Thursday, 22 November 2018

Coding convention

How to write “clean” code the Lex style:

Use tabs not spaces (if you are using VS code make sure your tab size is set to 4 by typing ”editor.tabSize”: 4 in the JSON version of your settings. (The space in the following examples is one tab)

Make sure to write comments and documentation for every function that you define (make sure this is done before pushing to GitHub).

When using if else statements the code should look like this (easier to show you than explain):

```
// code
if (condition) {
	//some code
} else {
	//some code
}
// code
```

Use semicolons!!!

Make sure when using arrow functions to include all the parentheses and curly brackets, example (also indent the promise to the next line, as well as lines that are too long). Regarding promises, look at the pictures at the end of this file.

```
doFetchRequest('POST',"/favorites", {'Content-Type': 'application/json'}, JSON.stringify({name:'New Name', dataURL:`${picture}`}))
.then((data)=>{
	getFavorites();
	socket.emit('favorite.create', 'A favorites has been created’);
```

});

For string equality use '===‘.

Use let instead of var (unless for some reason you *have* to use var).

Use good (in the English language) names for variables.

For for loops I like to use the following style:

```
for (let i = 0 ; i < array.length ; i++) {
        //some code
}
```

When i concat strings I like to do like this, also used for arithmetic operations:

console.log(“hello” + “world”); (notice the space between the plus)

For HTML and CSS:

If it’s a child it should be like this:

```
<div>
        <p>
        </p>
</div>


.toolbar {
        width: 56px;
        text-align: center;
}
```

```
Favorites.find(filter, function(err, found) {
    // if there was an error
    if (err) {
        // code of response
        res.status(500).end();
        // stop executing code
        return;
    }
    console.log(found);
    // request is HTML
    if (req.accepts("html")) {
        // respond in HTML
        res.render("favorites", {result: found});
    } else {
        // link handling
        for (let i = 0 ; i < found.length ; i++) {
            found[i].links = {"self": config.url + "/favorites/" + found[i]._id};
        }
        // respond in json
        res.json(found);
    }
});
```

```
Favorites.find(filter)
.then(function(found) {
    if (req.accepts("html")) {
        // respond in HTML
        res.render("favorites", {result: found});
    } else {
        // link handling
        for (let i = 0 ; i < found.length ; i++) {
            found[i].links = {"self": config.url + "/favorites/" + found[i]._id};
        }
        // respond in json
        res.json(found);
    }
})
.catch(function (err){
    // code of response
    res.status(500).end();
});
```

```
Favorites.find(filter)
.then(function(found) {
    if (req.accepts("html")) {
        // respond in HTML
        res.render("favorites", {result: found});
    } else {
        // link handling
        for (let i = 0 ; i < found.length ; i++) {
            found[i].links = {"self": config.url + "/favorites/" + found[i]._id};
        }
        // respond in json
        res.json(found);
    }
},function(err){
    res.status(500).end();
});
```