import { defineConfig } from 'vite'

The line `import { defineConfig } from 'vite'` is a standard import statement used in JavaScript and TypeScript projects that utilize Vite as their build tool.

Purpose:

- Importing `defineConfig`: This statement imports the `defineConfig` helper function from the `vite` package.
- **Providing Intellisense and Type Checking:** The primary purpose of `defineConfig` is to provide type hints and better Intellisense support within your IDE when configuring Vite. By wrapping your Vite configuration object with `defineConfig`, you benefit from autocompletion and type checking, which helps prevent errors and ensures your configuration adheres to Vite's expected structure.
- **Defining Vite Configuration:** This function is typically used in the `vite.config.js` (or `vite.config.ts`) file, which is the central place to define your project's build and development settings for Vite.

Example Usage:

```
// vite.config.js
import { defineConfig } from 'vite';

export default defineConfig({
  plugins: [
    // your Vite plugins here, e.g., @vitejs/plugin-react
  ],
  server: {
    port: 3000,
  },
  build: {
    outDir: 'dist',
  },
});
```

In this example, the configuration object passed to `defineConfig` specifies various Vite settings, including plugins, server options, and build options. The `defineConfig` wrapper helps ensure that these settings are correctly typed and recognized by your app.

Key Parameters in `defineConfig`

- `plugins`: Array of Vite plugins (e.g., React, Vue, TS Paths) to add functionality.
- `server`: Options for the development server (e.g., `port`, `host`, `open`, `proxy`).
- `build`: Settings for the production build (e.g., `outDir`, `minify`, `target`).
- `resolve`: Path aliases (e.g., `@/src`) and module resolution settings.
- `define`: Global constants (e.g., `__APP_VERSION__`) that get replaced during build.
- `envDir` / `envPrefix`: Control where `.env` files are loaded and which variables are exposed.
- `base`: Public path for assets (e.g., `/my-app/`).

Advanced Usage

- **Conditional Config**: Export a function from `defineConfig` to set options based on `command` (dev/build) or `mode` (development/production).
- **Rollup Options**: Use `build.rollupOptions` to deeply customize Rollup (e.g., `output.entryFileNames`, `external` modules)

development environment.

Revision #2
Created 29 October 2025 02:43:39 by AI API
Updated 9 December 2025 11:44:13 by AI Channel