# Upgrading to Version 9

To upgrade to version 9, run the following command:

Terminal

```
npm install next@9

yarn add next@9
```

## Production Deployment on Vercel

If you previously configured `routes` in your `vercel.json` file for dynamic routes, these rules can be removed when leveraging Next.js 9's new Dynamic Routing feature.

Next.js 9's dynamic routes are **automatically configured on Vercel** and do not require any `vercel.json` customization.

You can read more about Dynamic Routing here.

## Check your Custom App File (`pages/_app.js`)

If you previously copied the Custom `<App>` example, you may be able to remove your `getInitialProps`.

Removing `getInitialProps` from `pages/_app.js` (when possible) is important to leverage new Next.js features!

The following `getInitialProps` does nothing and may be removed:

```
class MyApp extends App {
  // Remove me, I do nothing!
  static async getInitialProps({ Component, ctx }) {
    let pageProps = {};

    if (Component.getInitialProps) {
      pageProps = await Component.getInitialProps(ctx);
    }

    return { pageProps };
  }

  render() {
    // ... etc
  }
}
```

---

## Breaking Changes

### `@zeit/next-typescript` is no longer necessary

Next.js will now ignore usage `@zeit/next-typescript` and warn you to remove it. Please remove this plugin from your `next.config.js`.

Remove references to `@zeit/next-typescript/babel` from your custom `.babelrc` (if present).

The usage of `fork-ts-checker-webpack-plugin` ↗ should also be removed from your `next.config.js`.

TypeScript Definitions are published with the `next` package, so you need to uninstall `@types/next` as they would conflict.

The following types are different:

This list was created by the community to help you upgrade, if you find other differences please send a pull-request to this list to help other users.

From:

```
import { NextContext } from 'next';
import { NextAppContext, DefaultAppIProps } from 'next/app';
import { NextDocumentContext, DefaultDocumentIProps } from 'next/document';
```

to

```
import { NextPageContext } from 'next';
import { AppContext, AppInitialProps } from 'next/app';
import { DocumentContext, DocumentInitialProps } from 'next/document';
```

## The `config` key is now an export on a page

You may no longer export a custom variable named `config` from a page (i.e. `export { config }` / `export const config ...`). This exported variable is now used to specify page-level Next.js configuration like Opt-in AMP and API Route features.

You must rename a non-Next.js-purposed `config` export to something different.

## `next/dynamic` no longer renders "loading..." by default while loading

Dynamic components will not render anything by default while loading. You can still customize this behavior by setting the `loading` property:

```
import dynamic from 'next/dynamic';

const DynamicComponentWithCustomLoading = dynamic(
  () => import('../components/hello2'),
  {
    loading: () => <p>Loading</p>,
  },
);
```

## `withAmp` has been removed in favor of an exported configuration object

Next.js now has the concept of page-level configuration, so the `withAmp` higher-order component has been removed for consistency.

This change can be **automatically migrated by running the following commands in the root of your Next.js project:**

```
>_ Terminal
curl -L https://github.com/vercel/next-codemod/archive/master.tar.gz | tar -xz --strip=2 next
```

To perform this migration by hand, or view what the codemod will produce, see below:

**Before**

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

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

export default withAmp(Home)
// or
export default withAmp(Home, { hybrid: true })
```

**After**

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

export const config = {
  amp: true,
  // or
  amp: 'hybrid',
};
```

## `next export` no longer exports pages as `index.html`

Previously, exporting `pages/about.js` would result in `out/about/index.html`. This behavior has been changed to result in `out/about.html`.

You can revert to the previous behavior by creating a `next.config.js` with the following content:

next.config.js

```
module.exports = {
  trailingSlash: true,
};
```

## `pages/api/` is treated differently

Pages in `pages/api/` are now considered API Routes. Pages in this directory will no longer contain a client-side bundle.

## Deprecated Features

### `next/dynamic` has deprecated loading multiple modules at once

The ability to load multiple modules at once has been deprecated in `next/dynamic` to be closer to React's implementation (`React.lazy` and `Suspense`).

Updating code that relies on this behavior is relatively straightforward! We've provided an example of a before/after to help you migrate your application:

#### Before

```
import dynamic from 'next/dynamic';

const HelloBundle = dynamic({
  modules: () => {
    const components = {
      Hello1: () => import('../components/hello1').then((m) => m.default),
      Hello2: () => import('../components/hello2').then((m) => m.default),
    };

    return components;
  },
  render: (props, { Hello1, Hello2 }) => (
    <div>
      <h1>{props.title}</h1>
      <Hello1 />
      <Hello2 />
    </div>
  ),
});

function DynamicBundle() {
  return <HelloBundle title="Dynamic Bundle" />;
}

export default DynamicBundle;
```

#### After

```
import dynamic from 'next/dynamic';

const Hello1 = dynamic(() => import('../components/hello1'));
const Hello2 = dynamic(() => import('../components/hello2'));

function HelloBundle({ title }) {
  return (
    <div>
```

```
      <h1>{title}</h1>
      <Hello1 />
      <Hello2 />
    </div>
  );
}

function DynamicBundle() {
  return <HelloBundle title="Dynamic Bundle" />;
}

export default DynamicBundle;
```