gatsby-plugin-typescript

Allows Gatsby to build TypeScript and TSX files. Does NOT run type checking during build (see Caveats).

This plugin is automatically included in Gatsby. The only reason you would need to explicitly use this plugin is if you need to configure its options.

How to customize usage

1. Include the plugin in your `gatsby-config.js` file with the specific options
2. Write your components in TSX or TypeScript.
3. Run TypeScript directly or with a build tool.
4. You're good to go.

   When creating pages programmatically, you can pass the `.tsx` filename directly as the `component` for `createPage`.

***Please note**: If packages don't ship with TypeScript definitions you'll need to manually install those type definitions, e.g. for React. A typical Gatsby project would need: `npm install --save-dev @types/react @types/react-dom @types/node`*

Options

When adding this plugin to your `gatsby-config.js`, you can pass in options to override the default @babel/preset-typescript config.

```
// gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-plugin-typescript`,
      options: {
        isTSX: true, // defaults to false
        jsxPragma: `jsx`, // defaults to "React"
        allExtensions: true, // defaults to false
      },
    },
  ],
}
```

For more detailed documentation on the available options, visit https://babeljs.io/docs/en/babel-preset-typescript#options. To add TypeScript Babel plugins (e.g. `@babel/plugin-proposal-decorators`), you can try using a custom .babelrc file.

Caveats

This plugin uses `babel-plugin-transform-typescript` to transpile TypeScript. It does *not do type checking*. Also since the TypeScript compiler is not involved, the following applies:

> Does not support namespaces. Workaround: Move to using file exports, or migrate to using the module { } syntax instead.
>
> Does not support const enums because those require type information to compile. Workaround: Remove the const, which makes it available at runtime.
>
> Does not support export = and import =, because those cannot be compiled to ES.next. Workaround: Convert to using export default and export const, and import x, {y} from "z".
>
> Does not support baseUrl. Workaround: use gatsby-plugin-root-import and configure it to point the baseUrl value (also set baseUrl option in tsconfig.json file).

https://babeljs.io/docs/en/babel-plugin-transform-typescript.html

Type checking

First of all you should set up your IDE so that type errors are surfaced. Visual Studio Code is very good in this regard.

In addition, you can see the instructions in TypeScript-Babel-Starter for setting up a `type-check` task.