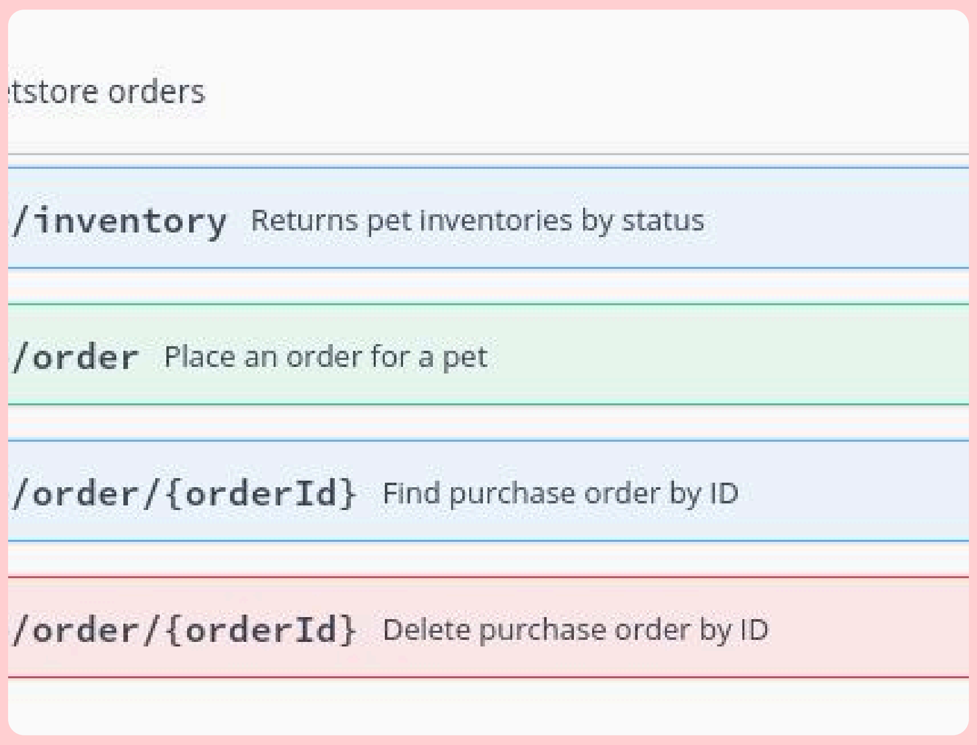


```
tstore orders
/inventory  Returns pet inventories by status
/order  Place an order for a pet
/order/{orderId}  Find purchase order by ID
/order/{orderId}  Delete purchase order by ID
```

Date: May 10th, 2025

HTTP METHODS

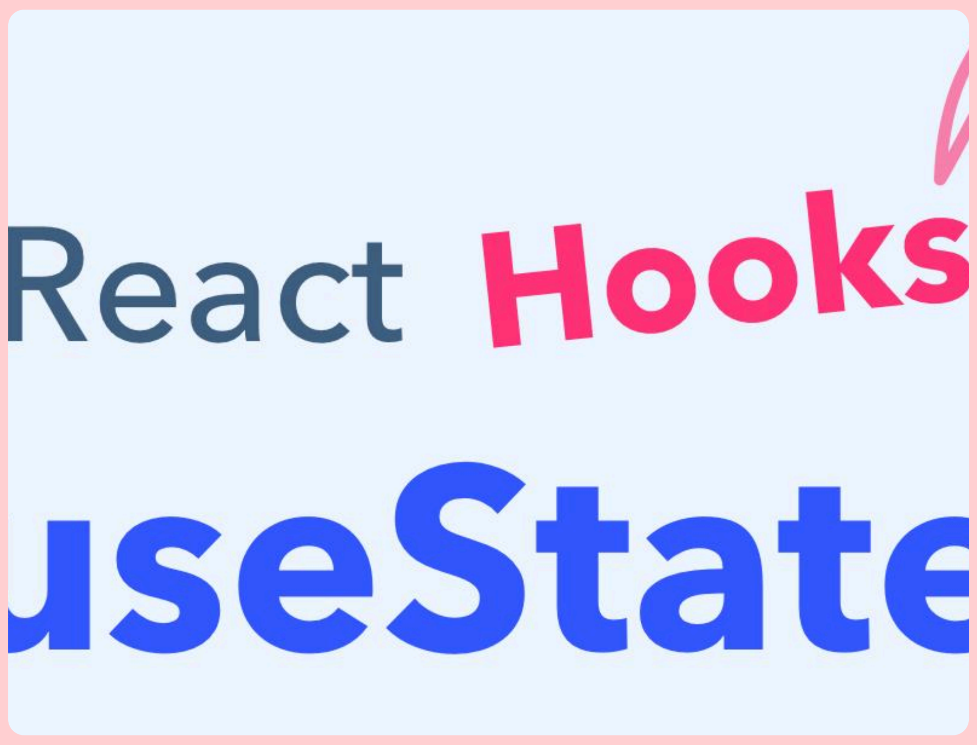
Four HTTP methods every developer should know

Mood Muzi
© 2025 Mood Muzik. All rights reserved.
Terms of Service Privacy Policy

Date: May 10th, 2025

MOOD MUZIK

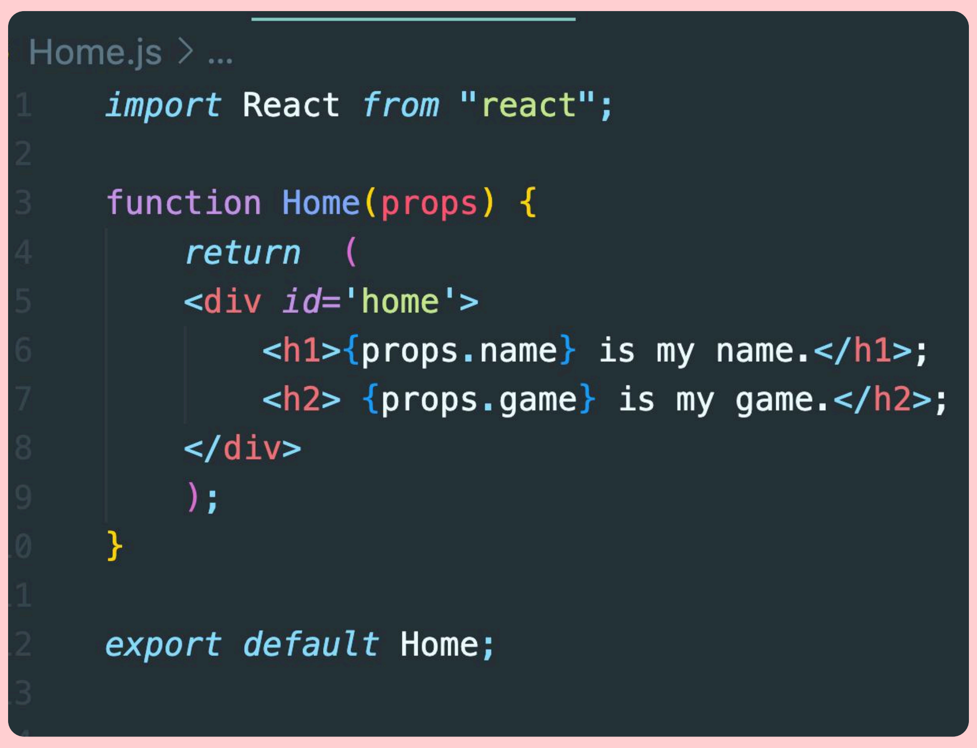
Four HTTP methods every developer should know



Date: May 10th, 2025

REACT HOOKS: USE STATE

Four HTTP methods every developer should know



```
Home.js > ...
    import React from "react";

    function Home(props) {
        return  (
        <div id='home'>
            <h1>{props.name} is my name.</h1>;
            <h2> {props.game} is my game.</h2>;
        </div>
        );
    }

    export default Home;
```

Date: May 10th, 2025

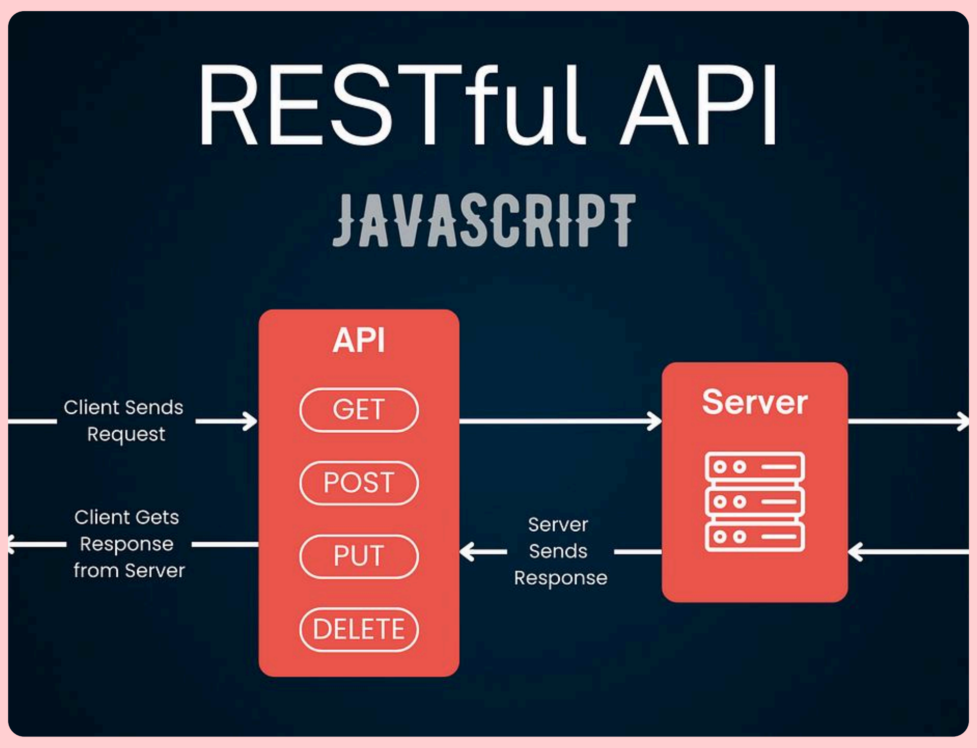
REACT PROPS

Four HTTP methods every developer should know

Date: May 10th, 2025

Connecting Express to PostgreSQL

Four HTTP methods every developer should know



Date: May 10th, 2025

Building REST APIs with Express

Four HTTP methods every developer should know

CONTACT ME

NAME

EMAIL

MESSAGE

SEND

HTTP METHODS

Four HTTP methods every developer should know

Whether you're building REST APIs or fetching data in a frontend app, understanding HTTP methods is essential. Here are the four fundamental HTTP methods every developer should know and when to use them.

1. GET — Retrieve Data
The GET method is used to fetch data from a server. It’s safe, meaning it doesn't modify any data, and can be cached or bookmarked.
Use GET when:

- Fetching posts from a blog
- Getting user profile info
- Viewing a product catalog

Example:
```
fetch('/api/posts') .then(res => res.json()) .then(data => console.log(data));
```

2. POST — Send New Data
The POST method is used to send new data to the server — commonly used for creating resources.
Use POST when:

- Submitting a form
- Adding a comment
- Creating a new user account

Example:
```
fetch('/api/posts', { method: 'POST', headers: { 'Content-Type': 'application/json' },
body: JSON.stringify({ title: 'Hello World', content: 'First post!' }) });
```

3. PUT — Update Existing Data
The PUT method is used to update an existing resource. It usually requires sending the entire updated object.
Use PUT when:

- Replacing an entire blog post
- Overwriting profile settings

Example:
```
fetch('/api/posts/1', { method: 'PUT', headers: { 'Content-Type': 'application/json' },
body: JSON.stringify({ title: 'Updated Title', content: 'New content here.' }) });
```

4. DELETE — Remove Data
The DELETE method removes a resource from the server.
Use DELETE when:

- Deleting a comment
- Removing a user
- Unpublishing a blog post

Example:
```
fetch('/api/posts/1', { method: 'DELETE' });
```

JAQUAVIA
SOFTWARE ENGINEER

I'm a full-stack developer who loves building things that work and look good. Here, I break down front-end and back-end concepts in a way that's fun, real, and actually makes sense

NEW BLOG POST

Title

Description

Content

SEND