NODE Notes 3-07-2020

The best way to store forms is as a MIXIN. This can be added to a page without retyping the code over and over.

Wes Bos uses \_ as the name of a Mixin. Which is not required but as he says, it is a good standard.

Think of Mixin as a way that you can bypass the fact that in JavaScript and hence NODE you can only inherit from a single object. The mix of two objects, objecta and object is what you may consider a mixin. In simple terms anyway. “A [mixin](https://en.wikipedia.org/wiki/Mixin) is a class containing methods that can be used by other classes without a need to inherit from it.”

I will post an event MIXIN.js as part of the homework for Video 16 after I return. This is a way to create an event handler you can subscribe to.

A mixin for PUG is a way to create reusable blocks of PUG code. This makes editing reused logic centralized.

#{ … } is interpolation. What is interpolation? Interpolation is simply a trick that lets you directly inject variables into a string. In JAVAScript and NODE it looks like #{ }

We use MONGOOSE for accessing Mongo DB

We need to import our SCHEMA to use it in the DB. We can get that from the Mongoose variable. This is a result of the singleton reference of Mongoose allowing it to be imported only once.

Using a strict schema means the only fields we will see is the ones we define in our schema.

You have to call the controller.save to save data to the DB. Until you do this all the DB work you are doing is temporary and not actually in the DB but is instead properties of the schema object.

By setting mongoose promise to a global promise you can use promise throughout the use of mongoose without redeclaring Promise.

Async Await : An await function can only be called inside a async function. A regular function cannot call await.

Because of the previous statement. There is no way to call await inside top-level code.

Await respects “then” logic like a promise

A flash is a message attached to the request. The flash is a middleware that allows us to flash messages to the user. This is obtained like this. app.use(flash());

These can be available after installing flash module

npm install express-session --save

npm install cookie-parser --save

npm install connect-flash –save

You can put this directly in your ES6 code and import to UI like this.

<% if(success\_msg) { %>

<div class="alert alert-success">

<%= success\_msg %>

</div>

<% } %>

<% if(error\_msg) { %>

<div class="alert alert-danger">

<%= error\_msg %>

</div>

<% } %>