BCVD1006 – Full Stack Development – Lab 1

- JavaScript Functions
- JavaScript Loops
- JavaScript Arrays
- JavaScript Objects

Developer Note:

- Please create a separate JavaScript file for each exercise
- You may use the HTML page to trigger your scripts or you may use the JS Playground Editors (REPL) to program and just submit the JS file.
    - https://repl.it/languages/javascript
    - https://jsfiddle.net/

Resources
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/toFixed

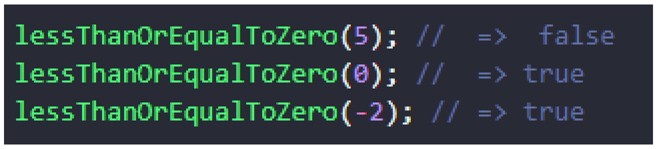
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/parseFloat
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/PI

Exercise 1:

Write a script that determines if the **given number is less than or equal to zero**.

- Create a **function expression** that takes a number as its only argument and **returns true** if it's less than or equal to zero, otherwise **return false**.
- Invoke the function and call it to return a result
- Hint: you may need to use conditional statements

Expected output is as follows:



```
lessThanOrEqualToZero(5); //  =>  false
lessThanOrEqualToZero(0); //  => true
lessThanOrEqualToZero(-2); // => true
```

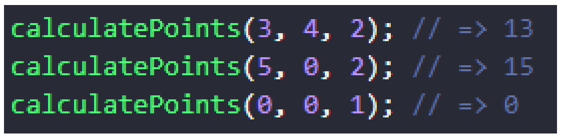
Exercise 2:

Write a script that determines the Football Points of a given team record.

- Create a **function expression** that takes **3 parameters**, the number of wins, draws and losses and calculates the number of points a football team has obtained so far.
    - **wins** get 3 points
    - **draws** get 1 point
    - **losses** get 0 points

- Invoke the function and call it to return a result

Expected output is as follows:



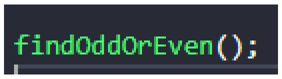
```
calculatePoints(3, 4, 2); // => 13
calculatePoints(5, 0, 2); // => 15
calculatePoints(0, 0, 1); // => 0
```

Exercise 3:

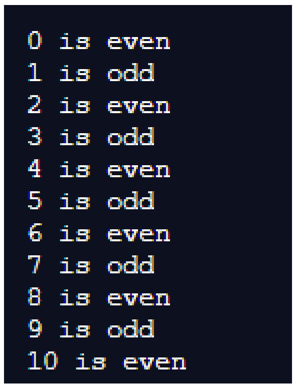
Create a function named **findOddOrEven** that has the following requirements:

- Takes zero parameters
- Write a for loop that will iterate from 0 to 10.
- For each iteration, it will check if the current number is even or odd, and then output
    - number is even or number is odd

Expected output is as follows:



```
findOddOrEven();
```



```
0 is even
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
10 is even
```

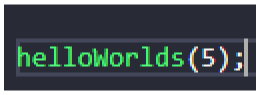
Exercise 4:

Create a function named **helloWorlds** using looping and conditionals that has the following requirements:

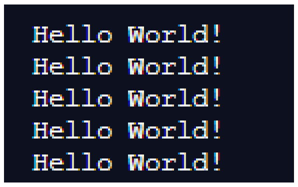
- Takes a **number** as parameter
- Will check whether the param is a number or not a number
    - If the param is a number, output “**Hello World**” x num
    - Otherwise, if not a number output “**Goodbye World**”

- Hint: you may need to a for loop for this exercise

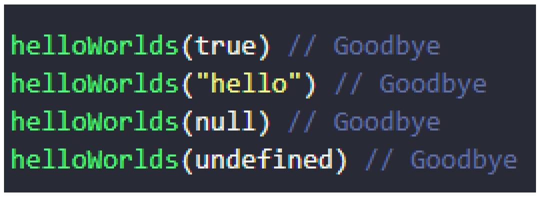
Expected output is as follows:



```
helloWorlds(5);
```



```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```



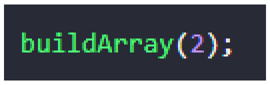
```
helloWorlds(true) // Goodbye
helloWorlds("hello") // Goodbye
helloWorlds(null) // Goodbye
helloWorlds(undefined) // Goodbye
```

<u>Exercise 5:</u>

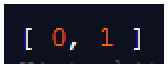
Create a function named **buildArray** that has the following requirements:

- Takes a parameter ***num***
- Create a local variable named ***myArray***
- Write a loop that will continue to iterate until the counter is greater than num.
- For each iteration, it add to your counter value to your variable ***myArray***
- Before the function exists it will return the value of ***myArray***

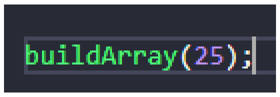
Expected output is as follows:



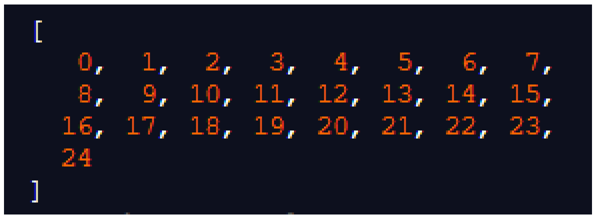
```
buildArray(2);
```



```
[ 0, 1 ]
```



```
buildArray(25);
```



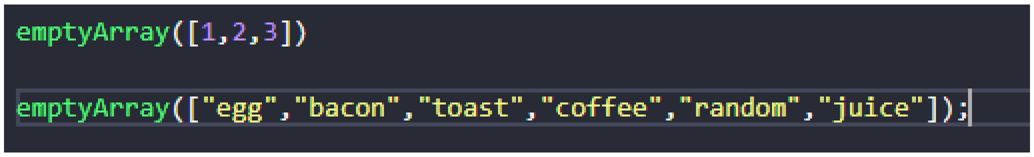
```
[
   0,  1,  2,  3,  4,  5,  6,  7,
   8,  9, 10, 11, 12, 13, 14, 15,
  16, 17, 18, 19, 20, 21, 22, 23,
  24
]
```

Exercise 6:

Create a function named **emptyArray** that has the following requirements:

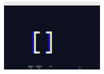
- Takes a parameter ***myArray*** which will always be an array
- Write a loop that will continue to iterate until the array is empty
- For each iteration, remove an element from the array
- Before the function exists it will return the value of ***myArray***

Expected output is as follows:



```
emptyArray([1,2,3])

emptyArray(["egg","bacon","toast","coffee","random","juice"]);
```



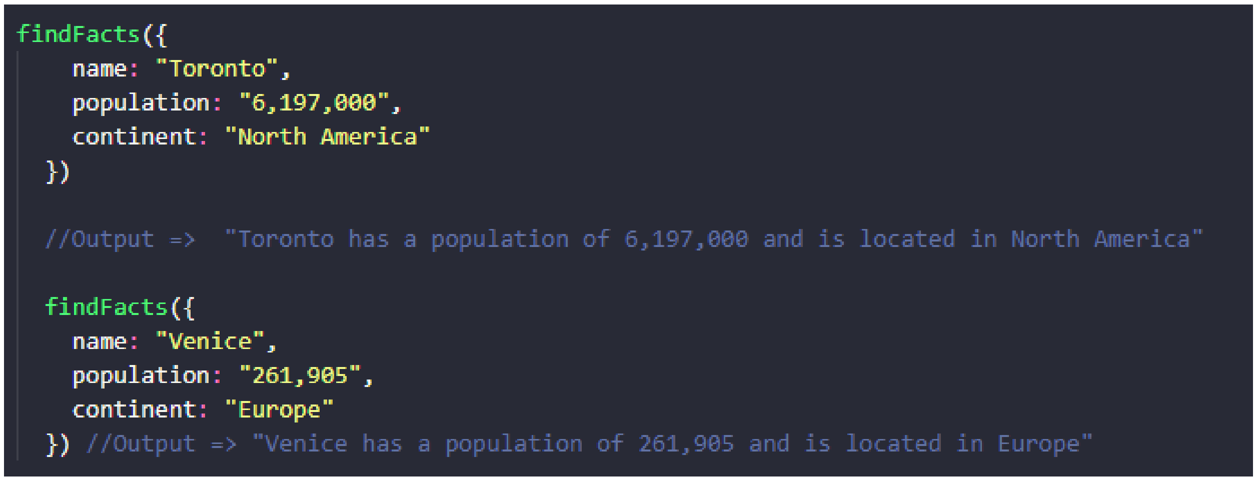
```
[]
```

Exercise 7:

Create a script file with the following requirements

- Create a function named findFacts that will take a city object as an parameter
- The function will output the values in string format to the console

The output should be as follows:



```
findFacts({
    name: "Toronto",
    population: "6,197,000",
    continent: "North America"
  })

  //Output =>  "Toronto has a population of 6,197,000 and is located in North America"

  findFacts({
    name: "Venice",
    population: "261,905",
    continent: "Europe"
  }) //Output => "Venice has a population of 261,905 and is located in Europe"
```

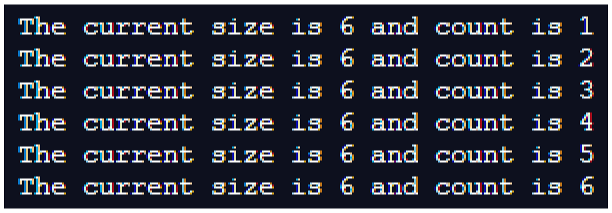
<u>Challenges:</u>

1. Create a function named **greaterThanSize** that has the following requirements:

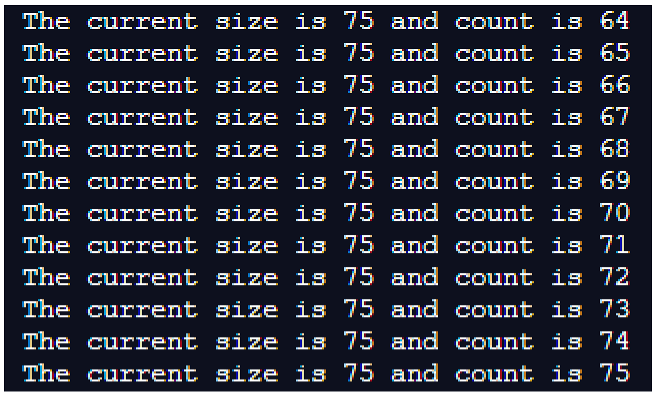
- Takes no parameters
- Calculates a random local variable named size
- Does a while loop that does the following
    - increments a counter by one
    - output the current size and counter
    - continues until the counter is greater than the size
- The output will be different every time, because the size number will be different

- <u>Hint: you may need use Math functions for the random size</u>

    Expected output is as follows:



```
The current size is 6 and count is 1
The current size is 6 and count is 2
The current size is 6 and count is 3
The current size is 6 and count is 4
The current size is 6 and count is 5
The current size is 6 and count is 6
```



```
The current size is 75 and count is 64
The current size is 75 and count is 65
The current size is 75 and count is 66
The current size is 75 and count is 67
The current size is 75 and count is 68
The current size is 75 and count is 69
The current size is 75 and count is 70
The current size is 75 and count is 71
The current size is 75 and count is 72
The current size is 75 and count is 73
The current size is 75 and count is 74
The current size is 75 and count is 75
```

2. Create a function named **reverseIt** that has the following requirements:

- Takes a parameter ***myArr*** which will always be an array
- Will **return** an array in reverse order

    <u>Expected output is as follows:</u>

```
reverseIt([1, 2, 3, 4]) // [4, 3, 2, 1]

reverseIt([9, 9, 2, 3, 4]) // [4, 3, 2, 9, 9]

reverseIt([]) // []
```

3. Create a script file **challenge3.js** with the following requirements

- Create a function named getKeyValuePairs that will take a object as an parameter
- Returns the keys and values as separate arrays
- The keys and values may be in the same original order

The output should be as follows:

```
getKeyValuePairs({ a: 5, b: 10, c: 16 })
//Output => [ [ 'a', 'b', 'c' ], [ 5, 10, 16 ] ]

getKeyValuePairs({ r: "React", a: "Angular", v: "Vue" })
// Output => [ [ 'r', 'a', 'v' ], [ 'React', 'Angular', 'Vue' ] ]

getKeyValuePairs({ k1: true, k2: false, k3: undefined })
// Output => [ [ 'k1', 'k2', 'k3' ], [ true, false, undefined ] ]
```