ToDo API Documentation

This document outlines the API endpoints for the Node.js ToDo backend server.

The base URL is `http://localhost:8000` .

1. Get All ToDos

Retrieves a list of all todo items.

- **Method:** `GET`
- **Path:** `/todos`
- **Request Body:** None
- **Success Response (200 OK):**
  - **Content-Type:** `application/json`
  - **Body:** An array of todo objects.
  - **Example:**

```
[
  {
    "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "text": "Buy milk",
    "completed": false
  },
  {
    "id": "b2c3d4e5-f6a7-8901-bcde-f12345678901",
    "text": "Finish Node.js server",
    "completed": true
  }
]
```

2. Get a Single ToDo

Retrieves a specific todo item by its unique ID.

- **Method:** `GET`
- **Path:** `/todos/:id`
- **URL Parameters:**
  - `:id` (string, required): The unique identifier of the todo.
- **Request Body:** None
- **Success Response (200 OK):**
  - **Content-Type:** `application/json`
  - **Body:** The requested todo object.
  - **Example:**

```
{
  "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "text": "Buy milk",
  "completed": false
}
```

- **Error Response (404 Not Found):**
  - **Content-Type:** `application/json`
  - **Body:**

```
{
  "message": "Todo not found"
}
```

3. Create a New ToDo

Adds a new todo item to the list.

- **Method:** `POST`
- **Path:** `/todos`
- **Request Body:**
  - **Content-Type:** `application/json`
  - **Body:**

```
{
  "text": "A new todo item"
}
```

  - **Fields:**
    - `text` (string, required): The content of the todo item.
- **Success Response (201 Created):**
  - **Content-Type:** `application/json`
  - **Body:** The newly created todo object, including its new ID.
  - **Example:**

```
{
  "id": "c3d4e5f6-a7b8-9012-cdef-123456789012",
  "text": "A new todo item",
  "completed": false
}
```

- **Error Response (400 Bad Request):**
  - **Content-Type:** `application/json`

- **Body:**

```
{
  "message": "Todo text is required"
}
```

- *(Also 400 if the request body is invalid JSON)*

4. Update a ToDo

Updates an existing todo item by its ID. You can update the `text` and/or `completed` status.

- **Method:** `PUT`
- **Path:** `/todos/:id`
- **URL Parameters:**
  - `:id` (string, required): The unique identifier of the todo.
- **Request Body:**
  - **Content-Type:** `application/json`
  - **Body:** An object containing the fields to update.
  - **Example (updating text):**

```
{
  "text": "An updated todo item"
}
```

  - **Example (updating status):**

```
{
  "completed": true
}
```

  - **Example (updating both):**

```
{
  "text": "An updated todo item",
  "completed": true
}
```

- **Success Response (200 OK):**
  - **Content-Type:** `application/json`
  - **Body:** The fully updated todo object.
  - **Example:**

```
{
  "id": "c3d4e5f6-a7b8-9012-cdef-123456789012",
  "text": "An updated todo item",
  "completed": true
}
```

- **Error Response (404 Not Found):**
  - **Content-Type:** `application/json`
  - **Body:**

```
{
  "message": "Todo not found"
}
```

5. Delete a ToDo

Removes a todo item from the list by its ID.

- **Method:** `DELETE`
- **Path:** `/todos/:id`
- **URL Parameters:**
  - `:id` (string, required): The unique identifier of the todo.
- **Request Body:** None
- **Success Response (204 No Content):**
  - The server returns an empty body for a successful deletion.
- **Error Response (404 Not Found):**
  - **Content-Type:** `application/json`
  - **Body:**

```
{
  "message": "Todo not found"
}
```