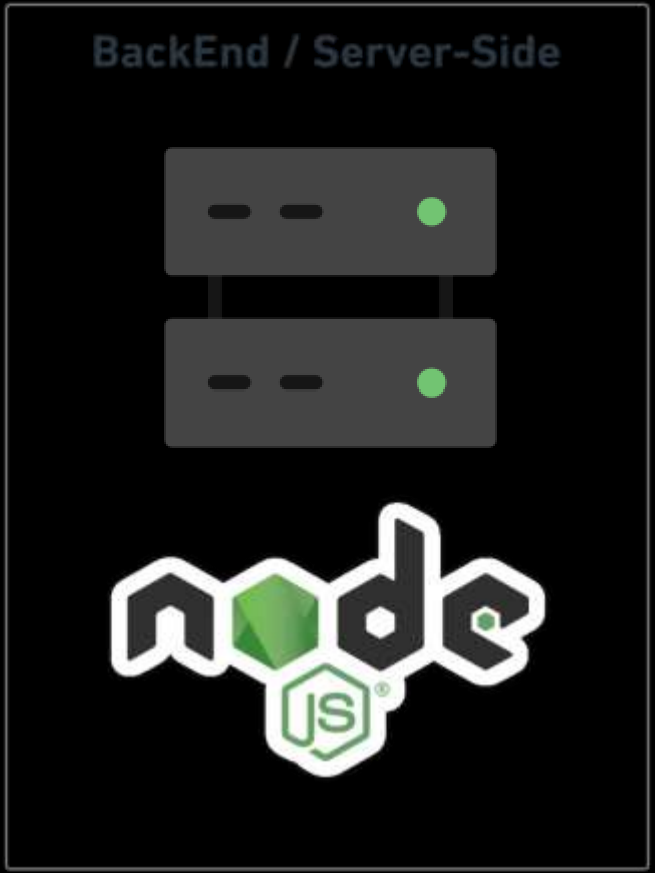
# Introduction to NodeJS

1. Pre-requisites
2. What is NodeJS
3. NodeJs Features
4. JavaScript on Client
5. JavaScript on Server
6. Client Code vs Server Code
7. Other uses of NodeJs
8. Server architecture with NodeJs

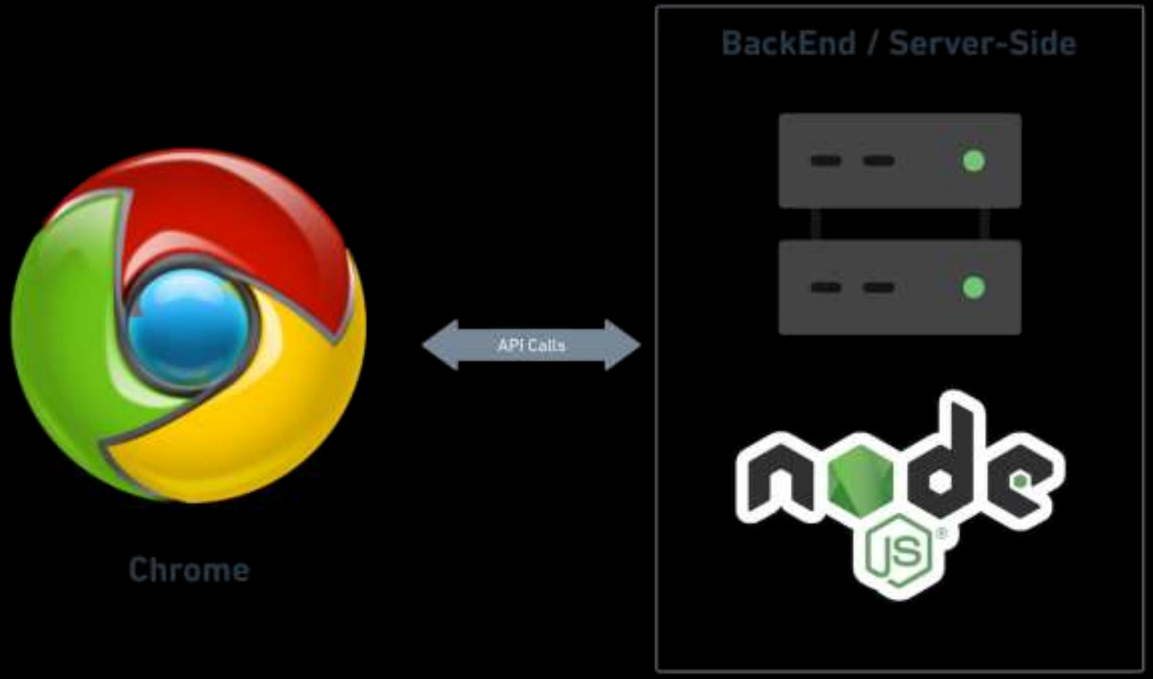


# 2. What is NodeJS

1. JavaScript Runtime: Node.js is an open-source, cross-platform runtime environment for executing JavaScript code outside of a browser.
2. NodeJs is a JavaScript in a different environment means Running JS on the server or any computer.
3. Built on Chrome's V8 Engine: It runs on the V8 engine, which compiles JavaScript directly to native machine code, enhancing performance.
4. V8 is written in C++ for speed.
5. V8 + Backend Features = NodeJs

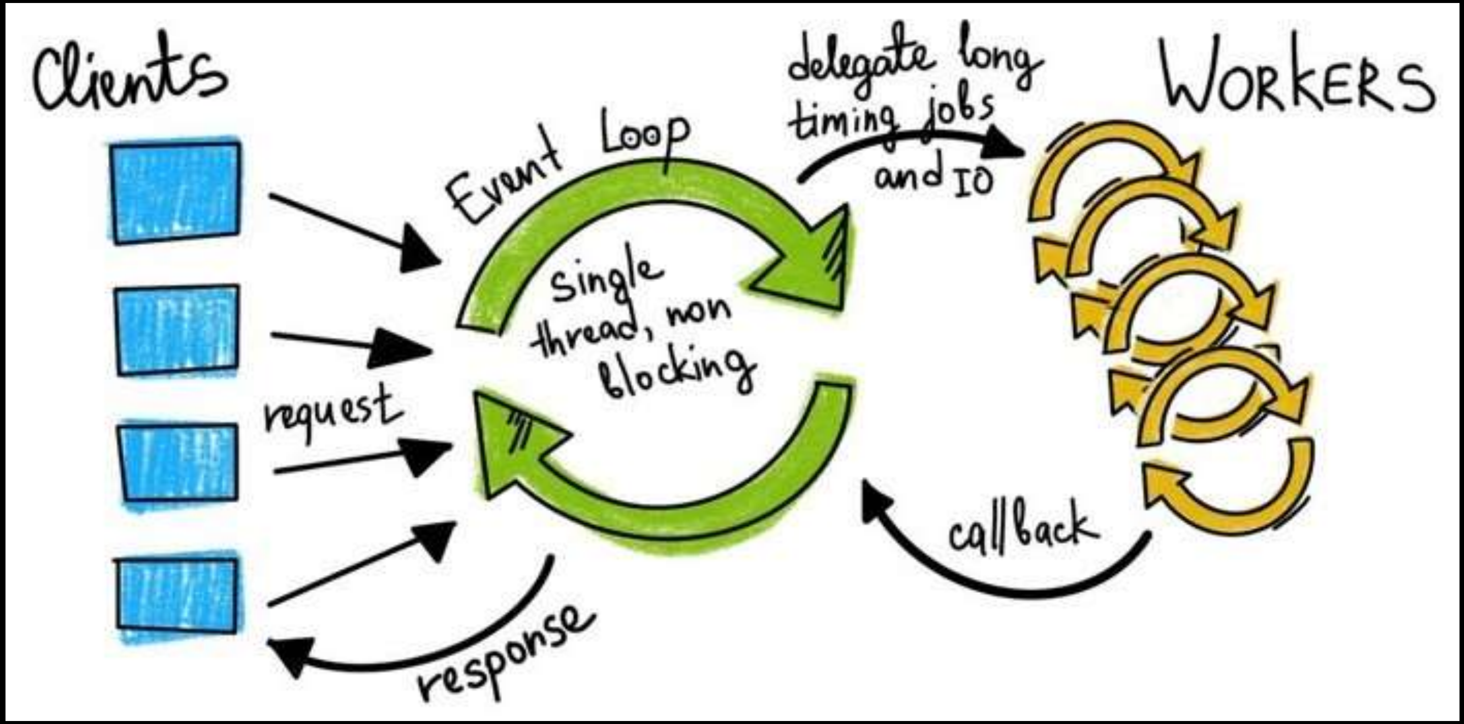
# 2. What is NodeJS

1. Design: Features an event-driven, non-blocking I/O model for efficiency.
2. Full-Stack JavaScript: Allows using JavaScript on both server and client sides.
3. Scalability: Ideal for scalable network applications due to its architecture.
4. Versatility: Suitable for web, real-time chat, and REST API servers.



# 3. NodeJs Features

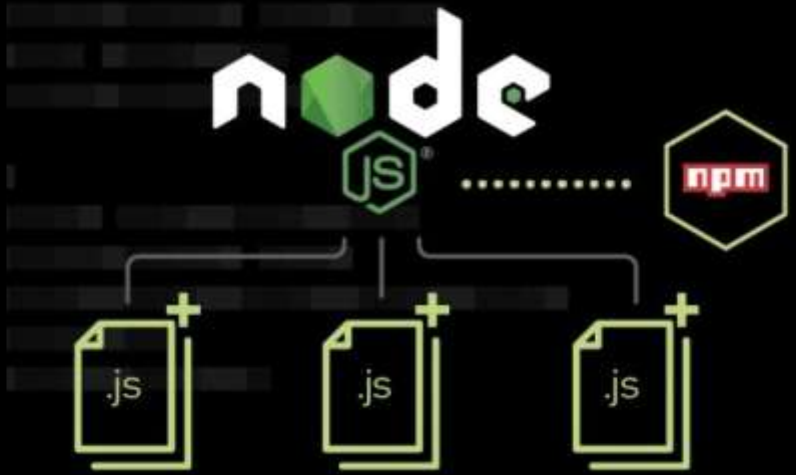
(Added)



1. Non-blocking I/O: Designed to perform non-blocking operations by default, making it suitable for I/O-heavy operations.
2. Networking Support: Supports TCP/UDP sockets, which are crucial for building lower-level network applications that browsers can't handle.

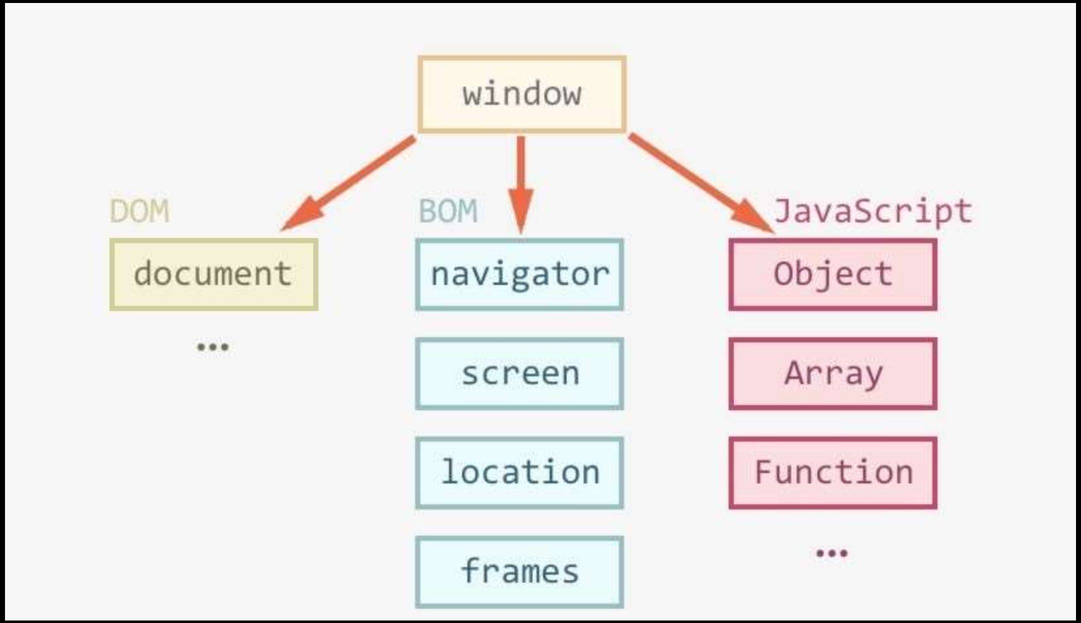
# 3. NodeJs Features

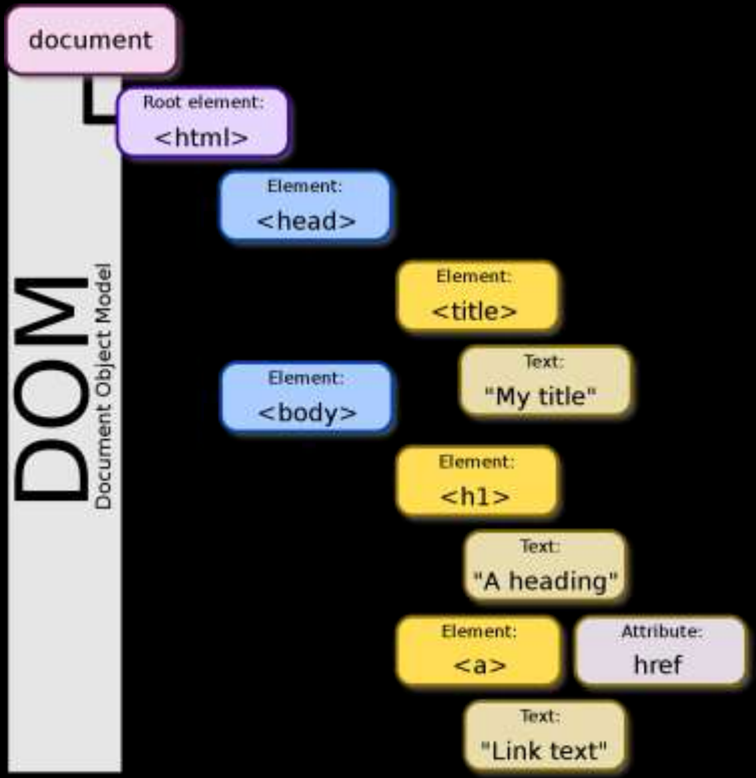
(Added)



1. File System Access: Provides APIs to read and write files directly, which is not possible in browser environments for security reasons.
2. Server-Side Capabilities: Node.js enables JavaScript to run on the server, handling HTTP requests, file operations, and other server-side functionalities.
3. Modules: Organize code into reusable modules using require().

# 3. NodeJs Features
(Removed)

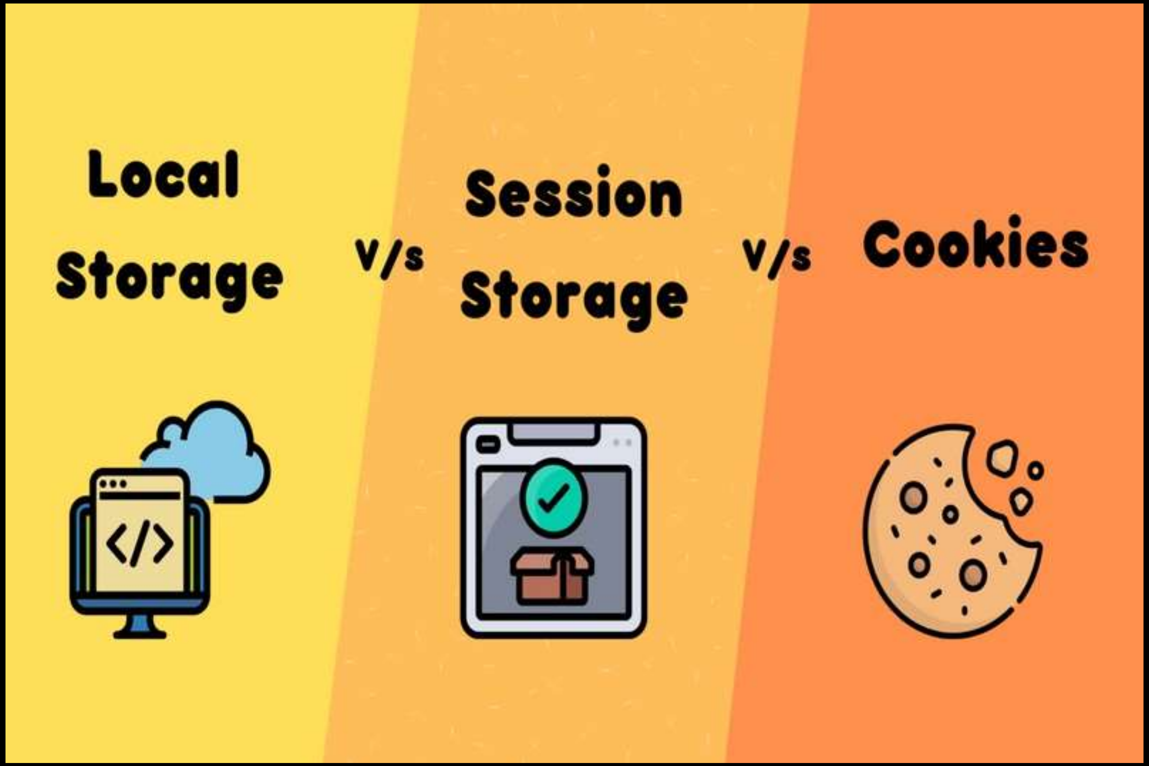


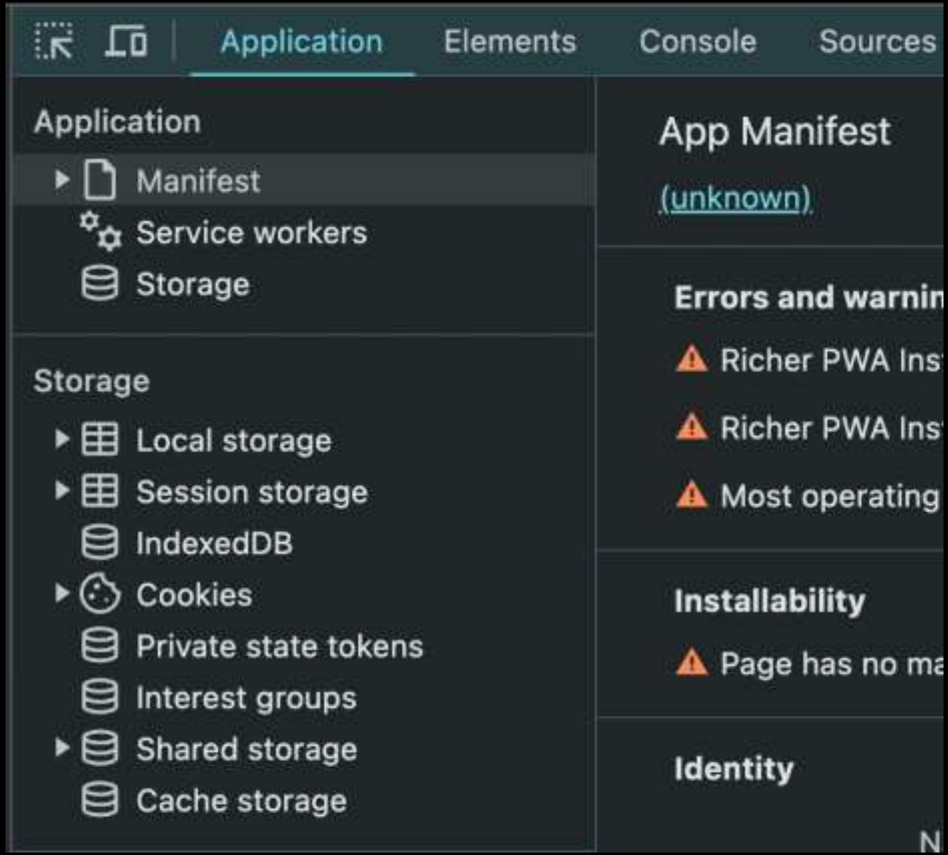


1. Window Object: The global window object, which is part of web browsers, is absent in Node.js.
2. DOM Manipulation: Node.js does not have a built-in Document Object Model (DOM), as it is not intended to interact with a webpage's content.
3. BOM (Browser Object Model): No direct interaction with things like navigator or screen which are part of BOM in browsers.

# 3. NodeJs Features

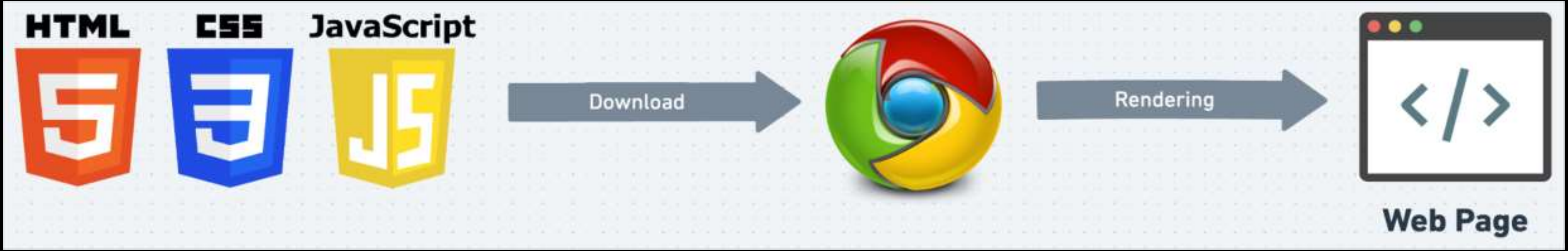
(Removed)





Web-Specific APIs: APIs like localStorage, sessionStorage, and browser-based fetch are not available in Node.js.

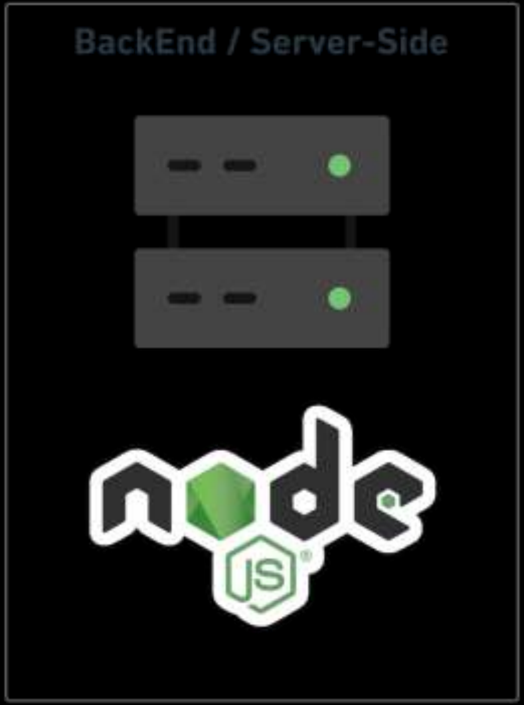
# 4. JavaScript on Client



1. Displays Web Page: Turns HTML code into what you see on screen.
2. User Clicks: Helps you interact with the web page.
3. Updates Content: Allows changes to the page using JavaScript.
4. Loads Files: Gets HTML, images, etc., from the server.

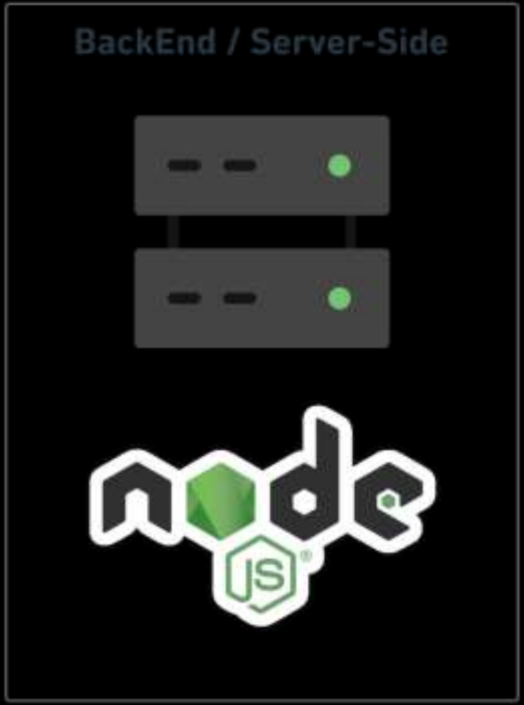
# 5. JavaScript on Server

1. Database Management: Stores, retrieves, and manages data efficiently through operations like CRUD (Create, Read, Update, Delete).
2. Authentication: Verifies user identities to control access to the system, ensuring that users are who they claim to be.
3. Authorization: Determines what authenticated users are allowed to do by managing permissions and access controls.
4. Input Validation: Checks incoming data for correctness, completeness, and security to prevent malicious data entry and errors.
5. Session Management: Tracks user activity across various requests to maintain state and manage user-specific settings.



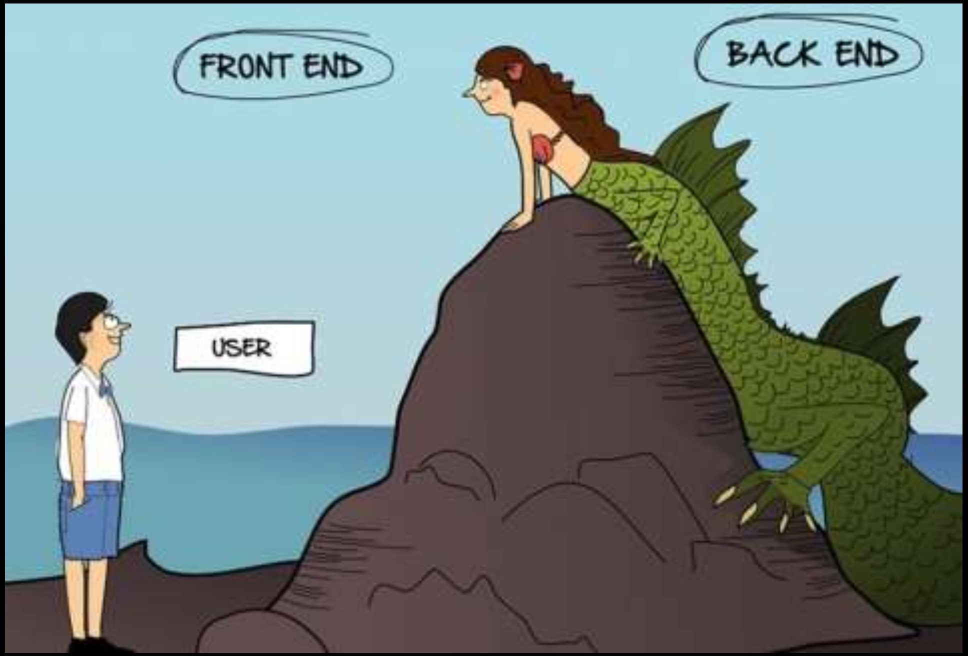
# 5. JavaScript on Server

6. API Management: Provides and handles interfaces for applications to interact, ensuring smooth data exchange and integration.
7. Error Handling: Manages and responds to errors effectively to maintain system stability and provide useful error messages.
8. Security Measures: Implements protocols to protect data from unauthorized access and attacks, such as SQL injection and cross-site scripting (XSS).
9. Data Encryption: Secures sensitive information by encrypting data stored in databases and during transmission.
10. Logging and Monitoring: Keeps records of system activity to diagnose issues and monitor system health and security.



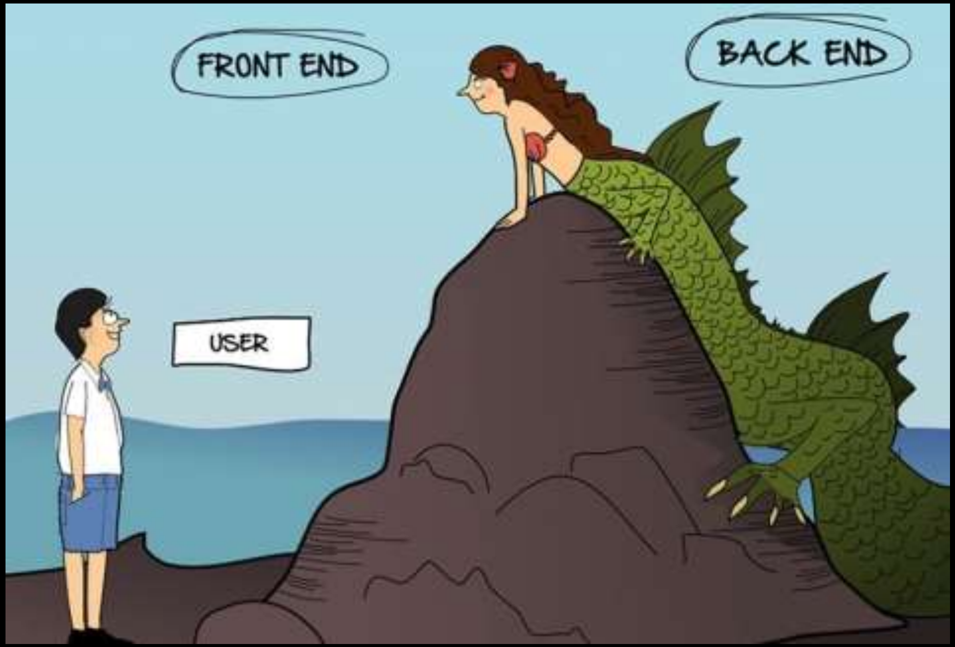
# 6. Client Code vs Server Code

1. User/client can't access server code directly.
2. Client must raise requests for particular APIs to access certain features or data.
3. Environment Access: Server-side JavaScript accesses server features like file systems and databases.
4. Security: Server-side code can handle sensitive operations securely, while client-side code is exposed and must manage security risks.
5. Performance: Heavy computations are better performed on the server to avoid slowing down the client.



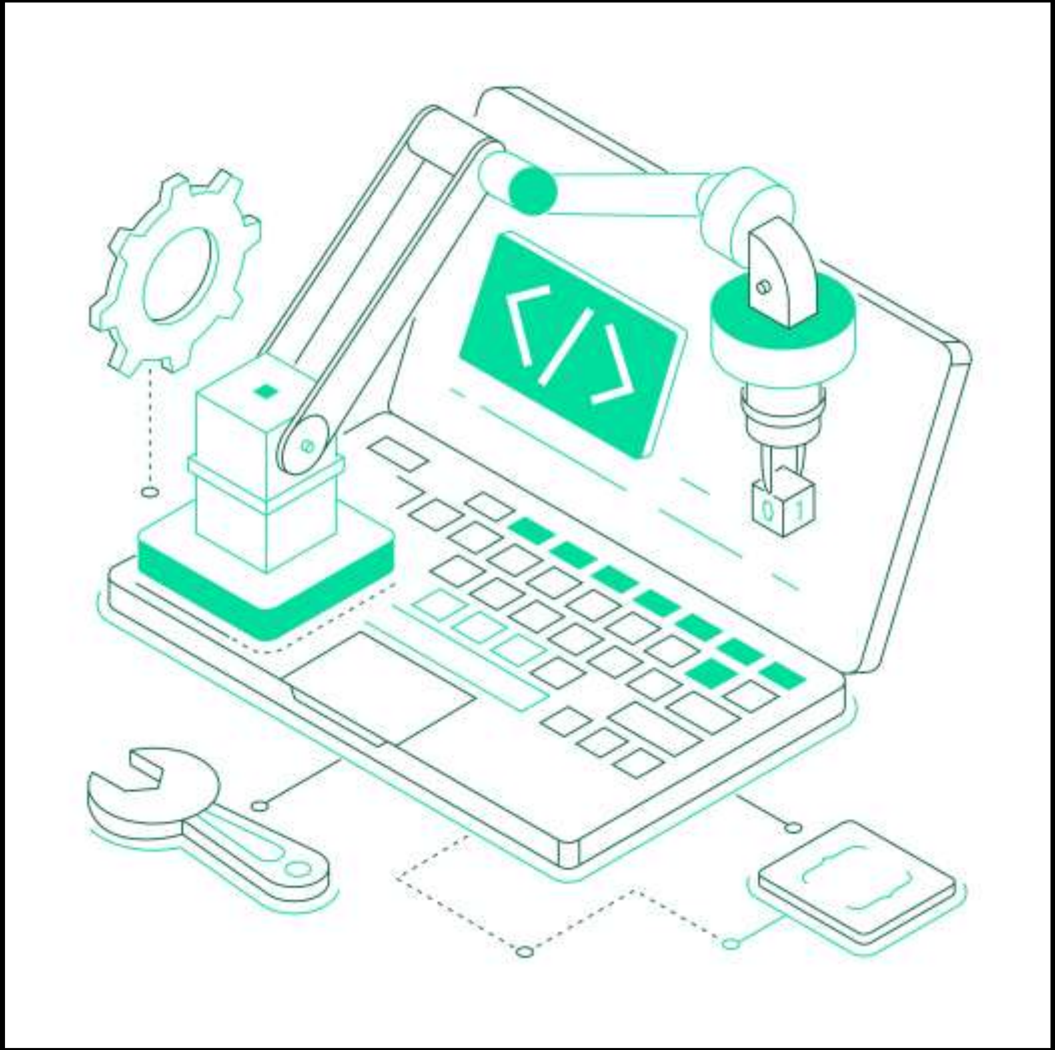
# 6. Client Code vs Server Code

6. Resource Utilization: Servers generally offer more powerful processing capabilities than client devices.
7. Data Handling: Server-side can directly manage large data sets and database interactions, unlike client-side JavaScript.
8. Asynchronous Operations: Server-side JavaScript is optimized for non-blocking I/O to efficiently manage multiple requests.
9. Session Management: Servers handle sessions and user states more comprehensively.
10. Scalability: Server-side code is designed to scale and handle requests from multiple clients simultaneously.

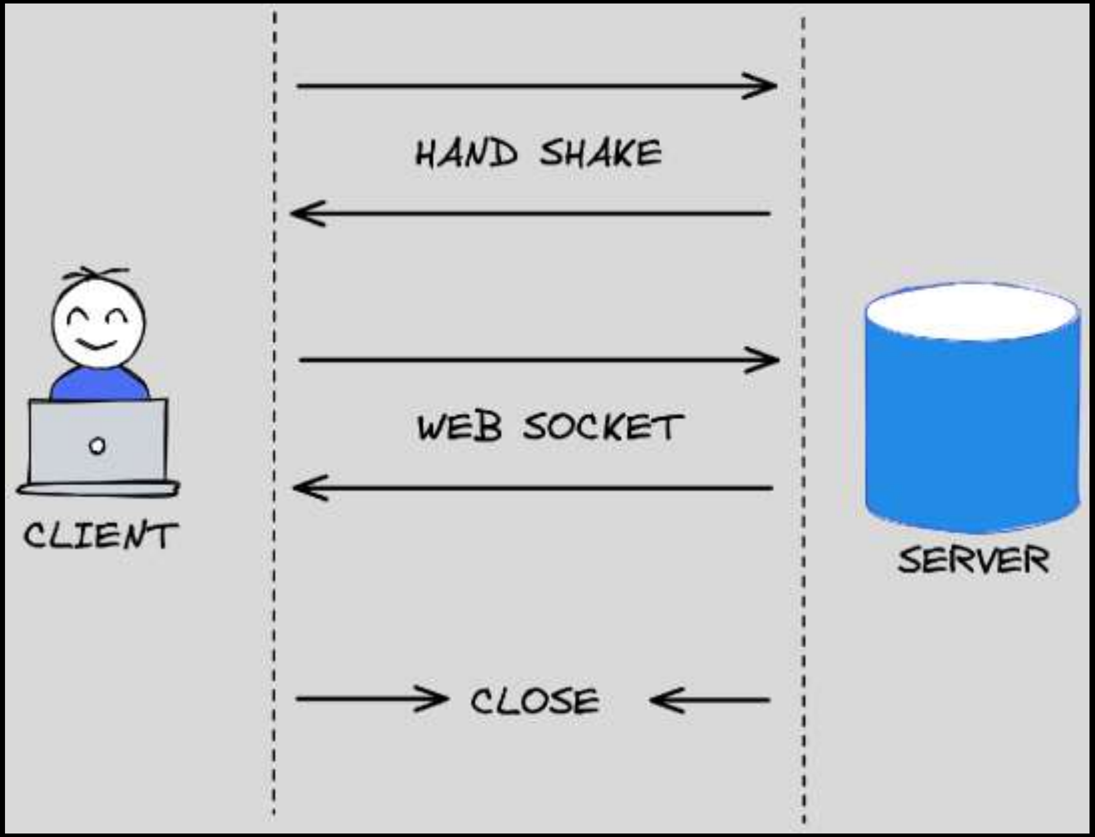


## 7. Other uses of NodeJs

1. Local Utility Scripts: Automates tasks and processes files locally, like using shell scripts but with JavaScript.
2. Internet of Things (IoT): Develops server-side applications for IoT devices, managing communications and data processing.
3. Scripting for Automation: Automates repetitive tasks in software development processes, such as testing and deployment.

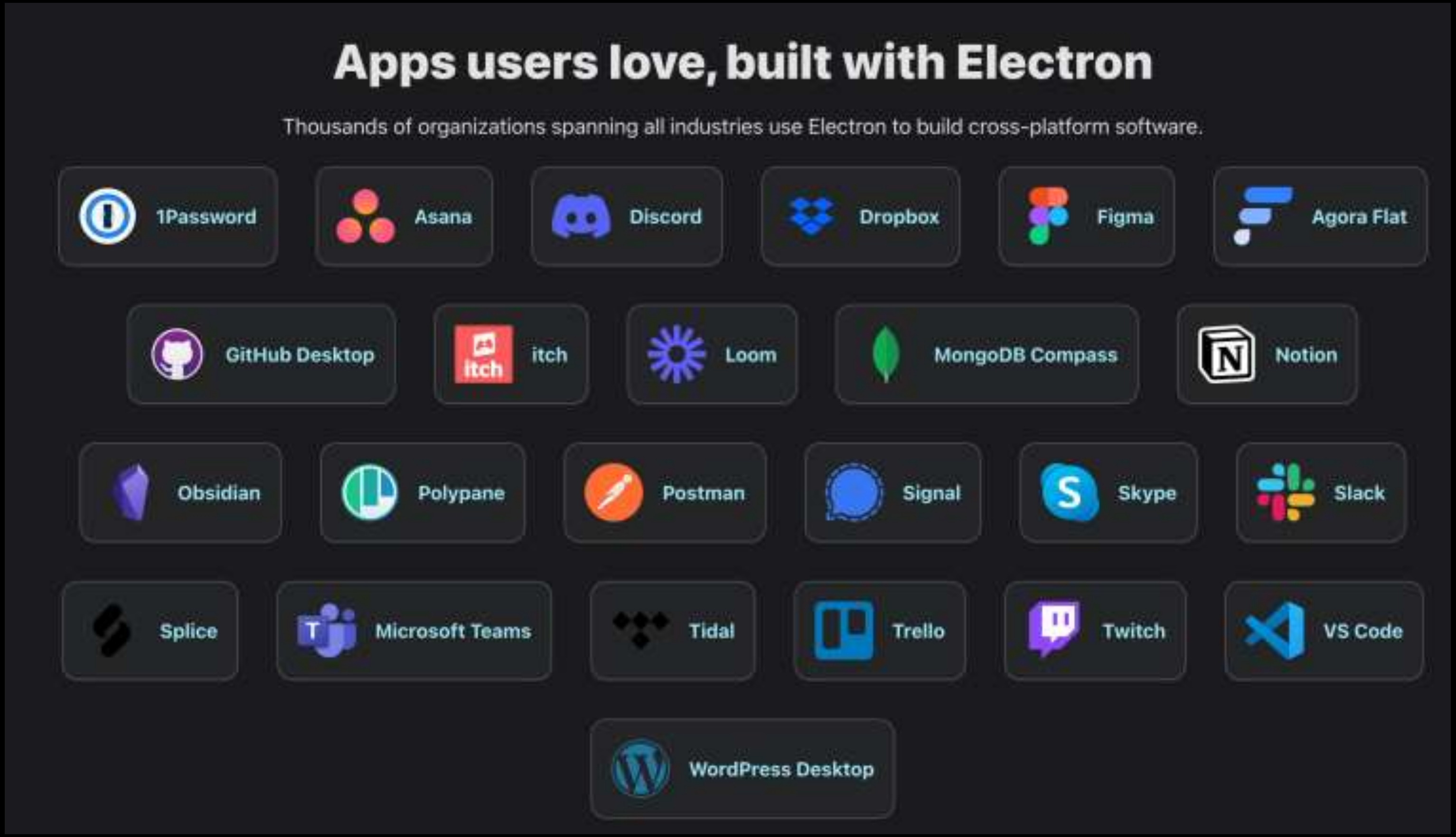


# 7. Other uses of NodeJs



Real-Time Applications: Efficiently manages real-time data applications, such as chat apps and live updates, using WebSockets.

# 7. Other uses of NodeJs



Apps users love, built with Electron

Thousands of organizations spanning all industries use Electron to build cross-platform software.

1Password · Asana · Discord · Dropbox · Figma · Agora Flat

GitHub Desktop · itch · Loom · MongoDB Compass · Notion

Obsidian · Polypane · Postman · Signal · Skype · Slack

Splice · Microsoft Teams · Tidal · Trello · Twitch · VS Code

WordPress Desktop

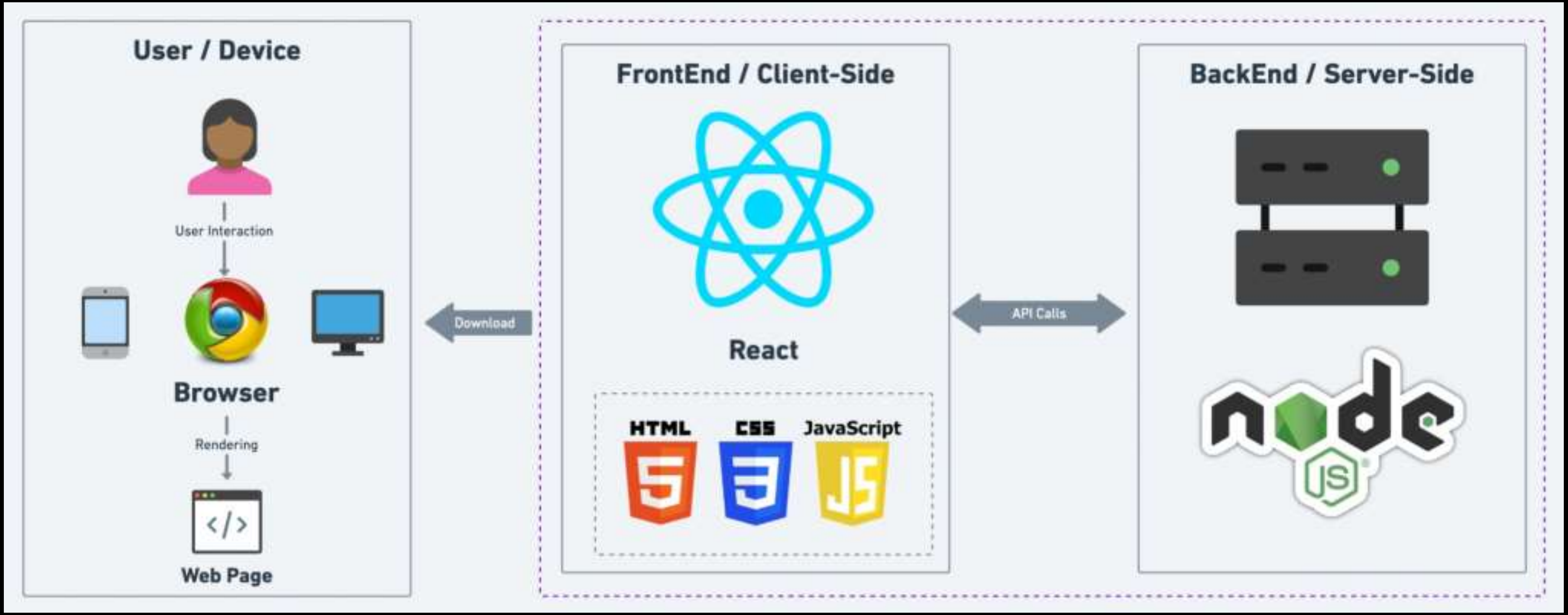
Desktop Applications: Creates cross-platform desktop applications using frameworks like Electron.

# 7. Other uses of NodeJs

Build Tools: Powers build processes for front-end technologies using tools like:

- Webpack
- Grunt
- Gulp
- Browserify
- Brunch
- Yeoman

# 8. Server architecture with NodeJs



Nodejs server will:

1. Create server and listen to incoming requests
2. Business logic: validation, connect to db, actual processing of data
3. Return response HTML, JSON, CSS, JS