Setup Documentation

1. Install Node.JS. I am running the app with Node.JS version 10.14.1, however the latest version should work fine.
2. Navigate to where you've installed the project solution and run the app.js file with Node.JS using the command "node app.js". Be sure to check that nothing else is running on port 3000. If anything else is already using port 3000 the project will crash.
3. Open an HTTP request simulation tool, and follow the unit tests below. I used Fiddler when testing my web api. Once you've made a request, you will also be able to view useful information about your activity via the Node.JS console.

Object Documentation

- **Store Object**
  - When completing the challenge, I wanted to create a solution with as few outside dependencies as possible so the project setup was straight forward. To avoid having an entire database, I've chosen to create a Store object that holds the list of products. The Store object contains an array called "Products" that is full of item objects.

- **Item Object**
  - Each item object has an ID, title, price, and inventory count.
  - The inventory count represents how many of that item there are in a given container (either the Store or the Cart)

- **Cart Object**
  - When a cart is created with an `/api/cart` Post request, it is also given a Products array. When an item is added to the cart, the inventory count variable of the item within the products array indicates how many of that item are in the cart.

Request Documentation

http://localhost:3000	GET	POST	DELETE
/api	Welcomes you to the api.		
/api/products	Fetches a list of all products.		
/api/products/available	Fetches a list of all available products.		
/api/products/:id	Fetches the product of the specified id.	Purchases the product of the specified id. This product's inventory count in the shop will decrease by one.	
/api/cart	Fetches a list of the items in your cart. [Requires a cart to exist].	Creates a cart.	Deletes your cart.
/api/cart/cost	Fetches the total cost of all the products currently in your cart.		
/api/cart/add/:id		Adds the item of this specified id from the shop into your cart.	
/api/cart/complete		Purchases all the items currently in your cart and removes them from the store's inventory.	

Unit Tests

- **View all products test**
  1. GET http://localhost:3000/api
  2. GET http://localhost:3000/api/products
     - ➢ Returns a json containing all products.

- **View all available products test**
  1. GET http://localhost:3000/api
  2. GET http://localhost:3000/api/products/available
     - ➢ Returns a json containing all products with an inventory greater than zero.

- **Purchase item test**
  1. GET http://localhost:3000/api
  2. GET http://localhost:3000/api/products
     - ➢ “Product 0” will have 10 in it’s inventory.
  3. POST http://localhost:3000/api/products/0
  4. GET http://localhost:3000/api/products
     - ➢ “Product 0” will now have 9 in it’s inventory.

- **Purchase items with cart test**
  1. POST http://localhost:3000/api/cart
  2. GET http://localhost:3000/api/cart
     - ➢ Your cart will be empty.
  3. POST http://localhost:3000/api/api/cart/add/2
  4. POST http://localhost:3000/api/api/cart/add/0
  5. POST http://localhost:3000/api/api/cart/add/0
  6. GET http://localhost:3000/api/cart
     - ➢ Your cart will now contain 2 “product 0” and 1 “product 2”.
  7. GET http://localhost:3000/api/cart/cost
     - ➢ Your total cost will be $9.97.
  8. POST http://localhost:3000/api/cart/complete
     - ➢ The console will show that you spent $9.97.
  9. GET http://localhost:3000/api/products
     - ➢ The store will now contain 8 “product 0” and 9 “product 2”.