JavaScript Basics Part- 1

1 GETTING STARTED WITH JAVASCRIPT

1.1 WHAT IS VARIABLE, DECLARE A VARIABLE, SEE OUTPUT

JAVASCRIPT

```
// declare Variable 5 things (var, varName, =, value, ;)
var bananaPrice = 12
console.log(bananaPrice)
```

1.2 VARIABLE TYPE, NUMERIC, STRING, BOOLEAN

JAVASCRIPT

```
// variable type & Keywords
var seenAfter = 21
console.log(seenAfter)

// number
console.log(typeof seenAfter) // number

// string
var name = 'Faisal Akbar'
console.log(name)
console.log(typeof name) // string

// boolean
var isHot = true
var isRich = false

console.log(isHot)
console.log(typeof isHot) // boolean
```


1.3 VARIABLE NAME NAMING CONVENTION AND BEST PRACTICE

JAVASCRIPT

```
// var keyword, most of the cases keywords are lower case
// google JS keywords
// variable name case sensitive
// no need to declare same variable more than once
// naming convention- Camel Case
// use Meaningful name e.g. userName
var myName = 'Tom Cruise'
myName = 'Tom Hanks'
MyName = 'Robert Smith'

console.log(myName) // Tom Hanks
```

1.4 EXPLORE STRING CASE CHANGE INDEX SPLIT

JAVASCRIPT

```
// string declaration use- double quote(" "); single quote (' '); punctuation mar
var promise = 'I promise I will word HARD to become a programmer'

// make lowercase
console.log(promise.toLowerCase())
// i promise i will word hard to become a programmer

// make uppercase
console.log(promise.toUpperCase())
// I PROMISE I WILL WORD HARD TO BECOME A PROGRAMMER

// find index, start at 0
console.log(promise.indexOf('will')) // 12
console.log(promise.indexOf('promise')) // 2

// -1 index meaning not found
console.log(promise.indexOf('pomise')) // -1
console.log(promise.indexOf('hard')) // -1

// split string
console.log(promise.split('I'))
// [ '', ' promise ', ' will word HARD to become a programmer' ]

console.log(promise.split(' '))
/* [
  'I',  'promise',
  'I',  'will',
  'word', 'HARD',
  'to',  'become',
  'a',  'programmer'
] */
```


1.5 INTEGER FLOAT PARSE INT PARSE FLOAT TYPE CONVERSION

JAVASCRIPT

```javascript
// integer and float
var number1 = 25
var number2 = 15.5
console.log(number1 + number2) // 40.5

var number1 = '25'
var number2 = 15.5
console.log(number1 + number2) // 2515.5

var number1 = 25
var number2 = '10.5'
console.log(number1 + number2) // 2510.5

// parsing string to float, use if you are not sure
number2 = parseFloat(number2)
console.log(number1 + number2) // 35.5

// parsing string to integer, use when whole number is necessary only
number2 = parseInt(number2)
console.log(number1 + number2) // 35

// shortcut to convert string number to float
var number3 = '15.5'
number3 = +number3
console.log(number1 + number3) // 40.5

// convert number to string, add an empty string('')
number4 = 12
number4 = '' + number4
console.log(typeof number4) // string

// issue with decimal
var number5 = 0.1
var number6 = 0.2
console.log(number5 + number6) // 0.30000000000000004

// fixed decimal precision, the toFixed() method formats a number using fixed-poi
var total = number5 + number6
```

```
total = total.toFixed(1)
console.log(total) // 0.3
```

1.6 MATHEMATICAL OPERATIONS IN JAVASCRIPT

JAVASCRIPT

```
// addition
var a = 35
var b = 20
var sum = a + b
console.log(sum) // 55

// subtraction
var sub = a - b
console.log(sub) // 15

// multiplication
var mul = a * b
console.log(mul) // 700

// division
var div = a / b
console.log(div) // 1.75

// remainder or modulus
var mod = a % b
console.log(mod) // 15

// plus plus; minus minus Operator
var c = 40
var d = 45

c++ /*same as c=c+1*/
console.log(c) // 41

d-- /* same as d=d-1*/
console.log(d) // 44
```

```
// string and number
var userName = 'jack'
var age = 30

// string + number
var result = userName + age
console.log(result) // jack30

// number + string
var result = age + userName
console.log(result) // 30jack

// string number
var price = '35'
var age = 30

// string + number
var result = price + age
console.log(result) // 3530

// number + String
var result = age + price
console.log(result) // 3035

// add String
var firstName = 'jack'
var lastName = 'Smith'
var fullName = firstName + ' ' + lastName
console.log(fullName) // jack Smith
```


1.7 MATH ABSOLUTE ROUND FLOOR CEIL RANDOM

JAVASCRIPT

```
// absolute number
var x = -5
var absoluteNumber = Math.abs(x)
console.log(absoluteNumber) // 5
```

```
// the Math.round() function returns the value of a number rounded to the nearest
var y = 5.4545
var z = 5.8924
var roundNumber = Math.round(y)
var roundNumber2 = Math.round(z)
console.log(roundNumber) // 5
console.log(roundNumber2) // 5

// the Math.ceil() function always rounds a number up to the next largest integer
var ceilingNumber = Math.ceil(y)
var ceilingNumber2 = Math.ceil(z)
console.log(ceilingNumber) // 6
console.log(ceilingNumber2) // 6

// the Math.floor() function returns the largest integer less than or equal to a
var floorNumber = Math.floor(y)
var floorNumber2 = Math.floor(z)
console.log(floorNumber) // 5
console.log(floorNumber2) // 5

// the Math.random() function returns a floating-point, pseudo-random number betw
var randomNumber = Math.random()
console.log(randomNumber)

var randomNumber = Math.random() * 100
var roundRandomNumber = Math.round(randomNumber)
console.log(roundRandomNumber)
```


1.8 MAKE CONDITIONAL DECISION, IF-ELSE, COMPARISON

JAVASCRIPT

```
// if else
// The if statement executes a statement if a specified condition is truthy. If t

/* less than */
var coffeePrice = 12
```

```
if (coffeePrice < 10) {
  console.log('I will drink coffee')
} else {
  console.log('I will not drink coffee')
}

/* greater than */
if (coffeePrice > 10) {
  console.log('I will not drink coffee')
} else {
  console.log('I will drink coffee')
}

/* equal ==, === */
if (coffeePrice == 10) {
  console.log('I will not drink coffee')
} else {
  console.log('I will drink coffee')
}

/* not equal */

if (coffeePrice != 10) {
  console.log('I will not drink coffee')
} else {
  console.log('I will drink coffee')
}

// Multiple Condition
var job = true
var savingAmount = 5000000

/* Both condition must be fullfil */
if (job == true && savingAmount > 200000) {
  console.log('Well done')
} else {
  console.log('You need to work hard')
}
```

```
var job = true
var savingAmount = 500

if (job == true && savingAmount > 200000) {
  console.log('Well done')
} else {
  console.log('You need to work hard')
}

/* any of one condition must be fullfil */
if (job == true || savingAmount > 200000) {
  console.log('Well done')
} else {
  console.log('You need to work hard')
}

// if else if
if (job == true && savingAmount > 200000) {
  console.log('Well done!!')
} else if (job == true) {
  console.log('Save some money!!')
} else {
  console.log('You need to work hard!!')
}

// Date & Time Zone
var newDate = new Date()
console.log(newDate)

var newDate = new Date('1971-12-16')
console.log(newDate)
```


2 JAVASCRIPT FUNDAMENTAL CONCEPTS

2.1 ARRAY, INDEX, SET BY INDEX, INDEXOF

JAVASCRIPT

```
// declare an array, index start at 0
var friendsAge = [15, 17, 14, 16]
console.log(friendsAge) // [ 15, 17, 14, 16 ]

// finding element or item of an array using index
console.log(friendsAge[0]) // 15
console.log(friendsAge[3]) // 16

// store array element in variable using index
var sonaliAge = friendsAge[2]
console.log(sonaliAge) // 14

// change array element using index
friendsAge[1] = 21
console.log(friendsAge) // [ 15, 21, 14, 16 ]

// find element position (index) of certain element
var position = friendsAge.indexOf(14)
console.log(position) // 2

// if index is not available for the specified element, then output will be -1
var position = friendsAge.indexOf(141)
console.log(position) // -1
```


2.2 ARRAY ADVANCED, PUSH, POP, ARRAY LENGTH

JAVASCRIPT

```
// the push() method adds new items to the end of an array, and returns the new l
var friendsAge = [21, 22, 24, 20]
friendsAge.push(23)
friendsAge.push(25)
console.log(friendsAge) // [ 21, 22, 24, 20, 23, 25 ]

// length of an array
console.log(friendsAge.length) // 6
console.log(friendsAge['length']) // 6
```

```
// the pop() method removes the last element of an array, and returns that elemen
friendsAge.pop()
console.log(friendsAge) // [ 21, 22, 24, 20, 23 ]
```

2.3 ARRAY ADD AND REMOVE ELEMENT FROM THE BEGINNING AND SLICE

JAVASCRIPT

```
var teaLine = ['Abby', 'Smith', 'Bob']
console.log(teaLine) // [ 'Abby', 'Smith', 'Bob' ]

teaLine.push('Alex')
console.log(teaLine) // [ 'Abby', 'Smith', 'Bob', 'Alex' ]

teaLine.pop()
console.log(teaLine) // [ 'Abby', 'Smith', 'Bob' ]

// the shift() method removes the first item of an array.
// this method changes the length of the array.
// the return value of the shift method is the removed item.
var teaLine = ['Abby', 'Smith', 'Bob', 'Robert']
var teaLineShift = teaLine.shift()
console.log(teaLineShift) // Abby
console.log(teaLine) // [ 'Smith', 'Bob', 'Robert' ]

// the unshift() method adds new items to the beginning of an array, and returns
// this method changes the length of an array.
var teaLine2 = ['Abby', 'Smith', 'Bob', 'Robert']
teaLine2.unshift('Jane')
console.log(teaLine2) // [ 'Jane', 'Abby', 'Smith', 'Bob', 'Robert' ]

// the slice() method returns the selected elements in an array, as a new array o
// the slice() method doesn't changes the contents of an array
var newTeaLine = ['Abby', 'Smith', 'Bob', 'Robert', 'Jane', 'Anna']
var part1 = newTeaLine.slice(2) /*start from position 2 to end*/
console.log(part1) // [ 'Bob', 'Robert', 'Jane', 'Anna' ]

var part2 = newTeaLine.slice(2, 4) /*start from position 2 to 3, excluding 4*/
```

```
console.log(part2) // [ 'Bob', 'Robert' ]

var part3 = newTeaLine.slice(2, 5) /*start from position 2 to 4, excluding 5*/
console.log(part3) // [ 'Bob', 'Robert', 'Jane' ]

// the splice() method changes the contents of an array by removing or replacing
// let arrDeletedItems = array.splice(start[, deleteCount[, item1[, item2[, ...]]]

var months = ['Jan', 'March', 'April', 'June']

/*starting index number, ending index number, will remove item(s) based on this n
months.splice(1, 2)
console.log(months) // [ 'Jan', 'June' ]

months.splice(1)
console.log(months) // [ 'Jan' ]

/*Insert at index 1 */
var months = ['Jan', 'March', 'April', 'June']
months.splice(1, 0, 'Feb')
console.log(months) // [ 'Jan', 'Feb', 'March', 'April', 'June' ]

/*Insert at index 4 */
months.splice(4, 0, 'May')
console.log(months) // [ 'Jan', 'Feb', 'March', 'April', 'May', 'June' ]

/*replaces 1 element at index 4 */
var months = ['Jan', 'Feb', 'March', 'April', 'June']
months.splice(4, 1, 'May')
console.log(months) // [ 'Jan', 'Feb', 'March', 'April', 'May' ]
```

Additional Resources

2.4 WHILE LOOP, DEBUG JAVASCRIPT CODE, LESS OR EQUAL

JAVASCRIPT

```javascript
// the while loop loops through a block of code as long as a specified condition
var num1 = 0
while (num1 < 15) {
  console.log(num1) // 1 -> 14
  // num = num + 1;
  num1++
}

var num2 = 10
while (num2 < 15) {
  console.log(num2) // 10 -> 14
  num2++
}

var num3 = 10
while (num3 <= 15) {
  console.log(num3) // 10 -> 15
  num3++
}
```


2.5 FOR LOOP, RUN A LOOP FOR EACH ELEMENT OF AN ARRAY

JAVASCRIPT

```javascript
// a for loop repeats until a specified condition evaluates to false
// less than
for (var i = 0; i < 10; i++) {
  console.log(i) // 0 -> 9
}

// less than and equal
for (var i = 0; i <= 10; i++) {
  console.log(i) // 0 -> 10
}

// Use for loop for array to find each items
var nums = [55, 66, 77, 88, 99, 11, 44]
```

```
for (var i = 0; i < nums.length; i++) {
  var element = nums[i]
  console.log(element) // 55 66 77 88 99 11 44
}
```

2.6 JAVASCRIPT SWITCH CASE BREAK AND DEFAULT

JAVASCRIPT

```
/* not efficient
var num=5;
if (num>1000) {}
else if (num>100) {}
else if (num>50) {}
else if (num>20) {}
else if (num>10) {}
else if (num>5) {}
else if (num>0) {}
else {}
*/

// A switch statement can replace multiple if checks.
// It gives a more descriptive way to compare a value with multiple variants.
// The switch has one or more case blocks and an optional default.
// Break is important, otherwise will check all conditions

var num = 5
switch (num) {
  case 1000:
    console.log("I'm 1000")
    break
  case 100:
    console.log("I'm 100")
    break
  case 20:
  case 10:
    console.log("I'm either 20 or 10")
    break
```

```
    default:
      console.log("I don't know who you are") // output
}
```

[Additional Resources-1]

[Additional Resources-2]

2.7 FUNCTION, CALL FUNCTION

JAVASCRIPT

```
// a JavaScript function is defined with the function keyword, followed by a name
function functionName() {
  console.log("I'm Function")
}

// call a function
functionName() // I'm Function
```

2.8 FUNCTION PARAMETER, MULTIPLE PARAMETER, FUNCTION RETURN

JAVASCRIPT

```
/* function Parameters are the names that are define in the function definition a
real values passed to the function in function definition are known as arguments

function doubleIt(num) {
  var result = num * 2
  console.log(result)
}

doubleIt(5) // 10
doubleIt(50) // 100

/*When a return statement is used in a function body, the execution of the functi
stopped. If specified, a given value is returned to the function caller. */
```

```
function doubleIt(num) {
  var result = num * 2
  return result
}

var first = doubleIt(6)
var second = doubleIt(40)
console.log(first, second) // 12 80

var total = first + second
console.log(total) // 92

function add(a, b) {
  var c = a + b
  return c
}

var sum = add(15, 17)
console.log(sum) // 32
```


2.9 COMMENT, MULTIPLE LINES COMMENT

JAVASCRIPT

```
// single line comment, Ctrl + /
/* multiline comment,
  Shift + Alt + A */
```

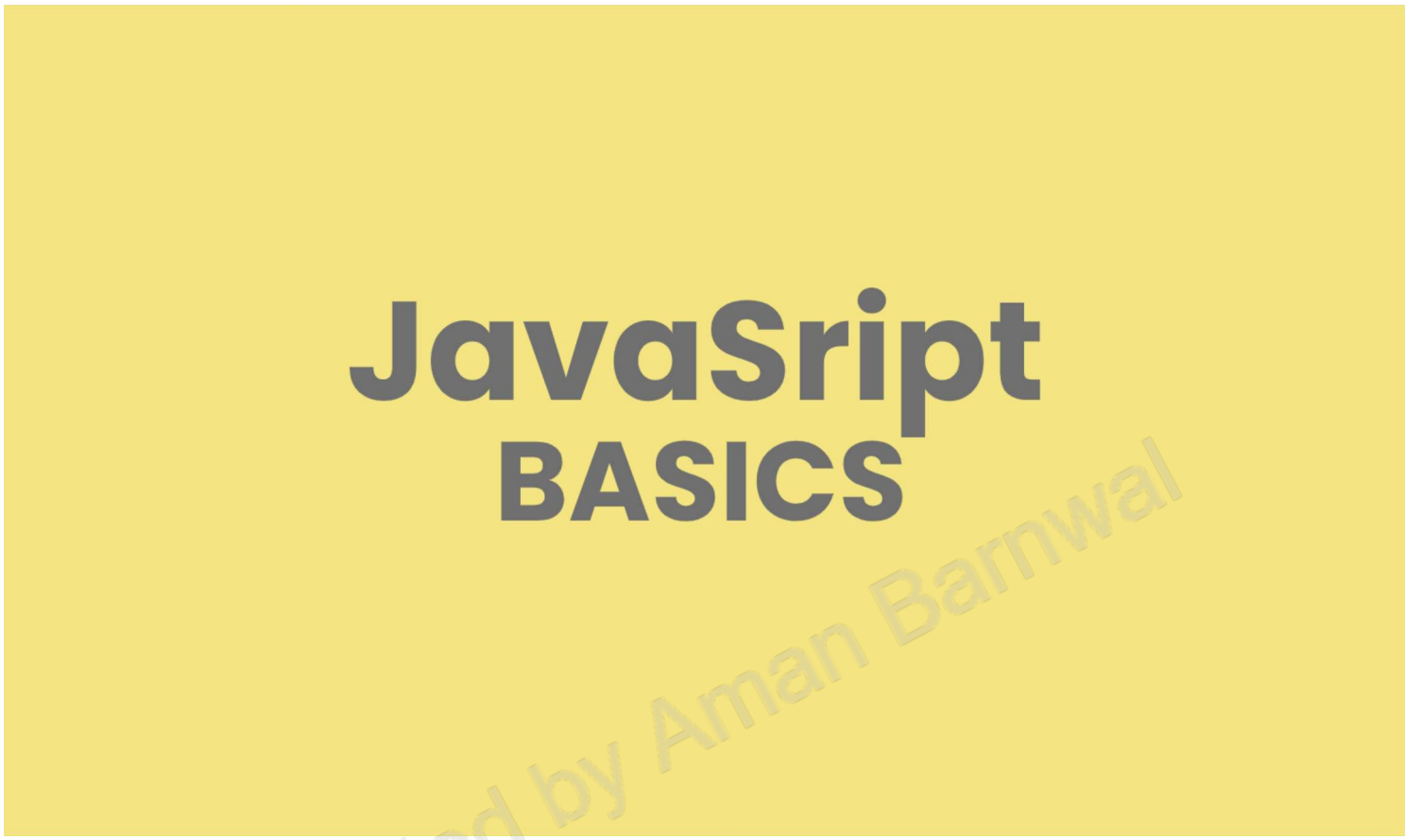
2.10 OBJECT, KEY VALUE PAIR, GET OBJECT PROPERTY, SET VALUE

JAVASCRIPT

```
/* An object is a collection of properties, and a property is an association betw
(or key) and a value. A property's value can be a function, in which case the pro
known as a method. */
```

```
// {propertyName: PropertyValue}
var student1 = {
  id: 121,
  phone: 347,
  name: 'Abir',
}

var student2 = {
  id: 131,
  phone: 929,
  name: 'Mahi',
}

console.log(student1) // { id: 121, phone: 347, name: 'Abir' }
console.log(student2) // { id: 131, phone: 929, name: 'Mahi' }

// access object property
var phoneNo1 = student1.phone /* option-1 */
console.log(phoneNo1) // 347

var phoneNo2 = student1['phone'] /* option-2 */
console.log(phoneNo2) // 347;

var phonePropName = 'phone' /* option-3 */
var phoneNo3 = student1[phonePropName]
console.log(phoneNo3) // 347;

// change phone number
student2.phone = 542854
student2['phone'] = 66688
student2[phonePropName] = 9999
console.log(student2)
// { id: 131, phone: 9999, name: 'Mahi' }

// add new property
student2.cinema = 'Ogni2'
student2['cinema'] = 'Smart girl'
console.log(student2)
// { id: 131, phone: 9999, name: 'Mahi', cinema: 'Smart girl' }
```

JavaScript Basics Part- 2



1 APPLY JAVASCRIPT CONCEPTS

1.1 UNIT CONVERT INCH TO FEET USE VARIABLE AND FUNCTION

JAVASCRIPT

```
// without function, not reusable:
var inch = 156
var feet = inch / 12
console.log(feet) // 13

// with function:
function inchToFeet(inch) {
  var feet = inch / 12
  return feet
```

```
}

var newFeet = inchToFeet(300)
console.log(newFeet) // 25

// using index of an array:
var inchArr = [156, 288, 300]

var newFeet = inchToFeet(inchArr[1])
console.log(newFeet) // 24

// using for loop:
var inchArr = [156, 288, 300]

for (i = 0; i < inchArr.length; i++) {
  var feet = inchArr[i] / 12
  console.log(feet) // 13 24 25
}
```

1.2 VARIABLE LET AND CONST AND HOW TO USE THEM

JAVASCRIPT

```
// use var or let (mostly) if you want to reassign
let name = 'Javed Akhter'
console.log(name.length) /* including space */

if (name.length > 4) {
  name = 'Juvu'
}
console.log(name)

// if variable will be fixed or no need to change use const
const country = 'Bangladesh'
console.log(country)

const age = 15
console.log(age)
```

```
/* apply const wherever possible */
function inchToFeet(inch) {
  const feet = inch / 12
  return feet
}

const newFeet = inchToFeet(300)
console.log(newFeet)

// using index of an array:
const inchArr = [156, 288, 300]

const newFeetArry = inchToFeet(inchArr[1])
console.log(newFeetArry)
```

1.3 CHECK WHETHER A YEAR IS A LEAP YEAR OR NOT

JAVASCRIPT

```
/* divisible by 4 for leap is not always true */
console.log(2030 / 4)
console.log(2032 / 4)

const year = 3996
const remainder = year % 4
console.log(remainder)
console.log(remainder == 0)

if (remainder == 0) {
  console.log('The year is a Leap Year')
} else {
  console.log('The year is not a Leap Year')
}

// using function:
function isLeapYear(year) {
  const remainder = year % 4

  if (remainder == 0) {
```

```
    return true
  } else {
    return false
  }
}

const checkYear1 = isLeapYear(2016)
console.log(checkYear1)

const checkYear2 = isLeapYear(1700)
console.log(checkYear2) /* not true */
```

Correct way of finding leap year:
To determine whether a year is a leap year, follow these steps:

1. If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
2. If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
3. If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
4. The year is a leap year (it has 366 days).
5. The year is not a leap year (it has 365 days).

or

1. If a year is divisible by 400, it's a leap year
2. Otherwise, if a year is divisible by 100, it's not a leap year
3. Otherwise, if a year is divisible by 4, it's a leap year

JAVASCRIPT

```
// Method: 1
function isLeapYear(year) {
  if (year % 400 === 0) return true
  if (year % 100 === 0) return false
  return year % 4 === 0
}

console.log(isLeapYear(1700)) // false
```

```
console.log(isLeapYear(2016)) // true
console.log(isLeapYear(2000)) // true
console.log(isLeapYear(100)) // false

// Method: 2
// function isLeapYear2(year) {
// return (year % 4 == 0 && year % 100 != 0) || year % 400 == 0;
// }

function isLeapYear2(year) {
  if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {
    return 'The year is a leap Year'
  } else {
    return 'The Year is not a leap year'
  }
}

console.log(isLeapYear2(1700)) // The Year is not a leap year
console.log(isLeapYear2(2016)) // The year is a leap Year
console.log(isLeapYear2(2000)) // The year is a leap Year
console.log(isLeapYear2(100)) // The Year is not a leap year

// Method 3:
function isLeapYear3(year) {
  return year % 100 === 0 ? year % 400 === 0 : year % 4 === 0
}

console.log(isLeapYear(1700)) // false
console.log(isLeapYear(2016)) // true
console.log(isLeapYear(2000)) // true
console.log(isLeapYear(100)) // false
```


[Additional Resources-1]

[Additional Resources-2]

[Additional Resources-3]

1.4 CALCULATE FACTORIAL OF A NUMBER USING FOR LOOP

JAVASCRIPT

```
/* Factorial e.g.
4! = 4*3*2*1
10!=1*2*3*4*5*6*7*8*9*10 */

/* assume initial value such that it will not affect the result,
 e.g. for multiply assume initial value 1, for addition 0 */

// iterative method, for loop:
let factorial = 1
for (let i = 1; i <= 10; i++) {
  factorial = factorial * i
  console.log(i, factorial)
}

// using function:
function newFactorial(n) {
  let factorial = 1
  for (let i = 1; i <= n; i++) {
    factorial = factorial * i
  }
  return factorial
}

let result = newFactorial(5)
console.log(result) // 120
```


1.5 CALCULATE FACTORIAL OF A NUMBER USING A WHILE LOOP

JAVASCRIPT

```
// 10!=1*2*3*4*5*6*7*8*9*10
// using while loop:
let i = 1
let factorial = 1
while (i <= 10) {
```

```
  factorial = factorial * i
  // console.log(i, factorial);
  i++
}

console.log(factorial) // 3628800

// using function and while:
function newFactorial(n) {
  let i = 1
  let newFactorial = 1
  while (i <= n) {
    newFactorial = newFactorial * i
    i++
  }
  return newFactorial
}

let result = newFactorial(6)
console.log(result) // 720

// reverse way:
// 10!=1*2*3*4*5*6*7*8*9*10

// Using For loop:
let factorial1 = 1
for (let i = 5; i > 1; i--) {
  factorial1 = factorial1 * i
  console.log(i, factorial1)
}

function factorialize(n) {
  let i = n
  let newNum = 1
  for (i; i > 1; i--) {
    newNum *= i /* newNum = newNum * i */
  }
  return newNum
}
```

```
console.log(factorialize(6)) // 720

// using while:
let i = 5
let factorial2 = 1
while (i >= 1) {
  factorial2 = factorial2 * i
  i--
}
console.log(factorial2) // 120

function factorialize2(n) {
  let i = n
  let factorial2 = 1
  while (i >= 1) {
    factorial2 = factorial2 * i
    i--
  }
  return factorial2
}

console.log(factorialize2(6)) // 720
```


1.6 CALCULATE FACTORIAL IN A RECURSIVE FUNCTION

JAVASCRIPT

```
// Recursive function:

// 10!=1*2*3*4*5*6*7*8*9*10
// 0! = 1
// 2! = 1*2
// 3! = 1*2*3
// 4! = 1*2*3*4
// 5! = 1*2*3*4*5 = 4!*5 = (5-1)! * 5
// n! = (n-1)! * n

function rFactorial(n) {
  if (n == 0) {
```

```
    return 1
  } else {
    return rFactorial(n - 1) * n
  }
}

let nResult = rFactorial(10)
console.log(nResult) // 3628800
```

Additional Resource

1.7 CREATE A FIBONACCI SERIES USING A FOR LOOP

About Fibonacci Series

JAVASCRIPT

```
/*
position 2, 3, 4, 5, n ,i respectively:
fibo[2] = fibo[2 - 1] + fibo[2 - 2];
fibo[3] = fibo[3 - 1] + fibo[3 - 2];
fibo[4] = fibo[4 - 1] + fibo[4 - 2];
fibo[5] = fibo[5 - 1] + fibo[5 - 2];
fibo[n] = fibo[n - 1] + fibo[n - 2];
fibo[i] = fibo[i - 1] + fibo[i - 2]; */

// using For loop:
let fibo = [0, 1]

for (let i = 2; i <= 12; i++) {
  fibo[i] = fibo[i - 1] + fibo[i - 2]
  // console.log(fibo[i], fibo[i - 1], fibo[i - 2])
}

console.log(fibo)
// [ 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 ]

// using function:
function fibonacci(n) {
```

```
  let fibo = [0, 1]

  for (let i = 2; i <= n; i++) {
    fibo[i] = fibo[i - 1] + fibo[i - 2]
  }
  return fibo
}

let fiboResult = fibonacci(12)
console.log(fiboResult)
// [ 0,  1,  1,  2,  3,  5, 8, 13, 21, 34, 55, 89, 144 ]
```

1.8 FIBONACCI ELEMENT IN A RECURSIVE WAY

JAVASCRIPT

```
function rFibonacci(n) {
  if (n == 0) {
    return 0
  }
  if (n == 1) {
    return 1
  } else {
    return rFibonacci(n - 1) + rFibonacci(n - 2)
  }
}

let rResult = rFibonacci(10)
console.log(rResult) // 55
```


Additional Resource

1.9 CREATE FIBONACCI SERIES IN A RECURSIVE WAY

JAVASCRIPT

```
// [0 , 1, 1, 2, 3, 5, 8,, 13, 21,]
```

```
function sFibonacci(n) {
  if (n == 0) {
    return [0]
  } else if (n == 1) {
    return [0, 1]
  } else {
    // find array with nth item
    let fibo = sFibonacci(n - 1)
    let nextItem = fibo[n - 1] + fibo[n - 2]
    fibo.push(nextItem)
    return fibo
  }
}

let sResult1 = sFibonacci(1)
let sResult2 = sFibonacci(2)
let sResult3 = sFibonacci(3)
let sResult4 = sFibonacci(4)
let sResult8 = sFibonacci(10)
console.log(sResult1) // [ 0, 1 ]
console.log(sResult2) // [ 0, 1, 1 ]
console.log(sResult3) // [ 0, 1, 1, 2 ]
console.log(sResult4) // [ 0, 1, 1, 2, 3 ]
console.log(sResult8) // [ 0, 1,  1,  2,  3, 5, 8, 13, 21, 34, 55]
```


1.10 CHECK WHETHER A NUMBER IS A PRIME NUMBER OR NOT

JAVASCRIPT

```
function isPrime(n) {
  for (i = 2; i < n; i++) {
    // console.log(i, n % i);
    if (n % i == 0) {
      return 'Not a Prime Number'
    }
  }
  return 'The number is a Prime number'
}
```

```
let pResult1 = isPrime(128)
let pResult2 = isPrime(79)

console.log(pResult1) // Not a Prime Number
console.log(pResult2) // The number is a Prime number
```

2 JAVASCRIPT CODING PROBLEMS, SIMPLE INTERVIEW QUESTIONS

2.1 SWAP VARIABLE, SWAP WITHOUT TEMP, DESTRUCTING

JAVASCRIPT

```
let a = 5
let b = 7
console.log('Before swap: a=', a, ', b=', b)
// Before swap: a= 5 , b= 7

// Swap using a temporary variable:
let temp = a /* store a value in temp variable */
a = b /* overwrite a value with a value */
b = temp /* overwrite b value with temp value */

console.log('After swap: a=', a, ', b=', b)
// After swap: a= 7 , b= 5

// Swap using addition and difference:
let x = 5
let y = 7
console.log('Before swap: x=', x, ', y=', y)
// Before swap: x= 5 , y= 7

x = x + y
/* add current x and current y so that we can get combined
value of x and y (in this case 12) and store it in new x */
y = x - y /* now new y equal the new x minus y, which is 12-7= 5 */
x = x - y /* now to transfer y value, overwrite new x, 12-5= 7 */
```

```
console.log('After swap: x=', x, ', y=', y)
// After swap: x= 7 , y= 5

// Swap using destructing assignment:
let p = 5
let q = 7
console.log('Before swap: p=', p, ', q=', q)
// Before swap: p= 5 , q= 7
;[p, q] = [q, p]
/* a temporary array [q, p] is created, which evaluates to [7,5] */
console.log('After swap: p=', p, ', q=', q)
// After swap: p= 7 , q= 5
```

Additional Resource

2.2 RANDOM NUMBER, RANDOM NUMBER BETWEEN 1 TO 6

JAVASCRIPT

```
let num = 2.12458

// the Math.floor() function returns the largest integer less than or equal to a
let result1 = Math.floor(num)
console.log(result1) // 2

// the Math.ceil() function always rounds a number up to the next largest integer
let result2 = Math.ceil(num)
console.log(result2) // 3

// the Math.round() function returns the value of a number rounded to the nearest
let result3 = Math.round(num)
console.log(result3) // 2

// random number 0 t0 10:
let dice = Math.random() * 6 /* with decimal */
console.log(dice) // example: 1.2463552614334286

let output = Math.round(dice) /* without decimal */
console.log(output) // example: 1
```

```
for (i = 0; i < 10; i++) {
  let dice = Math.random() * 6
  let output = Math.round(dice)
  console.log('For Loop: ', output)
}
```

2.3 FIND MAX OF TWO VALUES, FIND MAX OF THREE VALUES

JAVASCRIPT

```
let business = 450
let minister = 350
let sochib = 750

// if compare business and minister only:
if (business > minister) {
  console.log('Business is bigger') // output
} else {
  console.log('Minister is bigger')
}

// if compare business, minister sochib:
if (business > minister) {
  if (business > sochib) {
    console.log('Business is bigger')
  } else {
    console.log('Sochib is bigger')
  }
} else {
  if (minister > sochib) {
    console.log('Minister is bigger')
  } else {
    console.log('sochib is bigger')
  }
}
// output: sochib is bigger
```

```
// the Math.max() function returns the largest of the zero or more numbers given

let max = Math.max(business, minister, sochib)
console.log(max) // 750
// output: 750

console.log(Math.max(1, 3, 2))
// output: 3

console.log(Math.max(-1, -3, -2))
// output: -1

const array1 = [1, 3, 2]

console.log(Math.max(...array1))
// output: 3
```

Additional Resources- 1

Additional Resources- 2

2.4 FIND THE LARGEST ELEMENT OF AN ARRAY

JAVASCRIPT

```
const marks = [45, 81, 63, 98, 56, 35, 23]
let max = marks[0]

for (let i = 0; i < marks.length; i++) {
  // debugger;
  let element = marks[i]
  if (element > max) {
    max = element
  }
}

console.log('Highest value is: ', max)
// Highest value is:  98
```

2.5 SUM OF ALL NUMBERS IN AN ARRAY

JAVASCRIPT

```
let numbers = [45, 65, 68, 12, 3, 54, 65]

let sum = 0
for (let i = 0; i < numbers.length; i++) {
  let element = numbers[i]
  sum = sum + element
}

console.log('Total of the numbers: ', sum)
// Total of the numbers:  312

// using function:
function getArrSum(arrayNumbers) {
  let sum = 0
  for (let i = 0; i < arrayNumbers.length; i++) {
    let element = arrayNumbers[i]
    sum = sum + element
  }
  return sum
}

let num = [45, 65, 68, 12, 3]
let getArrSumResult = getArrSum(num)

console.log('Total: ', getArrSumResult) // Total:  193
console.log('Total: ', getArrSum([50, 100, 20, 10])) // Total:  180
```


Additional Resources- 1

Additional Resources- 2

2.6 REMOVE DUPLICATE ITEM FROM AN ARRAY

JAVASCRIPT

```
let arrNum = [3, 6, 2, 7, 3, 2, 8, 1, 9, 11, 56]

let uniqueNum = []
for (let i = 0; i < arrNum.length; i++) {
  let element = arrNum[i] /* each index element */
  let index = uniqueNum.indexOf(
    element
  ) /* check the element index in uniqueNum */

  if (index == -1) {
    /* when the element is not found in uniqueNum, push will add */
    uniqueNum.push(element)
  }
}
console.log(uniqueNum)
// [ 3, 6, 2, 7, 8, 1, 9, 11, 56 ]

// using function:
function findUnique(arrayX) {
  let uniqueNum = []
  for (let i = 0; i < arrayX.length; i++) {
    let element = arrayX[i]
    let index = uniqueNum.indexOf(element)

    if (index == -1) {
      uniqueNum.push(element)
    }
  }
  return uniqueNum
}

console.log(findUnique([5, 5, 7, 9, 0, 5, 9, 0, 3]))
// [ 5, 7, 9, 0, 3 ]
```


2.7 COUNT THE NUMBER OF WORDS IN A STRING

JAVASCRIPT

```
let speech = "I am a good person. I don't snore at night"
// console.log(speech.length);
// console.log(speech[3]);

// Target spaces between words
let count = 0
for (let i = 0; i < speech.length; i++) {
  let char = speech[i]
  // console.log(char);

  if (char == ' ' && speech[i - 1] != ' ') {
    /* speech[i - 1] is applied to ignor double spaces */
    count++
    /* 1 less than expected word count cause there is no space after last word */
  }
}
count++ /* add 1 to fix the prior count */

console.log(count) // 10
```


2.8 REVERSE A STRING

JAVASCRIPT

```
function reverseString(str) {
  let reverse = ''
  for (let i = 0; i < str.length; i++) {
    let char = str[i]
    reverse = char + reverse /* put char in the beginning to reverse */
  }
  return reverse
}

let statement = 'Hello Alien, How are you doing?'
let reverseStrOut = reverseString(statement)
console.log(reverseStrOut)
// ?gniod uoy era woH ,neilA olleH
```

3 Exercise

3.1 FIND THE LARGEST VALUE

JAVASCRIPT

```
// Write an function to find largest value.
function largestNumber(numbers) {
  var larger = numbers[0]
  for (var i = 0; i < numbers.length; i++) {
    var element = numbers[i]
    if (element > larger) {
      larger = element
    }
  }
  return larger
}

console.log(largestNumber([45, 78, 89, 23])) // 89
```

3.2 WRITE TWO FUNCTIONS TO FIND FACTORIAL

JAVASCRIPT

```
// Find factorial using iteration
// 5!=5*4*3*2*1
function factorial(num) {
  var fact = 1
  for (var i = 1; i <= num; i++) {
    fact = fact * i
  }
  return fact
}

console.log(factorial(5)) // 120

// Write a recursive function to find factorial.
```

```
function rFactorial(num) {
  if (num == 1) {
    return 1
  } else {
    return num * rFactorial(num - 1)
  }
}

console.log(rFactorial(5)) // 120
```

3.3 FIND THE NUMBER OF ANIMAL

JAVASCRIPT

```
/* assume you went to amazon forest, the more deeper you will go inside the fores
the chance of seeing more animals is greater.
Let's assume from the beginning of the forest to first 10 miles, you had saw 50 a
then second 10 miles 100 animals, and third 10 miles or greater 300 animals respe
Write a function named animalCalculator to find the totalanimal for a given input

var depth = 12

function animalCalculator(depth) {
  var animal = 0

  if (depth <= 10) {
    animal = depth * 5
  } else if (depth <= 20) {
    var firstPart = 10 * 50
    var remaining = depth - 10
    var secondPart = remaining * 100
    animal = firstPart + secondPart
  } else {
    var firstPart = 10 * 50
    var secondPart = 10 * 100
    var remaining = depth - 20
    var thirdPart = remaining * 300
    animal = firstPart + secondPart + thirdPart
  }
```

```
  return animal
}

console.log(animalCalculator(32)) // 5100
```

JavaScript Basics Part- 3

1 HOW JAVASCRIPT WORKS & DOM

1.1 CREATE SCRIPT TAG AND CONNECT EXTERNAL SCRIPT FILE

HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <style>
      body {
        background-color: aquamarine;
      }
    </style>
  </head>
```

```
  <body>
    <h1>Hello Javascript</h1>
    <h1>Hello Javascript</h1>
    <h1>Hello Javascript</h1>
    <h1>Hello Javascript</h1>
    <h1>Hello Javascript</h1>

    <!-- Internal JS -->
    <script>
      console.log(45)
      // alert("Hello JavaScript");
    </script>

    <!-- External JS -->
    <script src="scripts/second.js"></script>
    <script src="scripts/first.js"></script>
  </body>
</html>
```


1.2 HOW JAVASCRIPT RUN AND WHY SCRIPT ORDER IS IMPORTANT

HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Learning Javascript</title>
    <style>
      body {
        background-color: aquamarine;
      }
    </style>
  </head>
  <body>
    <h1>Hello Javascript: 1</h1>
    <script src="scripts/first.js"></script>
    <h1>Hello Javascript: 2</h1>
```

```
    <script src="scripts/second.js"></script>
    <h1>Hello Javascript: 3</h1>
    <h1>Hello Javascript: 4</h1>

    <!-- Internal JS -->
    <script>
      console.log(45)
      debugger
      console.log(145)
    </script>
    <h1>Hello Javascript: 5</h1>

    <!-- Output will be based on place of JS orders -->

    <!-- HOW JAVASCRIPT RUN AND WHY SCRIPT ORDER IS IMPORTANT -->
    <!-- JS is High level interpreted language, single threaded, non-blocking -->
  </body>
</html>
```

JAVASCRIPT

```
// first.js
console.log(81)
debugger

// second.js
console.log(120)
debugger
```


Additional Resource

1.3 WHAT IS DOM (DOCUMENT OBJECT MODEL)

HTML

```
<body>
  <main>
    <section>
```

```
        <h3>Exploring DOM</h3>
        <p>
          Lorem ipsum dolor sit amet consectetur adipisicing elit. Magni voluptate
          vitae animi corrupti ipsa dolorem eum deserunt, in voluptates explicabo
          quaerat amet, tenetur unde. Beatae deleniti eveniet rerum eligendi
          laboriosam?
        </p>

        <p>
          Lorem ipsum dolor sit amet consectetur adipisicing elit. Quia, fuga?
        </p>
      </section>
      <!-- Chrome browser ->Console -> type document -->
    </main>

    <script>
      var user = {
        name: 'Jalali Set',
        song: 'Kakatua',
        members: 6,
      }
      // DOM = Document Object Model;
      // console.log(document);
      console.log(document.body)
    </script>
  </body>
```


Additional Resources- 1

Additional Resources- 2

1.4 CAPTURE ELEMENTS FROM HTML FILE USING GETELEMENTBYID

HTML

```
<body>
  <main>
    <section>
```

```
      <h3>Exploring DOM</h3>
      <p id="first">
        Lorem ipsum dolor sit amet consectetur adipisicing elit. Magni voluptate
        vitae animi corrupti ipsa dolorem eum deserunt, in voluptates explicabo
        quaerat amet, tenetur unde. Beatae deleniti eveniet rerum eligendi
        laboriosam?
      </p>

      <p>
        Lorem ipsum dolor sit amet consectetur adipisicing elit. Quia, fuga?
      </p>
    </section>
    <!-- Chrome browser ->Console -> type document -->
  </main>

  <script>
    // document.getElementsByTagName;
    // document.getElementById("first");
    var first = document.getElementById('first')
    first.style.color = 'darkblue'
    first.style.fontSize = '30px'
    first.style.backgroundColor = 'grey'
  </script>
</body>
```


1.5 HOW TO USE GETELEMENTSBYCLASSNAME AND CHANGE INNERHTML

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Explore DOM</title>
    <style>
        article {
            border: 2px solid tomato;
```

```
            border-radius: 20px;
            margin: 5px;
            padding: 5px;
        }
    </style>
</head>
<body>
  <main>
        <section>
            <h3>Exploring DOM</h3>
            <!-- article*3>h3{article-$}+p.author{author-$}+p.description>lorem30
            <article>
                <h3>article-1</h3>
                <p class="author">author-1</p>
                <p class="description">Lorem ipsum dolor sit amet consectetur adi
                    nulla. Dicta delectus aut minus. Veritatis cumque consectetur
                    nam, perspiciatis distinctio nisi nobis esse doloremque.</p>
            </article>
            <article>
                <h3>article-2</h3>
                <p class="author">author-2</p>
                <p class="description">Architecto illum earum, reprehenderit assu
                    quidem corporis reiciendis quas incidunt nihil possimus unde
                    tempora error laborum consequatur atque similique sapiente ex
            </article>
            <article>
                <h3>article-3</h3>
                <p class="author">author-3</p>
                <p class="description">Quos dolorum, laborum voluptate culpa, nat
                    magni, exercitationem voluptates consectetur quibusdam dolore
                    nulla reiciendis officia, placeat delectus aut corrupti?</p>
            </article>

    </main>

   <script>
        // document.getElementsByClassName('author');
        var authors = document.getElementsByClassName('author');
        for(var i=0; i<authors.length; i++){
            var element = authors[i];
```

```
            // console.log(element);
            console.log(element.innerHTML);
            // element.innerHTML ="Lekhok- " + i;
            element.innerHTML ="Lekhok- " + (i+1);
            element.style.backgroundColor = 'lightblue';
            element.style.margin ='10px';
        }
    </script>
</body>
</html>
```

1.6 HOW TO USE QUERYSELECTOR AND QUERYSELECTORALL & NODE, NODE TYPE, NODELIST, HTMLCOLLECTION, SETATTRIBUTE

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Explore DOM</title>
    <style>
        article {
            border: 2px solid tomato;
            border-radius: 20px;
            margin: 5px;
            padding: 5px;
        }
        .special h3 {}
    </style>
</head>

<body>
  <main>
        <section>
            <h3>Exploring DOM</h3>
            <!-- article*3>h3{article-$}+p.author{author-$}+p.description>lorem30
```

```
        <article>
            <h3>article-1</h3>
            <p title="best seller author" id="best-seller" class="author">aut
            <p class="description">Lorem ipsum dolor sit amet consectetur adi
                nulla. Dicta delectus aut minus. Veritatis cumque consectetur
                nam, perspiciatis distinctio nisi nobis esse doloremque.</p>
        </article>
        <article class="special">
            <h3>article-2</h3>
            <p class="author">author-2</p>
            <p class="description">Architecto illum earum, reprehenderit assu
                quidem corporis reiciendis quas incidunt nihil possimus unde
                tempora error laborum consequatur atque similique sapiente ex
        </article>
        <article class="special">
            <h3>article-3</h3>
            <p class="author">author-3</p>
            <p class="description">Quos dolorum, laborum voluptate culpa, nat
                magni, exercitationem voluptates consectetur quibusdam dolore
                nulla reiciendis officia, placeat delectus aut corrupti?</p>
        </article>

</main>

<script>
    // HOW TO USE QUERYSELECTOR AND QUERYSELECTORALL
    // document.querySelector('.special h3'); will give only the first matchi
    // document.querySelectorAll('.special h3'); will give all matching

    document.body.style.backgroundColor = 'tomato';
    var authors = document.querySelectorAll('.special h3');

    for (var i = 0; i < authors.length; i++) {
        var element = authors[i];
        // console.log(element);
        // console.log(element.innerHTML);
        // element.innerHTML ="Lekhok- " + i;
        element.innerHTML = "Lekhok- " + (i + 1);
        element.style.backgroundColor = 'lightblue';
        element.style.margin = '10px';
```

```
            element.setAttribute('title', 'all author are same');
        }

        // document.querySelector('.special h3').style.backgroundColor = 'Tomato'

        // NODE, NODE TYPE, NODELIST, HTMLCOLLECTION, SETATTRIBUTE
        // Node
        // Node List

        document.querySelector('h3').setAttribute('title', "You are the title");
        document.getElementById('best-seller').setAttribute('title', "give me you
    </script>
</body>
</html>
```

[Additional Resource]

1.7 HOW TO ADD ELEMENTS TO HTML USING JAVASCRIPT

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Explore DOM</title>
    <style>
        article {
            border: 2px solid tomato;
            border-radius: 20px;
            margin: 5px;
            padding: 5px;
        }
        .special h3 {}
    </style>
</head>
<body>
```

```
<main>
      <section>
          <h3>Exploring DOM</h3>
          <!-- article*3>h3{article-$}+p.author{author-$}+p.description>lorem30
          <article id="first-article">
              <h3>article-1</h3>
              <p title="best seller author" id="best-seller" class="author">aut
              <p class="description">Lorem ipsum dolor sit amet consectetur adi
                  nulla. Dicta delectus aut minus. Veritatis cumque consectetur
                  nam, perspiciatis distinctio nisi nobis esse doloremque.</p>
          </article>
          <article class="special">
              <h3>article-2</h3>
              <p class="author">author-2</p>
              <p class="description">Architecto illum earum, reprehenderit assu
                  quidem corporis reiciendis quas incidunt nihil possimus unde
                  tempora error laborum consequatur atque similique sapiente ex
          </article>
          <article class="special">
              <h3>article-3</h3>
              <p class="author">author-3</p>
              <p class="description">Quos dolorum, laborum voluptate culpa, nat
                  magni, exercitationem voluptates consectetur quibusdam dolore
                  nulla reiciendis officia, placeat delectus aut corrupti?</p>
          </article>

          <article>
              <h3>Grocery List</h3>
              <ul id="gift-list">
                  <li>gift item- 1</li>
                  <li>gift item- 2</li>
                  <li>gift item- 3</li>
                  <li>gift item- 4</li>
                  <li>gift item- 5</li>
              </ul>
          </article>

  </main>

  <script>
```

```
        const article = document.getElementById('first-article');
        const newParagraph = document.createElement('p');
        newParagraph.innerHTML = 'This is added by JavaScript';
        article.appendChild(newParagraph);

        // add one more gift

        const ul = document.getElementById('gift-list');
        const li = document.createElement('li');
        li.innerHTML = "Brand New Gift";
        ul.appendChild(li);

        // document.getElementById('gift-list').childNodes
        // document.getElementById('gift-list').parentNode
        // document.getElementById('first-article').childNodes
    </script>
</body>
</html>
```


2 FUNCTION, ADD EVENT LISTENER, EVENT BUBBLE

2.1 WHEN TO USE A FUNCTION, FUNCTION INSIDE AN ARRAY

JAVASCRIPT

```javascript
let nums = [5, 12, 89, 45, 18, 8]

for (let i = 0; i < nums.length; i++) {
  const num = nums[i]
  // console.log(num * 2);
  // odd and even
  if (num % 2 == 0) {
    console.log(num, ': is even number')
  } else {
    // console.log(num, ':is odd number');
    console.log(num * 2, ':is odd number')
  }
}
```

```
// New array
let frndAge = [13, 17, 19, 20, 18]

for (let i = 0; i < frndAge.length; i++) {
  const age = frndAge[i]
  // console.log(age);
  if (age % 2 == 0) {
    console.log(age, ': is even number')
  } else {
    console.log(age * 2, ':is odd number')
  }
}

// repeated or duplicated same thing in both calculation

// create function to avoid duplicate code, call function inside
function evenify(num) {
  if (num % 2 == 0) {
    console.log(num, ': is even number')
  } else {
    console.log(num, ':is odd number')
  }
}

// call function inside another function
function evenifyAll(nums) {
  for (let i = 0; i < nums.length; i++) {
    const num = nums[i]
    evenify(num)
  }
}

// without writing multiple functions
function evenifyAll(nums) {
  for (let i = 0; i < nums.length; i++) {
    const num = nums[i]
    if (num % 2 == 0) {
      console.log(num, ': is even number')
    } else {
```

```
      console.log(num, ':is odd number')
    }
  }
}

// call function inside loop

let nums = [7, 13, 89, 45, 18, 8]

for (let i = 0; i < nums.length; i++) {
  const num = nums[i]
  evenify(num)
}

// call function inside loop
let frndAge = [13, 17, 19, 20, 18]
for (let i = 0; i < frndAge.length; i++) {
  const age = frndAge[i]
  // console.log(age);
  evenify(age)
}

// call the function which created with loop
evenifyAll(frndAge)
```


2.2 WHEN TO RETURN FROM A FUNCTION AND FROM WHERE

JAVASCRIPT

```
// return in all statement
function evenify(num) {
  if (num % 2 == 0) {
    // console.log(num, ':is even number');
    return num
  } else {
    // console.log(num, ':is odd number');
    return num * 2
  }
}
```

```
let result = evenify(13)
console.log('result', result)

// return after all statement
function evenify(num) {
  let result
  if (num % 2 == 0) {
    result = num
  } else {
    result = num * 2
  }
  return result
}

let result2 = evenify(15)
console.log('result', result2)

let square = result2 * result2
console.log('squre', square)

// array with even number
function evenifyAll(nums) {
  let evenArray = []
  for (let i = 0; i < nums.length; i++) {
    const num = nums[i]

    let result = evenify(num)
    evenArray.push(result)
  }
  return evenArray
}

let nums = [3, 6, 18, 19, 20, 21]
let numsEven = evenifyAll(nums)
console.log(numsEven)
```


2.3 CALLBACK FUNCTION AND PASS DIFFERENT FUNCTION

JAVASCRIPT

```
function explnCallback(name, age, task) {
  console.log('Hello', name)
  console.log('Your age', age)
  // washHand();
  // takeShower();
  // console.log(task);
  task()
}

function washHand() {
  console.log('Wash hand with soap')
}

function takeShower() {
  console.log('Take Shower')
}

function scrollFB() {
  console.log('Scrolling FB')
}
// explnCallback('Robert', 17);
// explnCallback('smith', 13);

explnCallback('Robert', 17, washHand)
/* Hello Robert
  Your age 17
  Wash hand with soap */
explnCallback('smith', 13, takeShower)
/* Hello smith
  Your age 13
  Take Shower */
explnCallback('Jane', 21, scrollFB)
/* Hello Jane
  Your age 21
  Scrolling FB */
```


2.4 ARGUMENTS AND DEAL WITH UNKNOWN NUMBER OF ARGUMENTS

JAVASCRIPT

```javascript
function addNumber(num1, num2) {
  return num1 + num2
}

let result = addNumber(3, 5)
console.log(result) // 8

// add more parameter value than specified in function
let result2 = addNumber(3, 5, 7, 15)
console.log(result2) // 8

// read parameter value using arguments, arguments work inside function
function addNumber2(num1, num2) {
  // console.log(arguments);
  console.log(arguments[3]) // 15
  return num1 + num2
}
let result2 = addNumber2(3, 5, 7, 15)
console.log(result2) // 8

// arguments
function addNumber3(num1, num2) {
  // console.log(arguments[3]);
  let sum = 0
  for (let i = 0; i < arguments.length; i++) {
    const num = arguments[i]
    console.log(num)
    sum = sum + num
  }
  // return num1 + num2;
  return sum
}

let result3 = addNumber3(3, 5, 8, 15, 29)
console.log(result3) // 60
```


2.5 HOW TO ORGANIZE CODE INSIDE A FUNCTION

JAVASCRIPT

```javascript
function addNumber4(num1, num2) {
  let sum = 0
  for (let i = 0; i < arguments.length; i++) {
    const num = arguments[i]
    console.log(num)
    sum = sum + num
  }
  logInfo('Good Morning') /* readability issue */

  // declare function before calling
  function logInfo(msg) {
    console.log(msg)
  }

  // logInfo("Good Morning");

  let double = sum * 2 /* assign variable before using */
  /* if variable use in multiple places declare on top */
  return sum
}

let result4 = addNumber4(3, 5, 8, 15, 29)
console.log(result4)
```


2.6 WHAT IS EVENT, DIFFERENT TYPES OF EVENT IN WEB

Resource- 1 Resource- 2 Resource- 3

2.7 ADD EVENT HANDLER DIRECTLY ON AN ELEMENT

HTML

```html
<body>
  <h1 onclick="console.log('Hello')">Exploring Event</h1>
  <!-- <button onclick="alert(47)">Click Me</button> -->
  <button onclick="handleClick()">Button-1</button>
  <button onmousemove="mouseMove()">Button-2</button>
```

```
  <button onmouseout="mouseOut()">Button-3</button>

  <script src="script.js"></script>
</body>
```

JAVASCRIPT

```
//script.js
function handleClick() {
  console.log('someone clicked me')
}

function mouseMove() {
  console.log('someone moved over me')
}

function mouseOut() {
  console.log('someone moved out')
}
```

2.8 ADD EVENT LISTENER USING JAVASCRIPT

HTML

```
<body>
  <button id="primary-btn">Primary Button</button>
  <button id="secondary-btn">Secondary Button</button>
  <button id="third-btn">Third Button</button>
  <button id="fourth-btn">Fourth Button</button>
  <script src="script.js"></script>
</body>
```

JAVASCRIPT

```
// script.js
// applied function method-1, direct function
let secondaryBtn = document.getElementById('secondary-btn')
```

```
secondaryBtn.onclick = function () {
  console.log('You just clicked me')
}

// applied function method-2, create function first
function handleClick2() {
  document.body.style.backgroundColor = 'cyan'
}

let primaryBtn = document.getElementById('primary-btn')
primaryBtn.onclick = handleClick2

// applied function method-3, event listener
function handleClick3() {
  document.body.style.backgroundColor = 'tomato'
}

let thirdBtn = document.getElementById('third-btn')

thirdBtn.addEventListener('click', handleClick3)

// applied function method-4, event listener
function handleClick4() {
  document.body.style.backgroundColor = 'orange'
}
// const fourthBtn = document.getElementById('fourth-btn').addEventListener('clic
const fourthBtn = document
  .getElementById('fourth-btn')
  .addEventListener('click', function () {
    document.body.style.color = 'tomato'
  })

// function() is called anonymous function
```


2.9 EVENT BUBBLE WITH EXAMPLE AND STOP PROPAGATING EVENT BUBBLE

HTML

```html
<body>
  <!-- EVENT BUBBLE WITH EXAMPLE -->
  <!-- div#container>ul#myList>li.item*5>lorem3 -->
  <div id="container">
    <h1>My List</h1>
    <ul id="myList">
      <li id="first" class="item">Lorem, ipsum dolor.</li>
      <li class="item">Obcaecati, minima mollitia.</li>
      <li class="item">Distinctio, quasi autem?</li>
      <li class="item">Sint, placeat labore.</li>
      <li class="item">Quibusdam, ipsam deserunt!</li>
    </ul>
  </div>
  <script src="script.js"></script>
</body>
```

JAVASCRIPT

```javascript
// script.js
document.getElementById('first').addEventListener('click', function (event) {
  console.log('First Item Clicked')
  // STOP PROPAGATING EVENT BUBBLE
  // event.stopPropagation();
})
document.getElementById('myList').addEventListener('click', function (event) {
  console.log('ul Clicked')
  event.stopPropagation()
})

document
  .getElementById('container')
  .addEventListener('click', function (event) {
    console.log('Container Clicked')
  })

// stopPropagation
// stopImmediatePropagation
```

[Additional Resources]

2.10 EVENT DELEGATE AND PURPOSE OF EVENT BUBBLE

HTML

```html
<body>
  <div id="container">
    <h1>My List</h1>
    <ul id="myList">
      <li id="first" class="item">Lorem, ipsum dolor.</li>
      <li class="item">Obcaecati, minima mollitia.</li>
      <li class="item">Distinctio, quasi autem?</li>
      <li class="item">Sint, placeat labore.</li>
      <li class="item">Quibusdam, ipsam deserunt!</li>
    </ul>

    <button id="addNew">Add New</button>
  </div>
  <script src="script.js"></script>
</body>
```


JAVASCRIPT

```javascript
// script.js
// EVENT DELEGATE AND PURPOSE OF EVENT BUBBLE
// in chrome:
// document.getElementsByClassName('item');

/*
        let items = document.getElementsByClassName('item');
        for (let i = 0; i < items.length; i++) {
            let item = items[i];
            // console.log("item");

            item.addEventListener('click', function (event) {
                // console.log(this);
                // console.log(this, event.target);
                //access event.target- target the event where clicked
```

```
        // console.log(event.target.innerText);
        // console.log(event.target.parentNode);
        //will give parent
        event.target.parentNode.removeChild(event.target);
    })
    }
    */

document.getElementById('addNew').addEventListener('click', function (event) {
  let itemsParent = document.getElementById('myList')
  let newItem = document.createElement('li')
  newItem.innerText = 'Brand New Item'
  itemsParent.appendChild(newItem)
})

document.getElementById('myList').addEventListener('click', function (event) {
  // console.log(this, event.target);

  event.target.parentNode.removeChild(event.target)
})
```


3 INTEGRATE JAVASCRIPT BONUS CONTENT

3.1 GLOBAL VS LOCAL VARIABLE IIFE FUNCTION EXPRESSION VS DECLARATION

JAVASCRIPT

```
function addToDo() {
  // Local variable, inside function
  const newTaskElement = document.createElement('li')
}

// Global Variable
var name = 'Rashed'

function addUser() {
  var localUser = 'Andy'
```

```
  //localUser = "Andy"; global leaking, wrong
  console.log(name)
}

addUser()

console.log(localUser) // will give an error: localUser is not defined

// . IIFE (Immediately Invoked Function Expression)
;(function () {
  var localUser = 'Andy'
  console.log(localUser)
})()

// Function Declaration vs Statement
// Example: Function Expression
alert(foo()) // ERROR! foo wasn't loaded yet
var foo = function () {
  return 5
}

// Example: Function Declaration
alert(foo()) // Alerts 5. Declarations are loaded before any code can run.
function foo() {
  return 5
}
```


[Additional Resources- 1]

[Additional Resources- 2]

3.2 FOUR SITUATION WHEN YOU SHOULD CREATE A FUNCTION

1. Handle Specific task or situation
2. Reduce Code Repetition (DRY- Do Not Repeat Yourself)
3. Specific unit or atomic task
4. Organize larger code

3.3 WHEN AND HOW TO USE ARGUMENTS IN A FUNCTION

JAVASCRIPT

```
//argument is array like object
function getFullName(firstName, lastName) {
  // console.log(arguments);
  // let fullName = firstName + ' ' + lastName;
  let fullName = ''
  for (let i = 0; i < arguments.length; i++) {
    const namePart = arguments[i]
    fullName = fullName + ' ' + namePart
  }
  return fullName
}

let name = getFullName('Hanif', 'Songkhet', 'Poribohon', 'Hasib')
console.log(name)

// convert arguments to a real Array:

let args = Array.from(arguments)
// or
let args = [...arguments]
```


Additional Resources

3.4 WHEN AND HOW TO USE JAVASCRIPT CALLBACK FUNCTION

JAVASCRIPT

```
function welcomeGuest(name, greetHandler) {
  // console.log(name);
  greetHandler(name)
}

function greetUserMorning(name) {
  console.log('Good Morning', name)
}
```

```
function greetUserEvening(name) {
  console.log('Good Evening', name)
}

function greetUserAfternoon(name) {
  console.log('Good Afternoon', name)
}

const actorName = 'Tom Hanks'
welcomeGuest(actorName, greetUserMorning)
welcomeGuest(actorName, greetUserEvening)
welcomeGuest(actorName, greetUserAfternoon)

welcomeGuest('Kate Winslet', greetUserMorning)

welcomeGuest('Shaib Khan', function (name) {
  console.log('Special Welcome to', name)
})

function handleClick() {
  console.log('Someone Clicked Me')
}
document.getElementById('demo').addEventListener('click', handleClick)
```