1. **Formatting**

* **4 spaces for indentation -** Use 4 spaces for indenting your code
* **Newlines -** Use UNIX-style newlines (\n), and a newline character as the last character of a file
* **Use semicolons**
* **80 characters per line -** Limit your lines to 80 characters.
* **Use single quotes**

Use single quotes, unless you are writing JSON. This helps you separate your objects’ strings from normal strings.

var foo = ‘bar’;

* **Opening braces go on the same line**

Your opening braces go on the same line as the statement.

if (true) {  
 console.log(‘winning’);  
}

if (true) {  
 console.log(‘winning’);  
} else if (false) {  
 console.log(‘this is good’);  
} else {  
 console.log(‘finally’);  
}

* **Declare one variable per var statement**

Declare one variable per var statement, it makes it easier to re-order the lines.

*Right:*

var keys = [‘foo’, ‘bar’];var values = [23, 42];var object = {};

*Wrong:*

var keys = [‘foo’, ‘bar’],values = [23, 42],object = {},key;

1. **Naming Conventions**

* **Use [lowerCamelCase](http://wiki.c2.com/?LowerCamelCase) for variables, properties and function names**

*Right:*

var adminUser = db.query(‘SELECT \* FROM users …’);

*Wrong:*

var admin\_user = db.query(‘SELECT \* FROM users …’);

* **Use [UpperCamelCase](http://wiki.c2.com/?UpperCamelCase) for class names**

Class names should be capitalised using UpperCamelCase.

function BankAccount() {}

* **Use UPPERCASE for Constants**

Constants should be declared as regular variables or static class properties, using all uppercase letters.

var SECOND = 1 \* 1000;  
function File() {  
}File.FULL\_PERMISSIONS = 0777;

1. **Variables**

* **Object / Array creation**

Use trailing commas and put *short* declarations on a single line. Only quote keys when your interpreter complains:

*Right:*

var a = ['hello', 'world'];var b = {  
 good: 'code',  
 'is generally': 'pretty',  
};

*Wrong:*

var a = [  
 'hello', 'world'  
];var b = {"good": 'code'  
 , is generally: 'pretty'  
 };

1. **Conditionals**

* **Use the === operator**

*Right:*

var a = 0;if (a !== '') {  
 console.log('winning');  
}

*Wrong:*

var a = 0;if (a == '') {  
 console.log('losing');  
}

* **Use descriptive conditions**

Any non-trivial conditions should be assigned to a descriptively named variable or function:

*Right:*

var isValidPassword = password.length >= 4 && /^(?=.\*\d).{4,}$/.test(password);if (isValidPassword) {  
 console.log('winning');  
}

*Wrong:*

if (password.length >= 4 && /^(?=.\*\d).{4,}$/.test(password)) {  
 console.log('losing');  
}

1. **Functions**

* **Write small functions**

Keep your functions short. A good function fits on a slide that the people in the last row of a big room can comfortably read.

* **Return early from functions**

To avoid deep nesting of if-statements, always return a function’s value as early as possible.

*Right:*

function isPercentage(val) {  
 if (val < 0) {  
 return false;  
 } if (val > 100) {  
 return false;  
 } return true;  
}

*Wrong:*

function isPercentage(val) {  
 if (val >= 0) {  
 if (val < 100) {  
 return true;  
 } else {  
 return false;  
 }  
 } else {  
 return false;  
 }  
}

}

* **Method chaining**

One method per line should be used if you want to chain methods.

You should also indent these methods so it’s easier to tell they are part of the same chain.

*Right:*

User.findOne({ name: ‘foo’ }).populate(‘bar’).exec(function(err, user) {return true;});

*Wrong:*

User  
.findOne({ name: 'foo' })  
.populate('bar')  
.exec(function(err, user) {  
 return true;  
});User.findOne({ name: 'foo' })  
 .populate('bar')  
 .exec(function(err, user) {  
 return true;  
 });User.findOne({ name: 'foo' }).populate('bar')  
.exec(function(err, user) {  
 return true;  
});User.findOne({ name: 'foo' }).populate('bar')  
 .exec(function(err, user) {  
 return true;  
 });

1. **Comments**

* **Use slashes for comments**

Use slashes for both single line and multi line comments. Try to write comments that explain higher level mechanisms or clarify difficult segments of your code

// 'ID\_SOMETHING=VALUE' -> ['ID\_SOMETHING=VALUE',   
// 'SOMETHING', 'VALUE']  
var matches = item.match(/ID\_([^\n]+)=([^\n]+)/));// This function has a nasty side effect where a failure to   
// increment a redis counter used for statistics will   
// cause an exception. This needs to be fixed in a later iteration.  
function loadUser(id, cb) {  
 // ...  
}var isSessionValid = (session.expires < Date.now());  
if (isSessionValid) {  
 // ...  
}