Module 3 Labs
JS Intermediate

1. Create a function that takes a string as a parameter and returns the string with the first character of each word changed into a capital letter, as in the example below. Test it with different strings.

```
console.log(ucFirstLetters("los angeles") ) //Los Angeles
```

2. Create a function `truncate(str, max)` that truncates a given string of text if its total length is greater than the `max` length. It should return either the truncated text, with an ellipsis (…) added to the end if it was too long, or the original text otherwise.
   b) Write another variant of the `truncate` function that uses a conditional operator.

```
console.log(truncate('This text will be truncated if it is too long', 25))
// This text will be truncat...
```

3. Use the following `animals` array for the below tasks. Test each one by printing the result to the console. Review the following link for tips:
   https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

   a) Add 2 new values to the end
   b) Add 2 new values to the beginning
   c) Sort the values alphabetically
   d) Write a function `replaceMiddleAnimal(newValue)` that replaces the value in the middle of the `animals` array with `newValue`
   e) Write a function `findMatchingAnimals(beginsWith)` that returns a new array containing all the animals that begin with the `beginsWith` string. Try to make it work regardless of upper/lower case.

```
const animals = ['Tiger', 'Giraffe']
console.log(animals)
```

4. Write a function `camelCase(cssProp)` that changes dash-separated CSS properties like `'margin-left'` into camel-cased `'marginLeft'`.

   The function should remove all dashes, and uppercase the first letter of each word after a dash.

   b) Create variants of the `camelCase` function that use different types of `for` loops, and c) with and without the conditional operator.

```
console.log(camelCase('margin-left')) // marginLeft
console.log(camelCase('background-image')) // backgroundImage
console.log(camelCase('display')) // display
```

5. Decimal number operations in JavaScript can lead to unexpected results, as in the following:

```
let twentyCents = 0.20
let tenCents = 0.10

console.log(`${twentyCents} + ${tenCents} = ${twentyCents + tenCents}`)
// 0.2 + 0.1 = 0.30000000000000004
```

   We can sometimes avoid this using the toFixed function to force the number of decimal places as below, but it's not always useful:

```
let fixedTwenty = twentyCents.toFixed(2);
let fixedTen = tenCents.toFixed(2);

console.log(fixedTwenty + fixedTen) //why is this not working?
```

   a) Explain why the above code returns the wrong answer
   b) Create a function `currencyAddition(float1, float2)` which safely adds the two decimal numbers `float1` and `float2` and returns the correct float result.
   c) Create a function `currencyOperation(float1, float2, operation)` which safely performs the given operation (either +, -, / or *) on the two numbers and returns the correct float result. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/switch may be useful.
   d) (Extension) Extend the above function to include a fourth argument `numDecimals` which allows the operation to support different amounts of decimal places from 1 to 10.

   HINT: Assume 2 decimal places for b) and c) and use a multiplication factor. Test with different values as well as the below:

```
console.log(0.3 == currencyAddition(0.1, 0.2)) // true
console.log(0.3 == currencyOperation(0.1, 0.2, '+')) // true
```

6. Create a function `unique(duplicatesArray)` which takes an array parameter that may include duplicates. Your function should return an array containing only the unique values from `duplicatesArray`.
   Test with the following arrays and create another one of your own.

```
const colours = ['red', 'green', 'blue', 'yellow', 'orange', 'red', 'blue', 'yellow']
const testScores = [55, 84, 97, 63, 55, 32, 84, 91, 55, 43]

console.log(unique(colours)) // [ 'red', 'green', 'blue', 'yellow', 'orange' ]
console.log(unique(testScores)) // [ 55, 84, 97, 63, 32, 91, 43 ]
```

7. Use the following array of book objects to practice the array functions for `map`, `find` and `filter`. Test each of your answers to the below tasks.

```
const books = [
    { id: 1, title: 'The Great Gatsby', author: 'F. Scott Fitzgerald', year: 1925 },
    { id: 2, title: 'To Kill a Mockingbird', author: 'Harper Lee', year: 1960 },
    { id: 3, title: '1984', author: 'George Orwell', year: 1949 },
    { id: 4, title: 'Brave New World', author: 'Aldous Huxley', year: 1932 },
    { id: 5, title: 'The Catcher in the Rye', author: 'J.D. Salinger', year: 1951 },
];
```

   a) Write a function `getBookTitle(bookId)` that uses the `find` function to return the title of the book object with the matching `id`.
   b) Write a function `getOldBooks()` that uses the `filter` function to return all book objects written before 1950.
   c) Write a function `addGenre()` that uses the `map` function to add a new `genre` property to all of the above books, with the value 'classic'.
   d) (Extension) Write a function `getTitles(authorInitial)` that uses `map` and `filter` together to return an array of book titles for books written by authors whose names start with `authorInitial`.
   e) (Extension) Write a function `latestBook()` that uses `find` and `forEach` to get the book with the most recent publication date.

8. The following code creates a new Map object for storing names beginning with A, B, or C with their phone numbers.

```
const phoneBookABC = new Map() //an empty map to begin with
phoneBookABC.set('Annabelle', '0412312343')
phoneBookABC.set('Barry', '0433221117')
phoneBookABC.set('Caroline', '0455221182')
```

   a) Create a new `phoneBookDEF` Map to store names beginning with D, E or F
   b) Initialise the contents of `phoneBookDEF` by passing in an array of keys/values
   c) Update the phone number for Caroline
   d) Write a function `printPhoneBook(contacts)` that prints the names and phone numbers in the given Map
   e) Combine the contents of the two individual Maps into a single `phoneBook` Map
   f) Print out the full list of names in the combined phone book

9. Given the below salaries object, perform the following tasks.

```
let salaries = {
    "Timothy" : 35000,
    "David" : 25000,
    "Mary" : 55000,
    "Christina" : 75000,
    "James" : 43000
};
```

   a) Write a function sumSalaries(salaries) that calculates and returns the total of all salaries
   b) Write a function topEarner(salaries) that calculates and returns the name of the person earning the highest salary

10. The following code uses the Date object to print the current time and the number of hours that have passed today so far. Extend the code to do the following:

```
const today = new Date();
console.log('Current time is ' + today.toLocaleTimeString())

console.log(today.getHours() + ' hours have passed so far today')
```

   a) Print the total number of minutes that have passed so far today
   b) Print the total number of seconds that have passed so far today
   c) Calculate and print your age as: 'I am x years, y months and z days old'
   d) Write a function daysInBetween(date1, date2) which calculates and returns the amount of days in between the two given dates.