Stefano Schmidt
Launch School
Introduction to Programming With JavaScript
Preparations – Exercises

1: This exercises asks us to create a directory named my_folder, and inside it create two JavaScript files one.js and two.js, and then add some code in each so that one.js logs "this is file one" and two.js logs "this is file two". Finally we are asked to run both files using node. I did this as follows:

```
~ <f,g>: mkdir my_folder
~ <f,g>: cd my_folder
~/my_folder <f,g>: touch one.js
~/my_folder <f,g>: touch two.js
~/my_folder <f,g>: nano one.js
~/my_folder <f,g>: nano two.js
~/my_folder <f,g>: node one.js
this is file one
~/my_folder <f,g>: node two.js
this is file two
```

note: I used the nano editor to add the two console logs, which I find is easier to use for small snippets than what I am using for larger edits (namely VS code.)

2: This exercise asks us to navigate to the directory above my_folder and delete all the content previously generated with one command. I did this as follows:

```
~/my_folder <f,g>: cd ../
~ <f,g>: rm -r my_folder
```

3: This exercises asks us to create a file named foo.js in a directory named preparations_exercises, and add this small snippet of JavaScript code to the file:

```
var foo = 'bar';
console.log(foo);
foo;
```

I did this as follows:

```
~/LaunchSchool/IntroJavaScriptExercises/preparations_exercises <f,g>: touch foo.js
~/LaunchSchool/IntroJavaScriptExercises/preparations_exercises <f,g>: nano foo.js
```

4: This asks us to run the code in foo.js using three different environments: node, the node REPL, and the Chrome browser. The outputs are the following:

node:

```
~/LaunchSchool/IntroJavaScriptExercises/preparations_exercises <f,g>: node foo.js
bar
```

node REPL:

```
> .load foo.js
var foo = 'bar';
console.log(foo);
foo;

bar
'bar'
```

Chrome browser:

```
> var foo = 'bar';
  console.log(foo);
  foo;
  bar
< "bar"
```

5: This exercise asks us to identify the Constructors of some methods and say in each case whether they are “Static” or “Prototype” methods:

substring: The constructor of this method is String, this method is a prototype method.
create: The constructor of this method is Object, this method is a static method.
fromCharCode: The constructor of this method is String, this method is a static method.
slice: There are multiple constructors of this method: String and Array. In both of these cases the methods are prototype methods.

6: This exercise asks us to identify which of the following names satisfy the style guidelines of non-constant variable names:

index: this name satisfies the style guidelines.
CatName: this name does not satisfy the style guidelines; it is not in camelCase.
snake_case: this name does not satisfy the style guidelines; it is not in camelCase.
lazyDog: this name satisfies the style guidelines.
quick_Fox: this name does not satisfy the style guidelines; it is not in camelCase.
1stCharacter: this name does not satisfy the style guidelines; variable names should not start with a number.
operand2: this name satisfies the style guidelines.
BIG_NUMBER: this name does not satisfy the style guidelines; it is not in camelCase.

7: This exercise asks us to identify which of the following names satisfy the style guidelines for function names:

index: this name satisfies the style guidelines.
CatName: this name satisfies the style guidelines (if the function in question is a constructor.)
snake_case: this name does not satisfy the style guidelines; function names should not have underscores anywhere.
lazyDog: this name satisfies the style guidelines.
quick_Fox: this name does not satisfy the style guidelines; function names should not have underscores anywhere.
1stCharachter: this name does not satisfy the style guidelines; function names should not begin with a numerical character.
operand2: this name satisfies the style guidelines.
BIG_NUMBER: this name does not satisfy the style guidelines; function names should not be all capitalized and should not have underscores anywhere.

8: This exercise asks us to identify which of the following names satisfy the style guidelines for naming constants:

index: this name does not satisfy the style guidelines, constant names should be uppercase.
CatName: this name does not satisfy the style guidelines; constant names should be uppercase
snake_case: this name does not satisfy the style guidelines; constant names should be uppercase.
lazyDog: this name does not satisfy the style guidelines; constant names should be uppercase.
quick_Fox: this name does not satisfy the style guidelines; constant names should be uppercase.
1stCharachter: this name does not satisfy the style guidelines; constant names should not begin with a numerical character and should be uppercase.
operand2: this name does not satisfy the style guidelines; constant names should be uppercase.
BIG_NUMBER: this name satisfies the style guidelines.

9: This exercise asks us to identify which of the following names don't satisfy the style guidelines for naming variables, functions, or constants:

index: this name satisfies the style guidelines.
CatName: this name satisfies the style guidelines (if the name is for a constructor function.)
snake_case: this name does not satisfy the style guidelines; lowercase snake_case should not be used in any (there are technically some execeptions) situation.
lazyDog: this name satisfies the style guidelines.
quick_Fox: this name does not satisfy the style guidelines; underscores can only be used in constant names, but all other characters should be capitalized
1stCharachter: this name does not satisfy the style guidelines; names should not begin with a numerical character.
operand2: this name satisfies the style guidelines.
BIG_NUMBER: this name satisfies the style guidelines (if the name is for a constant variable.)