App Config File

Nuxt 3 provides an `app.config` config file to expose reactive configuration within your application with the ability to update it at runtime within lifecycle or using a nuxt plugin and editing it with HMR (hot-module-replacement).

You can easily provide runtime app configuration using `app.config.ts` file. It can have either of `.ts`, `.js`, or `.mjs` extensions.

app.config.ts

```ts
export default defineAppConfig({
  foo: 'bar'
})
```

Do not put any secret values inside `app.config` file. It is exposed to the user client bundle.

Defining App Config

To expose config and environment variables to the rest of your app, you will need to define configuration in `app.config` file.

Example:

app.config.ts

```ts
export default defineAppConfig({
  theme: {
    primaryColor: '#abababab'
  }
})
```

When adding `theme` to the `app.config`, Nuxt uses Vite or webpack to bundle the code. We can universally access `theme` both when server-rendering the page and in the browser using useAppConfig composable.

```
const appConfig = useAppConfig()

console.log(appConfig.theme)
```

Manually Typing App Config

Nuxt tries to automatically generate a TypeScript interface from provided app config.

It is also possible to type app config manually. There are two possible things you might want to type.

Typing App Config Input

`AppConfigInput` might be used by module authors who are declaring what valid *input* options are when setting app config. This will not affect the type of `useAppConfig()`.

index.d.ts

```
declare module 'nuxt/schema' {
  interface AppConfigInput {
    /** Theme configuration */
    theme?: {
      /** Primary app color */
      primaryColor?: string
    }
  }
}

// It is always important to ensure you import/export something when augmenting a type
export {}
```

Typing App Config Output

If you want to type the result of calling `useAppConfig()`, then you will want to extend `AppConfig`.

Be careful when typing `AppConfig` as you will overwrite the types Nuxt infers from your actually defined app config.

index.d.ts

```ts
declare module 'nuxt/schema' {
  interface AppConfig {
    // This will entirely replace the existing inferred `theme` property
    theme: {
      // You might want to type this value to add more specific types than Nuxt can infer,
      // such as string literal types
      primaryColor?: 'red' | 'blue'
    }
  }
}

// It is always important to ensure you import/export something when augmenting a type
export {}
```

App Config Merging Strategy in Layers

Nuxt uses a custom merging strategy for the `AppConfig` within the layers of your application. This strategy is implemented using a Function Merger, which allows defining a custom merging strategy for every key in `app.config` that has an array as value.

The Function Merger should only be used in the base `app.config` of your application.

Here's an example of how you can use:

layer/app.config.ts app.config.ts

layer/app.config.ts

```ts
export default defineAppConfig({
  // Default array value
  array: ['hello'],
})
```

Edit on Github