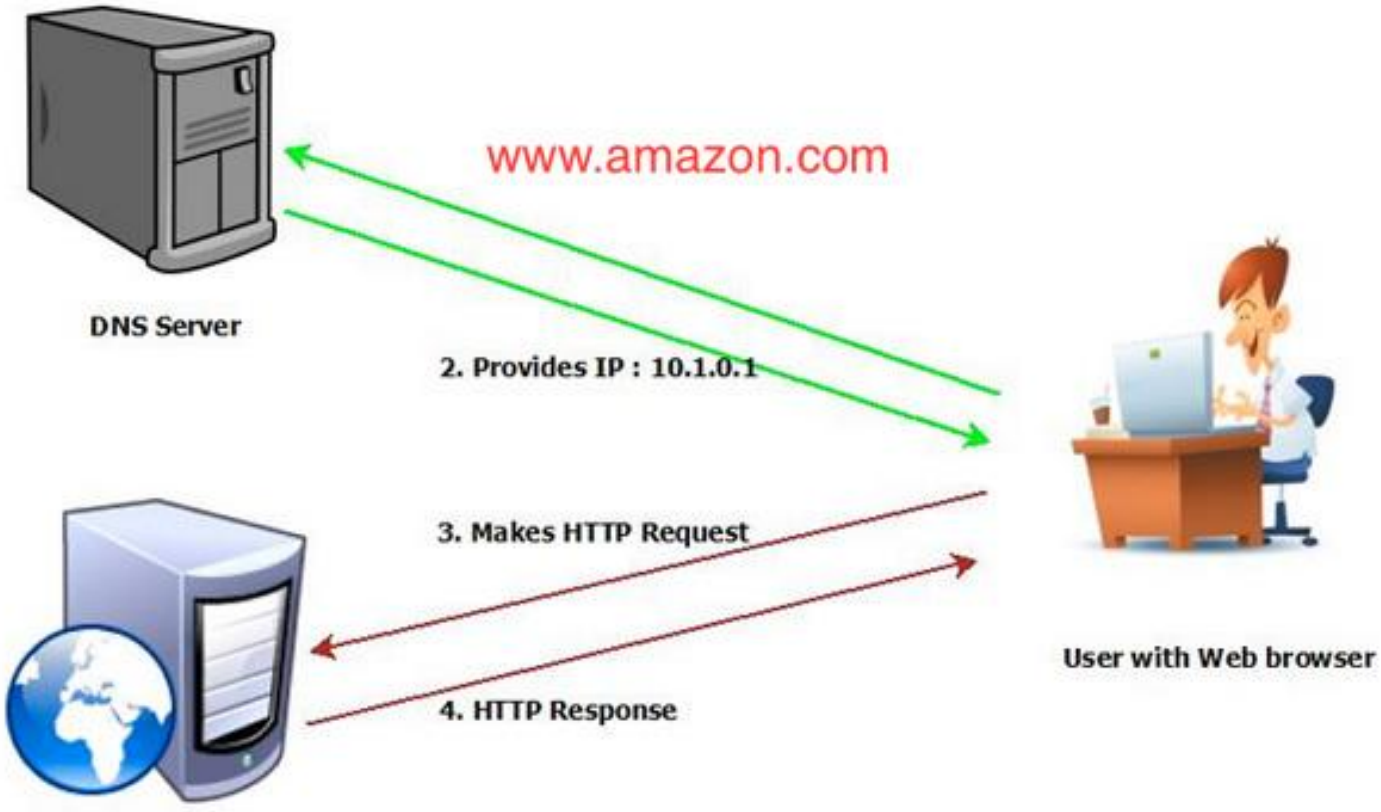
Node js
Lecture 1

By Mona Soliman

Agenda

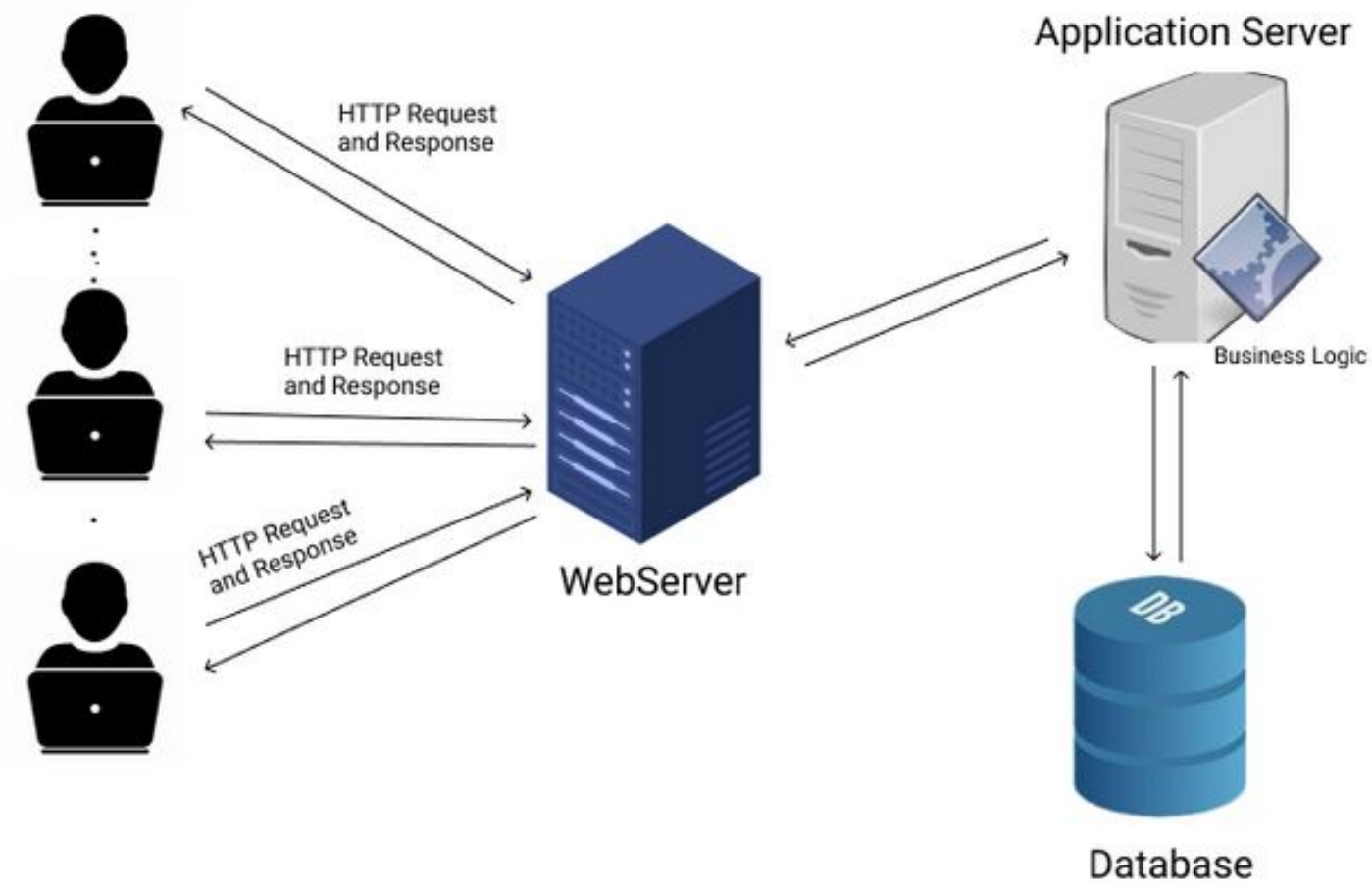
- Web
- What is Node.js
- Node.js vs Browser
- ENV variable, global, REPL, CLI Args
- Core Modules(fs)


www.amazon.com
DNS Server
2. Provides IP : 10.1.0.1
3. Makes HTTP Request
4. HTTP Response
User with Web browser

HTTP URL Anatomy

Key

1. Scheme – defines how the resource will be obtained.
2. Subdomain – www is most common but not required.
3. Domain – unique value within its top-level domain.
4. Top-level Domain – hundreds of options now exist.
5. Port – if omitted HTTP will connect on port 80, HTTPS on 443.
6. Path – specify and perhaps find requested resource.
7. Query String – data passed to server-side software, if present.
8. Fragment Identifier – a specific place within an HTML document.


Application Server
HTTP Request
and Response
HTTP Request
and Response
HTTP Request
and Response
Business Logic
WebServer
DB
Database

Node is a runtime environment for executing JS code.it’s not a programming language.Node is a C++ program that embeds Chrome’s v8 engine, the fastest JS engine in the world.

Node created by Ryan Dahl in 2009.

its an Open Source, Cross-platform, runtime environment for server-side and a high performance networking applications.

Popular companies using node js: Netflix,Nasa,Trello,Paypal,Linkedin ,Uber,....

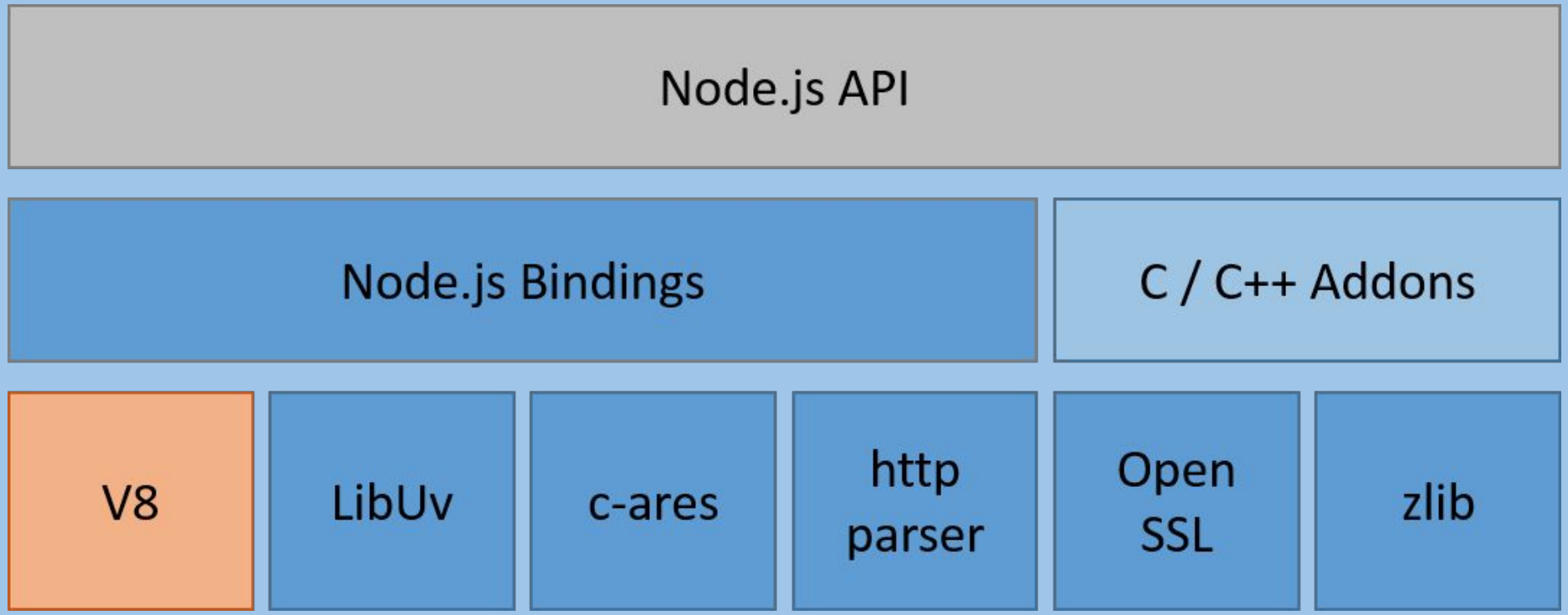
Last version :20.5.0

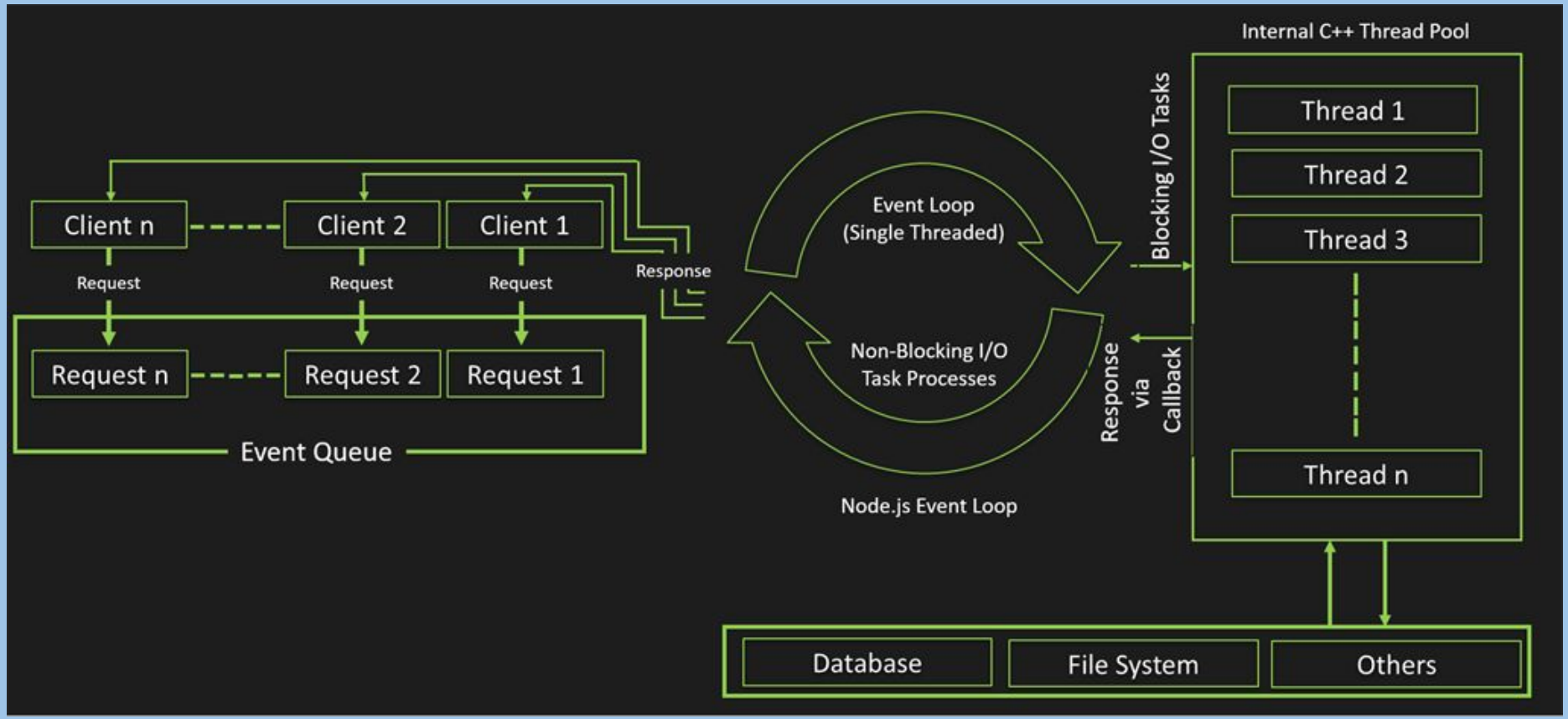
Node js:

Node.js uses an event-driven, non-blocking I/O model, which makes it lightweight.

Node js is single thread (single threaded but highly scalable).

In Node, we don’t have browser environment objects such as window or the document object. Instead, we have other objects that are not available in browsers, such as objects for working with the file system, network, operating system, etc


Node.js API
Node.js Bindings
C / C++ Addons
V8
LibUv
c-ares
http parser
Open SSL
zlib


Client n
Client 2
Client 1
Request
Request
Request
Response
Request n
Request 2
Request 1
Event Queue
Event Loop
(Single Threaded)
Non-Blocking I/O
Task Processes
Node.js Event Loop
Blocking I/O Tasks
Response
via
Callback
Internal C++ Thread Pool
Thread 1
Thread 2
Thread 3
Thread n
Database
File System
Others

Blocking vs non-Blocking

Consider a scenario where we request a backend database for the details of user1 and user2 and then print them on the screen/console. The response to this request takes time, but both of the user data requests can be carried out independently and at the same time

When to use node js?

- Node.js is good for creating streaming based real-time services, web chat applications, static file servers etc .

- If your server side code requires very few cpu cycles. In other words you are doing non blocking operation and does not have heavy algorithm/Job which consumes lots of CPU cycles.

- Following are the areas where Node.js is perfect technology:

  -I/O bound Applications
  -Data Streaming Applications
  -Data Intensive Real time Applications (DIRT)
  -JSON APIs based Applications

When not to use node js?

- When you are doing heavy and CPU intensive calculations on server side, because event-loops are CPU hungry , like : images manipulating , video conversion or file compression

Getting Started

- Install node js: https://nodejs.org/en/download/
- After downloading it, you have to install it.
- Enter your folder and run your app : node (entry point)

Intro to Modules :

- Module is a self contained series of one or more .js files presented by an object
- Modules is where we can encapsulate related functionality into a single file
- Modules act as libaries
- It’s a set reusable code that adds extra functionality to our application

Node provides core modules that can be included by their name:

- File System - require(‘fs’)
- Http - require(‘http’)
- Utilities - require(‘util’)

FS Module:

```
//file is already created
//this happens Asynch-->non blocking
var fs=require("fs");
console.log("starting");

fs.readFile("readingFile.txt",function(err,data){
    console.log("this is a new way to read");
    console.log("content: "+data)

});
console.log("exec");
```

```
// to make it synch -->blocking
var fs=require("fs");
console.log("starting");

var data=fs.readFileSync("readingFile.txt");
console.log("this is a new way to read");
console.log("content: "+data);
console.log("exec");
```

JS in browser and Node.js
Node REPL
Node Global Objects
Environment Variables
CLI argv

Thank you