13. FIRST CLASS FUNCTIONS 🔥ft. Anonymous Functions

📘 Index (Table of Contents)

1. **Introduction to First-Class Functions**
2. **Understanding Anonymous Functions**
3. **Function Statements vs. Function Expressions**
4. **Practical Examples**
5. **Summary and Key Takeaways**

❓ Questions and Answers

1. What Are First-Class Functions in JavaScript?

Q: What does it mean for functions to be first-class citizens in JavaScript?

A: In JavaScript, functions are first-class citizens, meaning they can be.

- **Assigned to variables:** Functions can be stored in variables.
- **Passed as arguments:** Functions can be passed as arguments to other functions.
- **Returned from other functions:** Functions can be returned from other functions.
- **Stored in data structures:** Functions can be stored in arrays, objects, etc.

Analogy: Think of functions as tools in a toolbox. You can use them, pass them to others, or store them for later use.

---

2. What Are Anonymous Functions?

Q: What is an anonymous function in JavaScript?

A: An anonymous function is a function that is defined without a name. It is often used as an argument to other functions or assigned to variables.

Example:

```
const greet = function() {
  console.log("Hello, World!");
};
greet(); // Logs: Hello, World!
```

Explanation: Here, the function is assigned to the variable `greet` and can be invoked using that variable.

---

3. What Is the Difference Between Function Statements and Function Expressions?

Q: How do function statements differ from function expressions?

A: The key differences are:

- **Function Statement (Declaration):** A function defined with the `function` keyword followed by a name. It is hoisted, meaning it can be called before its definition.

```
function sayHello() {
    console.log("Hello!");
}
sayHello(); // Logs: Hello!
```

- **Function Expression:** A function defined inside an expression, often anonymous. It is not hoisted and cannot be called before its definition.

```
const sayGoodbye = function() {
  console.log("Goodbye!");
};
sayGoodbye(); // Logs: Goodbye!
```

Explanation: Function statements are hoisted and can be called before their definition, whereas function expressions are not hoisted and must be defined before they are called.

---

4. How Are Anonymous Functions Used in JavaScript?

Q: Where are anonymous functions commonly used in JavaScript?

A: Anonymous functions are commonly used in:

- **Callbacks:** Passed as arguments to other functions.

```
setTimeout(function() {
  console.log("Executed after 1 second");
}, 1000);
```

- **Event Handlers:** Assigned to events.

```
document.getElementById("myButton").addEventListener("click", function
() {
  alert("Button clicked!");
});
```

- **Array Methods:** Used with methods like `map`, `filter`, and `reduce`.

```
const numbers = [1, 2, 3];
const doubled = numbers.map(function(num) {
  return num * 2;
});
console.log(doubled); // Logs: [2, 4, 6]
```

Explanation: In these examples, anonymous functions are used to define behavior inline without the need to create named functions.

🔑 Summary and Key Takeaways

- **First-Class Functions:** Functions in JavaScript can be treated as values, assigned to variables, passed as arguments, and returned from other functions.
- **Anonymous Functions:** Functions without names, often used as arguments or assigned to variables.
- **Function Statements vs. Function Expressions:** Function statements are hoisted and can be called before their definition, whereas function

expressions are not hoisted and must be defined before they are called.