JavaScript strings are sequences of characters used to represent text. In JavaScript, strings are a primitive data type, meaning they are not objects and are immutable. This means that once a string is created, it cannot be changed. However, you can create new strings based on existing ones through various methods and operations.

Here are some key characteristics and operations related to JavaScript strings:

1. **Creating Strings**: Strings in JavaScript can be created using single quotes (`'`), double quotes (`"`), or backticks (```). For example:
   ```javascript
   var singleQuotesString = 'Hello, world!';
   var doubleQuotesString = "Hello, world!";
   var backticksString = `Hello, world!`;
   ```

2. **String Length**: You can find the length of a string using the `length` property. For example:
   ```javascript
   var str = "Hello, world!";
   console.log(str.length); // Outputs: 13
   ```

3. **Accessing Characters**: You can access individual characters in a string using bracket notation (`[]`). JavaScript strings are zero-indexed, meaning the first character has an index of 0. For example:
   ```javascript
   var str = "Hello, world!";
   console.log(str[0]); // Outputs: 'H'
   console.log(str[6]); // Outputs: 'w'
   ```

4. **Concatenation**: You can concatenate (join together) strings using the `+` operator or the `concat()` method. For example:
   ```javascript
   var str1 = "Hello, ";
   var str2 = "world!";
   var result = str1 + str2; // Using +
   var result2 = str1.concat(str2); // Using concat()
   console.log(result); // Outputs: "Hello, world!"
   console.log(result2); // Outputs: "Hello, world!"
   ```

5. **String Methods**: JavaScript provides many built-in methods for working with strings, such as `toUpperCase()`, `toLowerCase()`, `substring()`, `indexOf()`, `split()`, `replace()`, and more. For example:

```javascript
   var str = "Hello, world!";
   console.log(str.toUpperCase()); // Outputs: "HELLO, WORLD!"
   console.log(str.substring(0, 5)); // Outputs: "Hello"
   console.log(str.indexOf("world")); // Outputs: 7
   ```

6. **Template Literals**: Template literals, introduced in ES6, allow you to create strings with embedded expressions. They are enclosed in backticks (```). For example:
   ```javascript
   var name = "Alice";
   var greeting = `Hello, ${name}!`;
   console.log(greeting); // Outputs: "Hello, Alice!"
   ```

These are some basic aspects of JavaScript strings. They are fundamental for working with textual data in JavaScript applications.