Listing 1 Express.js

```
var fs = require('fs');
var http = require('http');
var path = require('path');
var methods = require('methods');
var express = require('express');
var bodyParser = require('body-parser');
var session = require('express-session');
var cors = require('cors');
var passport = require('passport');
var errorhandler = require('errorhandler');
var mongoose = require('mongoose');

var isProduction = process.env.NODE_ENV === 'production';

// Create global app object
var app = express();

app.use(cors());

// Normal express config defaults
app.use(require('morgan')('dev'));
app.use(bodyParser.urlencoded({extended: false}));
app.use(bodyParser.json());

app.use(require('method-override')());
app.use(express.static(__dirname + '/public'));

app.use(session({secret: 'conduit', cookie: {maxAge: 60000},
    resave: false, saveUninitialized: false}));

if (!isProduction) {
  app.use(errorhandler());
}

app.use(require('./routes'));

/// catch 404 and forward to error handler
app.use(function (req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// finally, let's start our server...
var server = app.listen(process.env.PORT || 3000, function () {
  console.log('Listening on port ' + server.address().port);
});
```

Listing 2 JavaScript

```
// Expression bodies
var odds = evens.map(v => v + 1);
var nums = evens.map((v, i) => v + i);
var pairs = evens.map(v => ({even: v, odd: v + 1}));

// Statement bodies
nums.forEach(v => {
  if (v % 5 === 0)
    fives.push(v);
});

// Lexical this
var bob = {
  _name: "Bob",
  _friends: [],
  printFriends() {
    this._friends.forEach(f =>
        console.log(this._name + " knows " + f));
  }
}
```

Listing 3 JavaScript

```
Number.EPSILON
Number.isInteger(Infinity); // false
Number.isNaN("NaN"); // false

Math.acosh(3); // 1.762747174039086
Math.hypot(3, 4); // 5
Math.imul(Math.pow(2, 32) - 1, Math.pow(2, 32) - 2) // 2
;
"abcde".includes("cd"); // true
"abc".repeat(3); // "abcabcabc"

Array.from(document.querySelectorAll('*')); // Returns a real
    Array
Array.of(1, 2, 3) // Similar to new Array(...), but without
    special one-arg behavior
    [0, 0, 0].fill(7, 1) // [0,7,7]
    [1, 2, 3].find(x => x == 3) // 3
    [1, 2, 3].findIndex(x => x == 2) // 1
    [1, 2, 3, 4, 5].copyWithin(3, 0) // [1, 2, 3, 1, 2]
    ["a", "b", "c"].entries() // iterator [0, "a"], [1,"b"],
        [2,"c"]
    ["a", "b", "c"].keys() // iterator 0, 1, 2
    ["a", "b", "c"].values(); // iterator "a", "b", "c"

Object.assign(Point, { origin: new Point(0,0) });
```

Listing 4 Node.js

```
var fs = require('fs');
var path = require('path');
var http = require('http');
var request = require('request');
var TMP_FILE_PATH = path.join(path.sep, 'tmp', 'foo');

// write a temporary file:
fs.writeFileSync(TMP_FILE_PATH, 'foo bar baz quk\n');

http.createServer(function (req, res) {
  console.log('the server is receiving data!\n');
  req.on('end', res.end.bind(res)).pipe(process.stdout);
}).listen(3000).unref();

fs.createReadStream(TMP_FILE_PATH)
    .pipe(request.post('http://127.0.0.1:3000'));
```

Listing 5 Jasmine

```
describe('built-in matchers', function() {
  describe('toBeTruthy', function() {
    it('passes if subject is true', function() {
      expect(true).toBeTruthy();
      expect(false).not.toBeTruthy();
    });
  });

  describe('toBeFalsy', function() {
    it('passes if subject is false', function() {
      expect(false).toBeFalsy();
      expect(true).not.toBeFalsy();
    });
  });

  describe('toBeDefined', function() {
    it('passes if subject is not undefined', function() {
      expect({}).toBeDefined();
      expect(undefined).not.toBeDefined();
    });
  });

  describe('toBeNull', function() {
    it('passes if subject is null', function() {
      expect(null).toBeNull();
      expect(undefined).not.toBeNull();
      expect({}).not.toBeNull();
    });
  });
});
```