In JavaScript, the **arguments** keyword refers to an array-like object that is automatically available inside non-arrow functions. It contains all the arguments passed to that function when it is invoked.

Here are the key characteristics of the `arguments` object:

- **Array-like, not a true array:**

  It has a `length` property and elements can be accessed by index (e.g., `arguments[0]`, `arguments[1]`), similar to an array. However, it lacks array methods like `forEach()`, `map()`, or `slice()` directly on the `arguments` object itself.

- **Contains all passed arguments:**

  Regardless of how many parameters are formally declared in the function's definition, the `arguments` object will contain all values actually passed to the function during its call.

- **Local to the function:**

  The `arguments` object is only accessible within the function's scope where it's called.

- **Synchronization with parameters (in some cases):**

  In non-strict mode and without using rest, default, or destructured parameters, changes to elements within the `arguments` object can affect the corresponding named parameters, and vice versa.

- **Not available in arrow functions:**
  Arrow functions do not have their own `arguments` object. If you need to access arguments in an arrow function, you would typically use rest parameters (`...args`).

```
function greet(name) {
  console.log(arguments[0]); // Accesses the first argument
  console.log(arguments.length); // Shows the number of arguments passed
}

greet("Alice"); // Output: Alice, 1

function sum() {
  let total = 0;
  for (let i = 0; i < arguments.length; i++) {
    total += arguments[i];
  }
  return total;
}

console.log(sum(1, 2, 3, 4)); // Output: 10
```