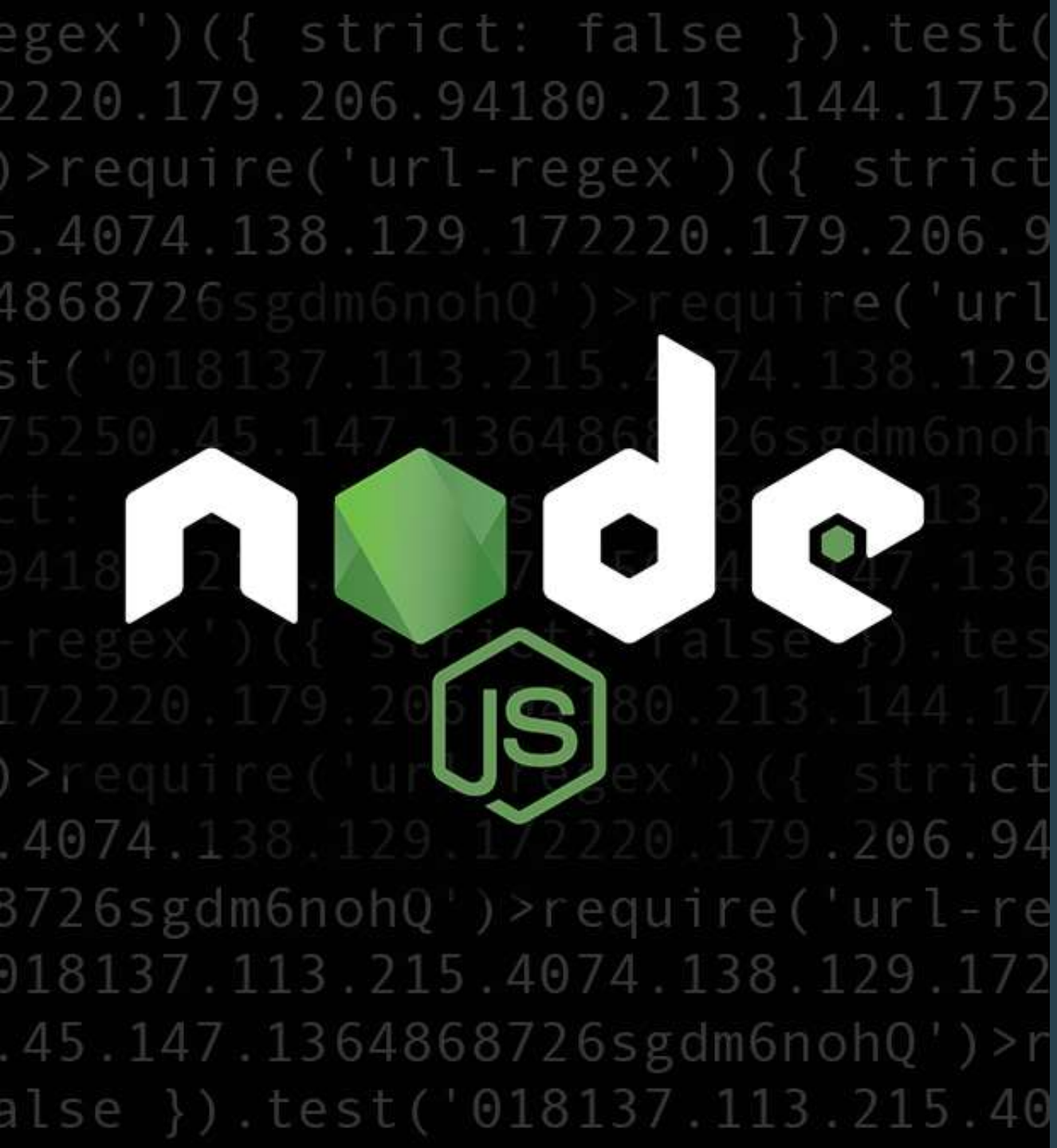
# Introduction to Node JS


Qinshift
Academy




# Hello there

## WhoAmI

Developer@Company

Trainer @SEDC

## WhoAmI

Developer@Company

Trainer @SEDC

# AIMS AND GOALS OF THIS COURSE

- Understand the basics of Node.js
- Learn to use npm packages
- Solve problems in Node.js by building applications
- Write extensive JavaScript code compiled in the Node.js runtime environment

# WHERE ARE WE NOW?

- Just completed the front-end part of the academy
- Have a solid understanding of:
  - HTML
  - CSS
  - JavaScript
- Capable of providing front-end solutions to problems

# A WEB SOLUTION

**Client-Side (User Interface)**

• What users see and interact with, built for their convenience

• Technologies: HTML, CSS, JavaScript, JavaScript Frameworks

**Server-Side (Backend)**

• Not visible to users; forms the foundation of the application and business logic

• Technologies:

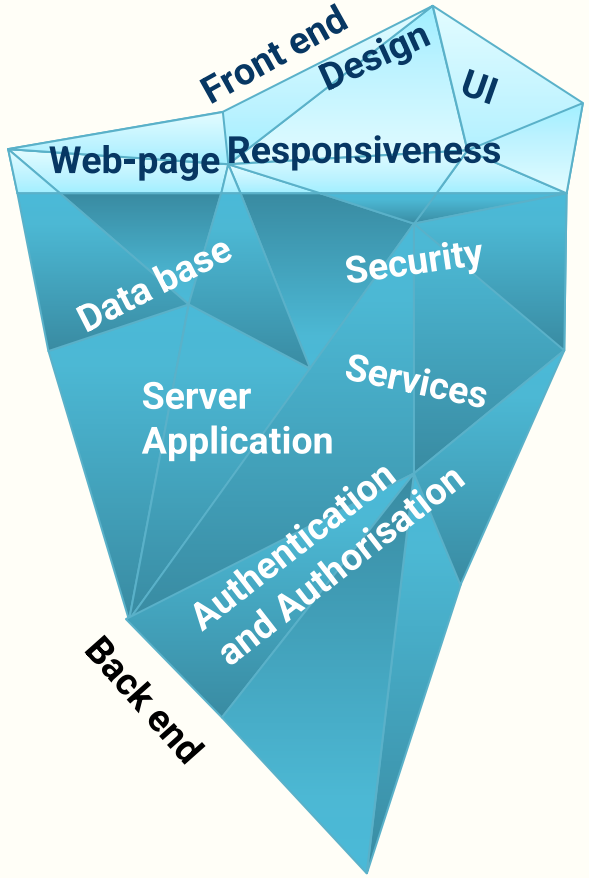
• Languages: C#, Java, JavaScript, Python, PHP

• Architectures: MVC, REST API, GraphQL

• Databases: SQL, MySQL, MongoDB

• Frameworks: .NET, Nest.js

• Features: Authentication and Authorization



# WHAT IS NODE.JS?

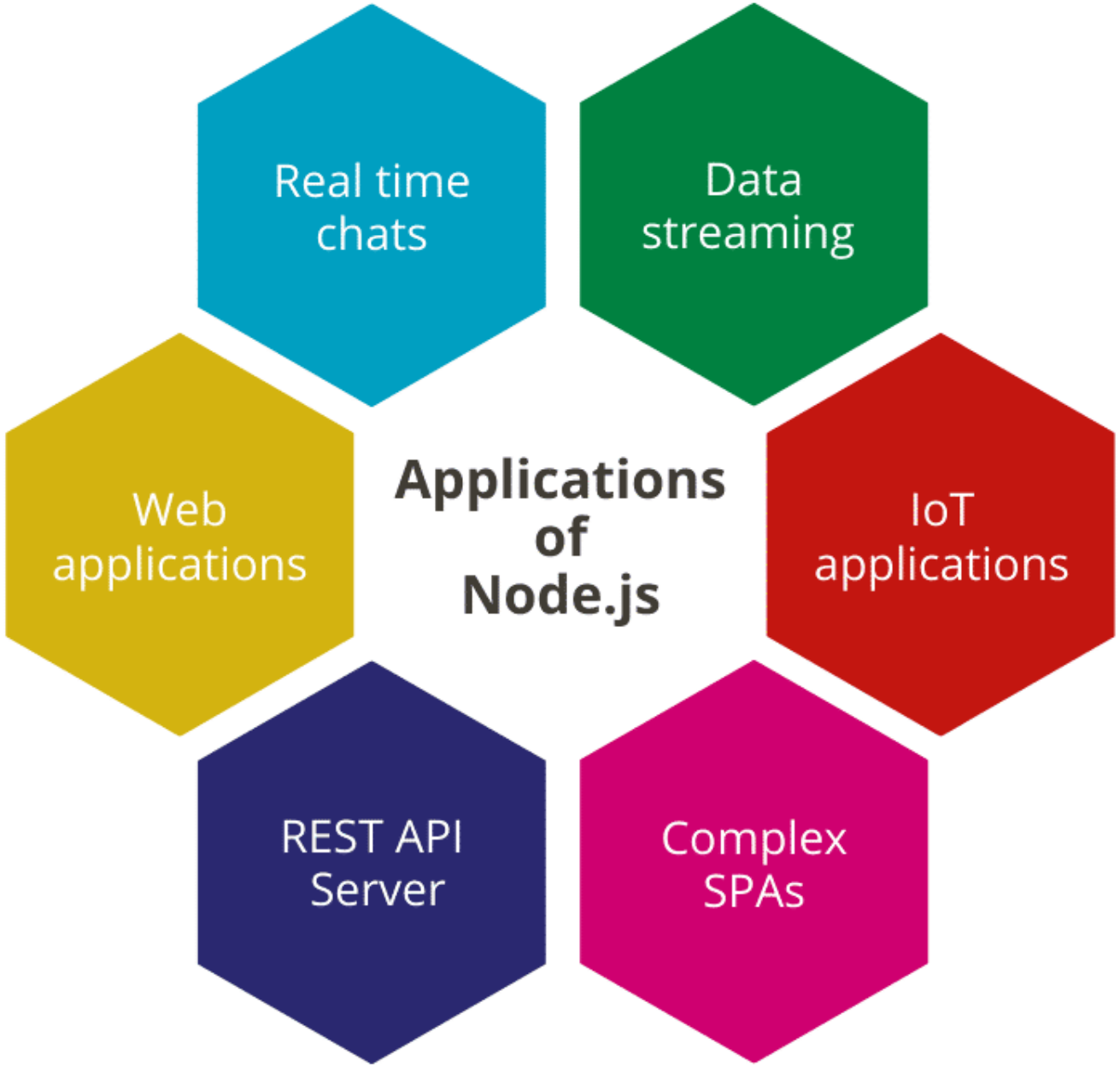
- Node.js is an open-source, cross-platform JavaScript runtime environment. It’s a versatile tool suitable for almost any project type.
- Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside the browser, which contributes to its high performance.

# SINGLE-THREADED AND ASYNC

- A Node.js application runs in a single process, without creating new threads for each request. Node.js provides asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking.
- When Node.js performs I/O operations (e.g., reading from the network, accessing a database, or the filesystem), it doesn’t block the thread. Instead, it resumes operations when the response returns.
- This design allows Node.js to handle thousands of concurrent connections with a single server.

# SAME LANGUAGE... YAAY!

- Node.js offers a unique advantage: millions of frontend developers who write JavaScript for browsers can now write server-side code without learning a completely different language.
- In Node.js, developers can use new ECMAScript standards without waiting for users to update their browsers. You control which ECMAScript version to use by changing the Node.js version and can enable specific experimental features by running Node.js with flags.




# Applications of Node.js

Qinshift Academy

# JS IN THE BROWSER & NODE.JS

While both the browser and Node.js use JavaScript, building browser apps differs significantly from building Node.js applications. The main difference lies in the ecosystem:

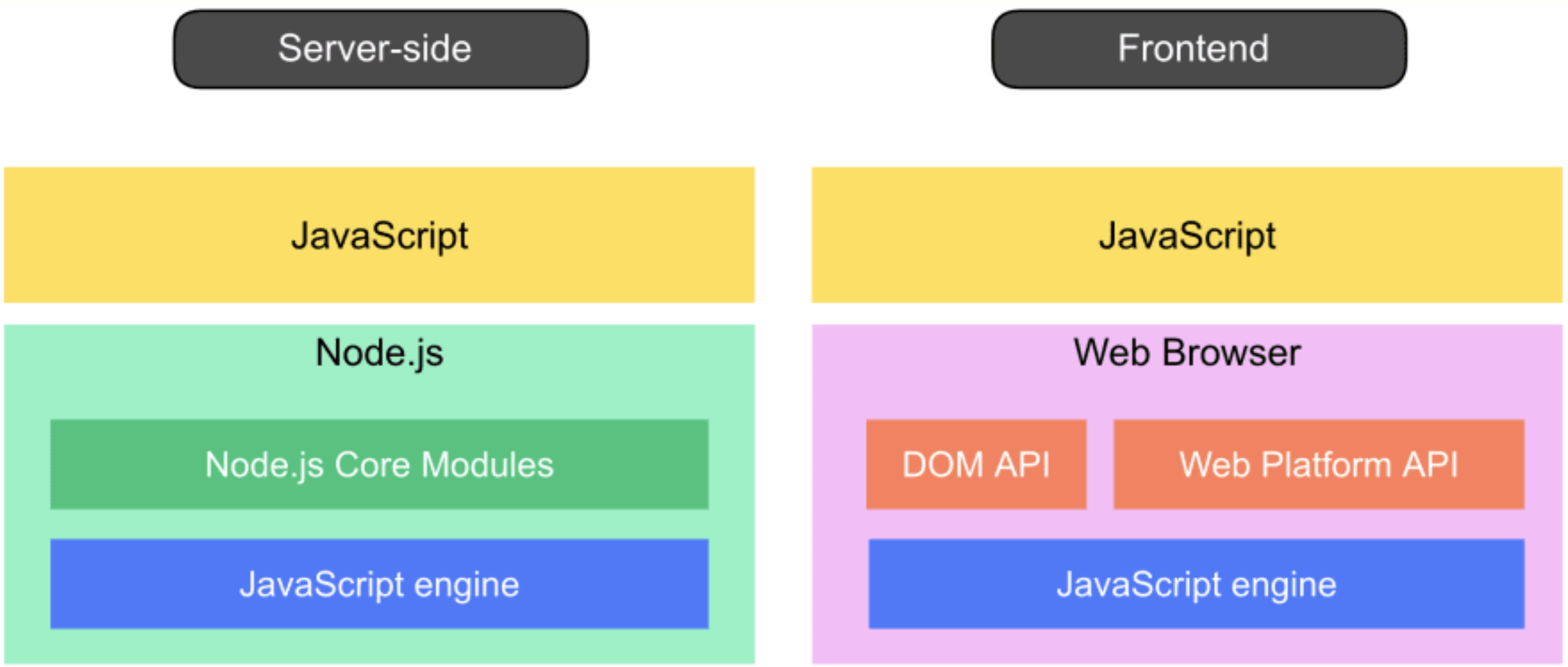
**Browser**

- Interacts with the DOM and Web Platform APIs (e.g., Cookies)
- Limited control over the user’s environment

**Node.js**

- Provides additional APIs through modules (e.g., file-system access)
- Allows greater control over the environment

# JS IN THE BROWSER & NODE.JS



# THE V8 JAVASCRIPT ENGINE

V8 is the JavaScript engine powering Google Chrome and Node.js. Key features include:

- Independence from the browser
- Written in C++
- Portable (runs on Mac, Windows, Linux, and other systems)
- Uses just-in-time (JIT) compilation to enhance execution speed

# NPM (Node Package Manager)

npm is the standard package manager for Node.js.

- Currently, over 3.1 million packages are listed in the npm registry
- Originally developed for Node.js packages, now used in frontend JavaScript as well

npm allows you to:

- Install and update project dependencies
- Control app versioning
- Run tasks using commands

# WHO IS USING NODE.JS?

# EXERCISE

1. Create a folder with a .js file in it
2. Add any content to the file
3. Open a terminal in that folder
4. Run the command ‘npm init’ and observe the results

# QUESTIONS?

You can reach us at:

mail

main

Qinshift Academy