Module 3 Labs
JS Fundamentals

1. What are the results of these expressions? *(answer first, then use console.log() to check)*

```
"" + 1 + 0
"" - 1 + 0
true + false
!true
6 / "3"
"2" * "3"
4 + 5 + "px"
"$" + 4 + 5
"4" - 2
"4px" - 2
"  -9  " + 5
"  -9  " - 5
null + 1
undefined + 1
undefined == null
undefined === null
" \t \n" - 2
```

2. Which of the below are not giving the right answer? Why are they not correct? How can we fix them?

```
let three = "3"
let four = "4"
let thirty = "30"

//what is the value of the following expressions?
let addition = three + four
let multiplication = three * four
let division = three / four
let subtraction = three - four

let lessThan1 = three < four
let lessThan2 = thirty < four
```

3. Which of the following console.log messages will print? Why?

```
if (0) console.log('#1 zero is true')
if ("0") console.log('#2 zero is true')
if (null) console.log('null is true')
if (-1) console.log('negative is true')
if (1) console.log('positive is true')
```

4. Rewrite this `if` using the ternary/conditional operator '`?`'. Test it with different values for `a` and `b`. What does the '`+=`' do?

```
let a = 2, b = 3;
let result = `${a} + ${b} is `;

if (a + b < 10) {
  result += 'less than 10';
} else {
  result += 'greater than 10';
}
```

5. Rewrite the following function using: a) function expression syntax, and b) arrow function syntax. Test each version to make sure they work the same.

```
function getGreeting(name) {
    return 'Hello ' + name + '!';
}
```

6. a) Complete the `inigo` object by adding a `lastName` property and including it in the `greeting`.
   b) Complete `getCatchPhrase` so that if the `person` argument has 6 fingers, it instead prints his famous catch phrase to the console. HINT: see https://www.imdb.com/title/tt0093779/characters/nm0001597.
   c) Update `getCatchPhrase` to use arrow function syntax and a conditional operator.

```
const westley = {
    name: 'Westley',
    numFingers: 5
}
const rugen = {
    name: 'Count Rugen',
    numFingers: 6
}

const inigo = {
    firstName: 'Inigo',
    greeting(person) {
        let greeting = `Hello ${person.name}, my name is ${this.firstName}. `;
        console.log(greeting + this.getCatchPhrase(person));
    },
    getCatchPhrase(person) {
        return 'Nice to meet you.';
    }
}

inigo.greeting(westley)
inigo.greeting(rugen)
```

7. The following object represents a basketball game and keeps track of the score as the game progresses.
   a) Modify each of the methods so that they can be 'chained' together and the last line of the example code works
   b) Add a new method to print the full time final score
   c) Add a new object property to keep track of the number of fouls and a method to increment it, similar but separate to the score. Include the foul count in the half time and full time console messages
   d) Test your object by chaining all the method calls together in different combinations.

```
const basketballGame = {
    score: 0,
    freeThrow() {
        this.score++;
    },
    basket() {
        this.score += 2;
    },
    threePointer() {
        this.score += 3;
    },
    halfTime() {
        console.log('Halftime score is '+this.score);
    }
}

//modify each of the above object methods to enable function chaining as below:
basketballGame.basket().freeThrow().freeThrow().basket().threePointer().halfTime();
```

8. The object below represents a single city.
   a) Write a function that takes an object as an argument and uses a `for…in` loop to access and print to the console each of those object properties and their values. Test it using the `sydney` object below.
   b) Create a new object for a different city with different properties and call your function again with the new object.

```
const sydney = {
    name: 'Sydney',
    population: 5_121_000,
    state: 'NSW',
    founded: '26 January 1788',
    timezone: 'Australia/Sydney'
}
```

9. Use the following variables to understand how JavaScript stores objects by reference.
   a) Create a new `moreSports` variable equal to `teamSports` and add some new sport values to it (using `push` and `unshift`)
   b) Create a new `dog2` variable equal to `dog1` and give it a new value
   c) Create a new `cat2` variable equal to `cat1` and change its `name` property
   d) Print the original `teamSports`, `dog1` and `cat1` variables to the console. Have they changed? Why?
   e) Change the way the `moreSports` and `cat2` variables are created to ensure the originals remain independent

```
let teamSports = ['Hockey', 'Cricket', 'Volleyball'];
let dog1 = 'Bingo';
let cat1 = { name: 'Fluffy', breed: 'Siberian' };
```

10. The following constructor function creates a new `Person` object with the given `name` and `age` values.
    a) Create a new person using the constructor function and store it in a variable
    b) Create a second person using different `name` and `age` values and store it in a separate variable
    c) Print out the properties of each person object to the console
    d) Rewrite the constructor function as a class called `PersonClass` and use it to create a third person using different name and age values. Print it to the console as well.
    e) Add a `canDrive` method to both the constructor function and the class that returns true if the person is old enough to drive.

```
function Person(name, age) {
    this.name = name;
    this.age = age;
    this.human = true;
}
```

Extension: If you have time, try the JS Challenge Rush at https://www.jschallenger.com/games/rush