Node JS Notes

* NodeJS allows JavaScript to be run on the server side and allow to be ran in the node environment instead of being run on the browser
* PHP was replaced by NodeJS and now Node can put together the server for the backend
* Being Asynchronous threading node will only need one thread to work on multiple requests fulfilling requests faster by finishing the fastest one even while fulfilling a longer request
* Difference in console logging .this in the global space will capture nothing in node js which is the difference in browser and node is that in browser it’ll capture the window
* Process.argv
* **Module.export = {key:value, key2:value2, key3:value3, ….}**
  + Is basically an object that holds key value pairs
  + Which will be exported
* Var example = **require(“pathway/to/object”) 🡪example.keyName** to call object
  + This will import whatever object you wanted to export through JavaScript
  + Can import files as well as packages
* **Node fs** can let you write on something through JavaScript
* NPM
  + Node packaging management
  + Has a bunch of packages which are basically APIs
* Appending Files for node
  + .appendFile(“sample.txt”, “\n”+ text, function(err))
  + If sample.txt did not exist .appendFile() will create it
* AXIOS
  + Used to hit APIs
  + This is a package that you install for your web app basically an API
* For **npm init -y**
  + Will create the JSON dependency package
* Modulization
  + each file is treated as a separate module. For example, consider a file named foo.js:
  + To include a module, use the require() function with the name of the module:
  + <https://www.w3schools.com/nodejs/nodejs_modules.asp>
  + Used to reuse the codes multiple times by exporting and importing certain module

var action = process.argv[2]

if (action === 'total') {

total()

}else

if (action === 'widthdraw'){

}else if (action === "deposit") {

}

function total(){

//read from the bank.txt and add up the number

fs.readFile("bank.txt", 'UTF8', function(err, data){

if(err){

return console.log(err)

}

console.log(data);

var dataArr = data.split(", ");

console.log(dataArr);

var sum = 0;

for (let i = 0; i < dataArr.length; i++) {

//add up the numbers in the array

//parseFloat will allow you to turn array strings into number disregarding the decimals/ commas

sum += parseFloat(dataArr[i]);

console.log ("your total balance is "+ sum.toFixed(2));

}

})

}

function widthdraw(){

var amount = process.argv[3]

fs.appendFile("bank.txt", ", -" + amount, function(err, data){

if(err){

return console.log(err)

}

console.log(d"you widthdrew $" + amount)

}

unction deposit(){

var amount = process.argv[3]

fs.appendFile("bank.txt", ", " + amount, function(err, data){

if(err){

return console.log(err)

}

console.log(d"you deposited $" + amount)

}

MAP QUEST PACKAGE EXAMPLE

var input = process.argv

// Build your address as an array or string

var address = '';

for (let i = 0; i < input.length; i++) {

address +=input[i] + " "

console.log(adress)

}

// Then use the geocoder object to search the address

geocoder.geocode(address, function(err, res){

if(err){ return console.log(err);

} console.log(res);

});

//or you can use this bottom one

var address = process.argv.slice(2).join(" ")

console.log(address)

geocoder.geocode(address, function(err, res){

if(err){ return console.log(err);

} console.log(res);

});

**Inquirer**

* Inquirer Prompt Documentation
  + Allows you to ask user questions to get answer as key value pairs

/==============soccer team=======================/

// Dependency for inquirer npm package

var inquirer = require("inquirer");

// Constructor function for creating player objects

function Player(name, position, offense, defense) {

this.name = name;

this.position = position;

this.offense = offense;

this.defense = defense;

// Flips a coin: if the the value is equal to 0 then this.offense goes up by one

// If the value is equal to 1 this.defense goes up by one

this.goodGame = function() {

if (Math.floor(Math.random() \* 2) === 0) {

this.offense++;

console.log(this.name + "'s offense has gone up!\n----------\n");

} else {

this.defense++;

console.log(this.name + "'s defense has gone up!\n----------\n");

}

};

this.badGame = function() {

if (Math.floor(Math.random() \* 2) === 0) {

this.offense--;

console.log(this.name + "'s offense has gone down!\n----------\n");

} else {

this.defense--;

console.log(this.name + "'s defense has gone down!\n----------\n");

}

};

this.printStats = function() {

console.log(

"Name: " +

this.name +

"\nPosition: " +

this.position +

"\nOffense: " +

this.offense +

"\nDefense: " +

this.defense +

"\n----------\n"

);

};

}

// Arrays used to contain all of our player objects

var starters = [];

var subs = [];

var score = 0;

var roundNumber = 0;

var maxRounds = 5; // determines the maximum number of rounds we play.

// calls the function initPlayers() to start the code

/\*\*

\* Logic:

\* initPlayers() creates players. It creates 3 players, then starts the game with playGame()

\* playGame() checks the roundNumber; if less than maxRounds, runs playRound(); otherwise ends the game with endGame(); playRound() will call playGame() after each round.

\*/

initPlayers();

// Function to run the game itself. the variable x is used here to keep track of the rounds

function playGame() {

console.log("\n--------\nplayGame()\n---------");

if (roundNumber < maxRounds) {

playRound();

} else {

endGame();

}

}

// Function which will allow the user to create 2 starting players and 1 substitution player

// Then will print each players stats afterwards

function initPlayers() {

console.log("\n--------\ninitPlayers()\n---------");

// if the length of the team array is 3 or higher, no more questions will be asked

if (starters.length + subs.length < 3) {

promptPlayerCreation();

} else {

// Starts first round

playGame();

}

}

function promptPlayerCreation() {

console.log("\n--------\npromptPlayerCreation: NEW PLAYER!\n--------\n");

inquirer

.prompt([

{

name: "name",

message: "Player's Name: "

},

{

name: "position",

message: "Player's position: "

},

{

name: "offense",

message: "Player's Offensive Ability: ",

validate: function(value) {

if (isNaN(value) === false && parseInt(value) > 0 && parseInt(value) <= 10) {

return true;

}

return false;

}

},

{

name: "defense",

message: "Player's Defensive Ability: ",

validate: function(value) {

if (isNaN(value) === false && parseInt(value) > 0 && parseInt(value) <= 10) {

return true;

}

return false;

}

}

])

.then(function(answers) {

// Runs the constructor and places the new player object into the variable `player`

// Turns the offense and defense variables into integers as well with parseInt

var player = new Player(

answers.name,

answers.position,

parseInt(answers.offense),

parseInt(answers.defense)

);

// Adds a player to the starters array if there are fewer than 2 player

// Objects in it. otherwise adds the newest player object to the subs array

if (starters.length < 2) {

starters.push(player);

console.log(player.name + " added to the starters");

} else {

subs.push(player);

console.log(player.name + " added to the subs");

}

// Runs the `initPlayers` function once more

initPlayers();

});

}

function playRound() {

// Adds one to `roundNumber` and prints the current round of the game

roundNumber++;

console.log("\n----------\nplayRound: ROUND " + roundNumber + "\n----------\n");

// Finds two random numbers between 1 and 20 to compare the starter objects' stats to

var offenseRandom = Math.floor(Math.random() \* 20) + 1;

var defenseRandom = Math.floor(Math.random() \* 20) + 1;

// Loops through the starter array to find if the total value of their offense and defense

var teamOffense = 0;

var teamDefense = 0;

for (var i = 0; i < starters.length; i++) {

teamOffense += starters[i].offense;

teamDefense += starters[i].defense;

}

console.log("Team Offense: " + teamOffense);

console.log("Team defense: " + teamDefense);

console.log("Random O: " + offenseRandom);

console.log("Random D: " + defenseRandom);

// Determines if `teamOffense` is less than `defenseRandom` and adds one to score if true

if (teamOffense > defenseRandom) {

console.log("YOU SCORED A POINT!");

score++;

}

// Determines if `teamDefense` is greater than `offenseRandom` and subtracts one from score if true

if (teamDefense < offenseRandom) {

console.log("YOU WERE SCORED UPON!");

score--;

}

substitutePlayer();

}

// The substitutePlayer function prompts the user to see if they would like to make a substitution

function substitutePlayer() {

inquirer

.prompt([

{

name: "confirm",

type: "confirm",

message: "Would you like to make a substitution?"

}

])

.then(function(answer) {

// If the answer is yes, start the substitution prompts

if (answer.confirm === true) {

// Sets the names of all those contained within the subs array as choices

inquirer

.prompt([

{

name: "sub",

type: "rawlist",

message: "Who would you like to sub in?",

choices: subs

}

])

.then(function(subIn) {

// Finds the player object within the subs array with the name that matches the user's choice

// Places it within the sideline variable

var sideline = {};

var number = 0;

for (var i = 0; i < subs.length; i++) {

if (subs[i].name === subIn.sub) {

number = i;

sideline = subs[i];

}

}

inquirer

.prompt([

{

name: "sub",

type: "rawlist",

message: "Who would you like to sub out?",

choices: starters

}

])

.then(function(subOut) {

// Finds the player object within the starters array with the name that matches the user's choice

// Swaps it with the value contained within sideline after moving them into the subs array

for (var i = 0; i < starters.length; i++) {

if (starters[i].name === subOut.sub) {

subs[number] = starters[i];

starters[i] = sideline;

console.log("SUBSTITUTION MADE!");

}

}

// Starts the next round (playGame() checks the round number and calls playRound() if we're still below maxRounds)

playGame();

});

});

} else {

// Starts the next round (playGame() checks the round number and calls playRound() if we're still below maxRounds)

playGame();

}

});

}

function endGame() {

// Prints the final score

console.log("\n--------endGame(): FINAL SCORE: " + score + " --------\n");

// If the score was greater than 0, prints the winning message and increases starters stats

if (score > 0) {

console.log("Good game, everyone!\nYour current starters' stats have improved!");

for (var i = 0; i < starters.length; i++) {

starters[i].goodGame();

}

}

// If the score was less than 0, prints the losing message and decreases starters stats

if (score < 0) {

console.log("That was a poor performance!\nYour current starters' stats have decreased!");

for (var i = 0; i < starters.length; i++) {

starters[i].badGame();

}

}

// If the score was zero, prints the tie message and does nothing to the starters stats

if (score === 0) {

console.log("It was a tie game! Not good. Not bad.");

}

// Prompts the user if they would like to play again. if yes, run playgame with a value of 0 being passed into it

// Otherwise print the "come back again soon message" and exit

inquirer

.prompt({

name: "again",

type: "confirm",

message: "Would you like to play another match?"

})

.then(function(answer) {

if (answer.again === true) {

// Starts new match with the same players

roundNumber = 0;

playGame();

} else {

console.log("Come back again soon!");

}

});

}