Module Not Found

Why This Error Occurred

A module not found error can occur for many different reasons:

- The module you're trying to import is not installed in your dependencies
- The module you're trying to import is in a different directory
- The module you're trying to import has a different casing
- The module you're trying to import uses Node.js specific modules, for example `dns`, outside of `getStaticProps` / `getStaticPaths` / `getServerSideProps`

Possible Ways to Fix It

The module you're trying to import is not installed in your dependencies

When importing a module from npm this module has to be installed locally.

For example when importing the `swr` package:

```
import useSWR from 'swr'
```

The `swr` module has to be installed using a package manager.

- When using `npm`: `npm install swr`
- When using `yarn`: `yarn add swr`

The module you're trying to import is in a different directory

Make sure that the path you're importing refers to the right directory and file.

The module you're trying to import has a different casing

Make sure the casing of the file is correct.

Example:

```
// components/MyComponent.js
export default function MyComponent() {
  return <h1>Hello</h1>
}
```

```
// pages/index.js
// Note how `components/MyComponent` exists but `Mycomponent` without the capital
`c` is imported
import MyComponent from '../components/Mycomponent'
```

Incorrect casing will lead to build failures on case-sensitive environments like most Linux-based continuous integration and can cause issues with Fast Refresh.

The module you're trying to import uses Node.js specific modules

`getStaticProps`, `getStaticPaths`, and `getServerSideProps` allow for using modules that can only run in the Node.js environment. This allows you to do direct database queries or reading data from Redis to name a few examples.

The tree shaking only runs on top level pages, so it can't be relied on in separate React components.

You can verify the tree shaking on next-code-elimination.vercel.app.

Example of correctly tree shaken code:

```
// lib/redis.js
import Redis from 'ioredis'

const redis = new Redis(process.env.REDIS_URL)

export default redis
```

```
// pages/index.js
import redis from '../lib/redis'

export async function getStaticProps() {
  const message = await redis.get('message')
  return {
    message,
  }
}

export default function Home({ message }) {
  return <h1>{message}</h1>
}
```

Example of code that would break:

```
// lib/redis.js
import Redis from 'ioredis'

const redis = new Redis(process.env.REDIS_URL)

export default redis
```

```
// pages/index.js
// Redis is a Node.js specific library that can't run in the browser
// Trying to use it in code that runs on both Node.js and the browser will result in
a module not found error for modules that ioredis relies on
// If you run into such an error it's recommended to move the code to
`getStaticProps` or `getServerSideProps` as those methods guarantee that the code is
only run in Node.js.
import redis from '../lib/redis'
import { useEffect, useState } from 'react'

export default function Home() {
  const [message, setMessage] = useState()
  useEffect(() => {
    redis.get('message').then((result) => {
      setMessage(result)
    })
  }, [])
  return <h1>{message}</h1>
}
```

Example of code that would break:

```
// lib/redis.js
import Redis from 'ioredis'

// Modules that hold Node.js-only code can't also export React components
// Tree shaking of getStaticProps/getStaticPaths/getServerSideProps is ran only on page files
const redis = new Redis(process.env.REDIS_URL)

export function MyComponent() {
  return <h1>Hello</h1>
}

export default redis
```

```
// pages/index.js
// In practice you'll want to refactor the `MyComponent` to be a separate file so that tree shaking ensures that specific import is not included for the browser compilation
import redis, { MyComponent } from '../lib/redis'

export async function getStaticProps() {
  const message = await redis.get('message')
  return {
    message,
  }
}

export default function Home() {
  return <MyComponent />
}
```