# TABLE OF CONTENT

**W03: JAVASCRIPT........................... 2**

1. THEORY................................................ 2

35. VS..........................................................3

**W04: NODE.JS EXPRESS............. 4**

THEORY....................................................4

18. HTTP.................................................... 4

19. EXPRESS...........................................5

31. Core Express...................................5

32. EJS........................................................ 5

35. VS..........................................................5

**W05: MONGODB..............................7**

1. THEORY.........................................7

42. VS.................................................9

48. INTERVIEW QUESTION............10

56. GOOD TO KNOW...................... 10

**FD 01: HTML & CSS..............................11**

1. HTML............................................ 11

9. VS.................................................. 11

13. GOOD TO KNOW.......................11

17. CSS............................................ 11

**FD 01: JAVASCRIPT............................. 12**

1. DOM............................................ 12

9. THEORY......................................12

56. VS.............................................14

57. GOOD TO KNOW..................... 14

**FD 01: NODE.JS EXPRESS..................15**

THEORY..........................................15

64. HTTP......................................... 15

65. EXPRESS..................................16

77. Core Express.............................16

78. EJS............................................ 17

81. VS.............................................17

**W07: GIT................................................18**

1. THEORY......................................18

23. GOOD TO KNOW...................... 18

**REACT.................................................. 19**

1. Basics.......................................... 19

8. Theory......................................... 19

22. Good to Know............................20

**OTHERS................................................21**

1. SASS..........................................21

4. SOLID Design Principles.............21

10. REST API.................................. 21

23. CI CD (git)..................................21

# W03: JAVASCRIPT

1. **THEORY**
2. **Function**
   a. Function Statement
   b. Function Expression
   c. Function Declaration
   d. Anonymous function
   e. Named Function Expression
   f. Functional Programing
   g. **Higher order function**
   h. **First class function**
   i. **Pure Function**
   j. **Function composition**

3. Advantages and disadvantages of JS
4. Scope, Lexical scope
5. Prototype
6. 
7. **Closure**
   a. Disadvantage
   b. Uses
8. Garbage collection
9. **Hoisting**
   a. TDZ
   b. let, const vs var
   c. Function vs arrow function
10. Call Apply Bind
11. This Keyword
12. Temporal Dead Zone
13. **String Methods**
    a. Length
    b. toUpperCase, LowerCase
    c. Trim
    d. Pad
    e. charAt
    f. Split
    g. Concat
    h. substring
14. **Array Methods**
    a. Map
    b. Filter
    c. Reduce
    d. Find
    e. Sort
    f. Foreach
    g. Push
    h. Pop
    i. Shift
    j. Unshift
    k. Slice
    l. Splice
15. **Object Methods**
    a. freeze
16. Callback and callback hell
17. **Promise**
    a. Promise.all
    b. Promise.allSettled
    c. Promise.race
    d. Thenable
    e. Finally
    f. Catch
18. Async await
19. Spread and Rest Operator
20. DOM, BOM
21. Call stack
22. Event loop
23. **ES6 and its features**
    a. Let, Var, Const
    b. Ternary operator
    c. Arrow function
    d. Template literals
    e. Default Parameters
    f. Classes
    g. Modules
    h. Iterators
    i. Object & Array Destructuring
    j. SetInterval

24. **Primitive and non-primitive**
    a. Pass by value and pass by reference
25. Message queue
26. Life
27. Generator
28. **Prototype**
    a. Prototype chain

    b. Prototypal Inheritance

29. JavaScript is dynamically types
30. Currying
31. **Type Casting**
    a. Implicite (Coercion)
    b. Explicit (Conversion)
32. Microtask queue
33. Shallow copy
34. Deep copy
35. Immutable

## 36. VS

    a. == and ===
    b. Let, const, var
    c. Synchronous vs asynchronous
    d. While vs do while
    e. Foreach Vs Map
    f. Parameters, Arguments
    g. for in, for of
    h. Undefined, Null
    i. Keywords & Identifiers
    j. Type casting vs Type coercion

# W04: NODE.JS EXPRESS

## THEORY

1. What is Node.js
2. why v8 Engine
3. Advantages & Disadvantages of Node.js
4. How node works
5. Node Module System
6. REPL, Cli
7. NPX
8. Globals
    a. __dirname
    b. __filename
    c. **Module**
    d. Process
9. **Modules**
    a. **Core Modules.**
    b. local Modules.
    c. Third-party Modules.
    d. module.exports:{}
    e. require
    f. ESM
        i. import and export
10. **NPM**
    a. local and global
    b. npm init
    c. npm install or i
11. Nodemon
    a. scripts
        i. start
        ii. dev
    b. npm run dev
12. package.json
13. package-lock.json
14. Event loop
15. Event Queue
16. Events
    a. **Events emitter**
    b. Http module
17. Streams
    a. type of streams
        i. writable, readable, duplex, transform
    b. createReadStream()
    c. pipe()
18. **HTTP**
    a. https
    b. How does it work?
    c. request response cycle
    d. Stateless protocol
        i. Local storage, Sessions and Cookies
    e. Request
        i. General (start line)
            1. method/target/version
        ii. header
        iii. body
    f. Response
        i. General (start line)
            1. version/statuscode/statustext
        ii. header
            1. content type
        iii. body
            1. requested resource
    g. **HTTP Methods**
        i. GET
        ii. POST
        iii. PUT
        iv. DELETE
    h. Idempotent
    i. Headers
    j. Status code
        i. 1xx: Informational
        ii. 2xx: Success
            1. 200 - Success
            2. 201 - Success and created
        iii. 3xx: Redirect
            1. 301: moved to new URL

            2. 304: not changed
        iv. 4xx: Client Error
            1. 401: Unauthorised
            2. 402: 402 Payment Required
            3. 403: Forbidden
            4. 404: page not found
        v. 5xx: Server Error
    k. MIME type
    l. HTTP v2
    m. TCP and IP

19. **EXPRESS**

20. npm install express –save
21. app = express()
    a. get()
        i. status()
        ii. send()
        iii. sendFile()
    b. post()
        i. express.urlencode()
        ii. Form vs JS
    c. put()
    d. patch()
    e. delete()
    f. all()
    g. use()
    h. listen()
22. Static files
    a. public
    b. express.static()
23. **API**
    a. json()
24. Params, Query String
25. Route Parameter
26. Query string/url Parameter
27. **MIddleware**
    **a.** what is middleware
    b. used for what?
    **c. Types of Middleware**
        i. Application-level middleware
        ii. Third party middleware
            1. morgan
            2. multer
        iii. Router-level middleware
        iv. Built-in middleware
        v. Error-handling middleware
            1. err.statusCode
            2. err.message
    d. req, res, next
    e. next()
    f. app.use in middleware
    g. passing two middleware
28. **Routing**
    a. router
    b. express.Router()
29. Cluster
30. Multithreading in node.js
    a. require('worker_theads')
    b. new Worker

31. **Core Express**

    **a. Session**
        i. i express-session
        ii. secret
        iii. resave
        iv. saveUninitialized
        v. destroy()
    **b. Cookies**
        i. i cookie-parser
    c. Core middleware
    d. Core routing
    e. Build own API
    f. Core views
    g. database integration

32. **EJS**

    a. i ejs
    b. server side rendering
    c. view engine
    d. render()
    e. <% %>, <%- %>, <%= %>
    f. partials
33. **Rest API**
    a. RESTful
34. fragment identifier

35. **VS**

36. API vs HTTP
37. API vs SSR

38. HTTP vs HTTPS
39. URIs vs URLs vs URNs
40. Session vs Cookies
41. GET vs POST
42. PUT vs PATCH
43. SSL vs TLS
44. **Build-in Modules (only imp)**
    a. os
    b. path
        i. join()
        ii. basename()
        iii. resolve()
    c. fs
        i. fs sync
        ii. - readFileSync()
        iii. - writeFileSync()
        iv. **fs async**
        v. - readFile()
        vi. - writeFile()
    d. http
        i. createServer()
            1. url
            2. listen()
            3. write()
            4. writeHead()
            5. end()
    e. util
        i. util.promisify
        ii. util.callbackify
    f. events
        i. eventEmmitter
            1. .emit()
            2. .on()
        ii. on()
    g. net
    h. crypto
        i. password hashing
        ii. .crateHmac(sha256, secret).update('<valuetoIncript>').digest('hex')

# W05: MONGODB

1. **THEORY**
2. SQL(relational) vs
3. NoSQL ()
4. What is MongoDB?
5. Run on JS Engine
6. How does mongoDB work?
7. Non-relational Document based
8. Advantage and Disadvantages
9. BSON
10. MongoDB Structure
11.
12. MongoDB architectureJSON vs BSON
13. MongoDB shell
14. CRUD Operations
15. Cursor, Iterate a Cursor
16. Time to Leave
17. Maximum Document Size : 16Mb
    a. GridFS
18. **Data types in MongoDB (BSON)**
    a. ObjectId
        i. timestamp
        ii. random value
        iii. incrementing counter
    b. String
    c. Int, longInt, Double
    d. Array, Object
    e. Boolean
    f. Date
    g. Decimal128
    h. Regex
    i. Javascript
        i. with scope
        ii. without scope
    j. MinKey, MaxKey
    k. Binary data
19. Cursor
    a. cursor methods
    b. - toArray
    c. - forEach
20. **Collection**
    a. db
    b. db.createCollection(collectionName)
    c. show collections
    d. renaming Collection
21. **Documents**
    **a.** adding new Documents
    b. Nested Documents
        i. advantage
22. **Inserting Document**
23. Insert One and Many
24. what are the additional methods used for inserting
25. **Finding / Querying**
    a. find()
        i. iterate (it)
        ii. pretty()
    b. findOne({ *filter* })
    c. finding In nested Array
        i. “*field.field*”
        ii. match
        iii. exact match
        iv. multiple match
    d. Array
        i. finding in specific order
        ii. without regard to order
        iii. query by array index
        iv. query by array length
    **e. Projection**
        i. explicitly include fields
    f. Null, $type: 10, $exists
26. **Filtering**
    a. find( *filter* )
    b. find( *{filter}, {fieldsToGet}* )
27. **Method Chaining**
    a. count()
    b. limit()
    c. sort( 1 or -1 )
    d. skip()
28. **Operators** (denoted by $)
    a. {$gt: number} $gte
    b. $lt, $lte
    c. $eq, $ne
    d. $or $and $not
    e. $in: [1,2,3], $nin: [1,2]

f. $all
g. $set, $unset
**h. $elemMatch**
i. $slice
j. $size
k. $inc: 1, $inc: -1
l. $pull, $push
m. $each [ 1, 2 ]
n. $eq, $ne
o. $currentDate
p. $exists
**q. $expr**
**r. $cond**
s. $rename
t. $min, $max
u. $mul
**v. Array Operator**
    i. $push
    ii. $each
    iii. $pull
    iv. $pullAll
    v. $pop
    vi. $
    vii. $elemMatch

**29. Deleting**
    a. deleteOne({ *field:value* })
    b. deleteMany()
    c. remove()

**30. Updating**
    a. updateOne( {*whichObject*} , {$set: {*field: value, field: value*} } )
    **b. Operators**
        i. $set
        ii. $unset
        iii. $rename
    c. updateMany()
    d. replaceOne()
    e. incrementing & decrementing
    f. adding and remove from array
    g. upsert
    h. update() vs updateOne()
    i. updateOne vs replaceOne

**31. bulkWrite()**
    a. ordered: false
    b. ordered vs unordered
    c. advantages and disadvantages

32. **Commands**
    a. mongosh
    b. db
    c. show dbs
    d. db.stats

**33. Aggregation**
    a. How does it work
    b. advantages
    c. types of aggregation
    d. distinct
    **e. Aggregate stages**
        i. $match
        ii. $group
            1. grouping by
            2. -nested field
            3. -multiple field
        iii. $sort
        iv. $count
        v. - other ways to count
        vi. - client and server side counting
        vii. $limit, $skip
        viii. $out
        ix. $project
        x. $lookup
        xi. $unwind
        xii. allowDiskUse: true
    f. “$name” vs ”name”
    **g. Accumulator Operators**
        i. $sum, $avg, $max, $min
    **h. Unary Operators**
        i. $type, $lt $gt $or $and $multiply
    **i. Aggregation Pipeline**
        **i.** How does aggregation pipeline work?
        ii. memory limit : 100mb
            1. spill to disk
    j. Batch sizing
    k. Iterator Size
    l. Query routing

m. **Map Reduce**
    i. for what is it used?
    ii. find sum, avg

34. **Indexes**
    a. pros and cons of Indexes
    b. createIndex({ *filed: value* })
    c. options when creating Index
        i. background: true
        ii. unique: true
        iii. name: "<*indexName*>"
    d. getIndex()
    e. dropIndex(), dropIndexes
    f. reIndex()
    g. rename Index
    h. hiding index
    **i. Types of Indexes**
        i. Single Field Index
        ii. Compound Index
        iii. Multikey Index
        iv. Text Index
        v. Geospatial, Hashed, Clustered Index

35. **Schema**
    a. pros and cons of using schema
    b. optional schema
    c. validation action

36. **Relationships**
    a. embedding
    b. referencing
    c. one-to-one
    d. one-to-many
    e. one-to-squillions
    f. many-to-many

37. **Replication**
    a. replica set
    b. advantage and disadvantages of replication
    **c. Replication Architecture**
        i. primary and secondary nodes
        ii. arbiter
        iii. process of election
        iv. heartbeat
    d. Process of Election
    e. Replication lag
    f. operation log (oplog)
    **g. Types of replication**
        i. Asynchronous Replication
        ii. Synchronous Replication
        iii. Majority Commit
        iv. etc...

38. **Sharding**
    a. advantages and disadvantages
    **b. Sharding Architecture**
        i. What is Mongos/Router
        ii. Config Server
    **c. Types of sharding**
        i. Hashed sharding
        ii. Ranged sharding
        iii. Zone Sharding
    **d. Shard key**
        i. shard hotspots
        ii. normal shard key
        iii. hashed shard key
    e. Vertical and horizontal scaling
    f. Zones
    g. mongos
    h. auto balancer
    i. scatter-gather

39. **Cluster**
    a. types of cluster
    b. config servers

40. **Data Modeling**
    a. embedded data model
    b. reference data model
    c. linking vs embedding

41. **Transactions**
    a. ACID Transaction
    b. A- Atomicity
    c. C- Consistency
    d. I - Isolation
    e. D - Durability

42. **VS**
    a. $or vs $in
    b. $all vs $in
    c. drop() vs remove()

    d. findAndModify() vs findOneAndUpdate()
    e. Primary key vs secondary key
    f. join vs lookup
    g. dot notation vs nested form
    h. $currentDate vs $$NOW
    i. delete() vs remove()
    j. bulkWrite vs InsertMany
    k. replace vs update
    l. shard vs node vs cluster
    m. Aggregation Pipeline vs Map Reduce
    n. vertical scalability vs horizontal scalability
    o. load balancer vs sharding
    p. odm vs driver
    q. stage operator vs accumulator operator
    r. normal shard key vs hashed shard key
    s. aggregate([$count:"tota"]) vs find({}).count()
    t. replication vs replica set
    u. transaction vs query
    v. scaling up vs scaling down vs scaling out?
    w. config servers vs mongos
    x. load balancer vs auto balancer
    y. countdocument vs count

43. What is a MongoDB driver?
44. Capped collection and it's advantages
45. Profiler
46. Explain
47. Soft deleting
48. **INTERVIEW QUESTION**
49. What to do when your quireing becomes slow?
50. What to do when your files are getting very big?
51. How to condense large volumes of data?
52. How to search for text in MongoDB?
53. How does MongoDB schema change?
54. How can we Backup and Restore in MongoDB?
55. What are the pros and cons of Normalising Data in MongoDB
56. **GOOD TO KNOW**
57. Atomicity
58. Type Bracketing
59. Dot Notation
60. Cursor behaviour
61. Aggregation Pipeline
62. Retryable Writes and Reads
63. MongoDB CRUD Concepts
64. B-Tree
65. ACID compliance
66. Mongoose
67. Network Components
    a. load balancer
    b. firewall
68. **CAP Theorem**
    a. consistency
    b. availability
    c. partition tolerance
69. Firewall
70. **Mongo Utilities**
    a. mongoexport
    b. mongoimport
    c. mongodump
    d. mongorestore
    e. mongostat
    f. mongotop
    g. mongooplog

# FD 01: HTML & CSS

1. **HTML**
2. **Basics**
3. **Block element and inline element**
4. Element
    a. Void elements
    b. Container Element
5. Attributes
    a. boolean attributes
    b. lang attribute
6. Nesting
7. <!DOCTYPE html>
8. **head**
    **a. <meta>**
    b. <meta charset="utf-8">
    c. Adding an author and description
9. **VS**
10. h1 vs title in head
11. <em> vs <i>
12. <b> vs <strong>
13. **GOOD TO KNOW**
14. Whitespace
15. entity references
    a. < &lt;
    b. > &gt;
    c. " &quot;
16. Open Graph Data
17. **CSS**
18. Anatomy of CSS ruleset
19. Selecters
    a. Element
    b. Id, Class
    c. Attribute
    d. Pseudo
20. Box model

# FD 01: JAVASCRIPT

1. **DOM**
2. querySelector
3. textContent
4. addEventListener
5. Order of Parsing
6. **event Propagation**
   a. event Bubbling
   b. event Capturing/ Trickling
   c. how to add both on program
7. event.stopPropagation();
8. event Delegation
   a. e.target
      i. id
      ii. tagName
      iii. pros and cons
9. **THEORY**
10. Data types
11. Operators
12. enum
    a. how to get enum in javascript
13. **Function**
    a. Function Statement
    b. Function Expression
    c. Function Declaration
    d. Anonymous function
    e. Named Function Expression
    f. Functional Programing
    g. **Higher order function**
    h. First class function
    i. **Decorator function**
       i. use
       ii. - count no of function call
       iii. - valid data of params
    j. **Pure function**
       i. pros and cons
       ii. rules
       iii. pure vs impure
14. Advantages and disadvantages of JS
15. **Set Map Flat**
    a. set
       i. add()
       ii. has()
       iii. delete()
    b. map
       i. get ()
       ii. set ()
       iii. <mapName>.size
       iv. iterating
    c. object vs map
    d. weekSet()
       i. features
    e. weekMap()
       i. features
       ii. key is private
    f. falt()
    g. flatMap()
    h. reduceRight()
    i. copyWithin()
16. **Operators**
    a. Nullish operator
    b. Optional chaining
    c. Ternary operator
    d. Type Operators
    e. **Unary operators**
       i. delete
       ii. typeof
       iii. !, ++, -, +
    f. **Bitwise Operators**
       i. bitwise OR
       ii. bitwise AND
       iii. uses
17. **Scope**
    a. Global scope
    b. Module scope
    c. Function scope
    d. Lexical scope
    e. Block scope
18. **Prototype**
19. Types of error
    a. syntax, logic
20. **Closure**
    a. Disadvantage
    b. Uses
    c. lexical scope vs closure

    d. IIFE
21. Garbage collection
22. **Hoisting**
    a. TDZ
    b. let, const vs var
    c. Function vs arrow function
23. **Call Apply Bind**
    a. function borrowing
    b. call vs apply vs bind
    c. polyfills
24. This Keyword
25. **String Methods**
    a. Length
    b. toUpperCase, LowerCase
    c. Trim
    d. Pad
    e. charAt
    f. Split
    g. Concat
    h. substring
26. **Array Methods**
    a. Map
    b. Filter
    c. Reduce
    d. Find
    e. Sort
    f. Foreach
    g. Push
    h. Pop
    i. Shift
    j. Unshift
    k. Slice
    l. Splice
27. **Object Methods**
    a. freeze
28. **Loop**
    a. for
    b. do while vs while
    c. labelled statements
    d. - break
    e. - continue
    f. for...in
    g. for...of
29. **Callback**
    a. callback hell
    b. inversion of control
30. **Promises**
    a. Promise states
    b. Promise chaining
    c. Promise.all
    d. Promise.allSettled
    e. Promise.any
    f. Promise.race
    g. Promise.resolve
    h. Thenable
    i. Finally
    j. Catch
    k. immutable
    l. promisify
    m. pros and cons
31. **Async await**
    a. async always return a promise
    b. error handling in async await
32. **Debouncing & Throttling**
    a. both are used for optimising performance of a web app
    b. by limiting the rate of API calls
33. Spread and Rest Operator
34. DOM, BOM
35. Call stack
36. Event loop
37. **ES6 and its features**
    a. Let, Var, Const
    b. Ternary operator
    c. Arrow function
    d. Template literals
    e. Default Parameters
    f. Classes
    g. Modules
    h. Iterators
    i. Object & Array Destructuring
38. **Primitive and non-primitive**
    a. Pass by value and pass by reference
39. Message queue
40. Life
41. Generator
42. **Prototype**
    a. Prototype chain
    b. Prototypal Inheritance

   c. uses?
   d. Circular reference
   e. Object.key
43. **Recursion**
   a. recursive call to function
   b. condition to exit
   c. pros and cons
   d. display the fibonacci sequence
   e. use
44. JavaScript is dynamically types
45. **Currying**
   a. function inside function
46. **Type Casting**
   a. Implicite (Coercion)
   b. Explicit (Conversion)
47. Microtask queue
48. **Shallow copy vs Deep copy**
   a. primitive vs structural
   b. how make these copies
   c. pros and cons
   d. Mutable vs Immutable
   e. Object.freeze()
49. TCP/IP
50. DNS
51. **IIFE**
   a. pros and cons
52. **Composition vs Inheritance**
53. Function recursion
54. [Symbol.iterator]
55. Truthy and  falsy value

## 56. VS

   a. == and ===
   b. Let, const, var
   c. Synchronous vs asynchronous
   d. While vs do while
   e. Foreach Vs Map
   f. Parameters, Arguments
   g. for in, for of
   h. Undefined, Null
   i. Keywords & Identifiers
   j. Type casting vs Type coercion
   k. textContent vs innerText
   l. identifiers vs variables
   m. defer vs async

## 57. GOOD TO KNOW

58. interpreted and compiled doe
59. Server-side vs client-side code
60.

# FD 01: NODE.JS EXPRESS

## THEORY

45. What is Node.js
46. why v8 Engine
47. Advantages & Disadvantages of Node.js
48. How node works
49. Node Module System
50. REPL, Cli
51. NPX
52. Globals
    a. __dirname
    b. __filename
    c. **Module**
    d. Process
53. **Modules**
    a. **Core Modules.**
    b. local Modules.
    c. Third-party Modules.
    d. module.exports:{}
    e. require
    f. ESM
        i. import and export
54. **NPM**
    a. local and global
    b. npm init
    c. npm install or i
55. Nodemon
    a. scripts
        i. start
        ii. dev
    b. npm run dev
56. package.json
57. package-lock.json
58. Event loop
59. Event Queue
60. Events
    a. **Events emitter**
    b. Http module
61. **Streams**
    a. type of streams
        i. writable, readable, duplex, transform
    b. createReadStream()
    c. pipe()
    d. Buffers
62. **Cron-job**
    a. * * * * *
    b. 1st* = second
    c. 2nd* = minute
    d. 3rd* = hour
    e. 4th* = day of month
    f. 5th* = month
    g. 6th* = day of week
    h. or, range selector
    i. time zone
    j. validation
63. **CORS**
    a. preflight request
        i. header
        ii. accept-control-allow-origin: *
        iii. accept-control-allow-methods: *
        iv.

### 64. HTTP

    a. https
    b. How does it work?
    c. request response cycle
    d. Stateless protocol
        i. Local storage, Sessions and Cookies
    e. Request
        i. General (start line)
            1. method/target/version
        ii. header
        iii. body
    f. Response
        i. General (start line)
            1. version/statuscode/statustext
        ii. header

            1. content type
        iii. body
            1. requested resource
    g. **HTTP Methods**
        i. GET
        ii. POST
        iii. PUT
        iv. DELETE
        v. HEAD
        vi. CONNECT
        vii. OPTIONS
        viii. TRACE
    h. Idempotent
    i. Headers
    j. writeHead vs setHead
    k. Status code
        i. 1xx: Informational
        ii. 2xx: Success
            1. 200 - Success
            2. 201 - Success and created
        iii. 3xx: Redirect
            1. 301: moved to new URL
            2. 304: not changed
        iv. 4xx: Client Error
            1. 401: Unauthorised
            2. 402: Payment Required
            3. 403: Forbidden
            4. 404: Page not found
        v. 5xx: Server Error
    l. MIME type
    m. HTTP v2
    n. TCP and IP

## 65.EXPRESS

66. npm install express –save
67. app = express()
    a. get()
        i. status()
        ii. send()
        iii. sendFile()
    b. post()
        i. express.urlencode()
        ii. Form vs JS
    c. put()
    d. patch()
    e. delete()
    f. all()
    g. use()
    h. listen()
68. Static files
    a. public
    b. express.static()
69. **API**
    a. json()
70. Params, Query String
71. Route Parameter
72. Query string/url Parameter
73. **MIddleware**
    a. what is middleware
    b. used for what?
    c. req, res, next
    d. next()
    e. app.use in middleware
    f. passing two middleware
    g. **Types of Middleware**
        i. Application-level middleware
        ii. Third party middleware
            1. morgan
            2. multer
        iii. Router-level middleware
        iv. Built-in middleware
        v. Error-handling middleware
            1. err.statusCode
            2. err.message
74. **Routing**
    a. router
    b. express.Router()
75. Cluster
76. Multithreading in node.js
    a. require('worker_theads')
    b. new Worker

## 77.Core Express

    a. **Session**
        i. i express-session

        ii. secret
        iii. resave
        iv. saveUninitialized
        v. destroy()
    b. **Cookies**
        i. i cookie-parser
    c. Core middleware
    d. Core routing
    e. Build own API
    f. Core views
    g. database integration

## 78. EJS

    a. i ejs
    b. server side rendering
    c. view engine
    d. render()
    e. <% %>, <%- %>, <%= %>
    f. partials

79. **Rest API**
    a. RESTful

80. fragment identifier

## 81. VS

82. API vs HTTP
83. API vs SSR
84. HTTP vs HTTPS
85. URIs vs URLs vs URNs
86. Session vs Cookies
87. GET vs POST
88. PUT vs PATCH
89. SSL vs TLS
90. **Build-in Modules (only imp)**
    a. os
    b. path
        i. join()
        ii. basename()
        iii. resolve()
    c. fs
        i. fs sync
        ii. - readFileSync()
        iii. - writeFileSync()
        iv. **fs async**
        v. - readFile()
        vi. - writeFile()
    d. http
        i. createServer()
            1. url
            2. listen()
            3. write()
            4. writeHead()
            5. end()
    e. util
        i. util.promisify
        ii. util.callbackify
    f. events
        i. eventEmmitter
            1. .emit()
            2. .on()
        ii. on()
    g. net
    h. crypto
        i. password hashing
        ii. .crateHmac(sha256, secret).update(‘<valuet oIncript>’).digest(‘hex’)

# W07: GIT

1. **THEORY**
2. Config
3. git init
4. git clone
5. git status
6. **Creating Version**
   a. git add *file*
      i. git add - - all
      ii. git add .
   b. git commit
      i. -m “<*message*>”
   c. commit id
      i. check sum
      ii. **content**
         1. author details
         2. preview details
         3. date
         4. etc..
      iii. sha-1 hash
   d. label
   e. branch
7. touch
8. git log
9. git diff
10. git stash
11. **git checkout**
    a. commit id
    b. branch name
12. git log - - all
13. **Branching**
    a. git branch <*branchName*>
    b. git branch
14. **Merging**
15. git merge <*branchName*>
16. **types of merging**
    a. fast-forward merge
    b. **recursive merge**
       i. conflict
17. **Git server**
    a. git remote add <*name*> <*url*>
       i. git remote
       ii. git remote -v
    b. git push <*remoteName*> <*branchName*>
    c. **Cloning**
    d. git clone <*url*>
    e. git pull
    f. pull vs pull request?
18. **Forking**
19. vim .gitignore
20. gist
21. **ci cd**
22. git projects
23. **GOOD TO KNOW**
24. rebase
25. tree
26. brew install tree

# REACT

1. **Basics**
2. npx create-react-app <appName >
3. components
   a. default is App
4. rafce
5. calling function on button click
   a. without parameter
   b. with parameter
6. Fragments
7. Children Prop
8. **Theory**
9. What is React
10. Virtual DOM
    a. Reconciliation
    b. Diffing Algorithm
11. props vs state
12. Server Side vs Client Side Rendering in React
13. React Fibre
14. Synthetic Events
15. Life Cycle
16. VIew Oriented
17. **Hooks**
    a. useState
       i. changeValue
       ii. changeValueWithFunction
    b. useRef
       i. html
       ii. useState vs useRef
    **c. useEffect**
       i. dependency
       ii. return in useEffect
       iii. useLayoutEffect
    d. useMemo
       i. sample
       ii. recache
       iii. pros and cons
       iv. referential equality
    e. useCallback
       i. sample
       ii. useMemo vs useCallback
       iii. uses
    f. useContext
       i. sample
    **g. useReducer**
    **h. Create custom hooks**
       i. useDebugValue
    i. useTransition
    j. useDeferredValue
    k. useId
       i. sample
    l. useImperativeHandle
18. map
19. **Props**
    a. default prop
    b. PropDrilling
    c. Children
20. **Components**
    a. Creating Components
    b. Controlled Components
       i. Inputs
    c. Higher order components
    d. Pure components
21. **React Router**
    a. install
    **b. Hooks**
       i. useHistory
    c. use
    **d. Link**
       i. replace
       ii. reloadDocument
       iii. state={}
       iv. - useLocation()
       **v. NavLink**
          1. style={}
          2. -isActive
          3. end
       **vi. Navigate**
          1. useNavigate
          2. navigate(-1)
    **e. Types of Router**
       i. BrowserRouter
       ii. HashRouter
       iii. HistoryRouter
       iv. MemoryRouter
       v. StaticRouter
       vi. NativeRouter

f. params (:id)
g. cont {<name>} = useParams()
h. useSearchParams
**i. Nesting Routes**
    i. index
    ii. location
    iii. shared element with children
    iv. outlet
    v. - useOutletContext()
    vi. Nesting in separate file
    vii. useRoute

## 22. Good to Know

23. Object.entries(e)
24. Icons
25. Experimental Hooks
    a. useEffectEvent
    b. use
    c. useFormStatus
    d. useOptimistic

# OTHERS

1. **SASS**
2. @import "../node_modules/bootstrap/scss/bootstrap";
3. @use & @forward
4. **SOLID Design Principles**
5. Single responsibility principle
6. Open-closed principle
7. Liskov substitution principle
8. Interface segregation principle
9. Dependency inversion principle
10. **REST API**
11. it’s about communication
12. RESTful
13. pros
    a. simple & standardised
    b. scalable & stateless
    c. high performance due to cachings
14. **Request**
    a. General (start line)
        i. method/target/version
    b. operation: get, post, put, delete
    c. endpoint
    d. header
        i. API key
        ii. authentication data
    e. body/ parameter
15. **Response**
    a. General (start line)
        i. version/statuscode/statustext
    b. header
        i. content type
    c. body
        i. requested resource
16. **HTTP Methods**
    a. GET
    b. POST
    c. PUT
    d. DELETE
17. Idempotent
18. Headers
19. Status code
    a. 1xx: Informational
    b. 2xx: Success
        i. 200 - Success
        ii. 201 - Success and created
    c. 3xx: Redirect
        i. 301: moved to new URL
        ii. 304: not changed
    d. 4xx: Client Error
        i. 401: Unauthorised
        ii. 402: 402 Payment Required
        iii. 403: Forbidden
        iv. 404: page not found
    e. 5xx: Server Error
20. MIME type
21. HTTP v2
22. TCP and IP
23. **CI CD (git)**