Client-Side Scripting Exercises

For each exercise, create the function and implement it inside of `exercises.js`. The unit tests defined in `tests.js` look for these functions and call them using the signature shown below.

Exercises

1. **sumDouble** Given two int values, return their sum. Unless the two values are the same, then return double their sum.

```
sumDouble(1, 2) → 3
sumDouble(3, 2) → 5
sumDouble(2, 2) → 8

function sumDouble(x, y) {
   // do logic here
   // return result;
   return x + y;
}
```

2. **hasTeen** A number is "teen" if it is in the range 13..19 inclusive. Given three int values, return true if 1 or more of them are teen.

```
hasTeen(13, 20, 10) → true
hasTeen(20, 19, 10) → true
hasTeen(20, 10, 13) → true
```

3. **lastDigit** Given two non-negative int values, return true if they have the same last digit, such as 27 and 57.

```
lastDigit(7, 17) → true
lastDigit(6, 17) → false
lastDigit(3, 113) → true
```

4. **seeColor** Given a string, if the string begins with "red" or "blue" return that color string; otherwise, return the empty string.

```
seeColor("redxx") → "red"
seeColor("xxred") → ""
seeColor("blueTimes") → "blue"
```

5. **oddOnly** Write a function that, given an array of integer of any length, filters out the even number, and returns a new array of just the odd numbers.

```
oddOnly([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]) → [1, 3, 5, 7, 9, 11];
oddOnly([2, 4, 8, 32, 256]); → []
```

6. **frontAgain** Given a string, return true if the first two chars in the string also appear at the end of the string, as with "edited."

```
frontAgain("edited") → true
frontAgain("edit") → false
frontAgain("ed") → true
```

7. **cigarParty** When squirrels get together for a party, they like to have cigars. A squirrel party is successful when the number of cigars is between 40 and 60, inclusive. If it is the weekend, there is no upper bound on the number of cigars. Write a squirrel party function that returns true if the party with the given values is successful, or false otherwise.

```
cigarParty(30, false) → false
cigarParty(50, false) → true
cigarParty(70, true) → true
```

8. **fizzBuzz** Because you know you can't live without it, FizzBuzz.

```
fizzBuzz(3) → "Fizz"
fizzBuzz(1) → 1
fizzBuzz(10) → "Buzz"
fizzBuzz(15) → "FizzBuzz"
fizzBuzz(8) → 8
```

9. **filterEvens** Write a function that filters an array to only include even numbers.

```
filterEvens([]) → []
filterEvens([1, 3, 5]) → []
filterEvens([2, 4, 6]) → [2, 4, 6]
filterEvens([100, 8, 21, 24, 62, 9, 7]) → [100, 8, 24, 62]
```

10. **filterBigNumbers** Write a function that filters numbers greater than or equal to 100.

```
filterBigNumbers([7, 10, 121, 100, 24, 162, 200]) → [121, 100, 162, 200]
filterBigNumbers([3, 2, 7, 1, -100, -120]) → []
filterBigNumbers([]) → []
```

11. **filterMultiplesOfX** Write a function to filter numbers that are a multiple of a parameter, `x` passed in.

```
filterMultiplesOfX([3, 5, 1, 9, 18, 21, 42, 67], 3) → [3, 9, 18, 21, 42]
filterMultiplesOfX([3, 5, 10, 20, 18, 21, 42, 67], 5) → [5, 10, 20]
```

12. **createObject** Write a function that creates an object with a property called firstName, lastName, and age. Populate the properties with your values.

```
createObject() →

{
    firstName,
    lastName,
    age
}
```

Challenge exercises

1. **iqTest** Bob is preparing to pass an IQ test. The most frequent task in this test is to find out which one of the given numbers differs from the others. Bob observed that one number usually differs from the others in evenness. Help Bob: to check his answers, he needs a program that, among the given numbers, finds one that is different in evenness and returns the position of this number. *Keep in mind that your task is to help Bob solve a real IQ test, which means indexes of the elements start from 1, not 0*.

```
 iqTest("2 4 7 8 10") → 3 //third number is odd, while the rest are even
 iqTest("1 2 1 1") → 2 // second number is even, while the rest are odd
 iqTest("") → 0 // there are no numbers in the given set
 iqTest("2 2 4 6") → 0 // all numbers are even, therefore there is no
position of an odd number
```

2. **titleCase** Write a function that converts a string into title case, given an optional list of exceptions (minor words). The list of minor words is given as a string with each word separated by a space. Your function should ignore the case of the minor words string. It should behave in the same way even if the case of the minor word string is changed.

- First argument (required): the original string to be converted.
- Second argument (optional): space-delimited list of minor words that must always be lowercase except for the first word in the string. The JavaScript tests pass undefined when this argument is unused.

```
  titleCase('a clash of KINGS', 'a an the of') // should return: 'A Clash of
Kings'
  titleCase('THE WIND IN THE WILLOWS', 'The In') // should return: 'The Wind
in the Willows'
  titleCase('the quick brown fox') // should return: 'The Quick Brown Fox'
```