Electron from Webphone

Electron is a framework for creating native applications with web technologies like JavaScript, HTML, and CSS. This enables us to build cross-platform desktop apps from the Webphone. Below we will discuss the necessary steps tools and steps to achieve this, by walking trough an example. This will be just a simple example, which can be further developed to fit your needs. Electron also provides a powerful and rich set of API which gives access to the underlying chrome engine and OS API.

Prerequisites

- Node.js (includes npm)
- Webphone package

Install Node.js and an Electron project directory

Let's follow the below steps to create an electron app using the Webphone.

1. First download and install Node.js from here. Also download and unzip the webphone package sent by Mizutech or the demo version from here.
2. Create a project directory which will contain all the necessary directory structure and files for the electron app. We will name our project directory: `electronWebphone`.
3. Open the command-line interface on your operating system: Command Prompt on Windows, Terminal on Mac OS X or Linux. Navigate to `electronWebphone` project directory.
4. Run `npm init` command to initialize the project.

Create the electron project

As far as development is concerned, an Electron application is essentially a Node.js application. The starting point is a `package.json` that is identical to that of a Node.js module.

Run `npm init` command and this will guide you through the steps for creating the `package.json` file.

Our `package.json` file looks like:

```
{
  "name": "emizuphone",
  "version": "1.0.0",
  "description": "Mizu VoIP Softphone Electron Module",
  "main": "webphone_main.js",
  "scripts": {
    "start": "electron ."
  },
  "author": "Mizutech",
  "license": "GPL-3.0",
  "devDependencies": {
    "electron": "^2.0.5"
  },
  "dependencies": {
    "electron-packager": "^12.1.0"
  }
}
```

The two most important fields are:

- "main" – the script file specified here will be the entry point to our project. This will be loaded when we start the app.
- *"start": "electron ."* - "start" field must always have the value: "electron ."

Install the electron module for our application.
Just run the following command: `npm install electron --save-dev`

Create a directory named `content` inside the `electronWebphone` directory, then copy the whole wephone package to this directory.

Create *webphone_main.js*

The `webphone_main.js` should create windows and handle all the system events your application might encounter.
Below is a ready to use example on wephone_main.js with explanation comments:

```
var WEBPHONE_HTML_FILE_PATH = 'content/softphone.html';
var APP_WINDOW_DEF_WIDTH = 370;
var APP_WINDOW_DEF_HEIGHT = 700;
var DEBUG = true;

const {app, BrowserWindow} = require('electron')

  // Keep a global reference of the window object, if you don't, the window will
  // be closed automatically when the JavaScript object is garbage collected.
  let win

  function createWindow ()
  {
    if (DEBUG === true) { APP_WINDOW_DEF_WIDTH = APP_WINDOW_DEF_WIDTH *2; }

// Create the browser window.
    win = new BrowserWindow({width: APP_WINDOW_DEF_WIDTH, height: APP_WINDOW_DEF_HEIGHT})

    // and load the index.html of the app.
    win.loadFile(WEBPHONE_HTML_FILE_PATH)

    if (DEBUG === true)
    {
        // Open the DevTools.
        win.webContents.openDevTools()
    }

    // Emitted when the window is closed.
    win.on('closed', () => {
      // Dereference the window object, usually you would store windows
      // in an array if your app supports multi windows, this is the time
      // when you should delete the corresponding element.
      win = null
    })
  }

  // This method will be called when Electron has finished
  // initialization and is ready to create browser windows.
  // Some APIs can only be used after this event occurs.
  app.on('ready', createWindow)

  // Quit when all windows are closed.
  app.on('window-all-closed', () => {
    // On macOS it is common for applications and their menu bar
    // to stay active until the user quits explicitly with Cmd + Q
    if (process.platform !== 'darwin') {
      app.quit()
```

```
  }
})

app.on('activate', () => {
  // On macOS it's common to re-create a window in the app when the
  // dock icon is clicked and there are no other windows open.
  if (win === null) {
    createWindow()
  }
})

// In this file you can include the rest of your app's specific main process
// code. You can also put them in separate files and require them here.
```

Running the Webphone electron app

Once you have completed all the above steps, we have a perfectly working electron application. To launch the application just run the following command: `npm start`

Application Distribution

We are going to use *electron-packager* package manager to help us build distribution packages of our app, for all desktop operating systems.

1. In command prompt navigate to `electronWebphone` app directory.
2. Run command `npm install -g electron-package` to install the package manager.
3. Run command `electron-packager . --all` to create the distribution packages.

Now we have all the distribution packages for all the desktop operating systems.

Documentations and useful links

https://electronjs.org/

https://electronjs.org/docs/tutorial/first-app

https://electronjs.org/docs

https://www.mizu-voip.com/Software/WebPhone.aspx

https://enupal.com/blog/como-crear-un-instalador-multiplataforma-con-electron-para-windows-linux-y-mac

Build instructions for Mbuilder (for Mizutech internal support only)

In template directory/package.json:

- Set brandname for “name” parameter
- Set description for “description” parameter
- Set company name for “author” parameter
- Copy newly built webphone package to template directory/content/
- Go to the electron root app directory and run command: `electron-packager . –all` to create the distribution packages for all Desktop operating systems