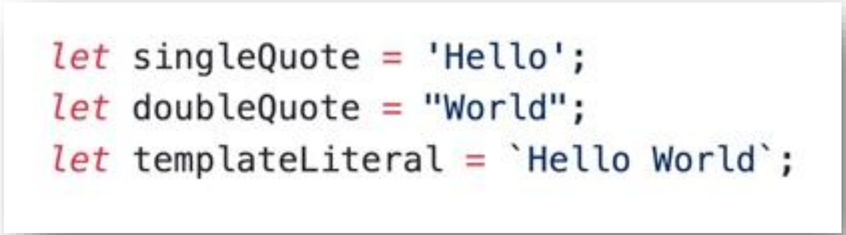
# Lecture 7: Strings & Error Handling

## What are Strings?

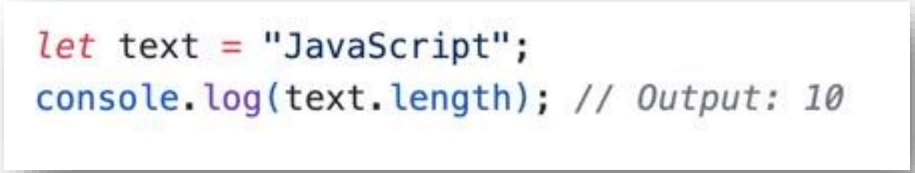
- A **string** is a sequence of characters used to represent text in JavaScript.
- It can contain letters, numbers, symbols, and spaces.
- **Strings** are defined within single **‘ ‘** , double **“ “** , or backticks **` `** for template literals.



```javascript
let singleQuote = 'Hello';
let doubleQuote = "World";
let templateLiteral = `Hello World`;
```

## String Properties

**Length property**

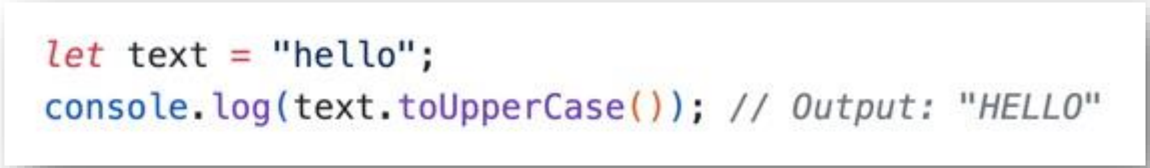


```
let text = "JavaScript";
console.log(text.length); // Output: 10
```

## String Methods

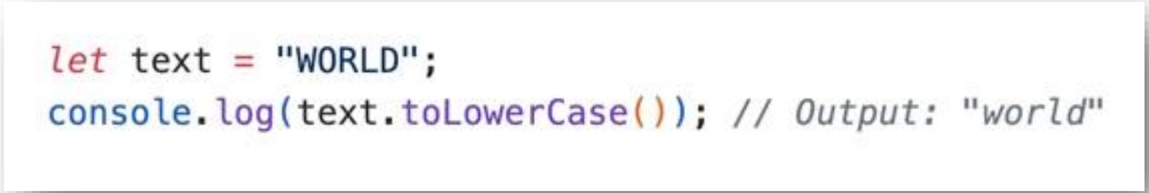
### Changing Case

#### To Upper Case



```javascript
let text = "hello";
console.log(text.toUpperCase()); // Output: "HELLO"
```

#### To Lower Case

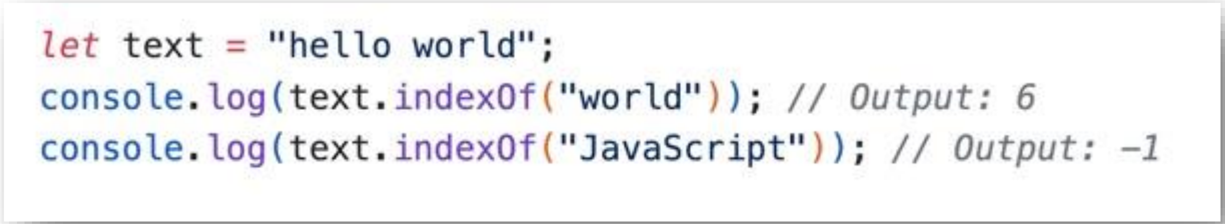


```javascript
let text = "WORLD";
console.log(text.toLowerCase()); // Output: "world"
```

## String Methods

### String Searching

**indexOf()** **-** Returns the index of the first occurrence of a substring, or -1 if not found

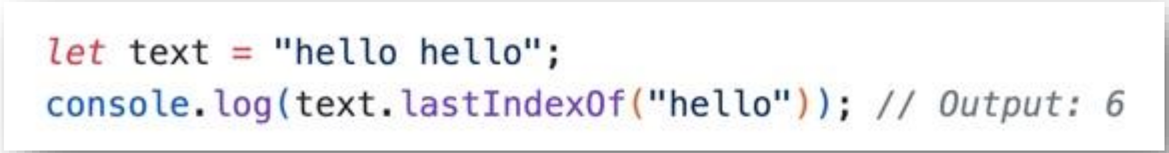


```
let text = "hello world";
console.log(text.indexOf("world")); // Output: 6
console.log(text.indexOf("JavaScript")); // Output: -1
```

# String Methods

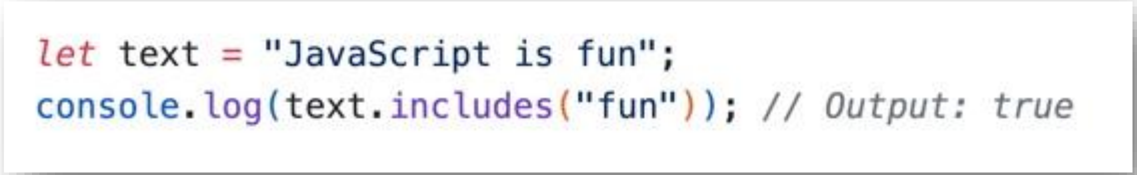
## String Searching

**lastIndexOf()** - Returns the index of the last occurrence of a substring



```
let text = "hello hello";
console.log(text.lastIndexOf("hello")); // Output: 6
```

**includes()** - Checks if a string contains a certain substring (true / false)

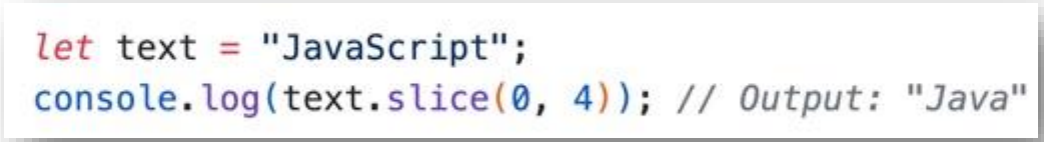


```
let text = "JavaScript is fun";
console.log(text.includes("fun")); // Output: true
```

# String Methods

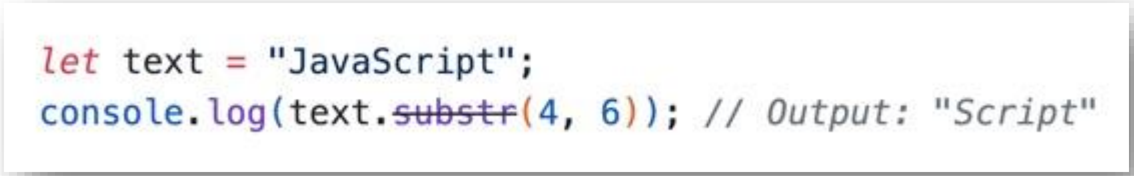
## Extracting Substrings

**slice(start, end) –** Extracts a part of a string from start index to end index, not including the end



```
let text = "JavaScript";
console.log(text.slice(0, 4)); // Output: "Java"
```

**substr(start, length) –** Extracts a substring from start index and continues for the given length

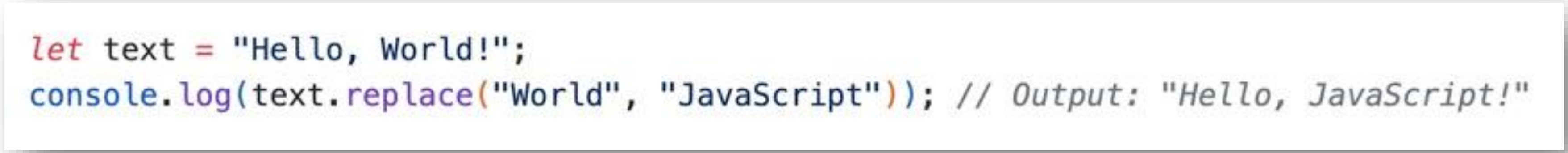


```
let text = "JavaScript";
console.log(text.~~substr~~(4, 6)); // Output: "Script"
```

# String Methods

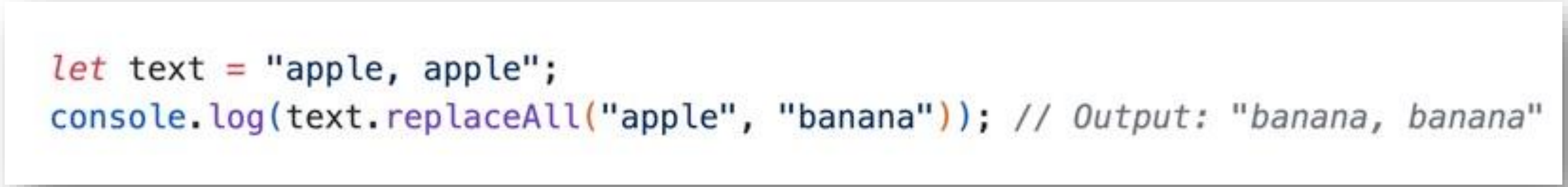
## Replacing Parts of String

**replace(oldValue, newValue) –** Replaces the first occurrence of the old value with the new value



```
let text = "Hello, World!";
console.log(text.replace("World", "JavaScript")); // Output: "Hello, JavaScript!"
```

**replaceAll(oldValue, newValue) –** Replaces all occurrences of the old value with the new value

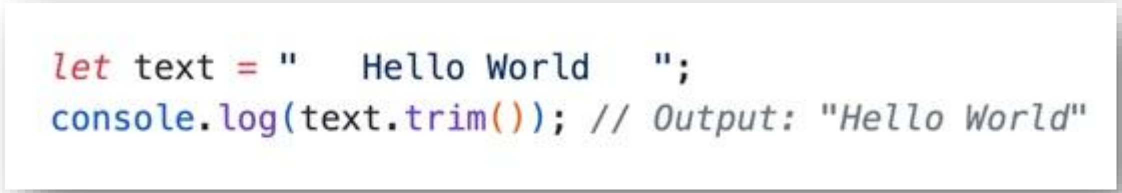


```
let text = "apple, apple";
console.log(text.replaceAll("apple", "banana")); // Output: "banana, banana"
```

## String Methods

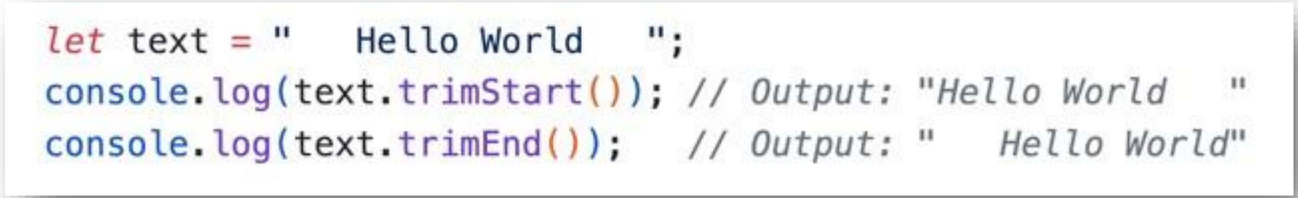
### Trimming whitespace

**trim() –** Removes whitespace from both ends of a string



```
let text = "   Hello World   ";
console.log(text.trim()); // Output: "Hello World"
```

**trimStart() / trimEnd()** **-** Removes whitespace from start or end

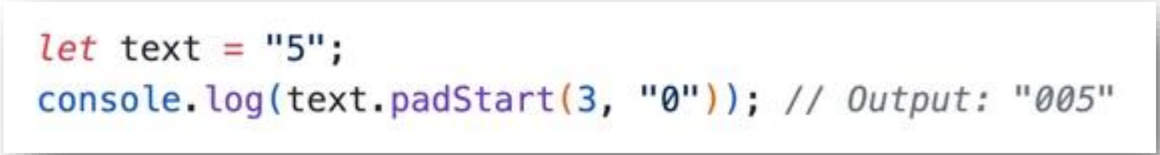


```
let text = "   Hello World   ";
console.log(text.trimStart()); // Output: "Hello World   "
console.log(text.trimEnd());   // Output: "   Hello World"
```

## String Methods

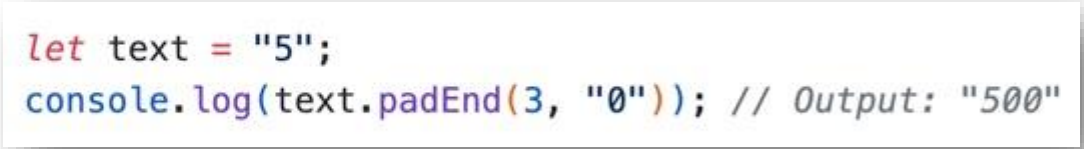
**Padding Strings**

**padStart(targetLength, padString) –** Pads the beginning of the string until it reaches target length



```
let text = "5";
console.log(text.padStart(3, "0")); // Output: "005"
```

**padEnd(targetLength, padString) –** Pads the end of the string until it reaches target length



```
let text = "5";
console.log(text.padEnd(3, "0")); // Output: "500"
```

## Escaping Characters

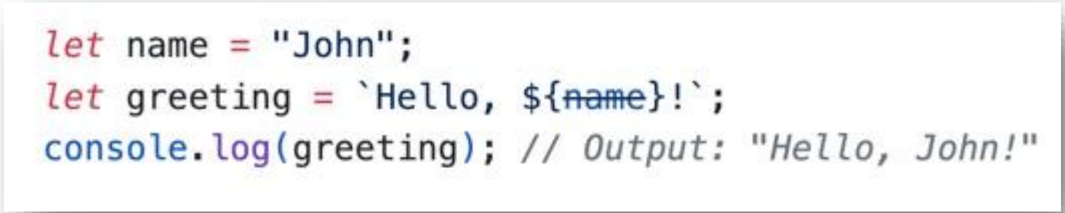
- **\n –** for a new line
- **\’ –** for a single quotes
- **\” –** for a double quotes
- **\\ -** for a backslash

## Template Literals

**Template literals** allow you to embed expressions and multi-line strings.

Use backticks **( ` )** instead of single or double quotes.

Embed JavaScript expressions inside **${ }**



```
let name = "John";
let greeting = `Hello, ${name}!`;
console.log(greeting); // Output: "Hello, John!"
```

## Error Handling in JavaScript

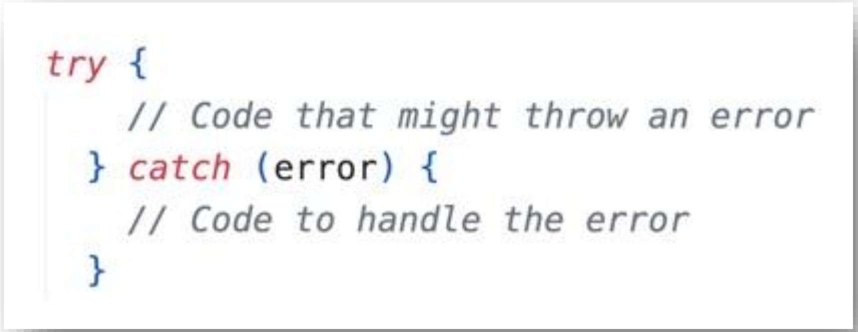
**Errors** in JavaScript occur when something goes wrong during code execution, causing the program to stop.

**Common error types:**

- **SyntaxError –** Incorrect syntax in the code
- **ReferenceError –** Using a variable that hasn’t  been declared
- **TypeError –** Inappropriate data type or a method

# The try...catch Statement

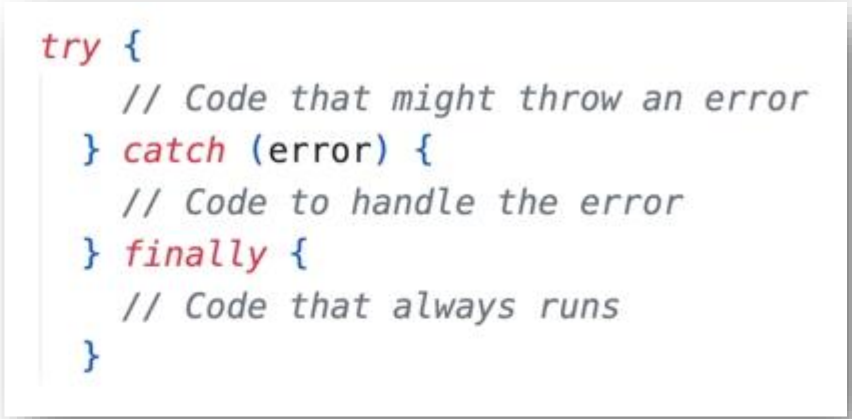
Used to handle errors without stopping the program.



```
try {
    // Code that might throw an error
  } catch (error) {
    // Code to handle the error
  }
```

## The finally Block

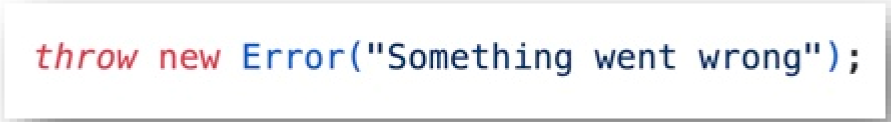
Executes code after try and catch, regardless of the result



```
try {
    // Code that might throw an error
  } catch (error) {
    // Code to handle the error
  } finally {
    // Code that always runs
  }
```

## Throwing Errors

Manually create (**throw**) an error in your code



```
throw new Error("Something went wrong");
```