Absolute Imports and Module path aliases

Examples

Absolute Imports

Next.js automatically supports the `tsconfig.json` and `jsconfig.json` `"paths"` and `"baseUrl"` options since Next.js 9.4.

> Note: `jsconfig.json` can be used when you don't use TypeScript
>
> Note: you need to restart dev server to reflect modifications done in `tsconfig.json` / `jsconfig.json`

These options allow you to configure module aliases, for example a common pattern is aliasing certain directories to use absolute paths.

One useful feature of these options is that they integrate automatically into certain editors, for example vscode.

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

An example of this configuration:

```
// tsconfig.json or jsconfig.json
{
  "compilerOptions": {
    "baseUrl": "."
  }
}
```

```
// components/button.js
export default function Button() {
  return <button>Click me</button>
}
```

```
// pages/index.js
import Button from 'components/button'

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

While `baseUrl` is useful you might want to add other aliases that don't match 1 on 1. For this TypeScript has the `"paths"` option.

Using `"paths"` allows you to configure module aliases. For example `@/components/*` to `components/*`.

An example of this configuration:

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

```
// components/button.js
export default function Button() {
  return <button>Click me</button>
}
```

```
// pages/index.js
import Button from '@/components/button'

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