**JavaScript Functions**

A JavaScript Function is a block of code designed to perform a specific task.

* The function is executed only when it is called.

**Syntax:**

A JS function is defined with keyword ‘function’ followed by name, followed by parenthesis ().

* Function name contains letters, numbers, underscores and dollar sign
* Inside parenthesis we can include parameters separated by comma.
* Inside curly brackets we have to mention the code to be executed.

**To creating a function:**

function name(){

// code

};

**To call a function:**

name()

* Parameters are the values inside the parenthesis
* Arguments are the values received when it invoked. Arguments behave as local when they are inside the function.

**Return:**

* When the execution reaches the return statement, then it stops executing.
* When function is invoked from a statement it will return some value and stores it in the variable mentioned in the statement.

### **Arrow Functions**

Arrow functions are a more concise syntax for writing functions. They are often used for short, inline functions.

Ex:

const greet = (name) => {

console.log("Hello, " + name);

};

**Function Hoisting:**

In JavaScript, function declarations are hoisted. This means you can call a function before its declaration in the code, and it will work.

Ex:

*greet("Charlie"); // Output: Hello, Charlie*  
  
*function greet(name) {*  
 *console.log("Hello, " + name);*  
*}*

However, function expressions are not hoisted. If you try to call the function before the expression is defined, you’ll get an error:

*greet("Charlie"); // Error: greet is not a function*  
  
*const greet = function(name) {*  
 *console.log("Hello, " + name);*  
*};*

### **Callback Functions**

A callback function is a function passed into another function as an argument, which is then executed later, typically after some event or operation.

Ex:

*function greet(name, callback) {*

*console.log("Hello, " + name);*

*callback();*

*}*

*greet("David", function() {*

*console.log("Welcome to the site!");*

*});*

### **Immediately Invoked Function Expression (IIFE)**

An **IIFE** is a function that runs as soon as it is defined.

It doesn't need to be called separately. Instead, it is created and executed immediately, usually to keep some code inside its own little "box" so that it doesn't interfere with other parts of the code.

**Bind, Call, and Apply**

JavaScript functions have three methods that allow you to change their, this context: bind, call, and apply.

* **call**: Immediately invokes the function with a specific this value and arguments.
* **apply**: Similar to call, but arguments are passed as an array.
* **bind**: Returns a new function with a specific this value and arguments, but does not invoke it immediately.

### **SetTimeout and SetInterval Functions**

* **setTimeout()**: Executes a function after a specified delay (in milliseconds).
* **setInterval()**: Repeatedly executes a function at specified intervals.

### **Asynchronous Functions: async and await**

Asynchronous functions help in dealing with operations that take time, like API requests. You use async to define a function that returns a Promise and await is used to wait for a promise to resolve.