Server-Side Rendering (SSR) allows you to render a page at run-time with data that is fetched when a user visits the page. The server generates the full HTML during HTTP request and sends it to the user. The API is focused on data fetching outside of the Gatsby data layer.

> ***Note:*** *This feature requires running NodeJS server. It is currently fully supported with* `gatsby serve` *and in Gatsby Cloud.*

Creating server-rendered pages

You can create server-rendered pages the same way as regular pages (including using the File System Route API).

The main difference is that page component must export an async function called `getServerData`:

```
export async function getServerData(context) {
  return {
    status: 200, // The HTTP status code that should be returned
    props: {}, // Will be passed to the page component as "serverData" prop
    headers: {}, // HTTP response headers for this page
  }
}
```

When a page component exports `getServerData` function, Gatsby treats all pages built with this component as server-rendered.

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

- `headers`: The request headers
- `method`: The request method, e.g. `GET`
- `url`: The request URL
- `query`: An object representing the query string
- `params`: If you use File System Route API the URL path gets passed in as `params`. For example, if your page is at `src/pages/{Product.name}.js` the `params` will be an object like `{ name: 'value' }`.

`getServerData` can return an object with several possible keys:

- `status` (optional): The HTTP status code that should be returned. Should be a valid HTTP status code.
- `props` (optional): Object containing the data passed to `serverData` page prop. Should be a serializable object.
- `headers` (optional): Object containing `headers` that are sent to the browser and caching proxies/CDNs (e.g., cache headers).

Use `serverData` prop in React page component

The `props` object you return from `getServerData` gets passed to the page component as `serverData` prop.

```
import * as React from "react"

const Page = ({ serverData }) => {
  const { dogImage } = serverData
```

```
  // Use dogImage in your page...
}

export async function getServerData() {
  const res = await fetch(`https://dog.ceo/api/breeds/image/random`)
  const data = await res.json()

  return {
    props: {
      dogImage: data,
    },
  }
}

export default Page
```

Interplay with build-time GraphQL queries

Server-rendered pages also support regular Gatsby GraphQL page queries. The page query is executed at build time, and the data is passed to React component as a `data` prop on each render (along with the `serverData` prop).

Keep in mind that `data` will be the same every time the page renders, but `serverData` will change according to return value of your `getServerData` function. Runtime GraphQL queries are not supported yet.

```
import * as React from "react"

const Page = ({ data, serverData }) => {
  const { site } = data
  const { dogImage } = serverData
  // Use dogImage and site info in your page...
}

export const pageQuery = graphql`
  query PageData {
    site {
      siteMetadata {
        title
      }
    }
  }
`

export async function getServerData() {
  const res = await fetch(`https://dog.ceo/api/breeds/image/random`)
  const data = await res.json()

  return {
    props: {
      dogImage: data,
    },
  }
```

```
}

export default Page
```

Working with server-rendered pages locally

Server-rendered pages work with both `gatsby develop` and `gatsby serve`. The page will be re-generated on each request.

Using in production

Server-Side Rendering requires a running NodeJS server. You can put NodeJS running `gatsby serve` behind a content delivery network (CDN) like Fastly, however that also requires additional infrastructure (like monitoring, logging, and crash-recovery).

Complete setup with auto-scaling is available for you in Gatsby Cloud out-of-the-box.

How it works

For SSR every request only runs on server-side. On Gatsby Cloud the request is sent to a worker process that runs your `getServerData` function, passes this data to React component and returns HTML back to the user. By default, every request is a cache miss and for caching you'll need to set custom HTTP Cache-Control headers.

When you directly visit a page you'll get served the HTML. If you request a page on client-side navigation through Gatsby's Link component the response will be JSON. Gatsby's router uses this to render the page on the client.

This all happens automatically and you'll only need to define a `getServerData` function in your page. You can't export it from non-page files.

Additional Resources

- How-To Guide: Server-Side Rendering
- Conceptual Guide: Rendering Options