Day 01 Fetch and Promises

https://youtu.be/-oZd9zw0vlo

Day 02 Node and Express API

<https://youtu.be/5sPXMu5t5tE>

Day 03 Mongo CLI

<https://youtu.be/_8oCXYMDimQ>

Day 04 Mongoose

<https://youtu.be/TmXl-5UzjcI>

Day 05 React

<https://youtu.be/lnqEiADeDIQ>

Day 06 React State

<https://youtu.be/lvU2euoQkfc>

Day 07 React Lifecycle Methods

<https://youtu.be/VoeeHfDDU_8>

Day 08 React Router

https://youtu.be/TXspU8o631k

Video 1

A promise is a value that will exist. But right now, it does not. We do not know when we will be receiving the response. For example, we do not know when the server is okay and send us a response, we will action them when the promise is return

Vanilla JS/ Browser JS

Video 2

Node js & Express

Nodejs is a server side language. What we execute on the server side

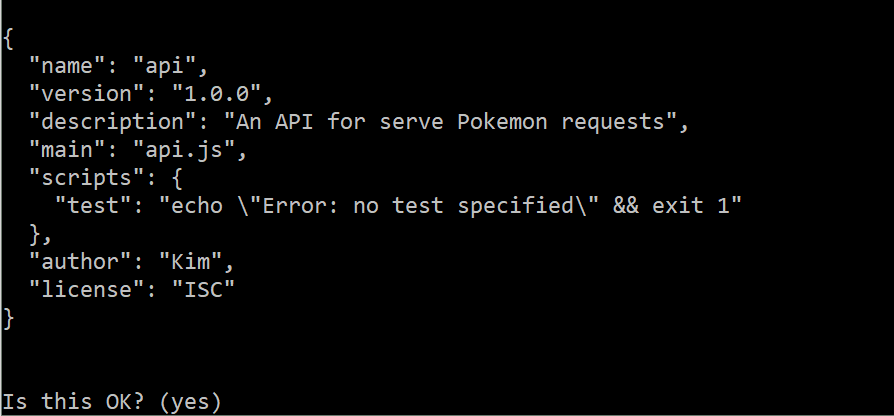
Vanilla js

Create an api folder

>cd api

>npm init

Node package manager. A tool to download packages on the internet. To interact with the repositories



A package.json file is created

>npm install express –save

It created the node module folder for us

Create api.js

<https://www.npmjs.com/package/nodemon> it allows us do not have to re run everything

>npm install nodemon --save-dev

>npm start

>npm i joi

In the api.js file,

const express = require ('express');

*//it is going to look for the package in the node module folder*

const app = new express ();

*// express gives us a function, it is going to return us a instance of express*

const port = process.env.PORT || 5000;

*//for deployment - heroku may have a different environment variable other than 5000*

const Joi = require('joi');

const pokemon = [

{

id: 1,

name: 'bulbasaur'

},

{

id: 2,

name: 'ivysaur'

},

{

id: 3,

name: 'venusaur'

},

{

id: 4,

name: 'charmander'

}

]

app.use(express.json()); *// middleware - without this statement we have no access to request body*

app.get('/', (*req*,*res*)=>{

return res.send('hello world! from API')

})

*//call back function, what people are going to do when people send a get request to the root path*

app.get('/pokemon/', (*req*,*res*)=>{

return res.send(pokemon)

})

app.get('/pokemon/:id/', (*req*,*res*)=>{

*//:id = query parameters*

*// express allow us to grab the parameter*

*// req.params gives us access to the value /pokemon/:id parameters*

*//this key (id) give us access to the value*

*// const id = req.params.id;*

const { id } = req.params;

*//object destructuring, which object we want to pull out*

const poke = pokemon.find(*p* => p.id === parseInt(id));

if(!poke){

*//200 success*

*//300 redirect*

*//400 user error*

*//500 server error*

return res.status(404).send('Pokemon not found!');

}

return res.send(poke);

*// find method : it takes a callback. passing in a function to do something over a*

*// collection of item of once it is ready*

});

app.post('/pokemon', (*req*,*res*)=>{

*//grab the values from the body*

const { id, name} = req.body

*//create new Pokemon Object*

const poke = { id, name };

*//validate we have ID and names*

const schema = {

id: Joi.number().required(),

name: Joi.string().min(3).required()

}

const valid = Joi.validate(poke, schema);

console.log(valid)

if (valid.error) {

const message = valid.error.details[0].message

return res.status(400).send(message);

}

*//insert new Pokemon into an Array*

pokemon.push(poke);

*//send back Pokemon that has been added*

return res.send(poke);

})

*//update pokemon data (don't forget to put (:id)so it know where to put)*

app.put('/pokemon/:id', (*req*, *res*) => {

*//3. update the record, and send back*

*//grab the values from the body*

const id = req.params.id;

const name = req.body.name;

*//looking for the poke id that matches with the params id in the array*

const changeName = pokemon.find(*poke* => poke.id == id)

*//declaring a variable that allow us to look for the index of the object*

const position = pokemon.indexOf(changeName);

*//chaning the value of the object*

pokemon[position].name = name;

console.log(req.body)

res.send(pokemon[position].name = name)

})

app.delete('/pokemon/:id', (*req*, *res*) => {

*//if the poke does not exist*

if (!poke) {

return res.status(404).send('Pokemon not found');

}

*//look for the id*

const id = req.params.id;

*// grab the object from the array*

const object = pokemon.find(*poke*=> poke.id == id);

*//index of the object*

const objectIndex = pokemon.indexOf(object);

*//delete the object from the array*

pokemon.splice(objectIndex, 1)

res.send("{it has been deleted successfully}")

})

app.listen(port, ()=>{

console.log(`listening at http://localhost:${port}`)

})

*//tell our app to run*

Video 3

>sudo mkdir -p /data/db

>sudo chown -R `id -un` /data/db

>mongod

Open one more terminal

>mongo

>show dbs

>use pokedex

>db.createCollection(‘pokemon’)

> show collections

>db.pokemon.insert({

... id: 1,

... name: 'bulbasaur',

... height: '6cm'

... })

> db.pokemon.find()

> db.pokemon.insert([{ id: 2, name: 'ivysaur', height: '8cm' }, { id: 3, name: 'newsaur', height: '9cm'}])

**Looking for one specific pokemon**

> db.pokemon.find({name: 'ivysaur'})

**Update a pokemon (the whole object)**

db.pokemon.update({ name: 'bulbasaur'}, { name: 'bulbesaur'})

**Update the attribute**

db.pokemon.update({ name: 'newsaur'}, {$set: { name: 'Charmander'}})

**Remove an object**

db.pokemon.remove({ name: ‘Charmander’})

**Remove all the pokemon**

db.pokemon.remove()

**Add an object if it does not exist**

db.pokemon.update({ name: 'charmeleon'}, { id, 5, name: 'charmeleon', height: '7m'}, {upsert : true} )

**Add an array in the object**

db.pokemon.insert({ name: 'test', moves: [“tackle”, “play”]})

db.pokemon.findOne({name: ‘test’}).moves

**Create a new collections of moves**

**Convert into an Array**

db.moves.find().toArray()

Video 4

Delete our in memory representation

>npm i mongoose

Api.js

const mongoose = require ('mongoose');

mongoose.connect('mongo://localhost:27017/pokedex')

Create a model

const Pokemon = require('./models/Pokemon.js')

create a pokemon model and folder

const mongoose = require('mongoose');

const pokemonSchema = new mongoose.Schema({

id: Number,

name: String

});