Node.js Basics

Naveen Pete

Agenda

- What is Node.js?
- Node.js Architecture
- What is I/O?
- The Node.js Event Loop
- Blocking vs Non-Blocking
- Threading
- REPL
- NPM
- NVM

What is Node.js?

- JavaScript runtime built on Chrome's V8 JavaScript engine
- Uses an event-driven, non-blocking I/O model
- Created by Ryan Dahl in 2009
- Allows JavaScript to run on your machine as a standalone process, outside of the browser
- Platform for building high performance networked, web apps using JavaScript
- Foundation of MEAN / MERN stack

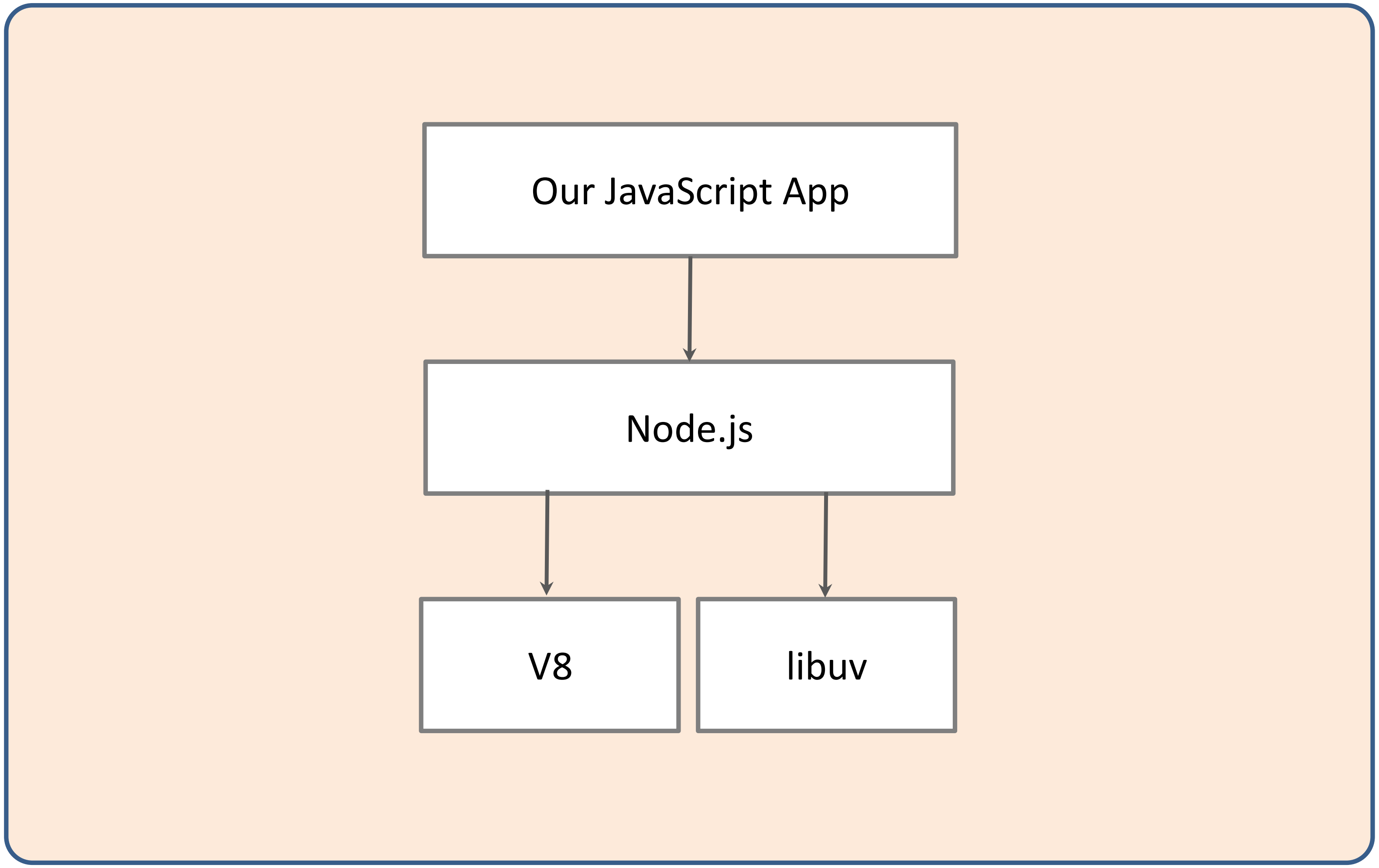
What is Node.js?

- Benefits
  - Node.js apps are lightweight and efficient
    - Efficient use of system resources
    - Serve more users on fewer resources
  - JavaScript code, leverage existing JS knowledge
  - No need to run a separate web server
  - Provides greater control over app logic as well as app environment

What is Node.js?

- Node.js is a platform to build
  - Real-time apps like chat servers
  - Collaborative multi-user apps
  - Real-time games
  - Data streaming servers
  - APIs for web & mobile apps
  - Apps that require heavy I/O (high concurrency)
  - Apps with high degree of scalability

Node.js Architecture



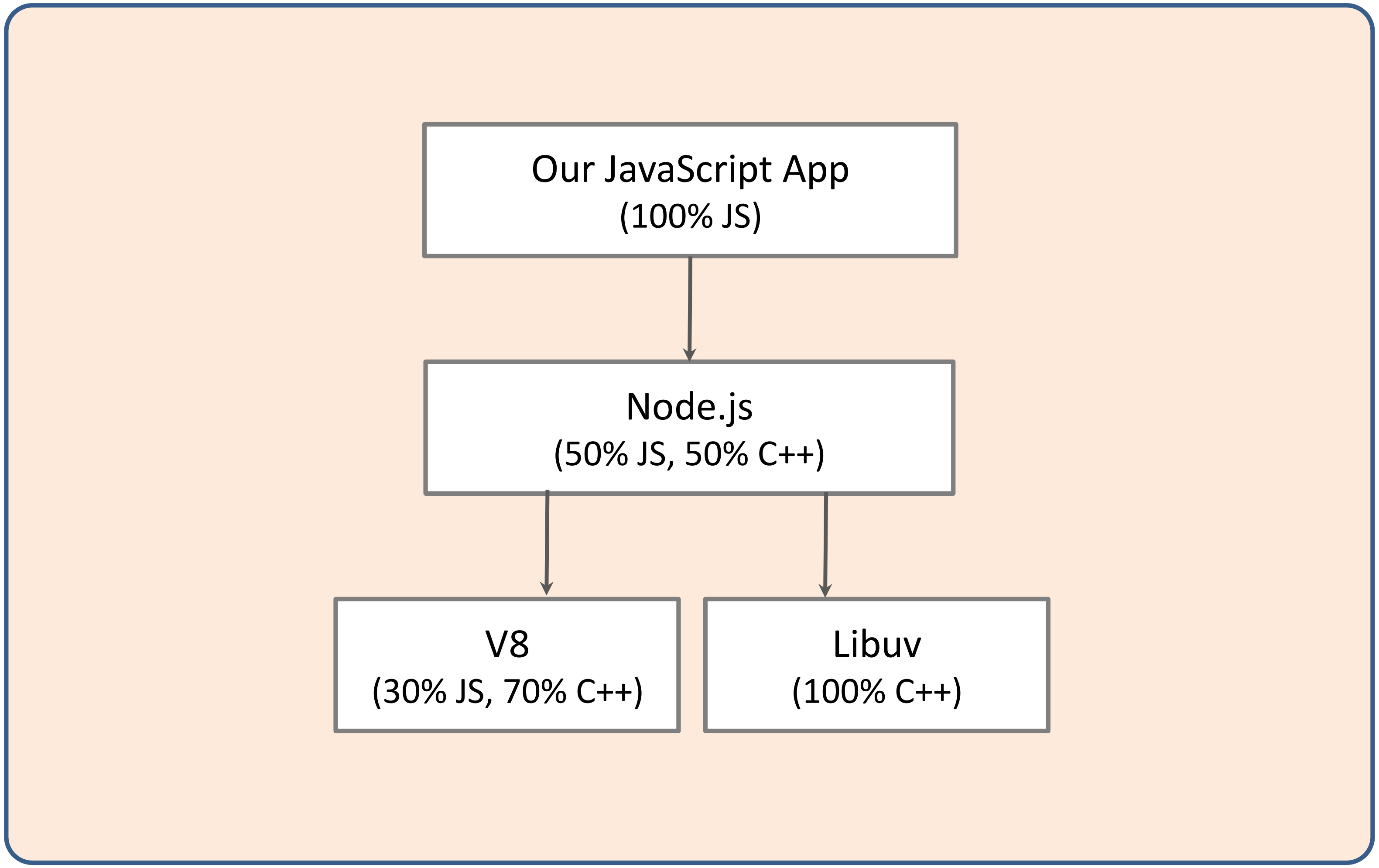
Node.js Architecture

- V8 Project
  - Open source JavaScript engine created by Google
  - Compiles JavaScript code into native machine code
  - Purpose is to execute JavaScript code outside of the browser
  - Written in C++, is highly performant
  - Enables Node.js apps to run across operating systems

Node.js Architecture

- libuv Project
  - C++ open source project
  - Provides Node.js
    - Access to the OS’ underlying file system
    - Access to networking
    - Ability to handle concurrency and processing
  - Enables Node.js to perform seamless I/O operations across operating systems in a non-blocking way

Node.js Architecture



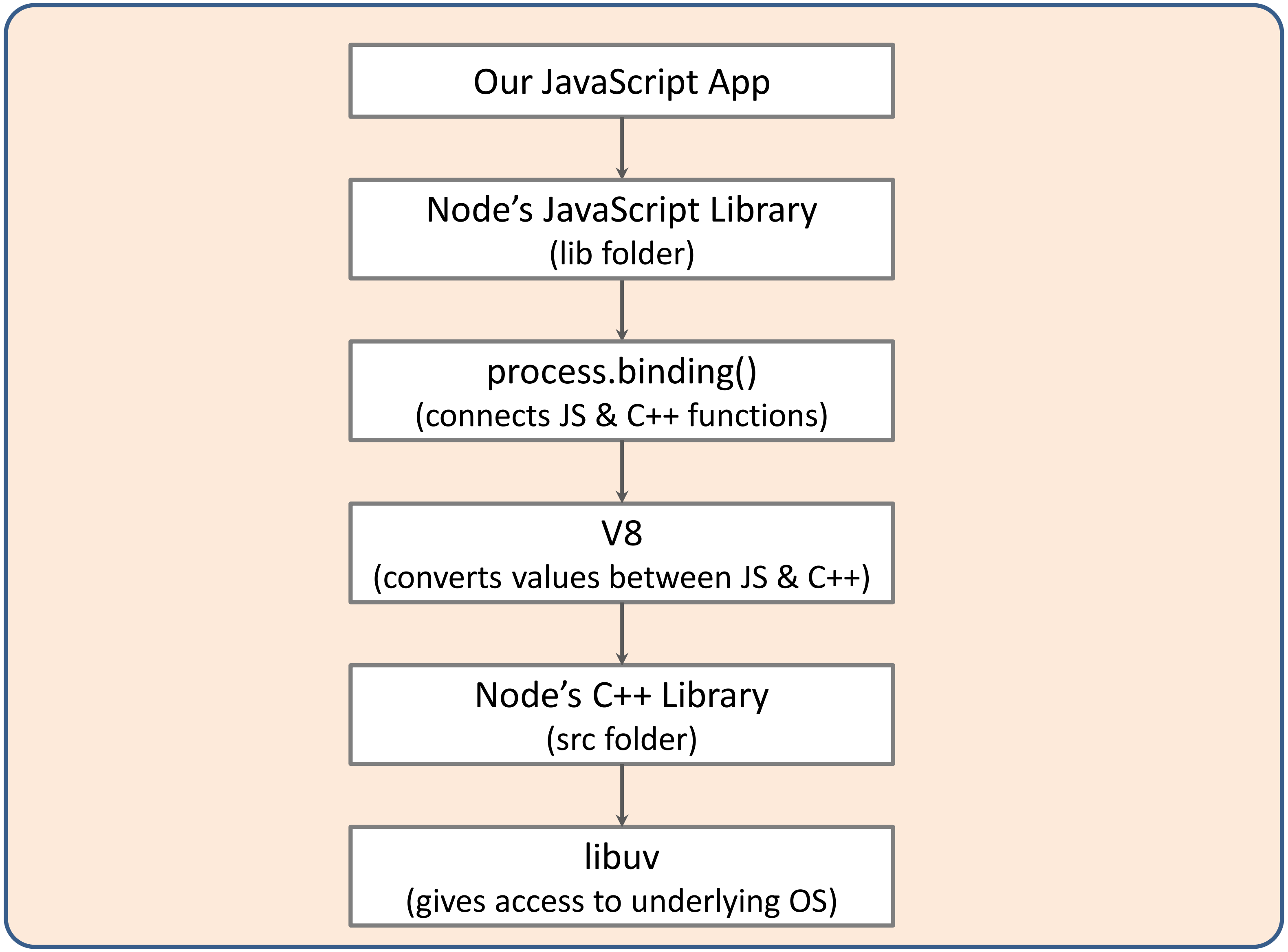
Node.js Architecture

- Node.js
  - Provides an interface that can be used to relate JavaScript code to C++ code
  - Provides wrappers and consistent API that can be used within our apps
    - For e.g. modules like HTTP, FS, Path, Crypto, etc.
  - Most of the functionality is implemented in C++ within V8 and libuv projects

Node.js Architecture

- GitHub Repository
  - URL
    - https://github.com/nodejs/node
  - ‘deps’ folder (Dependencies)
    - C++ libraries of code
    - Dependencies, built outside of Node.js, but are part of Node.js
  - ‘lib’ folder (The JavaScript Core)
    - JavaScript side of Node.js
    - Contains JavaScript definitions of functions & modules
    - Most of the code contains wrappers for C++ features
    - Helps making using the C++ features easier
    - Also contains other common tasks & utilities required for development
  - ‘src’ folder (The C++ Core)
    - Features & utilities coded and built in C++, made available to JavaScript
    - C++ implementation of all the functions
    - Contains code that interacts with V8 and libuv projects

Node.js Architecture


Our JavaScript App
Node’s JavaScript Library
(lib folder)
process.binding()
(connects JS & C++ functions)
V8
(converts values between JS & C++)
Node’s C++ Library
(src folder)
libuv
(gives access to underlying OS)

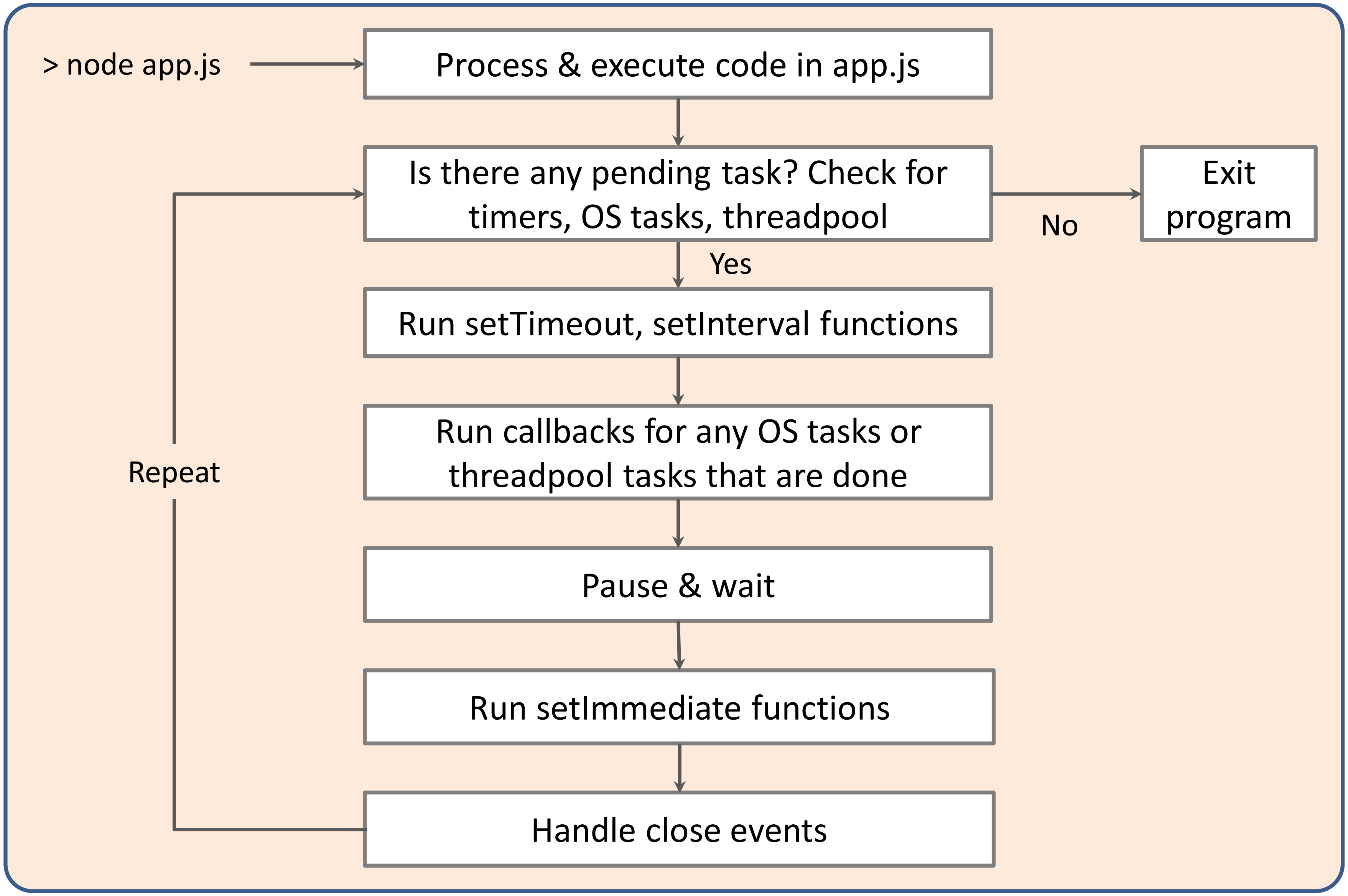
What is I/O?

- "I/O" means interaction with the system's disk and network
- Few examples
  - Reading from or writing to a database
  - File read / write
  - Issue Http request to a web server
    - Fetching a map using Google API
  - Send / receive information over the network

The Node.js Event Loop

- Initialized when Node.js starts
  - Allows Node.js to perform non-blocking I/O operations
  - Offloads operation(s) to the OS kernel whenever possible
  - OS kernel executes operation in the background
  - When the operation completes, the kernel informs Node.js so that the appropriate callback may be added to the poll queue
  - The callback eventually gets executed

The Node.js Event Loop


> node app.js
Process & execute code in app.js
Is there any pending task? Check for timers, OS tasks, threadpool
No
Exit program
Yes
Run setTimeout, setInterval functions
Run callbacks for any OS tasks or threadpool tasks that are done
Pause & wait
Run setImmediate functions
Handle close events
Repeat

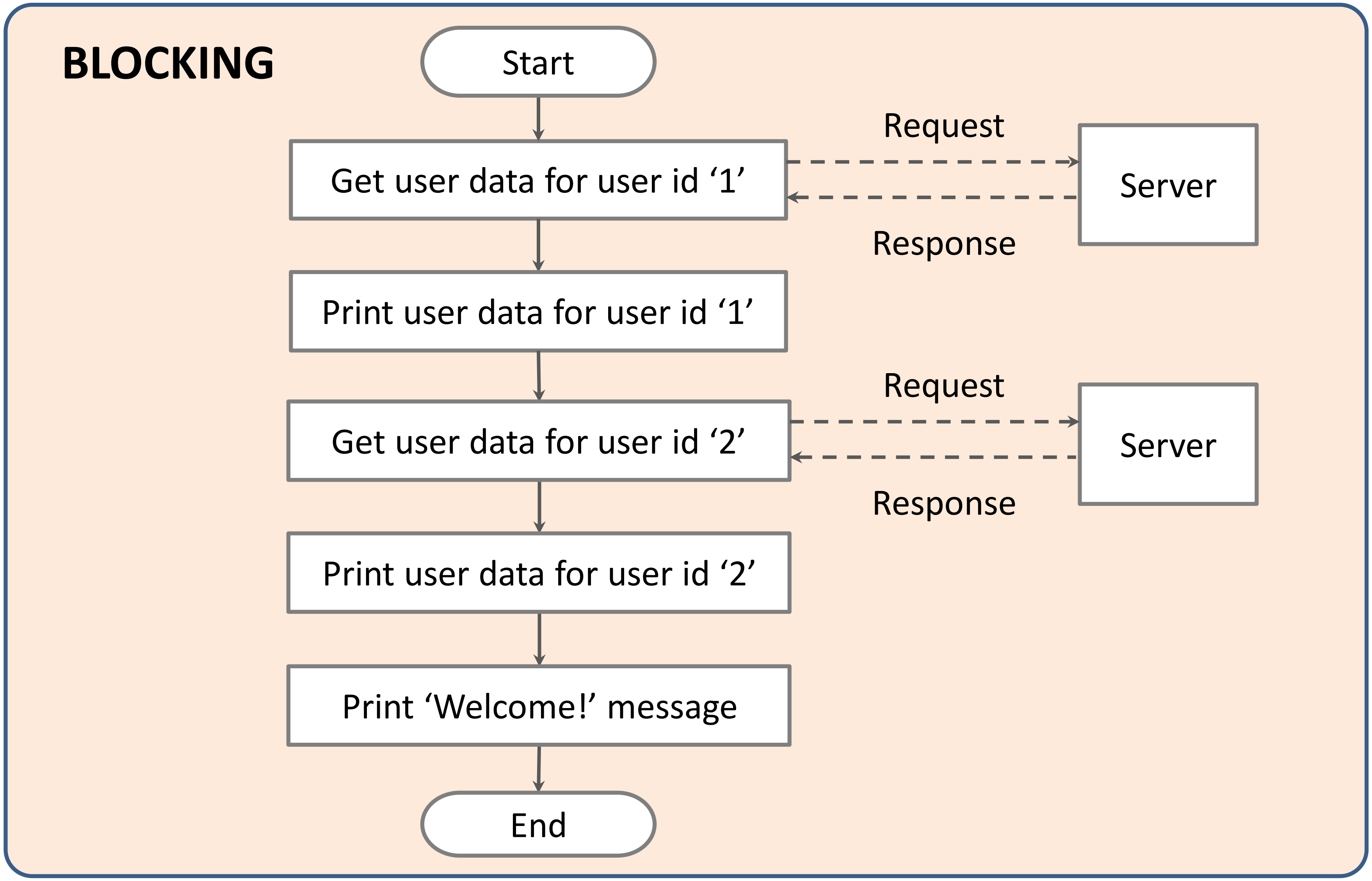
The Node.js Event Loop

- Each phase has a FIFO queue of callbacks to execute
- When the event loop enters a given phase, it will perform any operations specific to that phase, then execute callbacks in that phase's queue until the queue has been exhausted
- When the queue has been exhausted, the event loop will move to the next phase, and so on

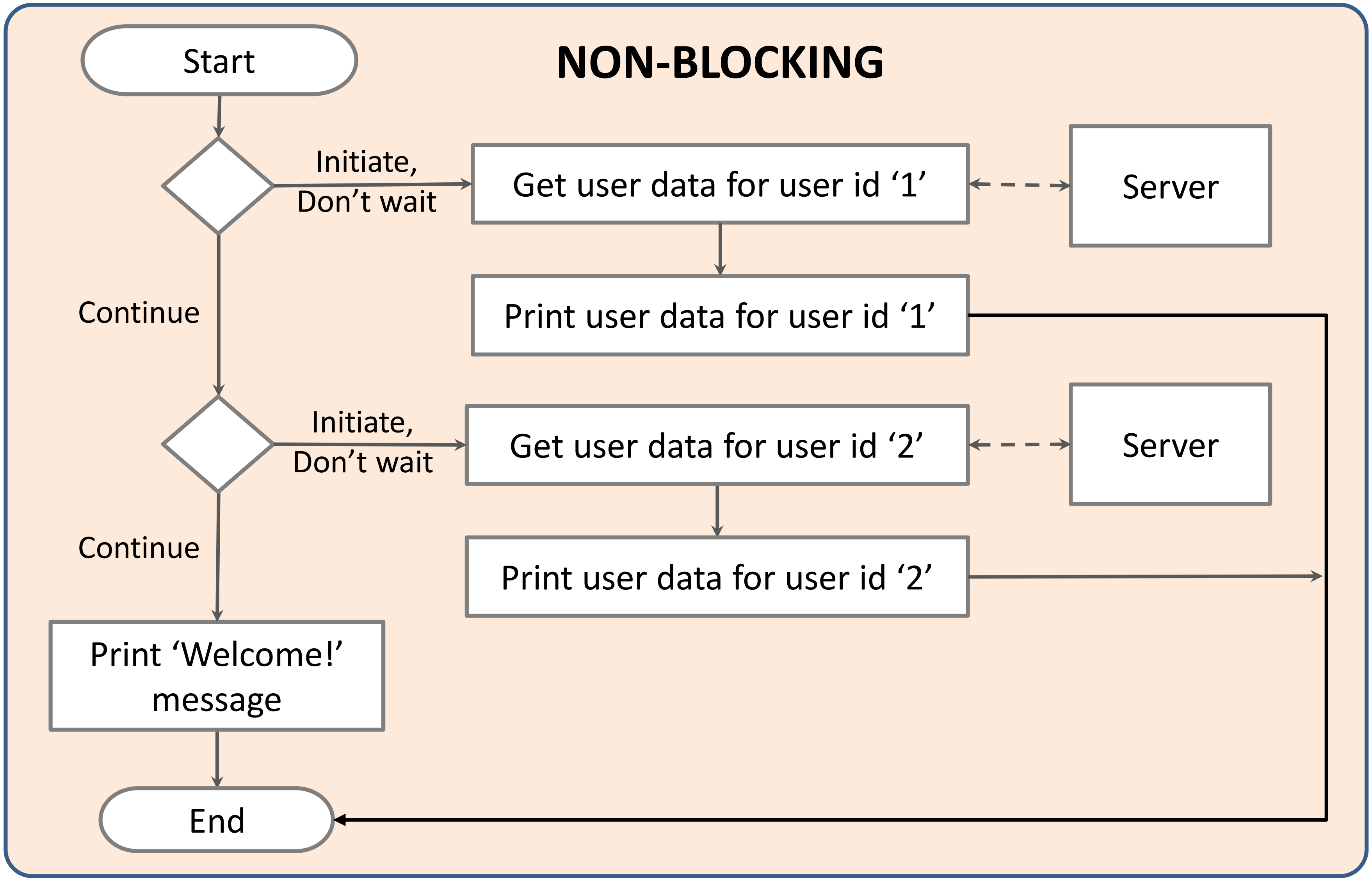
Blocking vs Non-Blocking

- Blocking
  - code executes synchronously
  - the execution of additional (successive) JavaScript code in the Node.js process must wait until an operation (previous) completes
- Non-Blocking
  - code executes asynchronously
  - method accepts a callback function
  - after the method is complete, callback function is called

Blocking vs Non-Blocking

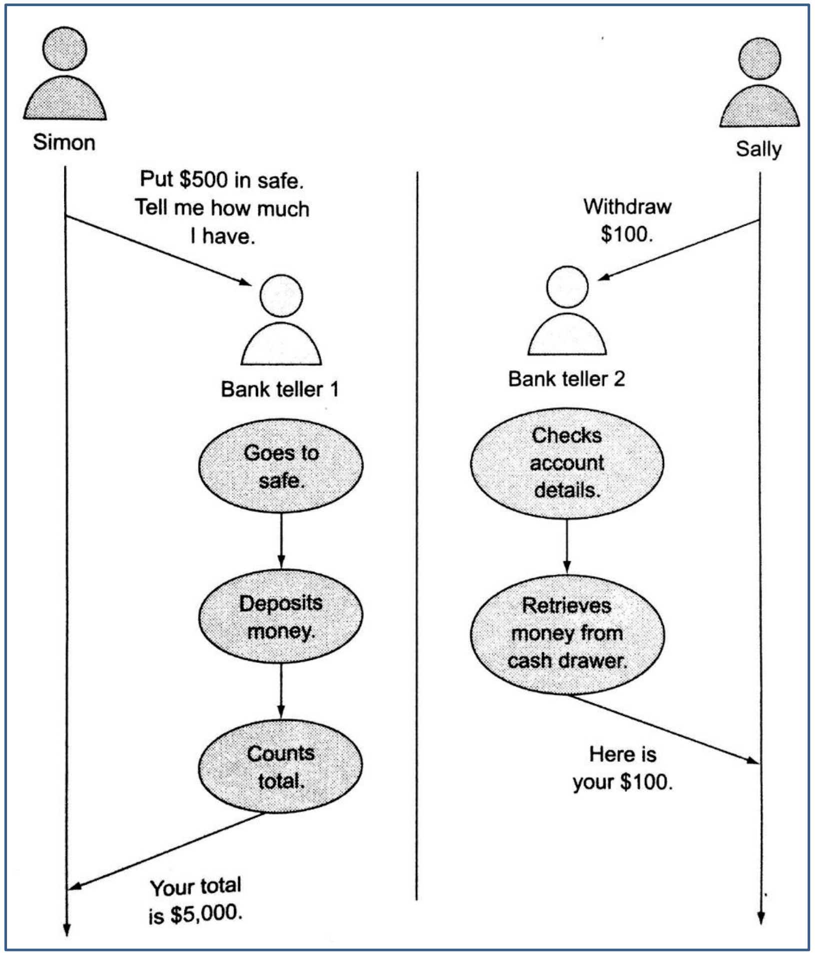

BLOCKING
Start
Get user data for user id ‘1’
Request
Server
Response
Print user data for user id ‘1’
Get user data for user id ‘2’
Request
Server
Response
Print user data for user id ‘2’
Print ‘Welcome!’ message
End

Blocking vs Non-Blocking

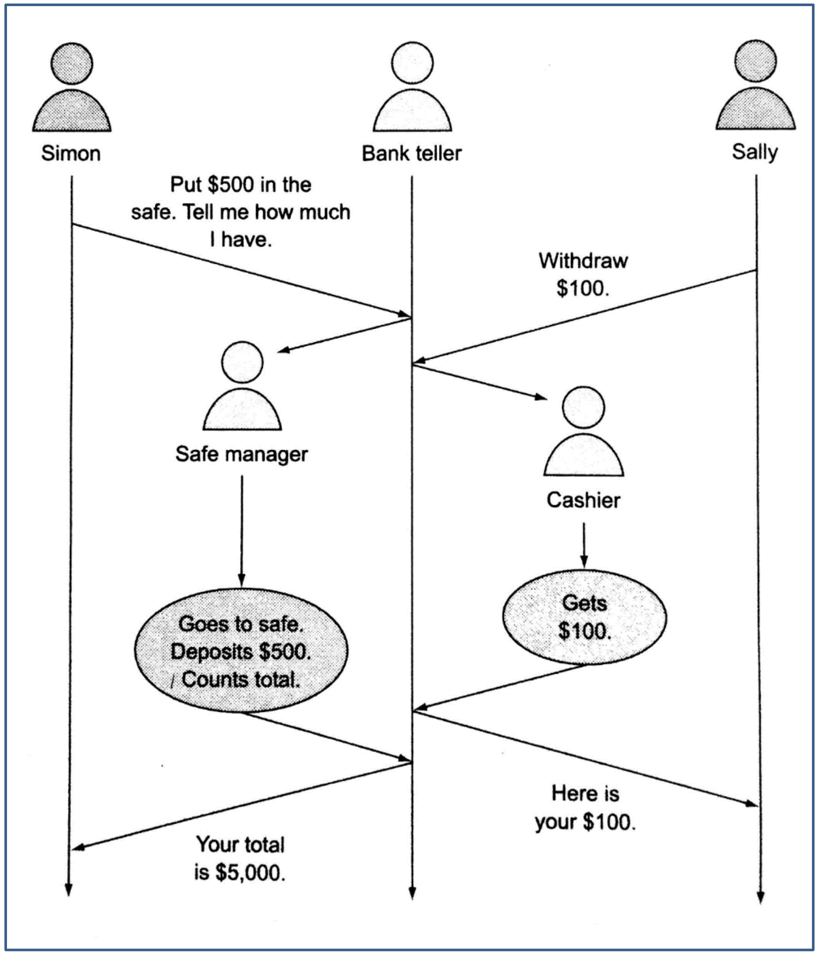

NON-BLOCKING
Start
Initiate,
Don’t wait
Get user data for user id ‘1’
Server
Print user data for user id ‘1’
Continue
Initiate,
Don’t wait
Get user data for user id ‘2’
Server
Print user data for user id ‘2’
Continue
Print ‘Welcome!’ message
End

Threading

Multi-threaded Approach

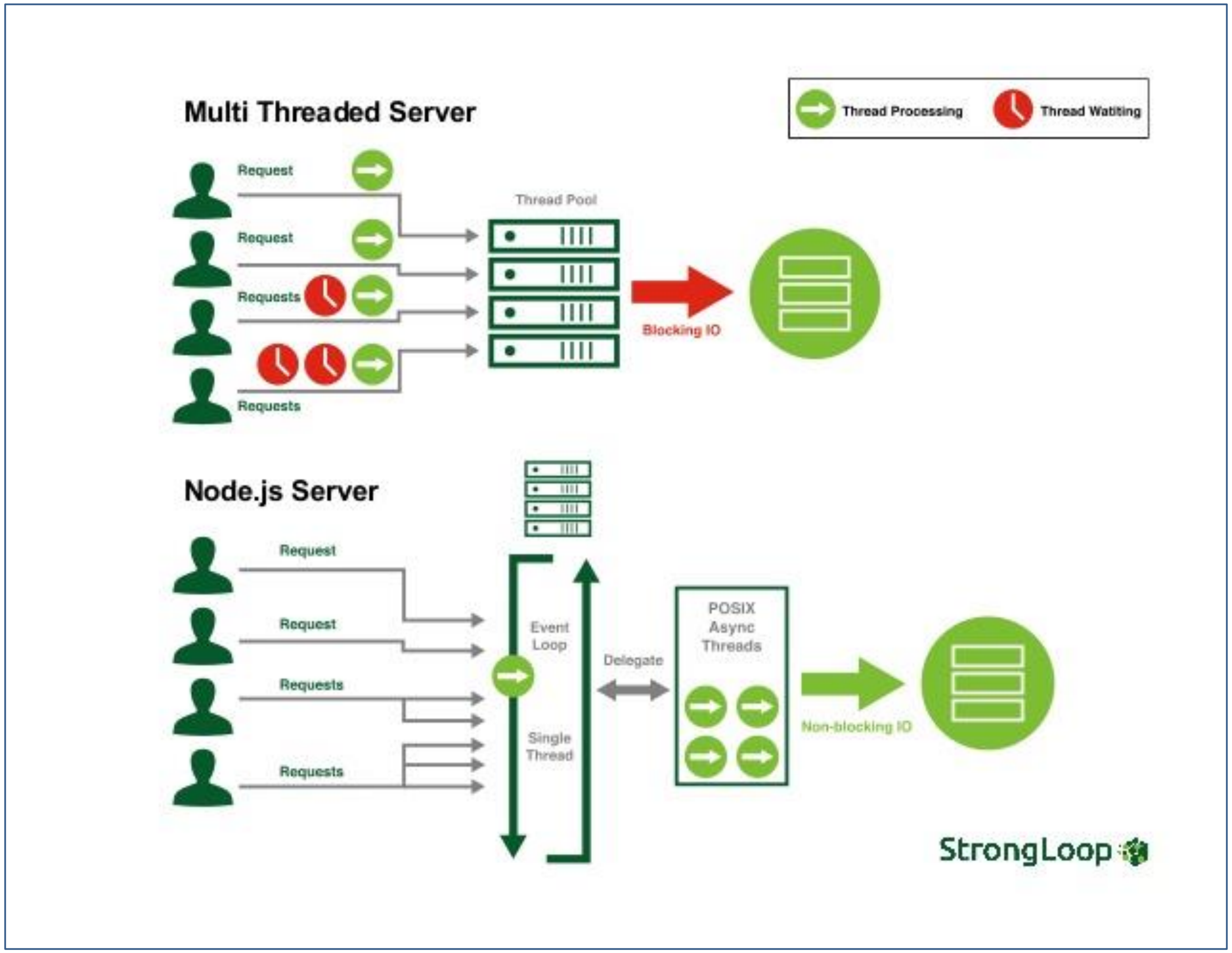


Single-threaded Approach



<u>Note:</u> JavaScript execution in Node.js is single threaded

Threading


Multi Threaded Server
Thread Processing
Thread Waiting
Request
Request
Requests
Requests
Thread Pool
Blocking IO
Node.js Server
Request
Request
Requests
Requests
Event Loop
Single Thread
Delegate
POSIX Async Threads
Non-blocking IO
StrongLoop

REPL

- REPL - Read – Eval – Print – Loop
- Provides an interactive environment for users similar to
  - Windows Command Prompt
  - Linux Shell
  - Mac Terminal
- Performs following tasks
  - Accepts individual lines of user input
  - Evaluates user input
  - Outputs the result
- Useful for experimenting with Node.js code

REPL

- Starting REPL
  - Ensure that Node.js is installed on your machine
  - In the command prompt / terminal type ‘node’ and press ‘Enter’ key
  - You will see the REPL command prompt ‘>’
  - Any Node.js command can be typed at this command prompt
- Exiting REPL
  - On a blank line, press ‘Ctrl’ + ‘C’ twice (or)
  - On a blank line, press ‘Ctrl’ + ‘D’ once (or)
  - On a blank line, type ‘.exit’ and press ‘Enter’ key

REPL

- Following special commands are supported by REPL
  - .break
    - When in the process of inputting a multi-line expression, entering the .break command (or pressing the <ctrl>-C key combination) will abort further input or processing of that expression.
  - .clear
    - Resets the REPL context to an empty object and clears any multi-line expression currently being input.
  - .exit
    - Close the I/O stream, causing the REPL to exit.
  - .help
    - Show this list of special commands.
  - .save
    - Save the current REPL session to a file: > .save ./file/to/save.js
  - .load
    - Load a file into the current REPL session. > .load ./file/to/load.js
  - .editor
    - Enter editor mode (<ctrl>-D to finish, <ctrl>-C to cancel)

REPL

- REPL commands continued…
  - Assignment of the _ (underscore) variable
    - Assign the result of the most recently evaluated expression to the special variable _ (underscore)
    - Explicitly setting _ to a value will disable this behavior
  - Up/Down Keys
    - See command history and modify previous commands.
  - Tab Key
    - When pressed on a blank line, displays global and local(scope) variables
    - When pressed while entering other input, displays relevant auto completion options

NPM

- NPM - Node Package Manager
- Allows developers to share and borrow packages
- Packages
  - extend the functionality of your app
  - promote reusability
- Consists of three distinct components
  - *The Website:* discover packages, set up profiles and manage other aspects
  - *The Registry:* database of information about packages
  - *The Command Line Interface (CLI):* runs from the terminal, gets installed with Node.js setup
- For more information, visit this link
  - https://docs.npmjs.com/getting-started/what-is-npm

NVM

- NVM - Node Version Manager
- Used to install npm
- Enables you to easily switch npm as well as Node.js versions
- Helps
  - In updating Node.js and npm
  - To test Node.js apps on multiple versions of npm
- For more information, visit these GitHub links
  - MacOS, Linux
    - https://github.com/creationix/nvm
  - Windows
    - https://github.com/coreybutler/nvm-windows

THANK YOU