Preview Mode (Legacy)

Warning: This feature is **legacy** and is superseded by Draft Mode.

▶ **Examples**

In the Pages documentation and the Data Fetching documentation, we talked about how to pre-render a page at build time (**Static Generation**) using `getStaticProps` and `getStaticPaths`.

Static Generation is useful when your pages fetch data from a headless CMS. However, it's not ideal when you're writing a draft on your headless CMS and want to **preview** the draft immediately on your page. You'd want Next.js to render these pages at **request time** instead of build time and fetch the draft content instead of the published content. You'd want Next.js to bypass Static Generation only for this specific case.

Next.js has a feature called **Preview Mode** which solves this problem. Here are instructions on how to use it.

Step 1: Create and access a preview API route

Take a look at the API Routes documentation first if you're not familiar with Next.js API Routes.

First, create a **preview API route**. It can have any name - e.g. `pages/api/preview.js` (or `.ts` if using TypeScript).

In this API route, you need to call `setPreviewData` on the response object. The argument for `setPreviewData` should be an object, and this can be used by `getStaticProps` (more on this later). For now, we'll use `{}`.

```
export default function handler(req, res) {
  // ...
  res.setPreviewData({});
  // ...
```

```
}
```

`res.setPreviewData` sets some **cookies** on the browser which turns on the preview mode. Any requests to Next.js containing these cookies will be considered as the **preview mode**, and the behavior for statically generated pages will change (more on this later).

You can test this manually by creating an API route like below and accessing it from your browser manually:

pages/api/preview.js

```
// simple example for testing it manually from your browser.
export default function handler(req, res) {
  res.setPreviewData({});
  res.end('Preview mode enabled');
}
```

If you open your browser's developer tools and visit `/api/preview`, you'll notice that the `__prerender_bypass` and `__next_preview_data` cookies will be set on this request.

Securely accessing it from your Headless CMS

In practice, you'd want to call this API route *securely* from your headless CMS. The specific steps will vary depending on which headless CMS you're using, but here are some common steps you could take.

These steps assume that the headless CMS you're using supports setting **custom preview URLs**. If it doesn't, you can still use this method to secure your preview URLs, but you'll need to construct and access the preview URL manually.

First, you should create a **secret token string** using a token generator of your choice. This secret will only be known by your Next.js app and your headless CMS. This secret prevents people who don't have access to your CMS from accessing preview URLs.

Second, if your headless CMS supports setting custom preview URLs, specify the following as the preview URL. This assumes that your preview API route is located at `pages/api/preview.js`.

Terminal

```
https://<your-site>/api/preview?secret=<token>&slug=<path>
```

- `<your-site>` should be your deployment domain.
- `<token>` should be replaced with the secret token you generated.

- `<path>` should be the path for the page that you want to preview. If you want to preview `/posts/foo`, then you should use `&slug=/posts/foo`.

Your headless CMS might allow you to include a variable in the preview URL so that `<path>` can be set dynamically based on the CMS's data like so: `&slug=/posts/{entry.fields.slug}`

Finally, in the preview API route:

- Check that the secret matches and that the `slug` parameter exists (if not, the request should fail).
- Call `res.setPreviewData`.
- Then redirect the browser to the path specified by `slug`. (The following example uses a 307 redirect↗).

```
export default async (req, res) => {
  // Check the secret and next parameters
  // This secret should only be known to this API route and the CMS
  if (req.query.secret !== 'MY_SECRET_TOKEN' || !req.query.slug) {
    return res.status(401).json({ message: 'Invalid token' });
  }

  // Fetch the headless CMS to check if the provided `slug` exists
  // getPostBySlug would implement the required fetching logic to the headless CMS
  const post = await getPostBySlug(req.query.slug);

  // If the slug doesn't exist prevent preview mode from being enabled
  if (!post) {
    return res.status(401).json({ message: 'Invalid slug' });
  }

  // Enable Preview Mode by setting the cookies
  res.setPreviewData({});

  // Redirect to the path from the fetched post
  // We don't redirect to req.query.slug as that might lead to open redirect vulnerabilit
  res.redirect(post.slug);
};
```

If it succeeds, then the browser will be redirected to the path you want to preview with the preview mode cookies being set.

Step 2: Update `getStaticProps`

The next step is to update `getStaticProps` to support the preview mode.

If you request a page which has `getStaticProps` with the preview mode cookies set (via `res.setPreviewData`), then `getStaticProps` will be called at **request time** (instead of at build time).

Furthermore, it will be called with a `context` object where:

- `context.preview` will be `true`.
- `context.previewData` will be the same as the argument used for `setPreviewData`.

```
export async function getStaticProps(context) {
  // If you request this page with the preview mode cookies set:
  //
  // - context.preview will be true
  // - context.previewData will be the same as
  //   the argument used for `setPreviewData`.
}
```

We used `res.setPreviewData({})` in the preview API route, so `context.previewData` will be `{}`. You can use this to pass session information from the preview API route to `getStaticProps` if necessary.

If you're also using `getStaticPaths`, then `context.params` will also be available.

Fetch preview data

You can update `getStaticProps` to fetch different data based on `context.preview` and/or `context.previewData`.

For example, your headless CMS might have a different API endpoint for draft posts. If so, you can use `context.preview` to modify the API endpoint URL like below:

```
export async function getStaticProps(context) {
  // If context.preview is true, append "/preview" to the API endpoint
  // to request draft data instead of published data. This will vary
  // based on which headless CMS you're using.
  const res = await fetch(`https://.../${context.preview ? 'preview' : ''}`);
  // ...
}
```

That's it! If you access the preview API route (with `secret` and `slug`) from your headless CMS or manually, you should now be able to see the preview content. And if you update your draft without publishing, you should be able to preview the draft.

Set this as the preview URL on your headless CMS or access manually, and you should be able to see the preview.

Terminal

```
https://<your-site>/api/preview?secret=<token>&slug=<path>
```

More Details

Note: during rendering `next/router` exposes an `isPreview` flag, see the router object docs for more info.

Specify the Preview Mode duration

`setPreviewData` takes an optional second parameter which should be an options object. It accepts the following keys:

- `maxAge`: Specifies the number (in seconds) for the preview session to last for.
- `path`: Specifies the path the cookie should be applied under. Defaults to `/` enabling preview mode for all paths.

```
setPreviewData(data, {
  maxAge: 60 * 60, // The preview mode cookies expire in 1 hour
  path: '/about', // The preview mode cookies apply to paths with /about
});
```

Clear the Preview Mode cookies

By default, no expiration date is set for Preview Mode cookies, so the preview session ends when the browser is closed.

To clear the Preview Mode cookies manually, create an API route that calls `clearPreviewData()`:

pages/api/clear-preview-mode-cookies.js

```
export default function handler(req, res) {
  res.clearPreviewData({});
}
```

Then, send a request to `/api/clear-preview-mode-cookies` to invoke the API Route. If calling this route using `next/link`, you must pass `prefetch={false}` to prevent calling `clearPreviewData` during link

prefetching.

If a path was specified in the `setPreviewData` call, you must pass the same path to `clearPreviewData`:

pages/api/clear-preview-mode-cookies.js

```
export default function handler(req, res) {
  const { path } = req.query;

  res.clearPreviewData({ path });
}
```

`previewData` size limits

You can pass an object to `setPreviewData` and have it be available in `getStaticProps`. However, because the data will be stored in a cookie, there's a size limitation. Currently, preview data is limited to 2KB.

Works with `getServerSideProps`

The preview mode works on `getServerSideProps` as well. It will also be available on the `context` object containing `preview` and `previewData`.

Works with API Routes

API Routes will have access to `preview` and `previewData` under the request object. For example:

```
export default function myApiRoute(req, res) {
  const isPreview = req.preview;
  const previewData = req.previewData;
  // ...
}
```

Unique per `next build`

Both the bypass cookie value and the private key for encrypting the `previewData` change when `next build` is completed. This ensures that the bypass cookie can't be guessed.

Note: To test Preview Mode locally over HTTP your browser will need to allow third-party cookies and local storage access.