

To Get the Response

Code

```
let responseData = pm.response.json();
console.log(responseData)
```

The screenshot shows the Postman API Workspace interface. On the left, there's a sidebar with 'collections', 'environments', 'History', and 'Flows'. Under 'Request Collection', there are several items: 'GET Status', 'GET List of Books', 'GET List of Books by Query', 'GET Book by ID', 'POST Register API Client', 'POST Submit Order', 'GET Order' (which is selected), and 'PATCH Update Order'. The main area shows a 'Request Collection / GET Order' tab. The 'GET' method is selected with the URL {{BaseUrl}} /orders/:orderId. Below the URL, there are tabs for 'Docs', 'Params', 'Authorization', 'Headers (8)', 'Body', 'Scripts', and 'Settings'. The 'Scripts' tab is active, containing the following code:

```
1 let responseData = pm.response.json();
2 console.log(responseData)
```

A red arrow points to the 'Send' button in the top right corner. Below the request, the 'Test Results' section shows a successful response: 200 OK, 280 ms, 848 B. A red box highlights the response body, which contains the JSON object: { "id": "y6ka50kBynzp3XQDhagkW", "bookId": 1, "customerName": "PostmanAPI" }.

Response

The screenshot shows the same Postman interface after sending the request. The 'Request Collection / GET Order' tab is still active. The 'GET' method is selected with the URL {{BaseUrl}} /orders/:orderId. The 'Scripts' tab is active, showing the same pre-request script. The 'Post-response' tab is also visible. The 'Body' tab is selected, showing the JSON response: { "id": "y6ka50kBynzp3XQDhagkW", "bookId": 1, "customerName": "PostmanAPI" }. Below the body, there are tabs for 'JSON', 'Preview', and 'Visualize'. The status bar at the bottom shows a 200 OK response, 929 ms, 848 B, and a timestamp of 1768398993734. A red box highlights the JSON response body.

Code:

```
/* {
```

```

"id": "y6ka5OkBynzp3XQDhagkW",
"bookId": 1,
"customerName": "PostmanAPI",
"quantity": 1,
"timestamp": 1768398993734
} */

let responseData = pm.response.json();

let jsonSchema = {
  "type": "object",
  "properties": {
    "id": {
      "type": "string"
    },
    "bookId": {
      "type": "integer"
    },
    "customerName": {
      "type": "string"
    },
    "quantity": {
      "type": "integer"
    },
    "timestamp": {
      "type": "integer"
    }
  }
}

pm.test("JSON schema Validation using direct method", () => {
  pm.response.to.have.jsonSchema(jsonSchema);
})

pm.test("Json Schema Validation", () => {
  pm.expect(tv4.validate(responseData, jsonSchema)).to.be.true;
})

```

Result

HTTP Request Collection / GET Order

Save Share

GET {{BaseUrl}} /orders/:orderId Send

Docs Params Authorization Headers (8) Body Scripts Settings Cookies

Pre-request Post-response

```
28 }  
29 }  
30 }  
31  
32 pm.test("JSON schema Validation using direct method", () => {  
33   pm.response.to.have.jsonSchema(jsonSchema);  
34 })
```

Test Results 200 OK • 261 ms • 848 B • Save Response

PASSED JSON schema Validation using direct method

PASSED Json Schema Validation

This screenshot shows the Postman interface for a 'GET Order' request. The top navigation bar includes 'HTTP Request Collection / GET Order', 'Save', 'Share', and a 'Send' button. Below the URL input field, there are tabs for 'Docs', 'Params', 'Authorization', 'Headers (8)', 'Body', 'Scripts', and 'Cookies'. The 'Scripts' tab is currently selected, displaying a code block for JSON schema validation. The 'Test Results' section shows a single test step named 'JSON schema Validation using direct method' which has passed. The status bar at the bottom indicates a 200 OK response with a duration of 261 ms and a size of 848 B.