Dynamic API Routes

Examples

Basic API Routes

API routes support dynamic routes, and follow the same file naming rules used for `pages`.

For example, the API route `pages/api/post/[pid].js` has the following code:

```
export default function handler(req, res) {
  const { pid } = req.query
  res.end(`Post: ${pid}`)
}
```

Now, a request to `/api/post/abc` will respond with the text: `Post: abc`.

Index routes and Dynamic API routes

A very common RESTful pattern is to set up routes like this:

- `GET api/posts` - gets a list of posts, probably paginated
- `GET api/posts/12345` - gets post id 12345

We can model this in two ways:

- Option 1:
  - `/api/posts.js`
  - `/api/posts/[postId].js`
- Option 2:
  - `/api/posts/index.js`
  - `/api/posts/[postId].js`

Both are equivalent. A third option of only using `/api/posts/[postId].js` is not valid because Dynamic Routes (including Catch-all routes - see below) do not have an `undefined` state and `GET api/posts` will not match `/api/posts/[postId].js` under any circumstances.

Catch all API routes

API Routes can be extended to catch all paths by adding three dots (`...`) inside the brackets. For example:

- `pages/api/post/[...slug].js` matches `/api/post/a`, but also `/api/post/a/b`, `/api/post/a/b/c` and so on.

  Note: You can use names other than `slug`, such as: `[...param]`

Matched parameters will be sent as a query parameter (`slug` in the example) to the page, and it will always be an array, so, the path `/api/post/a` will have the following `query` object:

```
{ "slug": ["a"] }
```

And in the case of `/api/post/a/b`, and any other matching path, new parameters will be added to the array, like so:

```
{ "slug": ["a", "b"] }
```

An API route for `pages/api/post/[...slug].js` could look like this:

```
export default function handler(req, res) {
  const { slug } = req.query
  res.end(`Post: ${slug.join(', ')}`)
}
```

Now, a request to `/api/post/a/b/c` will respond with the text: `Post: a, b, c`.

Optional catch all API routes

Catch all routes can be made optional by including the parameter in double brackets (`[[...slug]]`).

For example, `pages/api/post/[[...slug]].js` will match `/api/post`, `/api/post/a`, `/api/post/a/b`, and so on.

The main difference between catch all and optional catch all routes is that with optional, the route without the parameter is also matched (`/api/post` in the example above).

The `query` objects are as follows:

```
{ } // GET `/api/post` (empty object)
{ "slug": ["a"] } // `GET /api/post/a` (single-element array)
{ "slug": ["a", "b"] } // `GET /api/post/a/b` (multi-element array)
```

Caveats

- Predefined API routes take precedence over dynamic API routes, and dynamic API routes over catch all API routes. Take a look at the following examples:
  - `pages/api/post/create.js` - Will match `/api/post/create`
  - `pages/api/post/[pid].js` - Will match `/api/post/1`, `/api/post/abc`, etc. But not `/api/post/create`
  - `pages/api/post/[...slug].js` - Will match `/api/post/1/2`, `/api/post/a/b/c`, etc. But not `/api/post/create`, `/api/post/abc`

Related

For more information on what to do next, we recommend the following sections:

Dynamic Routes: Learn more about the built-in dynamic routes.