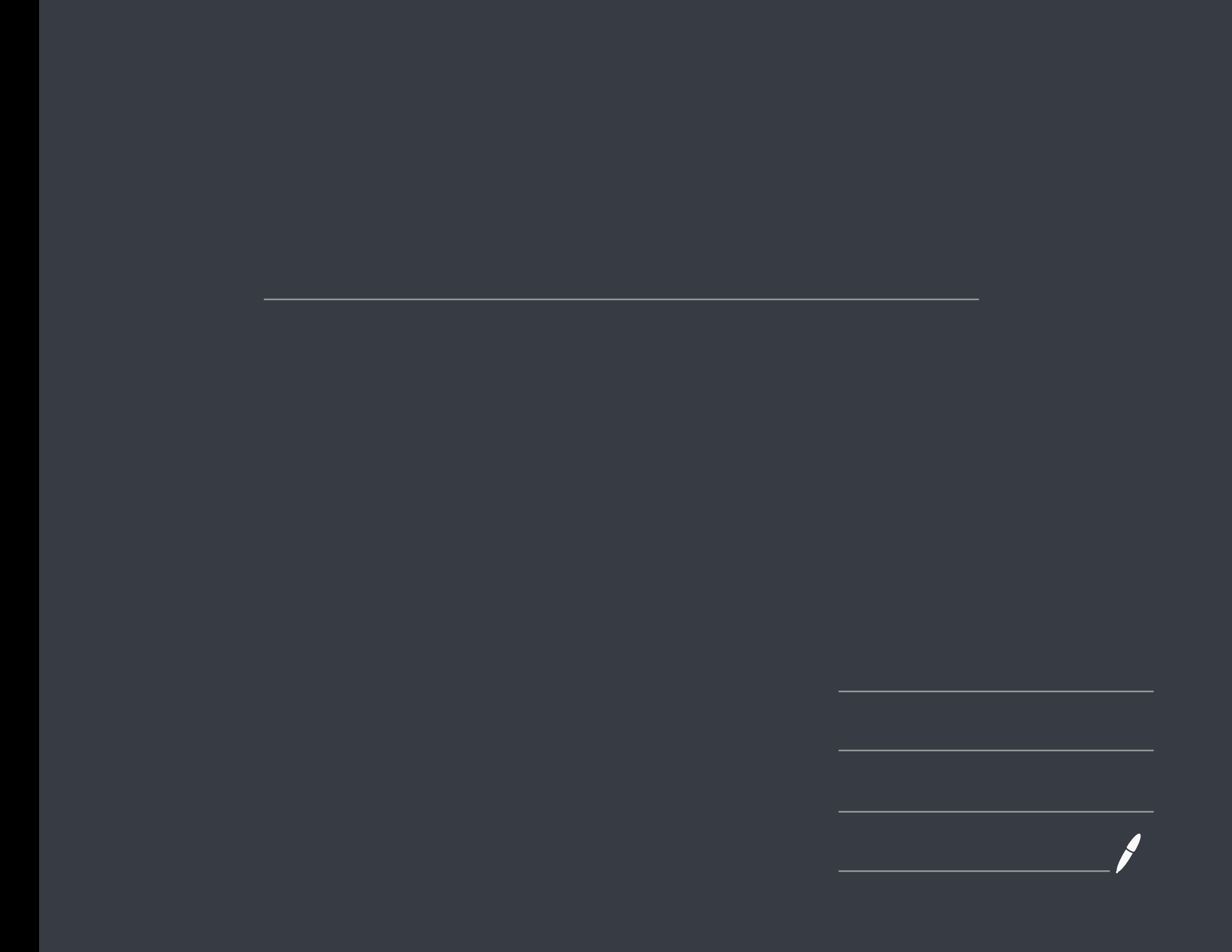

`app.use()` is intended for binding middleware to your application. The `path` is a "*mount*" or "*prefix*" path and limits the middleware to only apply to any paths requested that ***begin*** with it. It can even be used to embed another application:

```
// subapp.js
var express = require('express');
var app = modules.exports = express();
// ...
```

```
// server.js
var express = require('express');
var app = express();

app.use('/subapp', require('./subapp'));

// ...
```

By specifying `/` as a "*mount*" path, `app.use()` will respond to any path that starts with `/`, which are all of them and regardless of HTTP verb used:

- `GET /`
- `PUT /foo`
- `POST /foo/bar`
- etc.

`app.get()`, on the other hand, is part of Express' application routing and is intended for matching and handling a specific route when requested with the `GET` HTTP verb:

- `GET /`

And, the equivalent routing for your example of `app.use()` would actually be:

```
app.all(/^\/.*/, function (req, res) {
    res.send('Hello');
});
```

(*Update: Attempting to better demonstrate the differences.*)

The routing methods, including `app.get()`, are convenience methods that help you align responses to requests more precisely. They also add in support for features like parameters and `next('route')`.

Within each `app.get()` is a call to `app.use()`, so you can certainly do all of this with `app.use()` directly. But, doing so will often require (probably unnecessarily) reimplementing various amounts of boilerplate code.

Examples:

- For simple, static routes:

```
app.get('/', function (req, res) {
  // ...
});
```

vs.

```
app.use('/', function (req, res, next) {
  if (req.method !== 'GET' || req.url !== '/')
    return next();

  // ...
});
```

- With multiple handlers for the same route:

```
app.get('/', authorize('ADMIN'), function (req, res) {
  // ...
});
```

vs.

```
const authorizeAdmin = authorize('ADMIN');

app.use('/', function (req, res, next) {
  if (req.method !== 'GET' || req.url !== '/')
    return next();

  authorizeAdmin(req, res, function (err) {
    if (err) return next(err);

    // ...
  });
});
```

- With parameters:

```
app.get('/item/:id', function (req, res) {
  let id = req.params.id;
  // ...
});
```

vs.

```
const pathToRegExp = require('path-to-regexp');

function prepareParams(matches, pathKeys, previousParams) {
  var params = previousParams || {};

  // TODO: support repeating keys...
  matches.slice(1).forEach(function (segment, index) {
    let { name } = pathKeys[index];
    params[name] = segment;
  });

  return params;
}

const itemIdKeys = [];
const itemIdPattern = pathToRegExp('/item/:id', itemIdKeys);

app.use('/', function (req, res, next) {
  if (req.method !== 'GET') return next();

  var urlMatch = itemIdPattern.exec(req.url);
  if (!urlMatch) return next();

  if (itemIdKeys && itemIdKeys.length)
    req.params = prepareParams(urlMatch, itemIdKeys, req.params);

  let id = req.params.id;
  // ...
});
```

> Note: Express' implementation of these features are contained in its `Router`, `Layer`, and `Route`.