EXPRESSJS

SESSION HANDLING

EXPRESS-SESSION

```
const session =
   require('express-session');
```

SIMPLE CONFIGURATION

```
app.use(session({
    secret: ‘Dogs love beef‘
}));
```

Required

ALTERNATIVE CONFIGURATION **PART 1**

```
const sessionConfig = {
  secret: 'Cows hate fog',
  rolling: true,
  cookie: { secure: true,
            maxAge: 60000 },
  resave: false,
  saveUninitialized: false,
};
```

ALTERNATIVE CONFIGURATION **PART 2**

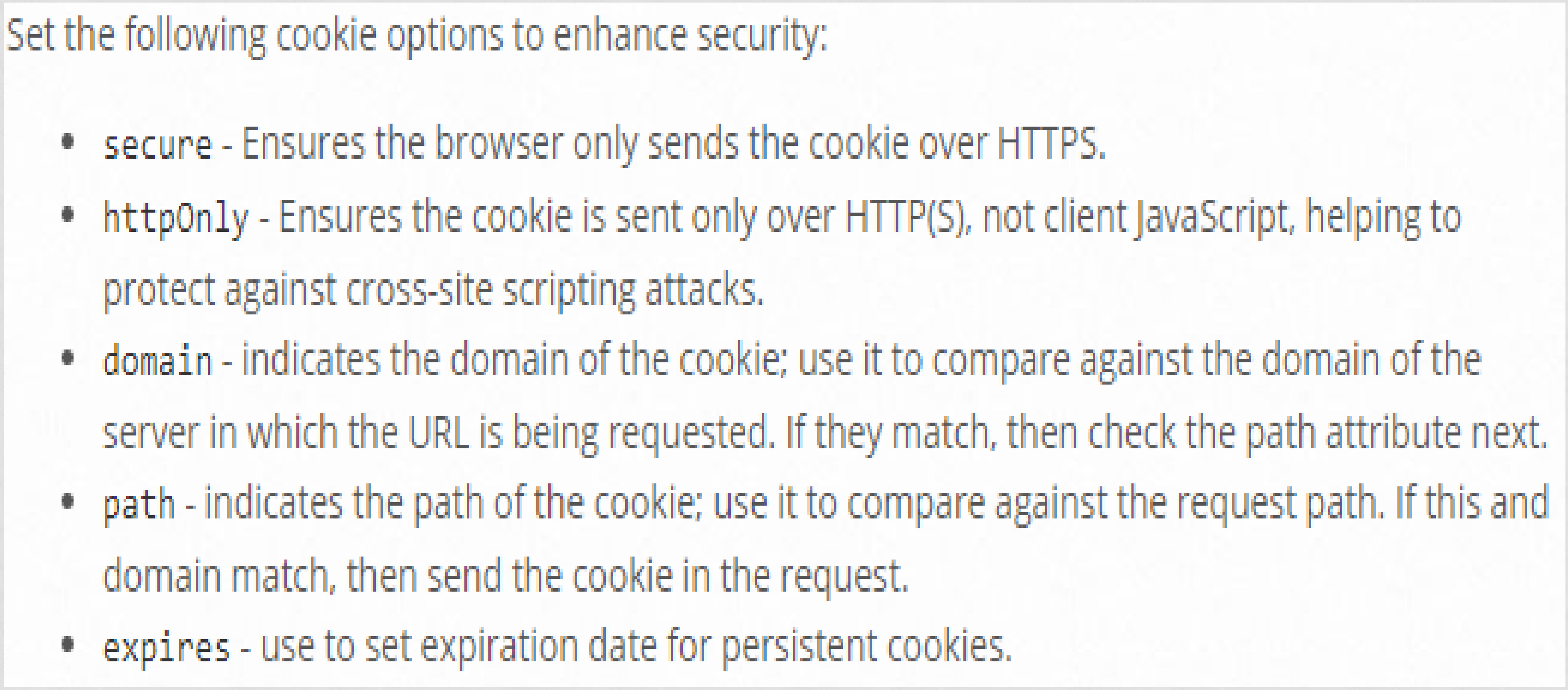
```
app.use(
    session(sessionConfig)
);
```

BEST PRACTICE
DEFAULT SESSION NAME

```
const sessionConfig = {
  secret: ‘Cows hate fog',
  name: ‘alpha’,
};
```

Do not use the default session name. Use something, ‘*not typical*’

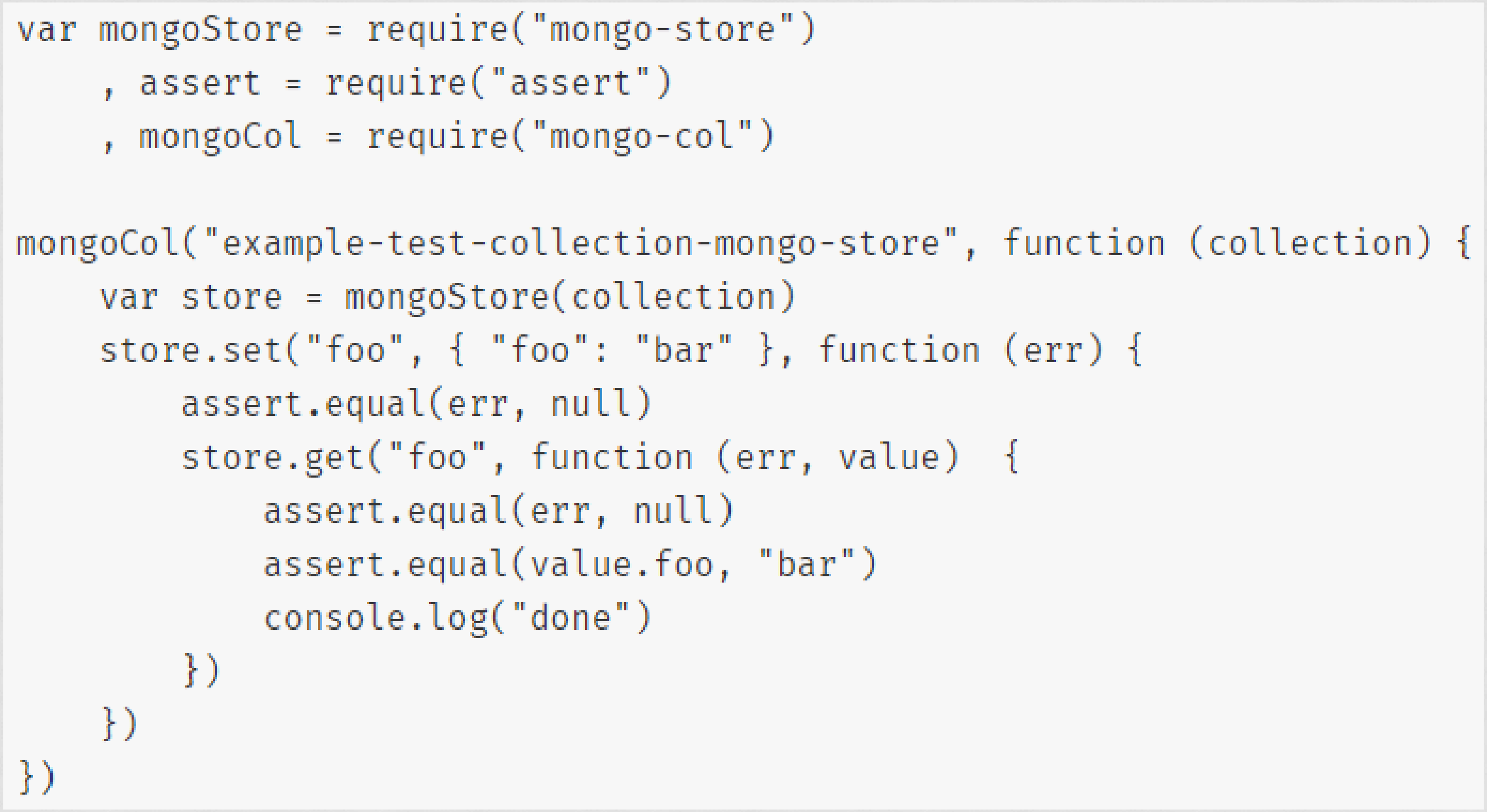
BEST PRACTICE
SETTING THE OPTIONS CORRECTLY



Set the following cookie options to enhance security:

- `secure` - Ensures the browser only sends the cookie over HTTPS.
- `httpOnly` - Ensures the cookie is sent only over HTTP(S), not client JavaScript, helping to protect against cross-site scripting attacks.
- `domain` - indicates the domain of the cookie; use it to compare against the domain of the server in which the URL is being requested. If they match, then check the path attribute next.
- `path` - indicates the path of the cookie; use it to compare against the request path. If this and domain match, then send the cookie in the request.
- `expires` - use to set expiration date for persistent cookies.

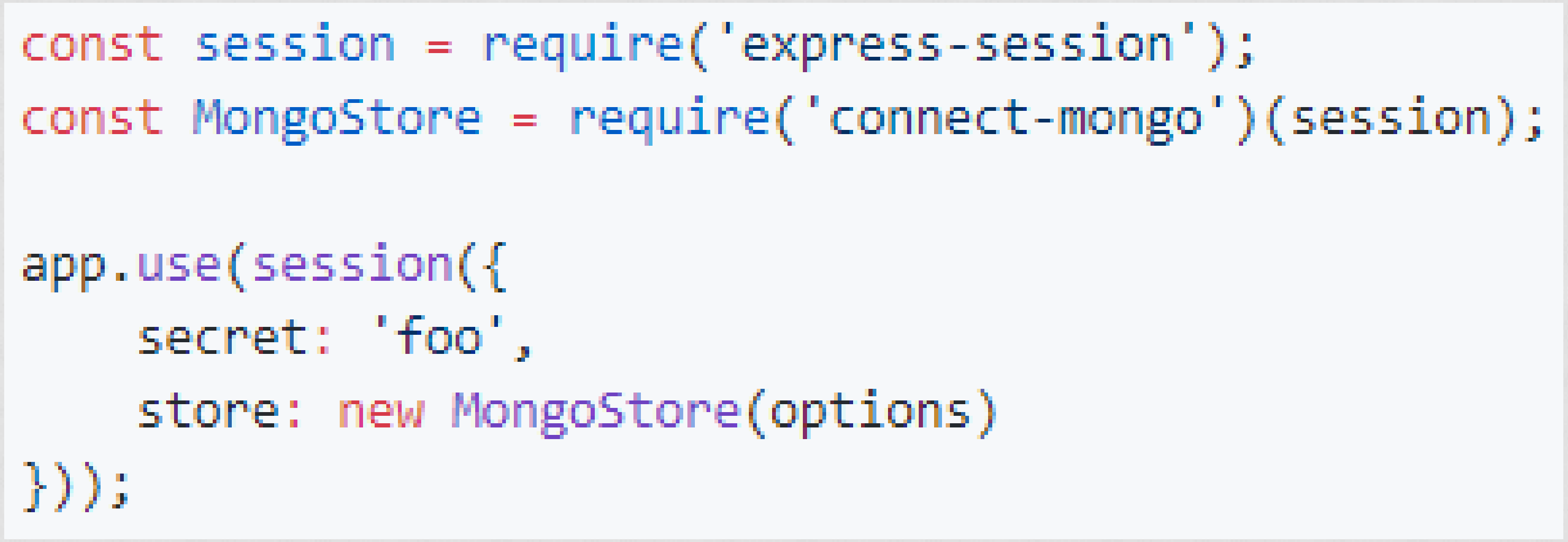
EXAMPLE
MONGO-STORE



```
var mongoStore = require("mongo-store")
    , assert = require("assert")
    , mongoCol = require("mongo-col")

mongoCol("example-test-collection-mongo-store", function (collection) {
    var store = mongoStore(collection)
    store.set("foo", { "foo": "bar" }, function (err) {
        assert.equal(err, null)
        store.get("foo", function (err, value)  {
            assert.equal(err, null)
            assert.equal(value.foo, "bar")
            console.log("done")
        })
    })
})
```

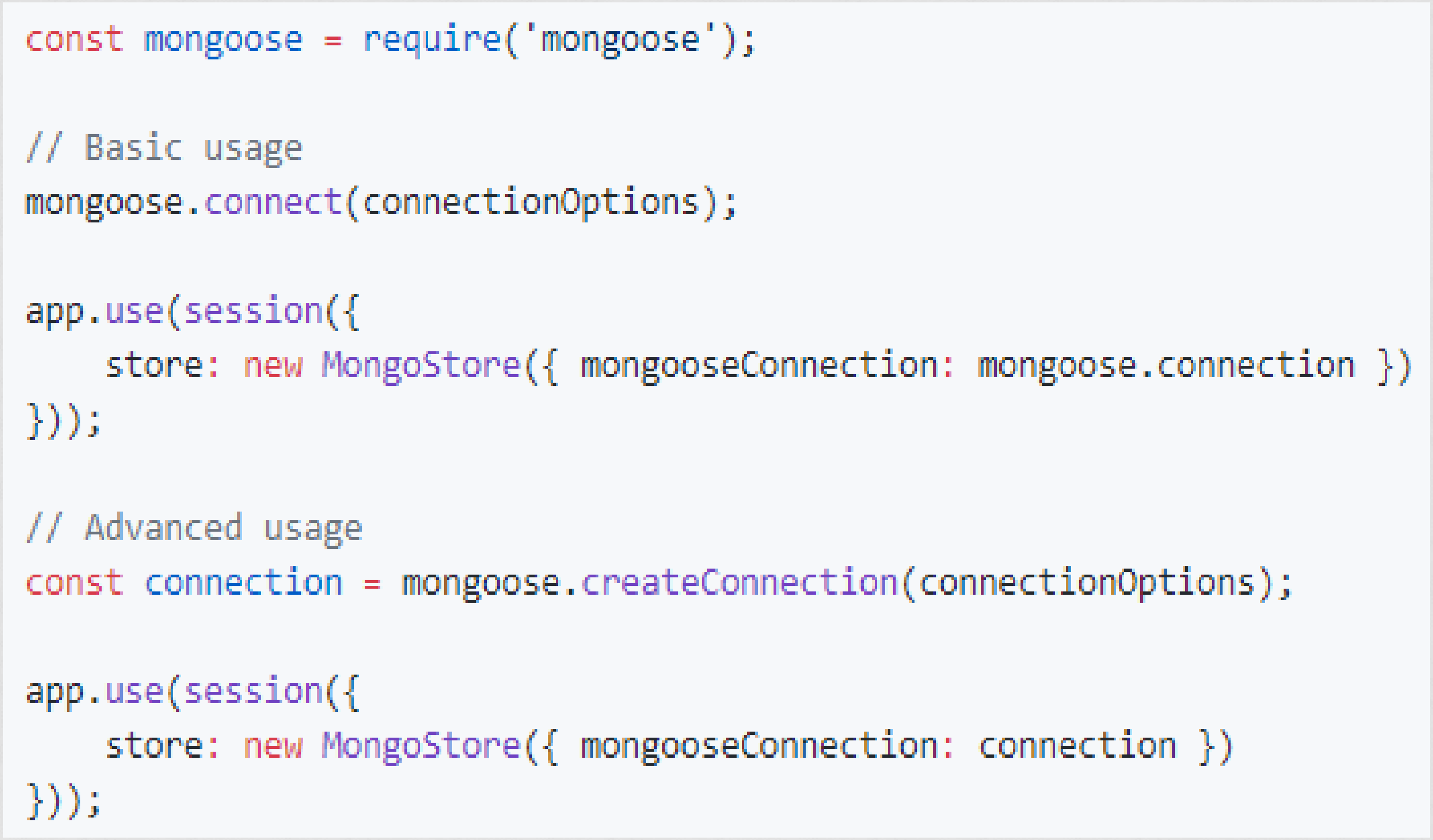
POSSIBILITY: CONNECT-MONGO

MongoDB session store for Connect and/or Express



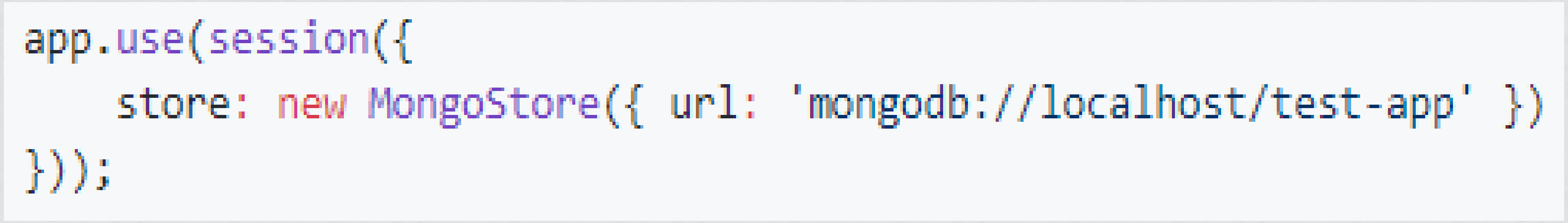
```
const session = require('express-session');
const MongoStore = require('connect-mongo')(session);

app.use(session({
    secret: 'foo',
    store: new MongoStore(options)
}));
```

POSSIBILITY: CONNECT-MONGO



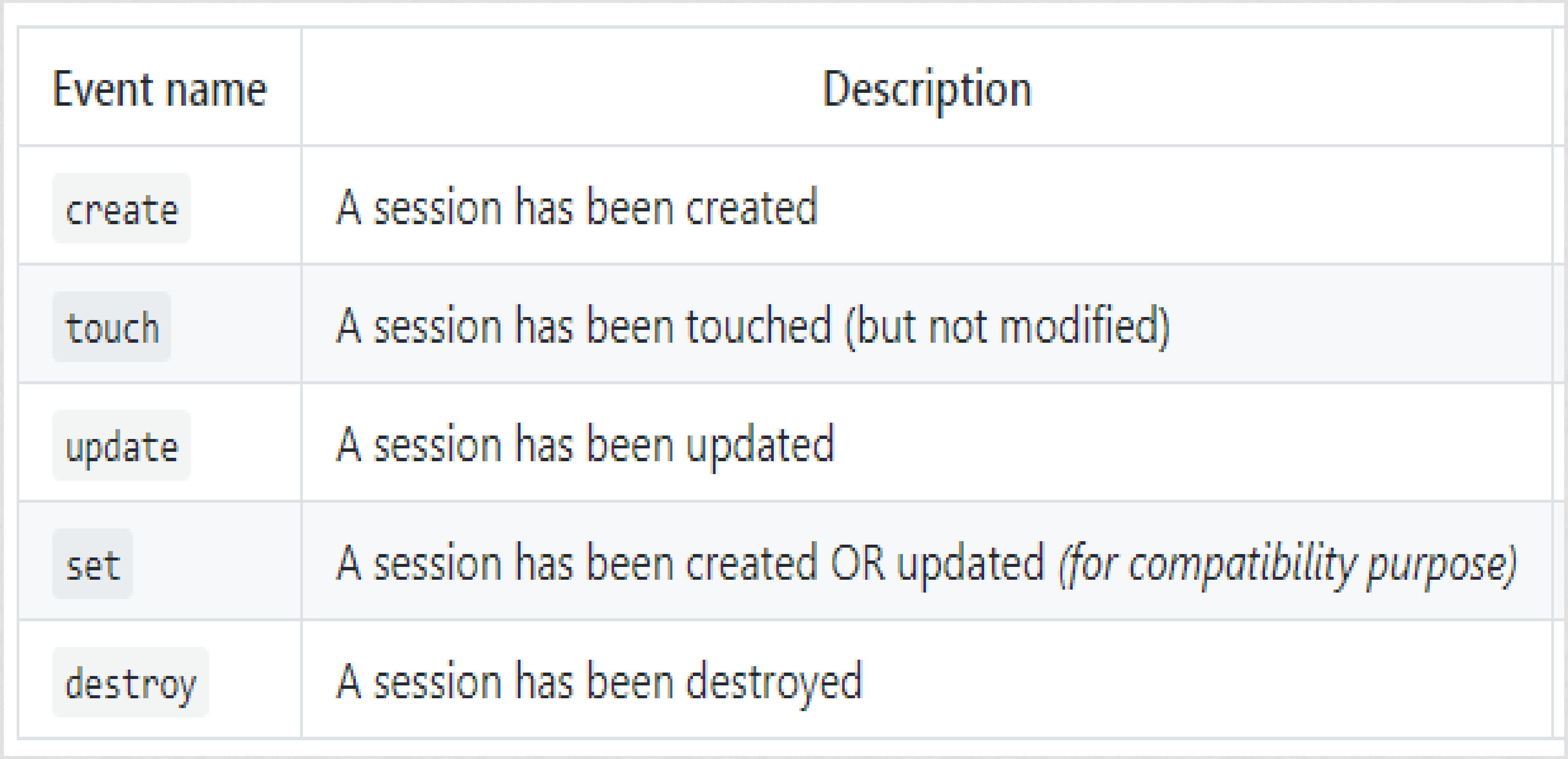
```javascript
const mongoose = require('mongoose');

// Basic usage
mongoose.connect(connectionOptions);

app.use(session({
    store: new MongoStore({ mongooseConnection: mongoose.connection })
}));

// Advanced usage
const connection = mongoose.createConnection(connectionOptions);

app.use(session({
    store: new MongoStore({ mongooseConnection: connection })
}));
```

CONNECT-MONGO:
STANDARD APPROACH



```
app.use(session({
    store: new MongoStore({ url: 'mongodb://localhost/test-app' })
}));
```

CONNECT-MONGO EVENTS TO USE



Event name	Description
`create`	A session has been created
`touch`	A session has been touched (but not modified)
`update`	A session has been updated
`set`	A session has been created OR updated *(for compatibility purpose)*
`destroy`	A session has been destroyed

SESSION STORE IN DB

The connect-mongo module stores sessions in the "sessions" collection by default

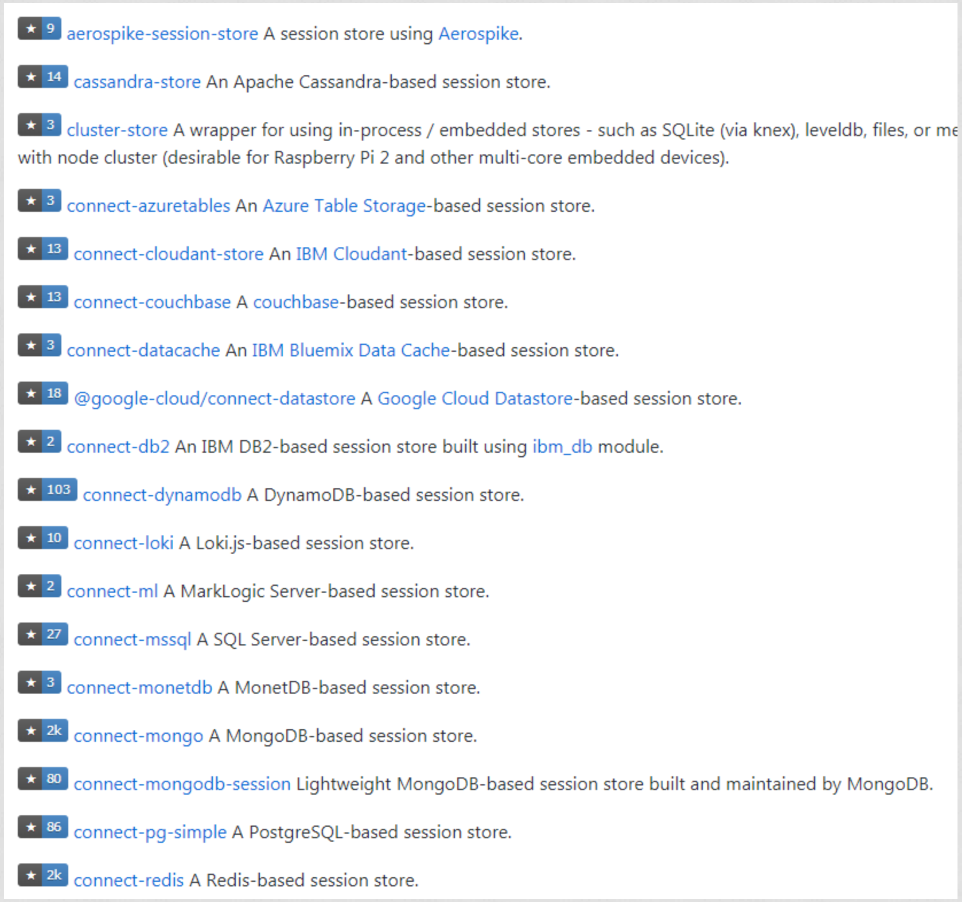
DEFAULT STORAGE: **MEMORYSTORE**

Warning: The default server-side session storage, **MemoryStore**, is *purposely* not designed for a production environment.

It will leak memory under most conditions, does not scale past a single process, **and is meant for debugging and developing.**

SEVERAL CHOICES..
SESSION STORES

https://github.com/expressjs/session#compatible-session-stores



★ 9 aerospike-session-store A session store using Aerospike.

★ 14 cassandra-store An Apache Cassandra-based session store.

★ 3 cluster-store A wrapper for using in-process / embedded stores - such as SQLite (via knex), leveldb, files, or me
with node cluster (desirable for Raspberry Pi 2 and other multi-core embedded devices).

★ 3 connect-azuretables An Azure Table Storage-based session store.

★ 13 connect-cloudant-store An IBM Cloudant-based session store.

★ 13 connect-couchbase A couchbase-based session store.

★ 3 connect-datacache An IBM Bluemix Data Cache-based session store.

★ 18 @google-cloud/connect-datastore A Google Cloud Datastore-based session store.

★ 2 connect-db2 An IBM DB2-based session store built using ibm_db module.

★ 103 connect-dynamodb A DynamoDB-based session store.

★ 10 connect-loki A Loki.js-based session store.

★ 2 connect-ml A MarkLogic Server-based session store.

★ 27 connect-mssql A SQL Server-based session store.

★ 3 connect-monetdb A MonetDB-based session store.

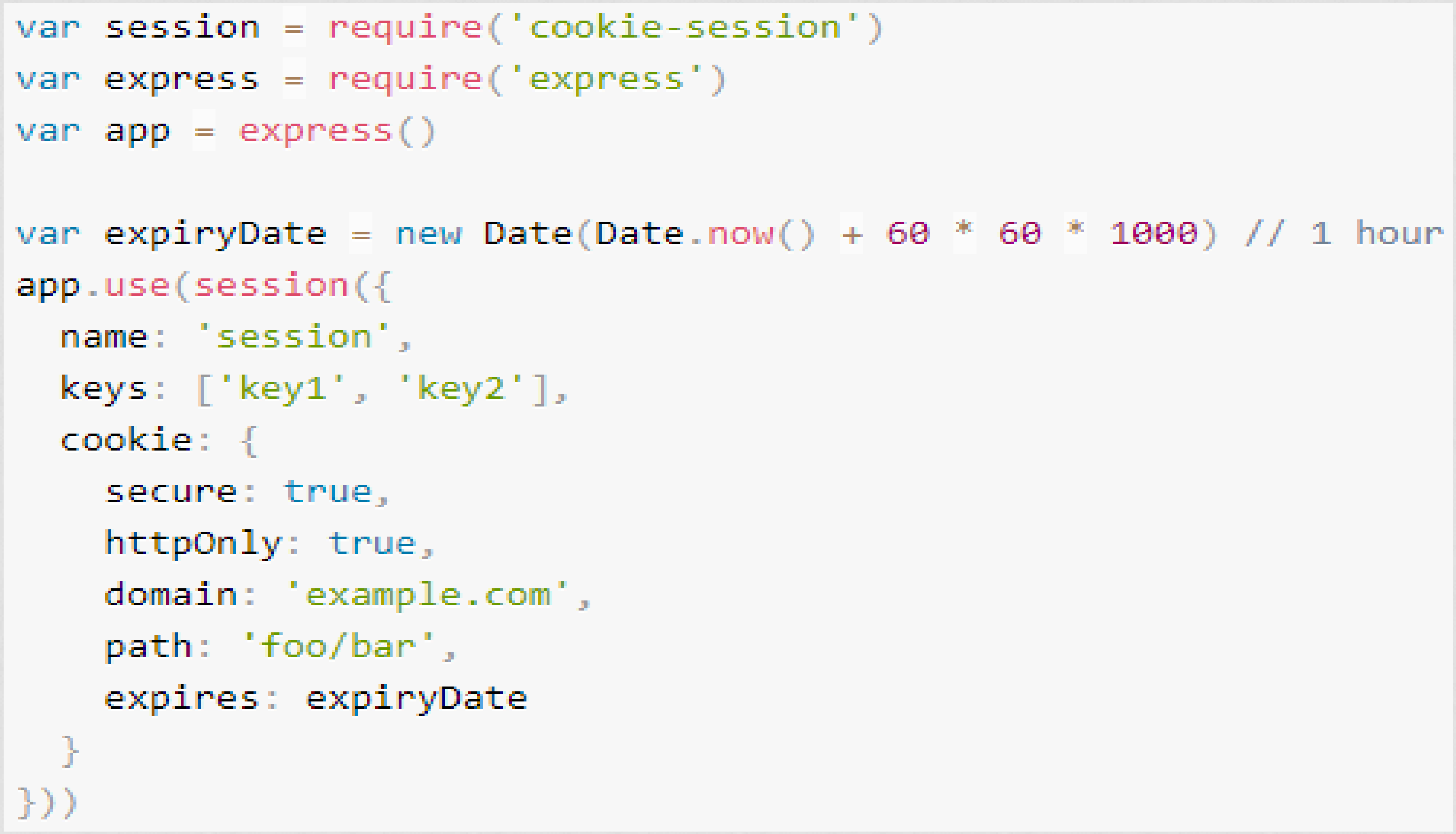
★ 2k connect-mongo A MongoDB-based session store.

★ 80 connect-mongodb-session Lightweight MongoDB-based session store built and maintained by MongoDB.

★ 86 connect-pg-simple A PostgreSQL-based session store.

★ 2k connect-redis A Redis-based session store.

COOKIE-SESSION **EXAMPLE**



```
var session = require('cookie-session')
var express = require('express')
var app = express()

var expiryDate = new Date(Date.now() + 60 * 60 * 1000) // 1 hour
app.use(session({
  name: 'session',
  keys: ['key1', 'key2'],
  cookie: {
    secure: true,
    httpOnly: true,
    domain: 'example.com',
    path: 'foo/bar',
    expires: expiryDate
  }
}))
```

EXAMPLE
MEMORY STORAGE

```
var express      = require('express');
var session      = require('express-session');
var cookieParser = require('cookie-parser');
var app          = express();
var MemcachedStore = require('connect-memcached')(session);

app.use(cookieParser());
app.use(session({
      secret   : 'some-private-key',
      key      : 'test',
      proxy    : 'true',
      store    : new MemcachedStore({
        hosts: ['127.0.0.1:11211'], //this should be where your Memcached serve
r is running
        secret: 'memcached-secret-key' // Optionally use transparent encryption
 for memcache session data
    })
}));
```

OPTIONS: **RESAVE**

```
app.use(session({
    resave: [true | false]
}));
```

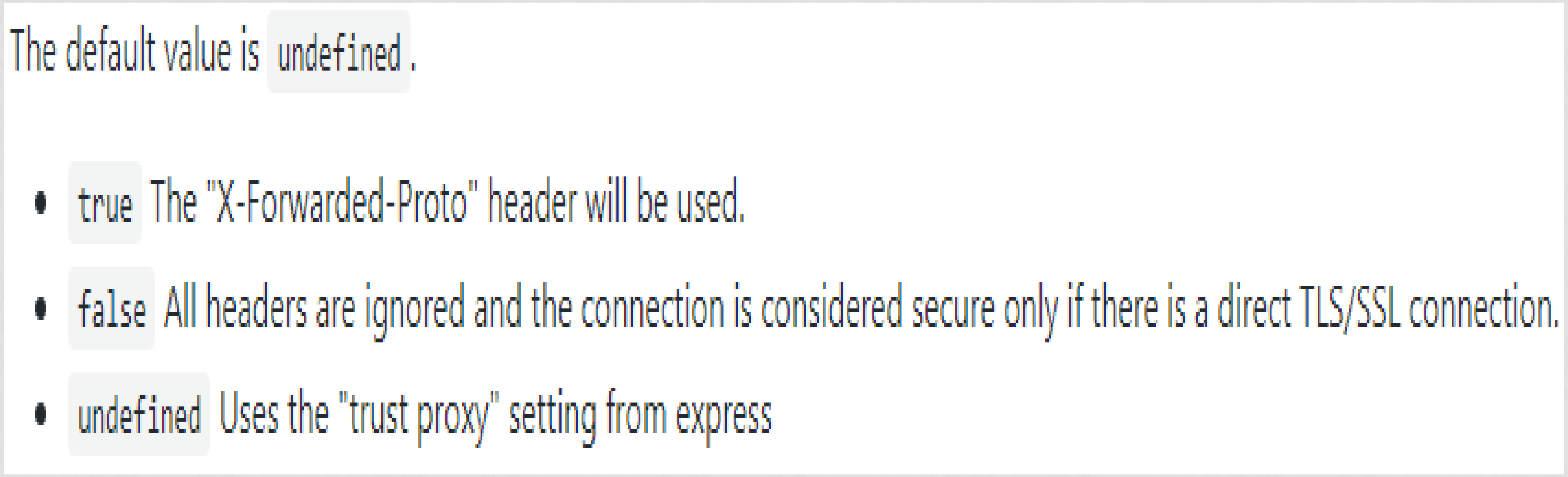
Forces the session to be saved back to the store, even if the session was never modified during the request.

OPTIONS: PROXY

```
app.use(session({
  proxy: true | false | undefined]
}));
```

Trust the reverse proxy when setting secure cookies (via the "X-Forwarded-Proto" header).

OPTIONS:
PROXY (DETAILS)



The default value is `undefined`.

- `true` The "X-Forwarded-Proto" header will be used.
- `false` All headers are ignored and the connection is considered secure only if there is a direct TLS/SSL connection.
- `undefined` Uses the "trust proxy" setting from express

OPTIONS: ROLLING

```
app.use(session({
    rolling: [true | false]
}));
```

Force a session identifier cookie to be set on every response. The expiration is reset to the original maxAge value

OPTIONS: SAVEUNINITIALIZED

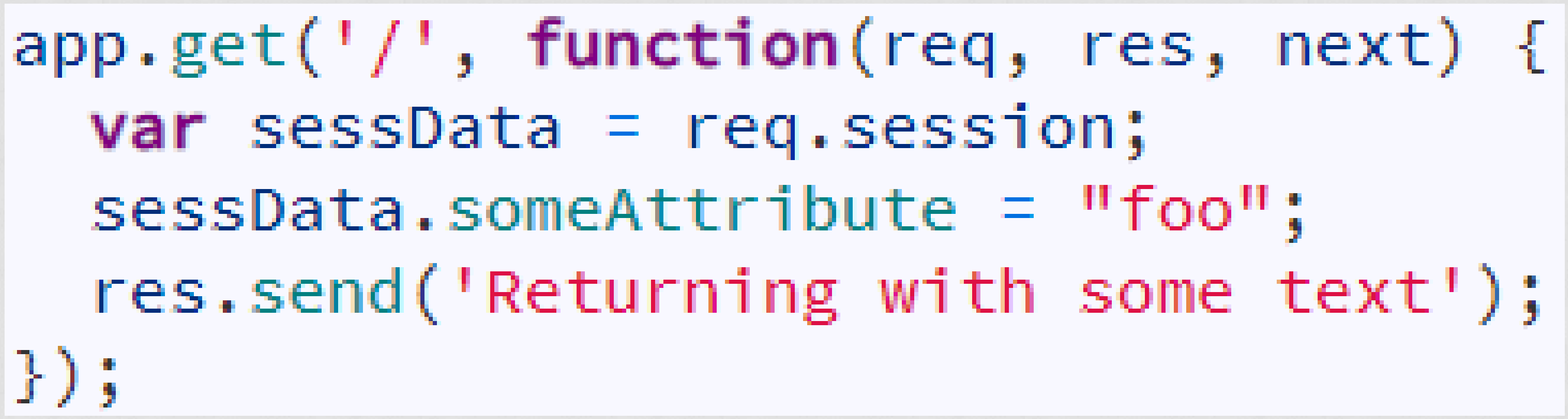
```
app.use(session({
    saveUninitialized: [true | false]
}));
```

Forces a session that is "uninitialized" to be saved to the store. A session is uninitialized when it is new but not modified.

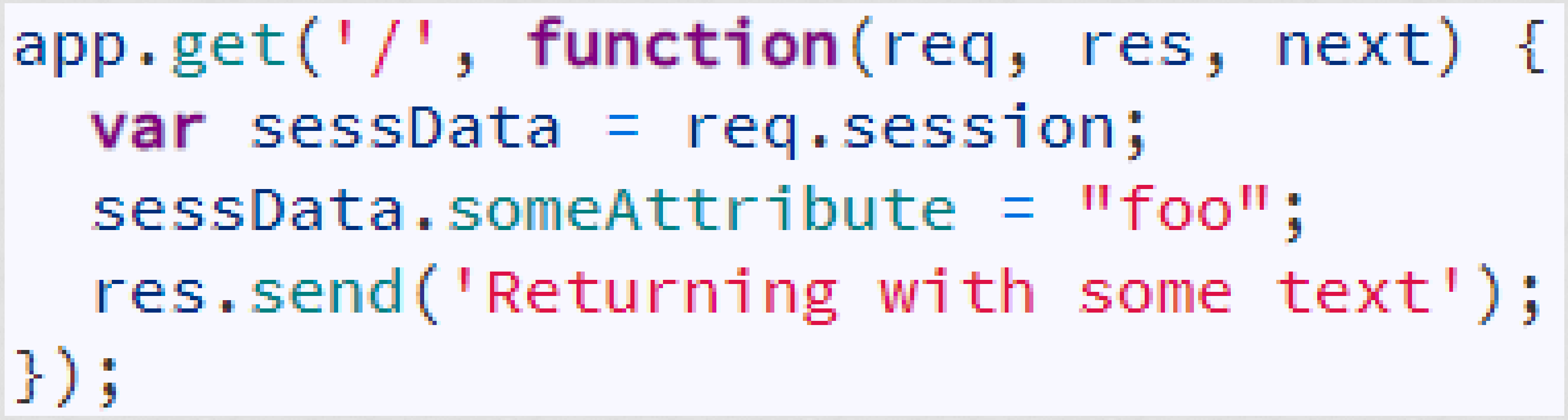
GETTING SESSION DATA

- To store or access session data, use req.session
- Serialized as JSON
- Nested JSON objects is typical

SET DATA ~~ GET DATA



```
app.get('/', function(req, res, next) {
  var sessData = req.session;
  sessData.someAttribute = "foo";
  res.send('Returning with some text');
});
```



```
app.get('/', function(req, res, next) {
  var sessData = req.session;
  sessData.someAttribute = "foo";
  res.send('Returning with some text');
});
```

SAVING SESSION DATA

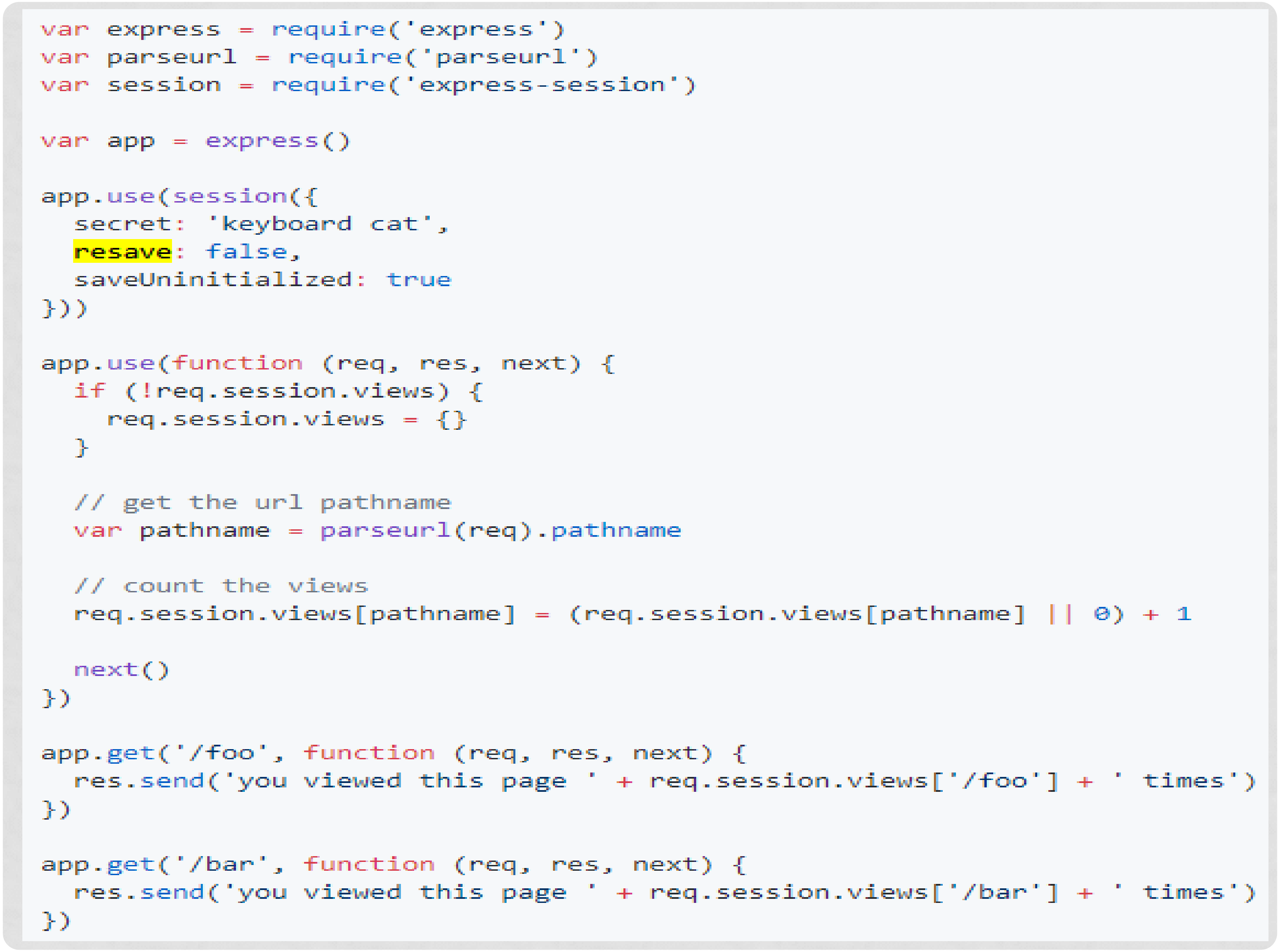
```
req.session.save( err => {  });
```

This method is automatically called at the end of the HTTP response if the session data has been altered

DESTROY

The method to call is:

```
session.destroy( err => {
    // cannot access or use
   //  session data in this callback
});
```



```
var express = require('express')
var parseurl = require('parseurl')
var session = require('express-session')

var app = express()

app.use(session({
  secret: 'keyboard cat',
  resave: false,
  saveUninitialized: true
}))

app.use(function (req, res, next) {
  if (!req.session.views) {
    req.session.views = {}
  }

  // get the url pathname
  var pathname = parseurl(req).pathname

  // count the views
  req.session.views[pathname] = (req.session.views[pathname] || 0) + 1

  next()
})

app.get('/foo', function (req, res, next) {
  res.send('you viewed this page ' + req.session.views['/foo'] + ' times')
})

app.get('/bar', function (req, res, next) {
  res.send('you viewed this page ' + req.session.views['/bar'] + ' times')
})
```