# Node.JS Style Guide
# CSE 115A: NoteQuest

**<u>Formatting for the style guide:</u>**

- Arial, 12 pt font
- Bold headers per rule, followed by an short explanation, example of the code example (if applicable)
- For code format, use codeblocks google drive extension setting the language to *javascript*, and the theme to *vs2015*

**<u>Style Guide</u>**

**Newlines**

Use UNIX-style newlines (`\n`), and a newline character as the last character in any file. Usage of windows newlines (`\r\n`) are forbidden.

**No trailing white space**

No lines should have trailing white spaces, every line should essentially end on a newline character.

**Limit of 80 characters per line**

No more than 80 characters per line. This will prevent large lines from occurring; allowing for neater and clearer code. If a line exceeds 80 characters, start a newline, and indent the line to show that it is just a continuation of the previous line.

Example:

```javascript
this.app.get('/create_class_run_on_sentence_here_it_is_an_example_troll',async
    (req,res)=>{
```

**Use double quotes**

Always use double quotes

Correct:

```javascript
var test = "cool"
```

Wrong:

```javascript
var test = 'cool'
```

**Opening braces go on the same line**

Opening braces go on the same line as the statement.

Correct:

```
if(true){
    console.log("works"
}
```

Wrong:

```
if(true)
{
    console.log("works"
}
```

**Avoid the use of callbacks whenever possible, use async/await in general**
For the majority of the routes, we should be using async/await to handle concurrency. The purpose of this is two avoid two things: the first is the term coined as *callback hell*, which implies having multiple levels of callback functions, and the second is to avoid messy code because *async/await* makes the code look much cleaner and concise.

**Do not use semicolons to indicate the end of a statement**
Statements should not have a semicolon termination.

Correct:

```
this.app.get("/some_function")
this.app.get("/other_function")
```

Wrong:

```
this.app.get("/some_function");
this.app.get("/other_function");
```

**Use lowerCamelCase for variables**
Variables take advantage of *lowerCamelCase*. Names should also have a clear meaning (more on this later).

Correct:

```
var compoundName = "cool"
```

Wrong:

```
var compound_name = 'cool'
```

**When creating variables, do not do a multi-declaration**
Whenever, we initialize a variable, do not do multi-declarations, or multi-initializations.

Correct:

```
var temp = 4
var temp2 = 5
```

Wrong:

```
var temp, temp2 = 4, 5
```

**Variable constants should be all caps**
For variable constants declare them with all caps

Correct:

```
const TEMP = 4
```

Wrong:

```
var temp = 4
```

**Use UpperCamelCase for Class Names**
Class Names should always use *UpperCamelCase*. Class names should be clear on their meaning. Followed by a short description on the next line on what the class is intended to be used for.

Correct:

```
class ServerA{
    //server stuff
}
```

Wrong:

```
class serverA{
    //server stuff
}
```

**Use underscore_spacing for functions/method names**
Take advantage of underscore spacing for any use of a function or method name. This implies between each word there is an underscore.

Correct:
```
function test_this()
```

Wrong:
```
function testThis()
```

**Creating functions define a variable name, and set it equal to a function**
Whenever we want to create a function, assign a variable equal to the function.

Correct:
```
var test = function(){
    ...
}
```

Wrong:
```
function test(){
    ...
}
```

**Use slashes for comments**
Use slashes for both single line and multi line comments. Make sure comments explain higher level mechanisms or to clarify difficult segments.

Correct:
```
//test
```

Wrong:
```
/* test*/
```

**When working on tasks, create a new branch**

Allows for more people to work on the source code at once. Whenever, we start a new task off of our monday board, we create a new branch, when that branch is complete, we pull from *master* to get the latest version, and then merge our branch with the current *master*.

**When implementing new routes, create a new file to contain the async function**
Whenever we specify another route in our server, always import it from another file. The reason for doing this will allow for cleaner code, as well as more modularity between routes. As well when dividing up the work it makes it easier to merge our branch with *master* because everyone will be working on their own file; keeping the source code separate from one another, thus reducing the number of merge conflicts.

**server.js should only have routes**
By no means should any function source code should exist in this file. This is to ensure modularity. For implementation functions, refer to the previous rule.

**Surround any call to a Database or Google API with try/catch**
This is done to prevent getting runtime errors or if the call to the database/google api fails, it will not crash the server.

**Do not extend built-in prototypes**
Do not extend the prototype of native node.js objects

Correct:

```
var a = []
if(!a.length){
    console.log("winning")
}
```

Wrong:

```
Array.prototype.empty = function(){
    return !this.length
}
```

**Sources:**

- https://github.com/felixge/node-style-guide
- https://google.github.io/styleguide/javascriptguide.xml