NextRequest and NextResponse

`next/server` provides server-only helpers for use in Middleware and Edge API Routes.

NextRequest

The `NextRequest` object is an extension of the native `Request` interface, with the following added methods and properties:

- `cookies` - A RequestCookies instance with cookies from the `Request`. It reads/mutates the `Cookie` header of the request. See also Using cookies in Middleware.
  - `get` - A method that takes a cookie `name` and returns an object with `name` and `value`. If a cookie with `name` isn't found, it returns `undefined`. If multiple cookies match, it will only return the first match.
  - `getAll` - A method that is similar to `get`, but returns a list of all the cookies with a matching `name`. If `name` is unspecified, it returns all the available cookies.
  - `set` - A method that takes an object with properties of `CookieListItem` as defined in the W3C CookieStore API spec.
  - `delete` - A method that takes either a cookie `name` or a list of names. and removes the cookies matching the name(s). Returns `true` for deleted and `false` for undeleted cookies.
  - `has` - A method that takes a cookie `name` and returns a `boolean` based on if the cookie exists (`true`) or not (`false`).
  - `clear` - A method that takes no argument and will effectively remove the `Cookie` header.
- `nextUrl`: Includes an extended, parsed, URL object that gives you access to Next.js specific properties such as `pathname`, `basePath`, `trailingSlash` and `i18n`. Includes the following properties:
  - `basePath` (`string`)
  - `buildId` (`string || undefined`)

    - `defaultLocale` (`string || undefined`)
    - `domainLocale`
        - `defaultLocale`: (`string`)
        - `domain`: (`string`)
        - `http`: (`boolean || undefined`)
        - `locales`: (`string[] || undefined`)
    - `locale` (`string || undefined`)
    - `url` (`URL`)
- `ip`: (`string || undefined`) - Has the IP address of the `Request`. This information is provided by your hosting platform.
- `geo` - Has the geographic location from the `Request`. This information is provided by your hosting platform. Includes the following properties:
    - `city` (`string || undefined`)
    - `country` (`string || undefined`)
    - `region` (`string || undefined`)
    - `latitude` (`string || undefined`)
    - `longitude` (`string || undefined`)

You can use the `NextRequest` object as a direct replacement for the native `Request` interface, giving you more control over how you manipulate the request.

`NextRequest` can be imported from `next/server`:

```
import type { NextRequest } from 'next/server';
```

NextFetchEvent

The `NextFetchEvent` object extends the native `FetchEvent` ↗ object, and includes the `waitUntil()` ↗ method.

The `waitUntil()` method can be used to prolong the execution of the function if you have other background work to make.

```ts
import { NextResponse } from 'next/server';
import type { NextFetchEvent, NextRequest } from 'next/server';

export function middleware(req: NextRequest, event: NextFetchEvent) {
  event.waitUntil(
    fetch('https://my-analytics-platform.com', {
      method: 'POST',
      body: JSON.stringify({ pathname: req.nextUrl.pathname }),
    }),
  );

  return NextResponse.next();
}
```

The `NextFetchEvent` object can be imported from `next/server`:

```ts
import type { NextFetchEvent } from 'next/server';
```

NextResponse

The `NextResponse` class extends the native Response ↗ interface, with the following:

Public Methods

Public methods are available on an instance of the `NextResponse` class. Depending on your use case, you can create an instance and assign to a variable, then access the following public methods:

- `cookies` - A ResponseCookies ↗ instance with the cookies from the `Response`. It reads/mutates the `Set-Cookie` header of the response. See also Using cookies in Middleware.
  - `get` - A method that takes a cookie `name` and returns an object with `name` and `value`. If a cookie with `name` isn't found, it returns `undefined`. If multiple cookies match, it will only return the first match.
  - `getAll` - A method that is similar to `get`, but returns a list of all the cookies with a matching `name`. If `name` is unspecified, it returns all the available cookies.
  - `set` - A method that takes an object with properties of `CookieListItem` as defined in the W3C CookieStore API ↗ spec.
  - `delete` - A method that takes either a cookie `name` or a list of names. and removes the cookies matching the name(s). Returns `true` for deleted and `false` for undeleted cookies.

Static Methods

The following static methods are available on the `NextResponse` class directly:

- `redirect()` - Returns a `NextResponse` with a redirect set
- `rewrite()` - Returns a `NextResponse` with a rewrite set
- `next()` - Returns a `NextResponse` that will continue the middleware chain

To use the methods above, **you must return the `NextResponse` object** returned. `NextResponse` can be imported from `next/server`:

```
import { NextResponse } from 'next/server';
```

---

userAgent

The `userAgent` helper allows you to interact with the user agent object from the request. It is abstracted from the native `Request` object, and is an opt in feature. It has the following properties:

- `isBot` : (`boolean`) Whether the request comes from a known bot
- `browser`
  - `name` : (`string || undefined`) The name of the browser
  - `version` : (`string || undefined`) The version of the browser, determined dynamically
- `device`
  - `model` : (`string || undefined`) The model of the device, determined dynamically
  - `type` : (`string || undefined`) The type of the browser, can be one of the following values: `console`, `mobile`, `tablet`, `smarttv`, `wearable`, `embedded`, or `undefined`
  - `vendor` : (`string || undefined`) The vendor of the device, determined dynamically
- `engine`
  - `name` : (`string || undefined`) The name of the browser engine, could be one of the following values: `Amaya`, `Blink`, `EdgeHTML`, `Flow`, `Gecko`, `Goanna`, `iCab`, `KHTML`, `Links`, `Lynx`, `NetFront`, `NetSurf`, `Presto`, `Tasman`, `Trident`, `w3m`, `WebKit` or `undefined`
  - `version` : (`string || undefined`) The version of the browser engine, determined dynamically, or `undefined`
- `os`

    - `name`: (`string || undefined`) The name of the OS, could be `undefined`
    - `version`: (`string || undefined`) The version of the OS, determined dynamically, or `undefined`
- `cpu`
    - `architecture`: (`string || undefined`) The architecture of the CPU, could be one of the following values: `68k`, `amd64`, `arm`, `arm64`, `armhf`, `avr`, `ia32`, `ia64`, `irix`, `irix64`, `mips`, `mips64`, `pa-risc`, `ppc`, `sparc`, `sparc64` or `undefined`

`userAgent` can be imported from `next/server`:

```
import { userAgent } from 'next/server';
```

```
import { NextRequest, NextResponse, userAgent } from 'next/server';

export function middleware(request: NextRequest) {
  const url = request.nextUrl;
  const { device } = userAgent(request);
  const viewport = device.type === 'mobile' ? 'mobile' : 'desktop';
  url.searchParams.set('viewport', viewport);
  return NextResponse.rewrite(url);
}
```

FAQ

Why does `redirect` use 307 and 308?

When using `redirect()` you may notice that the status codes used are `307` for a temporary redirect, and `308` for a permanent redirect. While traditionally a `302` was used for a temporary redirect, and a `301` for a permanent redirect, many browsers changed the request method of the redirect, from a `POST` to `GET` request when using a `302`, regardless of the origins request method.

Taking the following example of a redirect from `/users` to `/people`, if you make a `POST` request to `/users` to create a new user, and are conforming to a `302` temporary redirect, the request method will be changed from a `POST` to a `GET` request. This doesn't make sense, as to create a new user, you should be making a `POST` request to `/people`, and not a `GET` request.

The introduction of the `307` status code means that the request method is preserved as `POST`.

- `302` - Temporary redirect, will change the request method from `POST` to `GET`
- `307` - Temporary redirect, will preserve the request method as `POST`

The `redirect()` method uses a `307` by default, instead of a `302` temporary redirect, meaning your requests will *always* be preserved as `POST` requests.

If you want to cause a `GET` response to a `POST` request, use `303`.

Learn more ↗ about HTTP Redirects.

How do I access Environment Variables?

`process.env` can be used to access Environment Variables from Edge Middleware. They are evaluated during `next build`:

Works	Does not work
`console.log(process.env.MY_ENV_VARIABLE)`	`const getEnv = name => process.env[name]`
`const { MY_ENV_VARIABLE } = process.env`	
`const { "MY-ENV-VARIABLE": MY_ENV_VARIABLE } = process.env`	