getInitialProps

Recommended: `getStaticProps` or `getServerSideProps` instead of `getInitialProps`. These data fetching methods allow you to have a granular choice between static generation and server-side rendering.

`getInitialProps` enables server-side rendering in a page and allows you to do **initial data population**, it means sending the page with the data already populated from the server. This is especially useful for SEO.

`getInitialProps` will disable Automatic Static Optimization.

`getInitialProps` is an `async` function that can be added to any page as a `static method`. Take a look at the following example:

```
function Page({ stars }) {
  return <div>Next stars: {stars}</div>
}

Page.getInitialProps = async (ctx) => {
  const res = await fetch('https://api.github.com/repos/vercel/next.js')
  const json = await res.json()
  return { stars: json.stargazers_count }
}

export default Page
```

Or using a class component:

```
import React from 'react'

class Page extends React.Component {
  static async getInitialProps(ctx) {
    const res = await fetch('https://api.github.com/repos/vercel/next.js')
    const json = await res.json()
    return { stars: json.stargazers_count }
  }

  render() {
    return <div>Next stars: {this.props.stars}</div>
  }
}

export default Page
```

`getInitialProps` is used to asynchronously fetch some data, which then populates `props`.

Data returned from `getInitialProps` is serialized when server rendering, similar to what `JSON.stringify` does. Make sure the returned object from

`getInitialProps` is a plain `Object` and not using `Date`, `Map` or `Set`.

For the initial page load, `getInitialProps` will run on the server only. `getInitialProps` will then run on the client when navigating to a different route via the `next/link` component or by using `next/router`. However, if `getInitialProps` is used in a custom `_app.js`, and the page being navigated to implements `getServerSideProps`, then `getInitialProps` will run on the server.

Context Object

`getInitialProps` receives a single argument called `context`, it's an object with the following properties:

- `pathname` - Current route. That is the path of the page in `/pages`
- `query` - Query string section of URL parsed as an object
- `asPath` - `String` of the actual path (including the query) shown in the browser
- `req` - HTTP request object (server only)
- `res` - HTTP response object (server only)
- `err` - Error object if any error is encountered during the rendering

Caveats

- `getInitialProps` can **not** be used in children components, only in the default export of every page
- If you are using server-side only modules inside `getInitialProps`, make sure to import them properly, otherwise it'll slow down your app

TypeScript

If you're using TypeScript, you can use the `NextPage` type for function components:

```
import { NextPage } from 'next'

interface Props {
  userAgent?: string;
}

const Page: NextPage<Props> = ({ userAgent }) => (
  <main>Your user agent: {userAgent}</main>
)

Page.getInitialProps = async ({ req }) => {
  const userAgent = req ? req.headers['user-agent'] : navigator.userAgent
  return { userAgent }
```

```
}

export default Page
```

And for `React.Component`, you can use `NextPageContext`:

```
import React from 'react'
import { NextPageContext } from 'next'

interface Props {
  userAgent?: string;
}

export default class Page extends React.Component<Props> {
  static async getInitialProps({ req }: NextPageContext) {
    const userAgent = req ? req.headers['user-agent'] : navigator.userAgent
    return { userAgent }
  }

  render() {
    const { userAgent } = this.props
    return <main>Your user agent: {userAgent}</main>
  }
}
```

Related

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

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