Instructor Notes:

Instructor Notes:

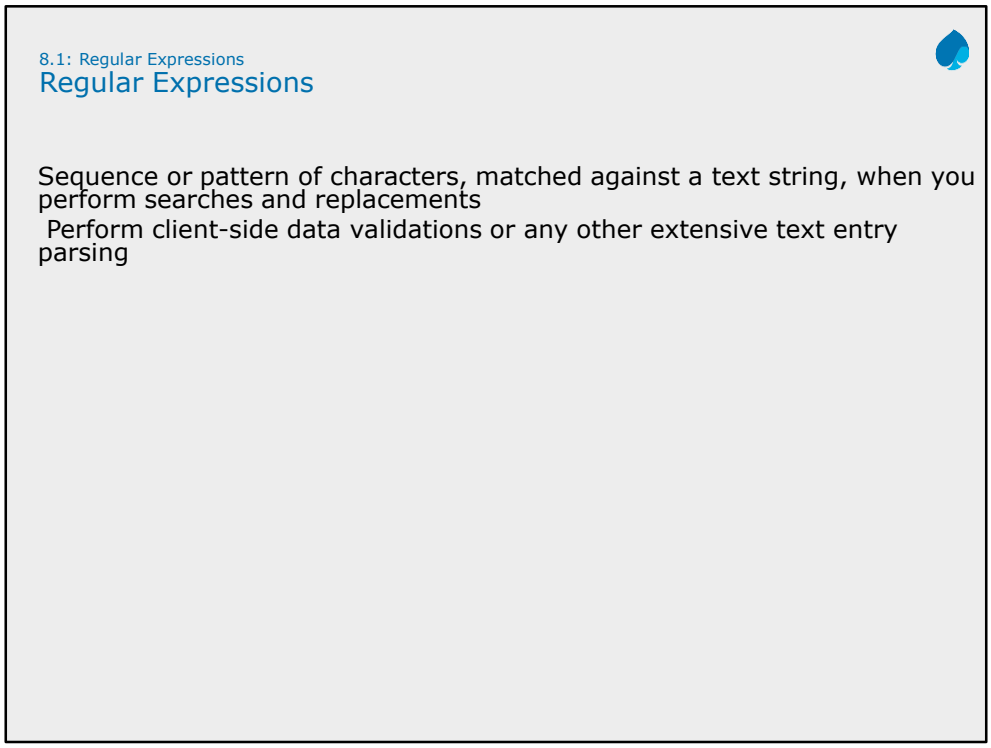
Lesson Objectives

After completing this lesson, you will be able to:

- Use regular expressions
- Search text using simple patterns and special characters

Work with RegExp objects

Instructor Notes:



Working with Regular Expressions

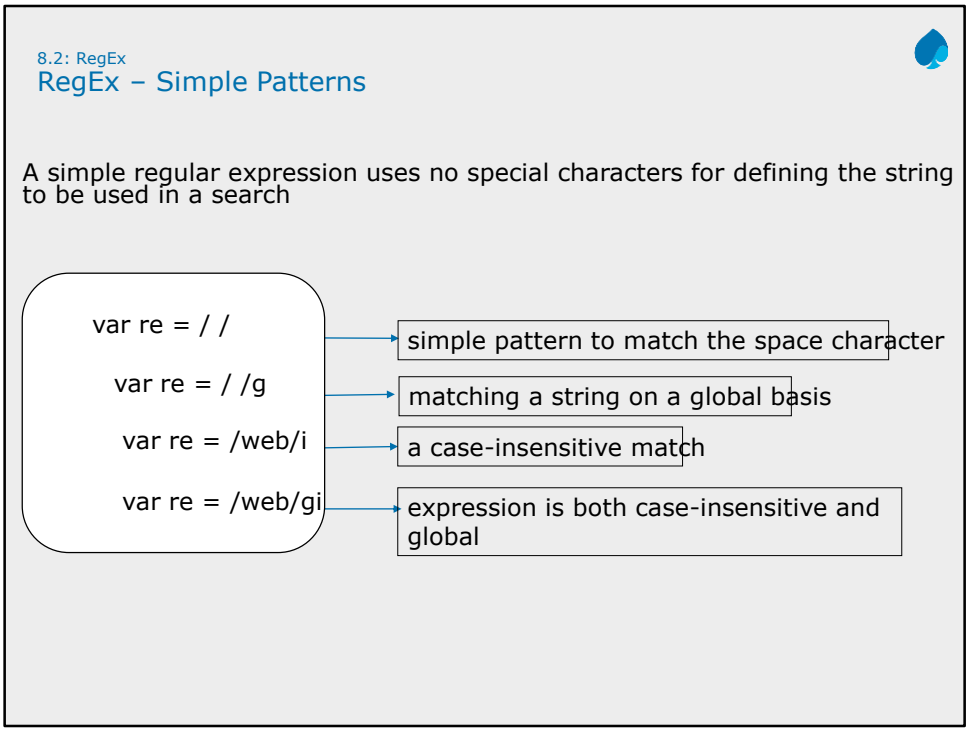
If your scripts perform client-side data validations or any other extensive text entry parsing, then you can use regular expressions, rather than cobbling together comparatively complex JavaScript functions to perform the same tasks.

JavaScript treats regular expressions as objects and distinguishes between them and the *RegExp* constructor.

To cover the depth of the regular expression syntax, we need to study the following:

- Simple expressions
- Range of special characters used to define specifications for search strings
- Introduction to the usage of parentheses in the language:
  - Group expressions to influence calculation precedence
  - Temporarily store intermediate results of more complex expressions for use in reconstructing strings after their dissection by the regular expression.

Instructor Notes:



Simple Patterns
A simple regular expression uses no special characters to define the string to use in a search. Therefore, if you wish to replace every space in a string with an underscore character, the simple pattern to match the space character is: var re = / /

A space appears between the regular expression start-end forward slashes. The problem with this expression, however, is that it knows only how to find a single instance of a space in a long string. Regular expressions can be instructed to apply the matching string on a global basis by appending the g modifier: var re = / /g

Regular expression matching — like a lot of other aspects of JavaScript — is case-sensitive. But you can override this behavior by using one other modifier that lets you specify a case-insensitive match. Therefore, the following expression, var re = /web/I, finds a match for “web,” “Web,” or any combination of upper and lowercase letters in the word. You can combine the two modifiers together at the end of a regular expression. For example, the following expression is both case-insensitive and global in scope: var re = /web/gi

Instructor Notes:

8.2: RegEx

RegEx – Special Characters

\b Word Boundary:
- Get a match at the beginning or end of a word in the string
- /\bor/ matches "origami" and "or" but not "normal".
- /or\b/ matches "traitor" and "or" but not "perform"
- /\bor\b/ matches full word "or" and nothing else

\B Word Non-Boundary:
- Get a match when it is not at the beginning or end of a word in the string
- /\Bor/ matches "normal" but not "origami"
- /or\B/ matches "normal" and "origami" but not "traitor"
- /\Bor\B/ matches "normal" but not "origami" or "traitor

Special Characters

The regular expression in JavaScript borrows most of its vocabulary from the Perl regular expression. In a few instances, JavaScript offers alternatives to simplify the syntax, and accepts their Perl version for developers with experience in that technology.

Instructor Notes:

8.2: RegEx

RegEx – Special Characters (Contd.)

\d Numeral:
- Find any single digit 0 through 9
  - /\d\d\d/ matches "212" and "415" but not "B17"

\D Non-numeral:
- Find any non-digit
  - /\D\D\D/ matches "ABC" but not "212" or "B17"

\s Single White Space:
- Find any single space character
  - /over\sbite/ matches "over bite" but not "overbite" or "over bite"

Instructor Notes:

8.2: RegEx

RegEx – Special Characters (Contd.)

\S Single Non-White Space:
- /over\Sbite/ matches "over-bite" but not "overbite" or "over bite"

\w Letter, Numeral, or Underscore:
- /A\w/ matches "A1" and "AA" but not "A+"

\W Not letter, Numeral, or Underscore:
- /A\W/ matches "A+" but not "A1" and "AA"

Instructor Notes:

8.2: RegEx

RegEx – Special Characters (Contd.)

"." Any Character Except Newline:
- /…/ matches "ABC", "1+3", "A 3" or any 3 characters

[…] Character Set:
- Finds any character in the specified character set
  - /[AN]BC/ matches "ABC" and "NBC"

[^…] Negated Character Set:
- Find any character not in the specified character set
  - /[^AN]BC/ matches "BBC" and "CBC" but not "ABC" or "NBC"

Instructor Notes:

8.2: RegEx

RegEx – Counting Metacharacters

"*" - Zero or More Times:

- /Ja*vaScript/ matches "JvaScript", "JavaScript", and "JaaavaScript" but not "JovaScript"

"?" - Zero or One Time:

- /Ja?vaScript/ matches "JvaScript" or "JavaScript" but not "JaaavaScript"

"+" - One or More Times:

- /Ja+vaScript/ matches "JavaScript" or "JaavaScript" but not "JvaScript"

Instructor Notes:

8.2: RegEx

RegEx – Counting Metacharacters (Contd.)

{n} - Exactly n Times:
- /Ja{2}vaScript/ matches "JaavaScript" but not "JvaScript" or "JavaScript"

{n,} - N or More Times:
- /Ja{2,}vaScript/ matches "JaavaScript" or "JaaavaScript" but not "JavaScript"

{n,m} - At Least n, At Most m Times:
- /Ja{2,3}vaScript/ matches "JaavaScript" or "JaaavaScript" but not "JavaScript"

Instructor Notes:

8.2: RegEx

RegEx – Positional Metacharacters

"^" - At the beginning of a string or line
- /^Fred/ matches "Fred is OK" but not "I'm with Fred" or "Is Fred here?"

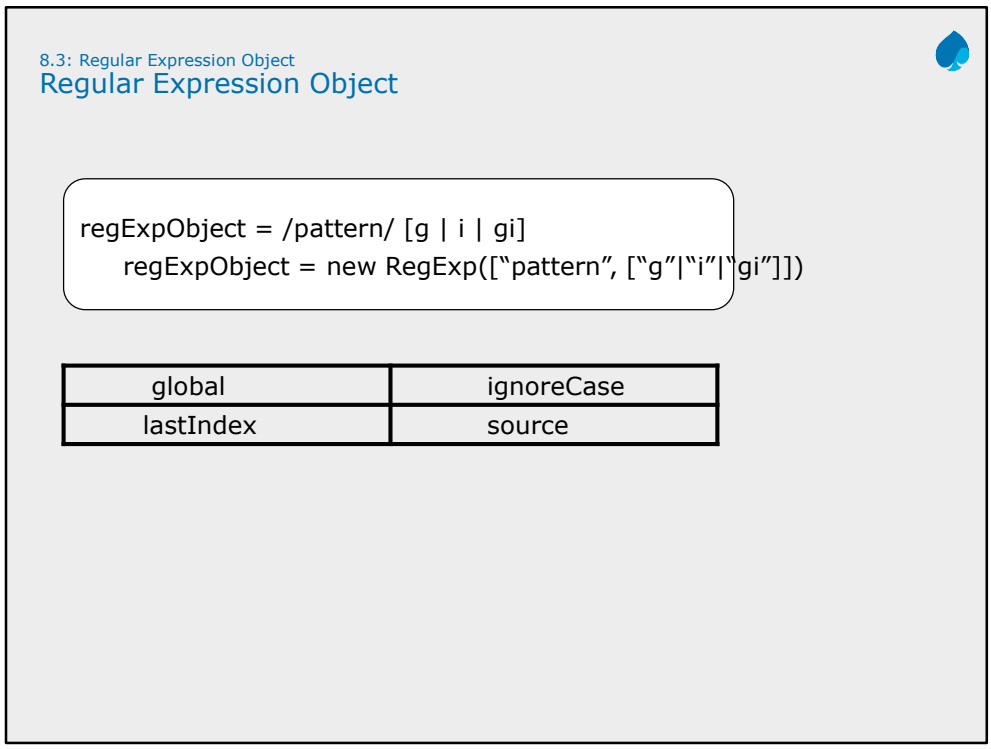
"$" - At the end of a string or line
- /Fred$/ matches "I'm with Fred" but not "Fred is OK" or "Is Fred here?"

Every metacharacter applies to the character immediately preceding it in the regular expression. Preceding characters might also be matching *metacharacters*. For example, a match occurs for the following expression if the string contains two digits separated by one or more vowels:

```
/\d[aeiouy]+\d/
```

The last major contribution of metacharacters is to help regular expressions search a particular position in a string. Following table shows positional metacharacters:

Instructor Notes:



Regular Expression Object

Each regular expression object contains its own pattern and other properties. To decide which object creation style to use depends on the way the regular expression is to be used in your scripts.

Syntax

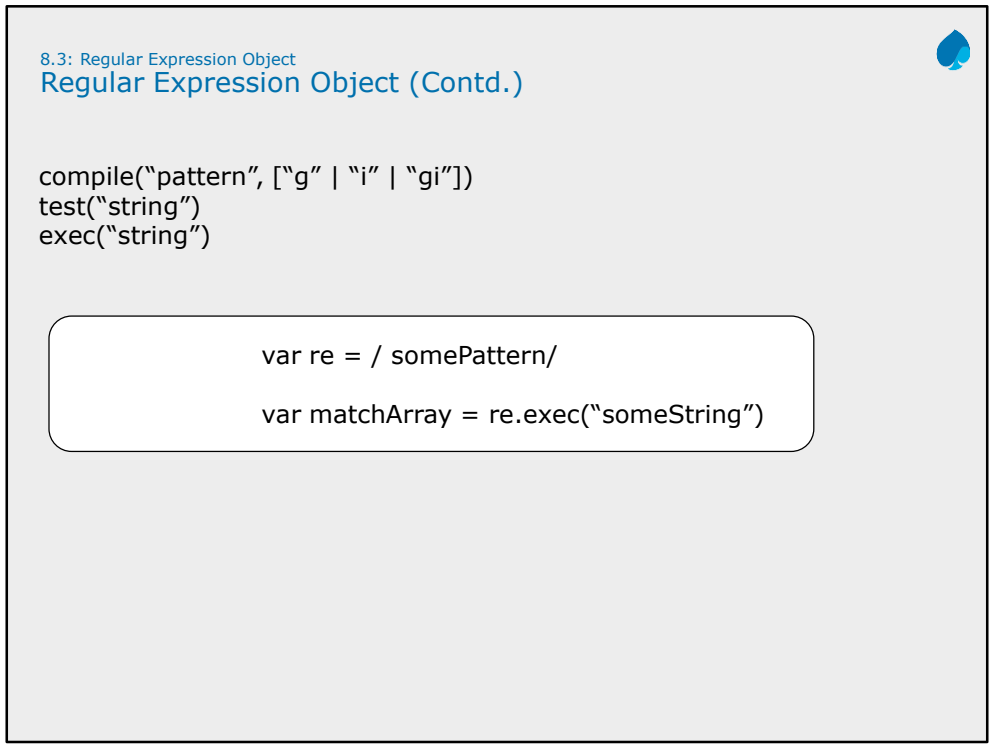
Regular expression creation:

```
regularExpressionObject = / pattern/ [g | i | gi]
regularExpressionObject = new RegExp([“ pattern”, [“g” |
“i” | “gi”]])
```

Access Regular Expression Properties or Methods:

```
regularExpressionObject.property | method([ parameters])
```

Instructor Notes:



Properties & Methods of Regular Expression object:
global : Specifies if the modifier "g" is set
ignoreCase : Specified if the modifier "I" is set
lastIndex : Specifies the index position from where to start the next match.
source **:** The source property is simply the string representation of the regular expression used to define the object. This property is read-only.

compile(" pattern", ["g" | "i" | "gi"])
Use the *compile()* method to compile on the fly, a regular expression whose content changes continually during script execution. Other regular expression creation statements (literal notation and the new *RegExp()* constructor passing a regular expression) automatically compile their expressions.

Instructor Notes:

Demo

Test_compiler.html
DemoRegExp.html

Add the notes here.

Instructor Notes:

Lab

Lab Exercise 9:

- Regular Expressions in JavaScript

Add the notes here.

Instructor Notes:

Summary

For client-side data validation we can use a regular expression

Regular expression object describes a pattern of characters

Simple regular expressions use no special characters used to match the space in a string with an underscore character

Regular Expressions use special characters such as \b, \d, \w etc

From this chapter, you know how to:

- Use Regular Expressions
- Search using Simple patterns
- Search using Special characters
- Work with RegExp Objects

Instructor Notes:

Review Questions

Question 1: The _____ property is the main string against which a regular expression is compared in search of a match.

- Option 1: RegExp.input
- Option 2: RegExp.inp
- Option 3: RegExpr.input

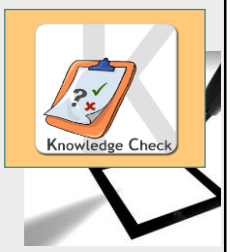
Question 2: Index property indicates the index counter of the main string to be searched against the current regular expression object.

- True / False

Question 3: Use the _____ method to compile on the fly a regular expression whose content changes continually during the execution of a script.

Instructor Notes:

Match the Following



1 . \b	a. Word non-boundary
2. \B	b. Word boundary
3 . \d	c. Numeral
4 . \s	d. Single non-white space
5 . \S	e. Single white space