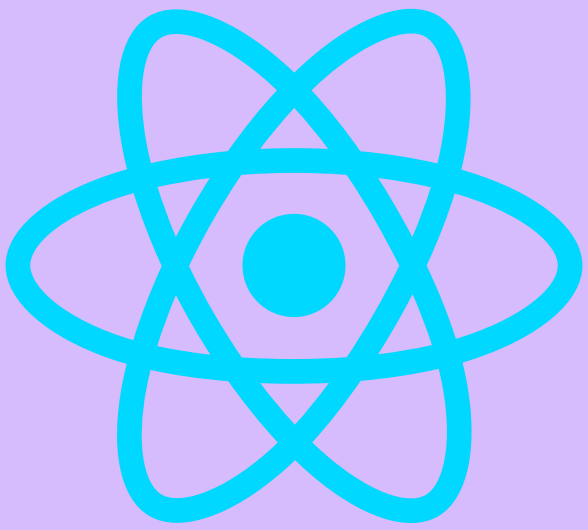
JSON Web Token (JWT) in React.

Simplified Security



Vladyslav Demirov

@vladyslav-demirov

What is JWT?

JSON Web Token (JWT): A compact, URL-safe token used to securely transmit information between parties.

- **Structure**:
  1. **Header**: Contains the type of token and signing algorithm.
  2. **Payload**: Contains claims (user data or metadata).
  3. **Signature**: Verifies the integrity of the token.

jwt

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
.eyJ1c2VySWQiOiIxMjM0NTY3ODkwIiwicm9sZSI6ImFkbWluIn0
.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Why Use JWT?

- Stateless authentication.
- No need for server-side session storage.
- Secure and efficient.

Vladyslav Demirov

How JWT Works in React

Authentication Flow:

- **User Login**:

• User submits credentials (email/password).

• Server validates credentials and returns a JWT.

- **Store Token**:

• Save the token in **localStorage** or **httpOnly cookies**.

- **Attach Token**:

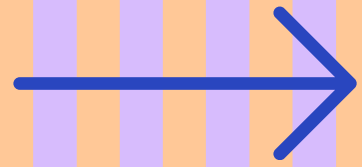
• Include the token in the **Authorisation header** for protected API calls.

- **Verify & Decode**:

• Server verifies the token and grants/denies access.

• Decode the payload on the client for user-specific info (e.g., roles).

Vladyslav Demirov



Implementation in React

Login & Store Token:

jsx

```jsx
import React, { useState } from 'react';

const Login = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleLogin = async () => {
    try {
      const response = await fetch('https://yourapi.com/login', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ email, password }),
      });

      const data = await response.json();
      if (data.token) {
        localStorage.setItem('jwt', data.token); // Save JWT
        alert('Login successful!');
      } else {
        alert('Invalid credentials');
      }
    } catch (err) {
      console.error(err);
    }
  };

  return (
    <div>
      <input
        type="email"
        placeholder="Email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
      />
      <input
        type="password"
        placeholder="Password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
      />
      <button onClick={handleLogin}>Login</button>
    </div>
  );
};

export default Login;
```

Vladyslav Demirov

Protect Routes

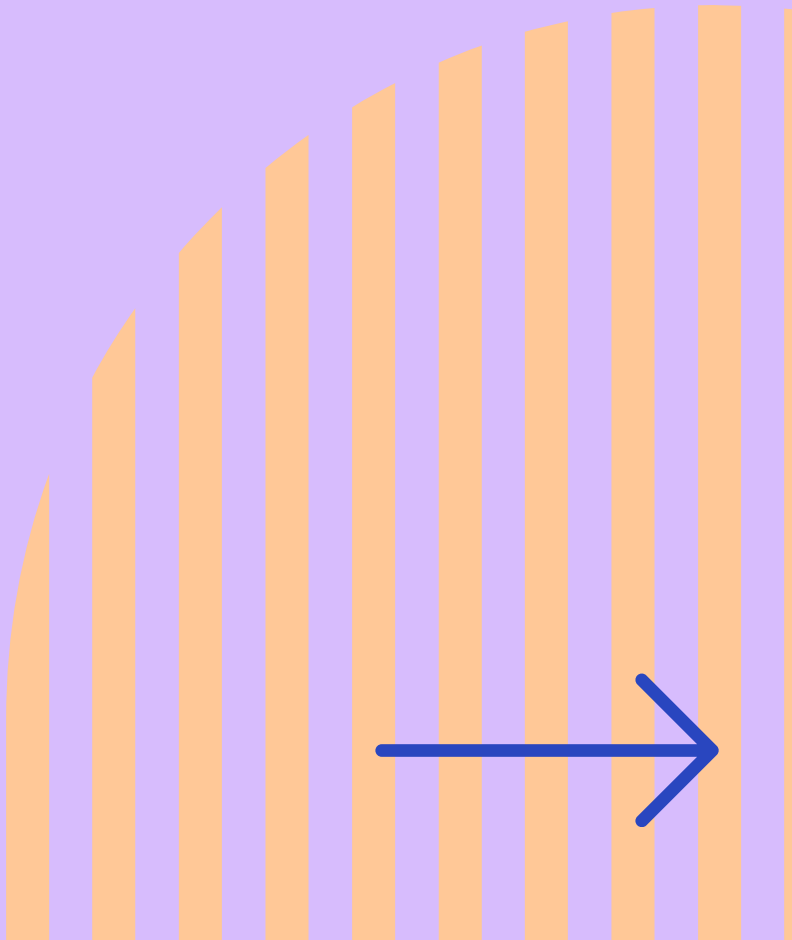
Create Protected Routes:

```jsx
import React from 'react';
import { Navigate } from 'react-router-dom';

const ProtectedRoute = ({ children }) => {
  const token = localStorage.getItem('jwt'); // Get JWT from storage
  return token ? children : <Navigate to="/login" />;
};

export default ProtectedRoute;
```

Usage:

```jsx
<ProtectedRoute>
  <Dashboard />
</ProtectedRoute>
```

Vladyslav Demirov



Decode JWT

Decode JWT for User Info:

jsx

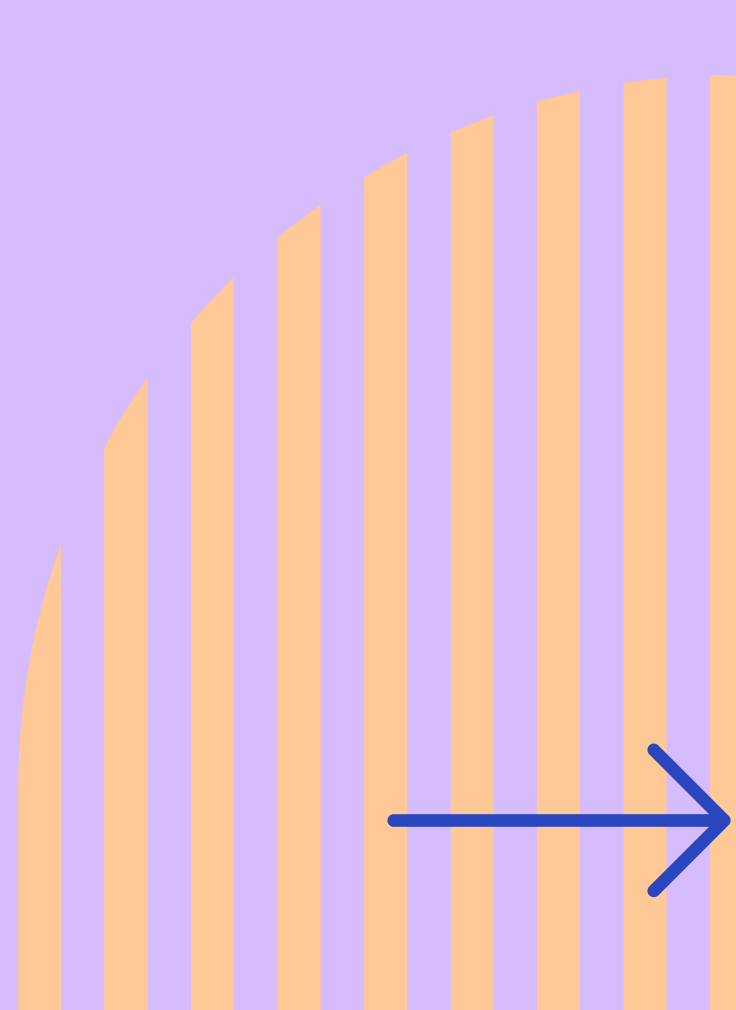
```jsx
import jwtDecode from 'jwt-decode';

const getUserInfo = () => {
  const token = localStorage.getItem('jwt');
  if (token) {
    const decoded = jwtDecode(token);
    console.log('User Info:', decoded);
    return decoded;
  }
  return null;
};
```

Common Use Case: Display user name or roles dynamically.

jsx

```jsx
const user = getUserInfo();
return <h1>Welcome, {user?.name}!</h1>;
```

Vladyslav Demirov



Conclusion

JWT in React allows you to:

- Securely handle user authentication.
- Protect routes easily.
- Decode and use user data dynamically.

Tips for Security:

- **Use httpOnly cookies** for better security against XSS attacks.
- Always **validate** JWT on the **server**.
- Implement **token expiration** and handle refresh tokens.

Vladyslav Demirov

HAPPY CODING

Vladyslav Demirov
@vladyslav-demirov