TypeScript

import { PackageManagers, Render } from "~/components";

Pages Functions supports TypeScript. Author any files in your `/functions` directory with a `.ts` extension instead of a `.js` extension to start using TypeScript.

You can add runtime types and Env types by running:

<PackageManagers type="exec" pkg="wrangler" args={"types --path='./functions/types.d.ts'"} />

Then configure the types by creating a `functions/tsconfig.json` file:

```
{
  "compilerOptions": {
    "target": "esnext",
    "module": "esnext",
    "lib": ["esnext"],
    "types": ["./types.d.ts"]
  }
}
```

See the `wrangler types` command docs for more details.

If you already have a `tsconfig.json` at the root of your project, you may wish to explicitly exclude the `/functions` directory to avoid conflicts. To exclude the `/functions` directory:

```
{
  "include": ["src/**/*"],
  "exclude": ["functions/**/*"],
  "compilerOptions": {}
}
```

Pages Functions can be typed using the `PagesFunction` type. This type accepts an `Env` parameter. The `Env` type should have been generated by `wrangler types` and can be found at the top of `types.d.ts`.

Alternatively, you can define the `Env` type manually. For example:

```
interface Env {
□KV: KVNamespace;
}

export const onRequest: PagesFunction<Env> = async (context) => {
□const value = await context.env.KV.get("example");
□return new Response(value);
};
```

If you are using `nodejs_compat`, make sure you have installed `@types/node` and updated your `tsconfig.json`.

```
{
□"compilerOptions": {
□□"target": "esnext",
□□"module": "esnext",
□□"lib": ["esnext"],
□□"types": ["./types.d.ts", "node"]
□}
}
```

:::note

If you were previously using `@cloudflare/workers-types` instead of the runtime types generated by `wrangler types`, you can refer to this migration guide.

:::

□□□ #2
□ □□□ □□□ 2026-11-01 03:02:59 UTC
□ □□□ □□□ 2026-11-01 03:03:00 UTC