



CRAFT
KNOWLEDGE

Objectives



JavaScript Functions

- WHAT IS FUNCTIONS
- DECLARE A FUNCTION
- CALLING A FUNCTION
- PARAMETERS VS. ARGUMENTS
- RETURNING A VALUE
- FUNCTION & VARIABLE SCOPE
- ANONYMOUS FUNCTIONS
- ARROW FUNCTIONS

What is JavaScript functions

- A JavaScript function is a block of code designed to perform a particular task.
- When developing an application, we often need to perform the same action in many places. To avoid repeating the same code all over places, we can use a function to wrap that code and reuse it.
- JavaScript provides many built-in functions such as `parseInt()` and `parseFloat()`
- A JavaScript function is executed when invokes it (calls it).
- Functions also make code readable & maintainable.

Declare a function

- To declare a function, you use the function keyword, followed by the function name, a list of parameters, and the function body as follows and return value or expression .

```
function calcArea(width, height) {  
  let result = width * height;  
  return result;  
};
```

The diagram illustrates the components of a JavaScript function declaration. Red arrows point from labels to specific parts of the code: 'function Keyword' points to 'function', 'function Name' points to 'calcArea', 'Parameters' points to '(width, height)', 'Body of the function Inside curly braces' points to the block between '{' and '}', and 'Return Statement' points to 'return result;'.

- Function keyword
- functionName
- Parameters
- function body
- return value

Calling a function

- To use a function, you need to call it.
- Calling a function is also known as invoking a function.
- To call a function, you use its name followed by arguments enclosing in parentheses.
- When calling a function, JavaScript executes the code inside the function body.

```
65  
66 functionName()  
67 functionName(argument)  
68  
69 hello()  
70 hello(Efrem)  
71
```

parameters vs. arguments

The diagram shows a function definition and a function call. In the function definition, the parameters 'param1' and 'param2' are highlighted in blue. An orange bracket labeled 'Parameters' points to these two parameters. In the function call 'sum(5, 6);', the arguments '5' and '6' are highlighted in green. A blue bracket labeled 'Arguments' points to these two arguments.

```
function sum(param1, param2){  
  return param1 + param2;  
}  
  
sum(5, 6);
```

- The terms parameters and arguments are often used interchangeably. However, they are essentially different.
- When declaring a function, you specify the parameters.
- However, when calling a function, you pass the arguments that are corresponding to the parameters.

Returning a value

- Every function in JavaScript implicitly returns undefined unless you explicitly specify a return value.
- To specify a return value for a function, you use the return statement followed by an expression or a value, like this:

```
16
17 function add(a, b)
18 {
19     return a + b;
20 }
21
```

Function & Variable Scope

- Scope refers to the availability of variables and functions in certain parts of the code.
- In JavaScript, a variable has two types of scope:
 - Global Scope
 - Local Scope

Global Scope

- A variable declared at the top of a program or outside of a function is considered a global scope variable.
- can be used anywhere in the program.
- The value of a global variable can be changed inside a function

Local Scope

- A variable can also have a local scope, i.e it can only be accessed within a function.
- can be accessed only inside the function

Anonymous Functions

- An anonymous function is a function without a name.
- The following shows how to define an anonymous function:
- Note that if you don't place the anonymous function inside the (), you'll get a syntax error. The () makes the anonymous function an expression that returns a function object.
- An anonymous function is not accessible after its initial creation. Therefore, you often need to assign it to a variable.

```
( function () {  
    console.log("hello")  
})();
```

Arrow functions

- ES6 introduced arrow function expressions that provide a shorthand for declaring anonymous functions:

```
63  
64  
65   let showagen = () => console.log('Anonymous function');  
66  
67
```