Software:software is set of programs

, program is set of instruction

or

a software is an interface(bridge) used to connect user with a service provider

eg: facebook (web , mobile app)

user : we are users

service provider: is facebook

classification of software:

1) system software : this s/w are used to enable

machine hardware part to perform operation are

system s/w

eg:

i) operating system

ii) drivers

iii) all utility related s/w

technology: c , c++

------------------------------------------------------

------------------------------------------------------

2) Application software : this are user defined

software (those s/w which are generated as per users

requirement)

eg:

i) mobile app

ii) android

iii) add two numbers

-> types of application software: application s/w are classified on the basis of platform on which they are implemented

\* Console based application

->this application are implemented on OS

->working is done on the basis of commands

->non user freindly

->fast in accessibility

tech: c/c++ , Python , NodeJS

eg:

-> system console

-> billing desk / ticket booking system

------------------------------------------------

\* Desktop based application

->this application are implemented on OS

->this are standalone applications

->this are locally accissible application

tech : JAVA , .NET , Python

eg:

-> antivirus

-> ms office

-> libre office

-> gaming application

-------------------------------------------------------

\* Web based application

->this are globally accessible application

->accessed via web(www)

->user freindly application

tech : jsp , asp , django , flask , ROR , MEAN stack , MERN stack , PHP (f/w - CI , laraval , cake php , zend || CMS : WP , magento , open cart)

eg:

-> facebook

-> flipkart

-> amazon

-> university application

------------------------------------------------

\* Mobile based application

-> this application are implemented on mobile OS

-> mobile apps

tech : Android , IOS , Hybrid apps(IONIC ,

React Native)

JS :-

\* Angular/Cli -> IONIC

\* ReactJS -> React Native

eg:

-> facebook app

-> flipkart app

----------------------------------------------------

Application:

-> console : 10%

-> desktop : 20%

-> web : 40%

-> mobile : 30%

JS Stack : These are bunch of JS based technology used to implement full stack development.

Full stack development :-It is web application development on every instance of web application like front end , back end & server side by taking base of single technology is full stack development.

Those technology which is used to deploy full stack implementation are Full stack technology

JS => MEAN Stack , MERN Stack

NodeJS : Javascript based server side development environment , used to develop server side for web based applications

ExpressJS : It is NodeJS based server side development framework , used to develop , server side for web based application by implementing framework approach

MongoDB : It is application level permanent storage ,used to store data at application level. Apart from traditional databases this database is NoSQL based ,used to manage bulk amount of arbitarily data(unstructured data) in form of JSON(Javascript object notation)

Angular CLI : It is Typescript based UI development framework ,used to develop UI for web based application by following all real world trending requirements.

or

ReactJS : It is ES6 based UI development library ,used to develop UI for web based application by following all real world trending requirements.

JS Stack : These are bunch of JS based technology used to implement full stack development.

Full stack development :-It is web application development on every instance of web application like front end , back end & server side by taking base of single technology is full stack development.

Those technology which is used to deploy full stack implementation are Full stack technology

JS => MEAN Stack , MERN Stack

NodeJS : Javascript based server side development environment , used to develop server side for web based applications

ExpressJS : It is NodeJS based server side development framework , used to develop , server side for web based application by implementing framework approach

MongoDB : It is application level permanent storage ,used to store data at application level. Apart from traditional databases this database is NoSQL based ,used to manage bulk amount of arbitarily data(unstructured data) in form of JSON(Javascript object notation)

Angular CLI : It is Typescript based UI development framework ,used to develop UI for web based application by following all real world trending requirements.

or

ReactJS : It is ES6 based UI development library ,used to develop UI for web based application by following all real world trending requirements.

Web Based Application : This are globally accessible application accessed via www(world wide web) ,this are user freindly applications & most popular among other applications. This are implemented in two parts.

1) Front end : It is application level UI(User interface),the part of application by which user will interact , generated by client side script.

2) Back end : It is application level permanent storage used to store bulk amount of data (Database) ,this storage are linked in web based application via server side script.

--------------------------------------------------------------------------------------------------------------------------------------------

Scripts : It is block of code which directly gets executed ,it is interpreter based , depending on execution platform scripts are classified in two parts.

1) Client side script : Those script which are executed on client (web browser) are client side script

eg : HTML , CSS , JS , JQUERY

2) Server side script : Those script which are executed on server(s/w program) are server side script.

eg : PHP , JSP , NodeJS , ExpressJS

--------------------------------------------------------------------------------------------------

script :

run

demo.html-------------------> output

programming :

compilation run

demo.c -------------> machine code ------------> output

--------------------------------------------------------------------------------------------------

1) Website : Static website : Static web application

2) Web applications : Dynamic website : Dynamic web application

Variable : This are reserve memory block used to store data ,type of data stored in variable will be given by its datatype. As JS(ES5) is object based object oreinted in nature. Hence in JS variables are refrence

JS reference variable : This are reserve memory block capable to store specific type of data, type of data will be manages internally as per data refered

syntax:

var ref\_var\_name

ref\_var\_name=value

or

var ref\_var\_name=value

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

c :

int a;

a=10;

int d=10;

Datatypes : Datatype refers to type of data stored in variable , as JS is fully object oreinted hence variables are refrences & datatype indicate typeof data refered by refrence variable.

Note :-

-> JS is dynamically typed

-> To handle each data prototypes are used (number , string......)

---------------------------------------------------------------

Types of datatypes :-

-> Primitive or Builtin datatypes : Which have default support by language

number , string , boolean , undefined , null

-> Non primitive or User defined datatypes : Which will be implemented as per users requirement

array , object , function

note : Special occuring datatypes(undefined , null)

---------------------------------------------------------------

typeof : It is special operator used to get typeof datatype

syntax : datatype = typeof ref\_var

---------------------------------------------------------------

typecasting : It is a mechanism to convert one type of data to another type

JS comes with number of functions for typecasting

---------------------------------------------------------------

-> WAP to get ASCII value of following keyboard keys

\* space

\* enter

\* tab

\* ascii range of 0-9

-> WAP to predict result of following

case 1 : a=100 , b=200.67

a+b , a/b , a\*b

case 2 : a=100 , b="hello"

a+b , a\*b

case 3 : a="hello" , b="world"

a+b , a\*b

case 4 : a=true , b=false

a+b , a\*b

case 5 : a=true , b=100

a+b , a\*b

Control structure : It is used to deploy code level decision making , iterations & switching at application level

-> Conditional statement :- Used to implement decision making at application level

\* if statement

syntax:

if(condition)

{

statements

}

------------------------------------------------------

\* if-else statement

syntax:

if(condition)

{

statements

}

else

{

statements

}

------------------------------------------------------

\* ladder ifelse statement(if-elseif-else)

syntax:

if(condition)

{

statements

}

else if(condition)

{

statements

}

else if(condition)

{

statements

}

else

{

statements

}

--------------------------------------------------------------

\* switch

--------------------------------------------------------------

--------------------------------------------------------------

-> Control statement :- Used to implement iteration application level(loop)

for number data :-

\* do-while loop

\* while loop

\* for loop

for sequence data :-

\* for-in loop

\* for-of loop

-> Switching statements : It is used to switch from one application end to another

\*break

\*continue

Operators : Those programming entity which perform operation on operands are operator

a+b -> (a+b):expression , (a,b):operands , (+):operator

--------------------------------------------------------------

Classification of operators :- operators are classified on the basis of number of operands

-> Unary operator : 1 operand : ++ , --

-> Binary operator : 2 operand : + , = , > , && , |

-> Ternary or Conditional operator : 3 operand

(condition)?true statement:false statement;

--------------------------------------------------------------

Types of operators :-

-> Arithmatic operator : + , - , \* , / , % , \*\*

-> Assignment operator : = , += , -= , \*= , /= , %=

-> Comparison operator : == , === , > , < , >= , <= , !=

-> Logical operator : && , || , !

-> Bitwise operator : & , | , ^ , ~ , << , >>

-> Increment/Decrement operator : ++ , --

-> Ternary or Conditional operator -- (con)?t:f;

-> Special operator : typeof

---------------------------------------------------------------

---------------------------------------------------------------

Bitwise operator : & , | , ^ , ~ , << , >>

a=121 : 0111 1001

b=113 : 0111 0001

---------------------------------------------------------------

a&b : 0111 0001 : 113

a|b : 0111 1001 : 121

a^b : 0000 1000 : 8

~a : -(a+1) : -122

a<<3 : 968

<--- MSB

0111 1001:LSB

011 1100 1000

b>>2 : 29

<-- MSB

0111 0001 : LSB

0001 1100 01-> ignore

---------------------------------------------------------------

truth table :-

a b a&b a|b a^b

0 0 0 0 0

0 1 0 1 1

1 0 0 1 1

1 1 1 1 0

---------------------------------------------------------------

Bit's representation :

<- MSB LSB

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

--------------------------------

octal

---------------

quad

---------------------------------------------------------------

Decimal representation :

128 64 32 16 8 4 2 1

---------------------------------------------------------------

121 : 0111 1001

113 : 0111 0001

-------------------

-> WAP to get ASCII value of following keyboard keys

\* space

\* enter

\* tab

\* ascii range of 0-9

-> WAP to predict result of following

case 1 : a=100 , b=200.67

a+b , a/b , a\*b

case 2 : a=100 , b="hello"

a+b , a\*b

case 3 : a="hello" , b="world"

a+b , a\*b

case 4 : a=true , b=false

a+b , a\*b

case 5 : a=true , b=100

a+b , a\*b

--------------------------

Control structure : It is used to deploy code level decision making , iterations & switching at application level

-> Conditional statement :- Used to implement decision making at application level

\* if statement

syntax:

if(condition)

{

statements

}

------------------------------------------------------

\* if-else statement

syntax:

if(condition)

{

statements

}

else

{

statements

}

------------------------------------------------------

\* ladder ifelse statement(if-elseif-else)

syntax:

if(condition)

{

statements

}

else if(condition)

{

statements

}

else if(condition)

{

statements

}

else

{

statements

}

--------------------------------------------------------------

\* switch

--------------------------------------------------------------

--------------------------------------------------------------

-> Control statement :- Used to implement iteration application level(loop)

for number data :-

\* do-while loop

\* while loop

\* for loop

for sequence data :-

\* for-in loop

\* for-of loop

-> Switching statements : It is used to switch from one application end to another

\*break

\*continue

-> WAP to add all elements of 7 elements array

-> WAP to create array containing 10-element , where element at odd index(1,3,5,7,9) must be printed

element at even index(0,2,4,6,8) must be added

-> WAP to search element inside array

-> WAP to reverse array elements

0 1 2 3 4

ar=[10,20,30,40,50]

{logic}

ar=[50,40,30,20,10]

-> WAP to sort array elements (bubble sort)

0 - 10 20 20 20 20

1 - 20 10 30 30 30

2 - 30 30 10 40 40 pass I(no of pass=total ele-1)

3 - 40 40 40 10 50

4 - 50 50 50 50 10

-> WAP to get max element from 10 elements array

---------------------------------------------------

JSON : key points

-> It stands for Javascript object notation

-> It is string formatted notation of JS object

-> It is light weight proccess , i.e easy to transmit , proccess & manage big amount of data

-> It is considered as common way to handle data between various application ends like web-web , web-mobile , api's etc

-> It is supported by almost all technologies of real world

Conversion methods :-

JSON.stringify(JS\_obj) : to convert JS object to string

JSON.parse(JS\_json) : to convert JS json to object

-> JSON viewer tool : https://jsonviewer.stack.hu/

-> JSON resource : https://jsonplaceholder.typicode.com/

https://dummyjson.com/products

---------------------------------------------------------------

JSON : key points

-> It stands for Javascript object notation

-> It is string formatted notation of JS object

-> It is light weight proccess , i.e easy to transmit , proccess & manage big amount of data

-> It is considered as common way to handle data between various application ends like web-web , web-mobile , api's etc

-> It is supported by almost all technologies of real world

Conversion methods :-

JSON.stringify(JS\_obj) : to convert JS object to string

JSON.parse(JS\_json) : to convert JS json to object

-> JSON viewer tool : https://jsonviewer.stack.hu/

-> JSON resource : https://jsonplaceholder.typicode.com/

https://dummyjson.com/products

-----------------------------------------

Introduction

JavaScript evolved in a very short time from callbacks to promises (ES2015), and since ES2017 asynchronous JavaScript is even simpler with the async/await syntax.

Async functions are a combination of promises and generators, and basically, they are a higher level abstraction over promises. Let me repeat: async/await is built on promises.

----------------------------------------------------------------------------------------------------------------------------------------------------

Why were async/await introduced?

They reduce the boilerplate around promises, and the "don't break the chain" limitation of chaining promises.

When Promises were introduced in ES2015, they were meant to solve a problem with asynchronous code, and they did, but over the 2 years that separated ES2015 and ES2017, it was clear that promises could not be the ultimate solution.

Promises were introduced to solve the famous callback hell problem, but they introduced complexity on their own, and syntax complexity.

They were good primitives around which a better syntax could be exposed to the developers, so when the time was right we got async functions.

They make the code look like it's synchronous, but it's asynchronous and non-blocking behind the scenes.

----------------------------------------------------------------------------------------------------------------------------------------------------Introduction

JavaScript evolved in a very short time from callbacks to promises (ES2015), and since ES2017 asynchronous JavaScript is even simpler with the async/await syntax.

Async functions are a combination of promises and generators, and basically, they are a higher level abstraction over promises. Let me repeat: async/await is built on promises.

----------------------------------------------------------------------------------------------------------------------------------------------------

Why were async/await introduced?

They reduce the boilerplate around promises, and they "don't break the chain" limitation of chaining promises.

When Promises were introduced in ES2015, they were meant to solve a problem with asynchronous code, and they did, but over the 2 years that separated ES2015 and ES2017, it was clear that promises could not be the ultimate solution.

Promises were introduced to solve the famous callback hell problem, but they introduced complexity on their own, and syntax complexity.

They were good primitives around which a better syntax could be exposed to the developers, so when the time was right we got async functions.

They make the code look like it's synchronous, but it's asynchronous and non-blocking behind the scenes.

----------------------------------------------------------------------------------------------------------------------------------------------------WAP to check a number is divisible by 10 ,

if divisible print its factorial

if not print its table

Note : using all async codes

--------------------------------

Functions : This are block of code which will be executed when it is called , it is used to deploy code reusability i.e write once & run any number of times. As JS is fully object oriented hence function is also considered as reference.

Types of functions : On the basis of function implementation , function are of following types :-

-> Default function

syntax :

function function\_name(args) -> if any

{

statements

return value -> if any

}

calling syntax : function\_name()

------------------------------------------------------

-> Anonymous or Variable function

syntax :

var function\_name=function(args){ -> if any

statements

return value -> if any

}

calling syntax : function\_name()

------------------------------------------------------

-> Arrow or Expression function

syntax :

var function\_name=(args)=>{ -> if any

statements

return value -> if any

}

calling syntax : function\_name()

------------------------------------------------------

-> Fat arrow or Lamda expression function

syntax :

((args)=>{

statements;

})();

key points :-

\* it will be executed where it is created

\* it will be auto invoked , no need of explicit calling

\* cannot return value

\* used to supply function as arguments

------------------------------------------------------

-> Callback function

-> Prototype function

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

Classification of function : On the basis of arguments & return value functions are classified

-> function without arguments & without return value

-> function with arguments & without return value

-> function without arguments & with return value

-> function with arguments & with return value

Node : It is Javascript based server side development environment , used to develop server side for web based application , along with it , it is used to generate network based application.

Node Development Environment :-

web : https://nodejs.org/en/

LTS : 22.14.0

Current : 23.9.0

Invented by Ryan Dahl in 2009

-----------------------------------------------------------------------

Features of NodeJS :-

-> Open source : It means freely available , i.e all node environments , module , tools are free to use

-> Platform independency : Node applications are platform independent , if provided with proper execution environment. But node environments are platform dependent.

Node App = Source code + App dependency

or modules

-> Portable : Node application can be carried from one platform to another

-> Modules or Dependency : It is collection of JS refrence variables , function or classes , this modules can be linked at application level if required & can be removed if not , modules can be generated as per users requirement & will be considered as custom modules.

eg : fs , http , url , events , express , @angular/cli , create-react-app ........

-> NPM : It stands for Node package manager , it is online repository which provide default support to node environment , those packages which are not in node env can be downloaded from NPM

-> package.json : It is node application configuration file , used to store application configs like

\* app name

\* app version

\* test commands

\* list of dependency

\* author info

-> Cross platform

-> Architecture neutral

-> Object oreinted

Event looping proccess : https://www.webdevolution.com/blog/Javascript-Event-Loop-Explained

Synchronous or Blocking functions & Asynchronous or Non-Blocking functions

It is function classification on the basis of function working approach

-> Synchronous or Blocking functions : Those function which on self execution will block further code from execution , till complete execution are blocking or synchronous function

-> Asynchronous or Non-Blocking functions : Those function which on self execution will not block further code from execution are non blocking or asynchronous function

Note : By default Node is unithreaded in nature , by the implementation of Async functions it act multithreaded in nature

-------------------------------------------------------------------

---------

--------- p1

---------

| |

| | p2 (function defination & calling)

| |

---------

--------- p3

---------

-------------------------------------------------------------------

Event looping proccess : https://www.webdevolution.com/blog/Javascript-Event-Loop-Explained

Synchronous or Blocking functions & Asynchronous or Non-Blocking functions

It is function classification on the basis of function working approach

-> Synchronous or Blocking functions : Those function which on self execution will block further code from execution , till complete execution are blocking or synchronous function

-> Asynchronous or Non-Blocking functions : Those function which on self execution will not block further code from execution are non blocking or asynchronous function

Note : By default Node is unithreaded in nature , by the implementation of Async functions it act multithreaded in nature

-------------------------------------------------------------------

---------

--------- p1

---------

| |

| | p2 (function defination & calling)

| |

---------

--------- p3

---------

-------------------------------------------------------------------

constructor :-

-> It is member method of class

-> It is used to initialize member of class with default value

-> No need of explicit calling

-> It will be auto invoked when object of class is created

-> In ES6 constructor have fixed name

constructor()

{

statements

}

-> types :

\* default constructor or 0-paramatrised constructor

\* paramatrised constructor

Note : To implement destructor , type script is to be implemented

ES6 : It is ECMA script standard which was launched in year 2015 & all above versions follow same approach to write code blocks

ES5 <= : @2014 -> object based OO

ES6 >= : @2015 -> class based OO

Latest version : @2024 ES15

-------------------------------------------------------

ES6 Features :-

-> Use strict is default enable & hoisting is implemented

-> Implementation of let , const like cases

-> Use of template literals (``)

-> Implementation of arrow function

-> Implementation of rest & spread operator (...)

-> It is class based object oreinted in nature

-> Follows all oops features : Encapsulation , Abstraction , Polymorphism & Inheritance

-> Transpilers & Transpilation : Transpilation is a process to convert ES6 compatible code to ES5 compatible code & module which make conversion is transpiler

babel : transpiler (task runner)

webpack : task bundeller

execute

source code ----------> platform

(ES5) (browser,node env)

transpilation execute

source ----> ES5 ----> platform

code

(ES6) (browser,node env)

Typescript over ES6 :-

-> Generic behaviour : In TS datatypes can be explicitly implemented

-> Implementation of typed collections

-> keywords like public private protected

-> Support to interface

----------------------------------------------

Inheritance (is-a relationship) : It is an object oreinted feature which signify sharing of properties & behaviour of one class in another class is inheritance , used to implement modularity at application level

ERP : College resources

Admin , Student , Accounts , Staff

Each module will be implemented as class & used to deploy code reusability & re-implementation

is-a

Class A <----- Class B

class A : Base - Parent - Super class

class B : Derived - Child - Sub class

syntax :

class Base

{

data member

member methods

}

class Derived extends Base

{

data member

member methods

}

Types of Inheritance :-

-> Single level inheritance

-> Multilevel level inheritance

-> Hierarchical inheritance

-> Multiple inheritance

-> Hybrid inheritance

Note : In case of multiple inheritance , multiple base class provides property to single derived class , in this case if multiple base contain data or functionality with same name is considered as ambiguity problem

class A

<------- class C

class B

\* Due to ambiguity problem , multiple inheritance is not supported in ES6

\* To implement interface Typescript is used

--------------------------------------------------------------

Overriding : In case of inheritance base class property or functionality is accquired by derived class , it is possible that both may contain data or functionality with same name. In this case derived class overrides base class implementations

--------------------------------------------------------------

Polymorphism : It is an OOPS feature which signify

poly means many

morphism means forms

i.e one entity having different different implementation

-> Overloading - (...) Rest operator

-> Overriding - In case of inheritance , base class property & behaviour is acquired by derived class , in this case it is possible that both base & derived may contain data or functionality with same name considered as overriding.

In overriding base class implementation are overriden by derived class

-------------------------------------------------------

Abstraction : It is an object oreinted feature which is used to implement application level security , by making application level data secure & functionality globally accessible.

Note : In ES6 abstraction is all about nomenculature (naming conventions) -> a:data member , aData:method

-> public parameter : a , aData()

-> private parameter : #a , #aData()

--------------------------------------------------------------

Encapsulation : It is an object oriented feature , which signify wrapping of data & functionality into a single unit is encapsulation , as it wraps data hence also considered as data binding.

Note : In oops object is unit which carries encapsulation

--------------------------------------------------------------

Modules or Dependency : It is a collection capable to store data refrences , function & classes or prototypes. This is used to deploy code reusability (write once & run any number of times) & re-implementation.

This modules can be linked at application level as required , if not required they can be removed.

-------------------------------------------------------------------

Types of Modules :-

1) Custom or User defined modules : those module which are created as per users requirement

2) Predefined or Builtin modules : those modules which are available by default are predefined

eg: fs , http , url , events , express , @angular/cli , create-react-app....

-------------------------------------------------------------------

-------------------------------------------------------------------

Depending on avaibility builtin modules are further classified :-

Level 1 : Application level non installable dependency , this modules are available in node environment. Eg: fs , events , http , url , path....

Level 2 : Application level installable dependency , this modules are available on NPM. Eg: express , express-session , express-fileupload....

command : npm install <package\_name>

Level 3 : Global dependency : not linked at application level but added in environment. Eg: nodemon , create-react-app , @angular/cli

npm install <package\_name> -g

-------------------------------------------------------------------

-------------------------------------------------------------------

oops : It stands for object oreinted programming style or structure , it is a way to make programming implementation to apply code reusability , re-implementation , application level security & modularity at application level.

To deploy above implementations oops features are used :-

Features of OOPS

-> Encapsulation

-> Abstraction

-> Polymorphism

-> Inheritance

------------------------------------------------------

------------------------------------------------------

Object :-

-> It is real world entity : Those entity which have its own state , data composition is real world entity

state:data composition:properties -> machine , accessories , body (data member)

car : object

behaviour:functionality -> it moves (member method)

eg: ar : array object (ar.length , ar.reverse())

------------------------------------------------------

-> It is run time entity : When a object will traverse from one location to another it will carry its state & behaviour with it hence considered as run time entity

------------------------------------------------------

-> It is instance of class : A class will come in existance if object of class is created & object is blue print of class implementation (result)

------------------------------------------------------

------------------------------------------------------

Class :-

-> It is collection of similar type of object : A class will come in existance when object is created , all object generated for same class will be of similar type (comman state & behaviour)

-> It is collection of data member & member method

------------------------------------------------------

------------------------------------------------------

syntax for class :-

class ClassName

{

data member

member methods

}

syntax for object :-

var object\_name = new ClassName();

------------------------------------------------------

------------------------------------------------------

Encapsulation : It is an object oreinted feature which signify , wrapping of data & fuctionality into a single unit is considered as encapsulation. As it binds data it is also considered as feature of data binding.

Note : In OOPS object is unit which carry encapsulation

------------------------------------------------------

------------------------------------------------------

URL : http://localhost:8081 /login/p2/p3

s1 s2

URL+Data : http://localhost:8081 /login/p2/p3

s1 s2

?username=admin&password=admin123

s3

s1 : domain

s2 : url path

s3 : url data (query string)

req.url : s2+s3

Note : URL routing must be done as per url path

Express : It is NodeJS based server side development framework , used to develop server side for web based application by following framework approach.

Framework : it is a approach to develop a application in distributed form , i.e to make different implementations of MVC architecture

MVC :-

Model : DB related implementation

Views : UI related implementation

Controller : Bussiness logic unit

It was developed to generate REST API by using node programming

Note : Developed on 16th nov 2010 , developed by IBM

LTS : 4.19.2

-----------------------------------------------------------

Features of ExpressJS :-

-> Open source : It means express is free to used , can be easily downloaded from npm

-> Platform independent : It is platform independent in nature , i.e a application can be implemented on different platforms. Only need is to check dependency are available or not

express app = source code + dependency(modules)

-> Portable

-> Framework architecture : It is approach to write application , express follows MVC architecture to write application

MVC -> M:models , V:views , C:controller

M : It is Database related implementation , used to develop database related logic in application

V : It is UI related implementation , used to develop UI application

C : It is Bussiness logic unit, used to develop , logic related to application working

-> Middleware functions : This are functions capable to work in request response procedure , without disturbing actual req-res procedure.

-> Database support : Express comes with various modules to make connectivity with different databases like MySQL , MongoDB , sqlite3 ,.....

-> Template engine & pages :- This are UI pages capable to load both client side as well as server side content , here client side content will be executed on browser while to execute server side content we use template engine.

Template engine : This are modules capable to execute template script implemented on template pages

Template script & template engine

pug script : pug engine

ejs script : ejs engine

jade script : jade engine

demo.pug

html

head

title PUG Demo

body

h1 Welcome to world of PUG

demo.ejs

<html>

<head>

<title>EJS Demo</title>

</head>

<body>

<h1>Welcome to world of EJS</h1>

</body>

</html>

-> Automatic management for

\* req-res process

\* response header management

\* built in methods for URL routing & method checking

Http methods & Data processing :-

base url : http://localhost:3000/admin/manageusers/

--------------------------------------------------

method POST : used to send request & to add content in resource. This method is used to generate save (insert) api i.e those api used to save data in resource

It manages data internally hence secure to transfer data

method : POST

URL+data : data will be managed internally using request body

eg url : http://example.com/

http://localhost:3000/admin/manageusers/

Note : to fetch request body params body parsing is used

--------------------------------------------------

method GET : used to send request & to bring response content. This method is used to generate fetch api i.e those api used to fetch resource data

method : GET

URL+data : data will be managed by using query string

eg url : http://example.com/?p1=v1&p2=v2.....

http://localhost:3000/admin/manageusers/?username=admin@gmail.com&password=admin123

Note : to fetch url params url parsing is used

--------------------------------------------------------------

Note : Apart from methods , url params can be used to attach data with any url request

URL+data : data will be managed by using url params

eg url : http://example.com/param\_value1/param\_value2......

Middleware : This are functions capable to work in between request response mechanism , without disturbing actual application flow are middleware.

actual req-res :-

response

<---------

client side ----------> server side

(web browser) request

middleware req-res :-

response

<------------------------------

client side ----> middleware ----> server side

(web browser) req req

key point :-

-> get request

-> if required work on request

-> generate response

-> send response to next server implementation

-> in some cases may respond directly to client side

-> will not affect req-res mechanism

-----------------------------------------------------------

Types of middleware functions :-

-> Application level middleware

-> Route level middleware

-> Builtin middleware

-> Third party middleware

-----------------------------------------------------------

-> Route level middleware : It is a middleware used to divide routes as per code requirement , generally routes are divided as per users or application level working modules

ERP : product resources

modules : Visitor , User , Admin

visitor capability : /home , /about , /contact , /service , /register , /login

user capability : /userhome , /cpuser , /epuser , /viewcategory

admin capability : /adminhome , /cpadmin , /epadmin , /addcategory , /addsubcategory , /addproduct

------------------------------

mongodb backup : For mongodb backup & restore , mongodb tools are required, install mongodb tools

------------------------------------------------------

to backup : mongodump command is used , move to specific location for backup

-> to back up all database : entire mongodb

mongodump

-> to back up specific database

mongodump --db databasename

-> to back up specific collection

mongodump --db databasename --collection collectionname

------------------------------------------------------

to restore : mongorestore command is used , move to specific location to restore , where backup is kept

-> mongorestore

window install :-

url : https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/#install-mongodb-community-edition

joins :

https://hevodata.com/learn/mongodb-join-two-collections/

--------------------------------------------------------------

React environment setup :- create-react-app is depricated from react version 19 , instead of create-react-app vite is used

------------------------------------------------------

------------------------------------------------------

Vite is a fast and modern build tool for creating web applications. It increases the development process by providing faster build times and better performance. Some of the benefits of using React with Vite are mentioned below:

-> Fast updates without page reloads.

-> Faster and smaller production builds.

-> Automatic JSX handling.

-> Faster Build

-> Steps to set up ReactJS with Vite

------------------------------------------------------

Steps to set up ReactJS with Vite

Step 1: Install the NodeJS

To install NodeJS and npm, visit the NodeJS official website and download the latest stable version.

command to check version : node -v , npm -v

------------------------------------------------------

Step 2: Create a New Project with Vite

Vite provides a simple way to scaffold new projects. You can use the following command to create a new React project with Vite.

command : npm create vite@latest my-react-app

-> my-react-app is the name of your project. You can change it to any name you prefer.

-> npm create vite@latest my-react-app: This command initializes a new Vite project with a React template.

Replace my-react-app with your desired project name.

-> cd my-react-app: Navigate into your newly created project directory.

------------------------------------------------------

Step 3: Select a Framework

React

------------------------------------------------------

Step 4: Select a Variant

React : Javascript

-------------------------------------------------------

Step 5: Install Dependencies

command : npm install

-------------------------------------------------------

Step 6: Start the Server

command : npm run dev

-------------------------------

-> React features : Currently, ReactJS gaining quick popularity as the best JavaScript framework among web developers. It is playing an essential role in the front-end ecosystem. The important features of ReactJS are as following.

\* JSX

\* Components

\* One-way Data Binding

\* Virtual DOM

\* Simplicity

\* Performance

------------------------------------------------------

JSX :

JSX stands for JavaScript XML. It is a JavaScript syntax extension. Its an XML or HTML like syntax used by ReactJS. This syntax is processed into JavaScript calls of React Framework. It extends the ES6 so that HTML like text can co-exist with JavaScript react code. It is not necessary to use JSX, but it is recommended to use in ReactJS.

<app/> , <header/> , <footer/> , <nav/>

------------------------------------------------------

Components :

ReactJS is all about components. ReactJS application is made up of multiple components, and each component has its own logic and controls. These components can be reusable which help you to maintain the code when working on larger scale projects.

function ComponentName()

{

logic(ES6)

return(

JSX

);

}

------------------------------------------------------

One-way Data Binding :

ReactJS is designed in such a manner that follows unidirectional data flow or one-way data binding. The benefits of one-way data binding give you better control throughout the application. If the data flow is in another direction, then it requires additional features. It is because components are supposed to be immutable and the data within them cannot be changed. Flux is a pattern that helps to keep your data unidirectional. This makes the application more flexible that leads to increase efficiency.

------------------------------------------------------

Virtual DOM :

A virtual DOM object is a representation of the original DOM object. It works like a one-way data binding. Whenever any modifications happen in the web application, the entire UI is re-rendered in virtual DOM representation. Then it checks the difference between the previous DOM representation and new DOM. Once it has done, the real DOM will update only the things that have actually changed. This makes the application faster, and there is no wastage of memory

------------------------------------------------------

Simplicity :

ReactJS uses JSX file which makes the application simple and to code as well as understand. We know that ReactJS is a component-based approach which makes the code reusable as your need. This makes it simple to use and learn.

------------------------------------------------------

Performance :

ReactJS is known to be a great performer. This feature makes it much better than other frameworks out there today. The reason behind this is that it manages a virtual DOM. The DOM is a cross-platform and programming API which deals with HTML, XML or XHTML. The DOM exists entirely in memory. Due to this, when we create a component, we did not write directly to the DOM. Instead, we are writing virtual components that will turn into the DOM leading to smoother and faster performance.

------------------------------------------------------

React Materials

------------------------------------------------------

Animation modules

------------------------------------------------------

ReactJS is a declarative , efficient , and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook (meta) and later used in its products like WhatsApp & Instagram.

web : https://react.dev/

https://en.wikipedia.org/wiki/React\_(JavaScript\_library)

LTS : 19.1 (5 Dec 2024)

Author : Jordan Walke

Release : May 29, 2013

Developed in : Javascript

meta

---------------------------------------------------------------------------------------------------------

Why we use ReactJS?

-> The main objective of ReactJS is to develop User Interfaces (UI) that improves the speed of the apps.

-> It uses virtual DOM (JavaScript object), which improves the performance of the app. The JavaScript virtual DOM is faster than the regular DOM.

-> We can use ReactJS on the client and server-side (web service) as well as with other frameworks. It uses component and data patterns that improve readability and helps to maintain larger apps.

ReactJS or React.js : It is Javascript based library used to develop user interface for web based application.

It is used to generated UI by following all real world trending requirements

-> History

Node version : 16.13.0

NPM : 8.1.0

React LTS : 17.0.2 (2013)

--------------------------------------------------

React vs Angular

-> React

\* It is UI development library

\* It follows MVC(model-view-controller) approach to write UI for application

\* It is implemented on the basis of ECMA script 6 (ES6 -> 2015)

\* External JS can be embedded

\* Used to develop SPA & hybrid mobile app (React Native)

\* Multiple tools are used for testing purpose

\* Works on Virtual DOM

\* Eg : Meta or Facebook , Netflix , yahoo.mail , Airbus........

--------------------------------------------------

--------------------------------------------------

-> Angular

\* It is UI development framework , entire UI will be develop using architecture

\* It is based on component driven architecture

\* It is implemented on the basis of Microsoft Typescript

\* External JS can not be embedded

\* Used to develop Web UI on basis of component based architecture & hybrid mobile app (Angular Native OR Ionic)

\* Single tools is used for testing purpose (karmajs)

\* Works on real DOM

\* Google , Paypal , GoogleGloud.....

--------------------------------------------------

JS Stack : These are bunch of JS based technology used to implement full stack development.

Full stack development :-It is web application development on every instance of web application like front end , back end & server side by taking base of single technology is full stack development.

Those technology which is used to deploy full stack implementation are Full stack technology

JS => MEAN Stack , MERN Stack

NodeJS : Javascript based server side development environment , used to develop server side for web based applications

ExpressJS : It is NodeJS based server side development framework , used to develop , server side for web based application by implementing framework approach

MongoDB : It is application level permanent storage ,used to store data at application level. Apart from traditional databases this database is NoSQL based ,used to manage bulk amount of arbitarily data(unstructured data) in form of JSON(Javascript object notation)

Angular CLI : It is Typescript based UI development framework ,used to generate UI for web based application by following all real world trending requirements.

or

ReactJS : It is ES6 based UI development library ,used to generate UI for web based application by following all real world trending requirements.

--------------------------------------------------------------------------------------------------------------------------------------------

basic themes : https://all-free-download.com/free-website-templates/free-html-css-templates.html

complex themes : https://www.free-css.com/free-css-templates