JS-4

Agenda

- Types of functions

Types of functions

By the end of this lecture, students will understand different types of functions in JavaScript, including function statements, function expressions, arrow functions, and Immediately Invoked Function Expressions (IIFE). They will also learn how to handle parameters and return values.

1. Introduction to Functions

A function is a block of code designed to perform a particular task. A function is executed when something invokes (calls) it. Functions allow you to reuse code and make your programs more modular and readable.

2. Function Statement

A function statement (or function declaration) defines a named function with the specified parameters. Here's how you define and use a function statement:

Syntax:

```
function functionName(parameters) {
    // code to be executed
}
```

Example:

```
function greet(name) {
    return `Hello, ${name}!`;
}

console.log(greet('Alice')); // Output: Hello, Alice!
```

Key Points

1. **Passing Parameters**:
    - Parameters are variables listed as a part of the function definition.
    - If you pass fewer arguments than the number of parameters, the remaining parameters are `undefined`.

```
function greet(name) {
    return `Hello, ${name}!`;
}

console.log(greet('Alice')); // Output: Hello, Alice!
console.log(greet()); // Output: Hello, undefined!
```

2. **Returning a Value**:
    - The `return` statement stops the execution of a function and returns a value to the caller.
    - If no `return` statement is used, the function returns `undefined`.

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

function noReturn() {
    console.log('This function does not return anything');
}

console.log(add(2, 3)); // Output: 5
console.log(noReturn()); // Output: This function does not return
anything \n undefined
```

3. Function Expression

A function expression defines a function as part of a larger expression syntax (typically a variable assignment). These functions can be named or anonymous.

Syntax:

```
const functionName = function(parameters) {
    // code to be executed
};
```

Example:

```
const greet = function(name) {
    return `Hello, ${name}!`;
};

console.log(greet('Bob')); // Output: Hello, Bob!
```

Key Points

1. **Anonymous Functions**:
    - Function expressions can be anonymous (without a name).

    ```
    const greet = function(name) {
        return `Hello, ${name}!`;
    };
    ```

2. **Named Functions**:
    - Function expressions can also have a name.

    ```
    const greet = function sayHello(name) {
        return `Hello, ${name}!`;
    };
    ```

4. Arrow Functions

Arrow functions provide a shorter syntax for writing functions and lexically bind the `this` value.

Syntax:

```
const functionName = (parameters) => {
    // code to be executed
};
```

Example:

```
const greet = (name) => `Hello, ${name}!`;

console.log(greet('Charlie')); // Output: Hello, Charlie!
```

Key Points

1. **Short Syntax**:
    - If the function body contains only one statement, you can omit the curly braces and `return` keyword.

    ```
    const add = (a, b) => a + b;
    ```

2. **No `this` Context**:
    - Arrow functions do not have their own `this` context. (we will understand it with example in level up)

5. Immediately Invoked Function Expressions (IIFE)

An IIFE is a function that is executed immediately after it is defined. This pattern is often used to create a private scope for variables.

Syntax:

```
(function() {
    // code to be executed immediately
})();
```

Example:

```
(function(name) {
    console.log(`Hello, ${name}!`);
})('Dave'); // Output: Hello, Dave!
```

Key Points

1. **Creating Private Scopes**:

- IIFEs are useful for avoiding global variables and creating private scopes.

```
(function() {
    let privateVar = 'This is private';
    console.log(privateVar);
})();

// console.log(privateVar); // Error: privateVar is not defined
```

Summary

1. **Function Statement**:

```
function greet(name) {
    return `Hello, ${name}!`;
}
console.log(greet('Alice')); // Output: Hello, Alice!
```

2. **Function Expression**:

```
const greet = function(name) {
    return `Hello, ${name}!`;
};
console.log(greet('Bob')); // Output: Hello, Bob!
```

3. **Arrow Function**:

```
const greet = (name) => `Hello, ${name}!`;
console.log(greet('Charlie')); // Output: Hello, Charlie!
```

4. **IIFE**:

```
(function(name) {
    console.log(`Hello, ${name}!`);
})('Dave'); // Output: Hello, Dave!
```

Conclusion

In this lecture, we covered:

- Function Statements
- Function Expressions
- Arrow Functions
- Immediately Invoked Function Expressions (IIFE)

We also discussed how to handle parameters and return values. Understanding these different types of functions will help you write more versatile and maintainable JavaScript code. Practice writing each type of function to solidify your understanding. Happy coding!

Note : rest of the topics that a covered in this current lecture is moved to next lecture . so you can refer to the next session notes for continuity