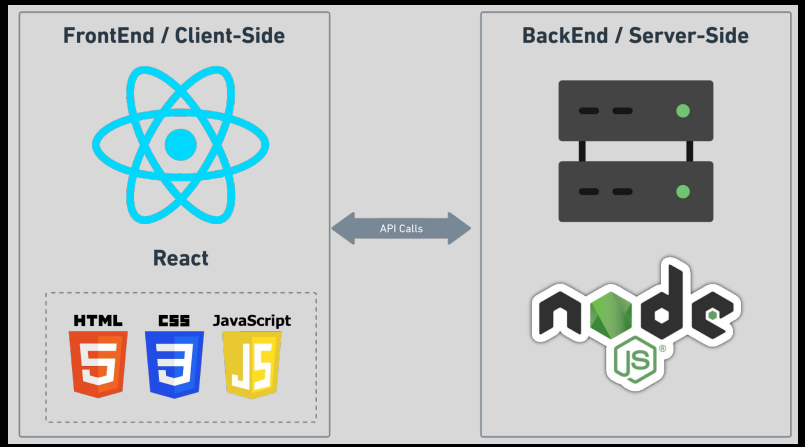
YouTube Playlist Link

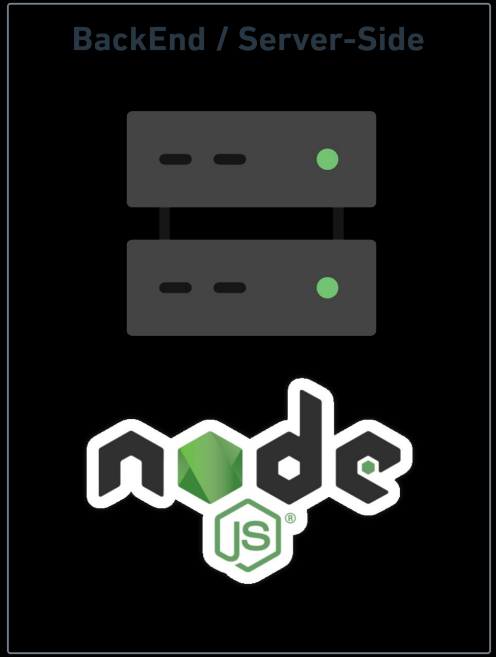
NodeJS

Complete Course



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



JS is required for NodeJS

COMPLETE JAVASCRIPT

14 HOURS

MYNTRA PROJECT

\+

CERTIFICATE + NOTES

React is recommended before NodeJS

COMPLETE

React + REDUX

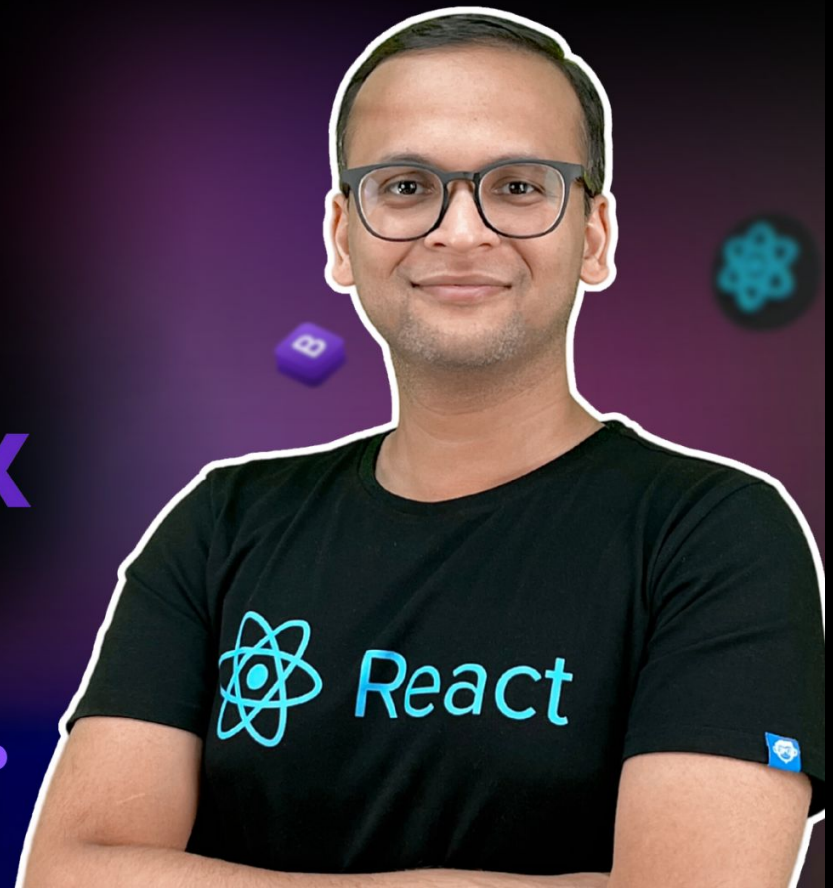
20 HOURS

6 PROJECTS

using Bootstrap

CERTIFICATE

NOTES

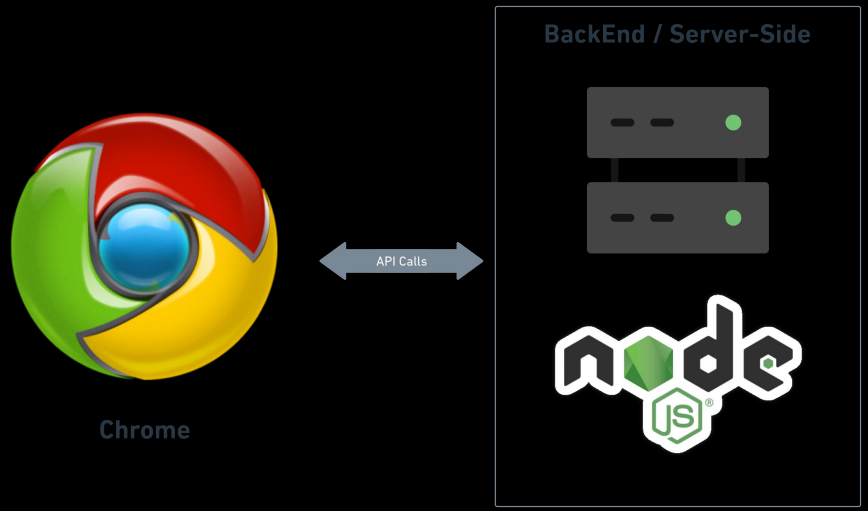


2.What is NodeJS

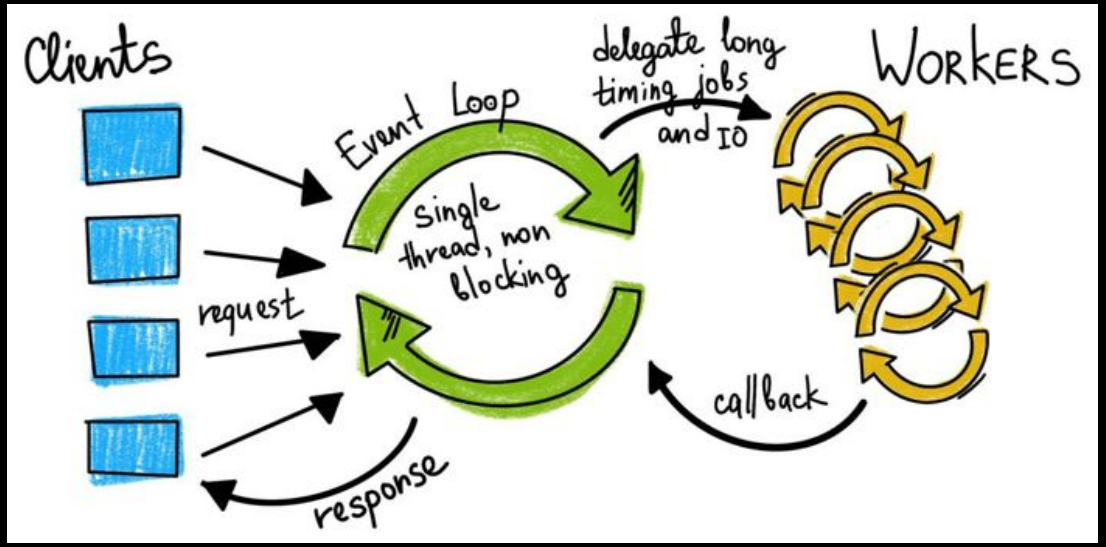
1. JavaScript Runtime: Node.js is an open-source, cross-platform runtime environment for executing JavaScript code outside of a browser.
2. NodeJs is a JavaScript in a different environment means Running JS on the server or any computer.
3. Built on Chrome's V8 Engine: It runs on the V8 engine, which 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.

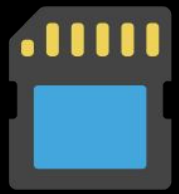


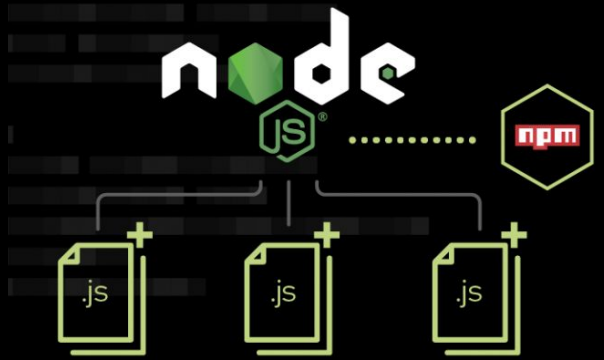
3. NodeJs Features
(Added)



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.

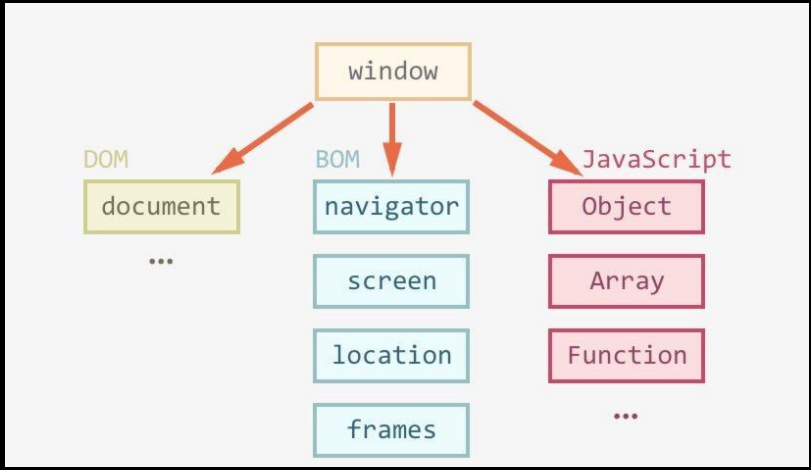
3. NodeJs Features
(Added)

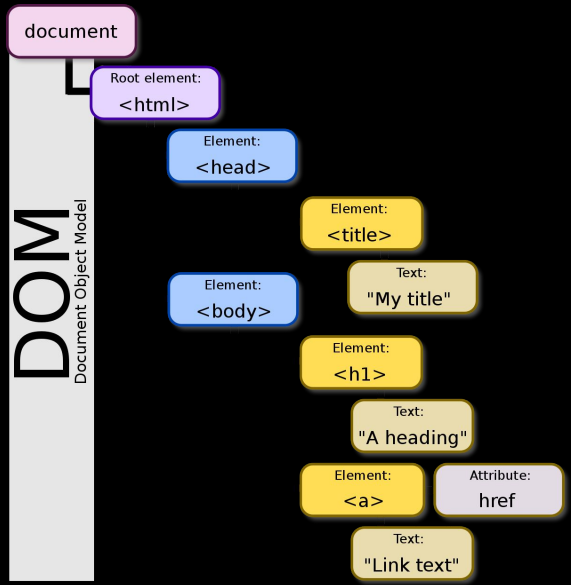




1. File System Access: Provides APIs to read and write files directly, which is not possible in browser environments for security reasons.
2. Server-Side Capabilities: Node.js enables JavaScript to run on the server, handling HTTP requests, file operations, and other server-side functionalities.
3. Modules: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

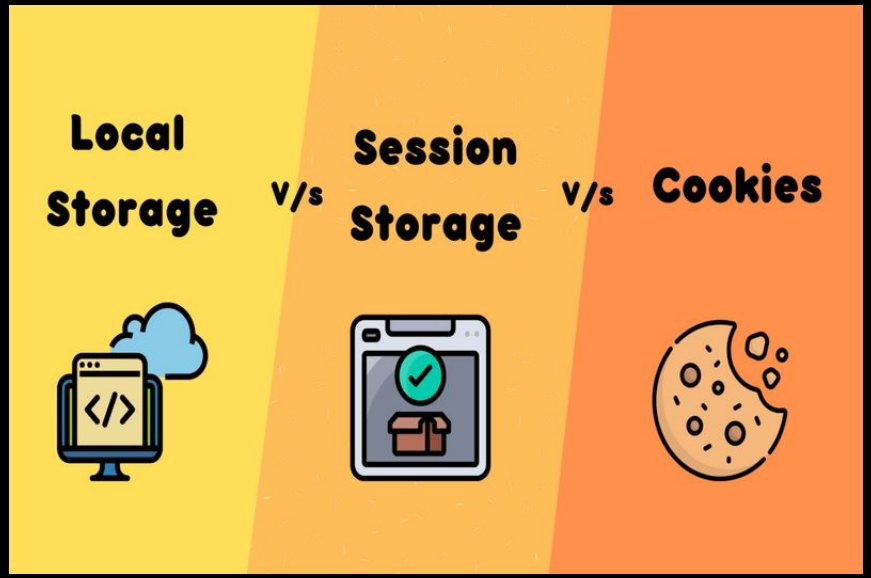
3. NodeJs Features
(Removed)

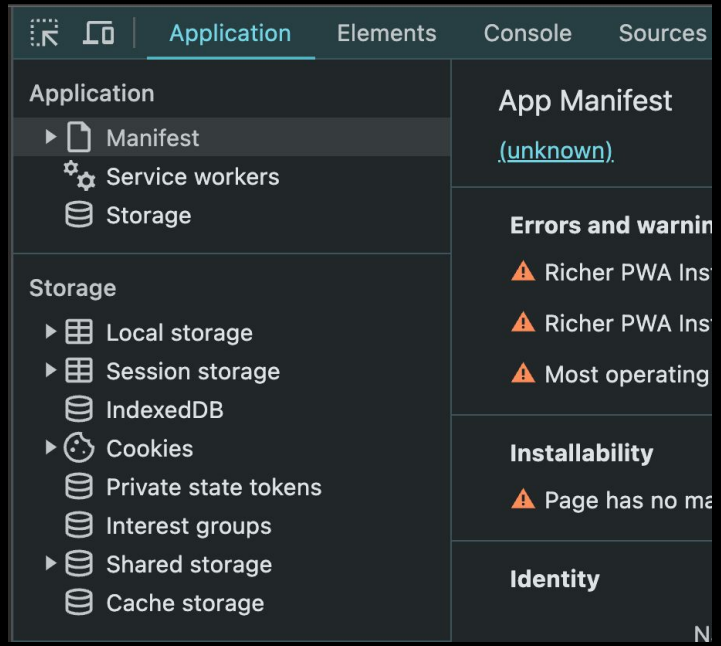




1. Window Object: The global window object, which is part of web browsers, is absent in Node.js.
2. DOM Manipulation: Node.js does not have a built-in Document Object Model (DOM), as it is not intended to interact with a webpage's content.
3. BOM (Browser Object Model): No direct interaction with things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

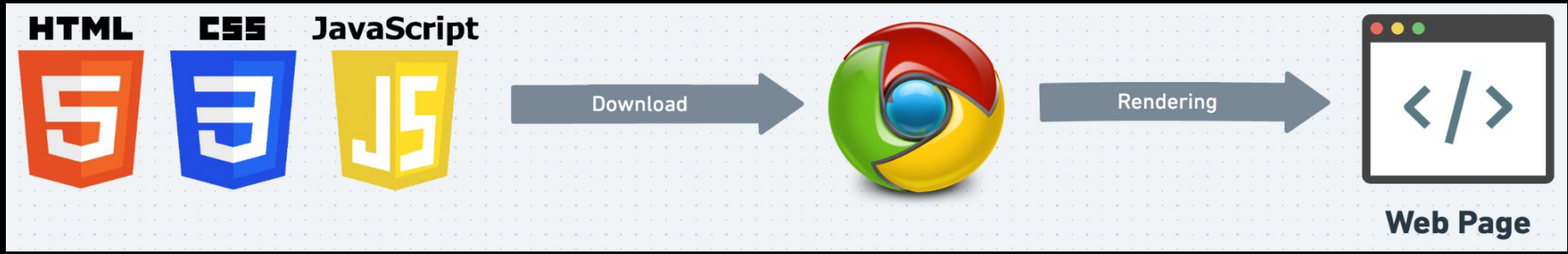
3. NodeJs Features
(Removed)





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

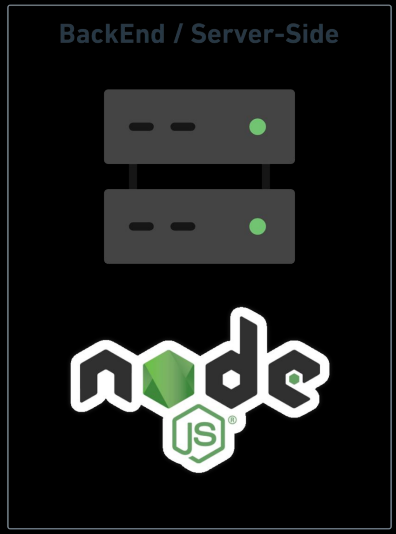
4. JavaScript on Client



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.

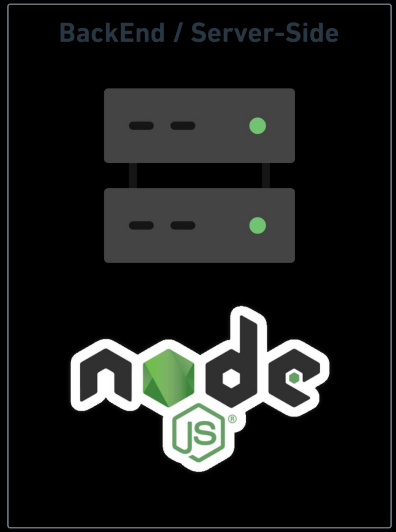
5. JavaScript on Server

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.



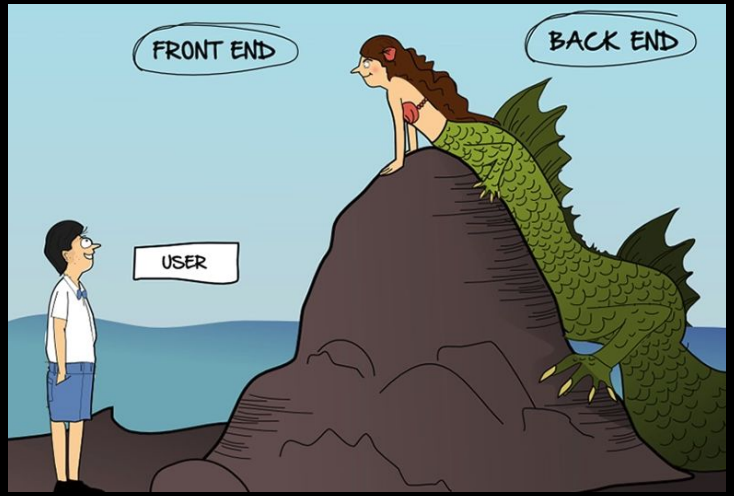
5. JavaScript on Server

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.



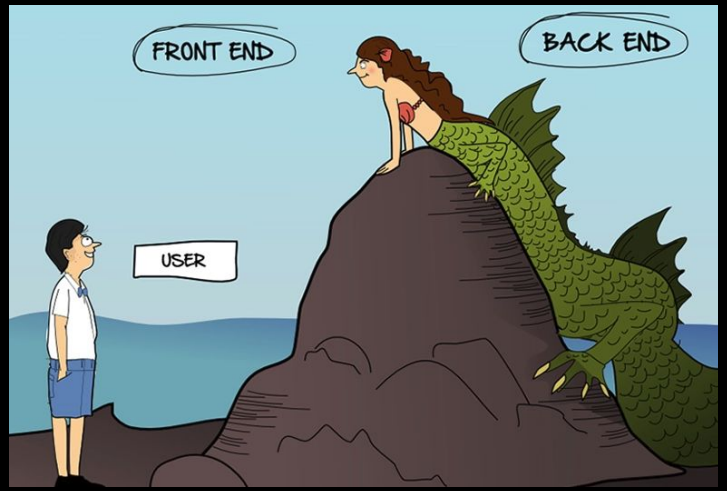
6. Client Code vs Server Code

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.



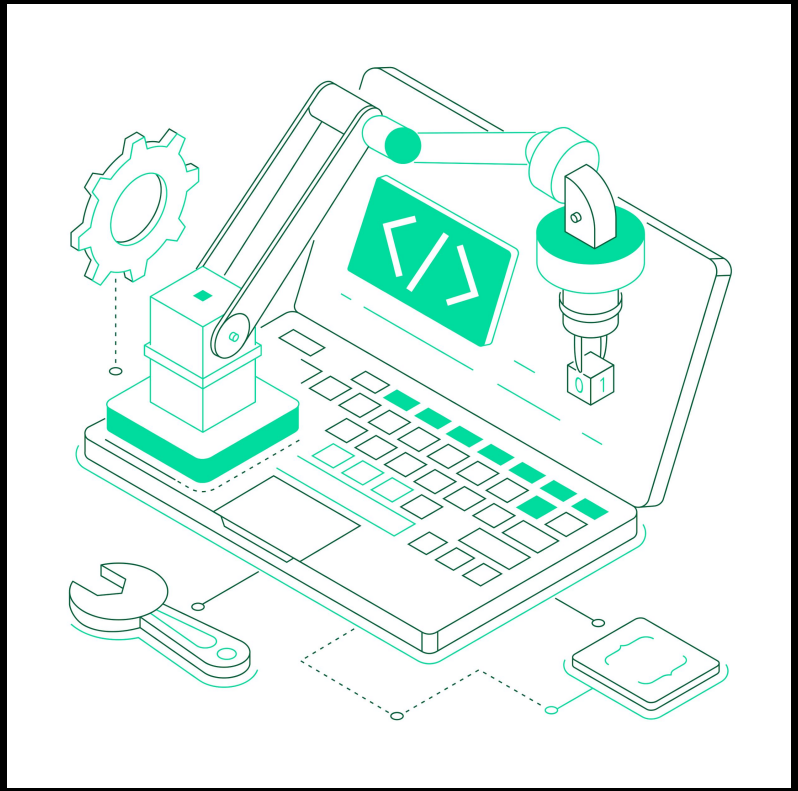
6. Client Code vs Server Code

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.

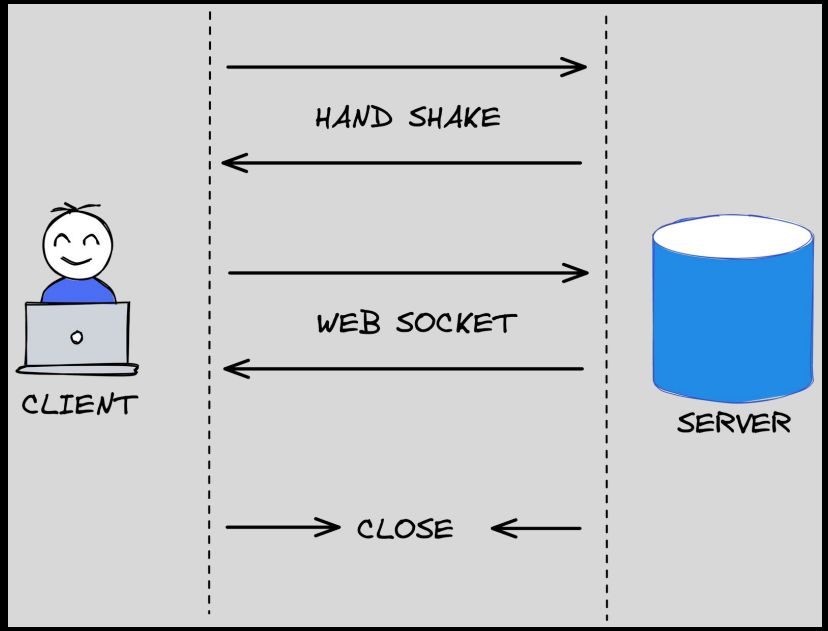


7. Other uses of NodeJs

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.

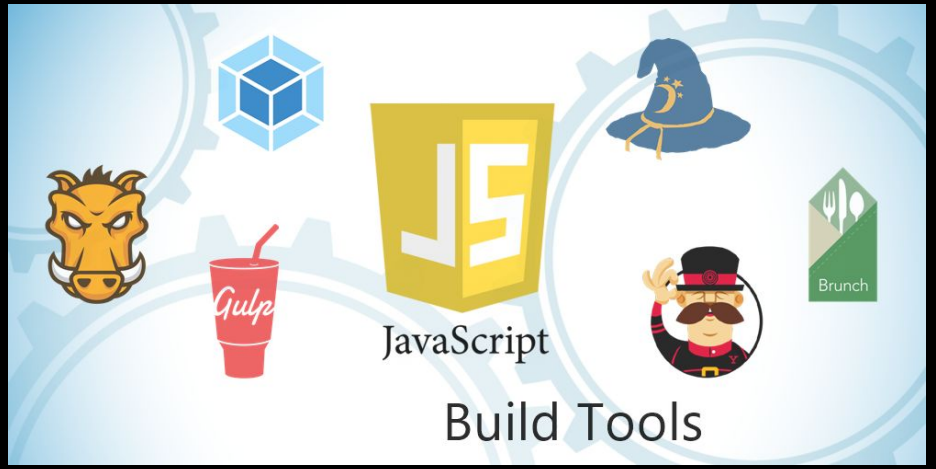
7. Other uses of NodeJs

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

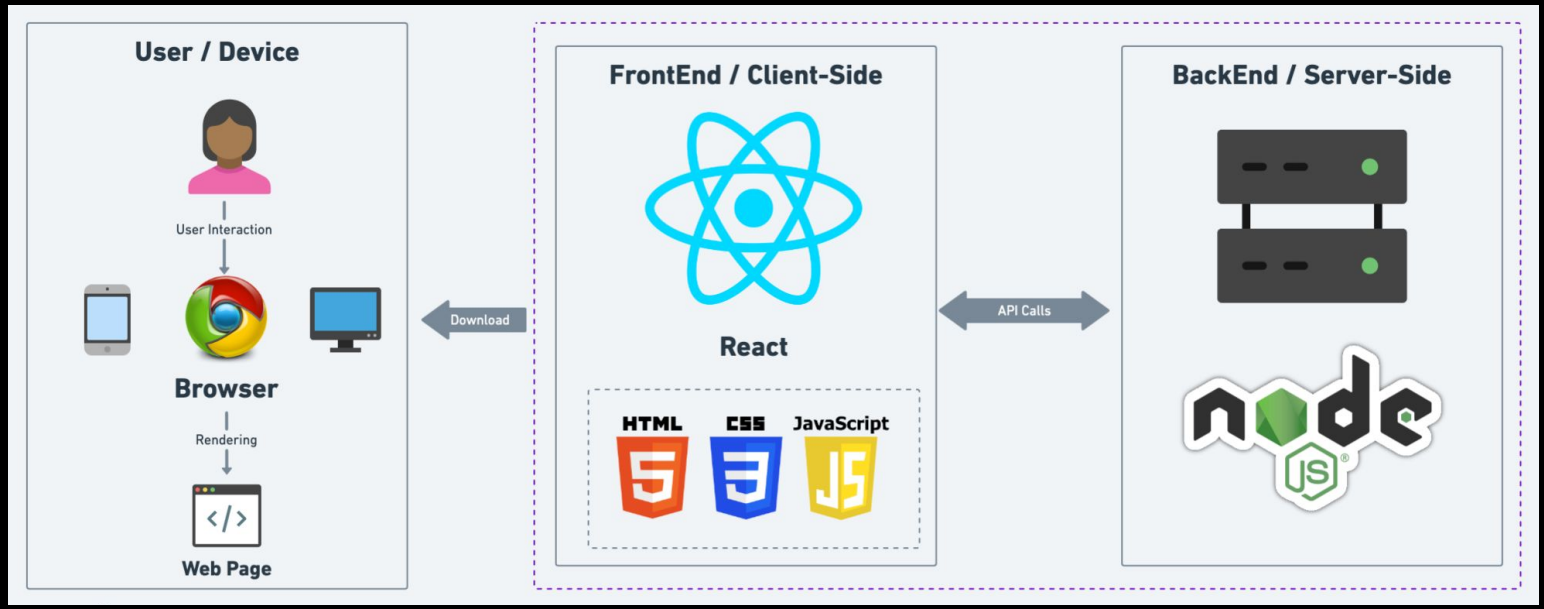
7. Other uses of NodeJs

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

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



8. Server architecture with NodeJs

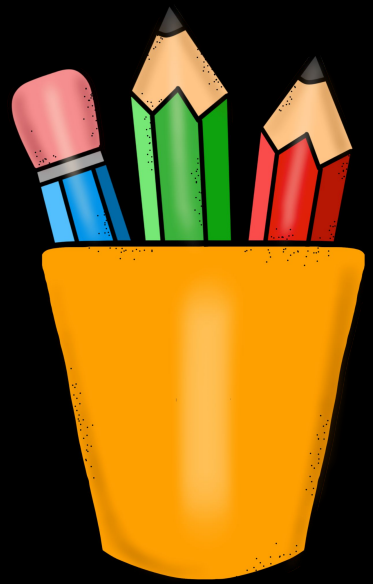


Nodejs server will:

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

Revision

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. Installation of NodeJS

1. What is IDE
2. Need of IDE
3. MAC Setup
   - Install latest Node & VsCode
4. Windows Setup
   - Install latest Node & VsCode
5. Linux Setup
   - Install latest Node & VsCode
6. VsCode (Extensions and Settings)
7. Executing first .js file
8. What is REPL
9. Executing Code via REPL

2.1 What is IDE

1. IDE stands for Integrated Development Environment.
2. Software suite that consolidates basic tools required for software development.
3. Central hub for coding, finding problems, and testing.
4. Designed to improve developer efficiency.

2.2 Need of IDE

1. Streamlines development.
2. Increases productivity.
3. Simplifies complex tasks.
4. Offers a unified workspace.
5. IDE Features
    1. Code Autocomplete
    2. Syntax Highlighting
    3. Version Control
    4. Error Checking

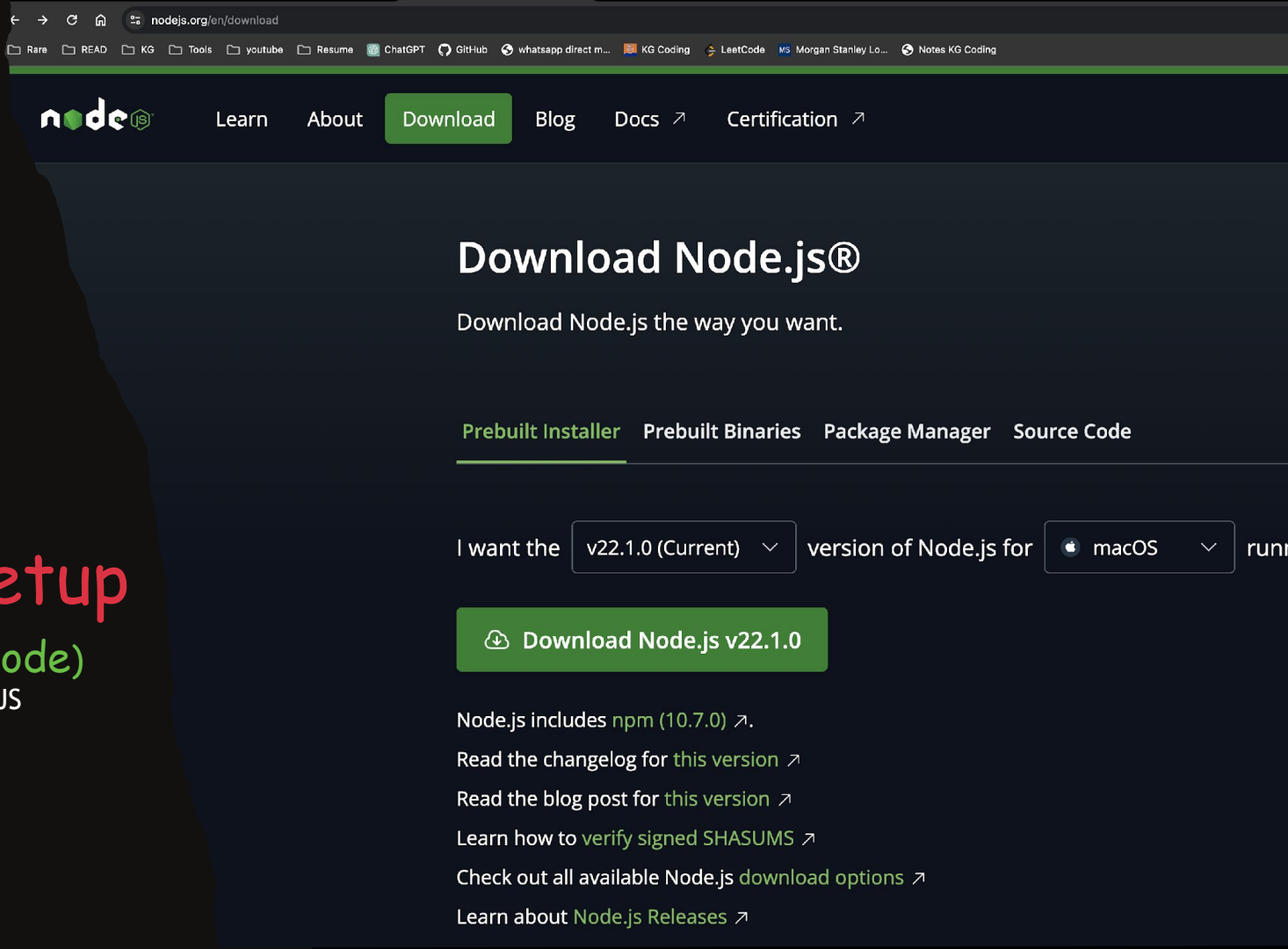
2.3 MAC Setup

mac
OS

2.3 MAC Setup

(Install latest Node)

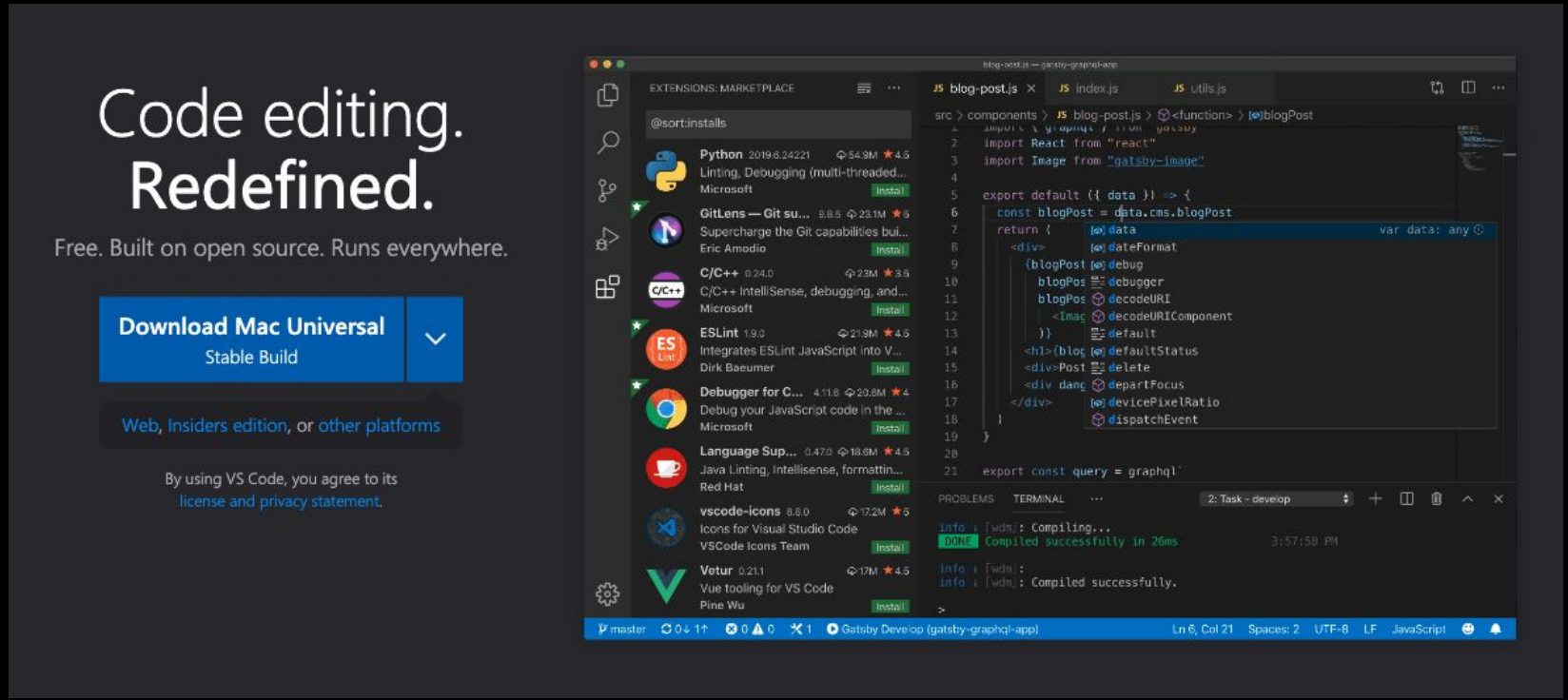
Search Download NodeJS



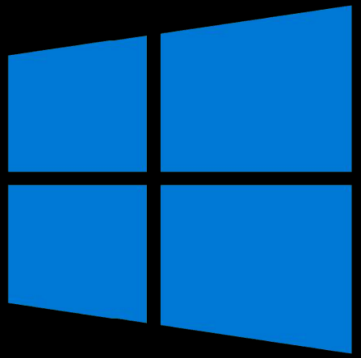
2.3 MAC Setup

(Install VsCode)

Search VS Code on Google



2.4 Windows Setup

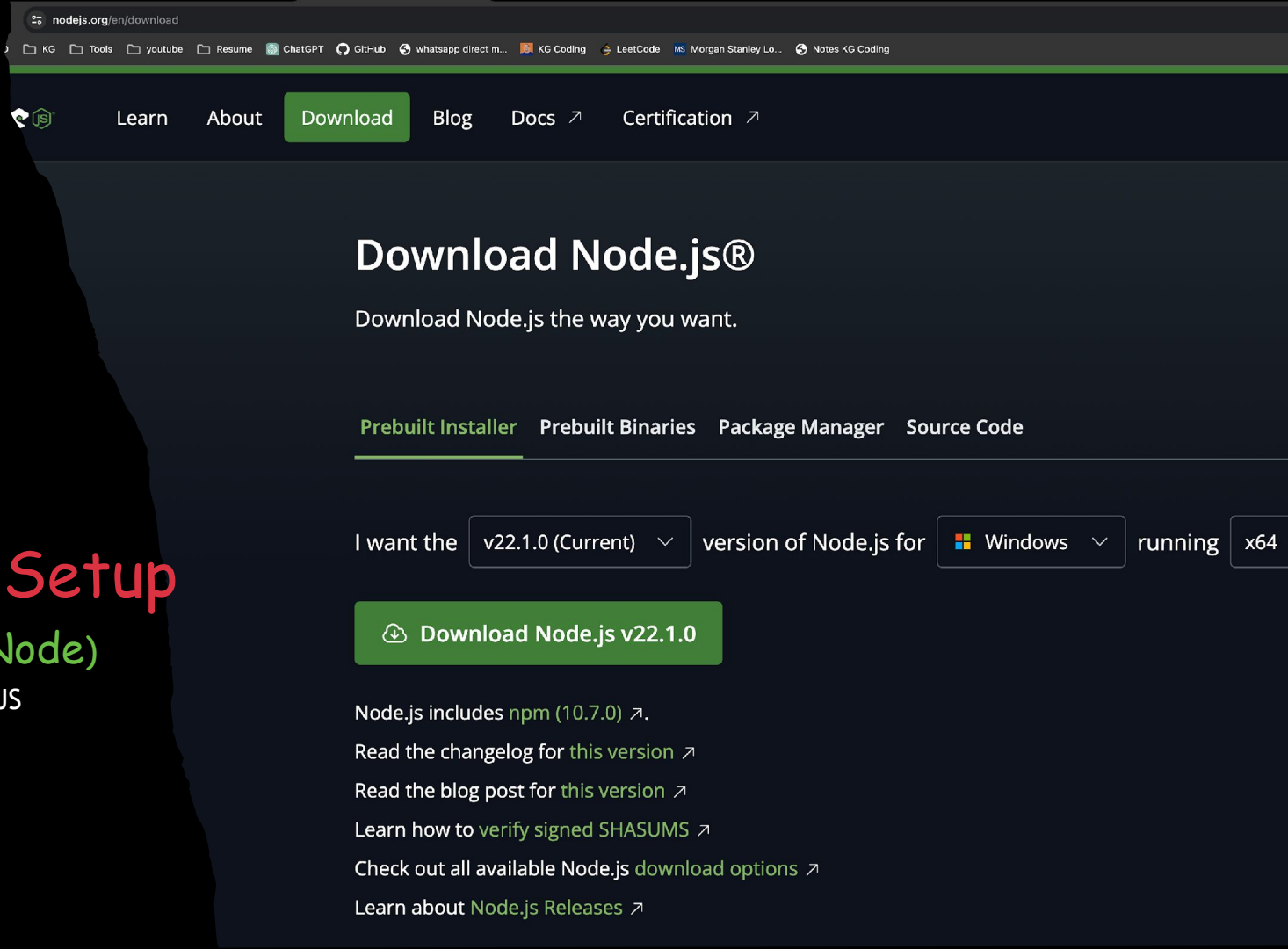


Windows

2.4 Windows Setup

(Install latest Node)

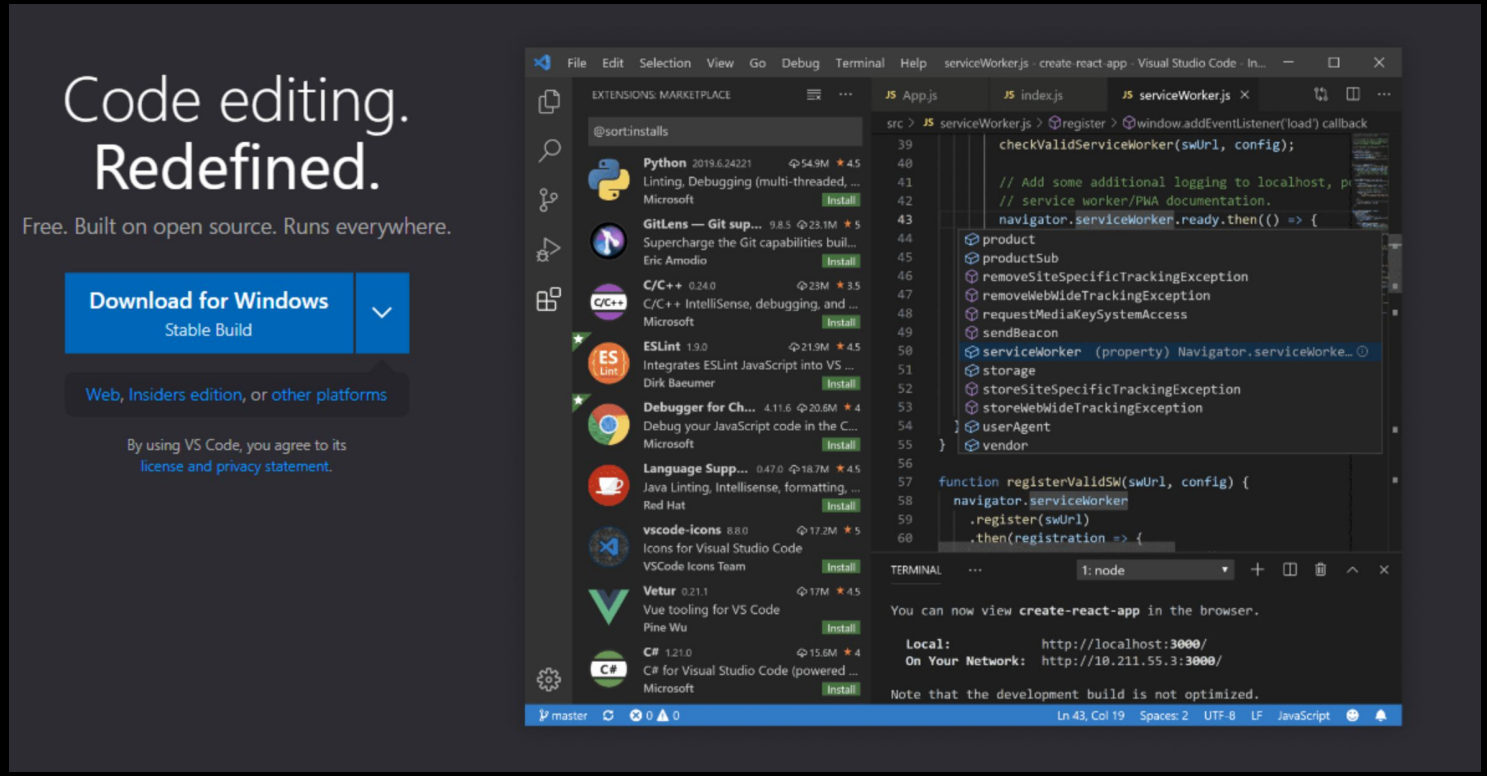
Search Download NodeJS



2.4 Windows Setup

(Install VsCode)

Search VS Code on Google



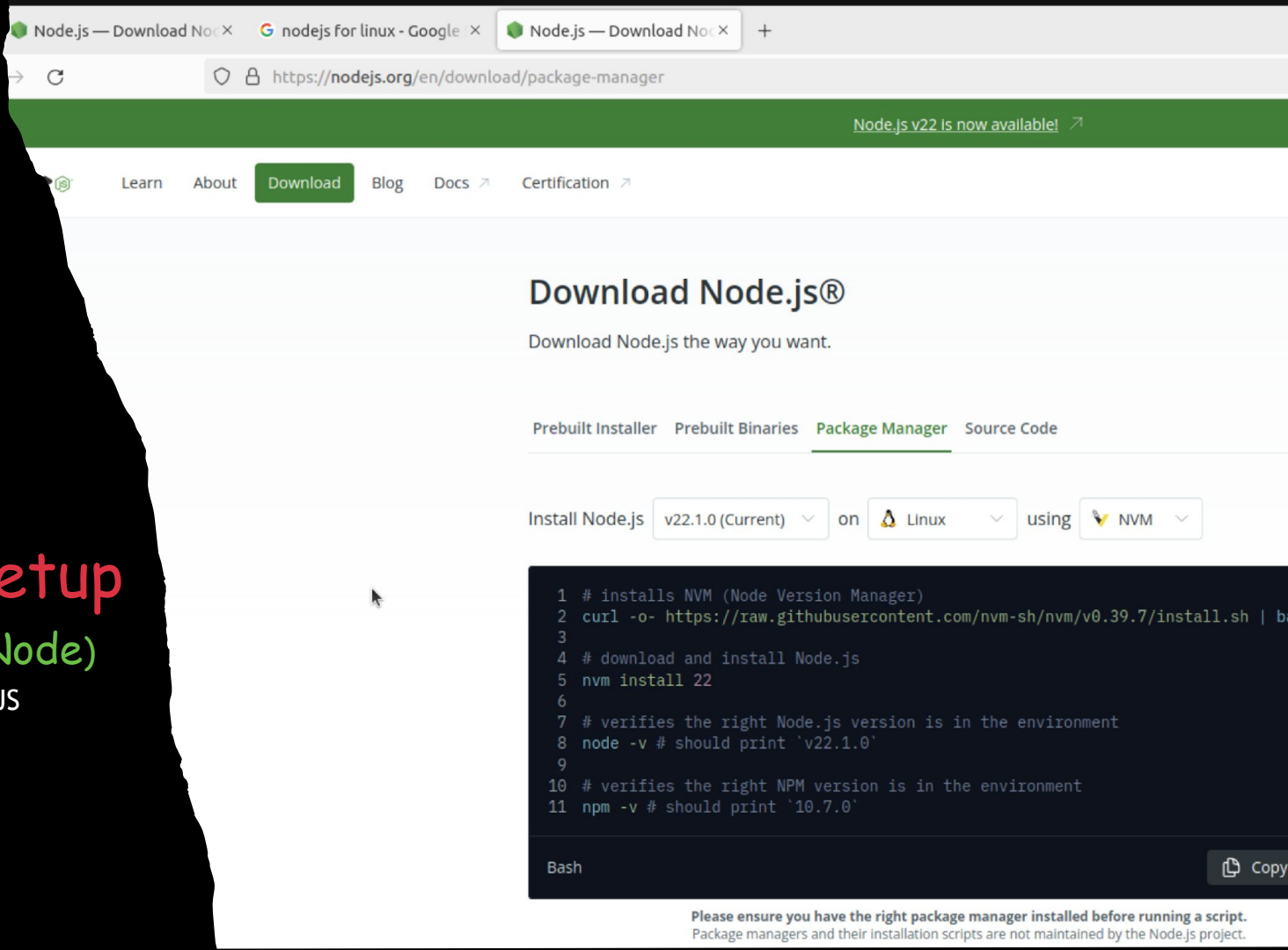
2.5 Linux Setup

Linux

2.5 Linux Setup

(Install latest Node)

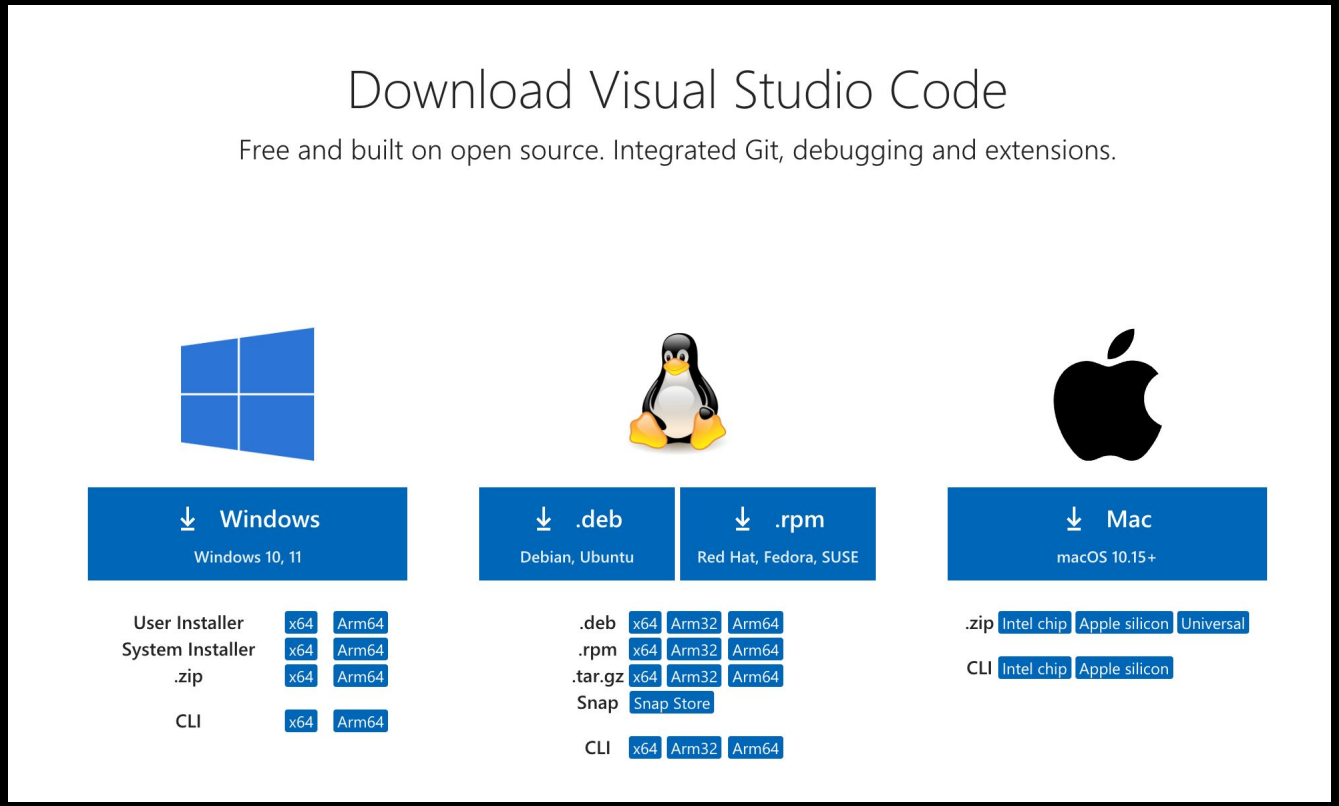
Search Download NodeJS



2.5 Linux Setup

(Install VsCode)

Search VS Code on Google



2.6 VsCode
(Extensions and Settings)

1. Prettier (Format on Save)
2. Line Wrap
3. Tab Size from 4 to 2

2.7 Executing first .js file

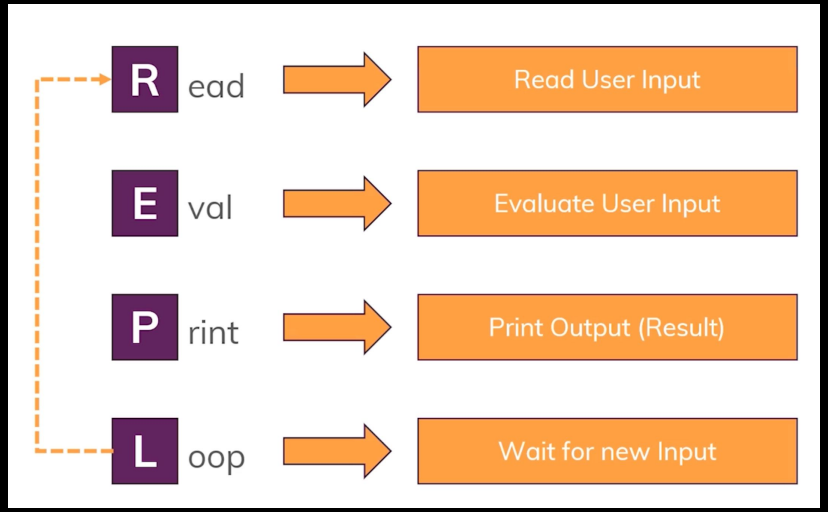
```
const fs = require('fs');

// Define two variables
let a = 10;
let b = 5;

// Basic arithmetic operations
let sum = a + b;
let product = a * b;

// Prepare data to write
let data = `Sum: ${sum}\nProduct: ${product}`;
console.log(data);

// Write data to a local file
fs.writeFile('output.txt', data, (err) => {
    if (err) throw err;
    console.log('Data written to file');
});
```

```
prashantjain@Mac-mini Desktop % node test.js
Sum: 15
Product: 50
Data written to file
```

1. Streamlines Node Command: Use node filename.js to execute a JavaScript file in the Node.js environment.
2. Require Syntax: Use require('module') to include built-in or external modules, or other JavaScript files in your code.
3. Modular Code: require helps organize code into reusable modules, separating concerns and improving maintainability.
4. Caching: Modules loaded with require are cached, meaning the file is executed only once even if included multiple times.

2.8 What is REPL

1. Streamlines Interactive Shell: Executes JavaScript code interactively.
2. Quick Testing: Ideal for testing and debugging code snippets on the fly.
3. Built-in Help: Offers help commands via .help.
4. Session Management: Supports saving (.save) and loading (.load) code sessions.
5. Node.js API Access: Provides direct access to Node.js APIs for experimentation.
6. Customizable: Allows customization of prompt and behaviour settings.

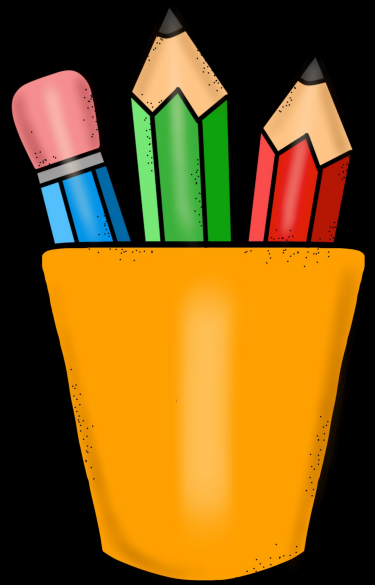


2.9 Executing Code via REPL

```
prashantjain@Mac-mini Desktop % node
Welcome to Node.js v20.9.0.
Type ".help" for more information.
> 5 + 6
11
> console.log('KG Coding is the best');
KG Coding is the best
undefined
> fs.writeFile('output.txt', 'Writing to file', (err) => {
...     if (err) throw err;
...     console.log('Data written to file');
... });
undefined
> Data written to file
```

Revision

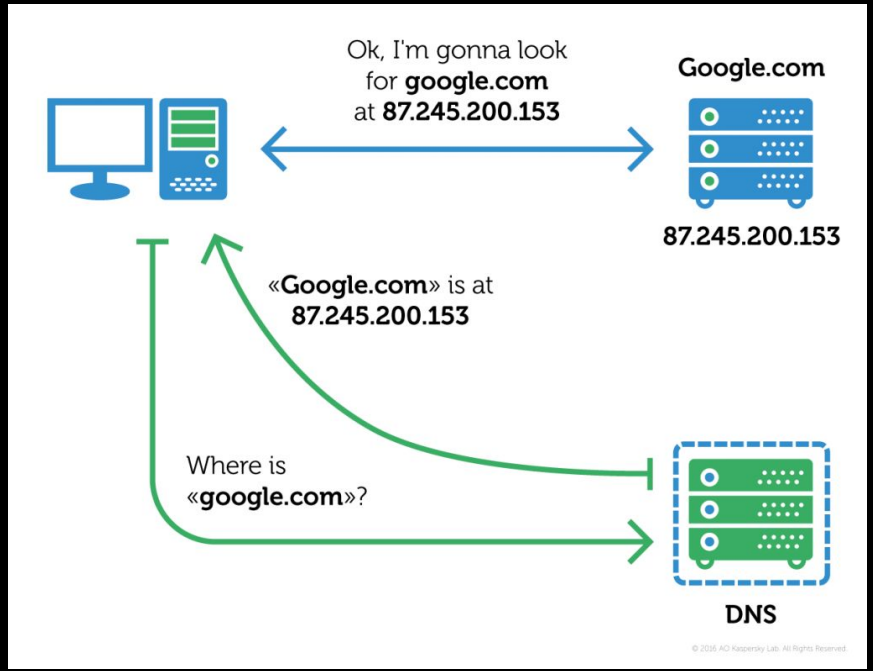
1. What is IDE
2. Need of IDE
3. MAC Setup
   - Install latest Node & VsCode
4. Windows Setup
   - Install latest Node & VsCode
5. Linux Setup
   - Install latest Node & VsCode
6. VsCode (Extensions and Settings)
7. Executing first .js file
8. What is REPL
9. Executing Code via REPL



3. First Node Server

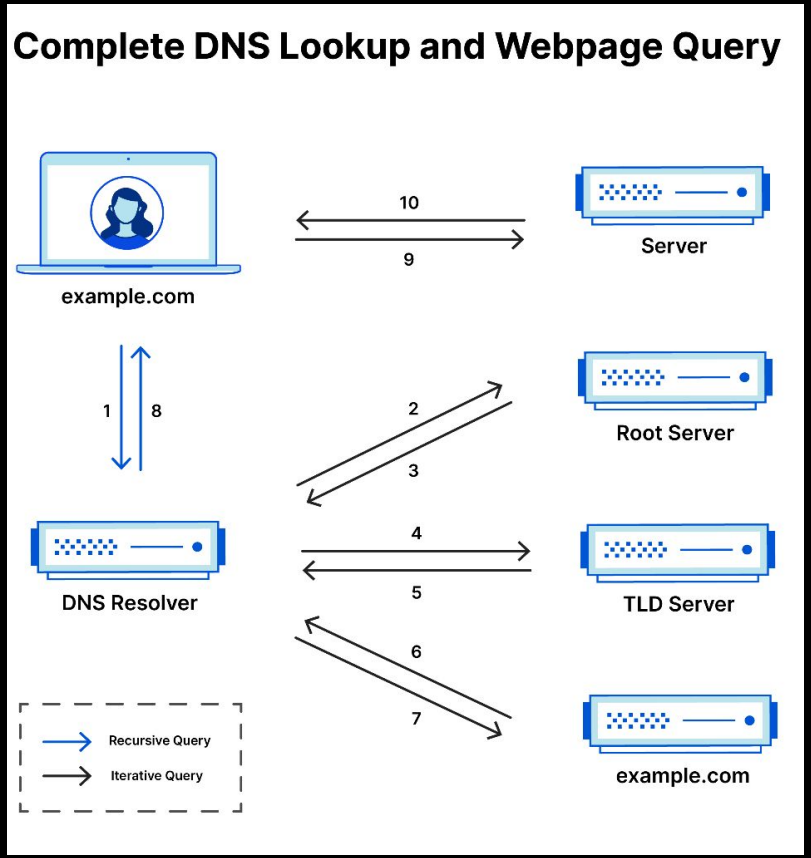
1. How DNS Works?
2. How Web Works?
3. What are Protocols?
4. Node Core Modules
5. Require Keyword
6. Creating first Node Server

3.1 How DNS Works?



1. Domain Name Entry: User types a domain (e.g., www.example.com) into the browser.
2. DNS Query: The browser sends a DNS query to resolve the domain into an IP address.
3. DNS Server: Provides the correct IP address for the domain.
4. Browser Connects: The browser uses the IP to connect to the web server and loads the website.

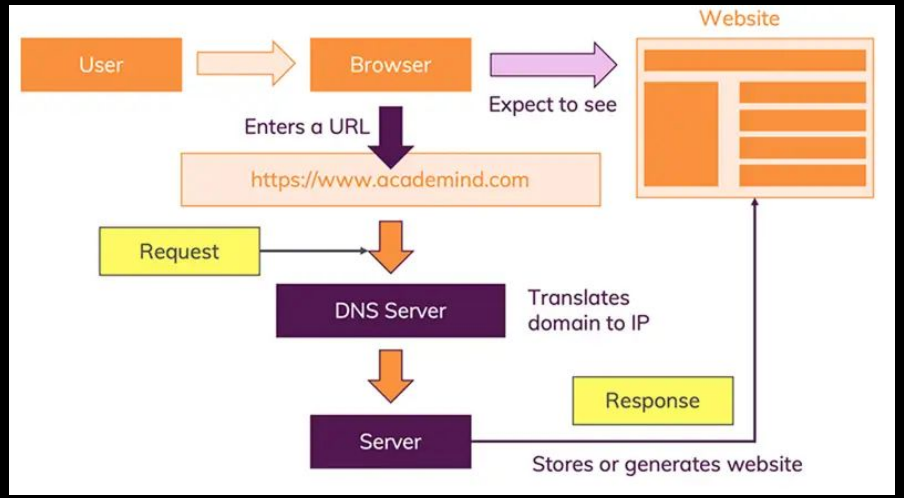
3.1 How DNS Actually Works?



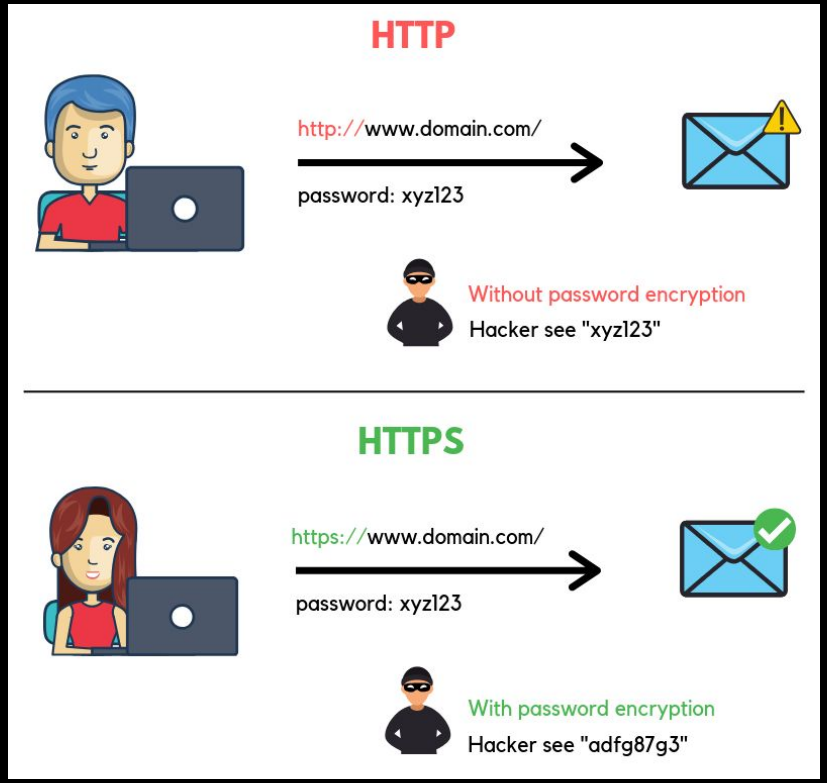
1. Root DNS: Acts as the starting point for DNS resolution. It directs queries to the correct TLD server (e.g., .com, .org).
2. TLD (Top-Level Domain) DNS: Handles queries for specific top-level domains (e.g., .com, .net) and directs them to the authoritative DNS server (e.g., Verisign for .com, PIR for .org)
3. Authoritative DNS: Contains the actual IP address of the domain and answers DNS queries with this information.(e.g., Cloudflare, Google DNS).

3.2 How Web Works?

1. Client Request Initiation: The client (browser) initiates a network call by entering a URL.
2. DNS Resolution: The browser contacts a DNS server to get the IP address of the domain.
3. TCP Connection: The browser establishes a TCP connection with the server's IP address.
4. HTTP Request: The browser sends an HTTP request to the server.
5. Server Processing: The server processes the request and prepares a response.
6. HTTP Response: The server sends an HTTP response back to the client.
7. Network Transmission: The response travels back to the client over the network.
8. Client Receives Response: The browser receives and interprets the response.
9. Rendering: The browser renders the content of the response and displays it to the user.



3.3 What are Protocols?



Http (HyperText Transfer Protocol):

- Facilitates communication between a web browser and a server to transfer web pages.
- Sends data in plain text (no encryption).
- Used for basic website browsing without security.

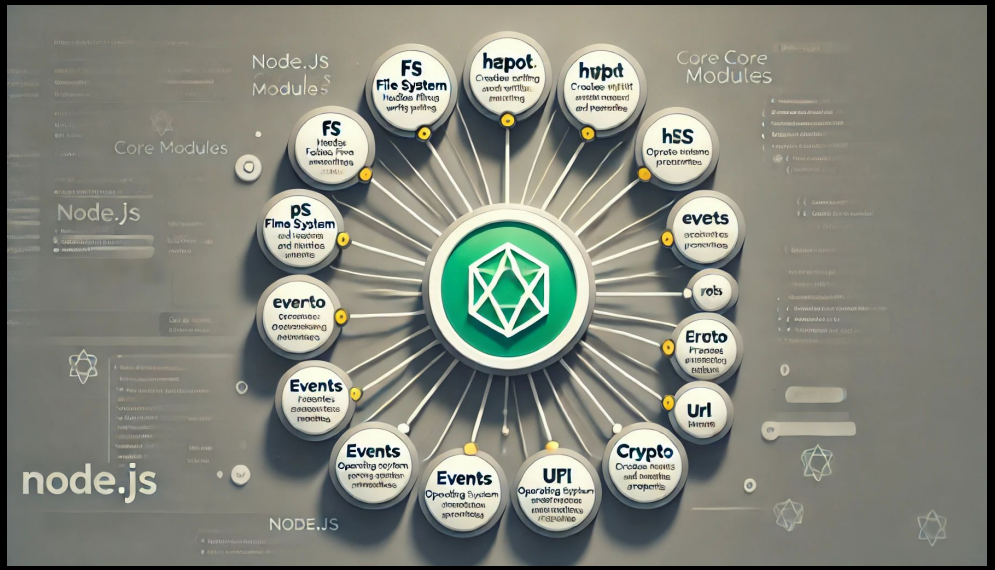
HTTPS (HyperText Transfer Protocol Secure):

- Secure version of HTTP, encrypts data for secure communication.
- Uses SSL/TLS to encrypt data.
- Used in online banking, e-commerce.

TCP (Transmission Control Protocol):

- Ensures reliable, ordered, and error-checked data delivery over the internet.
- Establishes a connection before data is transferred.

3.4 Node Core Modules



1. Built-in: Core modules are included with Node.js installation.
2. No Installation Needed: Directly available for use without npm install.
3. Performance: Highly optimized for performance.