Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)

Volume 7- Issue 1, January 2024
Paper : 94

Development of Standalone Application using Electron Framework

Anupam Kushwaha [1], Jayashimha S R[2]
[1,2]Master of Computer Application
[1,2]R V College of Engineering
Bengaluru,India
[1] anupamk.mca21@rvce.edu.in, [2] jayasimhasr@rvce.edu.in

***Abstract*—This article discusses the benefits, problems, and frameworks involved in the development of independent desktop applications using web technology. It emphasises the use of web technologies like HTML, CSS, and JavaScript to construct standalone apps that operate natively on desktop operating systems. Desktop apps have their own advantages that tend to overpower existing websites depends on the web browser acting as the client interface. There are several browsers and they differ from each other. They too constant updating, as a result of which one browser exists in several versions with different functions and capabilities. Sometimes certain web applications are not allowed to run in a particular web browser due to a lack of compatibility or a particular version. There are many browsers available, each with different versions and features, compatibility issues and limitations that prevent certain web applications from running smoothly. To overcome this challenge, developers often rely on the Electron Framework, a widely used solution for building desktop applications. Using web technologies such as HTML, CSS and JavaScript, Electron enables developers to create standalone desktop applications that can be installed and run consistently across all operating systems, providing users with a reliable and independent user experience without browser limitation.**

Keywords—Browser, Java-script, HTML, CSS, Electron, JSON, NPM, Node.

I. INTRODUCTION

Desktop applications initially gained considerable attention and became widely popular among users due to their ease of use and versatility in both personal and professional environments. However, with the spread of the Internet and the subsequent proliferation of Internet applications, the focus shifted to browser-based software solutions. This change raised concerns about the future of desktop apps, but they managed to survive and remain relevant. The fundamental differences between desktop and web apps affect their strengths and use cases [1]. Desktop applications offer advantages such as superior performance, direct access to system resources and the ability to work offline. They are particularly suitable for resource-intensive tasks, complex operations and applications that require a high level of protection. On the other hand, web applications stand out due to accessibility, cross-platform compatibility and centralized updates[2]. They can be accessed from any device using a web browser and are ideal for scenarios where usability and collaboration across multiple platforms are critical. The coexistence of both desktop and web applications allows users to choose a suitable solution according to their requirements and preferences.

A. Desktop Based Application

Desktop applications, also known as native applications, are software designed to be installed and run directly on a user's computer or desktop environment. Unlike web applications running in a web browser, desktop applications have direct access to system resources and can use all the capabilities of the local machine [3]-[6]. This allows desktop applications to offer better performance, responsiveness and offline functionality. They can interact with hardware devices, use local storage, and use advanced features provided by the operating system. Desktop applications also offer a higher level of control and security because they run in the user's local environment, reducing the risk of data breaches and unauthorized access. While web applications offer usability and cross-platform compatibility, desktop applications are recommended for resource-intensive tasks, complex operations and scenarios where offline access and local processing power are critical.

B. Web Based Application

Web applications have gained widespread popularity due to their accessibility and versatility. Unlike desktop applications, web applications can be accessed through web browsers on various devices without installation. They are created using web technologies such as HTML, CSS and JavaScript, so they are cross-platform compatible. Web applications provide centralized deployment and updates, where server-side changes are immediately visible to all users [5]. They promote collaboration and data sharing by allowing multiple users to access and interact with an application simultaneously. With proper security measures, web applications can ensure data encryption, user authentication and protection against vulnerabilities[7]. They are scalable and cost-effective because they can meet growing user demands without significant infrastructure investment. In general, web applications provide a flexible and easy-to-use solution for a wide range of user and business needs.

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7- Issue 1, January 2024
Paper : 94

II. Desktop Vs Web Apps

Developing standalone desktop applications using web technologies has emerged as an innovative approach that allows developers to use their existing web development skills and tools to create powerful and versatile desktop applications[8]. This approach takes web technologies such as HTML, CSS and JavaScript that have traditionally been used to create web applications and extends their capabilities to desktop environments. By adopting web technologies for desktop application development, developers can benefit from code reusability, as a significant portion of the code base can be shared between the web and desktop versions of the application. In addition, web technologies provide platform independence, which allows applications to run seamlessly on different desktop operating systems, eliminating the need for separate code bases for different platforms. Developing stand-alone desktop applications using web technologies offers the opportunity to streamline the development process and maximize efficiency by using familiar web tools and workflows[9]-[11]. However, challenges such as optimizing performance for desktop use and using native resources must be considered, as web technologies are primarily designed for web-based environments. To address these challenges, frameworks and tools such as Electron, NW.js, and React Native have been developed that allow developers to package web applications as standalone executables and provide access to native APIs and functionality. This framework allows developers to create desktop experiences leveraging the flexibility and versatility of web technologies. Real-world examples like popular apps like Slack and Visual Studio Code demonstrate the success and potential of web-based desktop apps.

III. Electron Framework

The Electron Framework is a popular open-source framework that enables developers to create cross-platform and stand-alone desktop apps with web technologies including HTML, CSS, and JavaScript. It was created by GitHub and has grown in popularity among developers because to its versatility and ease of use[12].

One of the key advantages of the Electron Framework is its ability to create applications that can run on different and multiple operating systems, including Windows, macOS, and Linux. The application code is packaged with a version of the Chromium browser that serves as a runtime environment to provide cross-platform compatibility.

Developers may construct desktop applications by using their existing web development expertise by utilizing web technology[14]. They can create the user interface with HTML structure, CSS style, and JavaScript interactivity. Developers may use a broad range of web development tools, libraries, and frameworks within the Electron environment, allowing for speedy development and easy prototyping.

The Electron Framework has sophisticated features and APIs that allow developers to use native operating system capabilities. Accessing the file system, manipulating windows and menus, dealing with hardware devices, and using native notifications are all examples of system-level interactions[17]. These features enable developers to design feature-rich desktop apps.

In Electron Framework there are two important processes. They are different from each other and they are as follows:

- The Main Process
- The Renderer Process

The primary process in the Electron Framework is in charge of operating the package. The primary script for JSON. The script that runs in the main process will be able to display a GUI by creating a number of web pages. The main process is always one and never more than one. There can be numerous renderer processes[10]. Each browser window runs a renderer process, resulting in a large number of renderer processes. The renderer process often renders the application's UI in the window.

The rendering process loads web pages to display a graphical user interface. Each process takes advantage of Chromium's multi-process architecture and runs in its own thread. Electron also includes the ability to facilitate inter-process communication so that the rendering process can communicate with the main process when needs.

Management of all web pages and their equivalents rendering processes are managed by the main process. The renderer processes are isolated from each other and are just care about the site using it.

The basic file structure is as follows:

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

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

main.js creates windows and handles system events.

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

render.js handles the application’s render processes

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7- Issue 1, January 2024
Paper : 94

Packaging and distribution are important considerations with the desktop application development process. Because a general purpose desktop application development framework, The electron must allow easy access to the package and distribution of applications to different platforms. Project known as Electron-Packager was developed Electron Community, which deals with this. The right action must be available for users to install it desktop application for your machines. It is taken managed by Electron Packager. Electronic packaging application simply means creating a desktop installer required operating system. It recognizes the platform system and integrate the application accordingly the platform thus creates an operating system-specific desktop application.

A. Electron Architecture

Electron combines Chromium and Node.js into a single runtime to provide cross-platform desktop applications. Electron can be thought of as a minimal browser that has the ability to interact with the operating system, and this browser is part of the application. With Electron developers can forget about any OS browser compatibility issues[14]. They can be sure that everyone using the app has the same version of Chrome and the same version of Node.js, regardless of the user's computer.

B. Node.js

Node.js is a JavaScript platform built on Google Chrome's JavaScript V8 Motor. It provides a runtime environment for server-side development building applications that use JavaScript and APIs to access the file system web server and download the code from the external module[13]. Node is open source and is used by thousands of developers around the world in the world Node.js allows developers to share and update code by name and use over 250,000 NPM packages. Node.js can be seen as pure web application framework, but the truth is that Node.js can be used for the desktop also applications. Electron is an example of frameworks using Node.js create cross-platform desktop applications.

C. Chromium browser

The Chromium browser is an opensource version of Google's Chrome browser. They share most of the code and features, but have some differences in features and different licenses. Chromium renders web pages as an independent process, loads CSS styles and executes JavaScript codes.

D. Electron’s multi-process architecture

Electron applications inherit Chromium's multi-process model. An electron applications mainly consist of two types of processes: the main process and zero or more image processes. Each process has a different role application The main process is responsible for creating and controlling it application life cycle[19]. He is also responsible for dealing with indigenous peoples operating system APIs[18]. The rendering process loads web pages to display a graphical user interface. Each process takes advantage of Chromium's multi-process architecture and runs in its own thread. It also contains an electron the ability to facilitate communication between processes so that imaging process to communicate with the main process in case they needs.

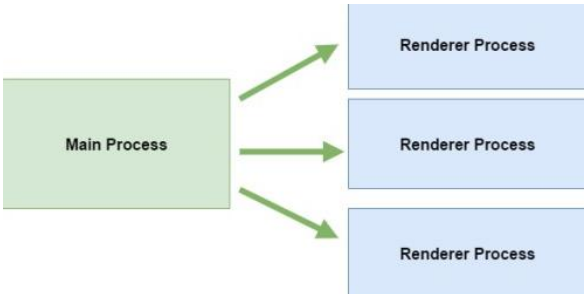

Figure 1: Electron’s multi-process architecture

E. The Programming language of Electron

As mentioned earlier, Electron uses web technologies JavaScript, HTML5 and CSS for developing desktop applications. These technologies are the basis when building web pages. Electron uses web pages for creating the graphical user interface of the app[16]. The structure of a page is created using HTML5, while the visual layout is made using CSS. JavaScript, which is a client-side programming language, can be used together with these technologies to make a web page dynamic.

F. Electron APIs and Features

In addition to the rich APIs of Node.js and HTML5, Electron has come up with a useful set of APIs and features for building desktop applications:

- Create application windows, each with their own JavaScript context.
- Desktop integration through the shell and screen APIs.
- Tracking the power status of the computer.
- Monitors the power state change.
- Creating tray applications.
- Copying and pasting from clipboard.
- Creating menus and menu items.
- Adding global keyboard shortcuts to the application.

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7- Issue 1, January 2024
Paper : 94

- Updating the application's code automatically through app updates.
- Crash reporting for when the application crashes.
- Customizing Dock menu item.
- Operating System Notifications.
- Creating Window Installers.
- Debugging and profiling.
- Showing native system dialog.

Electron offers many features, and this is not a complete list of the features available in the framework. in in particular, the collision reporting feature is unique to Electron[20]. Besides that Electron provides special tools for testing and debugging applications, called Spectron and Devtron.

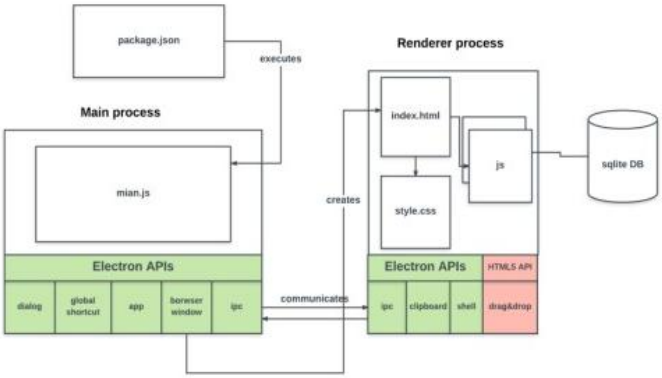

Figure 2: Electron application architecture

TABLE – I LITERATURE SURVEY

Sr. No.	Author and Paper title	Parameters	Summary of the Paper
1.	Responsive Webpage Using HTML CSS[5]. Authors: Gurinder Singh; Tarun Parashar	INSPEC Accession Number: 22478878 DOI: 10.1109/ICCR56254.2022.9995922 Publisher: IEEE Published in: 2022 International Conference on Cyber Resilience (ICCR)	Developing a webpage or website that is compatible with various operating systems, their versions, and different device screen sizes can be challenging. However, this paper provides a comprehensive understanding of essential concepts for creating a universally compatible webpage/website.
2.	Covid-19 Data Visualization and Data Analytics with a Smart Standalone Mobile Application[4]. Authors: Abhijit Poddar; Monali Poddar	INSPEC Accession Number: 20422217 DOI: 10.1109/INDICON49873.2020.9342143 Publisher: IEEE Published in: 2020 IEEE 17th India Council International Conference (INDICON)	In summary, it enables users to study and analyze data using engaging visualizations and user-friendly techniques. Moreover, the application is designed with an intuitive interface. Additionally, its open-ended nature allows for the swift introduction of new models to handle newly available Covid-19 data.
3.	Lyapunov Energy Function based Control of a Soft Switching Solid State Transformer for Three-phase Standalone Application[3]. Authors: Vikram Roy Chowdhury; Rajendra Prasad Kandula; Deepak Divan	INSPEC Accession Number: 20134834 DOI: 10.1109/ECCE44975.2020.9236395 Publisher: IEEE Published in: 2020 IEEE Energy Conversion Congress and Exposition (ECCE)	In summary, Control and operation of a three-phase solid state soft switching transformer based on Lyapunov energy function for three-phase standalone application has been presented in this paper. Soft switching solid state transformer is based on a current source inverter type of topology with a high frequency transformer, therefore, making the

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7- Issue 1, January 2024
Paper : 94

			converter immensely challenging to control and operate optimally.
4.	Detecting Malicious Behaviors in JavaScript Applications[2]. Authors: Jian Mao; Jingdong Bian; Guangdong Bai; Ruilong Wang; Yue Chen; Yinhao Xiao; Zhenkai Liang	INSPEC Accession Number: 17649136 DOI: 10.1109/ACCESS.2018.2795383 Publisher: IEEE Published in: IEEE Access (Volume: 6)	This paper presents a proposal for identifying malicious JavaScript behaviors in JavaScript applications. The proposed approach involves creating a behavior model that encompasses both the application's behaviors and function-level execution details. A prototype detection system is developed to automatically construct behavior models for hybrid apps, enabling the detection of anomalous behaviors.
5.	Performance and stability Comparison of React and Flutter: Cross-platform Application Development[1]. Authors: Kamal Kishore; Shanu Khare; Vaibhav Uniyal; Sahil Verma	INSPEC Accession Number: 22478851 DOI: 10.1109/ICCR56254.2022.9996039 Publisher: IEEE Published in: 2022 International Conference on Cyber Resilience (ICCR)	This research paper presents the study and comparison of the two most famous cross-platform application development technologies. It starts by discussing the basic functions of the application development methodologies. Followed by, it contains a comparison of the performance of the applications developed by these technologies.

V. Implementation

Basic steps of Electron App methodology the development is as follows:

- Install Node.js and NPM.
- Install a code editor like Visual Studio code, Atom, Sublime Text.
- Initiate a node file to the desired directory. By using the following command e.g. npm init -y
- Install the Electron package with the command e.g. npm install electron –save-dev.
- Create a new JavaScript file (e.g., main.js).
- Edit the main.js file and add application logic like
  const electron = require('electron') // load the electron module from NPM

  const app = electron.app // Module to control application life.

  const BrowserWindow = electron.BrowserWindow // Module to create native browser window.

  let mainWindow function createWindow () {
  // Create the browser window.

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)

Volume 7- Issue 1, January 2024
Paper : 94

```
mainWindow = new BrowserWindow({width: 950, height: 700}) // and load the index.html of the app.

mainWindow.loadURL(`file://${__dirname}/renderer/index.html`) // This method will be called when Electron has finished
// initialization and is ready to create browser windows.

app.on('ready', createWindow) // Emitted when the window is closed.

mainWindow.on('closed',  function(){
mainWindow = null })}
```

- Create an HTML file (e.g., index.html).

- Design and code the user interface of your application within the index.html file.
- Edit the main.js file and configure it to load index.html file as the main window of the application.
- Build and run the Electron application. By the following command (e.g. npm start).
- For packaging use a tool like “electron-builder” or “electron-packager” to package application for distribution on different platforms (Windows, macOS, Linux).

VI. Result and Analysis

The web application adapted to the Electronic Framework was successful works like a desktop application without using the web Browser. However, a desktop based on the Electron Framework applications use Chromium internally for HTML and CSS.

Electron Packager was able to detect the host system initialize and run the desktop application on different platforms such as Windows, Mac and Linux. The Electron Desktop application executes the same way feel, look and consistency across platforms.

The troubleshooting was done to ensure compatibility of Desktop application on different platforms and no problems found as a result of program testing on different platforms.

Some unfavorable characteristics of the Electron Desktop Applications might be detected.

The loading time of the desktop program is somewhat longer than that of the web application when launched through a web browser. This is due to the fact that HTML and CSS will be displayed and JavaScript will be performed by NodeJs. JavaScript is available as modules, which are all saved in the npm registry. The JavaScript modules that must be utilised are listed in the "package.json" file. Because these modules must be looked for and retrieved from the npm registry, the loading time for Electron Desktop Applications is significantly longer.
Electron apps are simply a fully-featured Chromium browser and a Node process that interact over IPC.In terms of memory, packaged Electron apps are often rather huge. More and more web pages may be translated to the Electron Framework and hosted as a single Desktop Application. However, as the number of web pages converted to Electron Framework increases, memory consumption increases steadily after the Electron-Packaging step.

When operated for an extended period of time, electron apps frequently require a substantial number of system resources as well as a significant quantity of battery power.

VII. Conclusions and Future Work

In conclusion, the development of stand-alone desktop applications using electron framework has changed the world of software development. Using web technologies such as HTML, CSS, and JavaScript, developers can create stand-alone applications that run natively on desktop operating systems. This approach offers several advantages, such as code reusability, platform independence, and a familiar development environment. However, the challenges of optimizing performance and using native resources have been solved by frameworks like Electron, allowing developers to package web applications as standalone executables and use native APIs. Real examples have proven the success and versatility of web-based desktop applications, while the constant development of web technologies continues to improve their capabilities. As the lines between web and native applications blur, the development of standalone desktop applications using electron framework represents a promising future for creating robust and versatile applications across multiple platforms.

Electron Framework effectively hosts an existing online application in the form of a cross-platform desktop application, providing a consistent feel, appearance, and consistency across multiple platforms after electron packaging.

Explored some crucial features of cross-platform and standalone desktop development using the Electron framework in this thesis. However, there are numerous

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7- Issue 1, January 2024
Paper : 94

avenues for further investigation. Other elements of both frameworks can be tested because they have additional features that we did not use owing to time constraints in this thesis. Other features worth investigating are:

- Native features like Notifications, trays, menus, etc.
- Multithreading.
- Security.
- Integration with different frameworks like react.js, next.js etc
- Packaging and distribution.

REFERENCES

[1] Kamal Kishore; Shanu Khare; Vaibhav Uniyal; Sahil Verma, " Performance and stability Comparison of React and Flutter: Cross-platform Application Development," Published in: 2022 International Conference on Cyber Resilience (ICCR).

[2] Jian Mao; Jingdong Bian; Guangdong Bai; Ruilong Wang; Yue Chen; Yinhao Xiao; Zhenkai Liang, "Detecting Malicious Behaviors in JavaScript Applications", Published in: IEEE Access (Volume: 6).

[3] Vikram Roy Chowdhury; Rajendra Prasad Kandula; Deepak Divan ,"Lyapunov Energy Function based Control of a Soft Switching Solid State Transformer for Three-phase Standalone Application", Published in: 2020 IEEE Energy Conversion Congress and Exposition (ECCE).

[4] Abhijit Poddar; Monali Poddar,"Covid-19 Data Visualization and Data Analytics with a Smart Standalone Mobile Application ", Published in: 2020 IEEE 17th India Council International Conference (INDICON).

[5] Gurinder Singh; Tarun Parashar ,"Responsive Webpage Using HTML CSS ", Published in: 2022 International Conference on Cyber Resilience (ICCR)

[6] 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

[7] 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

[8] Manuel P, Inderjeet Singh, Antonio Cicchetti, "Comparison of Cross-Platform Mobile Development Tools",16th International Conference on Intelligence in Next Generation Networks,2016

[9] Kitti Kredpattanakul, Yachai Limpiyakorn, "Transforming JavaScript-Based Web Application to Cross-Platform Desktop with Electron": ICISA 2018

[10] Lalit Garg , Keith Vassallo, Cross-Platform Development Frameworks: Overview of contemporary technologies and methods for cross-platform application development. 2nd International Conference on Computers & Management (ICCM 2016).

[11] Xanthopoulos S, Xinogalos S. "A comparative analysis of cross-platform development approaches for mobile applications", in 6th Balkan Conference in Informatics, 2013, Thessaloniki,Greece.

[12] A. Goldenberg, "Pros and cons of developing native vs. Cross-platform web-based mobile application," 2002. [Online]. Available: https://www.dbbest.com/blog/pros-and-cons-of-developing-native-vs-crossplatform-web-based-mobile-application/.

[13] Raj R, Tolety SB. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In: 2012 Annual IEEE India Conference (INDICON); 2012.

[14] Manuel Palmieri, Inderjeet Singh, Antonio Cicchetti, A. Comparison of cross-platform mobile development tools. In: 2012 16th International Conference on Intelligence in Next Generation Networks (ICIN); 2012.

[15] Xanthopoulos S, Xinogalos S. "A comparative analysis of cross-platform development approaches for mobile applications", in 6th Balkan Conference in Informatics, 2013, Thessaloniki,Greece.

[16] "Electron," Electron. [Online]. Available: http://electron.atom.io/.

[17] "Create cross-platform desktop Node Apps with electron," in JavaScript, SitePoint, 2016. [Online]. Available:https://www.sitepoint.com/desktopnodeapps-with-electron/

[18] P. Jensen, Cross-platform Desktop Applications, MEAP ed. ch. 1, pp. 3- 30 [Online]. Available: https://www.manning.com/books/cross-platformdesktop-applications.

[19] A Cross-Platform Application Environment for Nomadic Desktop Computing, Stefan Paal, Reiner Kammüller & Bernd Freisleben, (LNCS,volume 3263)

[20] Embankment Protection - React Native Application Cross-Platform Application for protection of embankments by crowd sourced data, AnshulVarshav Borawake; Minal Shahakar, Published in: 2021 International Conference on Computing, Communication and Green Engineering (CCGE