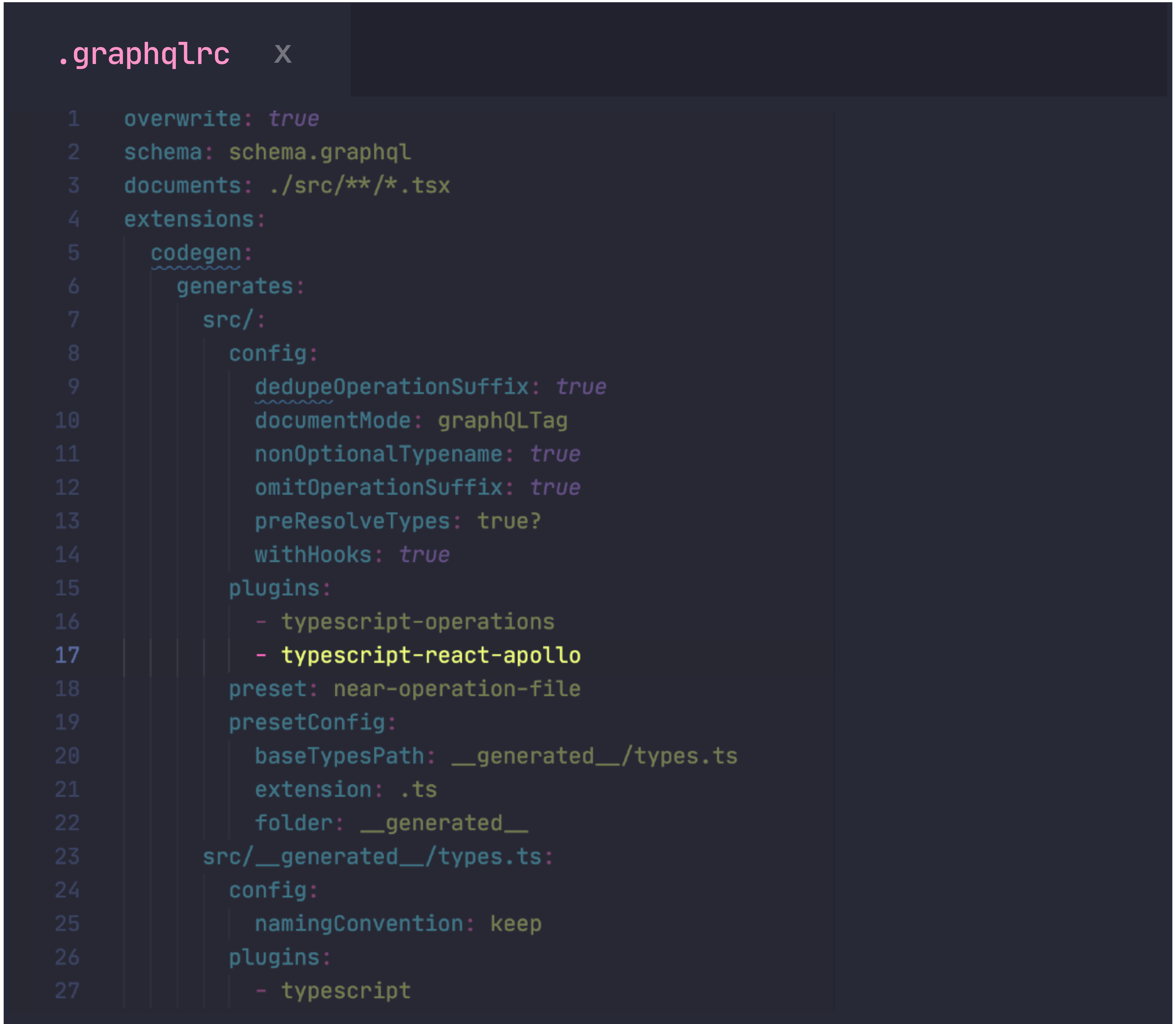
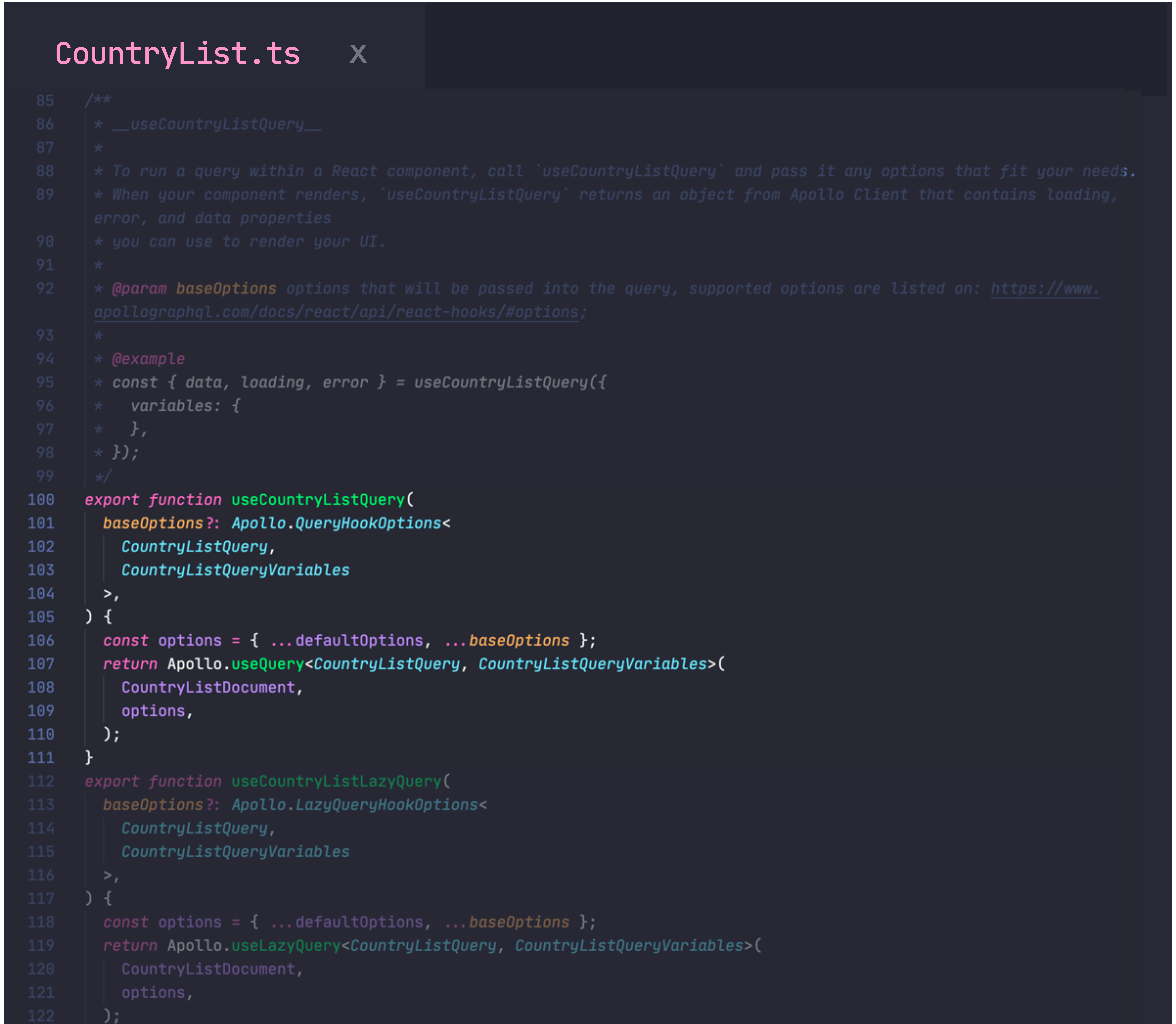


```yaml
overwrite: true
schema: schema.graphql
documents: ./src/**/*.tsx
extensions:
  codegen:
    generates:
      src/:
        config:
          dedupeOperationSuffix: true
          documentMode: graphQLTag
          nonOptionalTypename: true
          omitOperationSuffix: true
          preResolveTypes: true?
          withHooks: true
        plugins:
          - typescript-operations
          - typescript-react-apollo
        preset: near-operation-file
        presetConfig:
          baseTypesPath: __generated__/types.ts
          extension: .ts
          folder: __generated__
      src/__generated__/types.ts:
        config:
          namingConvention: keep
        plugins:
          - typescript
```



CountryList.ts x

```
/**
 * __useCountryListQuery__
 *
 * To run a query within a React component, call `useCountryListQuery` and pass it any options that fit your needs.
 * When your component renders, `useCountryListQuery` returns an object from Apollo Client that contains loading, error, and data properties
 * you can use to render your UI.
 *
 * @param baseOptions options that will be passed into the query, supported options are listed on: https://www.apollographql.com/docs/react/api/react-hooks/#options;
 *
 * @example
 * const { data, loading, error } = useCountryListQuery({
 *   variables: {
 *   },
 * });
 */
export function useCountryListQuery(
  baseOptions?: Apollo.QueryHookOptions<
    CountryListQuery,
    CountryListQueryVariables
  >,
) {
  const options = { ...defaultOptions, ...baseOptions };
  return Apollo.useQuery<CountryListQuery, CountryListQueryVariables>(
    CountryListDocument,
    options,
  );
}
export function useCountryListLazyQuery(
  baseOptions?: Apollo.LazyQueryHookOptions<
    CountryListQuery,
    CountryListQueryVariables
  >,
) {
  const options = { ...defaultOptions, ...baseOptions };
  return Apollo.useLazyQuery<CountryListQuery, CountryListQueryVariables>(
    CountryListDocument,
    options,
  );
```