skill academy

Javascript Functions

JavaScript Functions

skill academy

- JavaScript Functions are a **block of code** used to perform a particular **task**.
- JavaScript Code performing **similar tasks** can be grouped together as **functions** and can be **reused** any number of times.
- Functions can be called from **anywhere** in a program.
- A **big program** can be divided into a **number of small, manageable functions**.

Importance of JS Functions

- JavaScript Functions helps in **code reusability.**
- JavaScript Functions helps in creating compact programs with **less number of lines**.

Function Definition

skill academy

- A **Function** keyword is used to **define** a function.
- Function keywords should be followed by a **unique function name,** followed by **parentheses ().**
- Parentheses may include a **list of parameters** separated by comma **(*parameter1, parameter2, ...*)**
- The **code** to be executed should be surrounded by **curly braces**.

Syntax

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

Function Definition

skill academy

- When **invoked,** the Function receives values called **Function Arguments**.
- The arguments or parameters behave as **local variables** to a function.
- Functions can have **zero or any number of parameters**.
- The **Variable naming rule** can be used for **naming functions** as well.

An Example for zero parameter function

```
function sayHello() {
    document.write ("Hello there!");
}
```

Function Invocation

skill academy

- The code inside a function should be executed by calling / invoking it.

Syntax

```
functionName(arguments);
```

- We specify **parameters** within the parentheses while **declaring a function**.
- While **calling a function**, we pass **arguments** within the parentheses to the function.
- If **no argument** should be passed, then call the function with **empty** parentheses.
- If calling a function **without parentheses (),** it will return the **function definition** instead of the result.

Function Invocation

skill academy

```
<script>
function sayHello(f) {
  alert("Hello");
}
document.getElementById("demo").innerHTML = sayHello;
</script>
```

The above code will return the function definition itself

```
function sayHello(f) { alert("Hello"); }
```

Return Statement

skill academy

- By **default,** every function in JavaScript returns **undefined implicitly**.
- The **Return** statement should be the **last statement** in a function, i.e. the function will stop executing.
- If you want to **return a value** from a function, then you should **explicitly** mention a return statement.

Syntax

```
function functionName(parameter1,parameter2)
{
  return value;
}
```

Return Statement

skill academy

- The **return statement without a value** is used to **exit** the function prematurely.
- The **return value** is "**returned**" back to the function **invoking statement**.

```
function add(a, b) {
    return a + b;
}
```

The above example will return the result of the expression to actual code calling the function.

Local Variable

skill academy

- **Variables** declared **inside a function** will behave **local** to that function.
- These **Local variables** can be **accessed only** by that particular **function**.
- The **variables** with the **same name** can be used in **different functions**.
- When the function **starts** the local variables are **created** and when the function **completes** these variables will be **deleted.**

```
// code here can NOT use carName

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

// code here can NOT use carName
```