Persistent

Introduction to Node.js

Persistent Interactive | Persistent University

Key Learning Points

- What is Node.js?
- Advantages of Node.js
- Server-side JavaScript
- Single threaded nature on Node.js?
- When to use Node.js?

JAVASCRIPT

Popular language in web application development

Node.js Is A Program For Running JavaScript At Server Side

Server-side JavaScript

- The first incarnations of JavaScript lived in browsers. But this is just the context.
- JavaScript is a "complete" language: you can use it in many contexts and achieve everything with it you can achieve with any other "complete" language.
- **Node.js really is just another context**: it allows you to run JavaScript code in the backend, outside a browser.

Introduction to Node.js

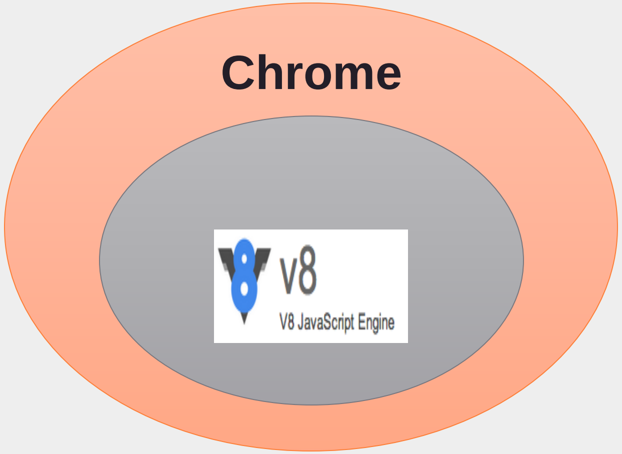
- Node.js was created by Ryan Dahl starting in 2009, and its growth is sponsored by Joyent, his employer.

- It is a command line tool that let's you run JavaScript programs.

- Node runs JavaScript, but isn’t a language of its own.

Google’s V8 engine

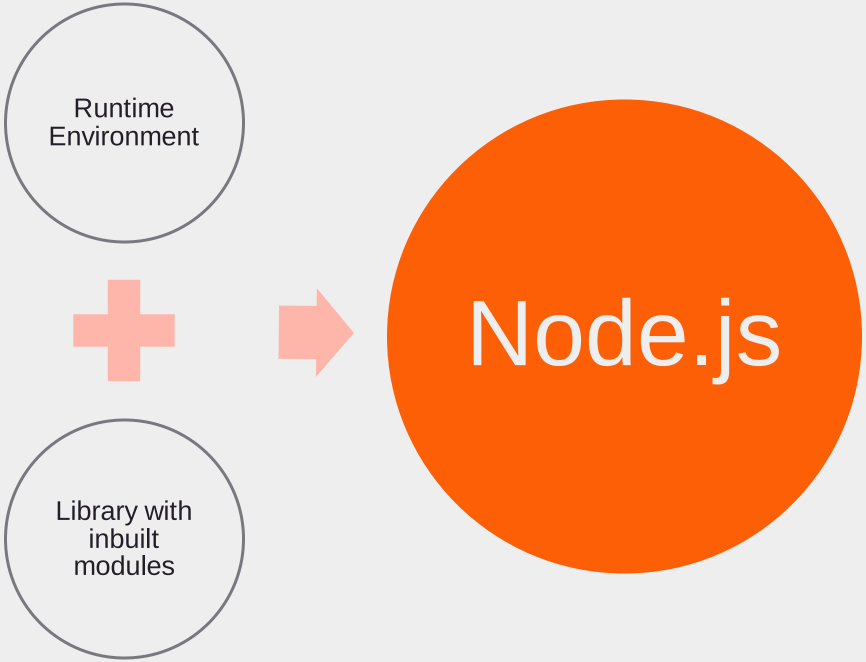
- In order to execute the JavaScript you intend to run in the backend, it needs to be interpreted and, well, executed.



- This is what Node.js does, by making use of Google's V8 VM, the same runtime environment for JavaScript that Google Chrome uses.

Node.js is really two things

- Node.js ships with a lot of useful modules, so you don't have to write everything from scratch.



Features of Node.js

Very Fast

Single Threaded

Asynchronous IO

Streams

Event Driven

Highly scalable

Node.js is Single Threaded

Analogy: Queue at a City Hospital

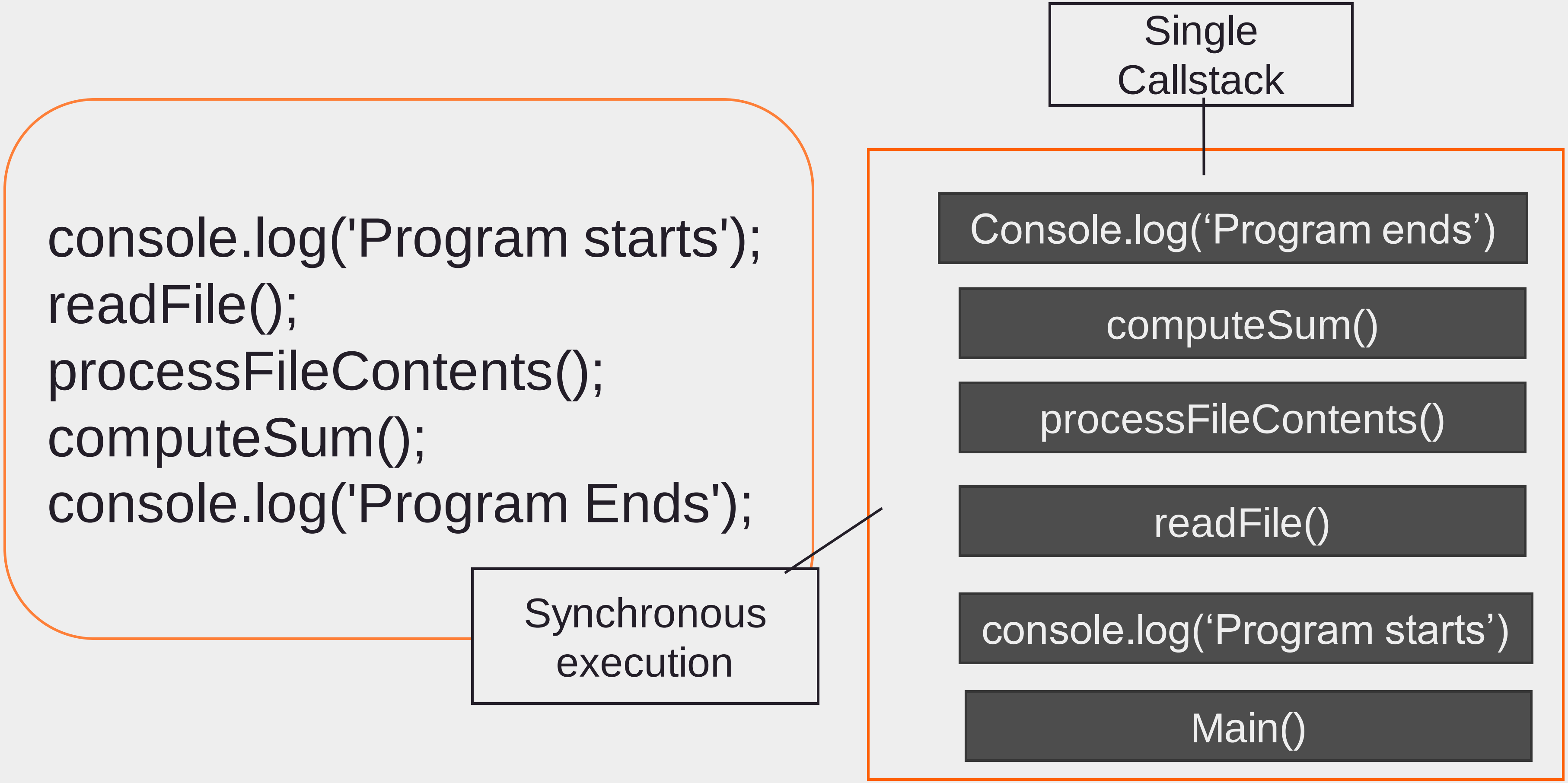
- Consider a typical queue at a city hospital.
- The people would have requests of all kind: Inquiry, visiting the doctor, taking an appointment payment etc.
- When visiting the doctor, one must fill up forms which are quite lengthy. While one is filling up the forms, the others in the queue get blocked and delayed.
- On a busy day, this can lead to long queues.
- Few solutions to this can be to add more receptionists and have separate queues. But that involves some cost.

How can it be optimized?

- Operations that are **time consuming** can be moved out of the main queue.
- When the long task is done, return back to the queue.
- Queue is single. **Same queue** for newly arrived people as well as old ones.
- Single receptionist to handle all the inquiries/requests of people.

Node.js follows a single threaded architecture.

JavaScript CallStack



```
console.log('Program starts');
readFile();
processFileContents();
computeSum();
console.log('Program Ends');
```

Synchronous execution

Single Callstack

Console.log(‘Program ends’)

computeSum()

processFileContents()

readFile()

console.log(‘Program starts’)

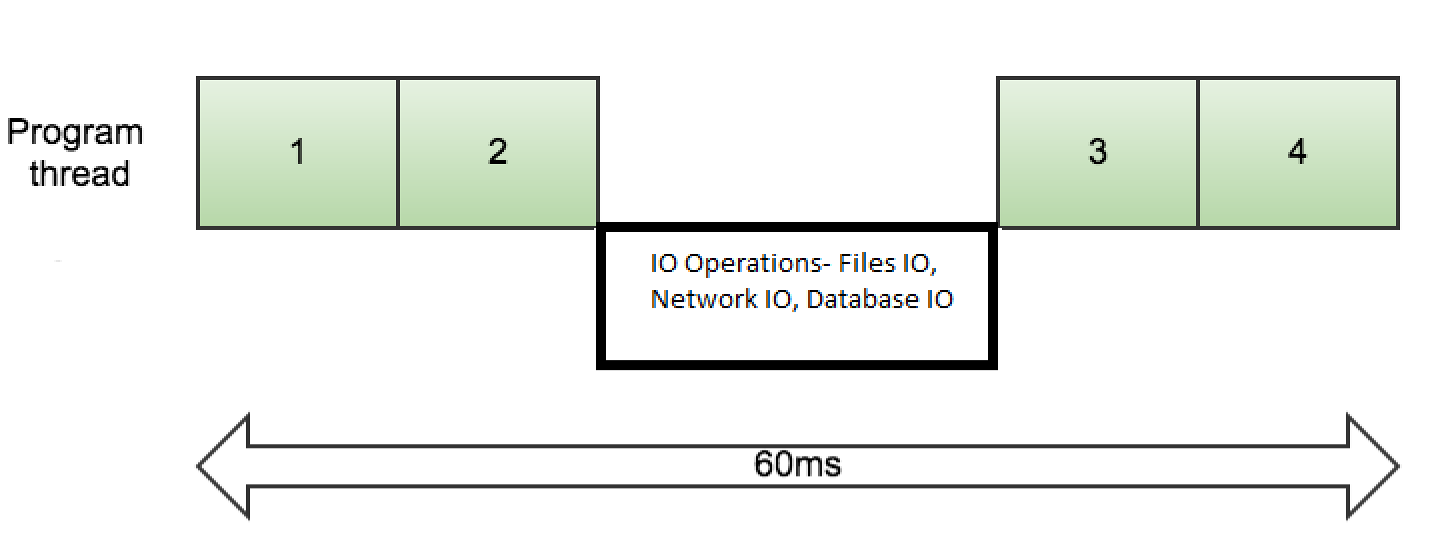
Main()

Blocking operations on Server side

- Server side code would involve lot of IO operations.
- All IO operations are Blocking.

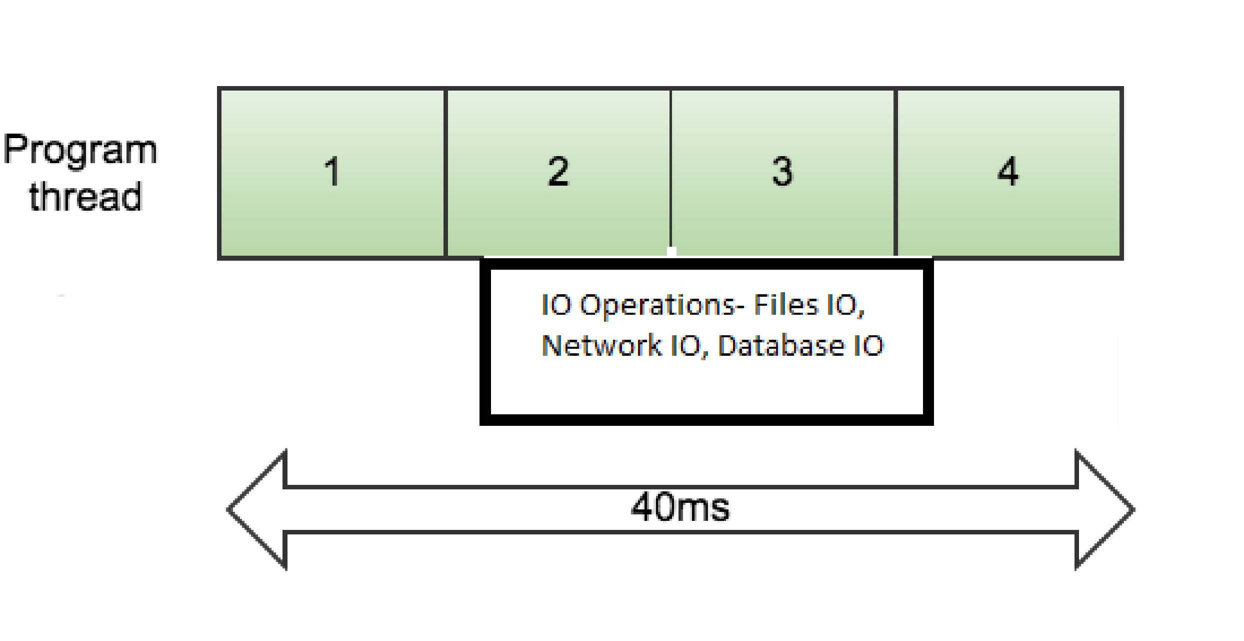
1. **File Operations**: Read/Write/Append
2. **Database Operations**: Fetch/Save/Update/Delete records
3. **Network IO:** Read/Write data over network (HTTP and Sockets)

Blocking Operations



While waiting for the I/O operation to complete, the whole process/thread is idle. This is called **Blocking code.**

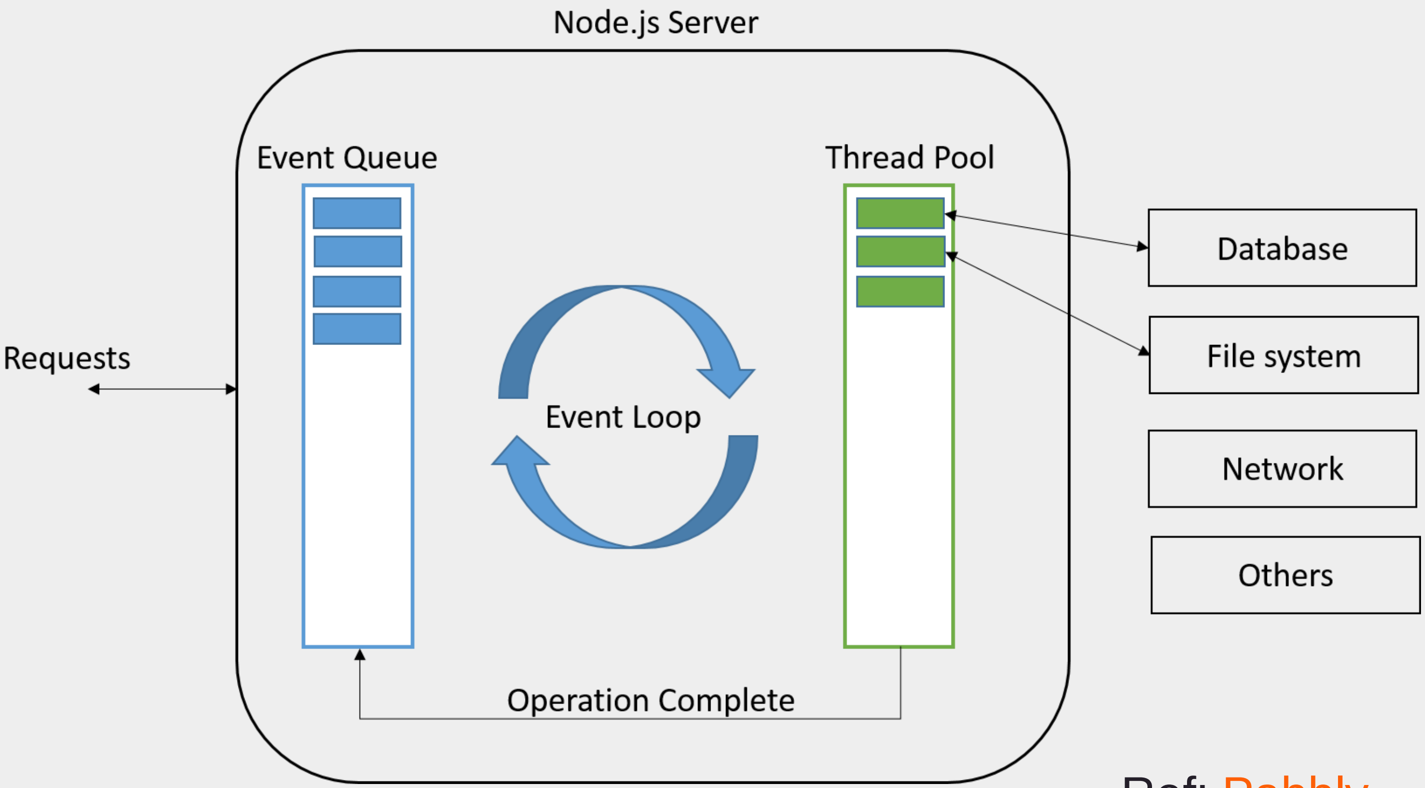
Non-Blocking IO Operations in Node.js



- All IO Operations in Node.js are Asynchronous by default.

How Node handles Asynchronous I/O operations?

IO Operations in Node



Ref: Pabbly

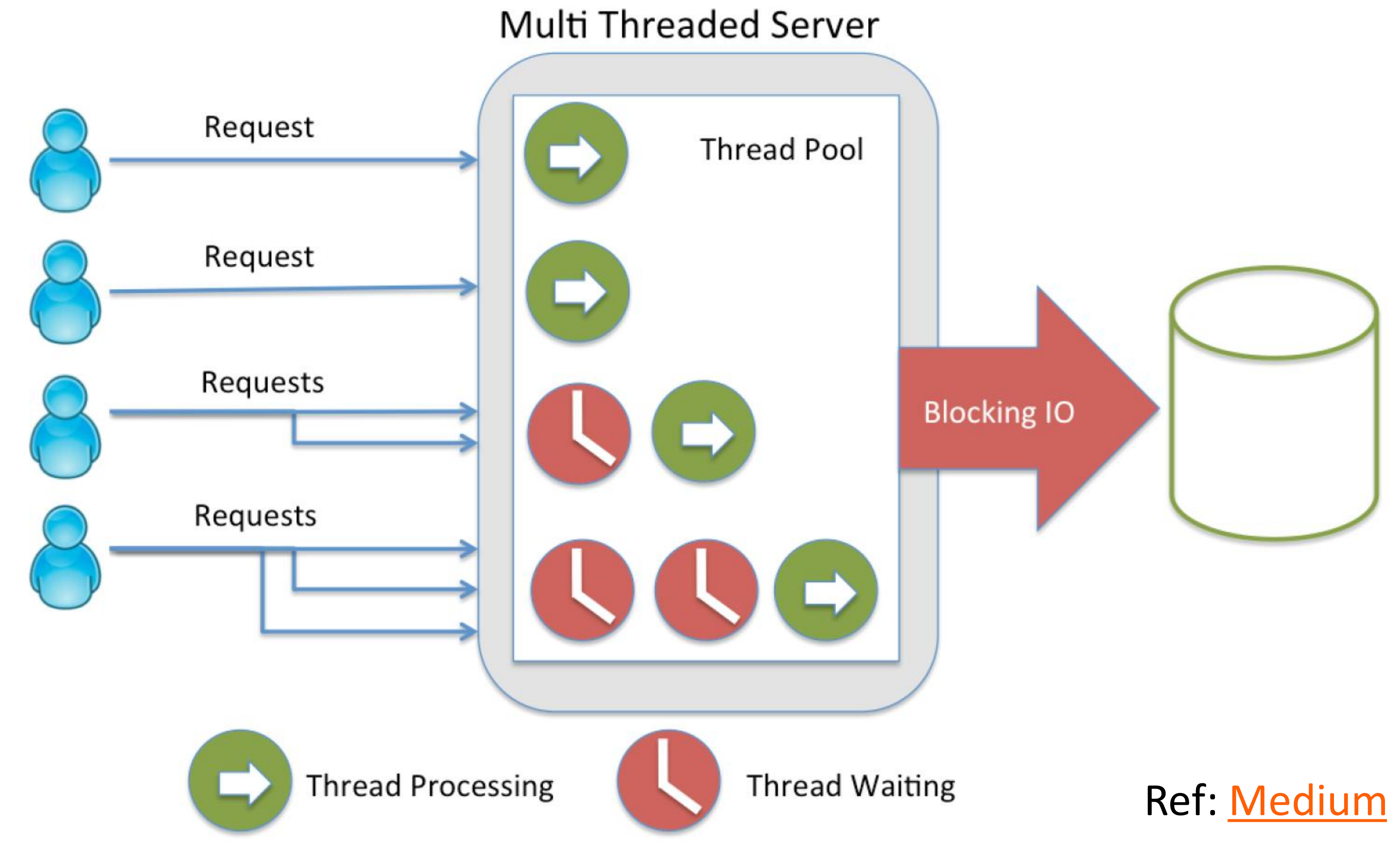
- Libuv is a library providing Node with its asynchronous I/O.

What goes to Threads?

- System calls, such as reading a file, are traditionally blocking as it waits for necessary operations at the **operating system level** to take place.
- One way of making a blocking system call non-blocking is to perform it in a separate thread. This is the solution Node.js uses for blocking system calls.
- There are non-blocking system call implementations available for some operations. They aren't widespread and available for every possible operation, but there are some at the OS level.
- It varies greatly across platforms but for some operations such as **network I/O** it is there. For such operations, Node.js can leverage non-blocking system calls directly without using worker threads.
- This is known as **Asynchronous processing model** used by Node.

How Node behaves as a Web Server?

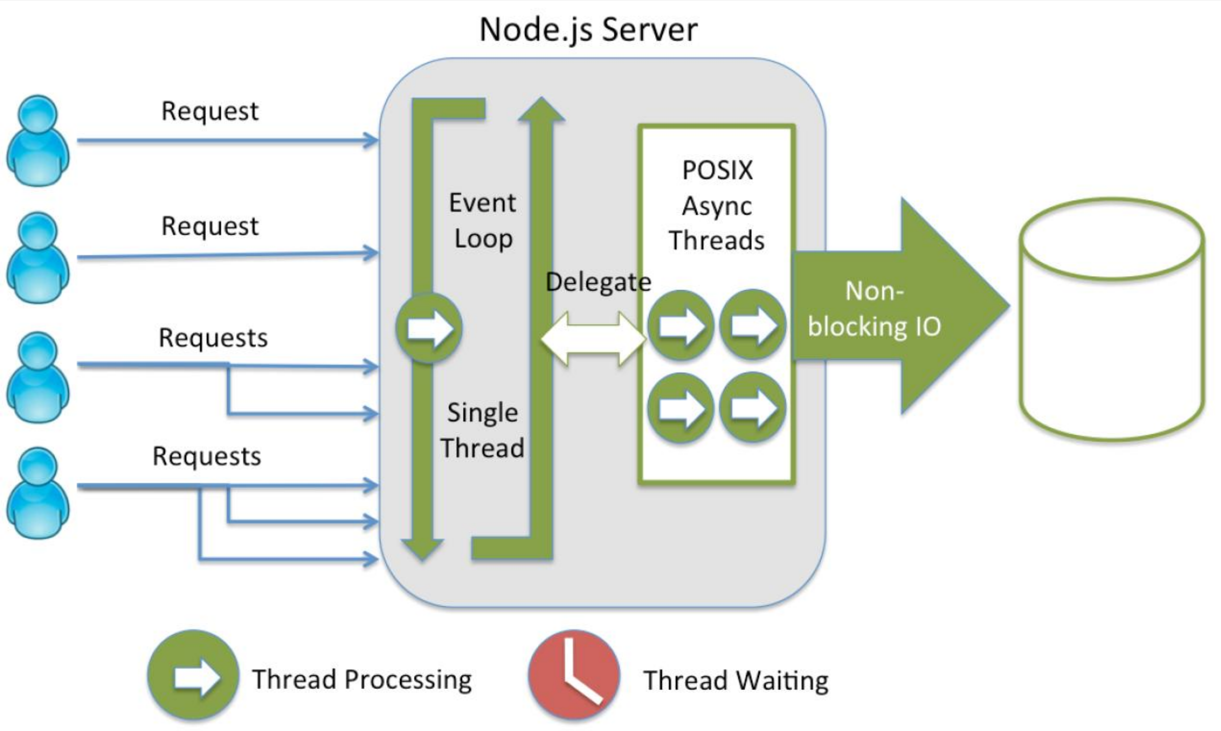
Traditional Web Applications



Ref: Medium

Node.js Server

HTTP Calls are Asynchronous in Node.js.



Ref: Medium

Comparision

- Apache (Thread-based web server)
  - Holds the connection open until the request is processed
  - the web server is ***blocking*** **on input/output operation**
  - to scale, launch additional copies of the server

- Node.js (Event-based)
  - the web server accepts the request, spins off I/O to be handled
  - *service the **next** web request*
  - When the original request is completed, it gets back in the processing queue
  - when it reaches the front of the queue the results are sent back

Use Cases of Node.js

When to use Node.js?

- What it really means is that Node.js is not a silver-bullet new platform that will dominate the web development world. Instead, it’s a platform that fills a specific need.

- Where Node really shines is in building I/O bound applications, data intensive real time applications scalable network applications, as it’s capable of handling a huge number of simultaneous connections with high throughput, which equates to high scalability.

- You definitely don’t want to use Node.js for CPU-intensive operations; in fact, using it for heavy computation will annul nearly all of its advantages.

Node.js Use cases

- Web Application Server
- RESTful server
- Chat Server
- Streaming Applications
- MEAN Stack

Who all uses Node.js?

- Netflix
- Trello
- PayPal
- LinkedIn
- Walmart
- Uber
- Ebay

Advantages of Node.js

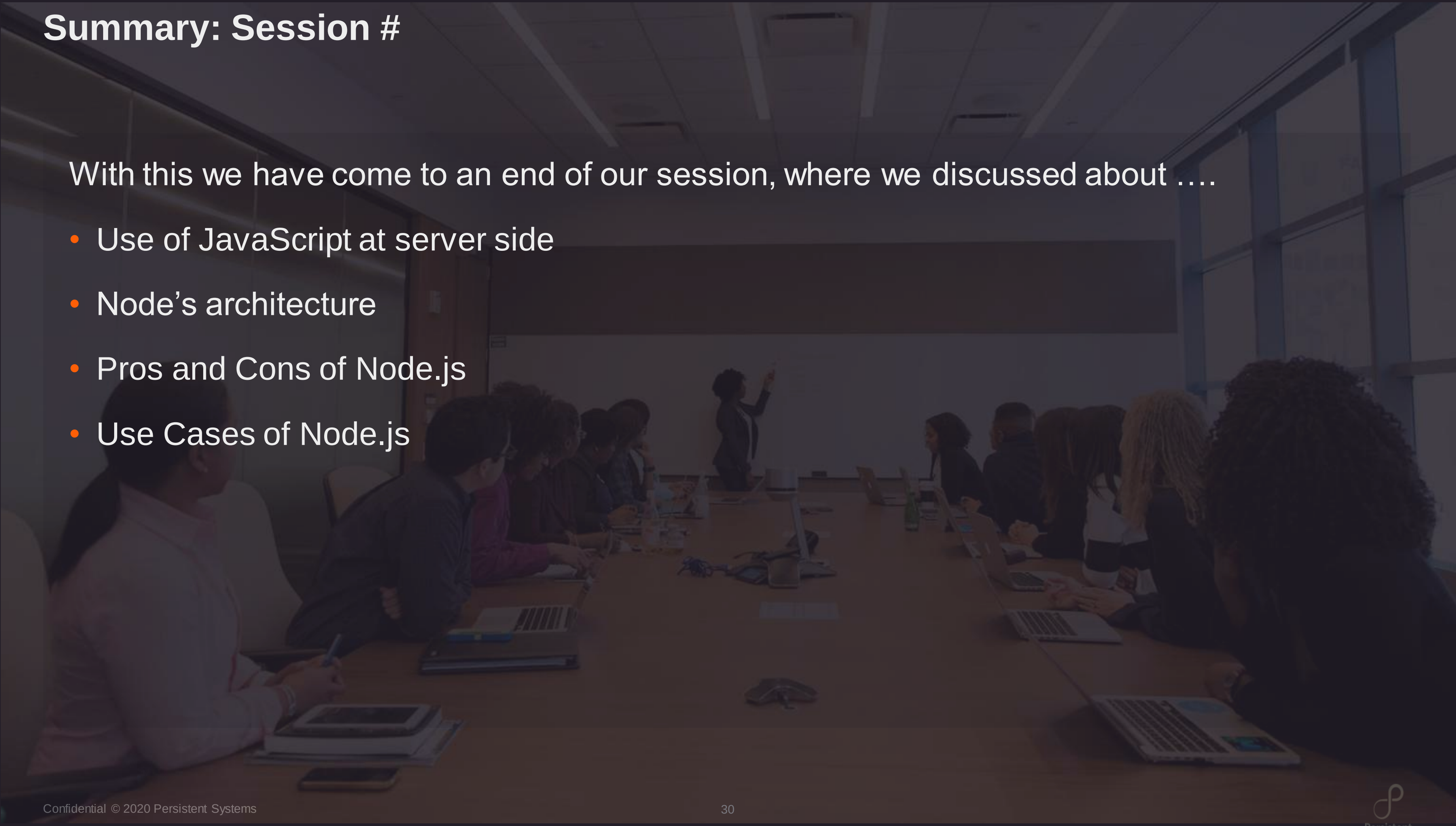
- **Raw speed:** V8 is constantly pushing the boundaries in being one of the fastest dynamic language interpreters. In addition to that, node's I/O facilities are really light weight, bringing you as close to fully utilizing your system's full I/O capacities as possible.
- **Efficiency**: In a web application, your main response time cost is usually the sum of time it takes to execute all your database queries. With node, you can execute all your queries at once, reducing the response time to the duration it takes to execute the slowest query.
- **JavaScript:** You can use node to share code between the browser and your backend. **JSON data** format is common.

Software Setup

- Node.js Installation
- Visual Studio Code Editor
- MongoDb Installation
- Postman Client
- Online Editor: repl

- https://repl.it/

Summary: Session



With this we have come to an end of our session, where we discussed about ….

- Use of JavaScript at server side
- Node’s architecture
- Pros and Cons of Node.js
- Use Cases of Node.js

Appendix

References

Thank You

Some Good Pointers

- Node Examples
  - http://amirrajan.net/nodejs-by-example/
- Good Node Tutorial
  - http://nodeguide.com/
  - http://nodetuts.com/pdf/handson-nodejs-sample.pdf
  - http://nodebeginner.org/
  - https://www.airpair.com/javascript/node-js-tutorial
- Why use Node for CPU bound tasks?
  - http://neilk.net/blog/2013/04/30/why-you-should-use-nodejs-for-CPU-bound-tasks/
- Who uses Node.js and Why?
  - https://businessingmag.com/8371/equipping/node-js/

Some Good Pointers

- Node is not single threaded.
  - http://www.journaldev.com/7462/node-js-processing-model-single-threaded-model-with-event-loop-architecture
  - http://rickgaribay.net/archive/2012/01/28/node-is-not-single-threaded.aspx
  - http://stackoverflow.com/questions/22644328/when-is-the-thread-pool-used
  - http://stackoverflow.com/questions/3629784/how-is-node-js-inherently-faster-when-it-still-relies-on-threads-internally
  - http://stackoverflow.com/questions/14795145/how-the-single-threaded-non-blocking-io-model-works-in-node-js
- Thread pool in Node
  - http://stackoverflow.com/questions/22644328/when-is-the-thread-pool-used
- Good read on Node
  - http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

Key Contacts :

Persistent University :

- Archana Ghatigar

  archana_ghatigar@persistent.co.in

Thank you!

Persistent Interactive | Persistent University