JavaScript Regular Expressions (Regex) - Detailed Notes

---

1. What is Regex?

- A regular expression (regex) is a sequence of characters that forms a search pattern.
- It can be used to check if a string contains a certain pattern, extract data, replace text, etc.

---

2. Creating Regex Patterns

Two ways to create a regex:
1. Literal: let pattern = /hello/;
2. Constructor: let pattern = new RegExp("hello");

---

3. Common Regex Methods

// test()
- Used on pattern. Returns true or false.
Example: /abc/.test("abcdef") → true

// match()
- Used on string. Returns all matches (if 'g' is used), else first match info.
Example: "hello hello".match(/hello/g) → ["hello", "hello"]

// exec()
- Used on pattern. Returns match object or null. Use multiple times to iterate.
Example:
let pattern = /hello/g;
pattern.exec("hello hello"); // run multiple times

// replace()
- Replaces matched substrings.
Example: "abc abc".replace(/abc/g, "xyz") → "xyz xyz"

// search()
- Returns the index of the match, else -1.

Example: "abcde".search(/cd/) → 2

// split()
- Splits string using regex delimiter.
Example: "a,b,c".split(/,/) → ["a", "b", "c"]

---

4. Flags

- g → Global match (all occurrences)
- i → Case-insensitive match
- m → Multiline mode

---

5. Anchors

- ^ → Start of string or line
- $ → End of string or line
- \b → Word boundary (match at word edges)
- \B → Non-word boundary

---

6. Quantifiers

- * → 0 or more times
- + → 1 or more times
- ? → 0 or 1 time
- {n} → Exactly n times
- {n,} → At least n times
- {n,m} → Between n and m times

---

7. Character Sets

- [abc] → a, b, or c
- [^abc] → Not a, b, or c
- [a-z] → Any lowercase letter
- [A-Z] → Any uppercase letter

- [0-9] → Any digit

---

8. Predefined Character Classes

- \d → Digit (0-9)
- \D → Non-digit
- \w → Word character (a-z, A-Z, 0-9, _)
- \W → Non-word character
- \s → Whitespace (space, tab, newline)
- \S → Non-whitespace

---

9. Dot Character

- . → Matches any single character except newline

---

10. Useful Regex Patterns

- Email: /^\w+@\w+\.\w+$/
- Phone: /^\d{10}$/
- Username: /^[a-zA-Z0-9]{6,12}$/
- Only Digits: /^\d+$/
- Only Letters: /^[a-zA-Z]+$/

---

11. Word Boundary Examples

- \bthe\b → Matches "the" only if it's a full word
- /\Bend\B/ → Match "end" not at word boundaries

---

12. Real Examples from Code

```
let text = "gaurav@gmail.com";
let pattern = /^\w+@\w+\.\w+$/;
pattern.test(text); // true
```

```
let phone = "9876543210";
let pattern = /^\d{10}$/;
pattern.test(phone); // true

let username = "gaurav123";
let pattern = /^[a-zA-Z0-9]{6,}$/;
pattern.test(username); // true
```

---

13. Tips

- Always use \ to escape special characters in JS strings.
- Use 'g' with exec() in loops to find all matches.
- Use \b for exact word matching.