Middleware

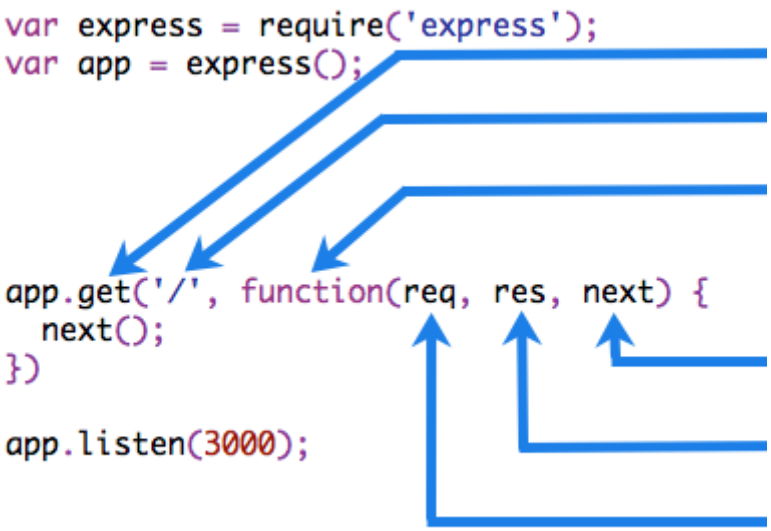
Middleware functions are functions that have access to the request object (req), the response object (res), and the next function in the application's request-response cycle. The next function is a function in the Express router which, when invoked, executes the middleware succeeding the current middleware.

Middleware functions can perform the following tasks:

- Execute any code.
- Make changes to the request and the response objects.
- End the request-response cycle.
- Call the next middleware in the stack.

If the current middleware function does not end the request-response cycle, it must call next() to pass control to the next middleware function. Otherwise, the request will be left hanging.

The following figure shows the elements of a middleware function call:



HTTP method for which the middleware function applies.

Path (route) for which the middleware function applies.

The middleware function.

Callback argument to the middleware function, called "next" by convention.

HTTP response argument to the middleware function, called "res" by convention.

HTTP request argument to the middleware function, called "req" by convention.

Ex:

Here is an example of a simple “Hello World” Express application. The remainder of this article will define and add three middleware functions to the application: one called myLogger that prints a simple log message, one called requestTime that displays the timestamp of the HTTP request, and one called validateCookies that validates incoming cookies.

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hello World!')
})

app.listen(3000)
```

Router

A router object is an isolated instance of middleware and routes. You can think of it as a “mini-application,” capable only of performing middleware and routing functions. Every Express application has a built-in app router.

A router behaves like middleware itself, so you can use it as an argument to app.use() or as the argument to another router’s use() method.

The top-level express object has a Router() method that creates a new router object.

Once you’ve created a router object, you can add middleware and HTTP method routes (such as get, put, post, and so on) to it just like an application. For example:

```
// invoked for any requests passed to this router
router.use(function (req, res, next) {
  // .. some logic here .. like any other middleware
  next()
})
```

```
// will handle any request that ends in /events
// depends on where the router is "use()'d"
router.get('/events', function (req, res, next) {
  // ..
})
```