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

File Edit Selection View Go Run Terminal Help PJS

Welcome | index.html

index.html > html > body > script

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <script>
        const text = "i am going to going islamabad where
        const regex = /going/
        console.log(regex)
        console.log(regex.source)

        let result = regex.exec(text)
        console.log(result)
        result = regex.exec(text)
        console.log(result)
    </script>
</body>
</html>
```

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorde

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

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <script>
        const text = "i am going to going islamabad where i am going to lahore"
        const regex = /going/g
        console.log(regex)
        console.log(regex.source)

        let result = regex.exec(text)
        console.log(result)
        result = regex.exec(text)
        console.log(result)
        result = regex.exec(text)
        console.log(result)
    </script>
</body>
</html>
```

```
/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

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <script>
        const text = "i am goinG to GoiNg islamabad where i am gOing to lahore"
        const regex = /going/  // literal javascript
         regex = /going/g       // global
         regex = /going/i       // case insensitive

        // console.log(regex)
        // console.log(regex.source)

        let result = regex.exec(text)
        console.log(result)
    </script>
</body>
</html>
```

Exec function

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <script>
        const text = "i am goinG to GoiNg islamabad where i am gOing to lahore"
        // const regex = /going/  // literal javascript
        // const  regex = /going/g       // global
        const  regex = /going/i       // case insensitive

        // console.log(regex)
        // console.log(regex.source)

        let result = regex.exec(text) // exec() function  will return an array for match or null  for no match
        console.log(result)
        console.log(result.index)
        console.log(result.input)

    </script>
</body>
</html>
```

Test function

```
<script>
    const text = "i am going to GoiNg islamabad where i am gOing to lahore"
    // const regex = /going/  // literal javascript
    // const  regex = /going/g       // global
    const regex = /going/i       // case insensitive

    // console.log(regex)
    // console.log(regex.source)

    let result = regex.exec(text) // 1. exec() function  will return an array for match or null  for no match

    if (result) {
        console.log(result)
        console.log(result.index)
        console.log(result.input)
    }

    let test = regex.test(text) // 2. test() true or false
    console.log(test)

```

Replace function

```
<script>
    const text = "i am going to GoiNg islamabad where i am gOing to lahore"
    // const regex = /going/  // literal javascript
    // const  regex = /going/g       // global
    const regex = /going/i       // case insensitive

    // console.log(regex)
    // console.log(regex.source)

    let result = regex.exec(text)       // 1. exec() function  will return an array for match or null  for no match

    if (result) {
        // console.log(result)
        // console.log(result.index)
        // console.log(result.input)
    }

    // let testReg  = regex.test(text)     // 2. test() true or false
    // console.log(testReg)

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

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

    let replaceReg = text.replace(regex, 'went')    // 5. replace() returns  new replace
    console.log(replaceReg)
```

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)
```

```
<script>
    const text = "run when you have to run, but never
    run from your responsibilities because the way you
    run determines how far you run in life."
     let regex = /run/

     let result = regex.exec(text)
     console.log(result)
  if (regex.test(text)) {
    console.log("the text matches")
  } else {
    console.log("the text does not match")

  }
```

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