**Exercise03\_02\_01 – Step 1**

In this Exercise, we will transition to ES6 in Node.js.



1. Create a folder named ***Exercise03\_02\_01***. Copy in all of the contents from ***Exercise03\_01\_01***. Open your IDE to the project folder.
2. Go to the ***/src/lib*** folder and open up the file ***fluxGen.js***. This file exports a helper function that generates random quotes based on some variables. We will change some of the variables declared at the top to properly use the scope rules of ES6. The first one is ***output***, which is assigned an empty array. It should be a ***const***, because it is not used outside its ***block***. It is an ***object*** and it is never reassigned. A ***const*** will let us modify its ***properties***, which are its elements:  
    ***const output = [];***
3. The ***current*** variable should be declared with ***let***. It is never used outside its arrow function scope, and its value will be assigned multiple times:  
    const output = []***;  
    let current = seed;***
4. We should delete the ***change*** variable. Declare it with a ***let*** inside the code block of the ***for*** loop. That is the only place that it is used:  
    for (var i = 0; i < times; i++) {  
    ***let change*** = (Math.random() \* variability).toFixed(0);
5. Finally, with ***block*** scope, the parameters of a ***for*** loop are considered to be scoped within its code block. We can change the loop variable to be declared with a ***let***, which will protect it from being used outside the loop:  
    for (***let*** i = 0; i < times; i++) {  
   Run the server with ***npm*** ***start*** and the browser to check syntax and logic. Open the market, highlight some different pizzas, and close the market to make sure nothing is broken.

**Exercise03\_02\_01 – Step 2**



1. Go to the ***/src/models*** folder and open up the file ***pizza.js***. Notice that the object ***constructor*** has numerous assignments from individual array elements. We can clean this up with destructuring as follows:  
    this.startingDate = startingDate; ***[  
    this.ticker,  
    this.name,  
    this.startingQuote  
    ] = pizzaProps;***Run the server with ***npm*** ***start*** and the browser to check syntax and logic. Open the market, highlight some different pizzas, and close the market to make sure nothing is broken.
2. There are still two more properties that are assigned from the array using a subscript. If we move them into the square brackets, we can use ***default*** parameter syntax. We can get rid of the logical ***OR*** statements to handle missing values. Destructuring will let us use ***default*** ***value*** syntaxon our assignments:  
    this.startingQuote***,  
    this.variability = getRand(),  
    this.positivity = getRand()***  
    ] = pizzaProps;  
   Run the server with ***npm*** ***start*** and the browser to check syntax and logic. Open the market, highlight some different pizzas, and close the market to make sure nothing is broken.

**Exercise03\_02\_01 – Step 3**



1. Go to the ***/src/handlers*** folder and open up the file ***pizza.js***. The ***req.params*** is an object; it is a good place to clean up code. We will first try it with just a single object destructure, and we can see how much simpler and more readable it is:  
    ***const { ticker } = request.params;***Run the server with ***npm*** ***start*** and the browser to check syntax and logic. Open the market, highlight some different pizzas, and close the market to make sure nothing is broken.
2. Go back to the ***/src/models*** folder and open up the file ***pizza.js***. Go down to the ***getDatedQuotes()*** function and look at the ***curDate*** assignment. Because the ***this*** keyword always refers to an ***object***, we can make elegant use of destructuring to assign to variable from it. We can do this because object destructuring always uses the name of ***name/value*** pairs:  
    ***{ startingDate: curDate } = this;***  
   Run the server with ***npm*** ***start*** and the browser to check syntax and logic. Open the market, highlight some different pizzas, and close the market to make sure nothing is broken.

**Exercise03\_02\_01 – Step 4**



1. Go to the ***/src/lib*** folder and open up the file ***popGen.js***. At the bottom of the file is the ***module.exports*** declaration, which is a JSON object. We can employ ES6 shortened ***property*** ***declarations*** here:  
   module.exports = {  
    ***getPopularSlices,  
    getMostPopular,  
    getNewestSlice,  
    getMostImproved***};  
   Run the server with ***npm*** ***start*** and the browser to check syntax and logic. Open the market, highlight some different pizzas, and close the market to make sure nothing is broken.
2. Now let’s open up the file ***dataStore.js***. Go down to the ***module.exports*** declaration, and let’s make the same ES6 changes as we did in the previous file:  
   module.exports = {  
    ***init,  
    getQuotes,  
    getAllQuotes,  
    getPizzas,  
    getPizza***};  
   Run the server with ***npm*** ***start*** and the browser to check syntax and logic. Open the market, highlight some different pizzas, and close the market to make sure nothing is broken.

**Exercise03\_02\_01 – Step 5**



1. Go to the ***/src/lib*** folder and open up the file ***market.js***. In the ***module.exports*** declaration, let’s employ ES6 shortened ***function property*** ***declarations***:  
   module.exports = {  
    ***run (socket) {*** runInterval = setInterval(function () {  
    quoteManager.updateQuotes(function (err, newData) {  
    socket.emit('new\_data', JSON.stringify(newData));  
    });  
    }, 1000);  
    },  
    ***stop () {*** clearInterval(runInterval);  
    }  
   };
2. Now let’s open up the file ***quoteManager.js***. Go down to the ***module.exports*** declaration, and let’s make the same ES6 changes as we did in the previous file to the one function property:  
   module.exports = {  
    ***updateQuotes (callback) {***Run the server with ***npm*** ***start*** and the browser to check syntax and logic. Open the market, highlight some different pizzas, and close the market to make sure nothing is broken.

**Exercise03\_02\_01 – Step 6**



1. Let’s open up the main ***server.js*** file. Go down to the bottom and find the ***server.start()*** method. We can convert the console.log() parameter into an ES6 ***template*** ***string***. Remove the quotes and replace them with ***back-ticks***. Then we can use dollar sign curly bracket notation to ***interpolate*** our variable:  
    server.start(err => {  
    if (err) throw err;  
    ***console.log(`Connected on ${server.info.uri}`);*** });
2. Go to the ***/src/lib*** folder and open up the file ***quoteManager.js***. For our ***console.log()*** output, we can use both string ***interpolation*** and ***multi-line*** capabilities:  
    ***console.log(`${newData}  
    updating quotes`);***  
   Run the server with ***npm*** ***start*** and the browser to check syntax and logic. Open the market, highlight some different pizzas, and close the market to make sure nothing is broken. We do not get what we expect in our node console.
3. The ***newData*** variable is an ***object***. In string interpolation, objects do not get auto-converted to ***JSON*** string syntax by console.log(). So we need to ***JSON.stringify()*** it:  
    **console.log(`${JSON.stringify(newData)}**Run the server with ***npm*** ***start*** and the browser to check syntax and logic. Open the market, highlight some different pizzas, and close the market to make sure nothing is broken. Notice the difference in console.log() output, it is now easier to read and debug.