

EXPRESS LAB 1: CART API - ARRAY

Task: Use Express to create an API server that provides a RESTful API for a collection of cart items. You can test your API using <https://gc-express-tester.surge.sh>.

Build Specifications:

1. Use Express to create your server.
2. Require the module that will contain the routes you have created.
3. Start your server out with a hard-coded array of cart items, each including **id**, **product**, **price**, and **quantity**.
4. Test your endpoints using Postman.
5. Also test your finished API using <https://gc-express-tester.surge.sh>. To do so, you must enable CORS support using the node **cors** package.

Endpoints:

The API should have the following endpoints.

1. **GET /cart-items**
 - a. Action: None
 - b. Response: a JSON array of all cart items
 - c. Response Code: **200** (OK)
2. **GET /cart-items/:id**
 - a. Action: None
 - b. Response: a JSON object of the item with the given ID
 - c. Response Code: **200** (OK)
 - d. However, if the item with that ID cannot be found in the array, return a string response "ID Not Found" with response code **404** (Not Found)
3. **POST /cart-items**
 - a. Action: Add a cart item to the array using the JSON body of the request. Also generate a unique ID for that item.
 - b. Response: the added cart item object as JSON.
 - c. Response Code: **201** (Created)
4. **PUT /cart-items/:id**
 - a. Action: Update the cart item in the array that has the given id. Use the JSON body of the request as the new properties.
 - b. Response: the updated cart item object as JSON.
 - c. Response Code: **200** (OK).
5. **DELETE /cart-items/:id**
 - a. Action: Remove the item from the array that has the given ID.
 - b. Response: Empty
 - c. Response Code: **204** (No Content)

Extended Challenges: Build an Angular app to call your API and display results. Then use a form to add items by calling your POST endpoint. Add clicks to call your DELETE endpoint.



Tests

1. **GET** `/cart-items` - responds with a JSON array of all cart items
2. **GET** `/cart-items` - responds with status code **200**
3. **GET** `/cart-items/:id` - responds with a JSON object of the item with the given ID
4. **GET** `/cart-items/:id` - responds with status code **200**
5. **GET** `/cart-items/:id` - responds with status code **404** when not found
6. **POST** `/cart-items` - add a cart item to the array using the JSON body of the request. Also generates a unique ID for that item.
7. **POST** `/cart-items` - responds with the added cart item object as JSON and status code **201**.
8. **PUT** `/cart-items/:id` - Updates the cart item in the array that has the given id.
9. **PUT** `/cart-items/:id` - Responds with the updated cart item as JSON and status code **200**.
10. **DELETE** `/cart-items/:id` - Removes the item from the array that has the given ID.
11. **DELETE** `/cart-items/:id` - Responds with no content and status code **204**.

