Accessing Query String Data

Express has a built-in query string parser unlike the core `http` module in which developers need to parse query strings manually. In Express, query string data can be accessed by `req.query.name` where name is the key of the value in a query string. Because query string parsing is a built-in feature of Express, there is no need to install anything with npm.

For example, an URL query string value `http://webapplog.com/search?term=node.js&page=1` can be accessed with `req.query.term` and `req.query.page` in a request handler such as `app.get()` or any other:

```
app.get('/search', (req, res) => {
  db.find(
    {term: req.query.term},
    {page: req.query.page, limit: 10}, (error, results)=> {
    // error handling
    res.send(results)
  })
})
```

By default, Express.js doesn't allow developers to route by query string arguments, such as the following:

```
GET: www.webapplog.com/?id=10233
GET: www.webapplog.com/about/?author=10239
GET: www.webapplog.com/books/?id=10&ref=201
```

However, it's trivial to write your own middleware. It might look like this:

```
app.use((req, res, next) => {
  if (req.query.id) {
    // process the id, then call next() when done
  else if (req.query.author) {
    // same approach as with id
  else if (req.query.id && req.query.ref) {
    // process when id and ref present
  } else {
    next()
  }
})

app.get('/about', (req, res, next) => {
  // this code is executed after the query string middleware
})
```

In this middleware, if/else is used to execute different code based on the value from query string `req.query`.