getServerSideProps

Version History

Version	Changes
`v10.0.0`	`locale`, `locales`, `defaultLocale`, and `notFound` options added.
`v9.3.0`	`getServerSideProps` introduced.

When exporting a function called `getServerSideProps` (Server-Side Rendering) from a page, Next.js will pre-render this page on each request using the data returned by `getServerSideProps`. This is useful if you want to fetch data that changes often, and have the page update to show the most current data.

```
export async function getServerSideProps(context) {
  return {
    props: {}, // will be passed to the page component as props
  }
}
```

You can import modules in top-level scope for use in `getServerSideProps`. Imports used will **not be bundled for the client-side**. This means you can write **server-side code directly in `getServerSideProps`**, including fetching data from your database.

Context parameter

The `context` parameter is an object containing the following keys:

- `params`: If this page uses a dynamic route, `params` contains the route parameters. If the page name is `[id].js` , then `params` will look like `{ id: ... }`.
- `req`: The `HTTP` IncomingMessage object.
- `res`: The `HTTP` response object.
- `query`: An object representing the query string.
- `preview`: `preview` is `true` if the page is in the Preview Mode and `false` otherwise.
- `previewData`: The preview data set by `setPreviewData`.
- `resolvedUrl`: A normalized version of the request `URL` that strips the `_next/data` prefix for client transitions and includes original query values.
- `locale` contains the active locale (if enabled).
- `locales` contains all supported locales (if enabled).
- `defaultLocale` contains the configured default locale (if enabled).

getServerSideProps return values

The `getServerSideProps` function should return an object with **any one of the following** properties:

`props`

The `props` object is a key-value pair, where each value is received by the page component. It should be a serializable object so that any props passed, could be serialized with `JSON.stringify`.

```
export async function getServerSideProps(context) {
  return {
    props: { message: `Next.js is awesome` }, // will be passed to the page component as pro
  }
}
```

`notFound`

The `notFound` boolean allows the page to return a `404` status and 404 Page. With `notFound: true`, the page will return a `404` even if there was a successfully generated page before. This is meant to support use cases like user-generated content getting removed by its author.

```
export async function getServerSideProps(context) {
  const res = await fetch(`https://.../data`)
  const data = await res.json()

  if (!data) {
    return {
      notFound: true,
    }
  }

  return {
    props: { data }, // will be passed to the page component as props
  }
}
```

`redirect`

The `redirect` object allows redirecting to internal and external resources. It should match the shape of `{ destination: string, permanent: boolean }`. In some rare cases, you might need to assign a custom status code for older `HTTP` clients to properly redirect. In these cases, you can use the `statusCode` property instead of the `permanent` property, but not both.

```
export async function getServerSideProps(context) {
  const res = await fetch(`https://.../data`)
  const data = await res.json()

  if (!data) {
    return {
```

```
      redirect: {
        destination: '/',
        permanent: false,
      },
    }
  }

  return {
    props: {}, // will be passed to the page component as props
  }
}
```

getServerSideProps with TypeScript

For TypeScript, you can use the `GetServerSideProps` type from `next`:

```
import { GetServerSideProps } from 'next'

export const getServerSideProps: GetServerSideProps = async (context) => {
  // ...
}
```

If you want to get inferred typings for your props, you can use `InferGetServerSidePropsType<typeof getServerSideProps>`:

```
import { InferGetServerSidePropsType } from 'next'

type Data = { ... }

export const getServerSideProps = async () => {
  const res = await fetch('https://.../data')
  const data: Data = await res.json()

  return {
    props: {
      data,
    },
  }
}

function Page({ data }: InferGetServerSidePropsType<typeof getServerSideProps>) {
  // will resolve posts to type Data
}

export default Page
```

Related

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

Data Fetching: Learn more about data fetching in Next.js.