Documentation on JS functions

JavaScript Functions

A JavaScript function is a block of code designed to perform a particular task. Functions are one of the fundamental building blocks in JavaScript.

1. Defining Functions

JavaScript supports several ways to define functions:

Function Declaration

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

Function Expression

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

Arrow Function

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

2. Calling Functions

To execute a function, you "call" it using its name followed by parentheses:

```
greet("Alice"));
```

3. Parameters and Arguments

Functions can accept parameters (placeholders) and arguments (values passed when calling the function):

```
function add(a, b) {
   return a + b;
}
console.log(add(3, 5));
```

Default parameters can be set:

```
function greet(name = "Guest") {
   return `Hello, ${name}!`;
}
console.log(greet());
```

4. Anonymous Functions

Functions without a name, often used as callbacks:

```
setTimeout(function() {
  console.log("This runs after 2 seconds");
}, 2000);
```

5. Immediately Invoked Function Expressions (IIFE)

These execute as soon as they are defined:

```
(function() {
  console.log("IIFE");
})();
```

6. Higher-Order Functions

Functions that accept other functions as arguments or return functions:

```
function firstFunction(a, b, secondFunction) {
  return secondFunction(a, b);
}
const result = firstFunction(5, 3, (x, y) => x * y);
console.log(result);
```

7. Closures

A closure is a function that retains access to variables in its lexical scope:

```
function makeCounter() {
  let count = 0;
  return function() {
    count++;
    return count;
  };
}
const counter = makeCounter();
console.log(counter()); // Output: 1
console.log(counter()); // Output: 2
```

8. Rest Parameters

Allows a function to accept an indefinite number of arguments:

```
function sum(...numbers) {
  return numbers.reduce((total, num) => total + num, 0);
}
console.log(sum(1, 2, 3, 4)); // Output: 10
```

9. Async Functions

To handle asynchronous operations:

```
async function fetchData() {
   const response = await fetch("https://api.example.com/data");
   const data = await response.json();
   return data;
}
```