# Absolute Imports and Module Path Aliases

▶ **Examples**

Next.js has in-built support for the `"paths"` and `"baseUrl"` options of `tsconfig.json` and `jsconfig.json` files.

These options allow you to alias project directories to absolute paths, making it easier to import modules. For example:

```
// before
import { Button } from '../../../components/button';

// after
import { Button } from '@/components/button';
```

**Good to know**: `create-next-app` will prompt to configure these options for you.

## Absolute Imports

The `baseUrl` configuration option allows you to import directly from the root of the project.

An example of this configuration:

tsconfig.json / jsconfig.json

```json
{
  "compilerOptions": {
    "baseUrl": "."
  }
}
```

components/button.tsx

```tsx
export default function Button() {
  return <button>Click me</button>;
}
```

app/page.tsx

```tsx
import Button from 'components/button';

export default function HomePage() {
  return (
    <>
      <h1>Hello World</h1>
      <Button />
    </>
  );
}
```

---

## Module Aliases

In addition to configuring the `baseUrl` path, you can use the `"paths"` option to "alias" module paths.

For example, the following configuration maps `@/components/*` to `components/*`:

tsconfig.json or jsconfig.json

```json
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/components/*": ["components/*"]
    }
  }
}
```

components/button.tsx

```tsx
export default function Button() {
  return <button>Click me</button>;
}
```

app/page.tsx

```tsx
import Button from '@/components/button';

export default function HomePage() {
  return (
    <>
      <h1>Hello World</h1>
      <Button />
    </>
  );
}
```