vite package different components

Vite's "library mode" is the primary method for packaging different components or a component library for distribution as an NPM package. This allows you to build your components into optimized bundles that can be easily consumed by other projects.
Here's a breakdown of the key steps and concepts involved:

- **Project Setup:**
  - Initialize a new Vite project or use an existing one.
  - Ensure your project structure clearly separates your library components (e.g., in a `lib` folder) from any demo or development-only code (e.g., in a `src` folder).

- **Vite Configuration (`vite.config.js`):**
  - `build.lib`: This is the core of library mode. You define the entry point of your library and the output formats.
    - `entry`: Path to your main library file (e.g., `lib/main.js` or `lib/index.ts`).
    - `name`: The global variable name for your library when used in UMD format.
    - `fileName`: The name of the output bundle file(s). You can use a function to customize file names based on format and entry name.
  - `build.rollupOptions`: Customize Rollup's behavior (the underlying bundler Vite uses).
    - `external`: Define dependencies that should not be bundled into your library (e.g., `react`, `vue`). These will be treated as external dependencies that the consuming project needs to provide.
    - `output.globals`: Map external dependencies to their global variable names for UMD builds.
  -
- **Component Implementation:**
  - Create your components within your designated library folder.
  - Use standard ES module `export` statements to make your components available for import.

- **Type Definitions (Optional but Recommended):**
  - If you're using TypeScript, generate type definitions (`.d.ts` files) for your library. Plugins like `vite-plugin-dts` can automate this process.

- **Building the Library:**

    - Run the Vite build command, which will generate the bundled output in your `dist` folder according to your `vite.config.js` settings.

- **Publishing (for NPM Packages):**
    - Ensure your `package.json` correctly defines your package name, version, main entry points (e.g., `main`, `module`, `types`), and any necessary dependencies.
    - Publish your package to NPM.

Example `vite.config.js` snippet for a component library:

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';

export default defineConfig({
  plugins: [react()],
  build: {
    lib: {
      entry: 'lib/main.js', // Your library's entry point
      name: 'MyComponentLibrary', // Global variable name for UMD
      fileName: (format) => `my-component-library.${format}.js`,
    },
    rollupOptions: {
      external: ['react', 'react-dom'], // Exclude React from the bundle
      output: {
        globals: {
          react: 'React',
          'react-dom': 'ReactDOM',
        },
      },
    },
  },
});
```

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