`import.meta.env` is a feature provided by Vite (and some other modern JavaScript bundlers) that allows you to access environment variables in your application. Here's a breakdown of what it does and how it works:

### What `import.meta.env` Does:

1. \*\*Access Environment Variables\*\*: It provides a way to access environment variables that are defined in your Vite project. These variables are typically defined in a `.env` file or passed through the command line or CI/CD environment.

2. \*\*Type Safety\*\*: When using TypeScript, you can define the shape of `import.meta.env` in a `.d.ts` file (like `vite-env.d.ts`) to ensure type safety. This helps catch errors at compile time if you try to access a variable that doesn't exist or is of the wrong type.

3. \*\*Build-Time Replacement\*\*: Vite replaces `import.meta.env` with the actual values of the environment variables at build time. This means that the environment variables are embedded directly into your built JavaScript code.

### Example Usage:

In your `dataFetcher.ts` file, you are using `config.apiUrl`, which is set up in your `env.ts` file using `import.meta.env.VITE\_POKEMON\_API\_URL`. Here's how it works:

- \*\*Environment Variable Definition\*\*: You define `VITE\_POKEMON\_API\_URL` in a `.env` file or through other means.

Example `.env` file:

```

VITE\_POKEMON\_API\_URL=https://pokeapi.co/api/v2

```

- \*\*Accessing the Variable\*\*: In your `env.ts` file, you access this variable using `import.meta.env.VITE\_POKEMON\_API\_URL`.

- \*\*Usage in Code\*\*: The `config` object in `env.ts` is then used in `dataFetcher.ts` to construct API URLs.

### Benefits:

- \*\*Security\*\*: By using environment variables, you can keep sensitive information like API keys out of your source code.

- \*\*Flexibility\*\*: You can easily change configuration values without modifying your code, just by changing the environment variables.

- \*\*Environment-Specific Configurations\*\*: You can have different configurations for development, testing, and production environments.

Overall, `import.meta.env` is a powerful feature for managing environment-specific configurations in a Vite project.

`.d.ts` files are TypeScript declaration files. They are used to provide type information about JavaScript code, allowing TypeScript to understand the types of variables, functions, and other entities in a JavaScript library or module. This is particularly useful when you want to use JavaScript libraries in a TypeScript project, as it enables type checking and IntelliSense in your editor.

### Purpose of `.d.ts` Files:

1. \*\*Type Definitions\*\*: They define the types of variables, functions, classes, and other entities in a module or library.
2. \*\*IntelliSense\*\*: They provide better code completion and documentation in editors that support TypeScript.
3. \*\*Type Checking\*\*: They enable TypeScript to perform type checking on JavaScript libraries, catching potential errors at compile time.
4. \*\*Documentation\*\*: They serve as a form of documentation for the API of a library or module.

### Specific Example from Your File:

In your vite-env.d.ts file, you have:

/// <reference types="vite/client" />

interface ImportMetaEnv {

readonly VITE\_POKEMON\_API\_URL: string;

}

interface ImportMeta {

readonly env: ImportMetaEnv;

}

* /// <reference types="vite/client" />: This is a triple-slash directive that includes type definitions from the vite/client module. It allows TypeScript to understand the types provided by Vite's client-side environment.
* interface ImportMetaEnv: This interface defines the shape of the env object within ImportMeta. It specifies that VITE\_POKEMON\_API\_URL is a string and is read-only.
* interface ImportMeta: This extends the ImportMeta interface to include the env property, which is of type ImportMetaEnv. This allows you to access environment variables in a type-safe manner using import.meta.env.VITE\_POKEMON\_API\_URL.

Overall, this .d.ts file is used to provide type definitions for environment variables in a Vite project, ensuring that you can access them safely and with type checking in your TypeScript code.

The vite-env.d.ts file is not explicitly referenced in the Vite configuration file (vite.config.js or vite.config.ts). Instead, it is automatically picked up by TypeScript when you run your Vite project. Here's how it works:

TypeScript Configuration: The vite-env.d.ts file is typically included in your TypeScript project by being located in the src directory or any directory that is included in your tsconfig.json file's include array. TypeScript automatically includes all .d.ts files it finds in the included directories.

Global Type Definitions: The vite-env.d.ts file provides global type definitions for Vite-specific features, such as import.meta.env. This allows you to use these types throughout your project without needing to import them explicitly.

Environment Variables: The types defined in vite-env.d.ts are used to provide type safety for environment variables accessed via import.meta.env. This is particularly useful for ensuring that you are accessing environment variables correctly and that they exist.

Example tsconfig.json:

Here's an example of how your tsconfig.json might look to ensure that TypeScript picks up the vite-env.d.ts file:

{

"compilerOptions": {

"target": "esnext",

"module": "esnext",

"moduleResolution": "node",

"strict": true,

"jsx": "react",

"esModuleInterop": true,

"skipLibCheck": true,

"forceConsistentCasingInFileNames": true

},

"include": ["src"]

}

In this configuration, the include array specifies that all files in the src directory should be included, which would encompass your vite-env.d.ts file. This setup ensures that TypeScript is aware of the type definitions provided in that file.

### import.meta.env

import.meta.env is a feature provided by Vite (and some other modern JavaScript bundlers) that allows you to access environment variables in your application. Here's a breakdown of what it does and how it works:

What import.meta.env Does:

1. Access Environment Variables: It provides a way to access environment variables that are defined in your Vite project. These variables are typically defined in a .env file or passed through the command line or CI/CD environment.

Type Safety: When using TypeScript, you can define the shape of import.meta.env in a .d.ts file (like vite-env.d.ts) to ensure type safety. This helps catch errors at compile time if you try to access a variable that doesn't exist or is of the wrong type.

Build-Time Replacement: Vite replaces import.meta.env with the actual values of the environment variables at build time. This means that the environment variables are embedded directly into your built JavaScript code.

Example Usage:

In your dataFetcher.ts file, you are using config.apiUrl, which is set up in your env.ts file using import.meta.env.VITE\_POKEMON\_API\_URL. Here's how it works:

Environment Variable Definition: You define VITE\_POKEMON\_API\_URL in a .env file or through other means.

Example .env file:

VITE\_POKEMON\_API\_URL=https://pokeapi.co/api/v2

Accessing the Variable: In your env.ts file, you access this variable using import.meta.env.VITE\_POKEMON\_API\_URL.

Usage in Code: The config object in env.ts is then used in dataFetcher.ts to construct API URLs.

Benefits:

Security: By using environment variables, you can keep sensitive information like API keys out of your source code.

Flexibility: You can easily change configuration values without modifying your code, just by changing the environment variables.

Environment-Specific Configurations: You can have different configurations for development, testing, and production environments.

Overall, import.meta.env is a powerful feature for managing environment-specific configurations in a Vite project.

Vite automatically looks for .env files in your project's root directory (where your package.json is located). This is built into Vite's configuration system and follows a specific loading order:

Loading Order:

your-project/

├── src/

│ └── ...

├── .env # base env vars

├── .env.local # local overrides (git ignored)

├── .env.development # development mode vars

├── .env.production # production mode vars

├── package.json

└── vite.config.ts

Important Notes:

Only variables prefixed with VITE\_ are exposed to your Vite-processed code:

# .env

VITE\_API\_URL=https://api.example.com # ✅ Available as import.meta.env.VITE\_API\_URL

SECRET\_KEY=123456 # ❌ Not exposed to your code

2. You can customize this behavior in your vite.config.ts:

export default defineConfig({

envDir: './config', // Change env files directory

envPrefix: 'APP\_' // Change env variables prefix from VITE\_ to APP\_

})

This automatic loading is one of Vite's built-in features, making environment configuration straightforward without requiring additional setup.