SVELTEKIT • REFERENCE

Types

ON THIS PAGE

Generated types

The `RequestHandler` and `Load` types both accept a `Params` argument allowing you to type the `params` object. For example this endpoint expects `foo`, `bar` and `baz` params:

src/routes/[foo]/[bar]/[baz]/+page.server.ts

JS TS

```ts
import type { RequestHandler } from '@sveltejs/kit';
export const GET: RequestHandler<{
  foo: string;
  bar: string;
  baz: string
  }> = async ({ params }) => {
  // ...
};
```

Needless to say, this is cumbersome to write out, and less portable (if you were to rename the `[foo]` directory to `[qux]`, the type would no longer reflect reality).

To solve this problem, SvelteKit generates `.d.ts` files for each of your endpoints and pages:

.svelte-kit/types/src/routes/[foo]/[bar]/[baz]/$types.d

```ts
import type * as Kit from '@sveltejs/kit';

type RouteParams = {
  foo: string;
  bar: string;
  baz: string;
};
```

These files can be imported into your endpoints and pages as siblings, thanks to the `rootDirs` option in your TypeScript configuration:

src/routes/[foo]/[bar]/[baz]/+page.server.ts

JS TS

```ts
import type { PageServerLoad } from './$types';

export const GET: PageServerLoad = async ({ params }) => {
  // ...
};
```

src/routes/[foo]/[bar]/[baz]/+page.ts

JS TS

```ts
import type { PageLoad } from './$types';

export const load: PageLoad = async ({ params, fetch }) => {
  // ...
};
```

> For this to work, your own `tsconfig.json` or `jsconfig.json` should extend from the generated `.svelte-kit/tsconfig.json` (where `.svelte-kit` is your `outDir`):
>
> ```
> { "extends": "./.svelte-kit/tsconfig.json" }
> ```

Default tsconfig.json

The generated `.svelte-kit/tsconfig.json` file contains a mixture of options. Some are generated programmatically based on your project configuration, and should generally not be overridden without good reason:

.svelte-kit/tsconfig.json

```
{
  "compilerOptions": {
    "baseUrl": "..",
    "paths": {
      "$lib": "src/lib",
```

```
    "rootDirs": [".", "./types"]
  },
  "include": ["../src/**/*.js", "../src/**/*.ts", "../src/**/*.svelte"],
  "exclude": ["../node_modules/**", "./**"]
}
```

Others are required for SvelteKit to work properly, and should also be left untouched unless you know what you're doing:

.svelte-kit/tsconfig.json

```
{
  "compilerOptions": {
    // this ensures that types are explicitly
    // imported with `import type`, which is
    // necessary as svelte-preprocess cannot
    // otherwise compile components correctly
    "importsNotUsedAsValues": "error",

    // Vite compiles one TypeScript module
    // at a time, rather than compiling
    // the entire module graph
    "isolatedModules": true,

    // TypeScript cannot 'see' when you
    // use an imported value in your
    // markup, so we need this
    "preserveValueImports": true,

    // This ensures both `vite build`
    // and `svelte-package` work correctly
    "lib": ["esnext", "DOM", "DOM.Iterable"],
    "moduleResolution": "node",
    "module": "esnext",
    "target": "esnext"
  }
}
```

$lib

without `../../../../` nonsense.

$lib/server

A subdirectory of `$lib`. SvelteKit will prevent you from importing any modules in `$lib/server` into client-side code. See server-only modules.

app.d.ts

The `app.d.ts` file is home to the ambient types of your apps, i.e. types that are available without explicitly importing them.

Always part of this file is the `App` namespace. This namespace contains several types that influence the shape of certain SvelteKit features you interact with.

Error

Defines the common shape of expected and unexpected errors. Expected errors are thrown using the `error` function. Unexpected errors are handled by the `handleError` hooks which should return this shape.

```
interface Error {…}
```

```
message: string;
```

Locals

The interface that defines `event.locals`, which can be accessed in server hooks (`handle`, and `handleError`), server-only `load` functions, and `+server.js` files.

PageData

Defines the common shape of the $page.data store - that is, the data that is shared between all pages. The `Load` and `ServerLoad` functions in `./$types` will be narrowed accordingly. Use optional properties for data that is only present on specific pages. Do not add an index signature (`[key: string]: any`).

```
interface PageData {}
```

PageState

The shape of the `$page.state` object, which can be manipulated using the `pushState` and `replaceState` functions from `$app/navigation` .

```
interface PageState {}
```

Platform

If your adapter provides platform-specific context via `event.platform` , you can specify it here.

```
interface Platform {}
```

Edit this page on GitHub

PREVIOUS
Command Line Interface

NEXT

Docs