Automatic Static Optimization

Next.js automatically determines that a page is static (can be prerendered) if it has no blocking data requirements. This determination is made by the absence of `getServerSideProps` and `getInitialProps` in the page.

This feature allows Next.js to emit hybrid applications that contain **both server-rendered and statically generated pages**.

> Statically generated pages are still reactive: Next.js will hydrate your application client-side to give it full interactivity.

One of the main benefits of this feature is that optimized pages require no server-side computation, and can be instantly streamed to the end-user from multiple CDN locations. The result is an *ultra fast* loading experience for your users.

How it works

If `getServerSideProps` or `getInitialProps` is present in a page, Next.js will switch to render the page on-demand, per-request (meaning Server-Side Rendering).

If the above is not the case, Next.js will **statically optimize** your page automatically by prerendering the page to static HTML.

During prerendering, the router's `query` object will be empty since we do not have `query` information to provide during this phase. After hydration, Next.js will trigger an update to your application to provide the route parameters in the `query` object.

The cases where the query will be updated after hydration triggering another render are:

- The page is a dynamic-route.

- The page has query values in the URL.
- Rewrites are configured in your `next.config.js` since these can have parameters that may need to be parsed and provided in the `query`.

To be able to distinguish if the query is fully updated and ready for use, you can leverage the `isReady` field on `next/router`.

> **Note**: Parameters added with dynamic routes to a page that's using `getStaticProps` will always be available inside the `query` object.

`next build` will emit `.html` files for statically optimized pages. For example, the result for the page `pages/about.js` would be:

```
Terminal
.next/server/pages/about.html
```

And if you add `getServerSideProps` to the page, it will then be JavaScript, like so:

```
Terminal
.next/server/pages/about.js
```

Caveats

- If you have a custom `App` with `getInitialProps` then this optimization will be turned off in pages without Static Generation.
- If you have a custom `Document` with `getInitialProps` be sure you check if `ctx.req` is defined before assuming the page is server-side rendered. `ctx.req` will be `undefined` for pages that are prerendered.
- Avoid using the `asPath` value on `next/router` in the rendering tree until the router's `isReady` field is `true`. Statically optimized pages only know `asPath` on the client and not the server, so using it as a prop may lead to mismatch errors. The `active-class-name` example ↗ demonstrates one way to use `asPath` as a prop.