7.8 JS Functions

factory city - answered by Kristanna Agnes Nical

- A **Javascript function** is a block of code designed to perform a particular task.
- A Javascript function is executed when "something" invokes it (calls it).

```
function getProduct(p1, p2) {
  return p1 * p2;
}

console.log(getProduct(5, 6));
```

- The main advantage of functions is that you can reuse code: Define the code once, and use it many times.

Keyword(s)	Description	Sample code	Console output

`function`	• Used to declare a named or anonymous function, which can be invoked and executed to perform a specific task or computation.	`function calculate_total_cost(item_price, quantity) {`
		`  const tax_rate = 0.08;`
		`  const subtotal = item_price * quantity;`
		`  const tax_amount = subtotal * tax_rate;`
		`  const total_cost = subtotal + tax_amount;`
		`  return total_cost;`
		`}`
			`Total cost: 54`

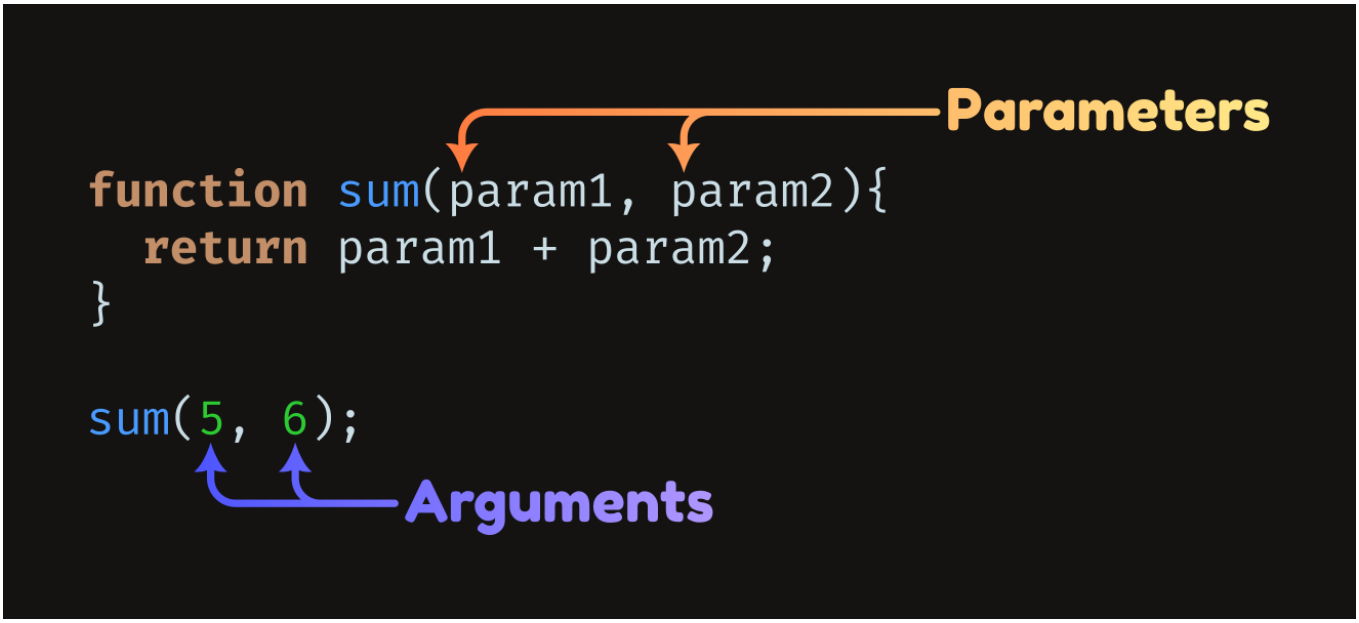
`return`	• Used to specify the value that a function should produce as its output, allowing data or results to be passed back to the calling code.	`const price = 10.0;`
		`const quantity = 5;`
		`const total = calculate_total_cost(price, quantity);`
		
		`console.log("Total cost:", total);`	

Writing functions

- A Javascript function is defined with the `function` keyword, followed by a name, followed by parentheses `()`.
  - Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
  - The parentheses may include parameter names separated by commas: (parameter1, parameter2, ...)
- The code to be executed, by the function, is placed inside curly brackets: `{}`

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

- **Function parameters** are listed inside the parentheses () in the function definition.
- **Function arguments** are the values received by the function when it is invoked.
  - Inside the function, the arguments (the parameters) behave as local variables.



Using a function

- The code inside the function will execute when "something" invokes (calls) the function:
  1. When an event occurs (when a user clicks a button)
  2. When it is invoked (called) from JavaScript code
  3. Automatically (self invoked)

Retrieving a value from a function

- When JavaScript reaches a `return` statement, the function will stop executing.
  - If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
- Functions often compute a **return** value. The return value is "returned" back to the "caller":
- The following code calculates the product of two numbers, and return the result:

```
let x = getProduct(4, 3);   // Function is called, return value will end up in x

function getProduct(a, b) {
  return a * b;             // Function returns the product of a and b
}
```

The return value is `12` and is assigned to the variable `x`.

Functions used as variable values

- Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations.
- Instead of using a variable to store the return value of a function:

```
let x = toCelsius(77);
let text = "The temperature is " + x + " Celsius";
```

- You can use the function directly, as a variable value:

```
let text = "The temperature is " + toCelsius(77) + " Celsius";
```

- Variables declared within a JavaScript function become local to the function.

```
// code here can NOT use carName

function myFunction() {
  let carName = "Volvo";
  // code here CAN use carName
}

// code here can NOT use carName
console.log(carName)
```

- Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.
- Local variables are created when a function starts, and deleted when the function is completed.

Additional Material

- **Learn more**
  - W3Schools