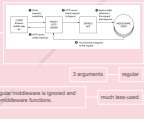
Node.js

- Ch8. MongoDB
  - Non-Relational Database (No SQL)
    - refer to a specific class
    - No schema
  - Mongoose
    - bcrypt : save password
- Ch9. Test Express applications
  - how to reduce errors
    - compiler and syntax checker
    - automated testing
    - TDD(Test-Driven Development)
      - red-green-refactor
        - 1. Red : When there is a failed test
        - 2. Green : Pass all tests
        - 3. Refactor : arrange code
        - 4. repeat the steps
    - Node.js Test
      - actual code(user-written)
        - everythings to test
      - test code(user-written)
        - test actual code
      - test runner(not user-written)
        - execute in computer
          - Chai : assertion library
          - Mocha : test framework
          - ```
            {
              "private" : true,
              "dependencies" : {
                "chai" : "*",
                "mocha" : "1.0.0"
              },
              "scripts" : {
                "test" : "mocha"
              }
            }
            ```
            In package.json
    - Express application
      - SuperTest
    - HTML test
      - Cheerio
- Ch10. Security
  - No bug
    - test
    - review codes
      - JSHint
  - Protecting Users
    - Using HTTPS
      - Express Middleware
        - Force users to HTTPS
          - express-enforces-ssl
            - If the request goes over HTTPS, connect to middleware or route
            - If not, redirection to the HTTPS version
            - 1. Enable the "trust proxy" setting
            - 2. call middleware
            - ex)
              ```
              var enforcer = require('express-enforces-ssl');
              app.enable('trust proxy');
              app.use(enforcer());
              ```
        - Keep users on HTTPS
          - HTTP Strict Transport Security (HSTS): HTTP header that allows HTTPS to be used for some time
            - `Strict-Transport-Security: max-age=31536000`
            - ex)stay HTTPS for 1 year
    - XSS(Cross-Site Scripting)
      - CSRF(Cross-Site Request Forgery)
        - 1. Generates a random CSRF token each time a user requests data
        - 2. Check for random tokens every time you deal with that data
      - csurf Middleware(Express team)
        - 1. Add a method to the request object called req.csrfToken (for example, send this token each time you send a form)
        - 2. If the request is something other than a GET, validate the request by looking for a parameter named _csrf.
          - If not valid, generate error
  - How to maintain dependencies
    - Self-checking code
    - check latest version
    - Check nsp(Node Security Project)
      - npm install -g nsp
      - nsp audit-package at package.json
  - conflictserver
    - Forever : try to restart app
  - etc
    - X-Powered-By
      - app.disabled("x-powered-by");
    - Clickjacking: cheat like clicking another part
- Ch11. Deployment : asset and Heroku
  - Expand Express app
    - LESS(easier than CSS)
      - Variables
      - Functions
      - Nesting selectors
      - Mixins
      - Includes
    - Heroku
      - distribute Express application
    - Browserify
      - Packaging Javascript to run in a browser
    - Grunt
      - CSS -> LESS compile / package JavaScript with Browserify
    - Connect-assets
      - connect JavaScript and CSS,compile able to reduce
        - CoffeeScript, Stylus, LESS, SASS, support some EJS
- Ch12. Best practice
- Ch1. Express
  - asynchronous
- Ch2. Basic of Node
  - require
    - requiring built-in modules
      - When the browser makes a request to the server, it asks for the URL
      - parse: Extract information using URL string
      - URL module
        - require function
          - ex)
            ```
            var url = require('url');
            var parsed0Obj = url.parse('http://www.example.com/profile?name=barry');
            console.log(parsed0Obj.protocol); // http:
            console.log(parsed0Obj.hostname); // www.example.com
            console.log(parsed0Obj.query); // name=barry
            ```
    - requiring third-party modules
      - package.json
        - ex) A simple package.json file
          ```
          {
            "name" : "express",
            "version" : "0.0.1",
            "dependencies" : {
              "express" : "3.4.8"
            }
          }
          ```
      - npm
        - Node helper
    - defining your own modules
      - Assigned to module.exports(global object)
  - external resources handled by Express
    - file system
    - network
  - http module
    - createServer
      - ```
        var http = require('http');
        var server = http.createServer(function (request, response) {
          console.log('Request received');
          response.end('Hello World\n');
        });
        server.listen(8000);
        console.log('Server running at http://127.0.0.1:8000/');
        ```
        ex1)
      - ```
        var http = require('http');
        var server = http.createServer(function (request, response) {
          console.log('Request received');
          response.end('Hello World\n');
        });
        var port = process.env.PORT || 1337;
        server.listen(port);
        console.log('Server running at http://127.0.0.1:' + port + '/');
        ```
        ex2)
- Ch3. Foundation of Express
  - Express
    - abstraction layer on top of embedded http server
    - main function(무슨 기능)
      - middleware stack as an effective array
      - request handler(vanilla Node)
        - common : Accept requests and send response
        - difference : Perform tasks in order, not one handler
        - without Middleware
          - Request -> Request handler function -> Response
        - with Middleware
          - Request -> Array of request handler functions(Middleware stack) -> Response
      - Middleware(Ch4)
        - ```
          var express = require('express');
          var app = express();
          var myLogger = function (req, res, next) {
            console.log('LOGGED');
            next();
          };
          app.use(myLogger);
          app.get('/', function (req, res) {
            res.send('Hello World');
          });
          app.listen(3000);
          ```
        - advantage : Easy to create and rearrange, Split application, third-party middleware
        - passive
          - next() : call next middleware
        - logger(morgan)
          - Know what user is doing
          - Know when to request server, server response time
          - Helpful during development
          - ```
            var logger = require('morgan');
            app.use(logger('dev'));
            ```
      - Routing(Ch5)
        - Mapping specific http requests to specific functions(running specific code)
          - app.get
            - ```
              var express = require('express');
              var app = express();
              // Respond with "Hello World!" on the homepage
              app.get('/', function (req, res) {
              res.send('Hello World!');
              });
              // Accept login.gif only
              app.get('/login.gif', function (req, res) {
              res.send('Login');
              });
              ```
            root route
      - Extending request and response
        - Additional methods and properties
          - redirect
            - ```
              response.redirect("/helloworld");
              response.redirect("http://express.com");
              ```
          - sendFile
            - ```
              response.sendFile("path/to/our_img.gif");
              ```
            send all files
      - View(Ch7)
        - can render HTML dynamically
          - ```
            var express = require('express');
            var app = express();
            var bodyParser = require('body-parser');
            var cookieParser = require('cookie-parser');
            var session = require('express-session');
            var multer = require('multer');
            var errorHandler = require('errorhandler');
            var morgan = require('morgan');
            var serveStatic = require('serve-static');
            var path = require('path');
            var fs = require('fs');
            var http = require('http');
            var app = express();
            var logDirectory = path.join(__dirname, 'log');
            var accessLogfile = fs.createWriteStream(logDirectory + '/access.log', {flags: 'a'});
            var errorLogfile = fs.createWriteStream(logDirectory + '/error.log', {flags: 'a'});
            app.set('port', process.env.PORT || 3000);
            app.set('views', path.join(__dirname, 'views'));
            app.set('view engine', 'ejs');
            app.disable('x-powered-by');
            app.use(morgan('dev'));
            app.use(morgan('combined', {stream: accessLogfile}));
            app.use(bodyParser.urlencoded({extended: true}));
            app.use(cookieParser('secret'));
            app.use(session({secret: 'secret', cookie: {maxAge: 60000}, resave: false, saveUninitialized: false}));
            app.use(multer({dest: path.join(__dirname, 'public/upload/temp')}));
            app.use(serveStatic(path.join(__dirname, 'public'), {maxAge: 86400000}));
            app.locals.title = 'My App';
            app.locals.strftime = require('strftime');
            app.locals.email = process.env.EMAIL;
            app.locals.delete = function(obj, prop) {
              var clone = {};
              for
            ```
            setting up views with Express
          - ```
            var express = require('express');
            var app = express();
            app.get('/', function (req, res) {
              res.render('index', {title: 'Hey', message: 'Hello there!'});
            });
            app.listen(3000);
            ```
            rendering a view from Express
          - ejs

Ch4. Middleware

- flow
  - function(request, response, next)
  - call response.end -> finish request
  - 
- type
  - regular
    - 3 arguments
  - much less-used
    - When the app is in error mode, all regular middleware is ignored and Express only runs the error handling middleware functions.
      - To start error mode, call next as an argument.

Ch5. Routing

- expression
  - `GET / olivia http/1.1`
    - GET is verb
    - olivia http is URI
    - 1.1 is version
    - code example
      ```
      var express = require('express');
      var app = express();
      // Accept GET requests at the root path
      app.get('/', function (req, res) {
        res.send('Got a GET request at /');
      });
      app.put('/', function (req, res) {
        res.send('Got a PUT request at /');
      });
      ```
- router to middleware
- feature
  - /users/
    - the simplest parameter
      ```
      app.get('/users/:userId', function(req, res)
        var userId = parseInt(req.params.userId, 10);
      ```
    - regular expression
      ```
      app.get(/users/([0-9]+)/, function(req, res)
        var userId = parseInt(req.params[0], 10);
      ```
  - query
    - req.query.arg
      - "arg=something"
        - string
      - "arg=something&arg=somethingelse"
        - array
    - Handling a search query string
      ```
      app.get('/search', function(req, res)
        var q = req.query.q || "default";
      ```
  - middleware
    - behave like middleware
    - Routers allow you to split a large application into several smaller applications for later use together
      - part of main app
        ```
        var express = require('express');
        var app = express();
        var apiRouter = express.Router();
        var usersRouter = express.Router();
        var ordersRouter = express.Router();
        // Mounting the routers at /api and /admin
        app.use('/api', apiRouter);
        app.use('/admin', adminRouter);
        // Mounting the routers at /users and /orders
        apiRouter.use('/users', usersRouter);
        apiRouter.use('/orders', ordersRouter);
        // Defining some routes
        usersRouter.get('/', function(req, res, next) {
          res.send('GET /api/users');
        });
        ordersRouter.get('/:id', function(req, res, next) {
          res.send('GET /api/orders/:id');
        });
        ```
        sample router definition example
    - Because it uses the API, URLs beginning with / api are sent directly to the router.
    - cannot run alone
      - difference
- Serving static files
  - ```
    var express = require('express');
    var app = express();
    // Serve static files from /public
    app.use(express.static('public'));
    // Serve static files from /assets
    app.use(express.static('assets'));
    // Serve static files from /uploads
    app.use(express.static('uploads'));
    ```
    Serving static files from multiple directories
  - The first command in app.use is executed but the second command is not executed
    - Can be executed without conflict by modifying the above command
    - ```
      app.get('/urlx', express.static(path.join(__dirname, 'public')));
      app.get('/urlx', express.static(path.join(__dirname, 'assets')));
      ```
- HTTPS
  - HTTP+ secure layer
    - TLS(Transport Layer Security)
      - certificate
      - special property
        - public key
        - private key
      - Encrypt the message with the public key, send the message to the eavesdropper, and decrypt the message using the private and public key.
    - OpenSSL
      - SSL(Secure Sockets Layer)
      - `openssl genrsa -out privatekey.pem 1024`
        - Create private key as privatekey.pem
      - `openssl req -new -key privatekey.pem -out request.pem`
        - certificate signing request
      - When receiving a certificate, you can use the built-in HTTPS module (HTTP + private key + certificate).
  - ```
    var fs = require('fs');
    var http = require('http');
    var express = require('express');
    var app = express();
    var port = 3000;
    var sslport = 23000;
    var options = {
      key: fs.readFileSync('./ssl/privatekey.pem'),
      cert: fs.readFileSync('./ssl/certificate.pem'
    };
    var server = http.createServer(options, app);
    var io = require('socket.io')(server);
    var fs = require('fs');
    var path = '/home/ec2-user/';
    app.get('/', function(req, res){
      res.sendFile(path + 'index.html');
    });
    io.on('connection', function(socket){
      console.log('a user connected');
      socket.on('disconnect', function(){
        console.log('user disconnected');
      });
      socket.on('chat message', function(msg){
        console.log('message: ' + msg);
        io.emit('chat message', msg);
      });
    });
    server.listen(port, function(){
      console.log('listening on *:' + port);
    });
    var server2 = express().listen(sslport);
    var io2 = require('socket.io')(server2);
    var fs = require('fs');
    var path = '/home/ec2-user/';
    io2.on('connection', function(socket){
      console.log('a user connected');
      socket.on('disconnect', function(){
        console.log('user disconnected');
      });
      socket.on('chat message', function(msg){
        console.log('message: ' + msg);
        io2.emit('chat message', msg);
      });
    });
    server2.listen(sslport, function(){
      console.log('listening on *:' + sslport);
    });
    ```
    ex)

Ch6. Building APIs

- JSON API
  - Use middleware and routing to process requests, parse them, and respond with JSON objects and HTTP status code
    - Steps
      - Metadata description of the application in package.json
      - Create an app.js file containing all the code
      - Create an Express application in app.js and connect to a single route
- HTTP
  - verb
    - GET
      - Read
        - Get resources, Load homepage or images, Immutable(인증 X?)
    - POST
      - Create
        - Write records to the server without modifying existing records, such as requesting server status changes, posting pictures, or signing up for a new account.
    - PUT
      - Update
        - update, push changes
    - DELETE
      - Delete
        - delete
  - Status code
    - classification
      - 1xx : hold on
      - 2xx : here you go
        - ex) 200(default) : OK : Load image, JSON response successfully
      - 3xx : go away
      - 4xx : you messed up
        - ex) 404 : Resource not found : expired link, incorrectly entered URL
      - 5xx : I messed up
    - setting
      - `res.status(404);`
        - code in Express
      - `res.status(404).send('Resource not found');`
        - sending some JSON

Ch7. Views and templates : Pug and EJS

- View engine : View rendering
  - Pug(Jade)
    - HTML replaceable, clean writing
      - ```
        doctype html
        html(lang="en")
          head
            title Hello world!
          body
            h1 This is a Pug example
            Hello World!
            p Bye.
        ```
        Pug example)
    - Main
      - The ability to reduce repetitive tasks
  - EJS(Embedded JavaScript)
    - <%= expression %>
    - EJS template
      - Useful because you can add headers and footers to the page and separate widgets
    - EJS filter
      - ejs.filters.
    - It doesn't completely replace HTML, but complements it with new features
    - call response.render => dynamic rendering

XMind : ZEN | Trial Mode