

Express API Article Example



```
import express from "express";
import cors from "cors";

const app = express();
const PORT = 5000;

app.use(cors());
app.use(express.json());

// Articles array of objects
const articles = [
  {
    id: 1,
    title: "Introduction to React",
    summary:
      "Learn the basics of React, a powerful library for building user interfaces",
    content:
      "React is a JavaScript library for building user interfaces. It is maintained by Facebook.,React is a powerful JavaScript library designed for building interactive and dynamic user interfaces. Originally developed by Facebook in 2013, it has become one of the most popular tools for creating single-page applications (SPAs). If you're new to React, this guide will introduce you to its core concepts and why it's worth learning.What is React?React is a component-based library that allows developers to break down their application into smaller, reusable pieces called components. These components can handle their own logic and rendering, making development more efficient and scalable.",
    featuredImage: "https://via.placeholder.com/300x200?text=React",
    reporterName: "John Doe",
  },
  {
    id: 2,
    title: "Getting Started with Express",
    summary:
      "An overview of Express.js, a lightweight Node.js web application framework.",
    content:
      "Express is a minimal and flexible Node.js web application framework that provides a robust set of features...",
    featuredImage: "https://via.placeholder.com/300x200?text=Express",
    reporterName: "Jane Smith",
  },
  {
    id: 3,
    title: "Understanding JavaScript Closures",
    summary: "A deep dive into closures, a core concept in JavaScript.",
    content:
      "Closures are a fundamental concept in JavaScript that allow functions to access variables...",
  }
]
```

```

    featuredImage: "https://via.placeholder.com/300x200?text=JavaScript",
    reporterName: "Alice Johnson",
  },
  {
    id: 4,
    title: "CSS Grid Layout",
    summary:
      "Explore the powerful features of CSS Grid for creating modern web layouts.",
    content:
      "CSS Grid Layout is a two-dimensional layout system for the web. It lets developers design layouts...",
    featuredImage: "https://via.placeholder.com/300x200?text=CSS+Grid",
    reporterName: "Robert Brown",
  },
  {
    id: 5,
    title: "Responsive Design Tips",
    summary:
      "Tips and tricks for making your websites look great on any device.",
    content:
      "Responsive design ensures that your website works on all devices, from mobile phones to desktops...",
    featuredImage: "https://via.placeholder.com/300x200?text=Responsive+Design",
    reporterName: "Emily Davis",
  },
  {
    id: 6,
    title: "Responsive Design Tips",
    summary:
      "Tips and tricks for making your websites look great on any device.",
    content:
      "Responsive design ensures that your website works on all devices, from mobile phones to desktops...",
    featuredImage: "https://via.placeholder.com/300x200?text=Responsive+Design",
    reporterName: "Emily Davis",
  },
];

// Route to get all articles
app.get("/articles", (req, res) => {
  res.json(articles);
});

// Route to get a single article by ID
app.get("/articles/:id", (req, res) => {
  const articleId = parseInt(req.params.id);
  const article = articles.find((a) => a.id === articleId);

  if (article) {
    res.json(article);
  } else {

```

```
res.status(404).json({ message: "Article not found" });
}
});

app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```

explain the code



This code defines a simple **Express.js** server that provides a REST API for managing and retrieving articles. Here's a breakdown of the code:

1. Importing Dependencies

javascript

```
import express from "express";
import cors from "cors";
```

- `express`: A Node.js framework for building web applications and APIs.
- `cors`: Middleware to enable **Cross-Origin Resource Sharing**, allowing the server to handle requests from different origins (e.g., a React frontend hosted on another domain).

2. Setting Up the Server

javascript

```
const app = express();
const PORT = 5000;
```

- `app`: The main Express application object.
- `PORT`: The port number on which the server listens for incoming requests.

3. Middleware

javascript

```
app.use(cors());
app.use(express.json());
```

- `cors()`: Enables cross-origin requests to the API.
- `express.json()`: Parses incoming JSON request bodies, making them accessible in `req.body`.

4. Articles Array

The `articles` array is a mock database containing objects representing articles, each with the following fields:

- `id`: A unique identifier for the article.
- `title`: The article's title.
- `summary`: A brief description of the article.
- `content`: The full content of the article.
- `featuredImage`: A URL for the article's image.
- `reporterName`: The name of the author.

Example:

```
javascript

{
  id: 1,
  title: "Introduction to React",
  summary: "Learn the basics of React, a powerful library for building user interfaces",
  content: "...",
  featuredImage: "https://via.placeholder.com/300x200?text=React",
  reporterName: "John Doe",
}
```

5. Routes

Route to Fetch All Articles

```
javascript

app.get("/articles", (req, res) => {
  res.json(articles);
});
```

- **Endpoint:** `GET /articles`
- **Response:** Returns the entire `articles` array as JSON.
- Example Output:

```
json

[
  { "id": 1, "title": "Introduction to React", ... },
  { "id": 2, "title": "Getting Started with Express", ... }
]
```

Route to Fetch a Single Article by ID

```
javascript
```

```
app.get("/articles/:id", (req, res) => {
  const articleId = parseInt(req.params.id);
  const article = articles.find((a) => a.id === articleId);

  if (article) {
    res.json(article);
  } else {
    res.status(404).json({ message: "Article not found" });
  }
});
```

- **Endpoint:** `GET /articles/:id`
- **Parameter:** `:id` is the ID of the requested article.
- **Response:**
 - If the article exists: Returns the matching article object as JSON.
 - If the article doesn't exist: Responds with a **404** status and an error message.
- Example Output for `/articles/1`:

```
json
```

```
{
  "id": 1,
  "title": "Introduction to React",
  "summary": "Learn the basics of React, a powerful library for building user interfaces",
  "content": "...",
  "featuredImage": "https://via.placeholder.com/300x200?text=React",
  "reporterName": "John Doe"
}
```

- Example Error Output for `/articles/999`:

```
json
```

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

6. Starting the Server

```
javascript
```

```
app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```

- Starts the Express server on the defined `PORT` (5000).
- Logs a message indicating the server is running and accessible.

Summary

This server provides:

1. **An endpoint** (`GET /articles`) to retrieve all articles.
2. **An endpoint** (`GET /articles/:id`) to retrieve a specific article by ID.
3. Uses **CORS** to allow cross-origin requests.
4. Uses an in-memory data structure (`articles`) instead of a database for simplicity.

This setup can be used as a backend for a simple React or any frontend application to display articles dynamically.