Class: Cookies

Query and modify a session's cookies.

Process: Main *This class is not exported from the `'electron'` module. It is only available as a return value of other methods in the Electron API.*

Instances of the `Cookies` class are accessed by using `cookies` property of a `Session`.

For example:

```
const { session } = require('electron')

// Query all cookies.
session.defaultSession.cookies.get({})
  .then((cookies) => {
    console.log(cookies)
  }).catch((error) => {
    console.log(error)
  })

// Query all cookies associated with a specific url.
session.defaultSession.cookies.get({ url: 'http://www.github.com' })
  .then((cookies) => {
    console.log(cookies)
  }).catch((error) => {
    console.log(error)
  })

// Set a cookie with the given cookie data;
// may overwrite equivalent cookies if they exist.
const cookie = { url: 'http://www.github.com', name: 'dummy_name', value: 'dummy' }
session.defaultSession.cookies.set(cookie)
  .then(() => {
    // success
  }, (error) => {
    console.error(error)
  })
```

Instance Events

The following events are available on instances of `Cookies`:

Event: 'changed' Returns:

- `event` Event
- `cookie` Cookie - The cookie that was changed.
- `cause` string - The cause of the change with one of the following values:

  - `explicit` - The cookie was changed directly by a consumer's action.
  - `overwrite` - The cookie was automatically removed due to an insert operation that overwrote it.
  - `expired` - The cookie was automatically removed as it expired.
  - `evicted` - The cookie was automatically evicted during garbage collection.
  - `expired-overwrite` - The cookie was overwritten with an already-expired expiration date.
- `removed` boolean - `true` if the cookie was removed, `false` otherwise.

Emitted when a cookie is changed because it was added, edited, removed, or expired.

Instance Methods

The following methods are available on instances of `Cookies`:

`cookies.get(filter)`

- `filter` Object
  - `url` string (optional) - Retrieves cookies which are associated with `url`. Empty implies retrieving cookies of all URLs.
  - `name` string (optional) - Filters cookies by name.
  - `domain` string (optional) - Retrieves cookies whose domains match or are subdomains of `domains`.
  - `path` string (optional) - Retrieves cookies whose path matches `path`.
  - `secure` boolean (optional) - Filters cookies by their Secure property.
  - `session` boolean (optional) - Filters out session or persistent cookies.

Returns `Promise<Cookie[]>` - A promise which resolves an array of cookie objects.

Sends a request to get all cookies matching `filter`, and resolves a promise with the response.

`cookies.set(details)`

- `details` Object
  - `url` string - The URL to associate the cookie with. The promise will be rejected if the URL is invalid.
  - `name` string (optional) - The name of the cookie. Empty by default if omitted.
  - `value` string (optional) - The value of the cookie. Empty by default if omitted.
  - `domain` string (optional) - The domain of the cookie; this will be normalized with a preceding dot so that it's also valid for subdomains. Empty by default if omitted.

  - `path` string (optional) - The path of the cookie. Empty by default if omitted.
  - `secure` boolean (optional) - Whether the cookie should be marked as Secure. Defaults to false unless Same Site=None attribute is used.
  - `httpOnly` boolean (optional) - Whether the cookie should be marked as HTTP only. Defaults to false.
  - `expirationDate` Double (optional) - The expiration date of the cookie as the number of seconds since the UNIX epoch. If omitted then the cookie becomes a session cookie and will not be retained between sessions.
  - `sameSite` string (optional) - The Same Site policy to apply to this cookie. Can be `unspecified`, `no_restriction`, `lax` or `strict`. Default is `lax`.

Returns `Promise<void>` - A promise which resolves when the cookie has been set

Sets a cookie with `details`.

`cookies.remove(url, name)`

- `url` string - The URL associated with the cookie.
- `name` string - The name of cookie to remove.

Returns `Promise<void>` - A promise which resolves when the cookie has been removed

Removes the cookies matching `url` and `name`

`cookies.flushStore()` Returns `Promise<void>` - A promise which resolves when the cookie store has been flushed

Writes any unwritten cookies data to disk.