MISSION READY

DARE TO
DEVELOP

# Functions

Ewan Zhang

## Journey so far

So far, we’ve looked at:

- Variables
- Comparison and Logical operators
- Conditionals
- Loops
- Data Types
- Objects
- Arrays

## What are Functions?

- A function is a type of object and is a ***block*** of ***organized*** code that is used to perform a task.
  - We “call” or “invoke” a function at any point in our program if we require to run it.
  - A function can be called by other code, by itself, or by a variable that refers to the function.
- Functions are generally used to ***perform some action***.
  - E. g: Sending an email when a user clicks on a button. We define the code for sending the email in our function, but we only call the function when the button is pressed.

# Functions

- A function is a block of organized, reusable lines of code that is used to perform a single, related action.
- A function definition (also called a function declaration, or function statement) consists of the **function keyword**, followed by:
  - The ***name*** of the function.
  - A ***list of parameters*** to the function, enclosed in parentheses and separated by commas.
  - The JavaScript ***statements*** that define the function, enclosed in curly brackets `{...}`.

```
function double(num) {
 return num * 2;
}
```

## Function syntax

- Let’s look at how we write functions in our code

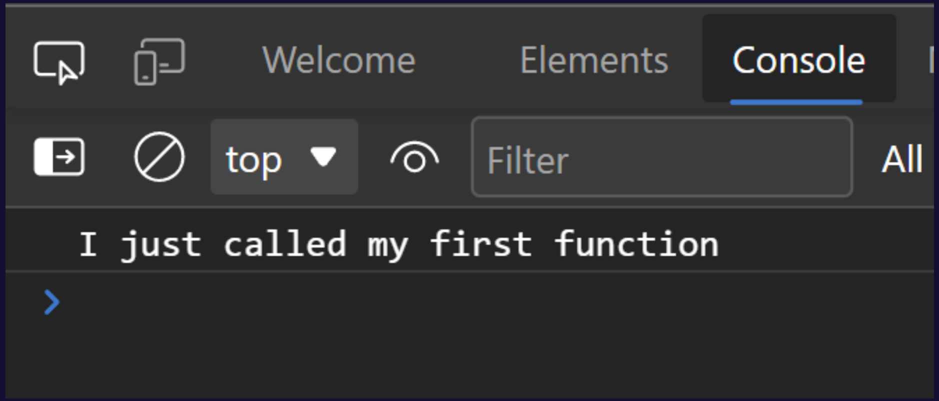
Function declaration

```
function myFirstFunction() {
    console.log('I just called my first function');
}
```

# Function calls

```
function myFirstFunction() {
    console.log('I just called my first function');
}
myFirstFunction(); // This is a function call
```

- To call a function we simply specify its ***name*** and then open and close the parentheses().



```
I just called my first function
>
```

# Function parameters

- When creating a function, you can define a set of ***optional parameters*** that it can take.
    - These parameters will be defined in the parentheses that we’ve previously left empty.

## Function parameters

```
function funcWithParams(num1, num2) {
    console.log(num1 + num2)
}

funcWithParams(3, 4);
```

- In this example, we are allowing our function to take two parameters num1 and num2
  - Separate each parameter with a comma (you can have as many as you'd like!)
- When we call our function, we can define what we want the values of our parameters to be, in this example **num1 = 3** and **num2 = 4**

*Parameters = Placeholder value.*
*Arguments = Actual value.*

## Exercise 1

- Create a function that takes in two strings as parameters and logs them to the console.
- Share a screenshot when you’re done.

## Exercise 2

- Create a function that takes in two strings as parameters, combines the two strings and logs the result to the console.
- Share a screenshot when you’re done.
  - Feel free to use google to help you out.

## return statement

- A return statement is the value that is given back by the function to whoever called it.
- Let’s look at the previous example.

## return statement continued...

- In this example we saw the result of num1 + num2 in the console because we told the function to log that result to the console.

```
function funcWithParams(num1, num2) {
    console.log(num1 + num2)
}
funcWithParams(3, 4);
```

- If we decided to change the console.log to a return statement, now we can store the value of that result to be used later.

```
function funcWithParams(num1, num2) {
    return (num1 + num2);
}
funcWithParams(3, 4);
```

# return statement continued…

- Changing our function to this produces no visual output in the console. That’s because we never logged anything to the console.
    - We can think of the function call as being the value of the return statement.
    - To prove that, we can console.log the function call itself which should show us the result in the console.

```
function funcWithParams(num1, num2) {
    return (num1 + num2);
}

funcWithParams(3, 4);
```

```
function funcWithParams(num1, num2) {
    return (num1 + num2);
}

console.log(funcWithParams(3, 4));
```

## return statement continued...

- We can also store the function call into a variable that we can use later

```
let funcAnswer = funcWithParams(3, 4);

if (funcAnswer === 7) {
    console.log('the answer is right');
} else {
    console.log('the answer is wrong');
}
```

- We can also store the function itself into a variable that we can use later

This is an anonymous function

```
// function expression
const funcWithParamsExpression = function (num1, num2) {
  return num1 + num2;
};
```

# return statement continued…

- Note: the return statement acts like a "break" for that function. So any code after that return statement won't be read.

```
function funcWithParams(num1, num2) {
    return (num1 + num2);
    console.log(num1 + num2);
}
```

- In this example, the console log comes after the return statement and would never be executed.

## Exercise 3

- Create a function that takes in a string and returns ***false*** if the string is ***empty*** and ***true*** if it is not.

  (Hint: an empty string always === false).

- After the result is returned, console.log the result to the browser.

## Functions in Objects

- A JavaScript function can be a value in an object.

```
const person = {
  firstName: "John",
  lastName: "Doe",
  greeting: function (name) {
    console.log(`Hi ${name}, weather is good.`);
  },
};
```

- When functions are stored as object properties, they are called methods.
- You access an object method with the following syntax:

```
objectName.methodName(argument);
person.greeting("Rob");
```

## Exercise 4

1. Write a function called `halfNumber` that will take one argument (a number), divide it by 2, and return the result.
    - Assign the return value of the function to a variable called `halvedNumber`.
    - Print out a log like "Half of 5 is 2.5.“ using the variable

2. Write a function called `timeInSeconds` that takes in an integer *minutes* as a parameter and returns seconds.
   *[Hint: You might need to convert the string output from the prompt to an integer]*
    - Prompt the user to enter minutes.
    - Call the function and alert the output.

MISSION READY
www.missionreadyhq.com

DARE TO
DEVELOP

Thank you | Ewan Zhang