LAB 3: ELECTRON DEVELOPMENT TOOLS

In this exercise, you will:

- Set up electron-reload for live reload.
- Set up devtron for debugging.
- Set up electron-is-dev to check if electron is running in development mode.
- Set up electron-log for logging.

1. Check out the `Lab-Electron-Development-Tools` branch from the remote repository.

```
git checkout origin/Lab-Electron-Development-Tools
```

Note: The HEAD of this branch provides the completed code for the entire exercise.

---

2. Run `npm install` to ensure all the dependencies have been installed.
3. Create a new local branch for your work based on the "starting point..." commit.

```
git log --oneline
git checkout <hash for starting point commit>
git checkout -b Lab-Electron-Development-Tools-mine
```

4. Start the app.

```
npm run start
```

electron-reload

5. Install electron-reload.

```
npm install --save-dev electron-reload
```

6. Navigate to main.js at the root level of project and add below code just under the first line. Save your file.

```
const path = require('path')

require('electron-reload')(__dirname, {

  electron: path.join(__dirname, 'node_modules', '.bin', 'electron')

});
```

7. Restart the app.

```
npm run start
```

Note: Now you can check your app for live reload by changing any of the files under assets folder. And you will observe electron app reloads itself. You can read more about its usage by visiting this URL **HTTPS://WWW.NPMJS.COM/PACKAGE/ELECTRON-RELOAD**

---

devtron

8. Install devtron.

```
npm install --save-dev devtron
```

9. Navigate to main.js at the root level of project. Add below lines of code inside createWindow function.

```
require('devtron').install()
```

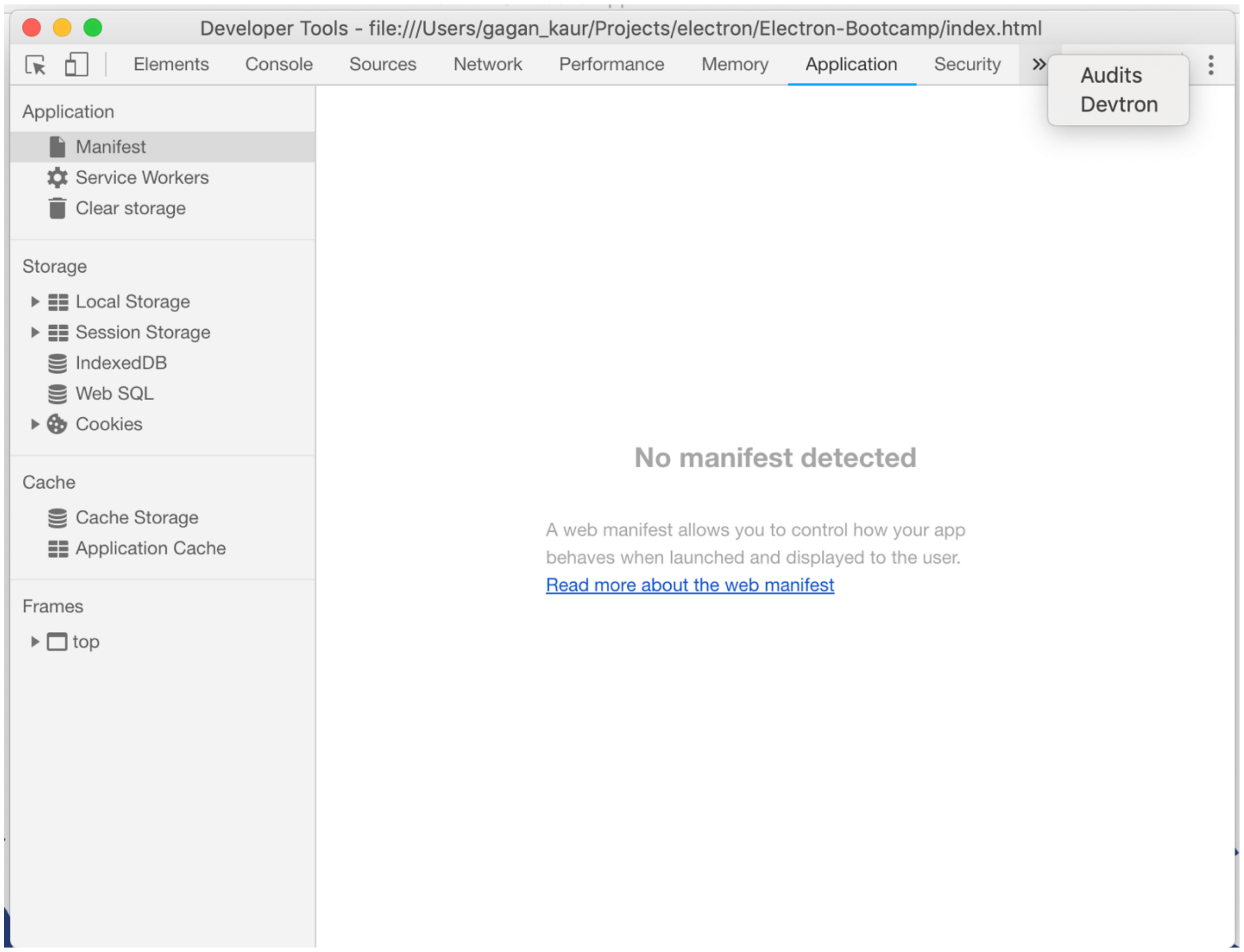
Note: You can add above line of code anywhere inside createWindow function.

---

10. Restart the app.

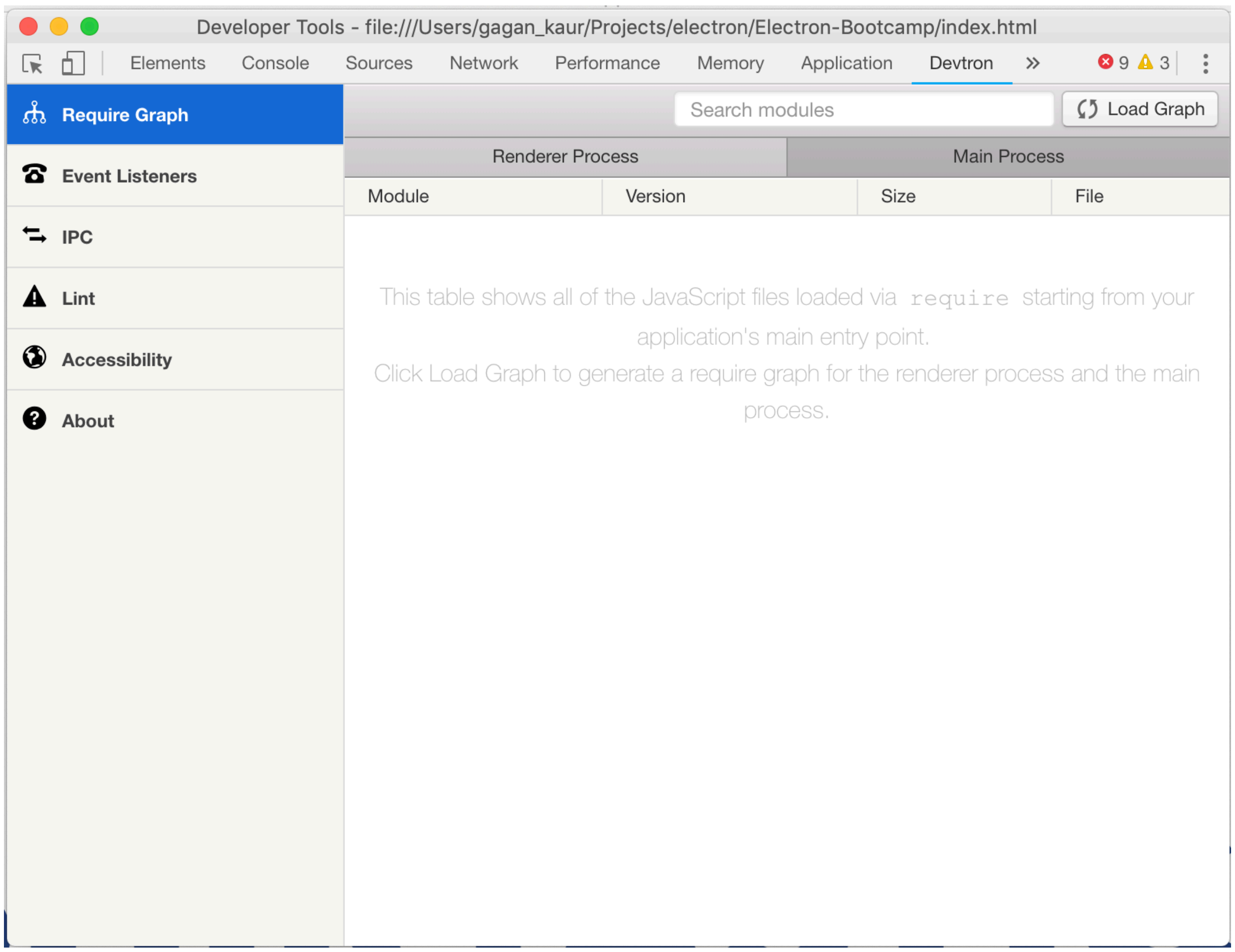
```
npm run start
```

Note: If you notice there is a line of code inside createWindow function win.webContents.openDevTools(). This opens the default developer tools for electron app. Now in addition to this we will get Devtron installed and ready to be used for more detailed analysis of electron app. You can read more about its usage by visiting this URL **HTTPS://ELECTRONJS.ORG/DEVTRON**



No manifest detected

A web manifest allows you to control how your app behaves when launched and displayed to the user.
Read more about the web manifest


Developer Tools - file:///Users/gagan_kaur/Projects/electron/Electron-Bootcamp/index.html

Module	Version	Size	File

This table shows all of the JavaScript files loaded via `require` starting from your application's main entry point.
Click Load Graph to generate a require graph for the renderer process and the main process.

electron-is-dev

11. Install electron-is-dev.

```
npm install --save electron-is-dev
```

12. Navigate to main.js file at the root level of project. Add below line of code at the top level just under first line.

```
const isDev = require('electron-is-dev');
```

13. Now you can use isDev variable in the main.js file. This variable can be used to place conditions for code which needs to be run only in development mode.
14. For example, we can place a check in front of live reload code. So, this code does not run in the production version of the app.

```
if(isDev) {

  require('electron-reload')(__dirname, {

    electron: path.join(__dirname, 'node_modules', '.bin', 'electron')

  });

}
```

15. Another important usage can be placing a check against dev tools and devtron debugger tool. We do not want the end users to see dev tools open in the production version.

```
if(isDev) {

      // Open the DevTools.

      win.webContents.openDevTools()

      // Devtron for debugging application

      require('devtron').install();

    }
```

Note: You can read more about its usage by visiting this URL
HTTPS://WWW.NPMJS.COM/PACKAGE/ELECTRON-IS-DEV

electron-log

16. Install electron-log.

```
npm install --save electron-log
```

17. Navigate to main.js file at the root level of project. Add below line of code at the top level just under first line.

```
const log = require('electron-log');
```

18. Now you can make use of log variable to place logging information in various parts of your application. For example, we can place a log inside createWindow function.

```
log.info('window creation');
```

19. Restart the app.

```
npm run start
```

Note: This will generate logs in the terminal where your app is running.