Just like C# MVC patterns.

M = Model (ex: db access)

The part of our application that will deal with the database or any data-related functionality.

Think of the Model as a way to abstract the data into a “model” of something that can store the data. So we can create functions that do things like save, delete, update and other functions to the data without messy manipulation.

V = View (pug)

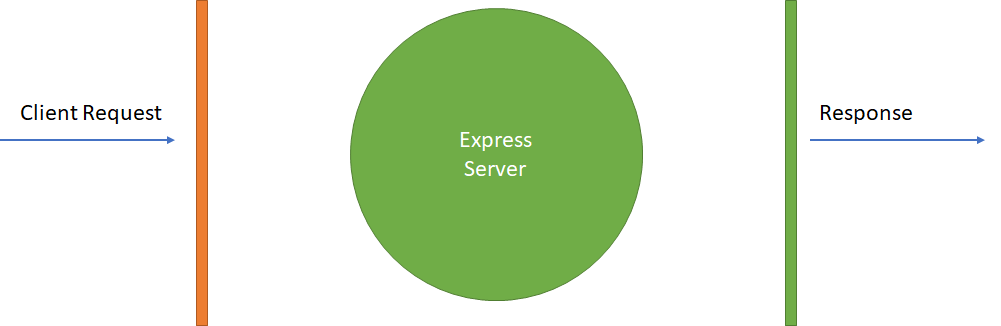
Everything the user will see. Basically, the pages that we’re going to send to the client.

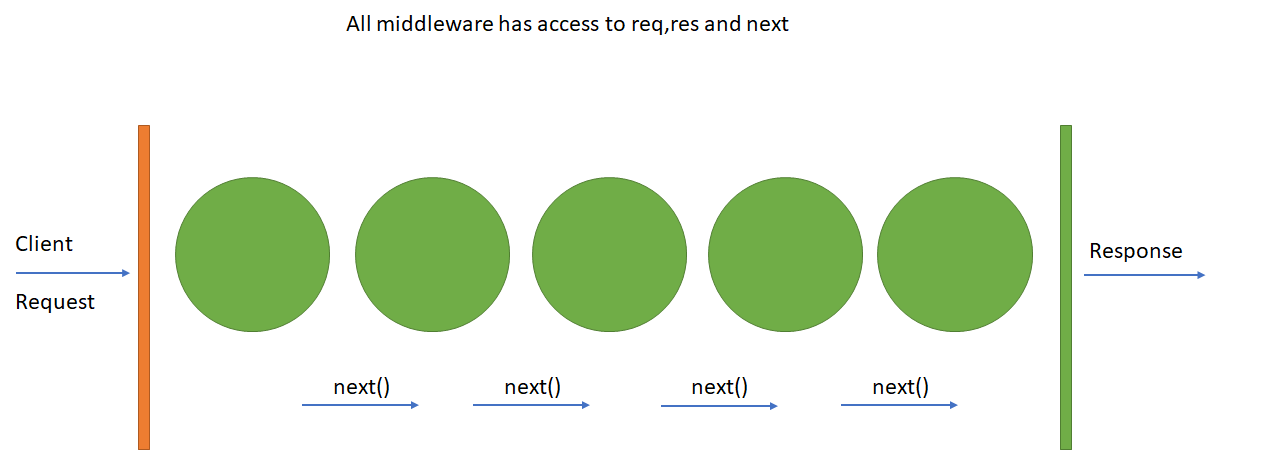
C = Controller (ex: controls flow of data )

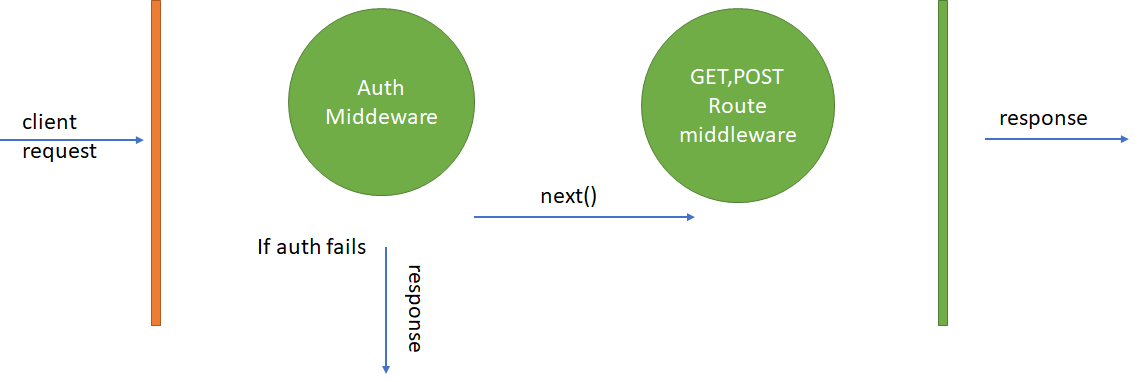
The logic of our site, and the glue between models and views. Call the models to get the data, then put that data on the views to be sent to the users.

Middleware - Middleware has access to request and response object

* Execute any code.
* Make changes to the request and the response objects.
* End the request-response cycle.
* Call the next middleware in the stack.







Then you can add layers to the middleware stack by calling .use as in app.use

// express

app.use(express.static(..));

// middleware

stack.use(function(data, next) {

next();

});

App.js mostly routes and app.use for middleware access

Examples of use

Cookies

Parsers

Authentication

Image serving

Routes

Session handling

Error handling

Local variable helpers

Passing any parameter to Next is assumed to be an error you are passing

Node has a callback type feature called a Promise. A callback is something you often see waiting for events to happen, task to complete, messages to be delivered and can be used for mundane task like updating the UI thread from the underlying thread.

Example :

var promise = doSomethingAsync()

promise.then(onFulfilled, onRejected)

A promise is just an enhanced callback. When a promise is returned, it can have 2 outputs. This is defined by the 'then clause'.

promise.then(onFulfilled, onRejected)

Either the operation can be a success which is denoted by the 'onFulfilled' parameter. Or it can have an error which is denoted by the 'onRejected' parameter.

promise.then(onFulfilled, onRejected)

\*\*\* Note \*\*\*

So the key aspect of a promise is the return value. There is no concept of a return value when working with normal callbacks in Node.js. Because of the return value, we have more control of how the callback function can be defined.

Get access to promise with a module called promise.

**npm install promise**

When defining promises, it needs to be noted that the "then" method itself returns a promise. So in a sense, promises can be nested or chained to each other.

Slugs are part of Markdown framework which is another class already posted on your share.

A slug is a URL markdown friendly way to show a title in the URL or a string. Here is a JAVAScript function to create a slug from a string.

function convertToSlug(Text)

{

return Text

.toLowerCase()

.replace(/[^\w ]+/g,'')

.replace(/ +/g,'-')

;

}

Wes Recommend doing data normalization close to the model. I agree with Wes on this point completely. It is where you go for data change and it should be where you go for normalization.