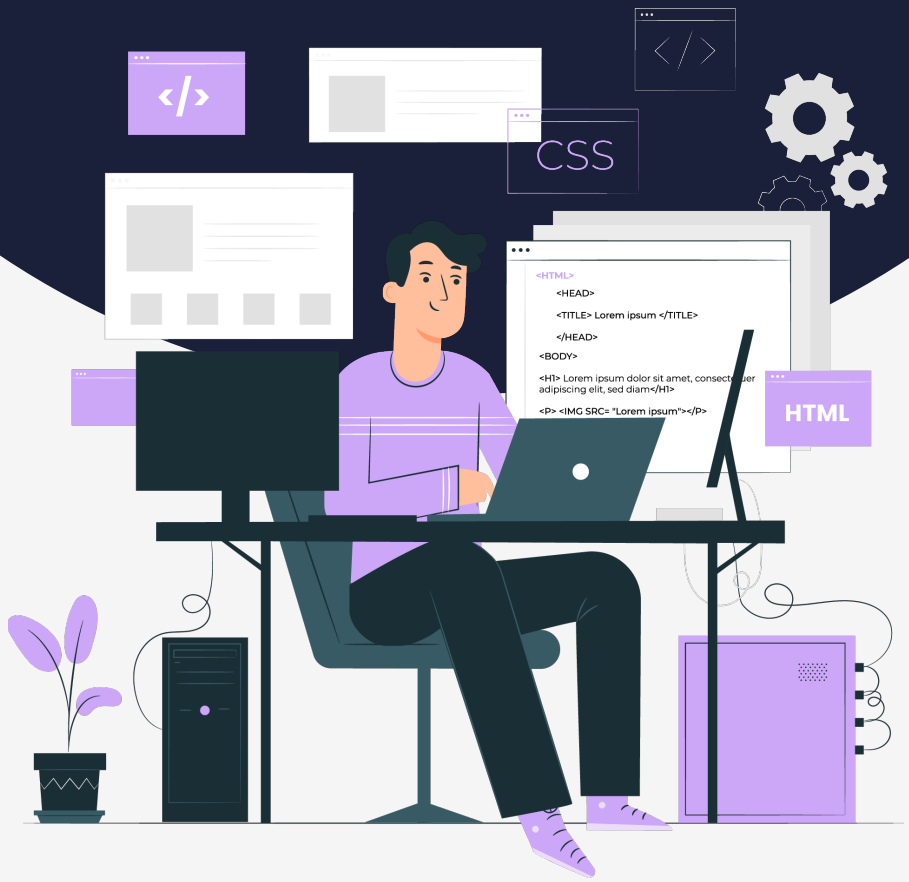
Lesson:

Why and What of Functions and Function Declaration



PW SKILLS

Topics Covered

1. Introduction to functions.
2. In Javascript, Functions are first-class citizens.
3. Advantages of using functions.
4. When should you consider using Functions?
5. Introduction to the function declaration.
6. The basic syntax for function declaration.
7. Calling a function.
8. Parameters, Arguments & Return statement.

The concept of functions came into existence to support the way we organize and structure code. A **function is a block of code** that **performs a specific task** and can be **reused throughout the program.**

This allows us to write more efficient and modular code, as well as the ability to easily update or change the behavior of the program by modifying the functions. Additionally, by using functions we can make the code more readable and easier to understand.

Some functions will be in-built in Javascript that can be directly used, and some are user-defined functions. We will be looking into them in further lectures.

In Javascript, functions are first-class citizens.

Earlier in JavaScript, functions were just simple blocks of code that performed a specific task.

But as the language evolved, the developers of JavaScript realized that functions could be much more powerful if they could be assigned to variables, passed as arguments to other functions, and returned from functions. With this, functions in JavaScript were able to become more versatile and flexible. They could be used in ways that were previously impossible.

This change was revolutionary for the developers of JavaScript, as it allowed them to write more efficient and modular code. They could now write reusable functions that could be used throughout their programs, making it easier to update and maintain their code.

Functions in JavaScript became an essential part of the language, allowing developers to write powerful and efficient code that could be easily reused and maintained.

Functions in JavaScript are first-class citizens, meaning they can be assigned to variables, passed as arguments to other functions, and returned from functions.

We will be looking into all of these in further lectures.

When developing an application, you often need to perform the same action in many places. Imagine a calculator application where you need to perform the operations such as sum, and difference multiple times. Here instead of writing a separate program for all conditions, we would write a function that can take any inputs, perform the operation and give us the result.

This is not only why we use functions. We use functions as it has many **advantages** like

1. Functions allow us to define a block of code once and reuse it multiple times throughout your program. So we need not to repeat the same lines of code every time.
2. While solving a problem we split them into subproblems so we could solve smaller problems to achieve the final result. Here functions can be used to break down complex problems into smaller, more manageable tasks.
3. Functions can make code more readable by giving it a clear structure and making it easy to understand what the code is doing.
4. Functions are tested separately before actual implementation. Because functions can be tested and debugged independently of the rest of the code, it can make it easier to identify and fix bugs.
5. Functions are the optimal way to manage the space which leads to higher performance of the program.

When should you consider using functions?

Functions should be used in a number of situations to take advantage of these benefits. One of the main use cases for functions is when you need to perform a specific task multiple times throughout your code. By defining that task in a function, you can call it whenever you need to perform that task, rather than duplicating the code. Functions can also be used to organize your code into logical units, making it easier to understand and maintain.

Functions can also be used to improve the readability and maintainability of your code by giving it a clear structure and making it easy to understand what the code is doing. Additionally, functions can be used to handle asynchronous operations like making an HTTP request, a timer, or an event listener, which we will be looking at in further sections.

Introduction to the function declaration.

A function declaration is a way to define a function and give it a name.

In JavaScript, functions must be declared before they can be used because of the way the JavaScript engine processes code.

When a function is declared, the JavaScript engine reserves a spot in memory for it and assigns the function's name to that memory location, so that it can be called later on.

Functions on declaration can be called multiple times with different inputs, without having to rewrite the same code over and over again.

The basic syntax for a function declaration is

```javascript
function functionName() {
  // function body
}
```

The function declaration in JavaScript has the following parts:

- The function keyword is used to declare a function. When the JavaScript engine encounters the function keyword, it knows that a function is being declared and it allocates memory for the function.
- In JavaScript, a function's name is used to identify and call the function. The name of a function is specified after the function keyword, followed by parentheses.

Function names are similar to variable names in that they must follow the same naming conventions
Function names are optional, you can define a function without giving it a name. We will be looking into this in further lectures.

- The function body is the block of code that is executed when the function is called. It is typically enclosed within curly braces {} and typically contains one or more statements that perform some action or calculation.
  The function body can contain any valid JavaScript code, including variable declarations, control flow statements, and other function calls.
  It's possible to have an empty function body, this doesn't raise any errors.

Calling a function.

Calling a function in JavaScript is very simple. Calling a function simply means executing a block of code that has been declared previously.

The most common way of calling a function is by using the function name followed by parentheses ().

```javascript
// Function declaration

function functionName() {

  // function body

}

// Calling a function

functionName()
```

parameters, argument, and return statement.

A parameter is a variable that is declared within the parentheses of the function declaration. It serves as a placeholder or a reference for the input value that will be passed to the function when it is called.

Syntax:

```javascript
function functionName(parameter1, parameter2) {
  // function body
}
```

Parameters are dummy names we pass/give a function when we are declaring a function.

When the function is called, the values are passed as arguments. These are assigned to the corresponding parameters. Arguments are the actual values that are passed to the function when it is called.

```javascript
// Function declaration
function functionName(parameter1, parameter2) {
  // function body
}

// Calling a function

functionName(argument1, argument2)
```

The number of arguments must always be equal to the number of parameters, else would raise an error.

The return statement is used to output a value from a function. The return statement terminates the execution of the function. It also specifies the value that should be returned to the calling code. The value returned by the function can be assigned to a variable or used in an expression.

Any statements that are written after the return statement are not executed.

JavaScript

```
// Function declaration

function functionName(parameter1, parameter2) {

  // function body

  return return_value;
}

// Calling a function

let result = functionName(argument1, argument2);

// The value of return_value will be stored in the result
```