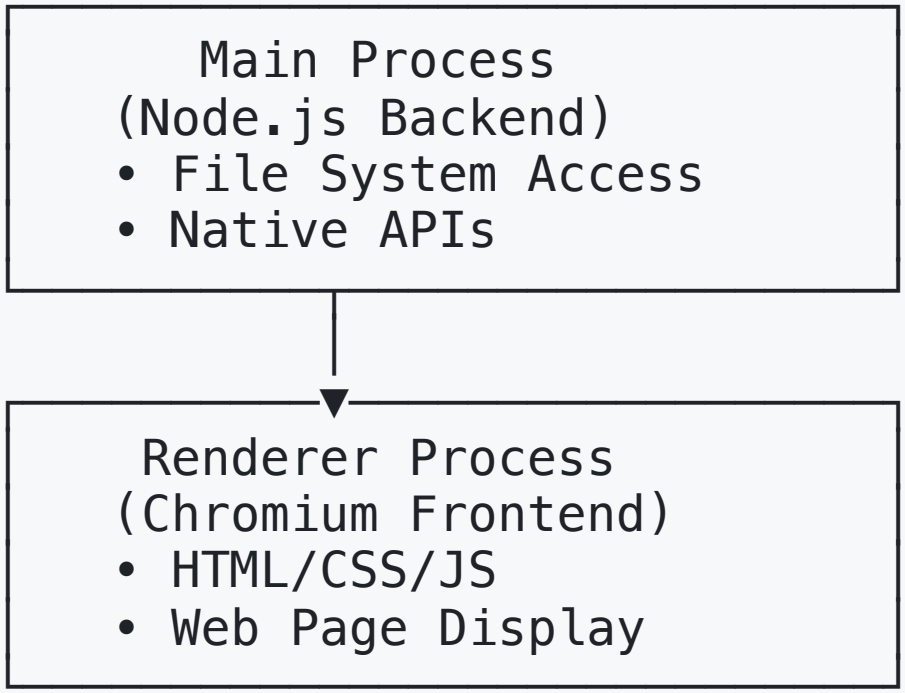
Electron Basics

Desktop Apps with Web Technologies

What is Electron?

- **Desktop application framework**
- Uses HTML, CSS, JavaScript
- Combines **Chromium** + **Node.js**
- Cross-platform (Windows, Mac, Linux)

Core Architecture



Project Structure

```
my-electron-app/
├── package.json
├── main.js           # Main process
├── index.html        # UI
└── renderer.js       # Renderer process
```

1. package.json

```
{
  "name": "my-electron-app",
  "version": "1.0.0",
  "main": "main.js",
  "scripts": {
    "start": "electron ."
  },
  "devDependencies": {
    "electron": "^28.0.0"
  }
}
```

2. main.js (Main Process)

Controls the application.

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

let mainWindow;

app.on('ready', () => {
  mainWindow = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      nodeIntegration: false,
      contextIsolation: true
    }
  });

  mainWindow.loadFile('index.html');
});
```

Electron App Lifecycle Basics"

⚙️ Key Concepts

Term	Meaning
`app`	Manages the app's lifecycle (start → quit).
`ready`	Triggered when Electron finishes setup.
`BrowserWindow`	Creates a desktop window to show web content.
`loadFile()`	Loads local HTML file into the window.

Common Events

Event	Purpose
`ready`	Create your window here.
`window-all-closed`	Quit when all windows close (except on macOS).
`activate`	Reopen window on macOS when clicked.
`before-quit`	Perform cleanup before quitting.

Using JavaScript Promise

Instead of using `app.on('ready')`, you can use:

```
app.whenReady().then(createMainWindow);
```

✅ Cleaner and modern — same behavior, easier to chain.

3. index.html

Main UI.

```
<!DOCTYPE html>
<html>
<head>
  <title>My Electron App</title>
</head>
<body>
  <h1>Hello Electron!</h1>
  <button id="btn">Click Me</button>

  <script src="renderer.js"></script>
</body>
</html>
```

4. renderer.js (Renderer Process)

It deals with the business logic + UI access.

```
// Runs in the web page (renderer process)
document.getElementById('btn').addEventListener('click', calculate);

function calculate() {
  // Business logic
  const num1 = 10;
  const num2 = 5;
  const result = num1 * num2;

  // UI access and update
  const output = document.getElementById('output');
  output.textContent = `Result: ${result}`;
  output.style.color = 'green';
}
```

Two Processes: main and Renderer

1. **Main Process**
    - Controls application lifecycle
    - Creates/manages windows
    - Access to Node.js APIs
2. **Renderer Process**
    - Displays web pages
    - Each window = separate process
    - Limited system access

Running and building the App

```
# Install dependencies
npm install

# Start the app
npm start

# Build web files only (dist/)

npm run build

# Create unpacked binary (release/ - for testing)
npm run pack

# Create distributable binary (release/ - for distribution)
npm run dist
```