Strings and RegExp

String Operations and Regular Expressions

SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Content

1. Strings
   - Definition
   - Comparing strings
   - Methods
2. RegExp
   - Definition
   - Patterns
   - Methods

Have a Question?

SoftUni Foundation

[sli.do](sli.do)

#JS-CORE

Strings in JavaScript

String Operations and Methods

What is a String?

- JavaScript strings are used for **storing** and **manipulating** text

```
let str = "Hello, World!";
```

- You can use the **+** operator to append multiple strings together:

```
let longString = "This is a very long string" +
                 "to wrap across multiple lines" +
                 "otherwise my code is unreadable."
```

Quotes in Strings

- Unlike some other languages, JavaScript makes **no distinction** between **single-quoted** strings and **double-quoted** strings.

```
let carName = "Volvo XC60"; // Double quotes
let carName = 'Volvo XC60'; // Single quotes
```

- Quotes can be used inside a string, as long as they don't match the quotes surrounding the strings:

```
let str1 = "It's alright";
let str2 = "He is called 'Johnny'";
let str3 = 'He is called "Johnny"';
```

Length and Special Characters

- The length of a string is found in the built in property **length**:

```
let myStr = "Find my length.";
let length = myStr.length; // 15
```

- **Special chars** can be encoded using **escape notation**:

Code	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

Escape Sequences

Code	Result
\b	Backspace
\f	Form Feed
\n	New Line
\r	Carriage Return
\t	Horizontal Tabulator
\v	Vertical Tabulator

```
let example = "This is an example \nfor a new line.";
// This is an example
// for a new line.
```

Comparing Strings

- Equality - "**==**" - True if **operands** are the same, otherwise false.

```
let sVal = "example";
if (sVal == "example") // true
```

- Strict Equality - "**===**" - True if **operands** and **data type** are the same, otherwise false.

```
let sVal = "example";
let sVal2 = new String("example");
if (sVal === sVal2) // not true
```

Comparing Strings

- Inequality - "!=" - True if operands are not the same, otherwise false

```
let firstStr = "9900";
let firstNum = 9900
if (firstStr != secondStr) // false
```

- Strict Inequality - "!==“ - True if operands and data type are not the same, otherwise false

```
let firstStr = "9900";
let firstNum = 9900
if (firstStr !== secondStr) // true
```

Comparing Strings (2)

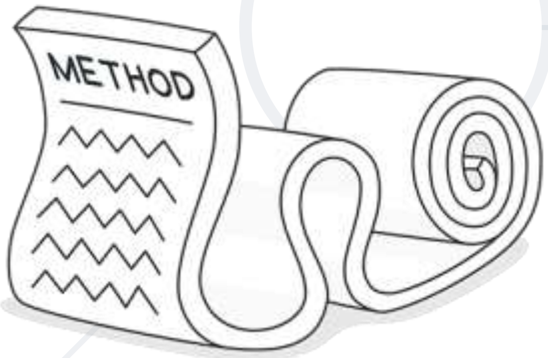
- Greater than - ">" (Greater than or equal - ">=")
  - True if first operand is greater than (or equal) to the second one.
- Less than - "<" (Less than or equal - "<=")
  - True if second operand is greater than (or equal) to the first one.

```
if ("b" > "a") // true
```

```
if ('Example of a long string' <= 'A short one') // false
```

String Methods

- **indexOf()** - returns the position of the first found occurrence of a specified value in a string
- **lastIndexOf()** - returns the position of the last found occurrence of a specified value in a string
- **search()** - searches a string for a specified value



String Methods (2)

- **slice()** - extracts a part of a string and returns a new one.
- **substring()** - extracts the characters from a string between two specified indices.
- **substr()** - extracts the characters from a string, beginning at a specified start position and through the specified length.

String Methods: Examples

```
let str = "JavaScript is fun!";
console.log(str.indexOf("JavaScript")); // 0
console.log(str.indexOf("java")); // -1
```

```
let str = "I am JavaScript developer";
let sub = str.substr(5);   // substr(start, length)
console.log(sub);          // JavaScript developer
```

```
let str = "I am JavaScript developer";
let sub = str.substring(5, 9); // startIndex, endIndex
console.log(sub);             // Java
```

String Methods: Examples

- Accessing elements like an Array

```
let str = "JavaScript is fun!";
let letter = str[0];
console.log(letter); // J
```

```
let str = "JavaScript is fun!";
let letter = str.charAt(0);
console.log(letter); // J
```

- Converting string to an array with the split method

```
let str = "I like JS";
let tokens = str.split(' ');
console.log(tokens); // ["I", "like", "", "", "", "JS"]
tokens = tokens.filter(s => s!='');
console.log(tokens.join(' ')); // I like JS
```

Problem: Pascal or Camel Case

- Convert the first string to either "Pascal Case" or "Camel Case".

```
function solve() {
  let input = document.getElementById("str1").value;
  let currentCase = document.getElementById("str2").value;
  function pascalOrCamelCase(input, currentCase) {
    let split = input.toLowerCase().split(' ').filter(a => a !== '');
    let output = "";
    if (currentCase === "Pascal Case") {
      for (let word of split) {
        if (word[0] !== word[0].toUpperCase()) {
          word = word.replace(word[0], word[0].toUpperCase())
        }
        output += word;
      }
    }
// Continues on the next slide
```

Problem: Pascal or Camel Case

```
else if (currentCase === "Camel Case") {
      for (let word of split) {
        if (word[0] !== word[0].toUpperCase()) {
          word = word.replace(word[0], word[0].toUpperCase())
        }
        output += word;
      }
      output = output.replace(output[0], output[0].toLowerCase());
    } else {
      output = "Error!";
    }

    document.getElementById("result").innerHTML = output;
  }
  pascalOrCamelCase(input, currentCase);
}
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1476#0

Problem: Find ASCII Equivalent

If the current element of the string is of type number, print its **ASCII** char equivalent. Else, print the corresponding **ASCII** number.

```
function solve() {
  let input = document.getElementById("str").value;
  let result = document.getElementById('result');

  function findAsciiEquivalent(input) {
    let split = input.split(' ').filter(a => a !== '');
    let output = "";
    for (let element of split) {
      if (Number(element)) {
        output += (String.fromCharCode(element));
      }
// Continues on the next slide
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1476#1

Problem: Find ASCII Equivalent

```
      else {
        let charToNum = [];
        for (let i = 0; i < element.length; i++) {
           charToNum.push(element[i].charCodeAt(0))
        }
        let p = document.createElement('p');
        p.innerHTML = charToNum.join(' ')
        result.appendChild(p);
      }
    }
    let p = document.createElement('p');
    p.innerHTML = output;
    result.appendChild(p);
  }
  findAsciiEquivalent(input);
}
```

Problem: Split String Equally

You will receive a **string** and a positive integer (**bigger than 0!**). Split the string into **equal sequences** by the number you received.

```
function solve() {
   let string = document.getElementById("str").value;
   let n = parseInt(document.getElementById("num").value);
   function splitStringEqually(string, n) {
     let arr = [];
     let indexCounter = 0;
     if (string.length % n !== 0) {
       let len = string.length;
       let symbolsCount = 0;

// Continues on the next slide
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1476#2

Problem: Split String Equally

```
      while (len % n !== 0) {
        len %= n;
        len++;
        symbolsCount++;
      }
      for (let i = 0; i < symbolsCount; i++) {
        string += string[indexCounter];
        indexCounter++;
      }
    }
    for (let i = 0; i < string.length; i += n) {
      arr.push(string.substr(i, n));
    }
    document.getElementById("result").innerHTML = arr.join(' ');
  }
  splitStringEqually(string, n);
}
```

Regular Expressions

The Beauty of Modern String Processing

What are Regular Expressions?

Match text by pattern

- RegExp in **string methods**

```
let str = "RegExp Example";
let search = str.search(/RegExp/i) // 0
```

/i -> makes the regex match case **insensitive**

```
let str = "Java Regex Example Java";
let search = str.replace(/Java/g, "JavaScript");
// JavaScript RegExp Example JavaScript
```

/g -> replaces all matches

Problem: Replace a Certain Word

- For each string in the array, **replace** the necessary word with the given one.

```
function solve() {
  let arr = JSON.parse(document.getElementById("arr").value);
  let word = document.getElementById("str").value;
  let result = document.getElementById('result');
  function replaceCertainWord(arr, word) {
    let wordToReplace = arr[0].split(' ').filter(a => a !== '')[2];
    let regex = new RegExp(wordToReplace, 'gi');
    for (let sentence of arr) {
      sentence = sentence.replace(regex, word);
      let p = document.createElement('p');
      p.innerHTML = sentence
      result.appendChild(p);
    }
  }
  replaceCertainWord(arr, word);
}
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1476#3

Patterns

Patterns are defined by special syntax, e.g.

- `[0-9]+` matches non-empty sequence of digits
- `[A-Z][a-z]*` matches a capital + small letters
- `\s+` matches whitespace (non-empty)
- `\S+` matches non-whitespace
- `[0-9]{3,6}` – matches 3-6 digits
- `\d+ matches digits`
- `\D+` matches non-digits
- `\w+ matches letters`
- `\W+` matches non-letters

RegEx Brackets

[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)
(x|y)	Find any of the alternatives specified

Quantifiers

- **n+** - matches any string that contains **at least one** n
- **n*** - matches any string that contains **zero or more** occurrences of n
- **n?** - matches any string that contains **zero or one** occurrences of n
- **n{X}** - matches any string that contains **a sequence of X** n's
- **n{X,Y}** - matches any string that contains **a sequence of X to Y** n's
- **n{X,}** - matches any string that contains a **sequence of at least X** n's
- **n$** - matches any string with n **at the end** of it
- **^n** – matches any string with n **at the beginning** of it

RegEx Methods

- **exec()**

```
let namePattern = (/[A-Z][a-z]+/g);
let names = "Test Testov, example, Example";
let match;
while(match = namePattern.exec(names)) {   // Test
     console.log(match[0]);                  // Testov
}                                            // Example
```

- **test()** – returns **true** or **false**

```
let pattern = (/[0-9]+/g);
let str = "Test Testov";
console.log(pattern.test(str)); // false
```

Problem: Extract User Data

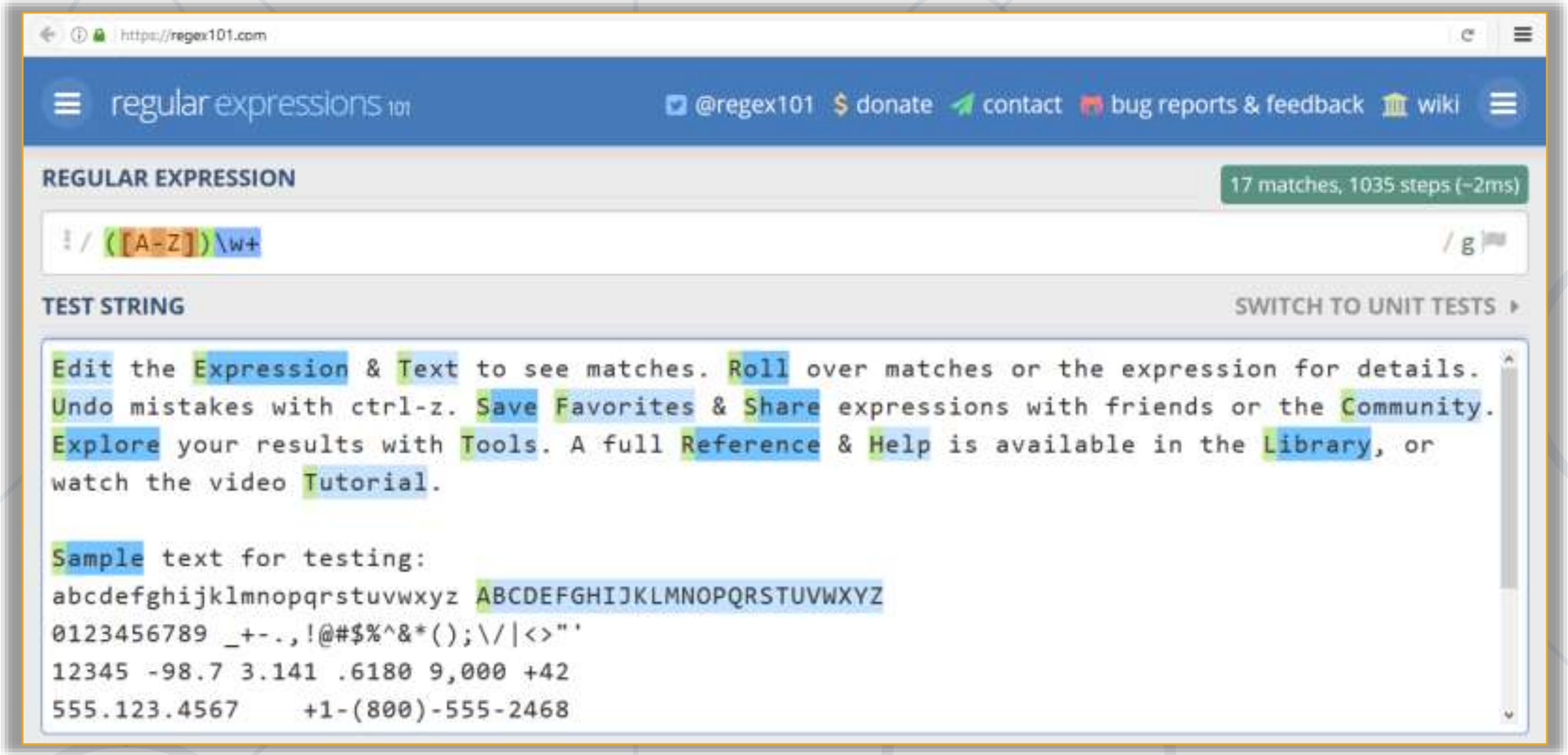
- You have to extract all valid user data from each string

```
function extractUserData() {
    let arr = JSON.parse(document.getElementById("arr").value);
    let result = document.getElementById('result');
    function userData(arr) {
        let pattern =
            /^([A-Z][a-z]* [A-Z][a-z]*) (\+359 [0-9] [0-9]{3} [0-9]{3}|\+359-[0-9]-[0-9]{3}-
[0-9]{3}) ([a-z0-9]+@[a-z]+\.[a-z]{2,3})$/;
        let match;
        for (let data of arr) {
            match = pattern.exec(data);
            if (match) {
                let firstParagraph = document.createElement('p');
                firstParagraph.textContent = `Name: ${match[1]}`;
                result.appendChild(firstParagraph);           // Continues on the next slide
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1476#4

Problem: Extract User Data

```
                let secondParagraph = document.createElement('p');
                secondParagraph.textContent = `Phone Number: ${match[2]}`;
                result.appendChild(secondParagraph);
                let thirdParagraph = document.createElement('p');
                thirdParagraph.textContent = `Email: ${match[3]}`;
                result.appendChild(thirdParagraph);
            } else {
                let errorParagraph = document.createElement('p');
                errorParagraph.textContent = 'Invalid data';
                result.appendChild(errorParagraph);
            }
            let dashes = document.createElement('p');
            dashes.textContent = '- - -';
            result.appendChild(dashes);
        }
    }
    userData(arr);
}
```



REGULAR EXPRESSION

17 matches, 1035 steps (~2ms)

```
/ ([A-Z])\w+ / g
```

TEST STRING

SWITCH TO UNIT TESTS

```
Edit the Expression & Text to see matches. Roll over matches or the expression for details.
Undo mistakes with ctrl-z. Save Favorites & Share expressions with friends or the Community.
Explore your results with Tools. A full Reference & Help is available in the Library, or
watch the video Tutorial.

Sample text for testing:
abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789 _+-.,!@#$%^&*();\/|<>"'
12345 -98.7 3.141 .6180 9,000 +42
555.123.4567    +1-(800)-555-2468
```

Live Demo

www.regex101.com

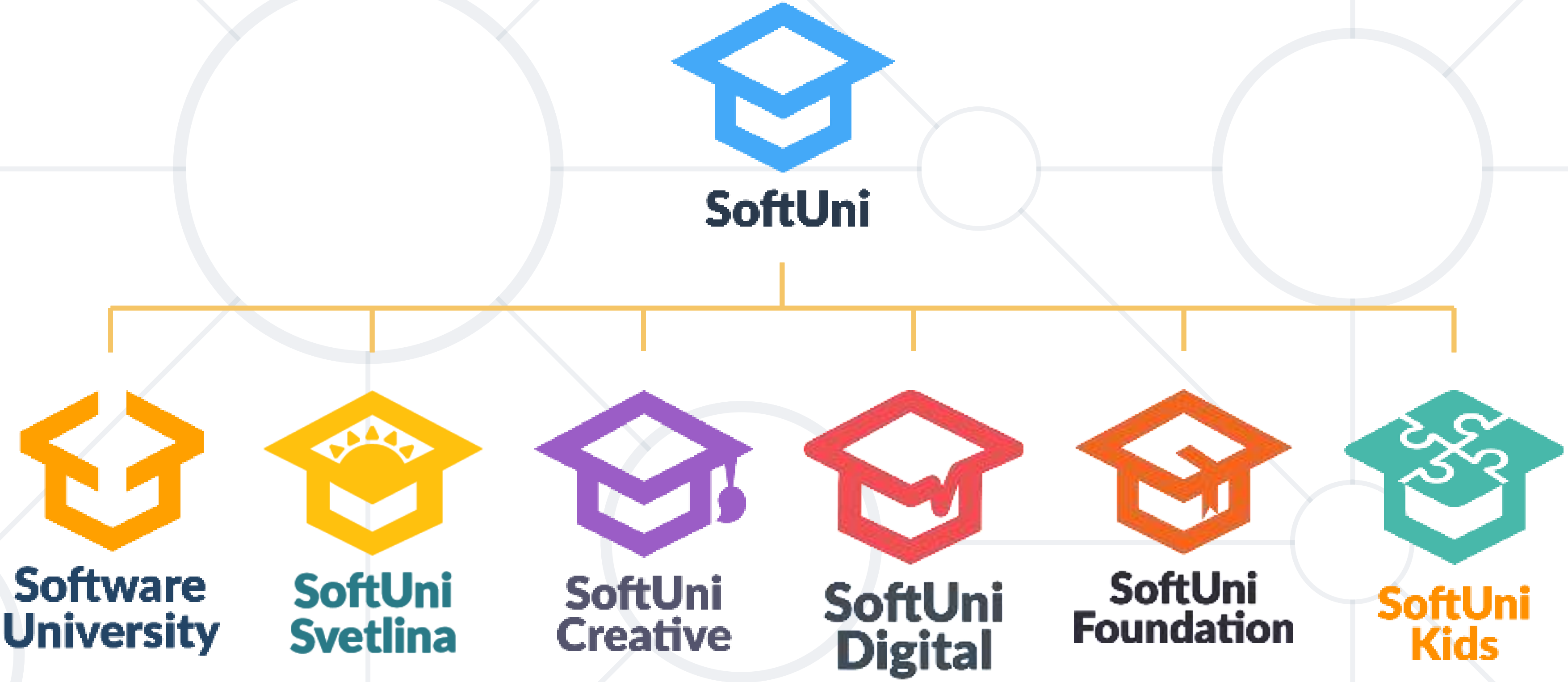
Live Exercises

Summary

SoftUni Foundation

- **Strings** in JavaScript
- String **Methods**: **split()**, **substring()**, **indexOf()**, **trim()**, **replace()** . . .
- Regular expressions match text by **pattern**
- Regex methods and quantifiers

Questions?



SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

XSsoftware

SBTech
we know sports

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

Postbank
Решения за твоето утре

SUPER HOSTING
®.BG

INDEAVR
Serving the high achievers

INFRAGISTICS®

LIEBHERR

æternity

SoftUni Organizational Partners

IS ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license