NodeJS

Overview of Module

NodeJS

browser architecture

Web architecture, backend, frontend

- Fundamentals

v8, libuv, C-Areas, modules

- NodeJS Architecture

arrow, rest, spread, async, await, array...

- JS Fundamentals

modular application design

- Modules (Reusability)

connection between front & backend

- REST Architecture

blocking and non-blocking api

- Sync vs Async

Backend

Express JS

web server → REST APIs

→ functions

- Introduction

JS modules - scalable

- Modular Architecture

mysql connectivity, mongo

- Database Connectivity

confidentiality tracing / debbugging

- Emails, Encryption, Logging

- File Upload, Authentication

Frontend

React JS

SPA vs MPA, comparison

- Introduction

React pages → UI, JS + html = JSX

- JSX fundamentals ***

Real DOM, virtual DOM

- ReactJS Architecture **

reusable units communicating with components

- Components, Props, State

useState, useEffect → navigation → axios

- Hooks, Router, REST

cloud deployment

- Deployment

About Instructor

- 17+ years of experience
- Associate Technical Director at Sunbeam
- Freelance Developer
- Worked in various domains using different technologies
- Developed 180+ mobile applications on iOS and Android platforms
- Developed various websites using PHP, MEAN and MERN stacks
- Languages I love: C, C++, Python, JavaScript, TypeScript, PHP, Go, Rust

let num = 200 → number

num + 300 = 500

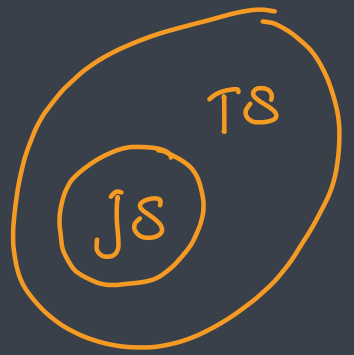
num = "test" → string

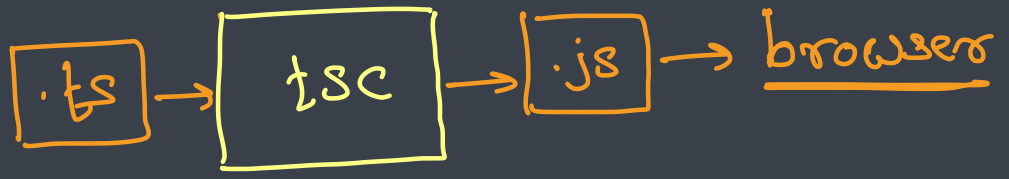
num + 300 = test300

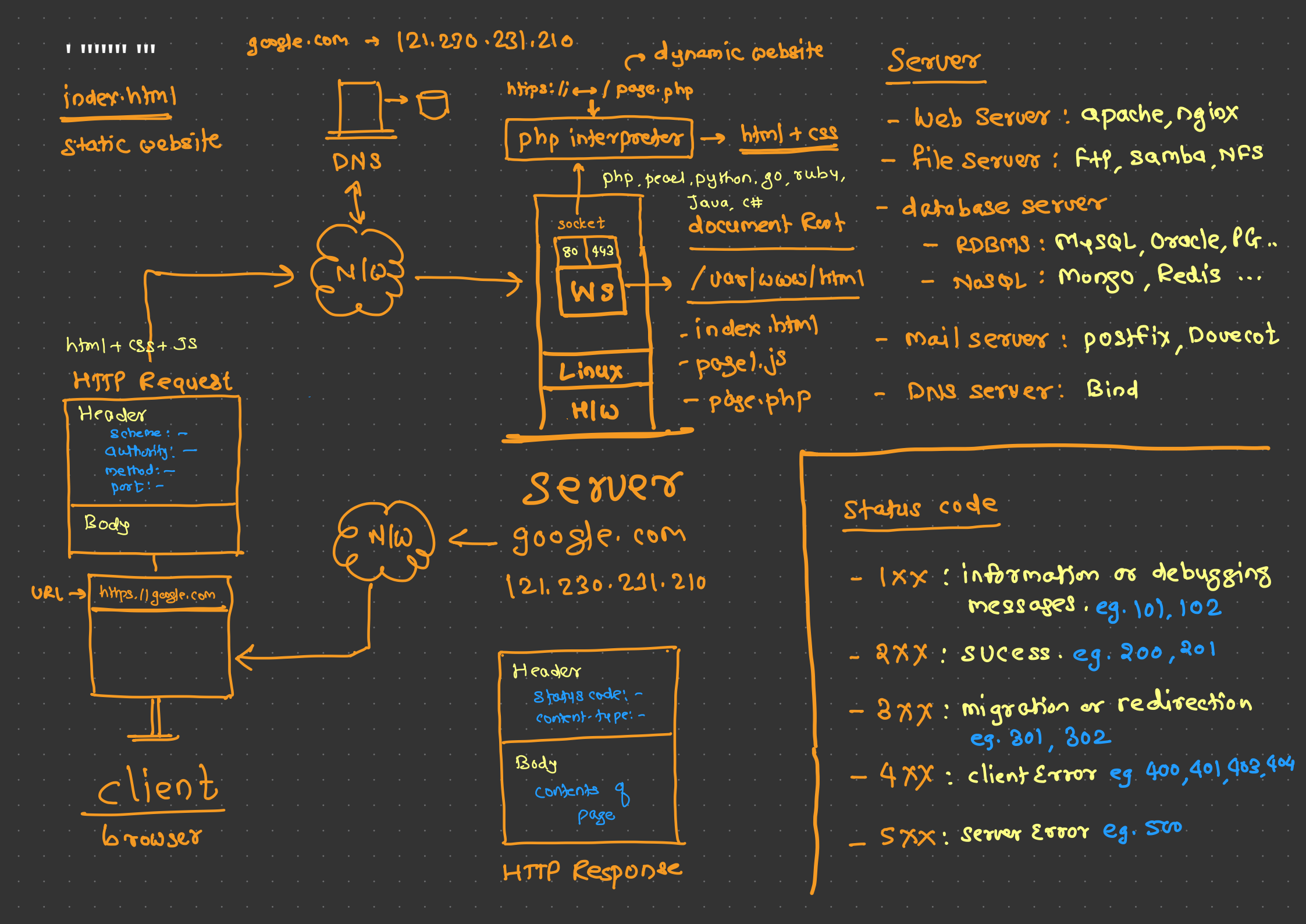
Fundamentals

Web Architecture

- Web uses client server architecture
- Client
  - The machine which uses the browser to browse a website
  - Client machine sends a request to the server demanding the required resource
- Server
  - The machine which runs a server program
  - The server program serves a request
  - Types
    - Web server
    - File server
    - Database server






google.com → 121.230.231.210
index.html
static website
dynamic website
https://page.php
php interpreter
html + css
DNS
php, perl, python, go, ruby, Java, c#
socket
80
443
WS
document Root
/var/www/html
- index.html
- page1.js
- page.php
Linux
HW
NW
html + CSS + JS
HTTP Request
Header
scheme: -
authority: -
method: -
port: -
Body
URL
https://google.com
client
browser
NW
server
google.com
121.230.231.210
Header
status code: -
content-type: -
Body
contents of page
HTTP Response
Server
- Web Server : apache, nginx
- File Server : ftp, samba, NFS
- database server
- RDBMS : MySQL, Oracle, PG..
- NoSQL : Mongo, Redis ...
- mail server : postfix, Dovecot
- DNS server: Bind
Status code
- 1XX : information or debugging messages. eg. 101, 102
- 2XX : sucess. eg. 200, 201
- 3XX : migration or redirection eg. 301, 302
- 4XX : client Error eg 400, 401, 403, 404
- 5XX : server Error eg. 500

URL

→ irrespective of os & browser, it is always same

→ any file hosted / copied on server

- Uniform Resource Locator
- Used to locate any resource (any file hosted on the server) uniquely
- Components
  - Scheme : http, https, ftp, tftp, file [protocol]
  - Domain name or IP address
  - Port Number (of web server)
  - Path or file name → http://localhost:4000/employee
  - Query String : used for passing input(s) to the page → https://google.co.in/?q=iphone
  - Proxy (hash) component → https://apple.com/article23.html#top

https://google.co.in

scheme | domain name

http://172.2.3.4

IP address

http://localhost:4000

port no

http — mandatory

http: 80, https: 443

file or path

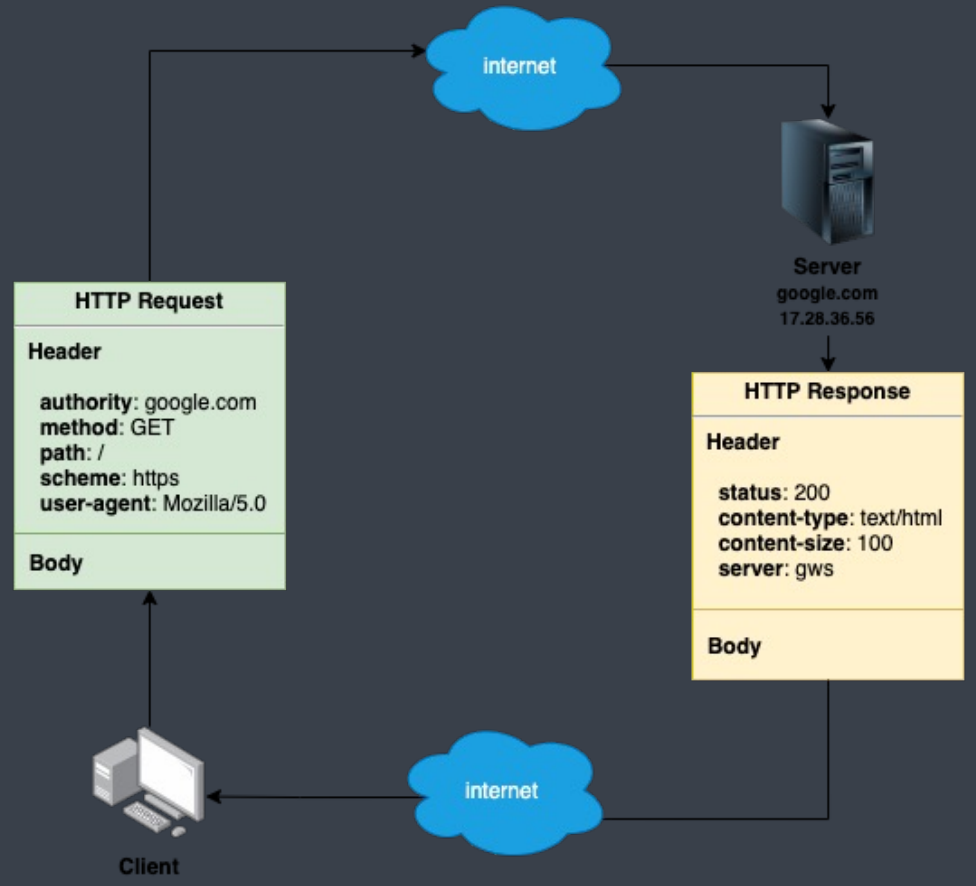
Key → q = iphone ← value

query string

bash / proxy

Web Architecture

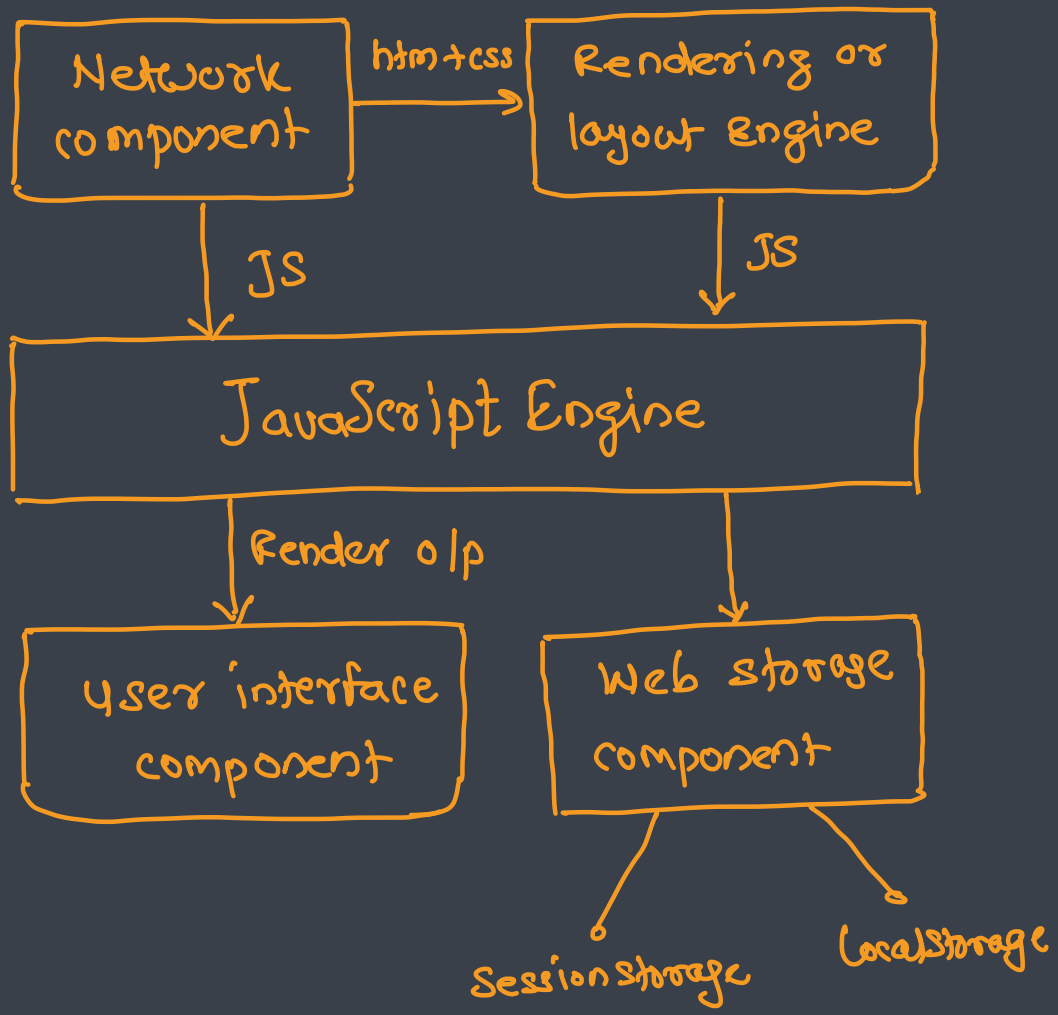
- It uses request-response pattern
- **Request**
  - An object created by the client (browser)
  - Contains all the information required by the server to send the response
  - Header contains
    - Method, authority, parameters, user-agent etc.
  - Body contains the body parameters
- **Response**
  - An object created by the server which contains the requested resource
  - Header contains
    - Status code, server type, content-type, content size
  - Body contains the requested resource



Browser → http client

- A special application used to browse the websites
- It contains parsers for HTML, XML, CSS, Text etc. which are used to parse HTML and CSS and can produce the UI
- It executes the JS code using JS engine
- Any browser contains
  - User interface component
  - Browser Engine (JS Engine)
  - Rendering engine
  - Networking component
  - JS interpreter / JS Engine
  - Data persistence component

browser architecture



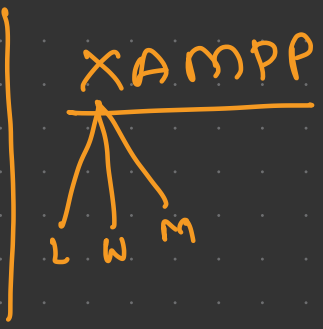
	Rendering Engine	Javascript Engine
① firefox	Gecko	SpiderMonkey
② Safari	WebKit	JavaScriptOne or NitroJS
③ Opera	blink	—
④ Edge	EdgeHTML	Chakra → chromium
⑤ chrome	blink	V8

Stack → platform + web server + frontend + db

① MERN →

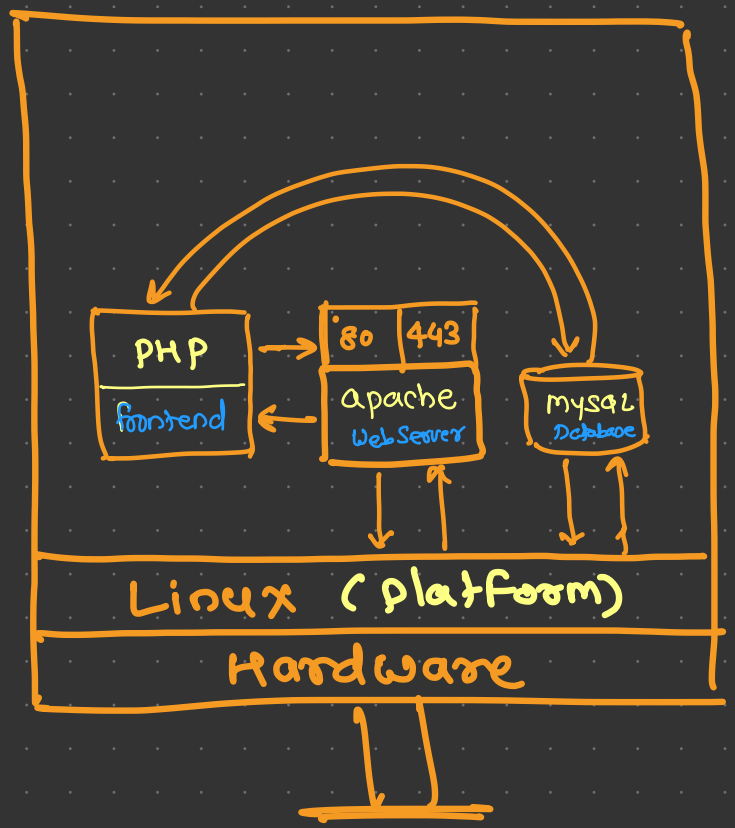
② MEAN → A → Angular

③ AMP → Apache + MySQL + PHP/python/Perl
LAMP → Linux
WAMP → Windows
MAMP → macOS

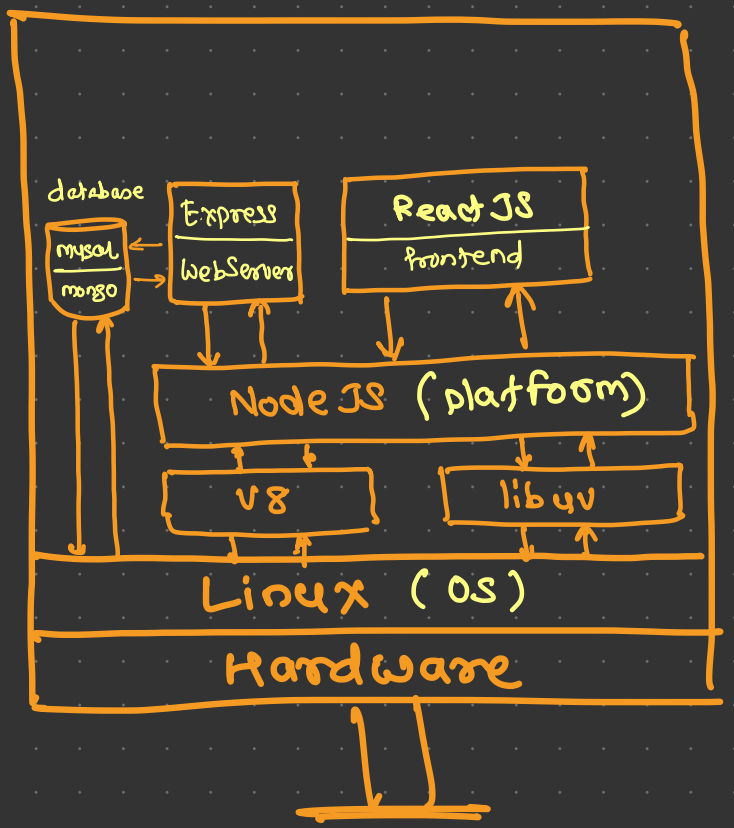


④ WISA → Windows + IIS server + SSL server + Asp.x

LAMP


PHP
frontend
80
443
apache
Web Server
mysql
Database
Linux (platform)
Hardware

MERN


database
mysql
mongo
Express
WebServers
React JS
frontend
Node JS (platform)
V8
libuv
Linux (OS)
Hardware

M → MySql / Mongo → database

E → Express → backend

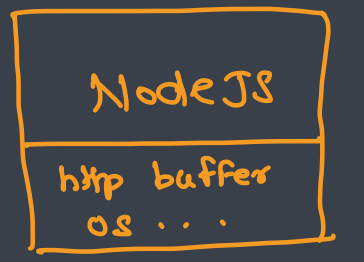
R → React → frontend

N → Node → platform

NodeJS

Rhyan Dahl

2009



What is Node ?

- Node.js is an open-source and cross-platform JavaScript runtime environment
- It is a platform to develop server sided and networking applications in JS
- Developed Ryan Dahl in 2009
- It is free and open source
- It can run on various Operating Systems like macOS, Linux and Windows

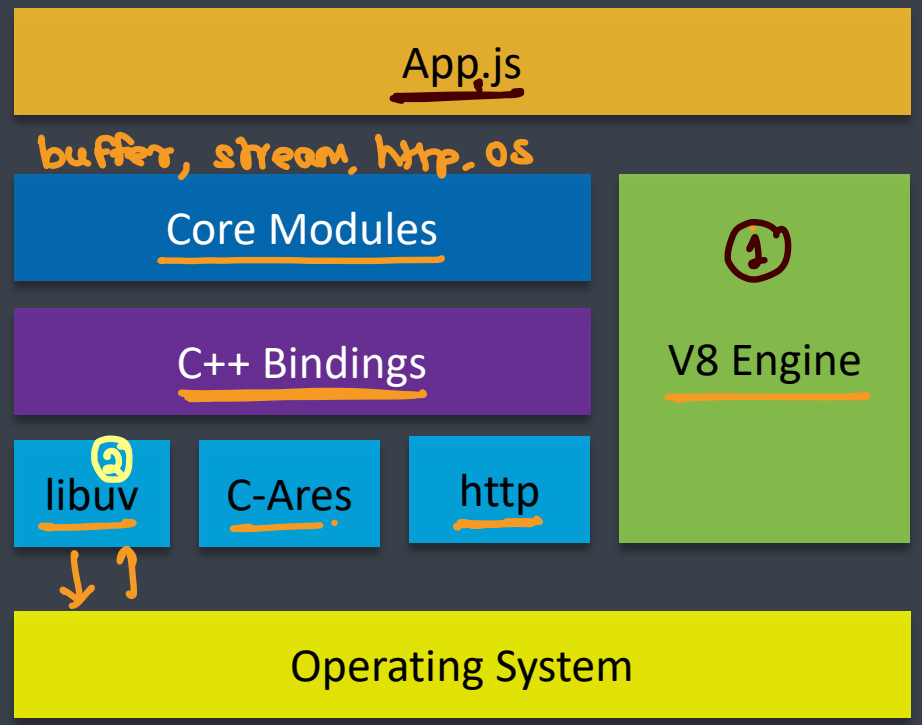
NodeJS Components

- Node.js relies on various dependencies under the hood for providing various features
  - V8
  - libuv
  - llhttp
  - c-ares
  - OpenSSL

platform & collection of APIs used for various functionalities

→ eg. fs, threading, asynchronous..

fopen(..)
↳ os:- open(..)
platform ↳ system call



javascript → Node → V8 → libuv
timers,
callbacks,
multi-threading

V8 → max performance written in C++

- V8 is the name of the JavaScript engine i.e. it parses and executes JavaScript code which powers Google Chrome
- The cool thing is that the JavaScript engine is independent of the browser in which it's hosted. This key feature enabled the rise of Node.js.
- V8 is written in C++, and it's continuously improved
- It is portable and runs on Mac, Windows, Linux and several other systems
- JavaScript is generally considered an interpreted language, but modern JavaScript engines no longer just interpret JavaScript, they compile it
- This has been happening since 2009, when the SpiderMonkey JavaScript compiler was added to Firefox 3.5, and everyone followed this idea
- JavaScript is internally compiled by V8 with just-in-time (JIT) compilation to speed up the execution

JavaScript code → Node → V8 compiles → WebAssembly

Libuv

- Libuv is an open-source library written in C, that uses a thread-pool (multi-thread) to manage parallel operations
- libuv enforces an asynchronous, event-driven style of programming
- Its core job is to provide an event loop and callback based notifications of I/O and other activities
- libuv offers core utilities like timers, non-blocking networking support, asynchronous file system access, child processes and more
- Features
  - Full-featured event loop backed by epoll (Linux), kqueue (OSX), IOCP (Windows), event ports (SunOS)
  - Asynchronous TCP (net module) and UDP (dgram module)
  - Asynchronous DNS resolution (used partly for the dns module)
  - Asynchronous file, file system operations & events (fs module)
  - ANSI escape code controlled TTY
  - Thread pool and Signal handling
  - Child processes
  - High-resolution clock
  - Threading and synchronization primitives.
  - Inter-Process Communication using sockets and Unix domain sockets (Windows)

NodeJS Features

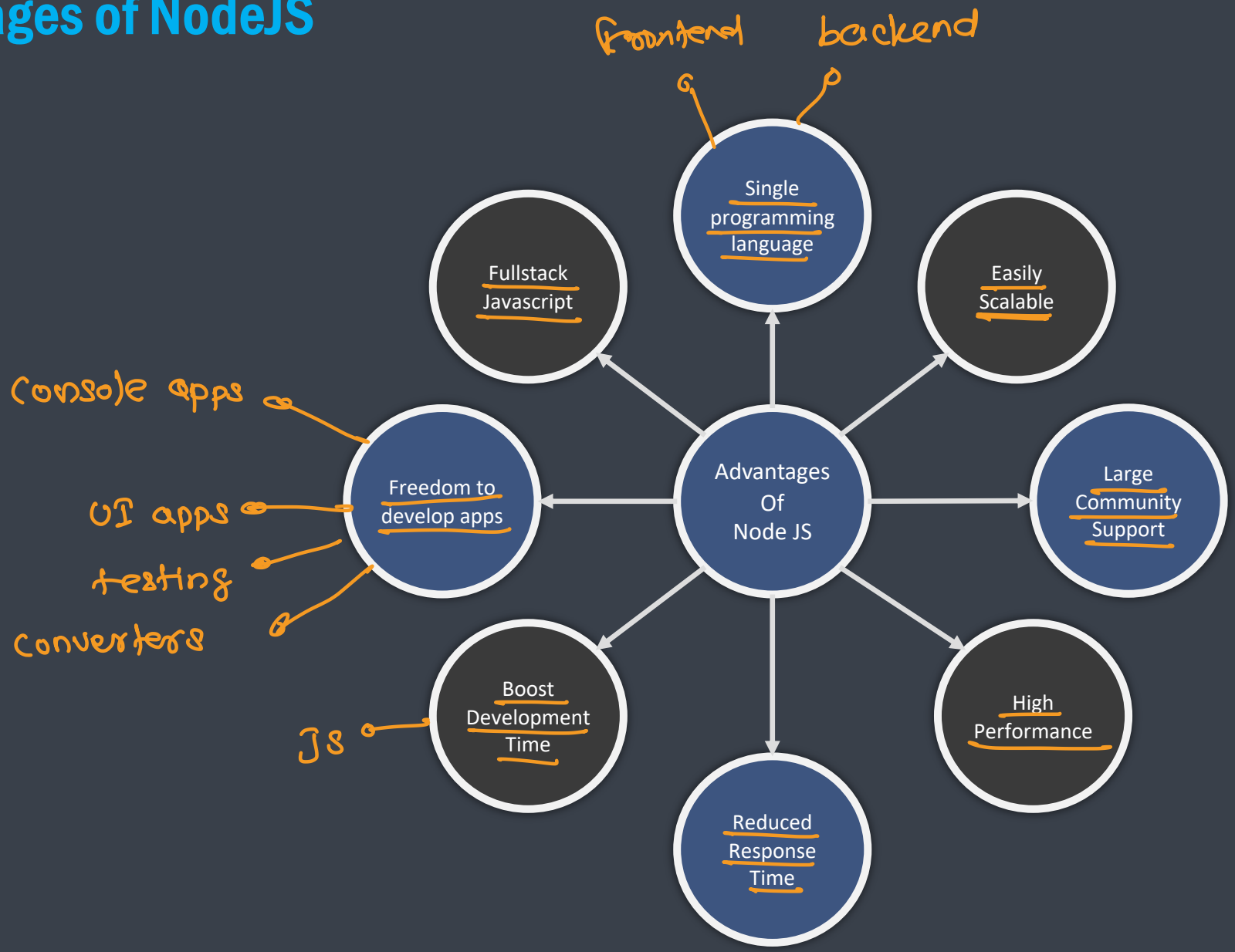
- It uses event-driven, non-blocking I/O model
- It has a single threaded but highly scalable architecture
- It is a lightweight and efficient
- It has its own package ecosystem which has thousands of packages for different purpose
- It never buffers the data. Rather it simply outputs the data in chunks

NodeJS Use Cases

- Where to use NodeJs
  - I/O bound applications → Database / File IO
  - Data streaming applications
  - IoT applications
  - JSON API based applications
  - Single Page Applications (SPA)

- Where not to use NodeJs
  - CPU intensive applications → video conversion
  - Heavy server sided processing

Advantages of NodeJS



How much JS is required to use NodeJS

- Lexical Structure → syntax ✓
- Expressions → statements ✓
- Data Types → number, string, boolean, object, undefined, null
- Variables → variable vs const → ~~var~~, let, const
- Functions → named, arrow (anonymous)
- Arrow Functions
- Loops → for, while
- Scopes → global, local (function)
- Arrays →
- ECMAScript 2015 (ES6) and beyond → class, interface etc

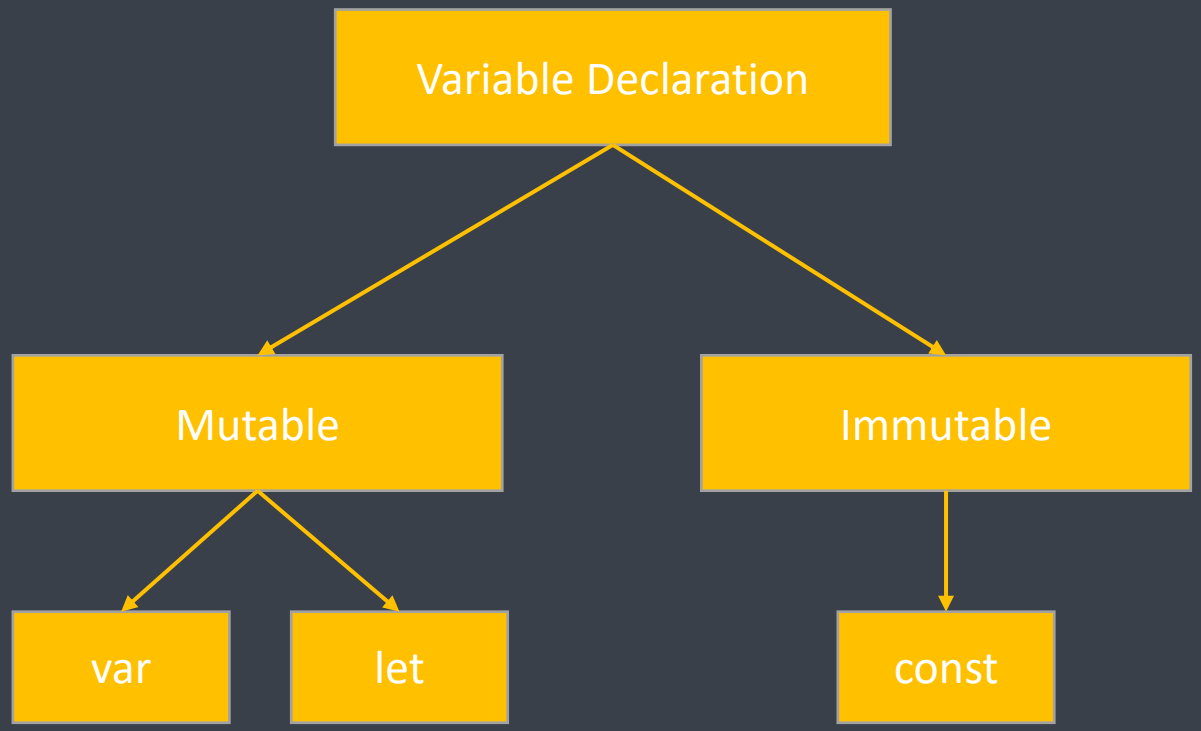
JavaScript

JavaScript

- Weekly Typed Language
  - No explicit type assignment
  - Data types can be switched dynamically
- Data types are inferred
  - Data types are assigned by JS by inspecting value in the variable
- Object Oriented language
  - Data can be organized in logical objects
  - Primitive and reference types
- Versatile language
  - Runs in browser as well as directly on desktop
  - Can perform variety of tasks


```
let num = 200
```

Variables and Constants




```
// function scoped
// redeclaration allowed
var num = 100

// block scoped
// redeclaratoin is not allowed
let firstName = 'steve'

// constant
// once declared, can not change
const address = pune
```

Functions

```
// named function
function add(p1, p2) {
  const addition = p1 + p2
  console.log(`addition = ${addition}`)
}

add(10, 20)
```

```
// arrow function
const add = (p1, p2) => {
  const addition = p1 + p2
  console.log(`addition = ${addition}`)
}

add(10, 20)
```

Objects

- Reference type
- Collection of Key-Value pairs
- Key must be string
- Value can be of any type
  - String
  - Number
  - Object
  - Boolean
  - Array
  - Function
- Object can be created using
  - JSON
  - Object
  - Constructor function

```
// person is a reference
const person = {
  name: "person 1",
  address: "Pune",
  age: 30,
  married: true,
  canVote: function () {
    if (this.age >= 18) {
      console.log("can vote")
    } else {
      console.log("can not vote")
    }
  }
}

// calling function
person.canVote()
```

Array

- Collection of values
- [] is used to create an array
- Array can hold similar or dissimilar objects
- Provides methods like
  - push
  - pop
  - insert
  - splice
  - includes
  - indexOf
  - map
  - filter
  - reduce
  - join

```
// collection of numbers
const numbers = [10, 20, 30, 40]

// collection of strings
const names = ["person1", "person2"]

// collection of mixed values
const mixed = [10, "test", true]

// collection of objects
const persons = [
  { name: "person1", age: 20 },
  { name: "person2", age: 22 },
  { name: "person3", age: 23 }
]
```

Loops

```
const numbers = [10, 20, 30, 40]

// for..of loop
for (const value of numbers) {
  console.log(`number = ${value}`)
}
```

```
const numbers = [10, 20, 30, 40]

// for..each loop
numbers.forEach(number => {
  console.log(`number = ${number}`)
});
```

```
const numbers = [10, 20, 30, 40]

// traditional for loop
for (let index = 0; index < numbers.length; index++) {
  const number = numbers[index];
  console.log(`number = ${number}`)
}
```

Spread and Rest Operators

```
// original array
const numbers = [10, 20, 30, 40, 50]

// copy of old array
const newArray = [...numbers]

// original object
const person = {
  name: "person 1",
  age: 20,
  address: "pune"
}

// copy of old object
const newPerson = {
  ...person,
  phone: "+911234123412"
}
```

```
// rest operator
function add(...args) {
  let addition = 0
  for (const value of args) {
    addition += value
  }

  console.log(`addition = ${addition}`)
}

add(10, 20, 30, 40)
```

De-structuring

```
const person = {
  name: "person 1",
  age: 20,
  address: "pune"
}

// destructuring object into variables
const { name, age, address } = person
console.log(`name: ${name}`)
console.log(`age: ${age}`)
console.log(`address: ${address}`)
```

```
const numbers = [10, 20, 30, 40]

// destructuring array
const [n1, n2, n3, n4] = numbers
console.log(`n1 = ${n1}`)
console.log(`n2 = ${n2}`)
console.log(`n3 = ${n3}`)
console.log(`n4 = ${n4}`)
```

Module

- Collection of functions
- JavaScript libraries which can be shared with multiple applications
- Types
  - Built-in modules
    - HTTP
    - File System
    - Buffer
    - OS
  - User defined modules

```
// math.js

function add(p1, p2) {
  console.log(`addition = ${p1 + p2}`)
}

// export all the entities need to
// be exported from this module
module.exports = {
  add: add
}
```

```
// importing math module
const math = require('./math.js')

// calling add function exported
// from math module
const result = math.add(10, 20)
console.log(`result = ${result}`)
```