Regular Expressions in JavaScript

Regular expressions (regex) are patterns used to match character combinations in strings. In JavaScript, regular expressions are used with the `RegExp` object or the `RegExp` literal syntax.

Basic Syntax

A regular expression can be created in two ways:

1. **Literal Syntax**:

```
let regex = /pattern/flags;
```

2. **RegExp Constructor**:

```
let regex = new RegExp("pattern", "flags");
```

Flags

- `g` - Global search
- `i` - Case-insensitive search
- `m` - Multi-line search

Common Patterns

Matching Specific Characters

- **Literal Characters**: Matches exact characters.

```
let regex = /abc/;
console.log(regex.test("abcdef")); // true
console.log(regex.test("abxyz")); // false
```

- **Dot (.)**: Matches any single character except newline.

```
let regex = /a.c/;
console.log(regex.test("abc")); // true
console.log(regex.test("a c")); // true
console.log(regex.test("ac")); // false
```

Special Characters

- **.** : Matches any single character except newline.
- **\d**: Matches any digit (0-9).
- **\w**: Matches any word character (alphanumeric + underscore).
- **\s**: Matches any whitespace character.
- **\b**: Matches a word boundary.
- **^**: Matches the beginning of the string.
- **$**: Matches the end of the string.
- **[]**: Matches any one of the characters inside the brackets.
- **|**: Alternation (acts like OR).

Character Classes

- **Digit (`\d`)**: Matches any digit (0-9).

```
let regex = /\d/;
console.log(regex.test("123")); // true
console.log(regex.test("abc")); // false
```

- **Non-Digit (`\D`)**: Matches any non-digit.

```
let regex = /\D/;
console.log(regex.test("123")); // false
console.log(regex.test("abc")); // true
```

- **Whitespace (`\s`)**: Matches any whitespace character.

```
let regex = /\s/;
console.log(regex.test("hello world")); // true
console.log(regex.test("helloworld")); // false
```

- **Non-Whitespace (`\S`)**: Matches any non-whitespace character.

```
let regex = /\S/;
console.log(regex.test("hello world")); // true
console.log(regex.test("     ")); // false
```

Quantifiers

- **Exact `{n}`**: Matches exactly `n` occurrences.

```
let regex = /a{3}/;
console.log(regex.test("aaa")); // true
```

```
console.log(regex.test("aa")); // false
```

- **Range** `{n,m}`: Matches between `n` and `m` occurrences.

```
let regex = /a{2,4}/;
console.log(regex.test("aa")); // true
console.log(regex.test("aaa")); // true
console.log(regex.test("aaaa")); // true
console.log(regex.test("a")); // false
```

- **Zero or More** `*`: Matches zero or more occurrences.

```
let regex = /ab*/;
console.log(regex.test("a")); // true
console.log(regex.test("ab")); // true
console.log(regex.test("abbbb")); // true
```

- **One or More** `+`: Matches one or more occurrences.

```
let regex = /ab+/;
console.log(regex.test("a")); // false
console.log(regex.test("ab")); // true
console.log(regex.test("abbbb")); // true
```

- **Zero or One** `?`: Matches zero or one occurrence.

```
let regex = /ab?/;
console.log(regex.test("a")); // true
console.log(regex.test("ab")); // true
console.log(regex.test("abbbb")); // true
```

Anchors

- **Start of String** `^`: Asserts position at the start of the string.

```
let regex = /^a/;
console.log(regex.test("abc")); // true
console.log(regex.test("bca")); // false
```

- **End of String** `$`: Asserts position at the end of the string.

```
let regex = /a$/;
console.log(regex.test("cba")); // true
console.log(regex.test("abc")); // false
```

Common Use Cases

Email Validation

```
let emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
console.log(emailRegex.test("test@example.com")); // true
console.log(emailRegex.test("test@ example.com")); // false
```