Contributing to Next.js

Watch the 40-minute walkthrough video on how to contribute to Next.js.

- Read about our Commitment to Open Source.
- To contribute to our examples, please see Adding examples below.
- Before jumping into a PR be sure to search existing PRs or issues for an open or closed item that relates to your submission.

Developing

The development branch is `canary`. This is the branch that all pull requests should be made against. The changes on the `canary` branch are published to the `@canary` tag on npm regularly.

To develop locally:

1. Fork this repository to your own GitHub account and then clone it to your local device.
2. Create a new branch:

   ```
   git checkout -b MY_BRANCH_NAME
   ```

3. Install yarn:

   ```
   npm install -g yarn
   ```

4. Install the dependencies with:

   ```
   yarn
   ```

5. Start developing and watch for code changes:

   ```
   yarn dev
   ```

6. In a new terminal, run `yarn types` to compile declaration files from TypeScript.

   Note: You may need to repeat this step if your types get outdated.

For instructions on how to build a project with your local version of the CLI, see **Developing with your local version of Next.js** below. (Naively linking the binary is not sufficient to develop locally.)

Building

You can build the project, including all type definitions, with:

```
yarn build
# - or -
yarn prepublish
```

By default the latest canary of the next-swc binaries will be installed and used. If you are actively working on Rust code or you need to test out the most recent Rust code that hasn't been published as a canary yet you can install Rust and run `yarn --cwd packages/next-swc build-native`.

If you need to clean the project for any reason, use `yarn clean`.

Testing

See the testing readme for information on writing tests.

Running tests

```
yarn testonly
```

If you would like to run the tests in headless mode (with the browser windows hidden) you can do

```
yarn testheadless
```

Running a specific test suite (e.g. `production`) inside of the `test/integration` directory:

```
yarn testonly --testPathPattern "production"
```

Running one test in the `production` test suite:

```
yarn testonly --testPathPattern "production" -t "should allow etag header support"
```

Linting

To check the formatting of your code:

```
yarn lint
```

If you get errors, you can fix them with:

```
yarn lint-fix
```

Running the example apps

Running examples can be done with:

```
yarn next ./test/integration/basic
# OR
yarn next ./examples/basic-css/
```

To figure out which pages are available for the given example, you can run:

```
EXAMPLE=./test/integration/basic
(\
  cd $EXAMPLE/pages; \
  find . -type f \
  | grep -v '\.next' \
  | sed 's#^\.##' \
  | sed 's#index\.js##' \
  | sed 's#\.js$##' \
  | xargs -I{} echo localhost:3000{} \
)
```

Developing with your local version of Next.js

There are two options to develop with your local version of the codebase:

Set as a local dependency in package.json

1. In your app's `package.json`, replace:

   ```
   "next": "<next-version>",
   ```

   with:

   ```
   "next": "file:/path/to/next.js/packages/next",
   ```

2. In your app's root directory, make sure to remove `next` from `node_modules` with:

   ```
   rm -rf ./node_modules/next
   ```

3. In your app's root directory, run:

   ```
   yarn
   ```

   to re-install all of the dependencies.

   Note that Next will be copied from the locally compiled version as opposed to from being downloaded from the NPM registry.

4. Run your application as you normally would.

5. To update your app's dependencies, after you've made changes to your local `next` repository. In your app's root directory, run:

   ```
   yarn install --force
   ```

Troubleshooting

- If you see the below error while running `yarn dev` with next:

```
Failed to load SWC binary, see more info here:
https://nextjs.org/docs/messages/failed-loading-swc
```

Try to add the below section to your `package.json`, then run again

```
"optionalDependencies": {
  "@next/swc-linux-x64-gnu": "canary",
  "@next/swc-win32-x64-msvc": "canary",
  "@next/swc-darwin-x64": "canary",
  "@next/swc-darwin-arm64": "canary"
},
```

Develop inside the monorepo

1. Move your app inside of the Next.js monorepo.
2. Run with `yarn next-with-deps ./app-path-in-monorepo`

This will use the version of `next` built inside of the Next.js monorepo and the main `yarn dev` monorepo command can be running to make changes to the local Next.js version at the same time (some changes might require re-running `yarn next-with-deps` to take effect).

Adding warning/error descriptions

In Next.js we have a system to add helpful links to warnings and errors.

This allows for the logged message to be short while giving a broader description and instructions on how to solve the warning/error.

In general, all warnings and errors added should have these links attached.

Below are the steps to add a new link:

1. Run `yarn new-error` which will create the error document and update the manifest automatically.
2. Add the following url to your warning/error: `https://nextjs.org/docs/messages/<file-path-without-dotmd>`.

   For example, to link to `errors/api-routes-static-export.md` you use the url: `https://nextjs.org/docs/messages/api-routes-static-export`

Adding examples

When you add an example to the examples directory, don't forget to add a `README.md` file with the following format:

- Replace `DIRECTORY_NAME` with the directory name you're adding.
- Fill in `Example Name` and `Description`.
- Examples should be TypeScript first, if possible.
- You don't need to add `name` or `version` in your `package.json`.
- Ensure all your dependencies are up to date.
- Ensure you're using `next/image`.
- To add additional installation instructions, please add it where appropriate.
- To add additional notes, add `## Notes` section at the end.
- Remove the `Deploy your own` section if your example can't be immediately deployed to Vercel.

~~~markdown
# Example Name

Description

## Deploy your own

Deploy the example using [Vercel](https://vercel.com?utm_source=github&utm_medium=readme&utm_campaign=next-example):

[![Deploy with Vercel](https://vercel.com/button)](https://vercel.com/new/git/external?repository-url=https://github.com/vercel/next.js/tree/canary/examples/DIRECTORY_NAME&project-name=DIRECTORY_NAME&repository-name=DIRECTORY_NAME)

## How to use

Execute [`create-next-app`](https://github.com/vercel/next.js/tree/canary/packages/create-next-app) with [npm](https://docs.npmjs.com/cli/init) or [Yarn](https://yarnpkg.com/lang/en/docs/cli/create/) to bootstrap the example:

```bash
npx create-next-app --example DIRECTORY_NAME DIRECTORY_NAME-app
# or
yarn create next-app --example DIRECTORY_NAME DIRECTORY_NAME-app
# or
pnpm create next-app -- --example DIRECTORY_NAME DIRECTORY_NAME-app
```

Deploy it to the cloud with [Vercel](https://vercel.com/new?utm_source=github&utm_medium=readme&utm_campaign=next-example) ([Documentation](https://nextjs.org/docs/deployment)).
~~~

Publishing

Repository maintainers can use `yarn publish-canary` to publish a new version of all packages to npm.