JONAS.IO
SCHMEDTMANN

NODE.JS, EXPRESS & MONGODB

THE COMPLETE BOOTCAMP

@JONASSCHMEDTMAN

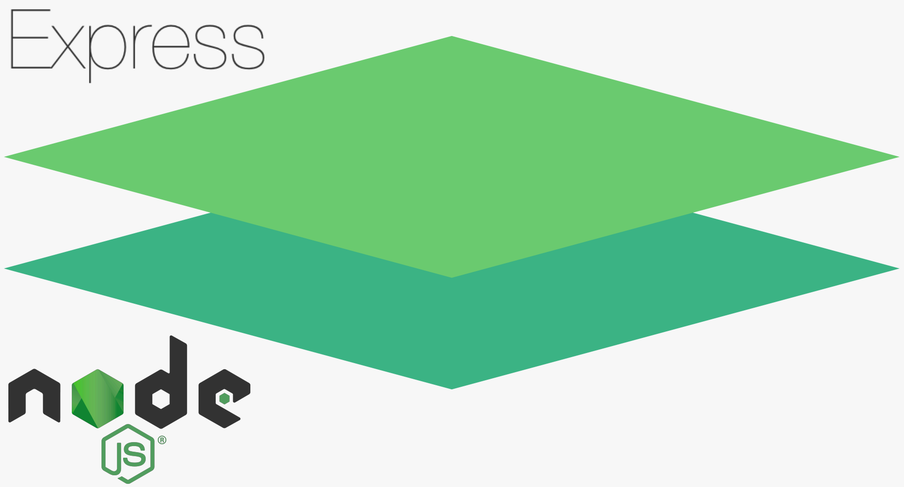
SECTION

EXPRESS: LET'S START BUILDING THE NATOURS API!

LECTURE

WHAT IS EXPRESS?

WHAT IS EXPRESS, AND WHY USE IT?



- 👉 Express is a minimal node.js framework, a higher level of abstraction;
- 👉 Express contains a very robust set of features: **complex routing**, **easier handling of requests and responses**, **middleware**, **server-side rendering**, etc.;
- 👉 Express allows for rapid development of node.js applications: *we don't have to re-invent the wheel*;
- 👉 Express makes it easier to organize our application into the MVC architecture.

JONAS.IO
SCHMEDTMANN

NODE.JS, EXPRESS & MONGODB

THE COMPLETE BOOTCAMP

@JONASSCHMEDTMAN

SECTION

EXPRESS: LET'S START BUILDING THE NATOURS API!

LECTURE

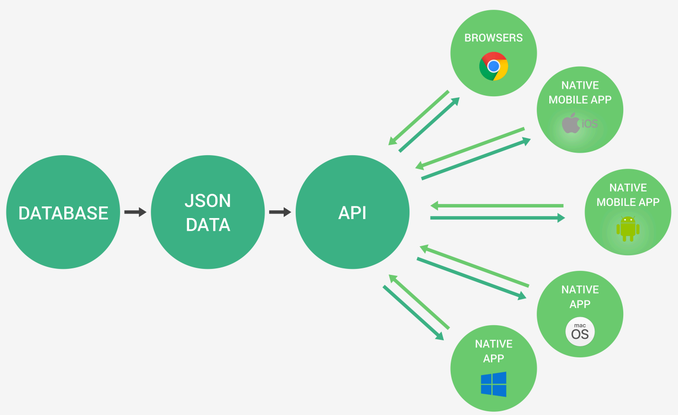
APIS AND RESTFUL API DESIGN

WHAT IS AN API ANYWAY?

API

Application **P**rogramming **I**nterface: a piece of software that can be used by another piece of software, in order to allow applications to talk to each other.

👉 Web APIs



👉 But, “Application” can be other things:

- 👉 Node.js’ `fs` or `http` APIs (“node APIs”);
- 👉 Browser’s DOM JavaScript API;
- 👉 With object-oriented programming, when exposing methods to the public, we’re creating an API;
- 👉 ...

THE REST ARCHITECTURE

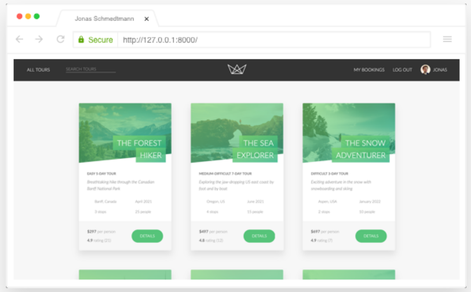
1. Separate API into logical **resources**
2. Expose structured, **resource-based URLs**
3. Use **HTTP methods** (verbs)
4. Send data as **JSON** (usually)
5. Be **stateless**

THE REST ARCHITECTURE

1. Separate API into logical **resources**
2. Expose structured, **resource-based URLs**
3. Use **HTTP methods** (verbs)
4. Send data as **JSON** (usually)
5. Be **stateless**

👉 **Resource:** Object or representation of something, which has data associated to it. Any information that can be **named** can be a resource.

`tours` `users` `reviews`



URL

`https://www.natours.com/addNewTour`

ENDPOINT

`/getTour`

`/updat`

BAD 👎

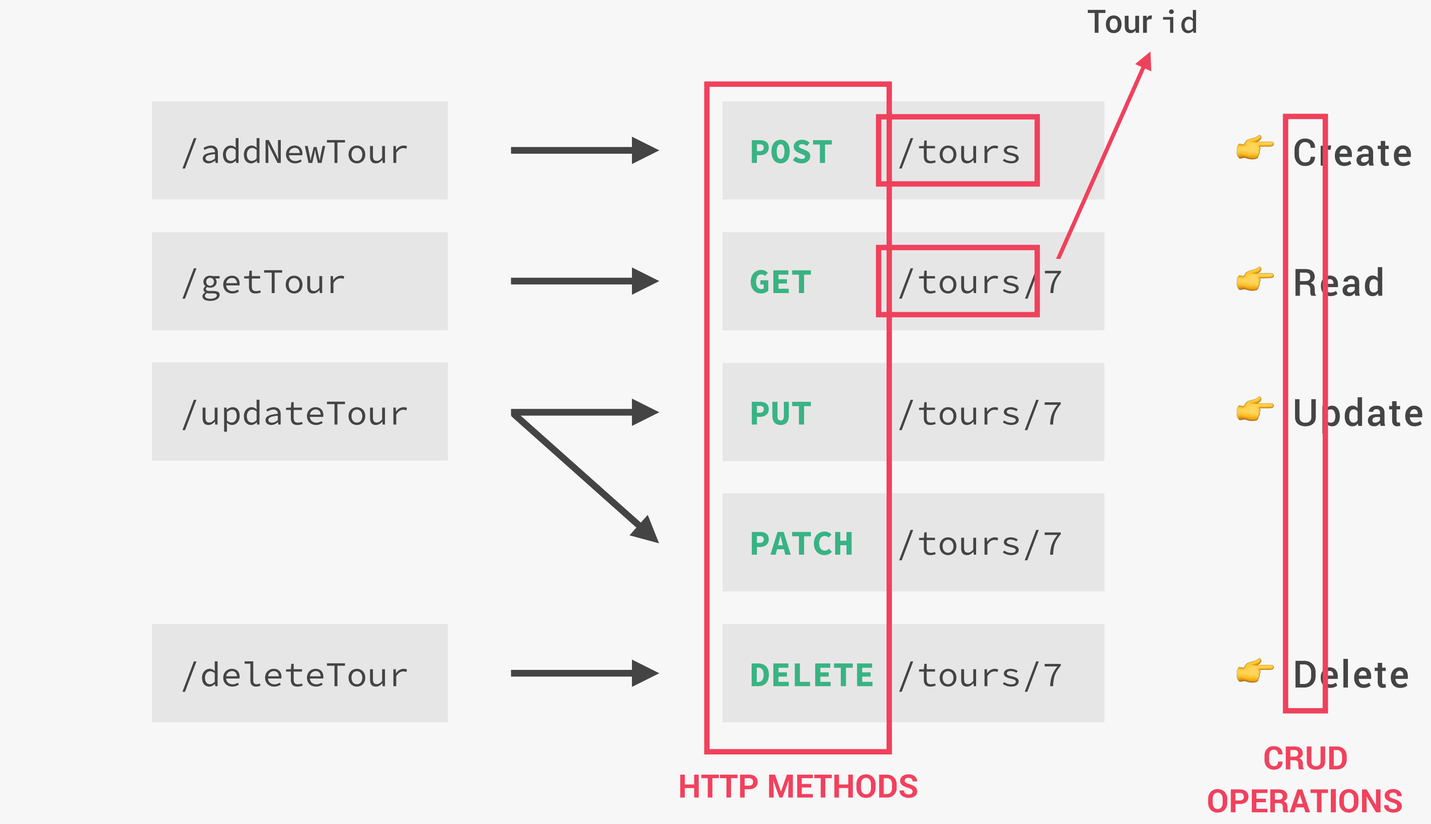
`/getToursByUser`

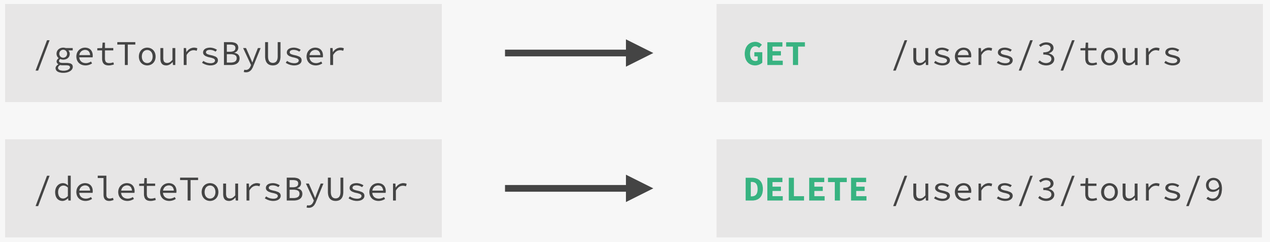
`/deleteToursByUser`

👉 Endpoints should contain **only resources** (nouns), and use **HTTP methods** for actions!

THE REST ARCHITECTURE

1. Separate API into logical **resources**
2. Expose structured, **resource-based URLs**
3. Use **HTTP methods** (verbs)
4. Send data as **JSON** (usually)
5. Be **stateless**





👉 Possibilities are endless!

THE REST ARCHITECTURE

1. Separate API into logical **resources**
2. Expose structured, **resource-based URLs**
3. Use **HTTP methods** (verbs)
4. Send data as **JSON** (usually)
5. Be **stateless**

RESPONSE FORMATTING

👉 **JSend**

👉 **JSON:API**

👉 **OData JSON Protocol**

👉 **...**

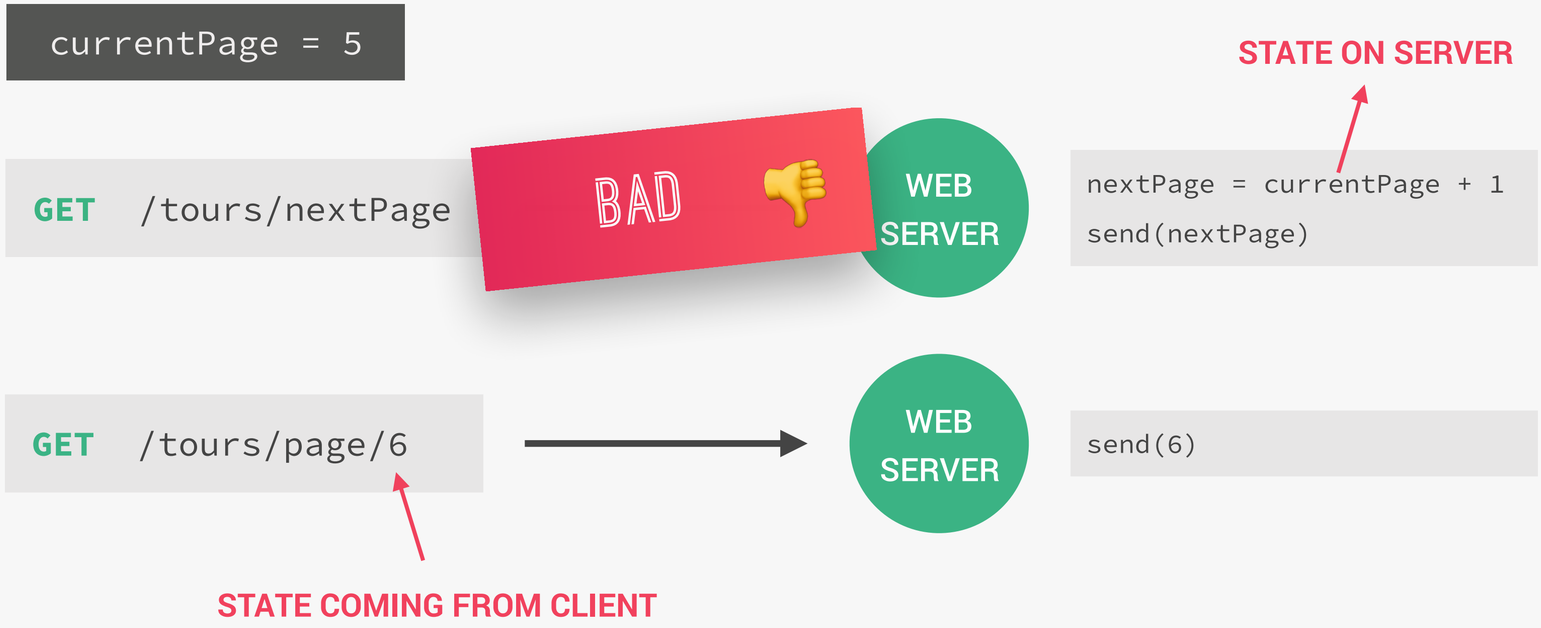
https://www.natours.com/tours/5

THE REST ARCHITECTURE

1. Separate API into logical **resources**
2. Expose structured, **resource-based URLs**
3. Use **HTTP methods** (verbs)
4. Send data as **JSON** (usually)
5. Be **stateless**

👉 **Stateless RESTful API:** All state is handled **on the client**. This means that each request must contain **all** the information necessary to process a certain request. The server should **not** have to remember previous requests.

👉 **Examples of state:** `loggedIn` `currentPage`



JONAS.IO
SCHMEDTMANN

NODE.JS, EXPRESS & MONGODB

THE COMPLETE BOOTCAMP

@JONASSCHMEDTMAN

SECTION

EXPRESS: LET'S START BUILDING THE NATOURS API!

LECTURE

MIDDLEWARE AND THE REQUEST-RESPONSE CYCLE

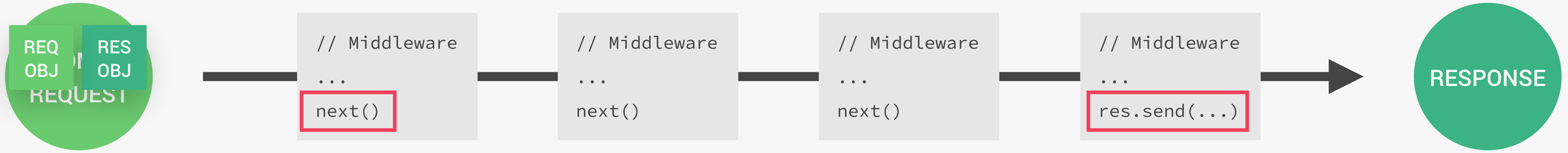
THE ESSENCE OF EXPRESS DEVELOPMENT: THE REQUEST-RESPONSE CYCLE

👉 *"Everything is middleware" (even routers)*

👉 *"Pipeline"*

👉 **Order as defined in the code!**

MIDDLEWARE STACK



👉 **E.g:** parsing body

👉 **E.g:** logging

👉 **E.g:** setting headers

👉 **E.g:** router

REQUEST-RESPONSE CYCLE