```
      'Content-Type': 'application/json',
      'API-Key': process.env.DATA_API_KEY!,
    },
    body: JSON.stringify({ time: new Date().toISOString() }),
  })

  const data = await res.json()

  return Response.json(data)
}
```

app/items/route.js (js)

```
export async function POST() {
  const res = await fetch('https://data.mongodb-api.com/...', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'API-Key': process.env.DATA_API_KEY,
    },
    body: JSON.stringify({ time: new Date().toISOString() }),
  })

  const data = await res.json()

  return Response.json(data)
}
```

Good to know: Like API Routes, Route Handlers can be used for cases like handling form submissions. A new abstraction for handling forms and mutations that integrates deeply with React is being worked on.

Route Resolution

You can consider a `route` the lowest level routing primitive.

- They **do not** participate in layouts or client-side navigations like `page`.
- There **cannot** be a `route.js` file at the same route as `page.js`.

Page	Route	Result
`app/page.js`	`app/route.js`	Conflict
`app/page.js`	`app/api/route.js`	Valid
`app/[user]/page.js`	`app/api/route.js`	Valid

Each `route.js` or `page.js` file takes over all HTTP verbs for that route.

app/page.js (jsx)

```
export default function Page() {
  return <h1>Hello, Next.js!</h1>
}

// ❌ Conflict
// `app/route.js`
export async function POST(request) {}
```

Examples

The following examples show how to combine Route Handlers with other Next.js APIs and features.

Revalidating Cached Data

You can revalidate cached data using the `next.revalidate` option:

app/items/route.ts (ts)

```
export async function GET() {
  const res = await fetch('https://data.mongodb-api.com/...', {
    next: { revalidate: 60 }, // Revalidate every 60 seconds
  })
  const data = await res.json()
```