DevTools Extension

Electron supports Chrome DevTools extensions, which can be used to extend the ability of Chrome's developer tools for debugging popular web frameworks.

Loading a DevTools extension with tooling

The easiest way to load a DevTools extension is to use third-party tooling to automate the process for you. electron-devtools-installer is a popular NPM package that does just that.

Manually loading a DevTools extension

If you don't want to use the tooling approach, you can also do all of the necessary operations by hand. To load an extension in Electron, you need to download it via Chrome, locate its filesystem path, and then load it into your Session by calling the [`ses.loadExtension`] API.

Using the React Developer Tools as an example:

1. Install the extension in Google Chrome.
2. Navigate to `chrome://extensions`, and find its extension ID, which is a hash string like `fmkadmapgofadopljbjfkapdkoienihi`.
3. Find out the filesystem location used by Chrome for storing extensions:
   - on Windows it is `%LOCALAPPDATA%\Google\Chrome\User Data\Default\Extensions`;
   - on Linux it could be:
     - `~/.config/google-chrome/Default/Extensions/`
     - `~/.config/google-chrome-beta/Default/Extensions/`
     - `~/.config/google-chrome-canary/Default/Extensions/`
     - `~/.config/chromium/Default/Extensions/`
   - on macOS it is `~/Library/Application Support/Google/Chrome/Default/Extensions`.
4. Pass the location of the extension to the `ses.loadExtension` API. For React Developer Tools `v4.9.0`, it looks something like:

```
const { app, session } = require('electron')
const path = require('path')
const os = require('os')

// on macOS
const reactDevToolsPath = path.join(
  os.homedir(),
  '/Library/Application Support/Google/Chrome/Default/Extensions/fmkadmapgofadopljbjfkapdkoienihi/4.9.0_
)

app.whenReady().then(async () => {
```

```
  await session.defaultSession.loadExtension(reactDevToolsPath)
})
```

Notes:

- `loadExtension` returns a Promise with an Extension object, which contains metadata about the extension that was loaded. This promise needs to resolve (e.g. with an `await` expression) before loading a page. Otherwise, the extension won't be guaranteed to load.
- `loadExtension` cannot be called before the `ready` event of the `app` module is emitted, nor can it be called on in-memory (non-persistent) sessions.
- `loadExtension` must be called on every boot of your app if you want the extension to be loaded.

Removing a DevTools extension

You can pass the extension's ID to the `ses.removeExtension` API to remove it from your Session. Loaded extensions are not persisted between app launches.

DevTools extension support

Electron only supports a limited set of `chrome.*` APIs, so extensions using unsupported `chrome.*` APIs under the hood may not work.

The following Devtools extensions have been tested to work in Electron:

- Ember Inspector
- React Developer Tools
- Backbone Debugger
- jQuery Debugger
- AngularJS Batarang
- Vue.js devtools
- Cerebral Debugger
- Redux DevTools Extension
- MobX Developer Tools

What should I do if a DevTools extension is not working?

First, please make sure the extension is still being maintained and is compatible with the latest version of Google Chrome. We cannot provide additional support for unsupported extensions.

If the extension works on Chrome but not on Electron, file a bug in Electron's issue tracker and describe which part of the extension is not working as expected.