4 => working with dynamic content in node js

So how do we pass our input data from user to server for the request so we can pass it in the form of variable and array of object.

const products = [];

// /admin/add-product => POST

router.post('/add-product', (req, res, next) => {

  products.push({ title: req.body.title }); // push object inside array

  res.redirect('/');

});

exports.routes = router;

exports.products = products; // export it so it used by other files too

which is passing in the product page as admin data.product

const adminData = require('./admin');

const router = express.Router();

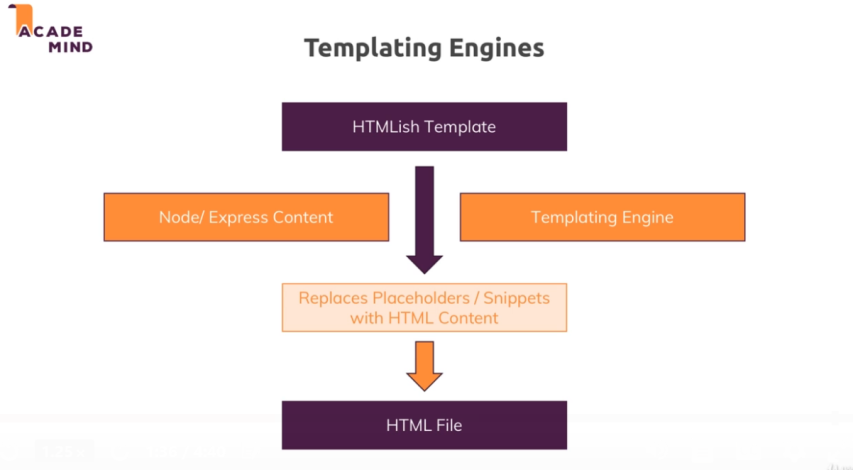
router.get('/', (req, res, next) => {

  console.log('shop.js', adminData.products);

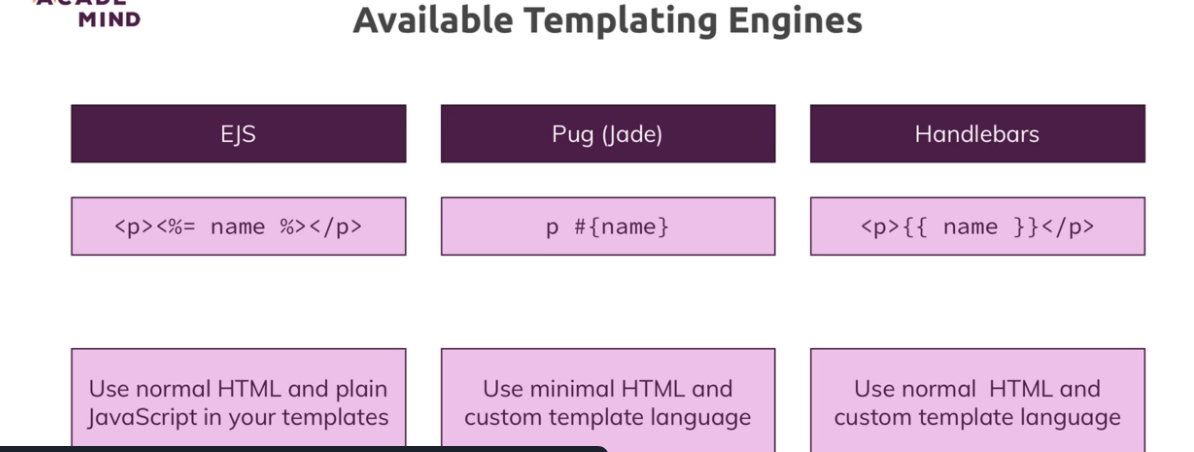
  res.sendFile(path.join(rootDir, 'views', 'shop.html'));

});

So to putting dynamic content to our html pages we uses templates to add templates we use templates engine so htmlish contains all the css,js and html with some place holders in between



so the templates engines are of different types some of them are and they use different syntax and you can add the dynamic content in it



So all the template can add dynamic content such as name in it but all have different syntax for that

* Installing templates

So we have to install these templates and we can do it by saving into dependencies



* Use of pug in place of html

So pug consider good in place of html if we are sending the file

To use pug first we have to define in our express that we also want to view the files with .pug extention

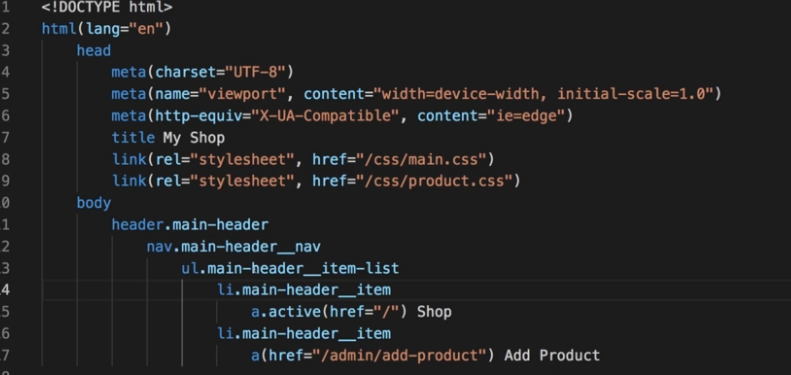
We are doing this so we can send the data in form of pug

We can do this by setting the global execution value so app.set will help us to set the value



So we are setting vie engine as pug and views as default

So the structure of pug file is



The class is defined by .---- and the links or things other the class and id will defined inside the ( curli breces ) and text at the end. So it work like html but in short line of code

Now to send it we have to give the path when ever we have to send and or this we have to render the file 

So it will render the file shop.html and get used in app as view engine

So we use pug to add some dynamic content and functionality as well so with pug we can call the data at runtime and use it

routs.get('/',(req,res,next) => {

    //res.send('<h1> send the data using Express </h1>');

    const products = product.product;

**res.render('shop',{prods: products, docTitle: 'shopPage'});**

    // now we can use product object b pods in our pug file as well as docTitle

});

Now we can add objects to the shop.pug file we can use the prods and docTitle and as we know product is an array of title page

So now our pug file look likes so we can add objects dynamically by using **#{inside it}**

doctype html

html(lang="en")

    head

        meta(charset="UTF-8")

        meta(http-equiv="X-UA-Compatible", content="IE=edge")

        meta(name="viewport", content="width=device-width, initial-scale=1.0")

        title **#{docTitle}**

    body

        header.head <!-- . add class to it -->

            ul

              li

                a.link(href="/product") go to link

        main

            if prods.length > 0

                .grid

                    each product in prods

                        article.card.product-item

                            header.card\_\_header

                                h1.product\_\_title **#{product.title}**

                            div.card\_\_image

                                img(src="data:image/jpeg;base64,/9j/4AAQSZJRgAB”, alt="A Book")

                            div.card\_\_content

                                h2.product\_\_price $19.99

                                p.product\_\_description A very interesting book about so many even more interesting things!

                            .card\_\_actions

                                button.btn Add to Cart

            else

                h1 No Products

we are using doctitle and productitle inside inside the pug file

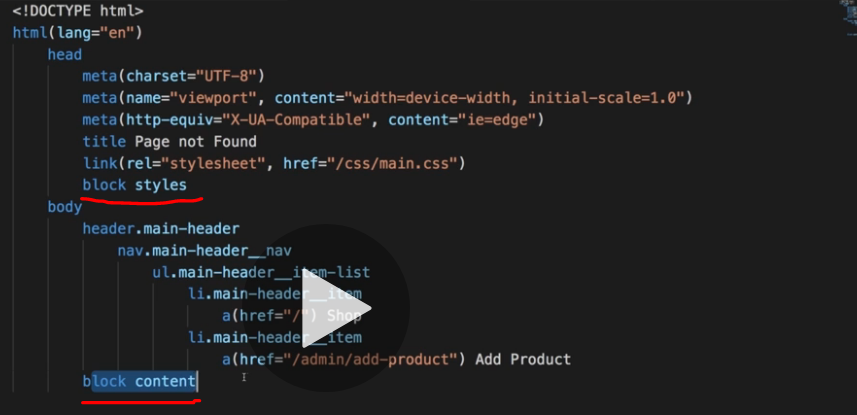
* Use of layout to store and optimize the repetitive code in a single folder

So we use layout to reduces the repetitive code, such as the html5 starting links,title,things before body and inside body two which is used repetitively in more then 1 file

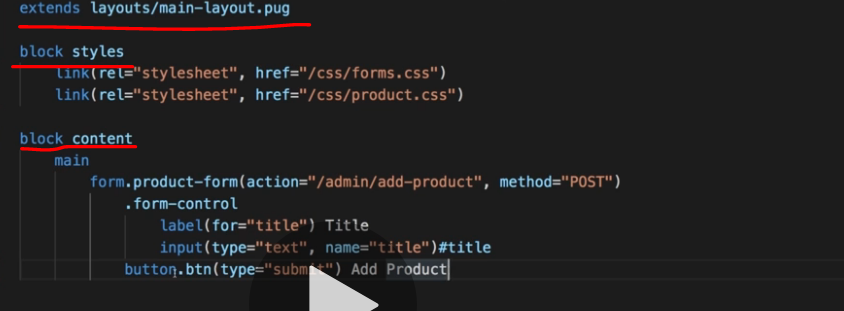
So if we have repetitive code using more then one time in more then 1 file so then we uses layout in it

So we can create a main layout pug file in it

But what happened if we wont to change a single line of code which is different for the different files. So we can name that line as **block** and name the line so that we can add something later in it



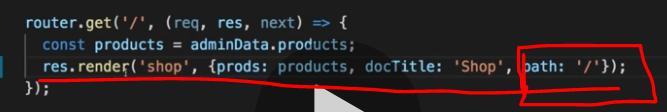
And then we can call this layout by **extends and the layoutfilepath** and fill the blocks if we want you can skeep all the blocks or any block that you want or not in use



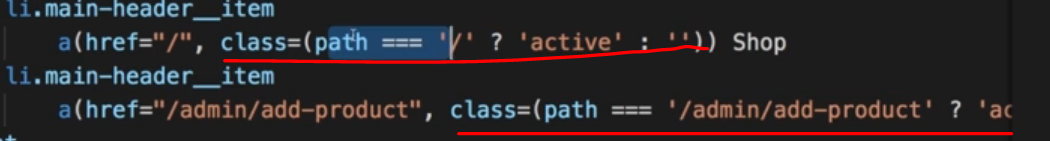
And if we add some class or id dynamically we have to use some js in the pug file

So we have to use js because what if all file have that same link but different stile and functionality for different page

So first we have to pass the path while render so that pug file can use it in run time



An after that for each different files we defines the path as different so that it can compare and check decide to add the class or not with adding class it also takes the css of it



* **EJS ENGINE**

We can use ejs instead of pug engine and the benefit of that is it uses normal html with the js in it.

Just like pug we have to define first that our engine is ejs and used in the folder

app.set(**'view engine','ejs'**);

app.set(**'views','views'**);

so we defined our view engine as ejs so we do not have to write the ejs extention while render and views as default folder to look the file at

so we are just passing the prods object containing the array and docTitle contains text

 res.render('shop',{prods: products, **docTitle: 'shopPage'**});

the render will only look to the set things in the app and go to the views folder

<title> **<%= docTitle %>** </title>

So we can add our data dynamically with the help of <%= inside this sing%>

**Difrent between <%= and <%- is that <%= takes the text formate while <%- takes includes the thing like path and <% takes javaScript code in it**

If some code is common in all then you can create a layout or copy the common code in somre different folder and used it by <%- includes(file path) %>

**<%- include('ejsLayouts/header.ejs')  %>**

Which is taken from the header.ejs

And also you can use common js code just like

**<% if (prods.length > 0) { %>**

            <div class="grid">

**<% for (let product of prods) { %>**

                    <article class="card product-item">

And close it when you have to by

               </article>

**<% } %>**

            </div>

**<% } else { %>**

            <h1>No Products Found!</h1>

**<% } %>**

So ejs having advantages over pug you can use the whole html code as it is with some java Script functionality on it.