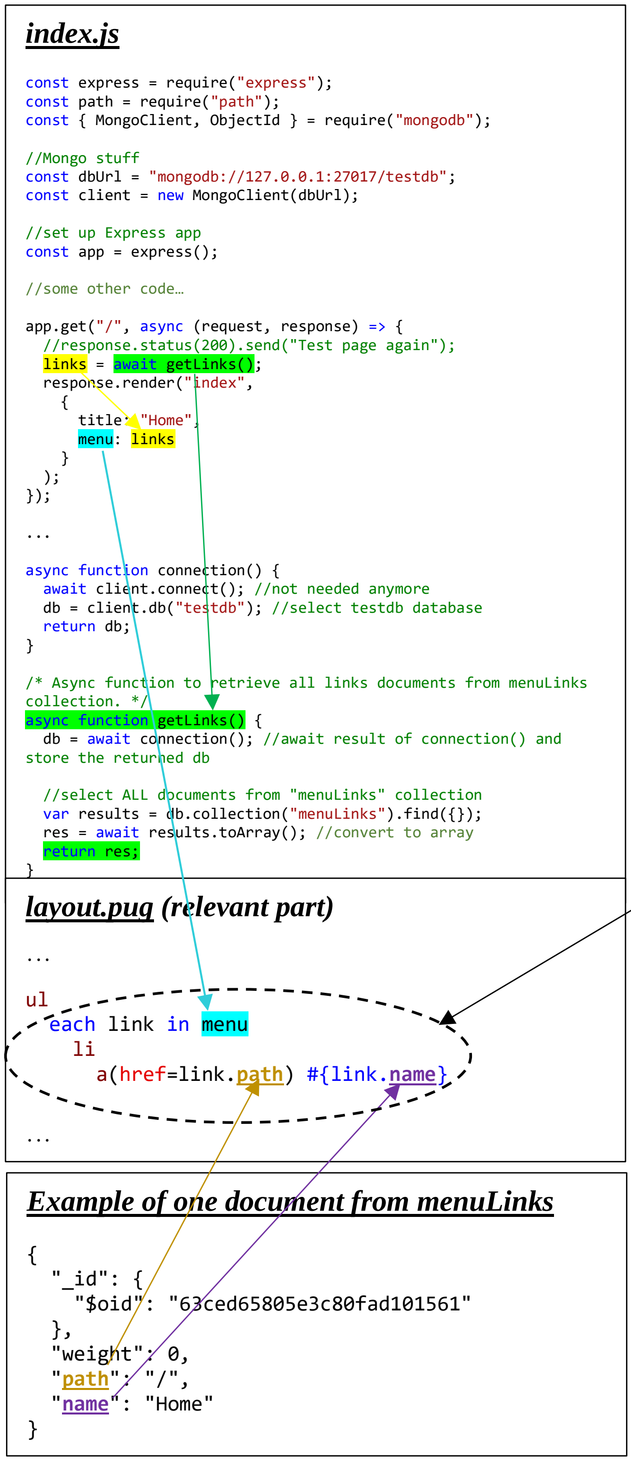
MongoDB and your Node app template code

When pulling data from MongoDB, what you're mainly doing is selecting content and displaying in a template. To get a better idea as to the relationships for what's actually happening, take a look at our code from week 3.



index.js

```
const express = require("express");
const path = require("path");
const { MongoClient, ObjectId } = require("mongodb");

//Mongo stuff
const dbUrl = "mongodb://127.0.0.1:27017/testdb";
const client = new MongoClient(dbUrl);

//set up Express app
const app = express();

//some other code…

app.get("/", async (request, response) => {
  //response.status(200).send("Test page again");
  links = await getLinks();
  response.render("index",
    {
      title: "Home",
      menu: links
    }
  );
});

...

async function connection() {
  await client.connect(); //not needed anymore
  db = client.db("testdb"); //select testdb database
  return db;
}

/* Async function to retrieve all links documents from menuLinks
collection. */
async function getLinks() {
  db = await connection(); //await result of connection() and
store the returned db

  //select ALL documents from "menuLinks" collection
  var results = db.collection("menuLinks").find({});
  res = await results.toArray(); //convert to array
  return res;
}
```

***layout.pug* (relevant part)**

```
...

ul
  each link in menu
    li
      a(href=link.path) #{link.name}

...
```

Example of one document from menuLinks

```
{
  "_id": {
    "$oid": "63ced65805e3c80fad101561"
  },
  "weight": 0,
  "path": "/",
  "name": "Home"
}
```

The code to the left is a snippet of from *index.js* with the relevant parts to demo reading data from MongoDB. I have expanded the `render()` method so that you can see the second parameter (JSON part) clearly.

Anything you want pass to the template file goes in this second parameter. In this case, the names **title** and **menu** are just variable names I've come up with. These are the variables I'll use in the template file. (Arrows are colour-coded according to what they correspond to.)

In *index.js* on the left, I have retrieved all documents from the *menuLinks* collection (`db.collection("menuLinks").find({})`) and passed the data (returned from `getLinks()`) to the render function. The **menu** variable's value is set to the value of returned array of MongoDB documents from the *menuLinks* collection.

Now, in *layout.pug* (which is using the **menu** variable), we're looping over the **menu** using Pug's foreach whose syntax is:

```
each <temp_var_name> in <array>
  -//whatever you're printing in the loop
```

Now, in terms of the property names being used in the loop in *layout.pug*, this depends on your MongoDB documents' fields you have available. These field names are not mandatory. These are field names you have come up with. (This is like when you come up with your own column names in a SQL database.) In this case, because the collection is to store data for menu links, I have included data for a ***path*** (URI), ***name*** (link text) and ***weight*** (for link display order, in case you want to implement this).

If you have different field names (and you should if you have different data), you would use whatever your field names are in your templates.

Testing queries (extra)

If you wanted to query based on a condition rather than select all (`find({})`), you can use a filter rather than empty curly brackets.

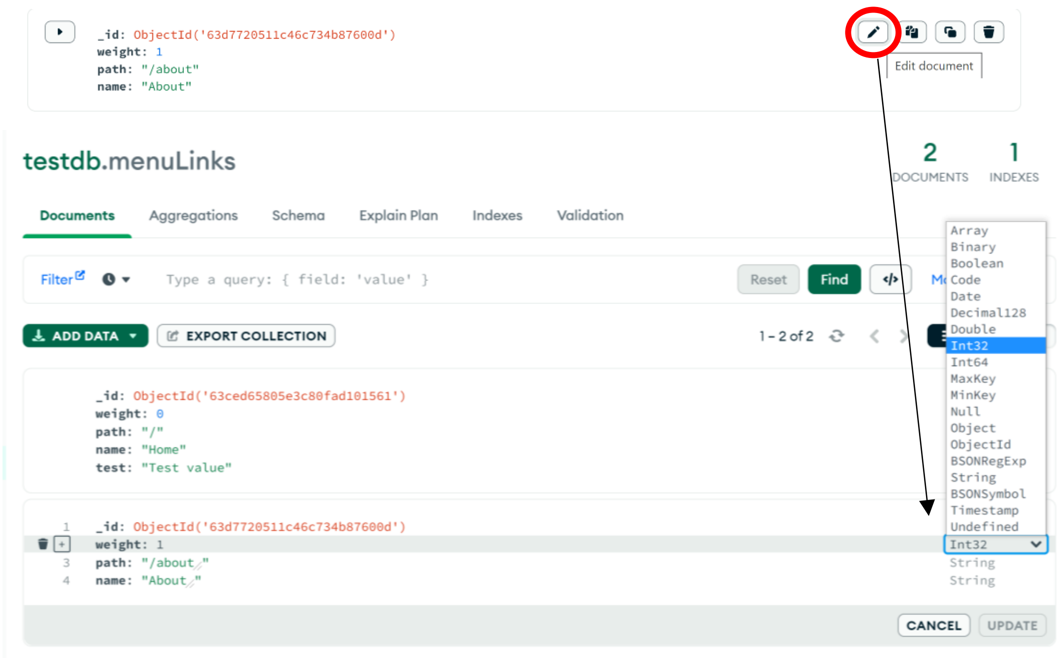
For example, if the documents in an *inventory* collection have a **category** field (e.g **category: "shirt"** is one of your documents' field value), you can find() all documents from the *inventory* collection whose **category** equals "shirt" by using { category: "shirt" } as the filter. In other words,

```
db.collection("inventory").find({ category: "shirt" })
```

The following resource has instructions for testing queries using Compass, but it's still relevant (looking at the filters). The nice thing is: you can test out filters in Compass by typing out in the **Filter** text field and click on the **Find** button to execute.

https://www.mongodb.com/docs/compass/current/query/filter/

Note: For numeric fields, make sure you enter the values without quotes if you want it to be saved as a number. This is important if you want to use queries with greater than ($gt) or less than ($lt). You can easily change data types for a field in Compass. Click the edit button for a document (hover over a document and click the pencil icon), then click on the dropdown arrow for the datatype of the field you're editing (the arrow will appear when you hover over the datatype name). After you select the datatype you're converting to, click the **UPDATE** button to save your changes.



```
_id: ObjectId('63d7720511c46c734b87600d')
weight: 1
path: "/about"
name: "About"
```

```
_id: ObjectId('63ced65805e3c80fad101561')
weight: 0
path: "/"
name: "Home"
test: "Test value"
```

```
_id: ObjectId('63d7720511c46c734b87600d')
weight: 1                                    Int32
path: "/about"                               String
name: "About"                                String
```