6.6 Functions

Introduction to functions

A ***function*** is a named group of statements. JavaScript functions are declared with the `function` keyword followed by the function name and parameter list in parentheses `()`. A ***parameter*** is a variable that supplies the function with input. The function's statements are enclosed in braces `{}`.

Invoking a function's name, known as a ***function call***, causes the function's statements to execute. An ***argument*** is a value provided to a function's parameter during a function call.

Construct 6.6.1: Function declaration.

```
function functionName(parameter1, parameter2,
...) {
   // Statements to execute when function is
called

}
```

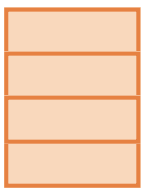
Feedback?

PARTICIPATION ACTIVITY 6.6.1: Declaring and calling a function.

Start 2x speed

```
function displaySum(x, y, z) {
   let sum = x + y + z;
   console.log(sum);
}

console.log("About to call function");
displaySum(2, 5, 3);
console.log("Returned from function");
```



About to call function
10
Returned from function

Captions

1. A function named displaySum is declared with three parameters: x, y, and z.

2. displaySum() is called with arguments 2, 5, and 3, which are assigned to parameters x, y, and z.
3. The variable sum is assigned the sum of x, y, and z, which is 10.
4. sum is output to the console. No more code exist in the function, so the function is finished executing.

Feedback?

Good practice is to use function names that contain a verb and noun. Ex: `display` is a vague function name, but `displayAverage` is better because `displayAverage` indicates what is being displayed.

Good practice is to use camel case for JavaScript function names, where the name starts with a lowercase letter and subsequent words begin with a capital letter.

PARTICIPATION ACTIVITY 6.6.2: Declaring and calling functions.

1) Which function call displays the numbers 5, 4, 3, 2, 1.

```
function countDown(firstNum) {
   for (let count = firstNum; count > 0; count--) {

console.log(count);
   }
}
```

- `countDown();`
- `countDown(5);`
- `countDown(5, 4, 3, 2, 1);`

Correct

The argument 5 is assigned to the parameter `firstNum`. The for loop displays the numbers from 5 down to 1.

2) Choose a better name for the function `test`.

```
function test(x, y) {
   if (x > y) {
      console.log(x);
   }
   else {
      console.log(y);
   }
}
```

- ○ `Largest`
- ○ `display_largest`
- ◉ `displayLargest`

Correct

The function displays the parameter that is largest. Using camel case to name functions with a verb and noun is good practice.

3) What is output to the console?

```
function
sayHello(name,
greeting) {

console.log(greeting
+ ", " + name +
"!");
}

sayHello("Maria");
```

- ○ Hello, Maria!
- ○ Hello, undefined!
- ◉ undefined, Maria!

Correct

The call to `sayHello()` has only one argument. The parameter `name` is assigned the argument "Maria", but `greeting` is not assigned a value. A variable that is used without being assigned a value is undefined.

4) The function below uses a default parameter value "Hello" that is assigned when the greeting is not supplied in the function call. What is output to the console?

```
sayHello("Sam");
sayHello("Juan",
"Hola");

function
sayHello(name,
greeting = "Hello")
{

console.log(greeting
+ ", " + name);
}
```

- Hello, Sam
  Hello, Juan
- Hello, Sam
  Hola, Juan
- undefined

Correct

The `sayHello("Sam")` call does not specify a second argument, so `greeting = "Hello"` assigns the "Hello" to `greeting`. The `sayHello("Juan", "Hola")` call specifies a second argument "Hola", so `greeting` is assigned "Hola" instead of "Hello". A parameter like `greeting`, which may not be assigned a value, is called an "optional" parameter.

Feedback?

PARTICIPATION ACTIVITY 6.6.3: Function practice.

The code below produces a 5 x 10 box of question marks. Convert the code into a function called `drawBox()` that has three parameters:

1. `numRows` - The number of rows for the box.
2. `numCols` - The number of columns for the box.
3. `boxChar` - The character to use to create the box. If no argument is supplied, use "X".

Ex: `drawBox(5, 4, "!")` and `drawBox(2, 6)` should display the boxes pictured below.

```
!!!!
!!!!
!!!!
!!!!
!!!!
XXXXXX
XXXXXX
```

```
// Convert into a drawBox function
for (let r = 0; r < 5; r++) {
   let line = "";
   for (let c = 0; c < 10; c++) {
      line += "?";
   }
   console.log(line);
}

```

Run JavaScript Reset code

Your console output

```
??????????
??????????
??????????
??????????
??????????
```

▼ View solution

Explain

```
--- START FILE: JavaScript ---


drawBox(5, 4, "!");
drawBox(2, 6);

function drawBox(numRows, numCols, boxChar = "X") {
   for (let r = 0; r < numRows; r++) {
      let line = "";
      for (let c = 0; c < numCols; c++) {
```

```
            line += boxChar;
        }
        console.log(line);
    }
}


--- END FILE: JavaScript ---
```

Feedback?

Returning a value

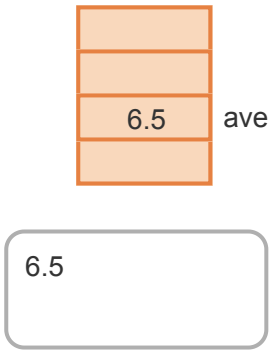
A function may return a single value using a ***return*** statement. A function that is missing a return statement returns `undefined`.

PARTICIPATION ACTIVITY 6.6.4: Function that returns a value.

Start 2x speed

```
function findAverage(num1, num2) {
   return (num1 + num2) / 2;
}

let ave = findAverage(6, 7);
console.log(ave);
```



Captions

1. A function named findAverage is declared with two parameters: num1 and num2.
2. findAverage() is called with arguments 6 and 7, which are assigned to parameters num1 and num2.
3. The return statement returns the average of num1 and num2, which is 6.5.

Feedback?

PARTICIPATION ACTIVITY 6.6.5: Functions that return values.

1) What is output to the console?

```
console.log(findSmallest(5, 2));

function findSmallest(x, y)
{
   if (x < y) {
      return x;
   }
   else {
      return y;
   }
}
```

- (●) 2
- ( ) 5
- ( ) undefined

Correct

`findSmallest(5, 2)` returns the smaller of the two values, which is 2.

2) What is the correct way to call `factorial()` and output the factorial of 5?

```
function factorial(num) {
   let result = 1;
   for (let count = 1;
count <= num; count++) {
      result *= count;
   }
   return result;
}
```

- ( )
```
factorial(5);
console.log(result);
```
- ( )
```
let answer =
factorial();
console.log(answer);
```
- (●)
```
let answer =
factorial(5);
console.log(answer);
```

Correct

`factorial(5)` finds 5! = 1 × 2 × 3 × 4 × 5 = 120, which is assigned to `answer` and output with `console.log()`.

3) What is output to the console?

```
function factorial(num) {
   let result = 1;
   for (let count = 1; count <= num; count++) {
      result *= count;
   }
   return result;
}

let answer = factorial(8 - factorial(3));
console.log(answer);
```

- (•) 2
- ( ) 6
- ( ) 8

Correct

`factorial(3)` returns 1 × 2 × 3 = 6. The 6 is subtracted from 8, and the call `factorial(2)` returns 1 × 2 = 2.

4) What is output to the console?

```
console.log(sayHello("Sam"));

function sayHello(name) {
   console.log("Hello, " + name + "!");
}
```

- ( ) Hello, Sam!
- (•) Hello, Sam!
undefined
- ( ) undefined

Correct

`sayHello()` outputs "Hello, Sam!", then `sayHello()`'s return value is displayed. `sayHello()` returns `undefined` because `sayHello()` does not have a `return` statement.

Feedback?

Function expressions and anonymous functions

JavaScript functions may be assigned to a variable with a function expression. A ***function expression*** is identical to a function declaration, except the function name may be omitted. A function without a name is called an ***anonymous function***. Anonymous functions are often used with arrays and event handlers, discussed elsewhere in this material.

Figure 6.6.1: Assigning a function expression to a variable.

```
// Function name is omitted
let displaySum = function(x, y, z)
{
   console.log(x + y + z);
}

// Function call
displaySum(2, 5, 3);
```

Feedback?

Unlike functions declared with a function declaration, a variable assigned with a function expression cannot be used until after the variable is assigned. Using a variable before the variable is assigned with a function expression causes an exception.

PARTICIPATION ACTIVITY 6.6.6: Using a function expression before assignment.

Start 2x speed

```
console.log(findLargest(5, 3));

function findLargest(x, y) {
   let largest;
   if (x > y) {
      largest = x;
   }
   else {
      largest = y;
   }

   return largest;
}

displaySum(2, 5, 3);

let displaySum = function(x, y, z) {
   console.log(x + y + z);
}
```

5
Uncaught ReferenceError: cannot access 'displaySum' before initialization

Captions

1. findLargest() may be called before the findLargest() function declaration.
2. Since x > y, findLargest() returns 5, and 5 is output to the console.
3. Calling displaySum() before displaySum is assigned with a function expression produces an exception.

Feedback?

PARTICIPATION ACTIVITY 6.6.7: Function expressions.

1) The variable `result` is assigned 4.

```
let square =
function(num) {
   return num *
num;
}
let result =
square(2);
```

- (●) True
- ( ) False

Correct

`square` is assigned a function expression that squares the function's argument.

2) The variable `result` is assigned 9.

```
let result =
square(3);
let square =
function(num) {
   return num *
num;
}
```

- ( ) True
- (●) False

Correct

`square` is not assigned a function expression until after `square()` is called. The call to `square(3)` causes an exception.

3) The variable `result` is assigned 9.

```
let square =
function(num) {
   return num *
num;
}
let result =
square;
```

- ( ) True
- (●) False

Correct

`result` is assigned `square`, so `result` is assigned a function expression, not the number 9. Calling `result(3)` returns 9.

Feedback?

Arrow functions

An ***arrow function*** is an anonymous function that uses an arrow => to create a compact function. An arrow function's parameters are listed to the left of the arrow. The right side of the arrow may be a single expression or multiple statements in braces.

Construct 6.6.2: Arrow function declaration that returns a single expression.

```
(parameter1, parameter2, ...) =>
expression
```

Feedback?

Construct 6.6.3: Arrow function with multiple statements.

```
(parameter1, parameter2, ...) => {
statements; }
```

Feedback?

PARTICIPATION ACTIVITY 6.6.8: Arrow functions that sum two numbers and square a number.

Start 2x speed

```
                          3 + 6 = 9
let findSum = (a, b) => a + b;

let sum = findSum(3, 6);
console.log(sum);
                     5 * 5 = 25
let square = x => x * x;

console.log(square(5));
```

```
9
25
```

Captions

1. An arrow function may be assigned to a variable, just like a function expression.
2. The function parameters are listed in parenthesis to the left of the arrow =>.
3. An expression listed by itself is the value returned by the arrow function.
4. An arrow function is called the same as any other function. The arguments 3 and 6 are assigned to parameters a and b.
5. The arrow function returns the sum of a and b, which is 9.
6. An arrow function with only one parameter does not require parentheses around the one parameter.

Feedback?

PARTICIPATION ACTIVITY 6.6.9: Arrow functions.

1) Complete the arrow function.

```
let max =
=> a > b ? a : b;
```

Check Show answer

Answer

(a, b)

The arrow function has two parameters `a` and `b` and returns the larger of the two values. Ex: `max(2, 3)` returns 3.

2) Complete the arrow function.

```
let countCapitals =
                    => {
   let count = 0;
   for (let i = 0; i <
str.length; i++) {
      let ch =
str.charAt(i);
      if (ch >= 'A' &&
ch <= 'Z') {
         count++;
      }
   }
   return count;
}
```

Check Show answer

Answer

str

Only the parameter `str` is required. Arrow functions that contain more than one statement must be enclosed in {}. The arrow function returns the number of capital letters in the `str` string. Ex: `countCapitals("TesT")` returns 2.

3) Convert `isEven()` into an equivalent arrow function.

```
function isEven(num) {
   return num % 2 ===
0;
}

let isEven = num =>
                  ;
```

Check Show answer

Answer

num % 2 === 0

No `return` keyword is needed to return a value in a single-statement arrow function. The arrow function returns true if the number is even or false if the number is odd. Ex: `isEven(4)` returns true.

Feedback?

CHALLENGE ACTIVITY 6.6.1: Functions.

530096.4000608.qx3zqy7

Start

1

Call the favoriteColor function with argument "blue".

```
1 function favoriteColor(myColor) {
2    console.log("My favorite color is " + myColor + ".");
3 }
4
5 /* Your solution goes here */
6
```

2 ✓ 3 ✓ 4

1 | 2 | 3 | 4

Check Next

View your last submission

Feedback?

Exploring further:

- Functions (MDN)

How was this section? Provide section feedback