Assignment 1

Instructions:

- Code will be executed using NodeJS. Prior to submitting, ensure your program works as expected by running the command: `node app.js` in a console window.

- Grading is based on implementing the required functionality and coding style, specifically: clearly organized code, appropriate variable naming, code readability, coding conventions demo'd in class, etc.

- Javascript syntax rules:

  - Variables must be declared using let/const, not var
  - Functions must be declared using arrow function syntax, not function() syntax
  - When checking equality, use strict equality (triple equals ===), **not** double equals (==)
  - Do **NOT** use higher order array functions: forEach, map, reduce, filter, closest, etc.

Submission Checklist:

1. Create NodeJS project

- ❑ Create a folder called Ex4FirstName. Replace **FirstName** with your name, example: Ex4David
- ❑ Inside the folder, create a new NodeJS project. (`npm init -y`)
- ❑ Within your project's **package.json** file, update the **author** field with your name, and the **description** with a description of your project.
- ❑ Within the project, create a Javascript file called `app.js`**.** Put your solution code in this file.

2. When you are ready to submit:

- ❑ Create a zip file containing your project folder.
- ❑ Rename your zip file Ex4FirstName.zip. Replace FirstName with your name, example: Ex4David.zip.
  - ❑ Ensure you use a zip file. Rar and 7zip files are **not** accepted.

Academic Integrity

- You are responsible for familiarizing yourself with the college's Academic Integrity Policy.

- This is an individual assessment

- Situations which often cause academic integrity issues:

  - Reposting any part of the assessment to online forums or homework help websites
  - Contract plagiarism: Purchasing a solution, or completing a solution for financial compensation
  - Sharing or receiving source code, references, or assistance from others

Problem Description:

Using NodeJS, create a standalone console-based Javascript program that lets the user reserve a car rental:

1. The program specifies what the user needs for the vehicle type, number of days of the rental, and whether a child seat is need
2. Based on this information, the program should search for an available vehicle.
3. If a vehicle is available, then calculate and output a receipt for the rental.
4. If no matching vehicle is found, output an error message.

Specifically, the program should be written as follows:

1. Create a function called createReservation(*carType, days, seatNeeded*)

The function accepts a car type (string), the number of days (number), and whether a car seat is required (boolean) as parameters. Using these values, the function should:

- Search a list of cars for a matching vehicle
- If no matching vehicle is found, output an error message to the console
- Otherwise, calculate the cost of the car rental and output a receipt. The receipt must contain a reservation id (4 digit random number), matching car type, license plate, reservation subtotal, tax amount, and final total to pay.

List of Cars

This list of cars should model the cars owned by the car rental agency.

- The list must contain **3 cars**.
- Each car in your list must be represented as an **object literal**.
- Each car in the list must have a **type** (string)**, license plate** (string), and **availability** (boolean). You should use different data for each car.

Here is an example of car data. You should customize this data to your own values.

Type	**License Plate**	**Availability**
SUV	ABC 124	false
Sedan	BXL 009	true
SUV	KML 155	true

Cost Calculation:

The formula for calculating total cost is: (**rental fee + car seat fee) + sales tax**

- ***Rental fee***: The rental fee is (**# of days requested x daily rate)**.
  - Sedans are $10 per day, SUVS are $15 per day, and all other vehicle types are $20 per day.
- ***Car seat fee***: If the user requests a car seat, then the fee is **$3 per day**.
- ***Sales tax*** is 13%.

2. After defining this function, write code that performs the following operations.

a. Display a welcome message to the user
b. Create variables that represent details about the user's rental (car type, days, car seat needed). Set the values of variables to the appropriate data of your choice.
c. Output the data in the variables to the screen
d. Using the variables, call the createReservation() function to book a car for the user.

<u>Format of JS file</u>

NOTE: The order of code in your Javascript file should be:

```
// function declarations
// program code (welcome message, variables, function call, etc)
```

<u>Sample output</u>

Example 1: Successful Reservation

```
-------------------------------
Welcome to David's Car Rentals
-------------------------------
Requested car type: SUV
Days: 3
Car seat needed?  true

-------------------
RECEIPT
-------------------
Reservation Number: 5819
Car Type: SUV
License Plate: KML 155
Subtotal: $51.00
Tax: $6.63
Total: $57.63
```

Example 2: Unsuccessful Reservation

- *This reservation fails because there are no trucks in the system*
- *Reservations may also fail if the selected vehicle type exists, but is unavailable for rent*

```
-------------------------------
Welcome to David's Car Rentals
-------------------------------
Requested car type: Truck
Days: 2
Car seat needed?  false
A matching vehicle cannot be found
```