Theory: Functions

7 minutes 0 / 5 problems solved

Skip this topic Start practicing

1542 users solved this topic. Latest completion was about 1 hour ago.

Often we have to repeat the same action in many parts of the script, for example, summing up different data entered by users each time, or displaying pop-ups with product descriptions for an online store.

To avoid code duplication and better structure large programs, there are **functions**.

Functions in JavaScript can be **built-in** or **user-defined**, that is, created specifically by the programmer. You have already worked with a built-in function `console.log()`. Today you will learn how to create your own.

§1. Basic syntax

Take a close look at the syntax of function creation:

```
function name(parameters) {
   // function body
}
```

To create a function, you need to write a keyword `function`, come up with a **name** for it and open parentheses. In parentheses, you can specify the **parameters**: data that you want to transfer to the program. The code of the function, also called **a function body**, must be written inside the curly brackets.

Based on the pseudo-code written above, let's try to write a function that outputs a string `"Find and book your ideal tour!"` to the console.

```
function writeMessage() {
   console.log("Find and book your ideal tour!");
}
```

Our new feature has an empty list of parameters and a name `writeMessage`. Let's try calling this function. To do this, you need to write the function name and a pair of parentheses.

```
function writeMessage() {
   console.log("Find and book your ideal tour!");
}

writeMessage(); // Find and book your ideal tour!
```

The call of `writeMessage()` executes the code written in the function body. The function can be called more than once:

```
function writeMessage() {
   console.log("Find and book your ideal tour!");
}

writeMessage(); // Find and book your ideal tour!
writeMessage(); // Find and book your ideal tour!
```

Here we display the message to the console twice. If we want, we can display it three, four, or even a hundred times.

The function is called anywhere in the file. You can call the function both before and after its creation.

Current topic:

Functions ···

Topic depends on:

✕ Arithmetic operators ···

Topic is required for:

- ForEach method ···
- Arrow functions ···
- setTimeout and setInterval ···
- Null and Undefined ···
- Increment and decrement ···
- Slicing ···
- Modules ···
- The "addEventListener" method ···

Table of contents:

```
writeMessage(); // Find and book your ideal tour!

function writeMessage() {
   console.log("Find and book your ideal tour!");
}

writeMessage(); // Find and book your ideal tour!
```

The ability to call a function before its creation is due to the peculiarities of JS file processing by browsers: the browser first goes through the whole code, finds all functions and only then starts executing the code.

§2. Parameters

In the previous examples, we left the parameters unattended: let's get back to that.

We can transfer any information inside the function using the parameters.

Let's try to pass two arguments to the function: `a` and `b`.

```
function quotient(a, b) {
  console.log(a / b);
}

quotient(10, 5); // 2
quotient(6, 2);  // 3
```

We called the function by passing the values that were copied into variables `a` and `b`, and used in the function body.

§3. Returning a value

In JS, it is possible to return the function result using the `return` statement:

```
function multiply(a, b) {
  return a * b;
}

let result = multiply(10, 2);
console.log(result); // 20
```

The `return` can be located anywhere in the body of the function. Once the execution of the code reaches the point of return (pun intended), the function stops and the value returns to the code that called it.

For now, you might doubt that `return` is worth your while, but bear with it: when you get to write larger and more complex functions, you will definitely find this operator useful.

Report a typo

137 users liked this theory. **1** didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

Comments (5)   Hints (0)   Useful links (0)   Show discussion