Node.js API (Requests/Responses)

Test URLs	Expected Output	Description / Hints
/welcome	Welcome to Just Do It!	- Use res.send() to display a simple page

/example	{
	"id": "01",
	"desc": "Complete Learn Node Lessons 1-10"
	}
		- Use res.json() to display an example JSON object

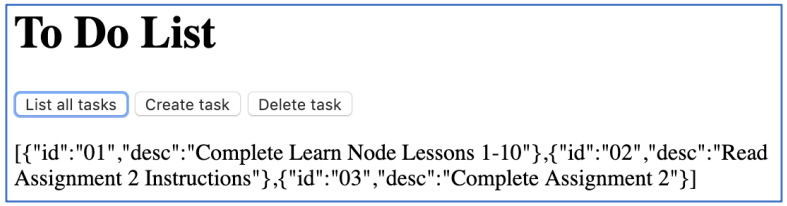
/list	[
	{
	"id":"01",
	"desc":"Complete Learn Node Lessons 1-10"
	},
	{
	"id":"02",
	"desc":"Read Assignment 2 Instructions"
	},
	{
	"id":"03",
	"desc":"Complete Assignment 2"
	}
	]
		- You can create a new JSON file by copying the contents to the left and saving it as todoList.json
		- The example to the left is an array of objects
		- An easy way to read a file is using the 'fs' module, and making use of fs.readFileSync()
		- You can convert the String into a JSON object by using JSON.parse()
		- Then use res.json()

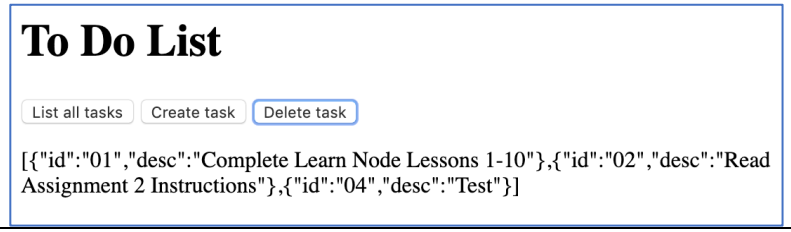
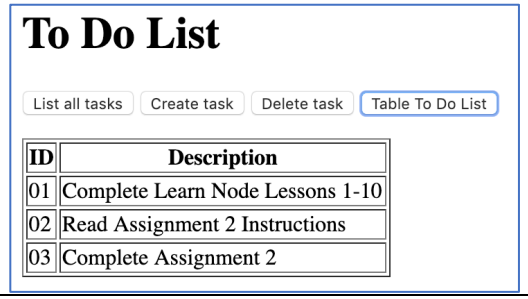
/add?id=04&desc=Test	{
	"id":"04",
	"desc":"Test"
	}
		To display the correct output:
		- Use req.query.id and req.query.desc to create a new object
		- Use res.json()
		
		To add the object to the JSON file:
		- Read the file into a JSON object (see above)
		- Use push() to append the obj to the array
		- You need to convert the OBJECT back into a string before writing to the JSON file, using JSON.stringify()
		- Use fs.writeFileSync to overwrite the file with new JSON contents

/delete/03	(The same output as /list minus the deleted item)
	[
	{
	"id":"01",
	"desc":"Complete Learn Node Lessons 1-10"
	},
	{
	"id":"02",
	"desc":"Read Assignment 2 Instructions"
	}
	]
		- Read the file into a JSON object
		- Loop through items in the array
		  ○ If the ID of the item is equal to the request parameter (req.params.id)
		  ▪ Remove the item from the array
		  ▪ You can use **break** to jump out of the loop
		- Use JSON.stringify() to convert the list of objects into a JSON String
		- Use fs.writeFileSync to overwrite the file with new JSON contents

Node.js Form

Node.Test URLs	Expected Output	Description / Hints

/form.html
 	
(List Button)
	(Assuming the original JSON list is used)
	
	 
		- Use the localhost /list URL to make updates to the JSON file, using fetch()
		- Display the new JSON object (returned by the API) by modifying the Inner HTML of a <p> Element]
		
		Sample code (below):
		- This must be inside an async function
		- The variable msg refers to a <p> element using document.getElementById()
		
		`let response = await fetch('http://localhost:3000/list');`
		`let data = await response.json();`
		`msg.innerHTML = JSON.stringify(data);`

/form.html (Create Button)	Input 1: 04 Input 2: Test 	- Two prompt windows, storing the ID and DESCRIPTION you are adding. - Use the localhost /add URL to make updates to the JSON file, using fetch() - Display the new JSON object (returned by the API) by modifying the Inner HTML of a <p> element
/form (Delete Button)	Input 1: 03 	- One prompt window, storing the ID of the object you are deleting - Use the localhost /delete URL to make updates to the JSON file, using fetch() - Display the new JSON object (returned by the API) by modifying the Inner HTML of a <p> element
/form (Table Button)	(Assuming the original JSON file is used) 	- Use the localhost /list URL to make updates to the JSON file - Create a table dynamically, displaying each object in a new row. - One way to do this is through a w3schools tutorial https://www.w3schools.com/jsref/met_table_insertrow.asp

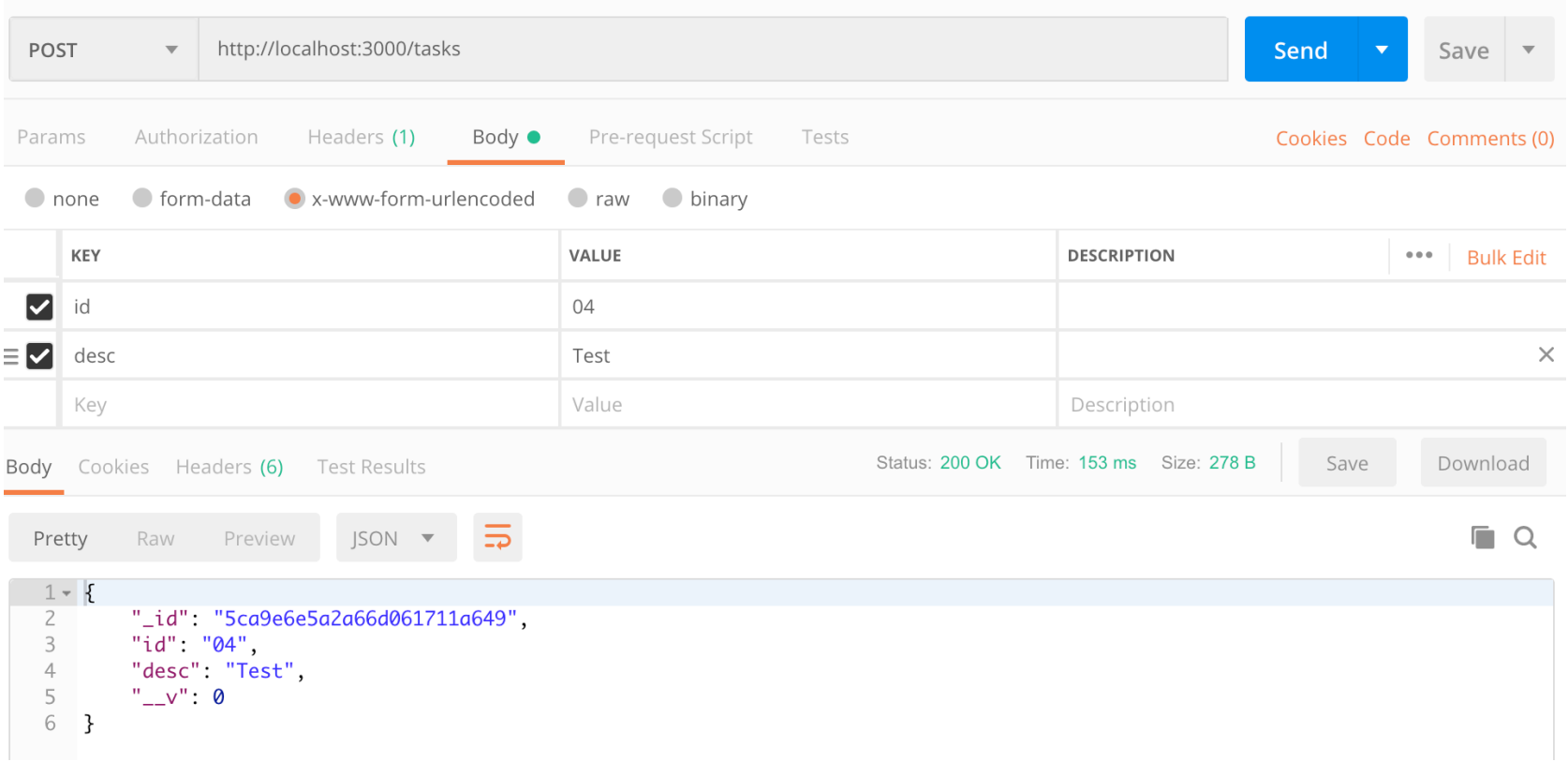
Node.js & MongoDB

Hint: https://www.codementor.io/olatundegaruba/nodejs-restful-apis-in-10-minutes-q0sgsfhbd (this is a useful tutorial, but it needs to be adapted to meet the specific requirements of the assignment)

Test URL: /tasks (POST)

Using Postman, you should be able to see that a new JSON object has been added to the database. In my example, this is created in a table called “todo” and a collection called “tasks”.

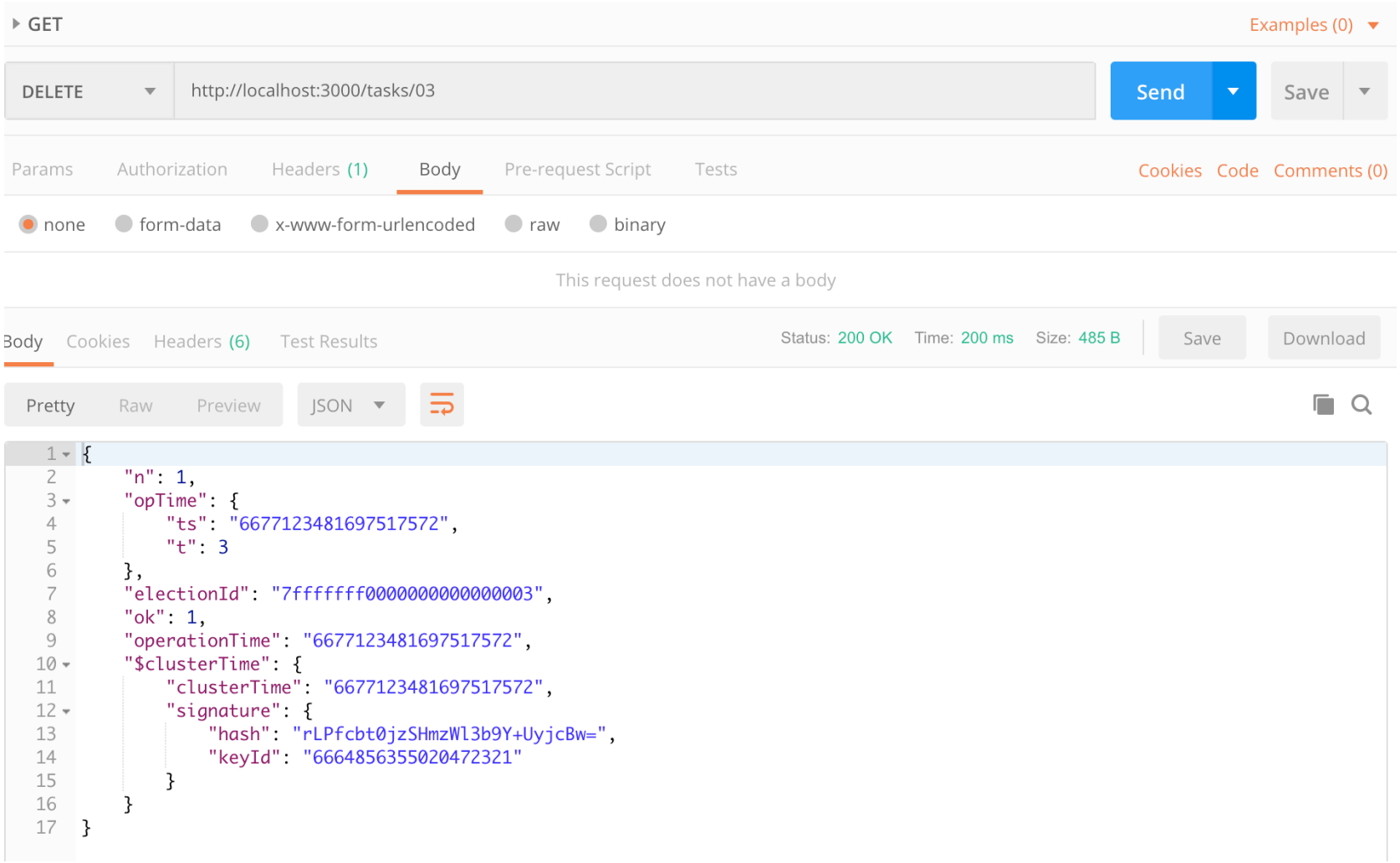
This example is making use of body parameters. For /dbform.html, the fetch() method can be provided with options to set a POST request with body parameters.



Test URL: /tasks (DELETE)

Using Postman, you should be able to see that the JSON object (id: 3) has been removed from the database.

This example is making use of a request parameter. For /dbform.html, the fetch() method can be provided with options to set a DELETE request.



Test URL: /tasks (GET)

Using Postman, you should be able to see all the JSON objects within the database.

Below is the list of objects displayed in the database after the following steps were completed:

- Added four JSON tasks
  - {“id”: “01”, “desc”: “Complete Learn Node Lessons 1-10”}
  - {“id”: “02”, “desc”: “Read Assignment 2 Instructions”} (This one was done manually using MongoDB Compass, so it looks slightly different)
  - {“id”: “03”, “desc”: “Complete Assignment 2”}
  - {“id”: “04”, “desc”: “Test”}
- Deleted the 3rd JSON task (using /delete/03 URL with DELETE specified in Postman)