Compression

Compression can greatly decrease the size of the response body, thereby increasing the speed of a web app.

For **high-traffic** websites in production, it is strongly recommended to offload compression from the application server - typically in a reverse proxy (e.g., Nginx). In that case, you should not use compression middleware.

Use with Express (default)

Use the compression middleware package to enable gzip compression.

First install the required package:

```
$ npm i --save compression
```

Once the installation is complete, apply the compression middleware as global middleware.

```
import * as compression from 'compression';
// somewhere in your initialization file
app.use(compression());
```

Use with Fastify

If using the `FastifyAdapter`, you'll want to use fastify-compress:

```
$ npm i --save @fastify/compress
```

Once the installation is complete, apply the `@fastify/compress` middleware as global middleware.

```
import compression from '@fastify/compress';
// somewhere in your initialization file
await app.register(compression);
```

By default, `@fastify/compress` will use Brotli compression (on Node >= 11.7.0) when browsers indicate support for the encoding. While Brotli is quite efficient in terms of compression ratio, it's also quite slow. Due to this, you may want to tell fastify-compress to only use deflate and gzip to compress responses; you'll end up with larger responses but they'll be delivered much more quickly.

To specify encodings, provide a second argument to `app.register`:

```
await app.register(compression, { encodings: ['gzip', 'deflate'] });
```

The above tells `fastify-compress` to only use gzip and deflate encodings, preferring gzip if the client supports both.