@electron/remote')

// In the main process:
require('@electron/remote/main').initialize()
```

Removed: `app.allowRendererProcessReuse`

The `app.allowRendererProcessReuse` property will be removed as part of our plan to more closely align with Chromium's process model for security, performance and maintainability.

For more detailed information see #18397.

Removed: Browser Window Affinity

The `affinity` option when constructing a new `BrowserWindow` will be removed as part of our plan to more closely align with Chromium's process model for security, performance and maintainability.

For more detailed information see #18397.

API Changed: `window.open()`

The optional parameter `frameName` will no longer set the title of the window. This now follows the specification described by the native documentation under the corresponding parameter `windowName`.

If you were using this parameter to set the title of a window, you can instead use win.setTitle(title).

Removed: `worldSafeExecuteJavaScript`

In Electron 14, `worldSafeExecuteJavaScript` will be removed. There is no alternative, please ensure your code works with this property enabled. It has been enabled by default since Electron 12.

You will be affected by this change if you use either `webFrame.executeJavaScript` or `webFrame.executeJavaScriptInIsolatedWorld`. You will need to ensure that values returned by either of those methods are supported by the Context Bridge API as these methods use the same value passing semantics.

Removed: BrowserWindowConstructorOptions inheriting from parent windows

Prior to Electron 14, windows opened with `window.open` would inherit BrowserWindow constructor options such as `transparent` and `resizable` from their parent window. Beginning with Electron 14, this behavior is removed, and windows will not inherit any BrowserWindow constructor options from their parents.

Instead, explicitly set options for the new window with `setWindowOpenHandler`:

```
webContents.setWindowOpenHandler((details) => {
  return {
    action: 'allow',
    overrideBrowserWindowOptions: {
      // ...
    }
  }
})
```

Removed: `additionalFeatures`

The deprecated `additionalFeatures` property in the `new-window` and `did-create-window` events of WebContents has been removed. Since `new-window` uses positional arguments, the argument is still present, but will always be the empty array `[]`. (Though note, the `new-window` event itself is deprecated, and is replaced by `setWindowOpenHandler`.) Bare keys in window features will now present as keys with the value `true` in the options object.

```
// Removed in Electron 14
// Triggered by window.open('...', '', 'my-key')
webContents.on('did-create-window', (window, details) => {
  if (details.additionalFeatures.includes('my-key')) {
    // ...
  }
})

// Replace with
webContents.on('did-create-window', (window, details) => {
  if (details.options['my-key']) {
    // ...
  }
})
```

Planned Breaking API Changes (13.0)

API Changed: `session.setPermissionCheckHandler(handler)`

The `handler` methods first parameter was previously always a `webContents`, it can now sometimes be `null`. You should use the `requestingOrigin`, `embeddingOrigin` and `securityOrigin` properties to respond to the permission check correctly. As the `webContents` can be `null` it can no longer be relied on.

```
// Old code
session.setPermissionCheckHandler((webContents, permission) => {
  if (webContents.getURL().startsWith('https://google.com/') && permission ===
'notification') {
    return true
  }
  return false
})

// Replace with
session.setPermissionCheckHandler((webContents, permission, requestingOrigin) => {
  if (new URL(requestingOrigin).hostname === 'google.com' && permission ===
'notification') {
    return true
  }
  return false
})
```

Removed: `shell.moveItemToTrash()`

The deprecated synchronous `shell.moveItemToTrash()` API has been removed. Use the asynchronous `shell.trashItem()` instead.

```
// Removed in Electron 13
shell.moveItemToTrash(path)
// Replace with
shell.trashItem(path).then(/* ... */)
```

Removed: `BrowserWindow` extension APIs

The deprecated extension APIs have been removed:

- `BrowserWindow.addExtension(path)`
- `BrowserWindow.addDevToolsExtension(path)`
- `BrowserWindow.removeExtension(name)`
- `BrowserWindow.removeDevToolsExtension(name)`
- `BrowserWindow.getExtensions()`
- `BrowserWindow.getDevToolsExtensions()`

Use the session APIs instead:

- `ses.loadExtension(path)`
- `ses.removeExtension(extension_id)`
- `ses.getAllExtensions()`

```
// Removed in Electron 13
BrowserWindow.addExtension(path)
BrowserWindow.addDevToolsExtension(path)
// Replace with
session.defaultSession.loadExtension(path)
```

```
// Removed in Electron 13
BrowserWindow.removeExtension(name)
BrowserWindow.removeDevToolsExtension(name)
// Replace with
session.defaultSession.removeExtension(extension_id)
```

```
// Removed in Electron 13
BrowserWindow.getExtensions()
BrowserWindow.getDevToolsExtensions()
// Replace with
session.defaultSession.getAllExtensions()
```

Removed: methods in `systemPreferences`

The following `systemPreferences` methods have been deprecated:

- `systemPreferences.isDarkMode()`
- `systemPreferences.isInvertedColorScheme()`
- `systemPreferences.isHighContrastColorScheme()`

Use the following `nativeTheme` properties instead:

- `nativeTheme.shouldUseDarkColors`
- `nativeTheme.shouldUseInvertedColorScheme`
- `nativeTheme.shouldUseHighContrastColors`

```
// Removed in Electron 13
systemPreferences.isDarkMode()
// Replace with
nativeTheme.shouldUseDarkColors

// Removed in Electron 13
systemPreferences.isInvertedColorScheme()
// Replace with
nativeTheme.shouldUseInvertedColorScheme

// Removed in Electron 13
systemPreferences.isHighContrastColorScheme()
// Replace with
nativeTheme.shouldUseHighContrastColors
```

Deprecated: WebContents `new-window` event

The `new-window` event of WebContents has been deprecated. It is replaced by `webContents.setWindowOpenHandler()`.

```
// Deprecated in Electron 13
webContents.on('new-window', (event) => {
  event.preventDefault()
})

// Replace with
webContents.setWindowOpenHandler((details) => {
  return { action: 'deny' }
})
```

Planned Breaking API Changes (12.0)

Removed: Pepper Flash support

Chromium has removed support for Flash, and so we must follow suit. See Chromium's Flash Roadmap for more details.

Default Changed: `worldSafeExecuteJavaScript` defaults to `true`

In Electron 12, `worldSafeExecuteJavaScript` will be enabled by default. To restore the previous behavior, `worldSafeExecuteJavaScript: false` must be specified in WebPreferences. Please note that setting this option to `false` is **insecure**.

This option will be removed in Electron 14 so please migrate your code to support the default value.

Default Changed: `contextIsolation` defaults to `true`

In Electron 12, `contextIsolation` will be enabled by default. To restore the previous behavior, `contextIsolation: false` must be specified in WebPreferences.

We recommend having contextIsolation enabled for the security of your application.

Another implication is that `require()` cannot be used in the renderer process unless `nodeIntegration` is `true` and `contextIsolation` is `false`.

For more details see: https://github.com/electron/electron/issues/23506

Removed: `crashReporter.getCrashesDirectory()`

The `crashReporter.getCrashesDirectory` method has been removed. Usage should be replaced by `app.getPath('crashDumps')`.

```
// Removed in Electron 12
crashReporter.getCrashesDirectory()
// Replace with
app.getPath('crashDumps')
```

Removed: `crashReporter` methods in the renderer process

The following `crashReporter` methods are no longer available in the renderer process:

- `crashReporter.start`
- `crashReporter.getLastCrashReport`
- `crashReporter.getUploadedReports`
- `crashReporter.getUploadToServer`
- `crashReporter.setUploadToServer`
- `crashReporter.getCrashesDirectory`

They should be called only from the main process.

See #23265 for more details.

Default Changed: `crashReporter.start({ compress: true })`

The default value of the `compress` option to `crashReporter.start` has changed from `false` to `true`. This means that crash dumps will be uploaded to the crash ingestion server with the `Content-Encoding: gzip` header, and the body will be compressed.

If your crash ingestion server does not support compressed payloads, you can turn off compression by specifying `{ compress: false }` in the crash reporter options.

Deprecated: `remote` module

The `remote` module is deprecated in Electron 12, and will be removed in Electron 14. It is replaced by the `@electron/remote` module.

```
// Deprecated in Electron 12:
const { BrowserWindow } = require('electron').remote
```

```
// Replace with:
const { BrowserWindow } = require('@electron/remote')

// In the main process:
require('@electron/remote/main').initialize()
```

Deprecated: `shell.moveItemToTrash()`

The synchronous `shell.moveItemToTrash()` has been replaced by the new, asynchronous `shell.trashItem()`.

```
// Deprecated in Electron 12
shell.moveItemToTrash(path)
// Replace with
shell.trashItem(path).then(/* ... */)
```

Planned Breaking API Changes (11.0)

Removed: `BrowserView.{destroy, fromId, fromWebContents, getAllViews}` and `id` property of `BrowserView`

The experimental APIs `BrowserView.{destroy, fromId, fromWebContents, getAllViews}` have now been removed. Additionally, the `id` property of `BrowserView` has also been removed.

For more detailed information, see #23578.

Planned Breaking API Changes (10.0)

Deprecated: `companyName` argument to `crashReporter.start()`

The `companyName` argument to `crashReporter.start()`, which was previously required, is now optional, and further, is deprecated. To get the same behavior in a non-deprecated way, you can pass a `companyName` value in `globalExtra`.

```
// Deprecated in Electron 10
crashReporter.start({ companyName: 'Umbrella Corporation' })
// Replace with
crashReporter.start({ globalExtra: { _companyName: 'Umbrella Corporation' } })
```

Deprecated: `crashReporter.getCrashesDirectory()`

The `crashReporter.getCrashesDirectory` method has been deprecated. Usage should be replaced by `app.getPath('crashDumps')`.

```
// Deprecated in Electron 10
crashReporter.getCrashesDirectory()
// Replace with
app.getPath('crashDumps')
```

Deprecated: `crashReporter` methods in the renderer process

Calling the following `crashReporter` methods from the renderer process is deprecated:

- `crashReporter.start`
- `crashReporter.getLastCrashReport`
- `crashReporter.getUploadedReports`
- `crashReporter.getUploadToServer`
- `crashReporter.setUploadToServer`
- `crashReporter.getCrashesDirectory`

The only non-deprecated methods remaining in the `crashReporter` module in the renderer are `addExtraParameter`, `removeExtraParameter` and `getParameters`.

All above methods remain non-deprecated when called from the main process.

See #23265 for more details.

Deprecated: `crashReporter.start({ compress: false })`

Setting `{ compress: false }` in `crashReporter.start` is deprecated. Nearly all crash ingestion servers support gzip compression. This option will be removed in a future version of Electron.

Default Changed: `enableRemoteModule` defaults to `false`

In Electron 9, using the remote module without explicitly enabling it via the `enableRemoteModule` WebPreferences option began emitting a warning. In Electron 10, the remote module is now disabled by default. To use the remote module, `enableRemoteModule: true` must be specified in WebPreferences:

```
const w = new BrowserWindow({
  webPreferences: {
    enableRemoteModule: true
  }
})
```

We recommend moving away from the remote module.

`protocol.unregisterProtocol`

`protocol.uninterceptProtocol`

The APIs are now synchronous and the optional callback is no longer needed.

```
// Deprecated
protocol.unregisterProtocol(scheme, () => { /* ... */ })
// Replace with
protocol.unregisterProtocol(scheme)
```

`protocol.registerFileProtocol`

`protocol.registerBufferProtocol`

`protocol.registerStringProtocol`

`protocol.registerHttpProtocol`

`protocol.registerStreamProtocol`

`protocol.interceptFileProtocol`

`protocol.interceptStringProtocol`

`protocol.interceptBufferProtocol`

`protocol.interceptHttpProtocol`

`protocol.interceptStreamProtocol`

The APIs are now synchronous and the optional callback is no longer needed.

```
// Deprecated
protocol.registerFileProtocol(scheme, handler, () => { /* ... */ })
```

```
// Replace with
protocol.registerFileProtocol(scheme, handler)
```

The registered or intercepted protocol does not have effect on current page until navigation happens.

`protocol.isProtocolHandled`

This API is deprecated and users should use `protocol.isProtocolRegistered` and `protocol.isProtocolIntercepted` instead.

```
// Deprecated
protocol.isProtocolHandled(scheme).then(() => { /* ... */ })
// Replace with
const isRegistered = protocol.isProtocolRegistered(scheme)
const isIntercepted = protocol.isProtocolIntercepted(scheme)
```

Planned Breaking API Changes (9.0)

Default Changed: Loading non-context-aware native modules in the renderer process is disabled by default

As of Electron 9 we do not allow loading of non-context-aware native modules in the renderer process. This is to improve security, performance and maintainability of Electron as a project.

If this impacts you, you can temporarily set `app.allowRendererProcessReuse` to `false` to revert to the old behavior. This flag will only be an option until Electron 11 so you should plan to update your native modules to be context aware.

For more detailed information see #18397.

Deprecated: `BrowserWindow` extension APIs

The following extension APIs have been deprecated:

- `BrowserWindow.addExtension(path)`
- `BrowserWindow.addDevToolsExtension(path)`
- `BrowserWindow.removeExtension(name)`
- `BrowserWindow.removeDevToolsExtension(name)`
- `BrowserWindow.getExtensions()`
- `BrowserWindow.getDevToolsExtensions()`

Use the session APIs instead:

- `ses.loadExtension(path)`
- `ses.removeExtension(extension_id)`
- `ses.getAllExtensions()`

```
// Deprecated in Electron 9
BrowserWindow.addExtension(path)
BrowserWindow.addDevToolsExtension(path)
```

```
// Replace with
session.defaultSession.loadExtension(path)
```

```
// Deprecated in Electron 9
BrowserWindow.removeExtension(name)
BrowserWindow.removeDevToolsExtension(name)
// Replace with
session.defaultSession.removeExtension(extension_id)
```

```
// Deprecated in Electron 9
BrowserWindow.getExtensions()
BrowserWindow.getDevToolsExtensions()
// Replace with
session.defaultSession.getAllExtensions()
```

Removed: `<webview>.getWebContents()`

This API, which was deprecated in Electron 8.0, is now removed.

```
// Removed in Electron 9.0
webview.getWebContents()
// Replace with
const { remote } = require('electron')
remote.webContents.fromId(webview.getWebContentsId())
```

Removed: `webFrame.setLayoutZoomLevelLimits()`

Chromium has removed support for changing the layout zoom level limits, and it is beyond Electron's capacity to maintain it. The function was deprecated in Electron 8.x, and has been removed in Electron 9.x. The layout zoom level limits are now fixed at a minimum of 0.25 and a maximum of 5.0, as defined here.

Behavior Changed: Sending non-JS objects over IPC now throws an exception

In Electron 8.0, IPC was changed to use the Structured Clone Algorithm, bringing significant performance improvements. To help ease the transition, the old IPC serialization algorithm was kept and used for some objects that aren't serializable with Structured Clone. In particular, DOM objects (e.g. `Element`, `Location` and `DOMMatrix`), Node.js objects backed by C++ classes (e.g. `process.env`, some members of `Stream`), and Electron objects backed by C++ classes (e.g. `WebContents`, `BrowserWindow` and `WebFrame`) are not serializable with Structured Clone. Whenever the old algorithm was invoked, a deprecation warning was printed.

In Electron 9.0, the old serialization algorithm has been removed, and sending such non-serializable objects will now throw an "object could not be cloned" error.

API Changed: `shell.openItem` is now `shell.openPath`

The `shell.openItem` API has been replaced with an asynchronous `shell.openPath` API. You can see the original API proposal and reasoning here.

Planned Breaking API Changes (8.0)

Behavior Changed: Values sent over IPC are now serialized with Structured Clone Algorithm

The algorithm used to serialize objects sent over IPC (through `ipcRenderer.send`, `ipcRenderer.sendSync`, `WebContents.send` and related methods) has been switched from a custom algorithm to V8's built-in Structured Clone Algorithm, the same algorithm used to serialize messages for `postMessage`. This brings about a 2x performance improvement for large messages, but also brings some breaking changes in behavior.

- Sending Functions, Promises, WeakMaps, WeakSets, or objects containing any such values, over IPC will now throw an exception, instead of silently converting the functions to `undefined`.

```
// Previously:
ipcRenderer.send('channel', { value: 3, someFunction: () => {} })
// => results in { value: 3 } arriving in the main process

// From Electron 8:
ipcRenderer.send('channel', { value: 3, someFunction: () => {} })
// => throws Error("() => {} could not be cloned.")
```

- `NaN`, `Infinity` and `-Infinity` will now be correctly serialized, instead of being converted to `null`.
- Objects containing cyclic references will now be correctly serialized, instead of being converted to `null`.
- `Set`, `Map`, `Error` and `RegExp` values will be correctly serialized, instead of being converted to `{}`.
- `BigInt` values will be correctly serialized, instead of being converted to `null`.
- Sparse arrays will be serialized as such, instead of being converted to dense arrays with `null`s.
- `Date` objects will be transferred as `Date` objects, instead of being converted to their ISO string representation.
- Typed Arrays (such as `Uint8Array`, `Uint16Array`, `Uint32Array` and so on) will be transferred as such, instead of being converted to Node.js `Buffer`.
- Node.js `Buffer` objects will be transferred as `Uint8Array`s. You can convert a `Uint8Array` back to a Node.js `Buffer` by wrapping the underlying `ArrayBuffer`:

```
Buffer.from(value.buffer, value.byteOffset, value.byteLength)
```

Sending any objects that aren't native JS types, such as DOM objects (e.g. `Element`, `Location`, `DOMMatrix`), Node.js objects (e.g. `process.env`, `Stream`), or Electron objects (e.g. `WebContents`, `BrowserWindow`, `WebFrame`) is deprecated. In Electron 8, these objects will be serialized as before with a DeprecationWarning message, but starting in Electron 9, sending these kinds of objects will throw a 'could not be cloned' error.

Deprecated: `<webview>.getWebContents()`

This API is implemented using the `remote` module, which has both performance and security implications. Therefore its usage should be explicit.

```
// Deprecated
webview.getWebContents()
// Replace with
const { remote } = require('electron')
remote.webContents.fromId(webview.getWebContentsId())
```

However, it is recommended to avoid using the `remote` module altogether.

```
// main
const { ipcMain, webContents } = require('electron')

const getGuestForWebContents = (webContentsId, contents) => {
  const guest = webContents.fromId(webContentsId)
  if (!guest) {
    throw new Error(`Invalid webContentsId: ${webContentsId}`)
  }
  if (guest.hostWebContents !== contents) {
    throw new Error('Access denied to webContents')
  }
  return guest
}

ipcMain.handle('openDevTools', (event, webContentsId) => {
  const guest = getGuestForWebContents(webContentsId, event.sender)
  guest.openDevTools()
})

// renderer
const { ipcRenderer } = require('electron')

ipcRenderer.invoke('openDevTools', webview.getWebContentsId())
```

Deprecated: `webFrame.setLayoutZoomLevelLimits()`

Chromium has removed support for changing the layout zoom level limits, and it is beyond Electron's capacity to maintain it. The function will emit a warning in Electron 8.x, and cease to exist in Electron 9.x. The layout zoom level limits are now fixed at a minimum of 0.25 and a maximum of 5.0, as defined here.

Deprecated events in `systemPreferences`

The following `systemPreferences` events have been deprecated:

- `inverted-color-scheme-changed`
- `high-contrast-color-scheme-changed`

Use the new `updated` event on the `nativeTheme` module instead.

```
// Deprecated
systemPreferences.on('inverted-color-scheme-changed', () => { /* ... */ })
systemPreferences.on('high-contrast-color-scheme-changed', () => { /* ... */ })

// Replace with
nativeTheme.on('updated', () => { /* ... */ })
```

Deprecated: methods in `systemPreferences`

The following `systemPreferences` methods have been deprecated:

- `systemPreferences.isDarkMode()`
- `systemPreferences.isInvertedColorScheme()`
- `systemPreferences.isHighContrastColorScheme()`

Use the following `nativeTheme` properties instead:

- `nativeTheme.shouldUseDarkColors`
- `nativeTheme.shouldUseInvertedColorScheme`
- `nativeTheme.shouldUseHighContrastColors`

```
// Deprecated
systemPreferences.isDarkMode()
// Replace with
nativeTheme.shouldUseDarkColors

// Deprecated
systemPreferences.isInvertedColorScheme()
// Replace with
nativeTheme.shouldUseInvertedColorScheme

// Deprecated
systemPreferences.isHighContrastColorScheme()
// Replace with
nativeTheme.shouldUseHighContrastColors
```

Planned Breaking API Changes (7.0)

Deprecated: Atom.io Node Headers URL

This is the URL specified as `disturl` in a `.npmrc` file or as the `--dist-url` command line flag when building native Node modules. Both will be supported for the foreseeable future but it is recommended that you switch.

Deprecated: https://atom.io/download/electron

Replace with: https://electronjs.org/headers

API Changed: `session.clearAuthCache()` no longer accepts options

The `session.clearAuthCache` API no longer accepts options for what to clear, and instead unconditionally clears the whole cache.

```
// Deprecated
session.clearAuthCache({ type: 'password' })
// Replace with
session.clearAuthCache()
```

API Changed: `powerMonitor.querySystemIdleState` is now `powerMonitor.getSystemIdleState`

```
// Removed in Electron 7.0
powerMonitor.querySystemIdleState(threshold, callback)
// Replace with synchronous API
const idleState = powerMonitor.getSystemIdleState(threshold)
```

API Changed: `powerMonitor.querySystemIdleTime` is now `powerMonitor.getSystemIdleTime`

```
// Removed in Electron 7.0
powerMonitor.querySystemIdleTime(callback)
// Replace with synchronous API
const idleTime = powerMonitor.getSystemIdleTime()
```

API Changed: `webFrame.setIsolatedWorldInfo` replaces separate methods

```
// Removed in Electron 7.0
webFrame.setIsolatedWorldContentSecurityPolicy(worldId, csp)
webFrame.setIsolatedWorldHumanReadableName(worldId, name)
webFrame.setIsolatedWorldSecurityOrigin(worldId, securityOrigin)
// Replace with
webFrame.setIsolatedWorldInfo(
  worldId,
  {
    securityOrigin: 'some_origin',
    name: 'human_readable_name',
    csp: 'content_security_policy'
  })
```

Removed: `marked` property on `getBlinkMemoryInfo`

This property was removed in Chromium 77, and as such is no longer available.

Behavior Changed: `webkitdirectory` attribute for `<input type="file"/>` now lists directory contents

The `webkitdirectory` property on HTML file inputs allows them to select folders. Previous versions of Electron had an incorrect implementation where the `event.target.files` of the input returned a `FileList` that returned one `File` corresponding to the selected folder.

As of Electron 7, that `FileList` is now list of all files contained within the folder, similarly to Chrome, Firefox, and Edge (link to MDN docs).

As an illustration, take a folder with this structure:

```
folder
├── file1
├── file2
└── file3
```

In Electron <=6, this would return a `FileList` with a `File` object for:

```
path/to/folder
```

In Electron 7, this now returns a `FileList` with a `File` object for:

```
/path/to/folder/file3
/path/to/folder/file2
/path/to/folder/file1
```

Note that `webkitdirectory` no longer exposes the path to the selected folder. If you require the path to the selected folder rather than the folder contents, see the `dialog.showOpenDialog` API (link).

API Changed: Callback-based versions of promisified APIs

Electron 5 and Electron 6 introduced Promise-based versions of existing asynchronous APIs and deprecated their older, callback-based counterparts. In Electron 7, all deprecated callback-based APIs are now removed.

These functions now only return Promises:

- `app.getFileIcon()` #15742
- `app.dock.show()` #16904
- `contentTracing.getCategories()` #16583
- `contentTracing.getTraceBufferUsage()` #16600
- `contentTracing.startRecording()` #16584
- `contentTracing.stopRecording()` #16584
- `contents.executeJavaScript()` #17312
- `cookies.flushStore()` #16464
- `cookies.get()` #16464
- `cookies.remove()` #16464
- `cookies.set()` #16464
- `debugger.sendCommand()` #16861
- `dialog.showCertificateTrustDialog()` #17181
- `inAppPurchase.getProducts()` #17355
- `inAppPurchase.purchaseProduct()` #17355
- `netLog.stopLogging()` #16862
- `session.clearAuthCache()` #17259
- `session.clearCache()` #17185
- `session.clearHostResolverCache()` #17229
- `session.clearStorageData()` #17249
- `session.getBlobData()` #17303
- `session.getCacheSize()` #17185
- `session.resolveProxy()` #17222
- `session.setProxy()` #17222
- `shell.openExternal()` #16176
- `webContents.loadFile()` #15855
- `webContents.loadURL()` #15855

- `webContents.hasServiceWorker()` #16535
- `webContents.printToPDF()` #16795
- `webContents.savePage()` #16742
- `webFrame.executeJavaScript()` #17312
- `webFrame.executeJavaScriptInIsolatedWorld()` #17312
- `webviewTag.executeJavaScript()` #17312
- `win.capturePage()` #15743

These functions now have two forms, synchronous and Promise-based asynchronous:

- `dialog.showMessageBox()` / `dialog.showMessageBoxSync()` #17298
- `dialog.showOpenDialog()` / `dialog.showOpenDialogSync()` #16973
- `dialog.showSaveDialog()` / `dialog.showSaveDialogSync()` #17054

Planned Breaking API Changes (6.0)

API Changed: `win.setMenu(null)` is now `win.removeMenu()`

```
// Deprecated
win.setMenu(null)
// Replace with
win.removeMenu()
```

API Changed: `electron.screen` in the renderer process should be accessed via `remote`

```
// Deprecated
require('electron').screen
// Replace with
require('electron').remote.screen
```

API Changed: `require()`ing node builtins in sandboxed renderers no longer implicitly loads the `remote` version

```
// Deprecated
require('child_process')
// Replace with
require('electron').remote.require('child_process')

// Deprecated
require('fs')
// Replace with
require('electron').remote.require('fs')

// Deprecated
require('os')
// Replace with
require('electron').remote.require('os')
```

```
// Deprecated
require('path')
// Replace with
require('electron').remote.require('path')
```

Deprecated: `powerMonitor.querySystemIdleState` replaced with `powerMonitor.getSystemIdleState`

```
// Deprecated
powerMonitor.querySystemIdleState(threshold, callback)
// Replace with synchronous API
const idleState = powerMonitor.getSystemIdleState(threshold)
```

Deprecated: `powerMonitor.querySystemIdleTime` replaced with `powerMonitor.getSystemIdleTime`

```
// Deprecated
powerMonitor.querySystemIdleTime(callback)
// Replace with synchronous API
const idleTime = powerMonitor.getSystemIdleTime()
```

Deprecated: `app.enableMixedSandbox()` is no longer needed

```
// Deprecated
app.enableMixedSandbox()
```

Mixed-sandbox mode is now enabled by default.

Deprecated: `Tray.setHighlightMode`

Under macOS Catalina our former Tray implementation breaks. Apple's native substitute doesn't support changing the highlighting behavior.

```
// Deprecated
tray.setHighlightMode(mode)
// API will be removed in v7.0 without replacement.
```

Planned Breaking API Changes (5.0)

Default Changed: `nodeIntegration` and `webviewTag` default to false, `contextIsolation` defaults to true

The following `webPreferences` option default values are deprecated in favor of the new defaults listed below.

Property	Deprecated Default	New Default
`contextIsolation`	`false`	`true`

`nodeIntegration`	`true`	`false`
`webviewTag`	`nodeIntegration` if set else `true`	`false`

E.g. Re-enabling the webviewTag

```
const w = new BrowserWindow({
  webPreferences: {
    webviewTag: true
  }
})
```

Behavior Changed: `nodeIntegration` in child windows opened via `nativeWindowOpen`

Child windows opened with the `nativeWindowOpen` option will always have Node.js integration disabled, unless `nodeIntegrationInSubFrames` is `true`.

API Changed: Registering privileged schemes must now be done before app ready

Renderer process APIs `webFrame.registerURLSchemeAsPrivileged` and `webFrame.registerURLSchemeAsBypassingCSP` as well as browser process API `protocol.registerStandardSchemes` have been removed. A new API, `protocol.registerSchemesAsPrivileged` has been added and should be used for registering custom schemes with the required privileges. Custom schemes are required to be registered before app ready.

Deprecated: `webFrame.setIsolatedWorld*` replaced with `webFrame.setIsolatedWorldInfo`

```
// Deprecated
webFrame.setIsolatedWorldContentSecurityPolicy(worldId, csp)
webFrame.setIsolatedWorldHumanReadableName(worldId, name)
webFrame.setIsolatedWorldSecurityOrigin(worldId, securityOrigin)
// Replace with
webFrame.setIsolatedWorldInfo(
  worldId,
  {
    securityOrigin: 'some_origin',
    name: 'human_readable_name',
    csp: 'content_security_policy'
  })
```

API Changed: `webFrame.setSpellCheckProvider` now takes an asynchronous callback

The `spellCheck` callback is now asynchronous, and `autoCorrectWord` parameter has been removed.

```
// Deprecated
webFrame.setSpellCheckProvider('en-US', true, {
  spellCheck: (text) => {
    return !spellchecker.isMisspelled(text)
```

```
  }
})
// Replace with
webFrame.setSpellCheckProvider('en-US', {
  spellCheck: (words, callback) => {
    callback(words.filter(text => spellchecker.isMisspelled(text)))
  }
})
```

API Changed: `webContents.getZoomLevel` and `webContents.getZoomFactor` are now synchronous

`webContents.getZoomLevel` and `webContents.getZoomFactor` no longer take callback parameters, instead directly returning their number values.

```
// Deprecated
webContents.getZoomLevel((level) => {
  console.log(level)
})
// Replace with
const level = webContents.getZoomLevel()
console.log(level)
```

```
// Deprecated
webContents.getZoomFactor((factor) => {
  console.log(factor)
})
// Replace with
const factor = webContents.getZoomFactor()
console.log(factor)
```

Planned Breaking API Changes (4.0)

The following list includes the breaking API changes made in Electron 4.0.

`app.makeSingleInstance`

```
// Deprecated
app.makeSingleInstance((argv, cwd) => {
  /* ... */
})
// Replace with
app.requestSingleInstanceLock()
app.on('second-instance', (event, argv, cwd) => {
  /* ... */
})
```

`app.releaseSingleInstance`

```
// Deprecated
app.releaseSingleInstance()
// Replace with
app.releaseSingleInstanceLock()
```

`app.getGPUInfo`

```
app.getGPUInfo('complete')
// Now behaves the same with `basic` on macOS
app.getGPUInfo('basic')
```

`win_delay_load_hook`

When building native modules for windows, the `win_delay_load_hook` variable in the module's `binding.gyp` must be true (which is the default). If this hook is not present, then the native module will fail to load on Windows, with an error message like `Cannot find module`. See the native module guide for more.

Breaking API Changes (3.0)

The following list includes the breaking API changes in Electron 3.0.

`app`

```
// Deprecated
app.getAppMemoryInfo()
// Replace with
app.getAppMetrics()

// Deprecated
const metrics = app.getAppMetrics()
const { memory } = metrics[0] // Deprecated property
```

`BrowserWindow`

```
// Deprecated
const optionsA = { webPreferences: { blinkFeatures: '' } }
const windowA = new BrowserWindow(optionsA)
// Replace with
const optionsB = { webPreferences: { enableBlinkFeatures: '' } }
const windowB = new BrowserWindow(optionsB)

// Deprecated
window.on('app-command', (e, cmd) => {
  if (cmd === 'media-play_pause') {
    // do something
  }
})
```

```
// Replace with
window.on('app-command', (e, cmd) => {
  if (cmd === 'media-play-pause') {
    // do something
  }
})
```

`clipboard`

```
// Deprecated
clipboard.readRtf()
// Replace with
clipboard.readRTF()

// Deprecated
clipboard.writeRtf()
// Replace with
clipboard.writeRTF()

// Deprecated
clipboard.readHtml()
// Replace with
clipboard.readHTML()

// Deprecated
clipboard.writeHtml()
// Replace with
clipboard.writeHTML()
```

`crashReporter`

```
// Deprecated
crashReporter.start({
  companyName: 'Crashly',
  submitURL: 'https://crash.server.com',
  autoSubmit: true
})
// Replace with
crashReporter.start({
  companyName: 'Crashly',
  submitURL: 'https://crash.server.com',
  uploadToServer: true
})
```

`nativeImage`

```
// Deprecated
nativeImage.createFromBuffer(buffer, 1.0)
```

```
// Replace with
nativeImage.createFromBuffer(buffer, {
  scaleFactor: 1.0
})
```

process

```
// Deprecated
const info = process.getProcessMemoryInfo()
```

screen

```
// Deprecated
screen.getMenuBarHeight()
// Replace with
screen.getPrimaryDisplay().workArea
```

session

```
// Deprecated
ses.setCertificateVerifyProc((hostname, certificate, callback) => {
  callback(true)
})
// Replace with
ses.setCertificateVerifyProc((request, callback) => {
  callback(0)
})
```

Tray

```
// Deprecated
tray.setHighlightMode(true)
// Replace with
tray.setHighlightMode('on')

// Deprecated
tray.setHighlightMode(false)
// Replace with
tray.setHighlightMode('off')
```

webContents

```
// Deprecated
webContents.openDevTools({ detach: true })
// Replace with
```

```
webContents.openDevTools({ mode: 'detach' })

// Removed
webContents.setSize(options)
// There is no replacement for this API
```

`webFrame`

```
// Deprecated
webFrame.registerURLSchemeAsSecure('app')
// Replace with
protocol.registerStandardSchemes(['app'], { secure: true })

// Deprecated
webFrame.registerURLSchemeAsPrivileged('app', { secure: true })
// Replace with
protocol.registerStandardSchemes(['app'], { secure: true })
```

`<webview>`

```
// Removed
webview.setAttribute('disableguestresize', '')
// There is no replacement for this API

// Removed
webview.setAttribute('guestinstance', instanceId)
// There is no replacement for this API

// Keyboard listeners no longer work on webview tag
webview.onkeydown = () => { /* handler */ }
webview.onkeyup = () => { /* handler */ }
```

Node Headers URL

This is the URL specified as `disturl` in a `.npmrc` file or as the `--dist-url` command line flag when building native Node modules.

Deprecated: https://atom.io/download/atom-shell

Replace with: https://atom.io/download/electron

Breaking API Changes (2.0)

The following list includes the breaking API changes made in Electron 2.0.

`BrowserWindow`

```
// Deprecated
const optionsA = { titleBarStyle: 'hidden-inset' }
```

```
const windowA = new BrowserWindow(optionsA)
// Replace with
const optionsB = { titleBarStyle: 'hiddenInset' }
const windowB = new BrowserWindow(optionsB)
```

`menu`

```
// Removed
menu.popup(browserWindow, 100, 200, 2)
// Replaced with
menu.popup(browserWindow, { x: 100, y: 200, positioningItem: 2 })
```

`nativeImage`

```
// Removed
nativeImage.toPng()
// Replaced with
nativeImage.toPNG()

// Removed
nativeImage.toJpeg()
// Replaced with
nativeImage.toJPEG()
```

`process`

- `process.versions.electron` and `process.version.chrome` will be made read-only properties for consistency with the other `process.versions` properties set by Node.

`webContents`

```
// Removed
webContents.setZoomLevelLimits(1, 2)
// Replaced with
webContents.setVisualZoomLevelLimits(1, 2)
```

`webFrame`

```
// Removed
webFrame.setZoomLevelLimits(1, 2)
// Replaced with
webFrame.setVisualZoomLevelLimits(1, 2)
```

`<webview>`

```
// Removed
webview.setZoomLevelLimits(1, 2)
// Replaced with
webview.setVisualZoomLevelLimits(1, 2)
```

Duplicate ARM Assets

Each Electron release includes two identical ARM builds with slightly different filenames, like `electron-v1.7.3-linux-arm.zip` and `electron-v1.7.3-linux-armv7l.zip`. The asset with the `v7l` prefix was added to clarify to users which ARM version it supports, and to disambiguate it from future armv6l and arm64 assets that may be produced.

The file *without the prefix* is still being published to avoid breaking any setups that may be consuming it. Starting at 2.0, the unprefixed file will no longer be published.

For details, see 6986 and 7189.