Next.js Codemods

Next.js provides Codemod transformations to help upgrade your Next.js codebase when a feature is deprecated.

Codemods are transformations that run on your codebase programmatically. This allows for a large amount of changes to be applied without having to manually go through every file.

Usage

`npx @next/codemod <transform> <path>`

- `transform` - name of transform, see available transforms below.
- `path` - files or directory to transform
- `--dry` Do a dry-run, no code will be edited
- `--print` Prints the changed output for comparison

Next.js 11

`cra-to-next` (experimental)

Migrates a Create React App project to Next.js; creating a pages directory and necessary config to match behavior. Client-side only rendering is leveraged initially to prevent breaking compatibility due to `window` usage during SSR and can be enabled seamlessly to allow gradual adoption of Next.js specific features.

Please share any feedback related to this transform in this discussion.

Next.js 10

`add-missing-react-import`

Transforms files that do not import `React` to include the import in order for the new React JSX transform to work.

For example:

```
// my-component.js
export default class Home extends React.Component {
  render() {
    return <div>Hello World</div>
  }
}
```

Transforms into:

```
// my-component.js
import React from 'react'
export default class Home extends React.Component {
  render() {
    return <div>Hello World</div>
  }
}
```

Next.js 9

`name-default-component`

Transforms anonymous components into named components to make sure they work with Fast Refresh.

For example:

```
// my-component.js
export default function () {
  return <div>Hello World</div>
}
```

Transforms into:

```
// my-component.js
export default function MyComponent() {
  return <div>Hello World</div>
}
```

The component will have a camel cased name based on the name of the file, and it also works with arrow functions.

Usage

Go to your project

```
cd path-to-your-project/
```

Run the codemod:

```
npx @next/codemod name-default-component
```

`withamp-to-config`

Transforms the `withAmp` HOC into Next.js 9 page configuration.

For example:

```
// Before
import { withAmp } from 'next/amp'

function Home() {
  return <h1>My AMP Page</h1>
}

export default withAmp(Home)
```

```
// After
export default function Home() {
  return <h1>My AMP Page</h1>
}
```

```
export const config = {
  amp: true,
}
```

Usage

Go to your project

```
cd path-to-your-project/
```

Run the codemod:

```
npx @next/codemod withamp-to-config
```

Next.js 6

`url-to-withrouter`

Transforms the deprecated automatically injected `url` property on top level pages to using `withRouter` and the `router` property it injects. Read more here: https://nextjs.org/docs/messages/url-deprecated

For example:

```
// From
import React from 'react'
export default class extends React.Component {
  render() {
    const { pathname } = this.props.url
    return <div>Current pathname: {pathname}</div>
  }
}
```

```
// To
import React from 'react'
import { withRouter } from 'next/router'
export default withRouter(
  class extends React.Component {
    render() {
      const { pathname } = this.props.router
      return <div>Current pathname: {pathname}</div>
    }
  }
)
```

This is one case. All the cases that are transformed (and tested) can be found in the `__testfixtures__` directory.

Usage

Go to your project

```
cd path-to-your-project/
```

Run the codemod:

```
npx @next/codemod url-to-withrouter
```