Use Context to Share a Full REST API

Peter Kellner
Developer, Consultant and Author

ReactAtScale.com @pkellner linkedin.com/in/peterkellner99

Demo

Overview of What’s Coming

Create generalized capability for(C)reate (R)ead (U)pdate (D)elete operations (CRUD)

Build out a local REST server supporting CRUD with the NextJS API Routes

Along the way, create a new SpeakerContext the includes CRUD operations and an expanded custom React hook for data updates

Goal for This Clip

CRUD

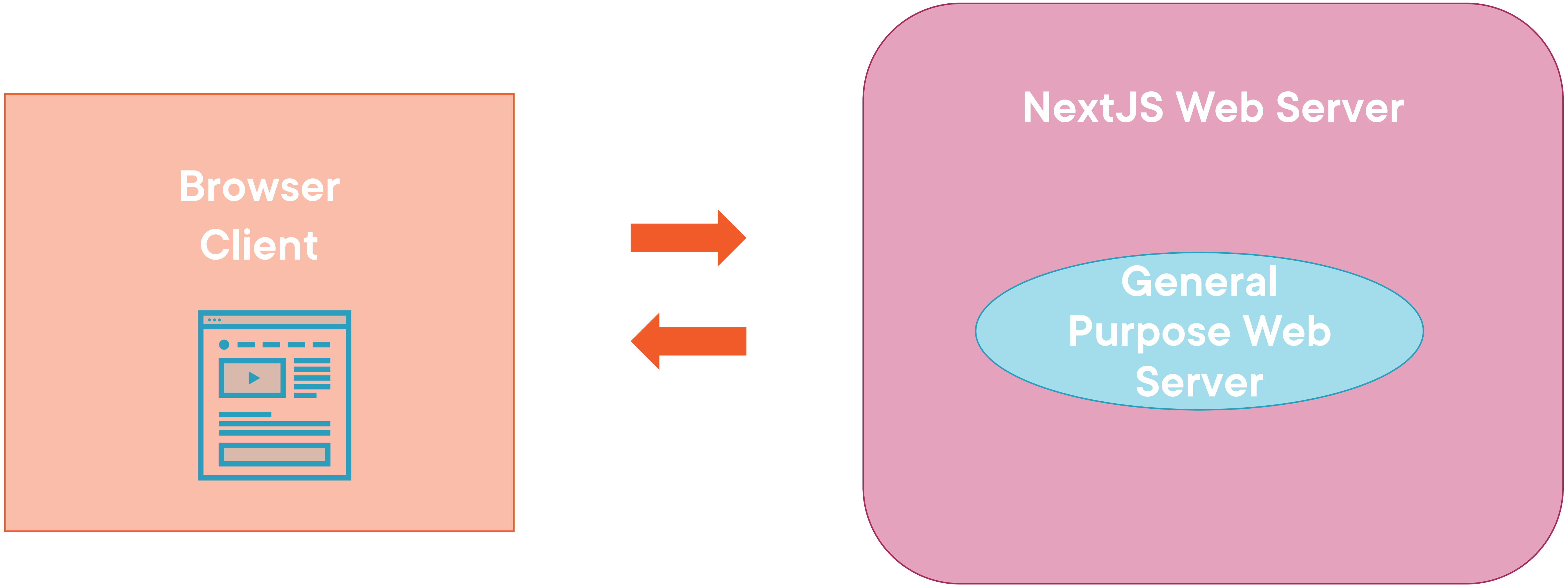
Support full (C)reate (R)read (U)pdate (D)elete operations

New methods

Add create and delete operations to our custom React hook

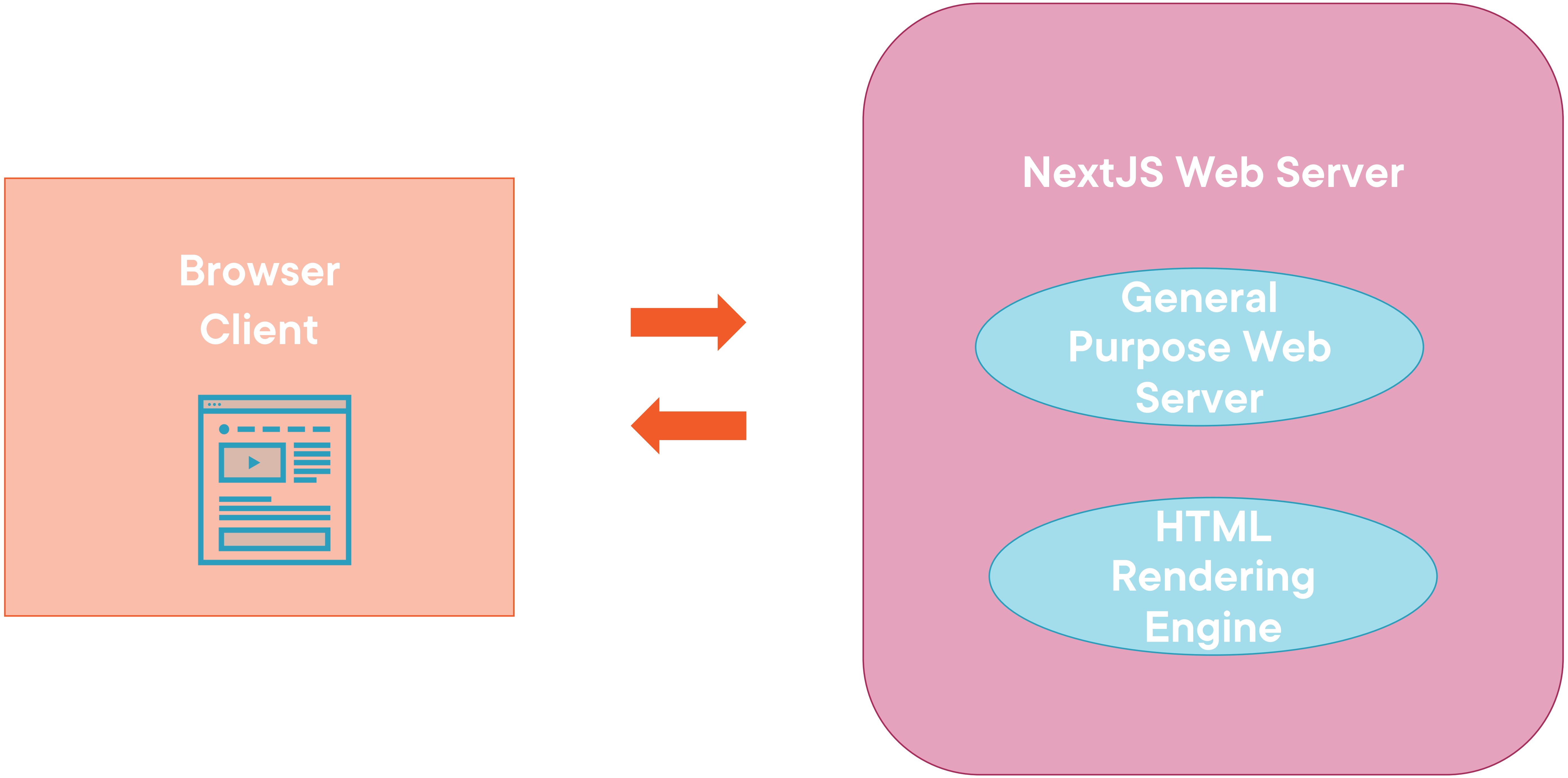
HTTP VERB	URL Endpoint
GET	/api/speakers
PUT	/api/speakers/$ID
POST	/api/speakers/$ID
DELETE	/api/speakers/$ID

How NextJS Works

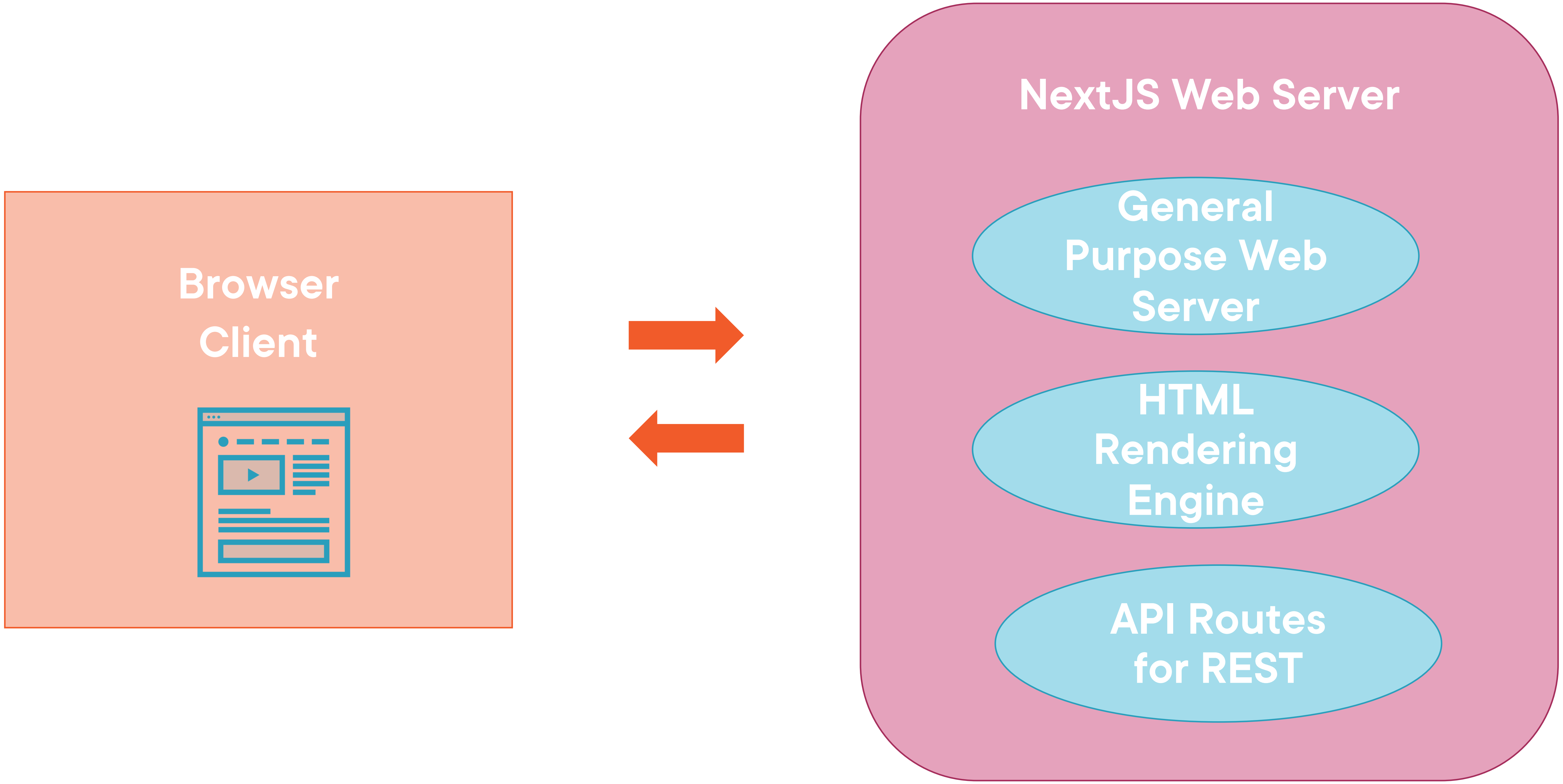


/pages/RouteName

How NextJS Works

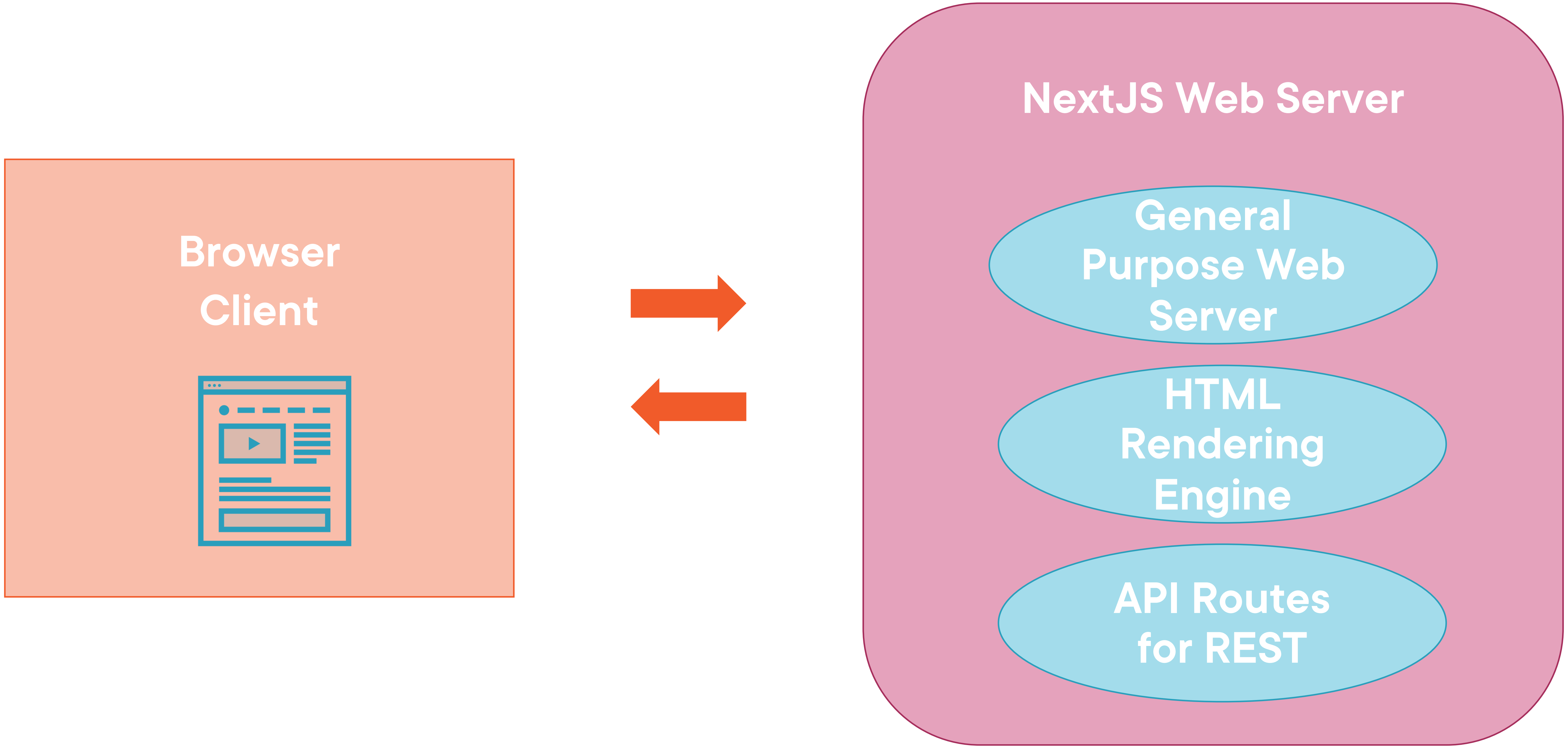


How NextJS Works



/pages/api/RESTRoutes

How NextJS Works



Browser Client

NextJS Web Server

General Purpose Web Server

HTML Rendering Engine

API Routes for REST

Module Takeaway

- Built a full REST server using NextJS API Routes
- Context Providers provide shared access to all children components
- Context Providers can leverage React lifecycle events like useEffect
- Context and custom React hooks provide a lightweight solution for easy state management