International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

Development of Cross-Platform Desktop Apps using Electron Framework

Aravind Kumar B[1], Anitha Sandeep[2]

Department of Computer Science and Engineering, R V College of Engineering, Affiliated to VTU, Bengaluru – 560059, India

Abstract: ***Desktop applications have their own benefits due to which they have the tendency to outperform web applications which are dependent upon a Web Browser which acts as the client interface. Browsers are many in number and vary from one another. They also keep getting upgraded as result of which one browser exists in many versions with varying feature and characteristics. At times, certain web applications may not be allowed to run on a particular web browser due to lack of compatibility or absence of specific version. One possible solution to this problem is development of Desktop Applications and in order to achieve this, a commonly used framework is the Electron Framework.***

Keywords: Browser, Compatibility, Electron JavaScript Modules, JSON, NodeJs, npm, Packaging, Platform

1. Introduction

Desktop applications grabbed the attention and became the center of attraction to the users and people when computers initially became popular among them and were often commonly used with ease at homes for personal use or at offices/workplaces for professional use. However, as a consequence of the advancement of internet and the online commerce boom, the ideas and mindset of people changed and web application became prominent. The future and further usage of desktop applications was regarded to be bleak, however it did not completely die out. There were some fundamental differences between the two that made both of them beneficial or essential in different circumstances.

Desktop Based Applications:
A software that has been installed on one single computer and performs functions and tasks specifically for which it was designed and developed is known as a desktop application.

Web Based Applications:
Web based applications are applications that mostly cannot function or operate without the aid of internet connectivity. Irrespective of the local network, they have the capability to run on multiple devices. They are called commonly referred to as cross-browser web applications as they can launched using different web browsers.

Web applications are mostly built on the client-server model and use a suitable and compatible web browser as the client interface.

An important difference between the two exists. Web applications must possess cross-browser compatibility and desktop applications must possess cross-platform compatibility. However, it not easy to achieve both in a simple budget. Web developers were able to create desktop applications for major OS's using modern languages like JavaScript and Python. However, they faced certain challenges. They had to learn the web technologies, languages and their APIs separately for developing the applications which was tedious and time consuming.

This is the point where the web developers had to think about transiting to desktop application development using existing web technologies and make them perform equally well on multiple platforms. But to do that they needed something that would enable them to build applications that would function across multiple platforms. Electron Framework came into existence at this crucial juncture and progressively seeked attention and popularity. Electron Framework allows the developers to make the best use of their skills and knowledge to build highly functional desktop applications.

2. Related Works

In [1], there is a comparison of Native Apps vs. Cross-platform and the differences are highlighted.It also explains working principle of Electron and why developers take the Electron way.It also tells when to choose Electron and gives example of certain notable desktop applications developed using Electron.

[2] explains as to why Electron Framework is a good choice for cross-platform desktop application development. Certain advantages are highlighted in the article. Developing a cross-platform desktop application saves time and coding is to be done only once to establish a unified codebase.

[3] describes cross-platform mobile application development using web technologies such as HTML,CSS and JavaScript.The paper presents a comparison among four very popular cross platform tools, which are Rhodes, PhoneGap, DragonRad and MoSync.

[4] provides a comparison between Native applications, Web applications and Hybrid applications indicating the pros and cons of each of them.It also gives a number of advantages of hybrid applications.

[5] speaks about Electrino and how it can contribute towards achieving the reduction in memory consumption of Electron desktop applications.

3. Web –VS- Desktop Apps

Building of desktop applications definitely have certain importance and benefits. Due to which it is not possible to completely replace or neglect desktop applications. The key motives for preferring desktop applications are quite a few in number. There is no worry about data being hacked or data being lost because, all the data is stored within the user's computer system.Where as this does not apply in case web applications. Protection from vulnerabilities can be achieved because the user has complete control over the standalone application. Web applications are dependent upon a web browser which acts as the client interface. Browsers are many in number and vary from one another. They also keep getting upgraded as result of which one browser exists in many versions with varying features and characteristics. At times, certain web applications may not be allowed to run on a particular web browser due to lack of compatibility or absence of specific version. In the development of desktop applications there is no question of web browser acting as a client interface. It is the desktop applications which itself acts as the standalone client. However, web technologies such as HTML5, CSS3, JavaScript etc. may be essential for developing desktop applications. The user can avail the desktop application providing the same functionalities in place of the corresponding web application.

4. Electron Framework

Electron Framework is mainly used for cross-platform desktop applications development with the existing web technologies such as HTML,CSS and JavaScript. Chromium and Node.Js are combined into one single runtime by Electron. Windows, Mac and Linux are the platforms to which Electron applications are typically packaged to. Figure 1 shows Electron Framework logo.

Figure 1: Electron Apps Logo

The number of processes that are available in the Electron Framework is two. They are fundamentallydifferent from one another and they are as follows:

1) The Main Process2) The Renderer Process

In the Electron Framework, the main process is responsible for running the package.json's main script. By means of creation of a number of web pages the script that runs in the main process will be able to display a GUI. Main process is always one in number and never more than that. Renderer processes can be multiple in number. A renderer process is run by each browser window due to which there are multiple number of renderer processes. The renderer process typically renders the UI of the application in the window. Hence, the term renderer process. This has been represented using Figure 2.2.

Managing all the web pages and their corresponding renderer processes is handled by the main process. The renderer processes are isolated from one another and are only concerned about the web page that is running it.

The basic file structure is as follows:

- *index.html*
- *main.js*
- *package.json*
- *render.js*

index.htmlwhich is an HTML5 web page serving one big purpose i.e our canvas.

main.jscreates windows and handles system events.

package.jsonis the startup script for the application. It will run in the main process and it contains information about the application.

render.jshandles the application's render processes.

Packaging and distribution are crucial aspects associated with the desktop application development process. Being a cross-platform desktop application development framework, Electron must enable easy approach for the packaging and distribution of applications for different platforms. A project known as Electron-Packager has been developed by the Electron Community that handles this. A proper executable must be available so that it would allow the users to install the Desktop Application onto their machines. This is taken care of by the Electron Packager. Packaging an Electron application simply means creating a desktop installer for the required operating system. It identifies the platform of the system and according integrates the application to the platform thus generating an OS-specific desktop application. Figure 2 represents role of electron packager.

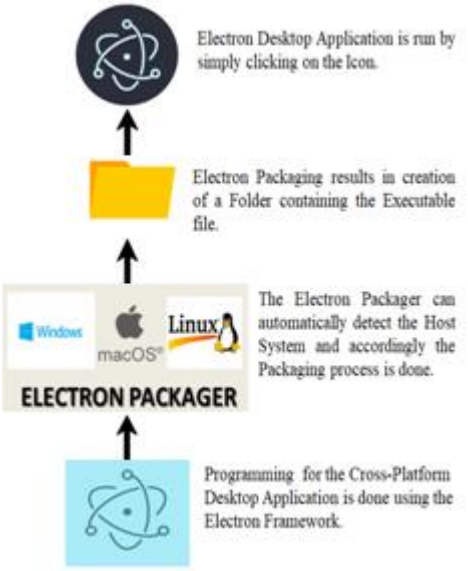

Figure 2: Role of Electron Packager

5. Implementation

The key steps involved in the methodology of Electron App development are as follows:

1) Development of Web App.
2) Porting Web App to Electron Framework to obtain Cross-Platform Desktop App.
3) Electron Packaging of Cross-Platform Desktop App to obtain OS Specific Desktop App.
4) Running the executable and installing OS-Specific Desktop App.
5) Clicking the Icon to start the Desktop App.

Figure 3 is a representation of the Electron App development methodology.

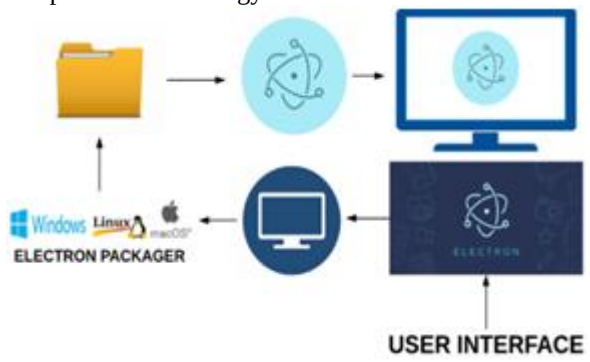

Figure 3: Electron App Development Methodology

6. Result and Analysis

The Web application ported to Electron Framework was able to run as a Desktop Application without the use of any Web Browser.However, Electron Framework based desktop applications use Chromium internally to render HTML and CSS.

Electron Packager was able to detect the host system's platform and perform packaging of Desktop Application across different platforms such as Windows, Mac and Linux. The Electron Desktop Application runs by giving the same feel, look and consistency across different platforms.

Troubleshooting was done to ensure compatibility of Desktop Application with different platforms and no issues were detected as a result of testing the program with different platforms.

However, certain negative aspects could also be identified w.r.t the Electron Desktop Applications.

The loading time consumed by the desktop application is slightly greater than running the web application using a web browser. This is because HTML and CSS are to be rendered and JavaScript is to be executed by NodeJs. JavaScript exists in the form of modules which are all stored in the npm registry. The JavaScript modules required to be used are specified in the 'package.json' file. These modules are to be searched and obtained from the npm registry due to which loading time is slightly high in case of Electron Desktop Applications. Figure 4 and Figure 5 represents performance of web application and Electron application respectively.

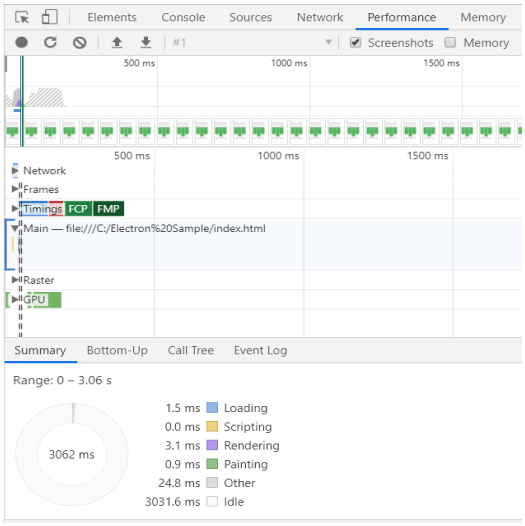

Figure 4: Performance of Web Application

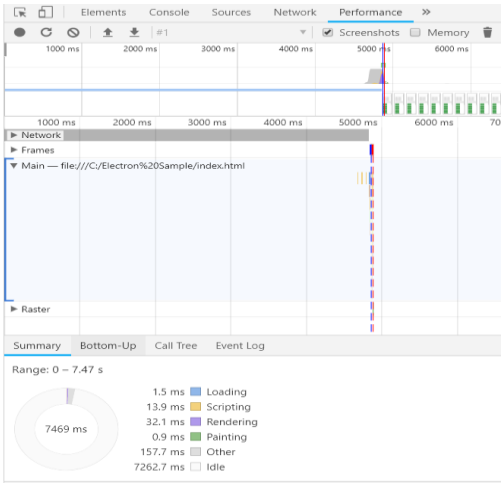

Figure 5: Performance of Electron Application

Electron applications are essentially a fully-featured Chromium browser and a Node process that communicate with one another via IPC.Packaged Electron applications are generally quite large in terms of memory. More and more web pages can be ported to Electron Framework and hosted together in the form of one Desktop Application. However with the increase in number of web pages ported to Electron Framework, post Electron-Packaging stage, it can be observed that there is steady increase in memory consumed.

Electron applications often consume a lot of system resources and also uses quite a large amount of battery power when run for a long span of time.

Electron does not allow the developer to have access the widget toolkits of the target operating system.

7. Conclusions

Electron Framework successfully enables hosting of the existing web application in the form of cross-platform desktop application which provides a common feel, look and consistency across different platforms post electron-packaging.

Electron Packager is able to detect the host system's platform and perform integration of Desktop Application with the platform as a result of which it becomes installed in the host system and can be run as a standalone application by simply clicking on the Icon appearing on screen instead of launching using any web browser.

However, there are few performance issues and drawbacks associated with the Electron Desktop Applications as well such as memory consumption, high loading time & response time when compared to web application.

References

[1] Janis B.,"Age of Electron : Rise of Cross Platform Desktop Apps",Netcore Web Development Services, 25 October, 2018
[2] Ryan Chatterton,"Electron, is it right for building cross-platform applications?",SharpNotions, August 9,2017
[3] Manuel P, Inderjeet Singh, Antonio Cicchetti, "Comparison of Cross-Platform Mobile Development Tools",16th International Conference on Intelligence in Next Generation Networks,2016
[4] Paulo R. M. de Andrade, Adriano B. Albuquerque,"Cross-Platform App-A comparative study",International Journal of Computer Science & Information Technology (IJCSIT),Vol 7, No 1, February 2015
[5] PauliOlaviOjala, "Put Electron app on a diet with Electrino", DailyJS, May 4,2018
[6] N. M. Hui,L. B. Chienget, W. Y. Ting, H.H.Mohamed and M. Rafie,"Cross-Platform Mobile Applications for Android and iOS", IFIP WMNC,IEEE 2015
[7] Danny Markov," Creating Your First Desktop App With HTML, JS and Electron", TutorialZine Articles, December 16th 2015
[8] Paul B. Jensen, "Cross-Platform Desktop Applications Using Node, Electron, and NW.js", Chapter 1,Introducing Electron and NW.js, Page 17-Page 23, Manning Publications, May 2017
[9] Chris Griffith and Leif Wells, "Electron: From Beginner to Pro-Learn to Build Cross Platform Desktop Applications using GitHub's Electron", Chapter 1: Welcome to Electron, Page 1-Page 7,Apress Publications,2017
[10] MuhammedJasim ,**"**Building Cross-Platform Desktop Applications with Electron**"**,Packt>Online Publications, Chapter 1 : Introducing Electron,April 2017