

Programming with JavaScript

Regular Expression

Introduction to Regular Expressions in JavaScript

- ▶ Regular expressions (Regex) in JavaScript are patterns used for matching character combinations in strings.

Why Use Regular Expressions in JavaScript?

- ▶ - Efficient pattern matching
- ▶ - Used for validation, searching, and replacing text
- ▶ - Simplifies complex string operations

Creating a Regular Expression

- ▶ Two ways to create regex in JavaScript:
- ▶ 1. Using literal syntax: `/pattern/flags`
- ▶ 2. Using RegExp constructor: `new RegExp('pattern', 'flags')`

Basic Example

- ▶ Example in JavaScript:
- ▶ `const regex = /hello/;`
- ▶ Let `str = "hello this is hello and bye hello"`
- ▶ `console.log(regex.replace(str, "bye"));` // Output: true

Common Regular Expression Flags

- ▶ - g (global): Match all occurrences
- ▶ - i (ignore case): Case-insensitive match
- ▶ - m (multiline): Multi-line matching

Example: Using Flags

- ▶ Example in JavaScript:
- ▶ `const regex = /hello/i;`
- ▶ `console.log(regex.test('Hello World')); // Output: true`

Character Classes

- ▶ - `\d` matches any digit (0-9)
- ▶ - `\w` matches any word character (a-z, A-Z, 0-9, _)
- ▶ - `\s` matches any whitespace character

Example: Character Classes

- ▶ Example in JavaScript:
- ▶ `const regex = /\d{0,6}/;`
- ▶ `console.log(regex.test('There are 123 apples')); //`
Output: true

Quantifiers

- ▶ - * (zero or more times)
- ▶ - + (one or more times)
- ▶ - ? (zero or one time)
- ▶ - {n} (exactly n times)

Example: Quantifiers

- ▶ Example in JavaScript:
- ▶ `const regex = /a{2,4}/;`
- ▶ `console.log(regex.test('aaa')); // Output: true`

Anchors: Start and End

- ▶ - ^ (caret) matches the start of a string
- ▶ - \$ (dollar) matches the end of a string
- ▶ `/^hello/`

Example: Anchors

- ▶ Example in JavaScript:
- ▶ `const regex = /^Hello/;`
- ▶ `console.log(regex.test('Hello World')); // Output: true`

Grouping and Capturing

- ▶ - () groups patterns
- ▶ - Allows extracting matched parts using `match()`

Example: Grouping

- ▶ Example in JavaScript:
- ▶ `const regex = /(hello) (world)/;`
- ▶ `console.log('hello world'.match(regex)); // Output: ['hello world', 'hello', 'world']`

Alternation | (OR Operator)

- ▶ - `a|b` matches either 'a' or 'b'
- ▶ - Useful for optional words or characters

Example: Alternation

- ▶ Example in JavaScript:
- ▶ `const regex = /apple|banana/;`
- ▶ `console.log(regex.test('I like apple')); // Output: true`

Escape Characters

- ▶ - \ (backslash) escapes special characters
- ▶ - Example: \. matches a literal dot

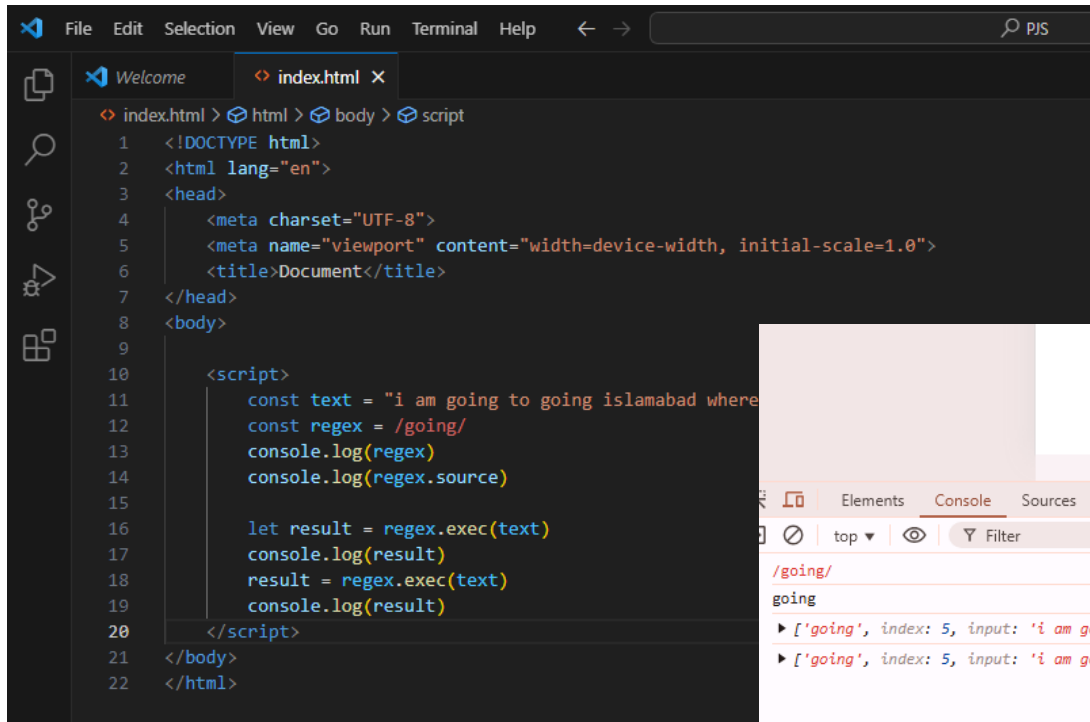
Example: Escape Characters

- ▶ Example in JavaScript:
- ▶ `const regex = /file\\.txt/;`
- ▶ `console.log(regex.test('file.txt')); // Output: true`

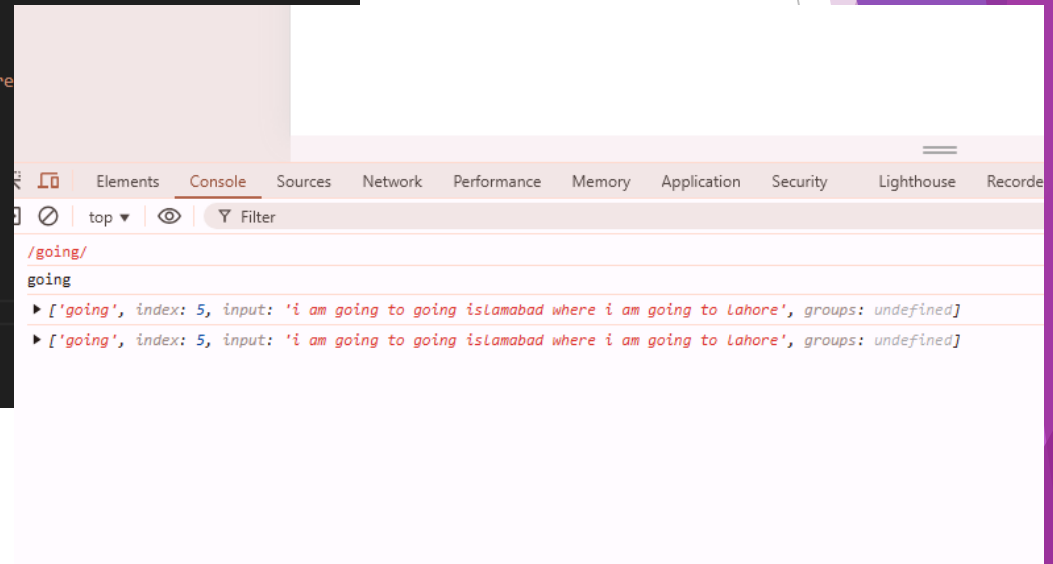
Replacing Text with Regex

- ▶ Example in JavaScript:
- ▶ `const text = 'I love regex';`
- ▶ `console.log(text.replace(/regex/, 'JavaScript')); //`
Output: 'I love JavaScript'

Exec function

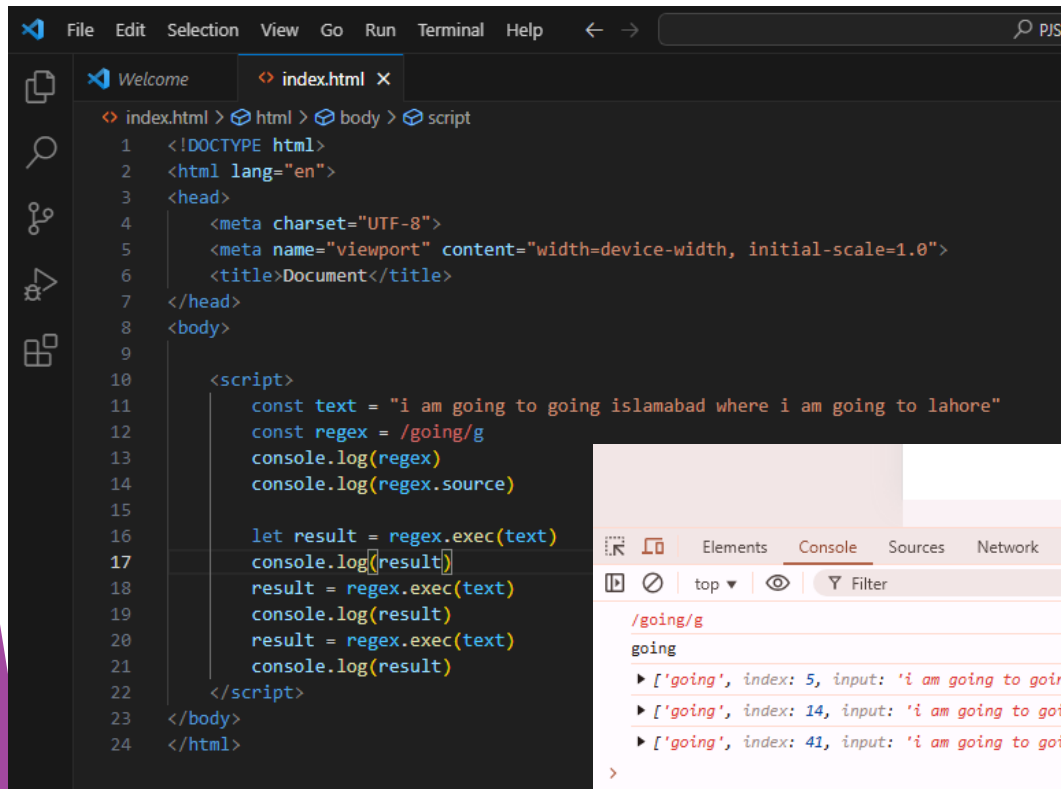


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9
10   <script>
11     const text = "i am going to going islamabad where i am going to Lahore";
12     const regex = /going/;
13     console.log(regex);
14     console.log(regex.source);
15
16     let result = regex.exec(text);
17     console.log(result);
18     result = regex.exec(text);
19     console.log(result);
20   </script>
21 </body>
22 </html>
```

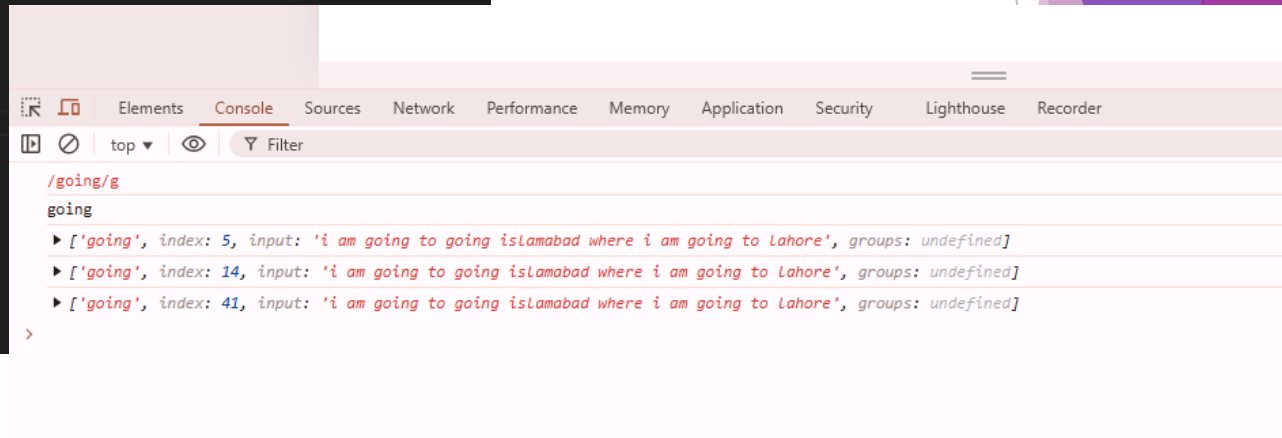


```
top Filter
/going/
going
▶ ['going', index: 5, input: 'i am going to going islamabad where i am going to Lahore', groups: undefined]
▶ ['going', index: 5, input: 'i am going to going islamabad where i am going to Lahore', groups: undefined]
```

Exec function with global flag

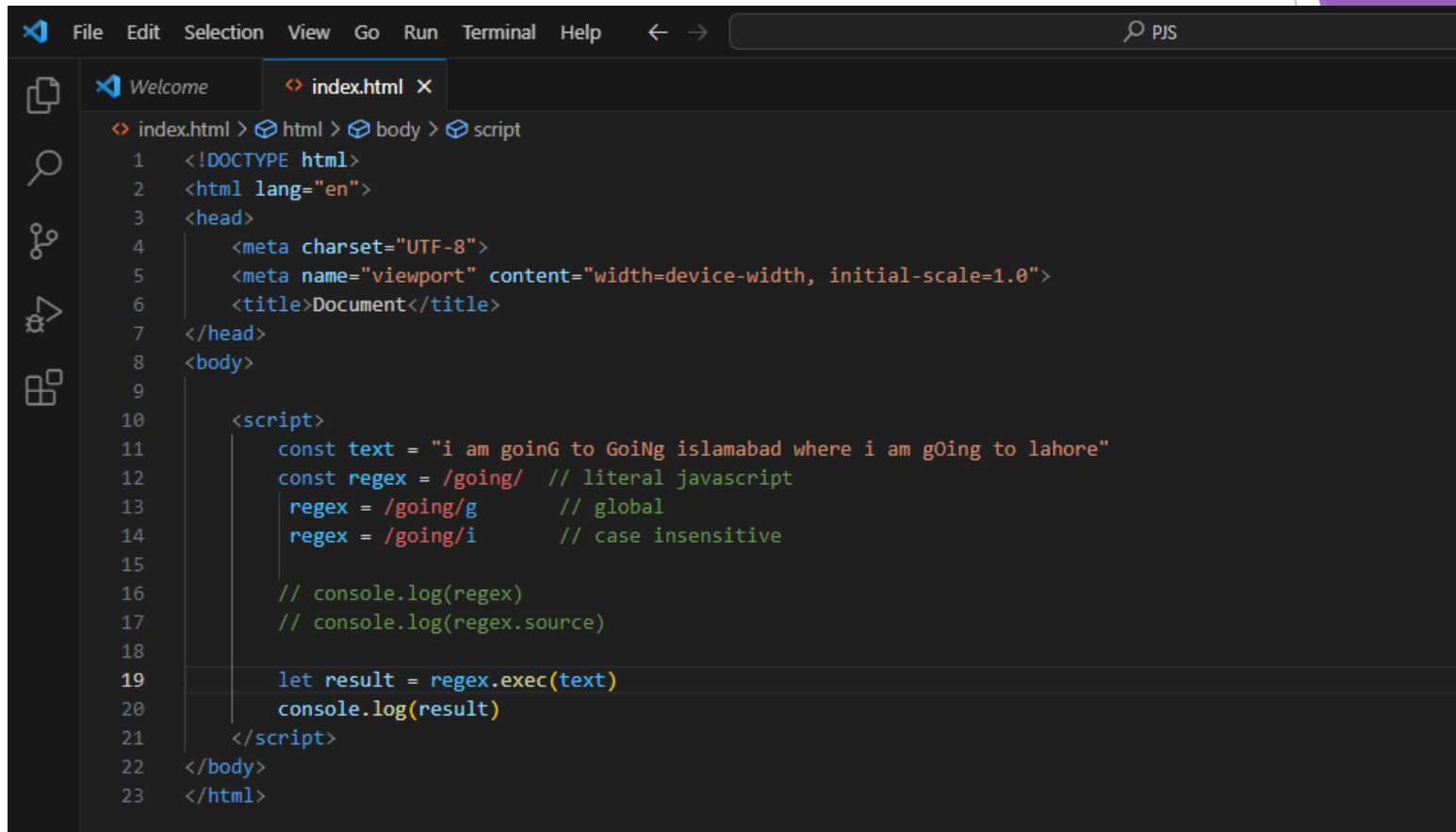


```
File Edit Selection View Go Run Terminal Help ← → PJS
Welcome index.html X
index.html > html > body > script
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9
10   <script>
11     const text = "i am going to going islamabad where i am going to lahore"
12     const regex = /going/g
13     console.log(regex)
14     console.log(regex.source)
15
16     let result = regex.exec(text)
17     console.log(result)
18     result = regex.exec(text)
19     console.log(result)
20     result = regex.exec(text)
21     console.log(result)
22   </script>
23 </body>
24 </html>
```



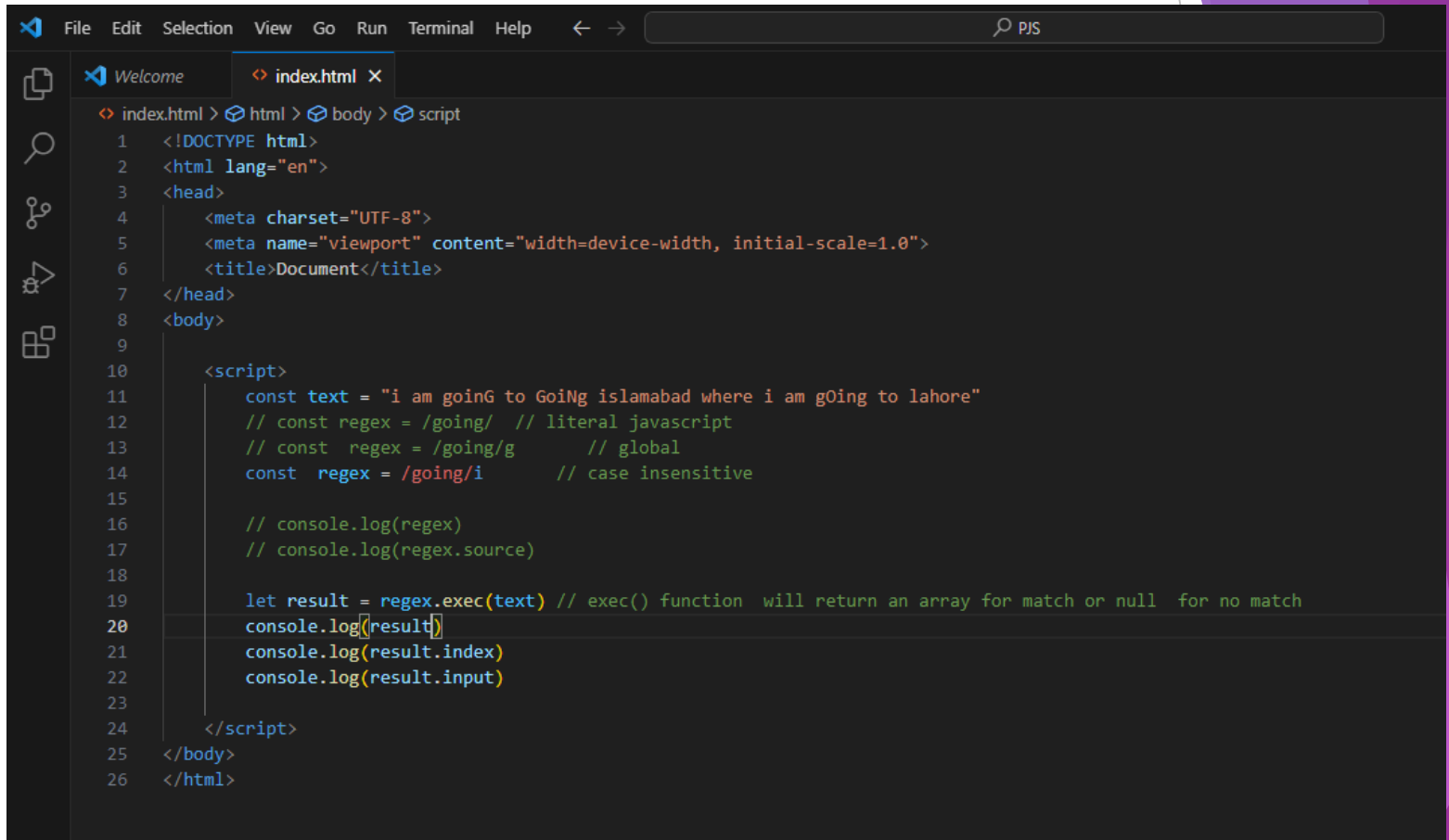
```
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder
top Filter
/going/g
going
▶ ['going', index: 5, input: 'i am going to going islamabad where i am going to lahore', groups: undefined]
▶ ['going', index: 14, input: 'i am going to going islamabad where i am going to lahore', groups: undefined]
▶ ['going', index: 41, input: 'i am going to going islamabad where i am going to lahore', groups: undefined]
>
```

Exec function with case insensitive



```
File Edit Selection View Go Run Terminal Help ← → PJS
Welcome index.html x
index.html > html > body > script
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9
10   <script>
11     const text = "i am goinG to GoiNg islamabad where i am gOing to lahore"
12     const regex = /going/ // literal javascript
13     regex = /going/g      // global
14     regex = /going/i      // case insensitive
15
16     // console.log(regex)
17     // console.log(regex.source)
18
19     let result = regex.exec(text)
20     console.log(result)
21   </script>
22 </body>
23 </html>
```

Exec function



```
File Edit Selection View Go Run Terminal Help ← → PJS
Welcome index.html X
index.html > html > body > script
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9
10   <script>
11     const text = "i am goinG to GoIng islamabad where i am gOing to lahore"
12     // const regex = /going/ // literal javascript
13     // const regex = /going/g // global
14     const regex = /going/i // case insensitive
15
16     // console.log(regex)
17     // console.log(regex.source)
18
19     let result = regex.exec(text) // exec() function will return an array for match or null for no match
20     console.log(result)
21     console.log(result.index)
22     console.log(result.input)
23
24   </script>
25 </body>
26 </html>
```


Test function

```
12 <script>
13   const text = "i am going to GoNg islamabad where i am gOing to lahore"
14   // const regex = /going/ // literal javascript
15   // const regex = /going/g // global
16   const regex = /going/i // case insensitive
17
18   // console.log(regex)
19   // console.log(regex.source)
20
21   let result = regex.exec(text) // 1. exec() function will return an array for match or null for no match
22
23   if (result) {
24     console.log(result)
25     console.log(result.index)
26     console.log(result.input)
27   }
28
29   let test = regex.test(text) // 2. test() true or false
30   console.log(test)
31
```

Replace function

```
11
12 <script>
13   const text = "i am going to GoIng islamabad where i am gOing to lahore"
14   // const regex = /going/ // literal javascript
15   // const regex = /going/g // global
16   const regex = /going/i // case insensitive
17
18   // console.log(regex)
19   // console.log(regex.source)
20
21   let result = regex.exec(text) // 1. exec() function will return an array for match or null for no match
22
23   if (result) {
24     // console.log(result)
25     // console.log(result.index)
26     // console.log(result.input)
27   }
28
29   // let testReg = regex.test(text) // 2. test() true or false
30   // console.log(testReg)
31
32   // let matchReg = text.match(regex) // 3. match() return Array result or null
33   // console.log(matchReg)
34
35   // let searchReg = text.search(regex) // 4. search() returns index else -1
36   // console.log(searchReg)
37
38   let replaceReg = text.replace(regex, 'went') // 5. replace() returns new replace
39   console.log(replaceReg)
40
```

Match function

```
let matchReg = text.match(regex)           // 3. match() return Array result or null  
console.log(matchReg)
```

Search function

```
let searchReg = text.search(regex) // 4. search() returns index else -1  
console.log(searchReg)
```

```
11
12 <script>
13   const text = "run when you have to run, but never
14   run from your responsibilities because the way you
15   run determines how far you run in life."
16   let regex = /run/
17
18   let result = regex.exec(text)
19   console.log(result)
20   if (regex.test(text)) {
21     console.log("the text matches")
22   } else {
23     console.log("the text does not match")
24   }
```

Conclusion

- ▶ - Regular expressions in JavaScript provide powerful text processing capabilities
- ▶ - They are essential for validation, searching, and replacing text
- ▶ - Mastering regex improves efficiency in programming