Coding Standard Document for BooXChange

Software Engineering Course, Sem 6
Prepared on 16th April 2020

PREPARED FOR
Prof. Khushru Doctor

Version 1.0

Prepared By :

Jainam Chhatbar - 1741002	Aman Dave - 1741003
Harshiv Patel - 1741005	Aditya Shah - 1741007
Dhairya Dudhatra - 1741058	Meet Modi - 1741071
Shantanu Sheth - 1741088	

Table of Contents

Braces

- Braces must be used for all control structures (ie. if, else, etc) even if there is only one statement inside the body
- The opening braces must be on the same line as the control statement

```
if (!product) {
        return res.status(404).send();
}
```

Indentation

- Every time a new block of code is started, the indent must move to the next level which is 1 tab (equivalent to 4 characters)
- When a block of code ends, the indent must return to the previous level

```
try {
    const user = await User.findByCredentials(
        req.body.email,
        req.body.password
    );
    const token = await user.generateAuthToken();
    res.send({ user, token });
} catch (e) {
    console.log(e);
    res.status(400).send();
}
```

Statements

- One statement per line
    - Exactly one statement per line which must be followed by a line-break

```
const express = require('express');
const ModelProducts = require('../models/product');
const ModelLog = require('../models/log');
```

- 80 characters per line
    - Maximum 80 characters per line for better readability
- Line-Wrapping
    - If a statement does not fit in a single line, it must be wrapped as follows:
        - Break after comma
        - Break after operator

```
const user = await User.findOne({
     _id: decodedId,
     'tokens.token': token,
 });
```

- Whitespace
    - Vertical whitespace
        - A single blank line must be used to separate different methods or properties on an object
        - Blank lines must not be used at the start or end of a function body

```
const express = require('express');
const ModelLog = require('../models/log');

const router = express.Router();

router.get('/log', async (req, res) => {
```

```
try {
      const logs = await ModelLog.find({});
      res.send(logs);
} catch (e) {
      res.status(500).send();
}
});

module.exports = router;
```

- Horizontal whitespace
  - Single space character must be used after the comma or semicolon. Space characters before these characters are not allowed
  - Single space character must be used before and after the colon while defining an object
  - Single space character must be used before and after the assignment operator while assigning value to any variable

```
const LogSchema = mongoose.Schema({
      type : {
      type : String,
      required : true,
      },
      time : Date,
      itemid : String,
      itemtitle : String,
});
```

- Quotes
  - Single colons must be used everywhere
  - Double colons can be used when writing JSON

```
const userRouter = require('./routes/user');
const authRouter = require('./routes/auth');

const env = process.env.NODE_ENV || 'development';
```

- Declaration of variables
  - By default, the variable must be defined using const keyword unless a variable needs to be redefined
  - Declare one variable per statement var/const statement
  - When declaring an array or object, the use of trailing commas must be made to separate elements or properties.

```
const allowedUpdates = [
      'name',
      'email',
      'address1',
      'address2',
      'age',
      'password',
];
```

Naming Conventions

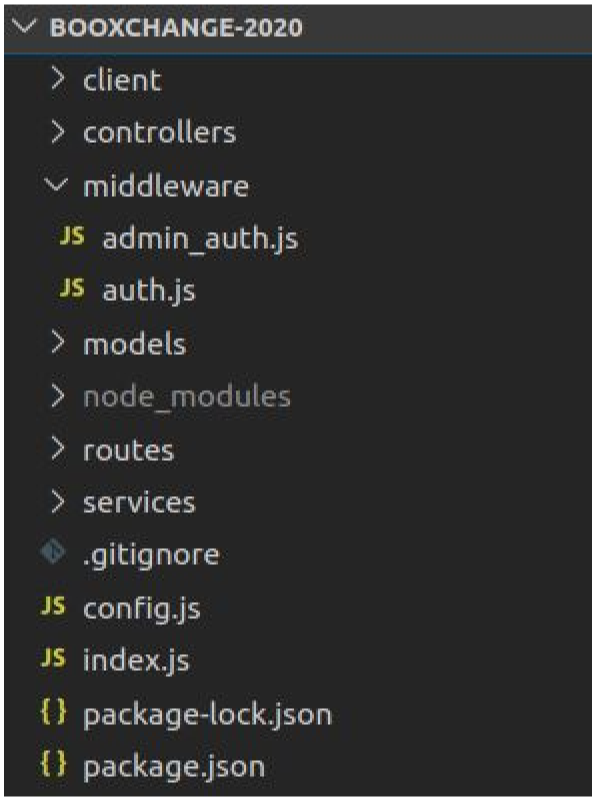
- **lowerCamelCase** must be used for variables, properties and function names

```
const express = require('express');
const router = express.Router();
```

- **UpperCamelCase** must be used for class names

```
const ModelProducts = require('../models/product');
const ModelLog = require('../models/log');
```

- **UPPERCASE** along with underscores must be used for constants
- **lowercase** along with underscores must be used for file names



```
BOOXCHANGE-2020
  client
  controllers
  middleware
    admin_auth.js
    auth.js
  models
  node_modules
  routes
  services
  .gitignore
  config.js
  index.js
  package-lock.json
  package.json
```

Use of descriptive conditions

- Any non-trivial condition must be assigned to a variable instead of directly passing the complex condition to the conditional statement

```
const isValidOperation = updates.every((update) =>
    allowedUpdates.includes(update)
);

if (!isValidOperation) {
    res.status(400).send({ error: 'Invalid updates' });
}
```

Comments

- Slashes must be used for both single line and multi line comments
- Comments must be used to explain difficult segments of code that are not readily apparent
- Comments must be avoided to explain segments of code that are readily apparent and may seem trivial to even a beginner programmer.

```
//Verify this email and password, call done with the user
//if it is the correct email and password
//otherwise, call done with false
User.findOne({ email: email }, function (err, user) {
    if (err) {
        return done(err);
    }
    if (!user) {
        return done(null, false);
    }

    //compare passwords - is password equal to user.password?
    user.comparePassword(password, function (err, isMatch) {
        if (err) {
            return done(err);
        }
        if (!isMatch) {
            return done(null, false);
        }
        return done(null, user);
    });
});
```

Dependencies

- All dependencies must be stated at the start of the file by using import or require statements.

```
const express = require('express');
const ModelOrders = require('../models/order');

const router = express.Router();

router.post('/add/orders', async (req, res) => {
   const order = new ModelOrders(req.body);
   try {
       await order.save();
       res.status(200).send();
   } catch (e) {
       res.status(500).send();
   }
});

module.exports = router;
```