Draft Mode

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 view 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 **Draft Mode** which solves this problem. Here are instructions on how to use it.

Step 1: Create and access the API route

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

First, create the **API route**. It can have any name - e.g. `pages/api/draft.ts`

In this API route, you need to call `setDraftMode` on the response object.

```
export default function handler(req, res) {
  // ...
  res.setDraftMode({ enable: true });
  // ...
}
```

This will set a **cookie** to enable draft mode. Subsequent requests containing this cookie will trigger **Draft Mode** changing the behavior for statically generated pages (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/draft.ts

```ts
// simple example for testing it manually from your browser.
export default function handler(req, res) {
  res.setDraftMode({ enable: true });
  res.end('Draft mode is enabled');
}
```

If you open your browser's developer tools and visit `/api/draft`, you'll notice a `Set-Cookie` response header with a cookie named `__prerender_bypass`.

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 draft URLs**. If it doesn't, you can still use this method to secure your draft URLs, but you'll need to construct and access the draft 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 draft URLs.

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

Terminal

```
https://<your-site>/api/draft?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 view. If you want to view `/posts/foo`, then you should use `&slug=/posts/foo`.

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

Finally, in the draft API route:

- Check that the secret matches and that the `slug` parameter exists (if not, the request should fail).

- Call `res.setDraftMode`.
- 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 draft mode from being enabled
  if (!post) {
    return res.status(401).json({ message: 'Invalid slug' });
  }

  // Enable Draft Mode by setting the cookie
  res.setDraftMode({ enable: true });

  // 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 view with the draft mode cookie.

Step 2: Update `getStaticProps`

The next step is to update `getStaticProps` to support draft mode.

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

Furthermore, it will be called with a `context` object where `context.draftMode` will be `true`.

```
export async function getStaticProps(context) {
  if (context.draftMode) {
    // dynamic data
  }
}
```

We used `res.setDraftMode` in the draft API route, so `context.draftMode` will be `true`.

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

Fetch draft data

You can update `getStaticProps` to fetch different data based on `context.draftMode`.

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

```
export async function getStaticProps(context) {
  const url = context.draftMode
    ? 'https://draft.example.com'
    : 'https://production.example.com';
  const res = await fetch(url);
  // ...
}
```

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

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

Terminal

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

More Details

Clear the Draft Mode cookie

By default, the Draft Mode session ends when the browser is closed.

To clear the Draft Mode cookie manually, create an API route that calls `setDraftMode({ enable: false })`:

pages/api/disable-draft.ts

```
export default function handler(req, res) {
  res.setDraftMode({ enable: false });
}
```

Then, send a request to `/api/disable-draft` to invoke the API Route. If calling this route using `next/link`, you must pass `prefetch={false}` to prevent accidentally deleting the cookie on prefetch.

Works with `getServerSideProps`

Draft Mode works with `getServerSideProps` as well. It will also be available on the `context` object containing `draftMode`

Works with API Routes

API Routes will have access to `draftMode` on the request object. For example:

```
export default function myApiRoute(req, res) {
  if (req.draftMode) {
    // get draft data
  }
}
```

Unique per `next build`

A new bypass cookie value will be generated each time you run `next build`.

This ensures that the bypass cookie can't be guessed.

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