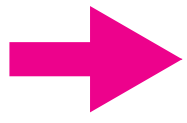
Middleware

Middle ware takes over the work that needs to be done inbetween a request and a response. An example would be that an image is uploaded but before it's displayed to the user, the image should be resized and croped and maybe even renamed. This is what the middle ware would do.

`next();` 

next() is key in middleware.
next() passes the **request Object** on to the next middleware in line.

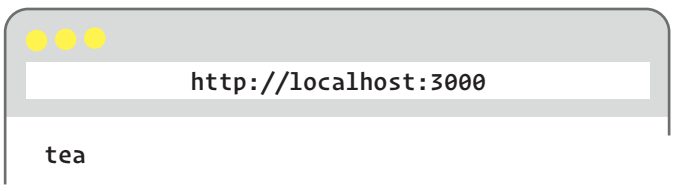
tablecontroller.js

```
exports.cupFill = (req,res,next) => {
      req.cup = "tea" ;
      next();
});

exports.cupCheck = (req,res) => {
      const cup = req.cup;
      res.render('examplefor', {fill : cup});
});
```

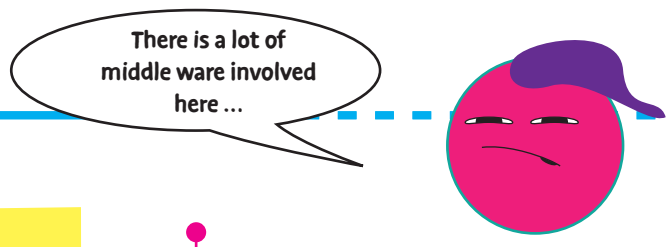
INDEX.JS

```
const tableController = require('../controllers/tableController');
...

router.get('/', tableControler.cupFill, tableController.cupCheck);

...
```

request coming in ... req.cup = "tea" and next()



a lot of middleware …



APP.JS

```
const express = require('express');
const bodyParser = require('bodyParser');
const routes = require('./routes/index');
const helpers = require('./helpers');

// create our express app
const app = express();

// set our template engine
app.set('views', './views');
app.set('view engine', 'pug');

//raw requests to readable req.body thing
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true}));

// create a middleware for our helpers

app.use((req,res,next)=>{
  res.locals.h = helpers;
  res.locals.currentPath = req.path;
  next();
});


app.use('/', routes);



app.use(errorHandlers.notFound);

app.use(errorHandlers.what);

app.use(errorHandlers.whatnot);
```

load all the stuff

middleware
request gets processed

routing
request gets route to endpoint

middleware
request gets processed

And if everthing goes fine the user gets they response

And if everthing goes goes wrong a Error is thrown...

middleware
error
gets processed

404