30 Days Of JavaScript: Functions

LinkedIn Follow @asabeneh

Author: Asabeneh Yetayeh

January, 2020

&lt;&lt; Day 6 | Day 8 &gt;&gt;

Thirty Days Of JavaScript

📔 Day 7

Functions

So far we have seen many builtin JavaScript functions. In this section, we will focus on custom functions. What is a function? Before we start making functions, lets understand what function is and why we need function?

A function is a reusable block of code or programming statements designed to perform a certain task. A function is declared by a function key word followed by a name, followed by parentheses (). A parentheses can take a parameter. If a function take a parameter it will be called with argument. A function can also take a default parameter. To store a data to a function, a function has to return certain data types. To get the value we call or invoke a function. Function makes code:

- clean and easy to read
- reusable
- easy to test

A function can be declared or created in couple of ways:

- *Declaration function*
- *Expression function*
- *Anonymous function*
- *Arrow function*

Function Declaration

Let us see how to declare a function and how to call a function.

```
//declaring a function without a parameter
function functionName() {
  // code goes here
}
functionName() // calling function by its name and with parentheses
```

Function without a parameter and return

Function can be declared without a parameter.

Example:

```
// function without parameter,  a function which make a number square
function square() {
  let num = 2
  let sq = num * num
  console.log(sq)
}

square() // 4

// function without parameter
function addTwoNumbers() {
  let numOne = 10
  let numTwo = 20
  let sum = numOne + numTwo

  console.log(sum)
}

addTwoNumbers() // a function has to be called by its name to be executed
```

```
    function printFullName (){
        let firstName = 'Asabeneh'
        let lastName = 'Yetayeh'
        let space = ' '
        let fullName = firstName + space + lastName
        console.log(fullName)
  }

  printFullName() // calling a function
```

Function returning value

Function can also return values, if a function does not return values the value of the function is undefined. Let us write the above functions with return. From now on, we return value to a function instead of printing it.

```
  function printFullName (){
        let firstName = 'Asabeneh'
        let lastName = 'Yetayeh'
        let space = ' '
        let fullName = firstName + space + lastName
        return fullName
  }
  console.log(printFullName())
```

```
    function addTwoNumbers() {
        let numOne = 2
        let numTwo = 3
        let total = numOne + numTwo
        return total

    }

  console.log(addTwoNumbers())
```

Function with a parameter

In a function we can pass different data types(number, string, boolean, object, function) as a parameter.

```
// function with one parameter
function functionName(parm1) {
  //code goes her
}
functionName(parm1) // during calling or invoking one argument needed

function areaOfCircle(r) {
  let area = Math.PI * r * r
  return area
}

console.log(areaOfCircle(10)) // should be called with one argument

function square(number) {
  return number * number
}

console.log(square(10))
```

Function with two parameters

```
// function with two parameters
function functionName(parm1, parm2) {
  //code goes her
}
functionName(parm1, parm2) // during calling or invoking two arguments needed
// Function without parameter doesn't take input, so lets make a function with parameters
function sumTwoNumbers(numOne, numTwo) {
  let sum = numOne + numTwo
  console.log(sum)
}
sumTwoNumbers(10, 20) // calling functions
// If a function doesn't return it doesn't store data, so it should return

function sumTwoNumbers(numOne, numTwo) {
  let sum = numOne + numTwo
  return sum
}

console.log(sumTwoNumbers(10, 20))
function printFullName(firstName, lastName) {
  return `${firstName} ${lastName}`
}
console.log(printFullName('Asabeneh', 'Yetayeh'))
```

Function with many parameters

```
// function with multiple parameters
function functionName(parm1, parm2, parm3,...){
  //code goes here
}
functionName(parm1,parm2,parm3,...) // during calling or invoking three arguments needed


// this function takes array as a parameter and sum up the numbers in the array
function sumArrayValues(arr) {
  let sum = 0;
  for (let i = 0; i < arr.length; i++) {
    sum = sum + arr[i];
  }
  return sum;
}
const numbers = [1, 2, 3, 4, 5];
    //calling a function
console.log(sumArrayValues(numbers));


    const areaOfCircle = (radius) => {
      let area = Math.PI * radius * radius;
      return area;
    }
console.log(areaOfCircle(10))
```

Function with unlimited number of parameters

Sometimes we do not know how many arguments the user going to pass. Therefore, we should know how to write a function which can take unlimited number of arguments. The way we do it has a significant difference between a function declaration(regular function) and arrow function. Let us see examples both in function declaration and arrow function.

Unlimited number of parameters in regular function

A function declaration provides a function scoped arguments array like object. Any thing we passed as argument in the function can be accessed from arguments object inside the functions. Let us see an example

```
// Let us access the arguments object

function sumAllNums() {
 console.log(arguments)
}
```

```
sumAllNums(1, 2, 3, 4)
// Arguments(4) [1, 2, 3, 4, callee: ƒ, Symbol(Symbol.iterator): ƒ]
```

```
// function declaration

function sumAllNums() {
  let sum = 0
  for (let i = 0; i < arguments.length; i++) {
    sum += arguments[i]
  }
  return sum
}

console.log(sumAllNums(1, 2, 3, 4)) // 10
console.log(sumAllNums(10, 20, 13, 40, 10))  // 93
console.log(sumAllNums(15, 20, 30, 25, 10, 33, 40))  // 173
```

Unlimited number of parameters in arrow function

Arrow function does not have the function scoped arguments object. To implement a function which takes unlimited number of arguments in an arrow function we use spread operator followed by any parameter name. Any thing we passed as argument in the function can be accessed as array in the arrow function. Let us see an example

```
// Let us access the arguments object

const sumAllNums = (...args) => {
 // console.log(arguments), arguments object not found in arrow function
 // instead we use a parameter followed by spread operator (...)
 console.log(args)
}

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

```
// function declaration

const sumAllNums = (...args) => {
  let sum = 0
  for (const element of args) {
    sum += element
  }
  return sum
}
```

```
console.log(sumAllNums(1, 2, 3, 4)) // 10
console.log(sumAllNums(10, 20, 13, 40, 10))  // 93
console.log(sumAllNums(15, 20, 30, 25, 10, 33, 40))  // 173
```

Anonymous Function

Anonymous function or without name

```
const anonymousFun = function() {
  console.log(
    'I am an anonymous function and my value is stored in anonymousFun'
  )
}
```

Expression Function

Expression functions are anonymous functions. After we create a function without a name and we assign it to a variable. To return a value from the function we should call the variable. Look at the example below.

```
// Function expression
const square = function(n) {
  return n * n
}

console.log(square(2)) // -> 4
```

Self Invoking Functions

Self invoking functions are anonymous functions which do not need to be called to return a value.

```
(function(n) {
  console.log(n * n)
})(2) // 4, but instead of just printing if we want to return and store the data, we do as sho

let squaredNum = (function(n) {
  return n * n
})(10)

console.log(squaredNum)
```

Arrow Function

Arrow function is an alternative to write a function, however function declaration and arrow function have some minor differences.

Arrow function uses arrow instead of the keyword *function* to declare a function. Let us see both function declaration and arrow function.

```
// This is how we write normal or declaration function
// Let us change this declaration function to an arrow function
function square(n) {
  return n * n
}

console.log(square(2)) // 4

const square = n => {
  return n * n
}

console.log(square(2))  // -> 4

// if we have only one line in the code block, it can be written as follows, explicit return
const square = n => n * n  // -> 4
```

```
const changeToUpperCase = arr => {
  const newArr = []
  for (const element of arr) {
    newArr.push(element.toUpperCase())
  }
  return newArr
}

const countries = ['Finland', 'Sweden', 'Norway', 'Denmark', 'Iceland']
console.log(changeToUpperCase(countries))

// ["FINLAND", "SWEDEN", "NORWAY", "DENMARK", "ICELAND"]
```

```
const printFullName = (firstName, lastName) => {
  return `${firstName} ${lastName}`
}

console.log(printFullName('Asabeneh', 'Yetayeh'))
```

The above function has only the return statement, therefore, we can explicitly return it as follows.

```
const printFullName = (firstName, lastName) => `${firstName} ${lastName}`
```

```
console.log(printFullName('Asabeneh', 'Yetayeh'))
```

Function with default parameters

Sometimes we pass default values to parameters, when we invoke the function if we do not pass an argument the default value will be used. Both function declaration and arrow function can have a default value or values.

```
// syntax
// Declaring a function
function functionName(param = value) {
  //codes
}

// Calling function
functionName()
functionName(arg)
```

Example:

```
function greetings(name = 'Peter') {
  let message = `${name}, welcome to 30 Days Of JavaScript!`
  return message
}

console.log(greetings())
console.log(greetings('Asabeneh'))


function generateFullName(firstName = 'Asabeneh', lastName = 'Yetayeh') {
  let space = ' '
  let fullName = firstName + space + lastName
  return fullName
}

console.log(generateFullName())
console.log(generateFullName('David', 'Smith'))


function calculateAge(birthYear, currentYear = 2019) {
  let age = currentYear - birthYear
  return age
}

console.log('Age: ', calculateAge(1819))
```

```
function weightOfObject(mass, gravity = 9.81) {
  let weight = mass * gravity + ' N' // the value has to be changed to string first
  return weight
}

console.log('Weight of an object in Newton: ', weightOfObject(100)) // 9.81 gravity at the sur
console.log('Weight of an object in Newton: ', weightOfObject(100, 1.62)) // gravity at surfac
```

Let us see how we write the above functions with arrow functions

```
// syntax
// Declaring a function
const functionName = (param = value) => {
  //codes
}

// Calling function
functionName()
functionName(arg)
```

Example:

```
const greetings = (name = 'Peter') => {
  let message = name + ', welcome to 30 Days Of JavaScript!'
  return message
}

console.log(greetings())
console.log(greetings('Asabeneh'))
```

```
const generateFullName = (firstName = 'Asabeneh', lastName = 'Yetayeh') => {
  let space = ' '
  let fullName = firstName + space + lastName
  return fullName
}

console.log(generateFullName())
console.log(generateFullName('David', 'Smith'))
```

```
const calculateAge = (birthYear, currentYear = 2019) => currentYear - birthYear
console.log('Age: ', calculateAge(1819))
```

```
const weightOfObject = (mass, gravity = 9.81) => mass * gravity + ' N'

console.log('Weight of an object in Newton: ', weightOfObject(100)) // 9.81 gravity at the sur
console.log('Weight of an object in Newton: ', weightOfObject(100, 1.62)) // gravity at surfac
```

Function declaration versus Arrow function

It Will be covered in other section.

🌕 You are a rising star, now you knew function . Now, you are super charged with the power of functions. You have just completed day 7 challenges and you are 7 steps a head in to your way to greatness. Now do some exercises for your brain and for your muscle.

💻 Exercises

Exercises: Level 1

1. Declare a function *fullName* and it print out your full name.
2. Declare a function *fullName* and now it takes firstName, lastName as a parameter and it returns your full - name.
3. Declare a function *addNumbers* and it takes two two parameters and it returns sum.
4. An area of a rectangle is calculated as follows: *area = length x width*. Write a function which calculates *areaOfRectangle*.
5. A perimeter of a rectangle is calculated as follows: *perimeter= 2x(length + width)*. Write a function which calculates *perimeterOfRectangle*.
6. A volume of a rectangular prism is calculated as follows: *volume = length x width x height*. Write a function which calculates *volumeOfRectPrism*.
7. Area of a circle is calculated as follows: *area = π x r x r*. Write a function which calculates *areaOfCircle*
8. Circumference of a circle is calculated as follows: *circumference = 2πr*. Write a function which calculates *circumOfCircle*
9. Density of a substance is calculated as follows:*density= mass/volume*. Write a function which calculates *density*.
10. Speed is calculated by dividing the total distance covered by a moving object divided by the total amount of time taken. Write a function which calculates a speed of a moving object, *speed*.

11. Weight of a substance is calculated as follows: *weight = mass x gravity*. Write a function which calculates *weight*.

12. Temperature in oC can be converted to oF using this formula: *oF = (oC x 9/5) + 32*. Write a function which convert oC to oF *convertCelsiusToFahrenheit*.

13. Body mass index(BMI) is calculated as follows: *bmi = weight in Kg / (height x height) in m2*. Write a function which calculates *bmi*. BMI is used to broadly define different weight groups in adults 20 years old or older.Check if a person is *underweight, normal, overweight* or *obese* based the information given below.

    - The same groups apply to both men and women.
    - *Underweight*: BMI is less than 18.5
    - *Normal weight*: BMI is 18.5 to 24.9
    - *Overweight*: BMI is 25 to 29.9
    - *Obese*: BMI is 30 or more

14. Write a function called *checkSeason*, it takes a month parameter and returns the season:Autumn, Winter, Spring or Summer.

15. Math.max returns its largest argument. Write a function findMax that takes three arguments and returns their maximum with out using Math.max method.

    ```
    console.log(findMax(0, 10, 5))
    10
    console.log(findMax(0, -10, -2))
    0
    ```

Exercises: Level 2

1. Linear equation is calculated as follows: *ax + by + c = 0*. Write a function which calculates value of a linear equation, *solveLinEquation*.

2. Quadratic equation is calculated as follows: *ax2 + bx + c = 0*. Write a function which calculates value or values of a quadratic equation, *solveQuadEquation*.

    ```
    console.log(solveQuadratic()) // {0}
    console.log(solveQuadratic(1, 4, 4)) // {-2}
    console.log(solveQuadratic(1, -1, -2)) // {2, -1}
    console.log(solveQuadratic(1, 7, 12)) // {-3, -4}
    console.log(solveQuadratic(1, 0, -4)) //{2, -2}
    console.log(solveQuadratic(1, -1, 0)) //{1, 0}
    ```

3. Declare a function name *printArray*. It takes array as a parameter and it prints out each value of the array.

4. Write a function name *showDateTime* which shows time in this format: 08/01/2020 04:08 using the Date object.

```
showDateTime()
08/01/2020 04:08
```

5. Declare a function name *swapValues*. This function swaps value of x to y.

```
swapValues(3, 4) // x => 4, y=>3
swapValues(4, 5) // x = 5, y = 4
```

6. Declare a function name *reverseArray*. It takes array as a parameter and it returns the reverse of the array (don't use method).

```
console.log(reverseArray([1, 2, 3, 4, 5]))
//[5, 4, 3, 2, 1]
console.log(reverseArray(['A', 'B', 'C']))
//['C', 'B', 'A']
```

7. Declare a function name *capitalizeArray*. It takes array as a parameter and it returns the - capitalizedarray.
8. Declare a function name *addItem*. It takes an item parameter and it returns an array after adding the item
9. Declare a function name *removeItem*. It takes an index parameter and it returns an array after removing an item
10. Declare a function name *sumOfNumbers*. It takes a number parameter and it adds all the numbers in that range.
11. Declare a function name *sumOfOdds*. It takes a number parameter and it adds all the odd numbers in that - range.
12. Declare a function name *sumOfEven*. It takes a number parameter and it adds all the even numbers in that - range.
13. Declare a function name evensAndOdds . It takes a positive integer as parameter and it counts number of evens and odds in the number.

```
evensAndOdds(100);
The number of odds are 50.
The number of evens are 51.
```

14. Write a function which takes any number of arguments and return the sum of the arguments

    ```
    sum(1, 2, 3) // -> 6
    sum(1, 2, 3, 4) // -> 10
    ```

15. Writ a function which generates a *randomUserIp*.
16. Write a function which generates a *randomMacAddress*
17. Declare a function name *randomHexaNumberGenerator*. When this function is called it generates a random hexadecimal number. The function return the hexadecimal number.

    ```
    console.log(randomHexaNumberGenerator());
    '#ee33df'
    ```

18. Declare a function name *userIdGenerator*. When this function is called it generates seven character id. The function return the id.

    ```
    console.log(userIdGenerator());
    41XTDbE
    ```

Exercises: Level 3

1. Modify the *userIdGenerator* function. Declare a function name *userIdGeneratedByUser*. It doesn't take any parameter but it takes two inputs using prompt(). One of the input is the number of characters and the second input is the number of ids which are supposed to be generated.

   ```
   userIdGeneratedByUser()
   'kcsy2
   SMFYb
   bWmeq
   ZXOYh
   2Rgxf
   '
   userIdGeneratedByUser()
   '1GCSgPLMaBAVQZ26
   YD7eFwNQKNs7qXaT
   ycArC5yrRupyG00S
   UbGxOFI7UXSWAyKN
   dIV0SSUTgAdKwStr
   '
   ```

2. Write a function name *rgbColorGenerator* and it generates rgb colors.

```
rgbColorGenerator()
rgb(125,244,255)
```

3. Write a function ***arrayOfHexaColors*** which return any number of hexadecimal colors in an array.
4. Write a function ***arrayOfRgbColors*** which return any number of RGB colors in an array.
5. Write a function ***convertHexaToRgb*** which converts hexa color to rgb and it returns an rgb color.
6. Write a function ***convertRgbToHexa*** which converts rgb to hexa color and it returns an hexa color.
7. Write a function ***generateColors*** which can generate any number of hexa or rgb colors.

```
console.log(generateColors('hexa', 3)) // ['#a3e12f', '#03ed55', '#eb3d2b']
console.log(generateColors('hexa', 1)) // '#b334ef'
console.log(generateColors('rgb', 3)) // ['rgb(5, 55, 175)', 'rgb(50, 105, 100)', 'rgb(15,
console.log(generateColors('rgb', 1)) // 'rgb(33,79, 176)'
```

8. Call your function *shuffleArray*, it takes an array as a parameter and it returns a shuffled array
9. Call your function *factorial*, it takes a whole number as a parameter and it return a factorial of the number
10. Call your function *isEmpty*, it takes a parameter and it checks if it is empty or not
11. Call your function *sum*, it takes any number of arguments and it returns the sum.
12. Write a function called *sumOfArrayItems*, it takes an array parameter and return the sum of all the items. Check if all the array items are number types. If not give return reasonable feedback.
13. Write a function called *average*, it takes an array parameter and returns the average of the items. Check if all the array items are number types. If not give return reasonable feedback.
14. Write a function called *modifyArray* takes array as parameter and modifies the fifth item of the array and return the array. If the array length is less than five it return 'item not found'.

```
console.log(modifyArray(['Avocado', 'Tomato', 'Potato','Mango', 'Lemon','Carrot']);
```

```
['Avocado', 'Tomato', 'Potato','Mango', 'LEMON', 'Carrot']
```

```
console.log(modifyArray(['Google', 'Facebook','Apple', 'Amazon','Microsoft',  'IBM']);
['Google', 'Facebook','Apple', 'Amazon','MICROSOFT',  'IBM']
```

```
console.log(modifyArray(['Google', 'Facebook','Apple', 'Amazon']);
```

```
  'Not Found'
```

15. Write a function called *isPrime*, which checks if a number is prime number.
16. Write a functions which checks if all items are unique in the array.
17. Write a function which checks if all the items of the array are the same data type.
18. JavaScript variable name does not support special characters or symbols except $ or _. Write a function **isValidVariable** which check if a variable is valid or invalid variable.
19. Write a function which returns array of seven random numbers in a range of 0-9. All the numbers must be unique.

```
sevenRandomNumbers()
[(1, 4, 5, 7, 9, 8, 0)]
```

20. Write a function called reverseCountries, it takes countries array and first it copy the array and returns the reverse of the original array

🎉 CONGRATULATIONS ! 🎉

<< Day 6 | Day 8 >>