Express JS

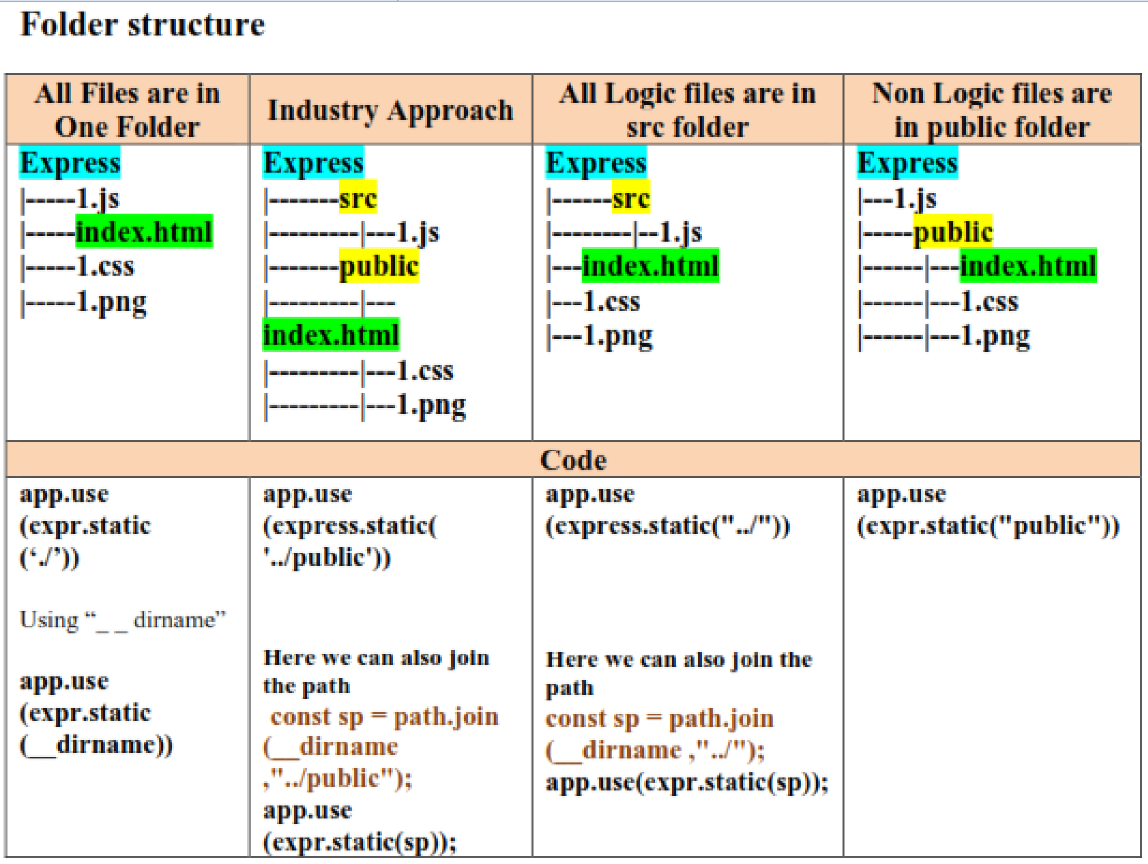
Topic	Syntax
Basic Requirement for Express	
Import Express	const express = require('express');
Initialize App	const app = express();
Start Server	app.listen(port, () => { ... });
Two Method for route	
GET Route	app.get('/route path', (req, res) => { ... });
POST Route	app.post('/route path', (req, res) => { ... });
Response methods	
Set Header	res.set('Content-Type', 'text/html');
res.write()	res.write('Some content');

Send Response (Text)	res.send('Hello World!'); // Must blank if use res.write in same route.
	
	Example
	app.get ("/", (req,res)=>
	{
	    res.set ("content-type","text/plain");
	    res.send ("<h1>Hello</h1>");
	});
	
	Or
	app.get ("/about", (req,res)=>
	{
	    res.set ("content-type","text/html");
	    res.write ("Hello");
	    res.send ();
	});
	• If you pass in a string, it sets the Content-Type header to text/html .
	• If you pass in an object or an array, it sets the application/json Content-Type header, and parses that parameter into JSON.

Send JSON	res.json({ key: 'value' }); // Also close connection
res.sendFile()	res.sendFile(__dirname + '/index.html');
Redirect	res.redirect('/new-url');
Serve Static Files	

Serve Static Files	app.use(**express**.**static**('folder')); // incase index.html
	app.use(**express**.**static**('folder',{index: " filename"}); // any other name

Serve Static (with path module)	app.use(**express**.**static**(**path**.**join**(__dirname, 'folder')));



Folder structure

All Files are in One Folder	Industry Approach	All Logic files are in src folder	Non Logic files are in public folder
Express |-----1.js |-----index.html |-----1.css |-----1.png	Express |-------src |---------|---1.js |-------public |---------|--- index.html |---------|---1.css |---------|---1.png	Express |------src |--------|--1.js |---index.html |---1.css |---1.png	Express |---1.js |-----public |------|---index.html |------|---1.css |------|---1.png
Code			
app.use (expr.static ('./')) Using "_ _ dirname" app.use (expr.static (__dirname))	app.use (express.static('../public')) Here we can also join the path const sp = path.join (__dirname ,"../public"); app.use (expr.static(sp));	app.use (express.static("../")) Here we can also join the path const sp = path.join (__dirname ,"../"); app.use(expr.static(sp));	app.use (expr.static("public"))

Route Parameters	

Route Parameters	app.get('/calendar/:**day**/event/:**ename**', (req, res) => {
	});
	To access mention **req.params**
	url: localhost:portno/calendar/Monday/event/birthday
	o/p: {"day":"monday","ename":"birthday"}

Data fetch from Get/Post

Use get method **Form "method= get"**	app.**get**('/route', (req, res) => { ... }); To access **req.query**

Use post method
Form
"method=post, action=
"/data"
	app.**use**(express.**urlencoded**({ extended: true }));
	app.**post**('/route', (req, res) => { ... });
	To access **req.post**
	
	Note: Use app.post on same route mention in form's action. For different route use app.get then,
	app.post("/data", (req, res) => { ... })
	app.get("?aboutdata", (req, res) => { ... })

Handle 404	app.**use**((req, res) => { res.status(404).send('Not Found');});

app.get('/'): Handles only GET requests to the exact / path.
Example: Only runs for http://localhost:3000/.

app.use('/'): Handles all HTTP methods for all paths starting with /.
Example: Runs for /, /contact, /about, /anything.

Middleware

Middleware Use	app.use((req, res, **next**) => { ... **next**(); });
Chained Middleware	app.use('/route', cb1, cb2, cb3); cb1,cb2,cb3 are functions which handled in sequence. Mention next() in cb1, cb2 and mention res.send() in cb3

Cookie

Cookie Parser	const **cp**=require("cookie-parser");
	app.use(**cp**());

Set Cookie	res.cookie('name', 'value',option); option: {maxAge:2000} 2000 ms cookie will expire
Clear Cookie	res.clearCookie('name');
Read Cookie	req.cookies.name_of_cookie;

Session	

Session Setup	const **es**=require(“express-session”);
	app.use(**es**({ **secret**: 'key',
	                    resave: false, //optional
	                    saveUninitialized:false //optional
	 }));

Set Session Value	**req**.**session**.property = value;
Destroy Session	**req**.**session**.destroy(err => { ... });
Pug – View Engine	

Set Views Folder	app.set(view engine,”pug”)
	app.set('views',__dirname);
	app.get(“/”,(req,res)=>{
	   res.render(”test”)
	})

Set view engine (If only view engine set mention __dirname in render)	app.set(view engine,”pug”)
	app.get(“/”,(req,res)=>{
	    res.render(__dirname+”/test”)
	})

Syntax in pug	h1
	    Hello
	p(style=”color:cyan” title= “Welcome to pug”)
	    b
	      i PUG
	
	O/P:
	Hello ← h1
	PUG ← in bold and italic, cyan color, with title tooltip

Raw/Plain text	|this is plain text H1 hi |hello How

	O/P: • this is plain text → Outputs raw text directly into the document, outside of any tags. • H1 hi → Starts an <h1> element with text hi. • hello indented under H1 → Appended as plain text **inside** the <h1>. • How will treated as <how></how> and display nothing
JavaScript Code	- let msg = "Hello"
Interpolation based output	#{variable} means #{msg}
Inline Output	h4= msg
HTML Comments(buffered)	// This is a comment
Block Comments (ignored/unbuffered)	//- Hidden comment
Multer – File Upload **Must mention enctype="multipart/form-data" in form.html**	
Import multer	const multer = require('multer');

diskStorage Setup	const storage = multer.diskStorage({
	destination: 'updocumnet' // Folder name "updocument" where uploaded files are saved
	filename: (req, **file**, cb) => cb(null, file.originalname)
	});

File key names	**fieldname** :Field name specified **in the form**. **originalname:** Name of the file on the user's computer. **size**: Size of the file in bytes. **destination:** The folder to which the file has been saved. **filename**: The name of the file within the destination.
File Size Limit	limits: { fileSize: 1024 * 1024} (1MB)
Initiate Multer () middleware	var upload = multer({ storage: **store** }).**single('myfile')** //**single file** fetch from **req.file**

	var upload = multer({ storage: **store** }).**array('myfile',5)**
	// **multiple** files fetched from **req.files** accept upto 5 file, also mention multiple attribute in form.html
	
	//myfile is field name mention in form
	<input type= “file” name= “myfile” multiple>
	
	//to inset limit
	var upload = multer({ storage: **store,** limits: { fileSize: 1024 * 1024} }).single('myfile')

RestAPI	
Create Router in data file	const router = express.Router(); also export it
Define Route	router.get('/users', (req, res) => { ... });
Use in App file	app.use('/api', router);
Route Parameters	/users/:id, access via req.params.id
Nodemailer	
Import Nodemailer	const nodemailer = require('nodemailer');
Create Transporter	const transporter = nodemailer.createTransport({ host: 'smtp.gmail.com' port: 587/465 auth: { user: 'user@email.com', pass: 'App password' } })
Send Mail	transporter.sendMail({ to: “…….”, subject:…………….., text or html:………………. })