EXERCISE 1

- Create a function that list out all the number divisible by 3 from given array of numbers

const array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];

EXERCISE 2

- Write a simple destructuring expression to find the value of title, author firstname and publisher web url from the given object
- Change the variable name title to **doc,** firstname to **name,** and url to **web**

```
const meta = {
    title: 'Destructuring Assignment',
    authors: [ { firstname: 'Craig', lastname: 'Buckler' } ],
    publisher: {
        name: 'SitePoint',
        url: 'http://www.sitepoint.com/'
    }
};
```

EXERCISE 3

- Write a function using destructuring syntax to extract the value of title, author for each book element in a books array

```
const books = [
    { title: 'Full Stack JavaScript', author: 'Colin Ihrig and Adam Bretz', url: 'http://www.sitepoint.com/store/full-stack-javascript/' },
    { title: 'JavaScript: Novice to Ninja', author: 'Darren Jones', url: 'http://www.sitepoint.com/store/leaern-javascript/' },
    { title: 'Jump Start CSS', author: 'Louis Lazaris', url: 'http://www.sitepoint.com/store/jump-start-css/' },
];
```

EXERCISE 4

- Write a function that calculates the prices after tax calculation for each price element in the prices array

var prices = [1.1, 2.32, 3.68, 4.9];

result should be **pricesWithTax**

Assume taxRate to be 0.07

Use Currying and ES6 construct to solve this

EXERCISE 5

- Create an mapper function using arrow symantics that returns an object having following attribute when operating on array

```
console.log([1,2,3].map(mapperFns)
```

Result

```
[{
  isEven: false,
  number: 1,
  squared: 1
},
{
  isEven: true,
  number: 2,
  squared: 4
},
{
  isEven: false,
  number: 3,
  squared: 9
}
]
```

EXERCISE 6

- Write following code using object literal expression

```
const first = 'Mohan';
const last = 'Singh';

const person = new Object();
person.first = first;
person.last = last;
person.favouriteColor = 'blue';
person.address = new Object();
person.address.city = 'Gurgaon';
person.address.street = 'Sector 21';
person.address.state = 'HR';
```

EXERCISE 7

- Write a function **request** which when called with parameter as shown below prints **url** and **method** value

```
request({url: 'http://www.google.com', method: 'GET' });
```

Prints value like
http://www.google.com
GET

EXERCISE 8

- Create a function to Extract the value marked in red from the given object

```
const person = {
  first: 'Mohan',
  last: 'Singh',
  address: {
   lines: ['Sector 47', 'Chestor Lane', 'Apt 23'],
   city: 'Gurgaon',
   state: 'Haryana'
  }
}
```

EXERCISE 69

- Open the developer tools on your favorite news site. Locate the first character of all headings, and log the concatenation of the first characters.

```
let text = '';
let nodes = // Write code here
for ( let node of nodes ) {
   // Write code here
};
console.log( text );
```

PRACTICAL EXERCISES - EXERCISE 10

1. Write a function that returns all elements of array except first element

Use Destructuring to solve

EXERCISE 11

1. Write a function that returns an array from a given string

wrap(“package”) // result should be [“package”]

2. Write a function that takes an array of string and returns plain string

unwrap([“package”]) // result should be “package”

Use Destructuring to solve

PRACTICAL EXERCISES - EXERCISE 12

1. Parse the day, month and year from the date **('2025-09-26')**

Use Destructuring to solve

EXERCISE 13

- Create one destructuring expression that declares exactly one variable to retrieve x.A[2]

```
let x = { A: ['t', 'e', 's', 't']
```

EXERCISE 14

- Suppose the following configuration object of a financial chart is given:

```
let config = {
        chartType : 0,
        bullColor : 'green',
        bearColor : 'red',
        days : 30
};
```

- Complete the function signature below such that the function may be called with any config objects (null and undefined are not allowed as inputs). If any of the four keys are missing, substitute their default values. The default values are the same as in the example configuration object.

```
function drawChart( data, /* insert your solution here */ ) {
// do not implement the function body
};
```

SOLUTION 14

```
let config = {
   chartType: 0,
   bullColor: "green",
   bearColor: "red",
   day: 30
};
 function drawChart(data, options = {}) {
   _drawChart(data, options);
 }
function _drawChart(  data,  {
   chartType = 0,
   bullColor = "green",
   bearColor = "red",
   day = 30
}) {
  console.log(chartType, bullColor, bearColor, day)
;}
```

SOLUTION 14 - MODIFIED

```
let config = {
   chartType: 0,
   bullColor: "green",
   bearColor: "red",
   day: 30
};
function drawChart(  data,  {
   chartType = 0,
   bullColor = "green",
   bearColor = "red",
   day = 30
}= {}) {
  console.log(chartType, bullColor, bearColor, day)
;}
console.log(drawChart([], { chartType: 4 }));
```

EXERCISE 15

- Create an array that contains the last four characters of another array

EXERCISE 16

- Write function that accepts two String arguments, and returns the length of the longest common substring in the two strings.

```
maxCommon( ’523', ’5' ); // 1

maxCommon( ’55444', ’55f444g'); // 3

maxCommon( 'abc', ’334cab' ); // 2
```

SOLUTION 16 : DISCUSSION

- We will use an optional len argument to store the number of character matches before the current iteration of maxCommon was called.
- We will use recursion to process the strings.
- If any of the strings have a length of 0, either head1, or head2 becomes undefined.
  - This is our exit condition for the recursion, and we return len, i.e. the number of matching characters right before one of the strings became empty.
- If both strings are non-empty, and the heads match, we recursively call maxCommon on the tails of the strings, and increase the length of the counter of the preceding common substring sequence by 1.
- If the heads don't match, we remove one character from either the first string or from the second string, and calculate their maxCommon score, with len initialized to 0 again.
- The longest string may either be in one of these branches, or it is equal to len, counting the matches preceding the current strings [head1,...tail1] and [head2,...tail2].

EXERCISE 17

- Create a 10 x 10 matrix of **null** values

EXERCISE 18

1. Write a function that executes a callback function after a given delay in milliseconds. The default value of delay is one second.

EXERCISES 19

- Define a function that concatenates several strings
  - Function should take a separator and list of arguments
  - Function should return list of arguments concatenated and separated by separator

- Define a function that creates HTML lists
  - Creates a string containing HTML for a list
  - Only formal argument for a function is a string that is "**u**" for unordered list and **"o"** for ordered list

- Create a factorial calculator

EXERCISES

1. Create an implementation where we can extract substring from a given string
   a) **str.substr(start, length)**

2. Modify the Program to find the first **n**th characters

3. Modify the Program to find the first character from a given string

Use the Currying Concept to Solve these exercises