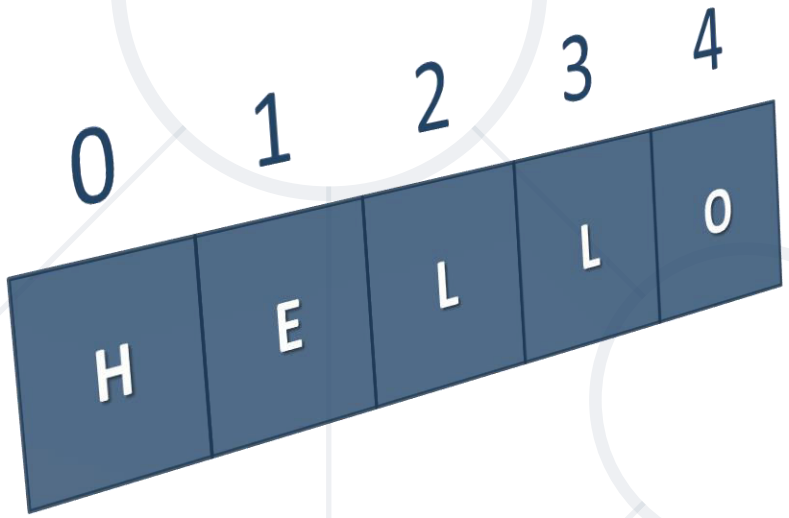
Strings and Regular Expressions

String Operations and Regular Expressions



SoftUni Team

Technical Trainers

Software University

http://softuni.bg

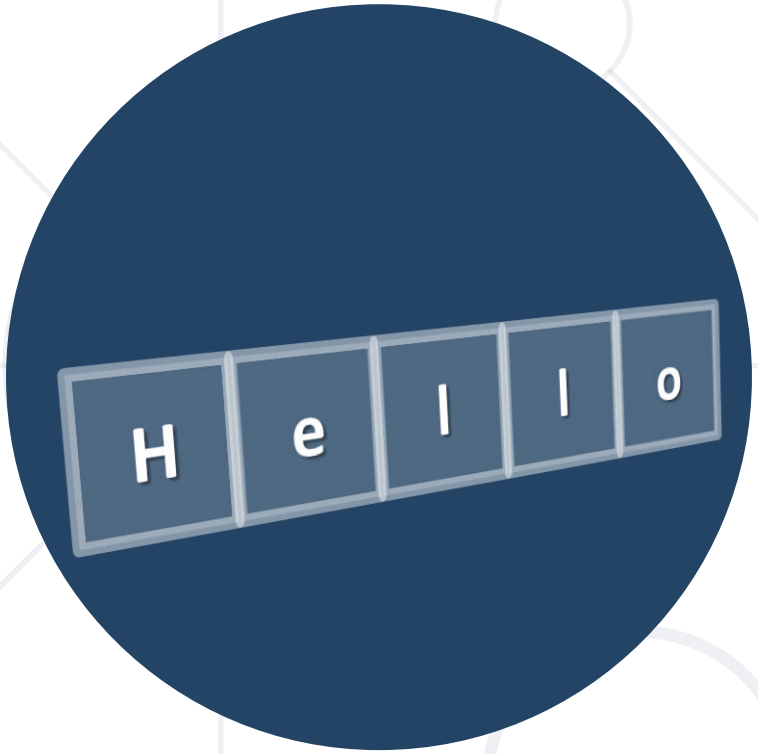
Questions?

SoftUni Foundation

sli.do

#JSCORE

Table of Contents



Strings in JavaScript

What is String?

Strings in JavaScript

- **Strings** in JS hold a sequence of **Unicode** characters
  - **Immutable** by design → cannot be changed
  - Like arrays have **length** and provide access by index **[ ]**

```
let str1 = "Text in double quotes";
let str2 = 'Text in single quotes';
let str = str1 + ' ' + str2;
for (let i = 0; i < str.length; i++) {
  console.log(`${i} -> ${str[i]}`);
}
```

Problem: Print String Letters

- Read a **string** and **print its letters** as shown below

```
SoftUni
```

```
str[0] -> S
str[1] -> o
str[2] -> f
str[3] -> t
str[4] -> U
str[5] -> n
str[6] -> i
```

```
function printStringLetters(str) {

  for (let i in str)
    console.log(
      `str[${i}] -> ${str[i]}`);
}

printStringLetters('Hello');
printStringLetters('SoftUni');
```

Check your solution here: https://judge.softuni.bg/Contests/312

Problem: Concatenate and Reverse Strings

SoftUni Foundation

- Read an **array of strings**, **concatenate** them and **reverse** them

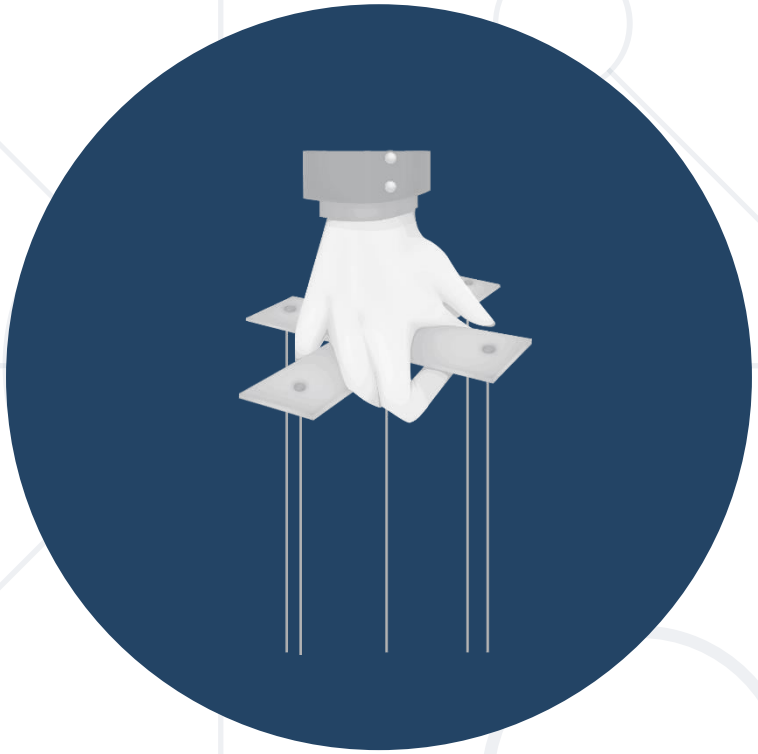
```
I
am
student
```

```
tnedutsmaI
```

```
function concatenateAndReverse(arr) {
  let allStrings = arr.join('');
  let chars = Array.from(allStrings);
  let revChars = chars.reverse();
  let revStr = revChars.join('');
  return revStr;
}
```

```
concatenateAndReverse(['I', 'am', 'student'])
```

Check your solution here: https://judge.softuni.bg/Contests/312



String Operations

Substring, Split, Join, IndexOf, …

String Operations: Index-Of / Sub-String

SoftUni Foundation

```
let str = "I am JavaScript developer";
console.log(str.indexOf("Java"));      // 5
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 Operations: Split / Replace

SoftUni Foundation

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

```
let s = "I like JS. JS is cool";
console.log(s.replace('JS',"C#")); // I like C#. JS is cool
console.log(s.replace(/JS/g,"C#"));// I like C#. C# is cool
```

Problem: Count Occurrences

- Count the number of times a string occurs in a text

```
the
```

```
the quick brown fox jumps over the lazy dog
```

```
2
```

```
function countStringInText(str, text) {
  let count = 0;
  let index = text.indexOf(str);
  while (index > -1) {
    count++;
    index = text.indexOf(str, index + 1);
  }
  return count;
}
```

```
countStringInText('am', 'I am cool. Bam') // 2
```

Check your solution here: https://judge.softuni.bg/Contests/312

Problem: Extract Text from Parentheses

- Extract all **text snippets** between **parentheses**
  - Parentheses cannot be nested

Rakiya (Bulgarian brandy) is home-made liquor (alcoholic drink). It can be made of grapes, plums or other fruits (even apples).

Bulgarian brandy,

alcoholic drink,

even apples

Solution: Extract Text from Parentheses

```
function extractTextFromParenthesis(text) {
  let result = [];
  let startIndex = text.indexOf('(');
  let endIndex = text.indexOf(')', startIndex);
  while (startIndex > -1 && endIndex > -1) {
    let snippet = text.substring(startIndex + 1, endIndex);
    result.push(snippet);
    startIndex = text.indexOf('(', endIndex);
    endIndex = text.indexOf(')', startIndex);
  }
  console.log(result.join(', '));
}
```

Check your solution here: https://judge.softuni.bg/Contests/312

Problem: Aggregate Table

- Extract all **towns** and their combined **incomes** from a text table:

```
['| Sofia              |
300',
 '| Veliko Tarnovo     |
500',
 '| Yambol             |
275']
```

```
Sofia, Plovdiv, Varna,
Yambol

1275
```

```
function aggregateTable(lines) {
  let sum = 0;
  let list = [];
  for (let line of lines) {
    let townData = line.split('|');
    list.push(townData[1].trim());
    sum += Number(townData[2].trim();
  }
  console.log(list.join(', ') + '\n' + sum);
}
```

Check your solution here: https://judge.softuni.bg/Contests/312

Problem: Restaurant Bill

- Write a function that prints purchased **products** (comma separated) and their **sum** from a list of products and sums (given as string array):

```
function printBill(input) {
  let items = input.filter((x,i) => i%2==0);
  let sum = input.filter((x,i) => i%2==1)
    .map(Number).reduce((a,b) => a + b);
  console.log(`You purchased ${items.join(', ')} for a
total sum of ${sum}`);
}
```

```
printBill(['Cola','1.35', 'Pancakes', '2.88']);
```

Check your solution here: https://judge.softuni.bg/Contests/312

Problem: Extract Username by Email

- You are given a list of email addresses
- Write a JS program that generates usernames by combining:
  - Email's **alias** (e.g. "**someone@domain.tld**" → "**someone**") + "**.**" +
  - The **first letters** of email's domain words (e.g. "**softuni.bg**" → "**sb**")

```
peshoo@gmail.com
todor_43@mail.dir.bg
foo@bar.com
bay.ivan@users.sf.net
```

```
peshoo.gc
todor_43.mdb
foo.bc
bay.ivan.usn
```

Check your solution here: https://judge.softuni.bg/Contests/312

Solution: Extract Username by Email

```
function extractUsernames(inputEmails) {
  let results = [];
  for (let email of inputEmails) {
    let [alias, domain] = email.split('@');
    let username = alias + '.';
    let domainParts = domain.split('.');
    domainParts.forEach(p => username += p[0]);
    results.push(username);
  }
  console.log(results.join(', '));
}
extractUsernames(['pesho@gmail.com', 'tod_or@mail.dir.bg']);
```

Check your solution here: https://judge.softuni.bg/Contests/312

Problem: Censorship

- You are given a text and a list of strings that need to be censored
- Write a JS program that **replaces** all occurrences of the banned strings with dashes of **equal length**

```
'roses are red, violets are blue',
[', violets are',
 'red']
```

```
roses are ---------------- blue
```

Solution: Censorship

```
function censor(text, words) {
  for (let current of words) {
    let replaced = '-'.repeat(current.length);
    while (text.indexOf(current) > -1) {
      text = text.replace(current, replaced);
    }
  }
  return text;
}
```

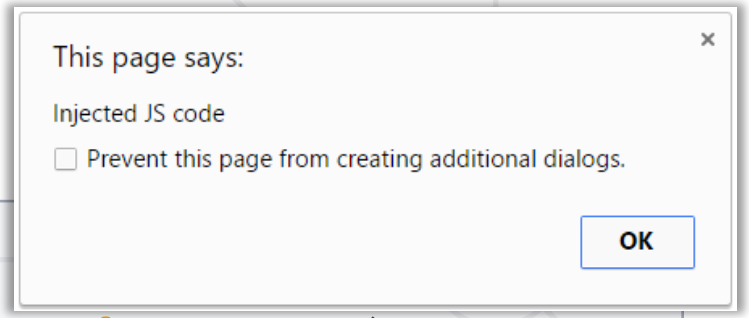
```
censor('I like C#, HTML, JS and PHP',
['C#', 'HTML', 'PHP'])
```

Check your solution here: https://judge.softuni.bg/Contests/312

HTML Escaping

HTML Escaping

- What is **HTML escaping**?
  - Replacing special characters with their escape sequence
  - Prevents JavaScript code injection in HTML pages
- In HTML escape the following characters:
  - '<', '>', '&', " ' " and '"'




```
document.write(
    'Hello, <script>alert("Injected JS code")</script>');
```

```
document.write('Hello, &lt;script&gt;
alert(&quot;...&quot;);&lt;/script&gt; ');
```

Implementing HTML Escaping

SoftUni Foundation

- Just replace the special characters with their escaped sequences
- **htmlEscape()** function can be attached to the **String** class:

```
String.prototype.htmlEscape = function() {
  return this.replace(/&/g, '&amp;')
    .replace(/</g, '&lt;')
    .replace(/>/g, '&gt;')
    .replace(/"/g, '&quot;')
    .replace(/'/g, '&#39;');
}
```

```
console.log('<script>'.htmlEscape()); // &lt;script&gt;
```

Problem: Print Strings as HTML List

- Write a JS function to return an **array of strings** as **HTML list**

```
"Hello", he said
<script>alert('hi');</script>
Use the <div> tag.
```

```
<ul>
  <li>&quot;Hello&quot;, he said</li>
  <li>&lt;script&gt;alert(&#39;hi&#39;);&lt;/script&gt;</li>
  <li>Use the &lt;div&gt; tag.</li>
</ul>
```

Solution: Print Strings as HTML List

```
function htmlList(items) {
  return "<ul>\n" +
    items.map(htmlEscape).map(
      item => `  <li>${item}</li>`).join("\n") +
    "\n</ul>\n";

  function htmlEscape(text) {
    let map = { '"': '&quot;', '&': '&amp;',
      "'": '&#39;', '<': '&lt;', '>': '&gt;' };
    return text.replace(/[\"&'<>]/g, ch => map[ch]);
  }
}
```

```
document.write(htmlList(["<br>", "It's OK"]))
```

Check your solution here: https://judge.softuni.bg/Contests/312

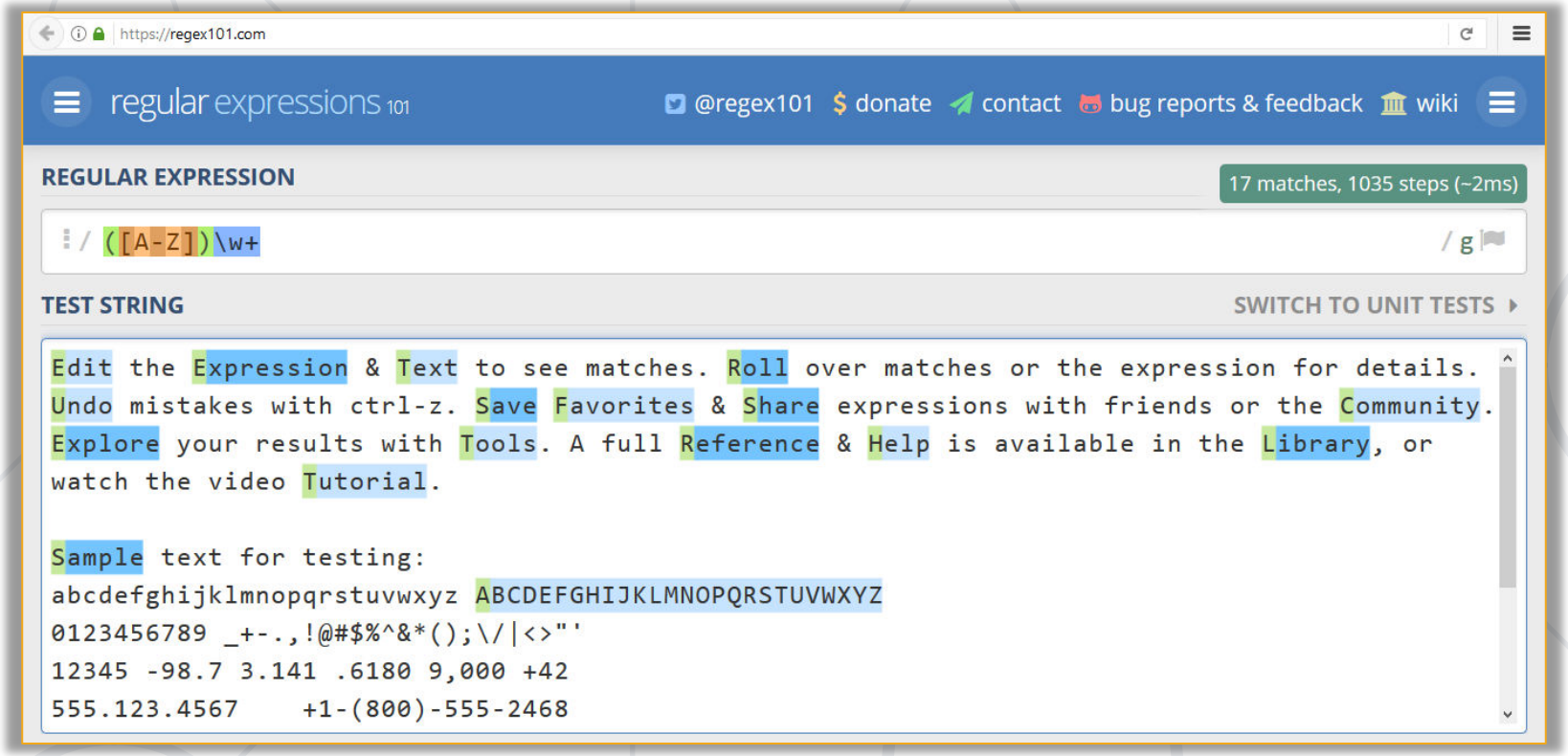
Live Exercises

Regular Expressions

The Beauty of Modern String Processing

What are Regular Expressions?

- **Regular expressions** (regex)
  - Match text by pattern
- **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



www.regex101.com

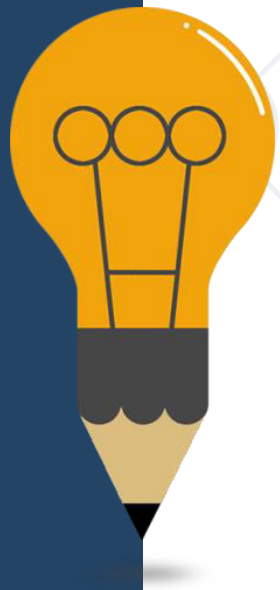
Live Demo

More RegExp Patterns

- `\d+` matches digits
  - `\D+` matches non-digits
- `\w+` matches letters (Unicode)
  - `\W+` matches non-letters
- `\+\d{1,3}([ -]*[0-9]){6,}`
  - Matches international phone, e.g. **+359 2 123-456**

Validation by Regex

- ^ matches start of text
- $ matches end of text
- ^\+\d{1,3}([-]*[0-9]){6,}$
  - Validates international phone
  - +359 2 123-456 is a valid phone
  - +359 (888) 123-456 is a invalid phone



Validation by RegExp in JS

SoftUni Foundation

```
let emailPattern =
  /^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,20}$/i;
```

- Always use ^ and $ in the regex validation patterns!

```
console.log(emailPattern.test("test@abv.bg"));
console.log(emailPattern.test("a.hills@gtx.de"));
console.log(emailPattern.test("invalid@@mail"));
console.log(emailPattern.test("err test@abv.bg"));
```

Problem: Email Validation

- Write a JS function that performs simple **email validation**
  - An email consists of: **username @ domain name**
  - **Usernames** are **alphanumeric**
  - **Domain names** consist of **two strings**, separated by a **period**
  - **Domain names** may contain only **English letters**

Valid: `valid123@email.bg`

Invalid: `invalid*name@emai1.bg`

Solution: Email Validation

```
function validateEmail(email) {
  let pattern =
    /^[a-zA-Z0-9]+@[a-z]+(\.[a-z]+)+$/g;
  let result = pattern.test(email);

  if (result) {
    console.log("Valid");
  } else {
    console.log("Invalid");
  }
}
```

Returns **true** if the email matches the pattern

```
validateEmail(['bai.ivan@mail.sf.net'])
```

Check your solution here: https://judge.softuni.bg/Contests/312

Regex Literals

- The classical (Perl syntax) is:
  - `/<regex>/<options>`
- Examples:
  - `/[a-z]+/gi` matches all non-empty sequences of Latin letters , case-insensitively
  - `/[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,20}/gi` matches emails (simplified pattern)

Split by RegExp in JS

```
let towns =
  "Sofia, Varna,Pleven,  Veliko Tarnovo;
Paris - London--Viena\n\n Пловдив|Каспичан";

console.log(towns.split(/\W+/)); // incorrect

console.log(towns.split(/\s*[.,|;\n\t-]+\s*/));
```

Problem: Expression Split

SoftUni Foundation

- Write a JS function that **splits** a given JS code into elements
  - All string literals will contain **only Latin letters**
  - The code should be split on the following:
    - **Whitespace** (including **tabulation**)
    - **Parentheses** and control **punctuation**: `( ) , ; .`
  - The **output** should contain **no empty elements**

Solution: Expression Split

```
function expressionSplit(expression) {

  let elements = expression
    .split(/[\s.();,]+/);
  console.log(elements.join("\n"));
}
```

```
expressionSplit(
    'let sum = 4 * 4,b = "wow";')
```

```
let
sum
=
4
*
4
b
=
"wow"
```

Check your solution here: https://judge.softuni.bg/Contests/312

Find All Matches by RegExp in JS

SoftUni Foundation

```
let text = "I was born at 14-Jun-1980. Today
is 29-Sep-2016. Next year starts at 1-Jan-2017
and ends at 31-Dec-2017.";


let dateRegex = /\d{1,2}-\w{3}-\d{4}/g;


console.log(text.match(dateRegex));
// ["14-Jun-1980", "29-Sep-2016", "1-Jan2017",
"31-Dec-2017"]
```

Problem: Match All Words

- Extract all word char sequences from given text

```
_ (Underscores) are also word characters!
```

→

```
_|Underscores|are|also|word|characters
```

```
function matchAllWords(text) {
  let words = text.match(/\w+/g);
  return words.join('|');
}
matchAllWords("Hello, how are you?")
```

Check your solution here: https://judge.softuni.bg/Contests/312

Problem: Match Dates

- Extract **all dates** from given text (array of strings)
  - Valid date format: **dd-MMM-yyyy**
  - Examples: **12-Jun-1999**, **3-Nov-1999**

```
I am born on 30-Dec-1994.
My father is born on the 9-Jul-1955.
01-July-2000 is not a valid date.
```

```
30-Dec-1994 (Day: 30, Month: Dec, Year: 1994)
9-Jul-1955 (Day: 9, Month: Jul, Year: 1955)
```

Solution: Match Dates (Using Groups)

SoftUni Foundation

```
function extractDates(inputSentences) {
  let pattern =
    /\b([0-9]{1,2})-([A-Z][a-z]{2})-([0-9]{4})\b/g;
  let dates = [], match;
  for (let sentence of inputSentences)
    while (match = pattern.exec(sentence))
      dates.push(`${match[0]} (Day: ${match[1]},
Month: ${match[2]}, Year: ${match[3]})`);
  console.log(dates.join("\n"));
}
```

```
extractDates(['1-Jun-2012 is before 14-Feb-2016'])
```

Check your solution here: https://judge.softuni.bg/Contests/312

Problem: Parse Employee Data

- **Validate** employee data and **store** it
  - Valid format is: `name - salary - position`
  - Names contain only **letters** and are **capitalized**
  - Position is **alphanumeric** and may hold **dashes** and **spaces**
  - Salary is a positive integer **number**
  - Invalid entries are **ignored**

```
Jonathan - 2000 - Manager
Peter- 1000- Chuck
George - 1000 - Team Leader
```

```
Name: Jonathan
Position: Manager
Salary: 2000
Name: George
Position: Team Leader
Salary: 1000
```

Analysis: Parse Employee Data

SoftUni Foundation

- Valid data:

Employee name:
`[A-Z][a-zA-Z]*`

Position:
`[a-zA-Z0-9 -]+`

```
Jonathan - 2000 - Manager
```

Salary:
`[1-9][0-9]*`

```
Name: Jonathan
Position: Manager
Salary: 2000
```

- Invalid data:

```
Peter- 1000-Chuck
```

No spaces around the "-"

Solution: Parse Employee Data

```
function parseEmployeeData(input) {
  let regex =
    /^([A-Z][a-zA-Z]*) - ([1-9][0-9]*) - ([a-zA-Z0-9 -]+)$/;
  for (let element of input) {
    let match = regex.exec(element);
    if (match)
      console.log(`Name: ${match[1]}\n` +
        `Position: ${match[3]}\n` +
        `Salary: ${match[2]} `);
  }
}
```

```
parseEmployeeData(['Jeff - 1500 - Staff', 'Ko - 150 - Ne'])
```

Check your solution here: https://judge.softuni.bg/Contests/312

Regex Replace

```
let str = '<img src="[imgSource]" />';
str = str.replace(/\[imgSource\]/, './smiley.gif');
```

```
let str = 'Visit <link>http://fb.com</link> or
<link>http://softuni.bg</link>.';
str = str.replace(/<link>(.*?)<\/link>/g,
  '<a href="$1">Link</a>');
```

Problem: Form Filler

- Write a function that replaces **username**, **email** and **phone placeholders** with supplied values
  - Username placeholder: `<!{letters}!>`
  - Email placeholder: `<@{letters}@>`
  - Phone placeholder: `<+{letters}+>`
  - The `{letters}` in the placeholders can hold only Latin letters
  - Any placeholder that does not meet these restrictions is invalid and should be left as is

Example: Form Filler

```
Pesho
pesho@gmail.com
90-60-90
Hello, <!username!>!
Welcome to your Personal profile.
Here you can modify your profile freely.
Your current username is: <!fdsfs!>. Would you like to change it? (Y/N)
Your current email is: <@DasEmail@>. Would you like to change it? (Y/N)
Your current phone number is: <+num+>. Would you like to change it? (Y/N)
```

```
Hello, Pesho!
Welcome to your Personal profile.
Here you can modify your profile freely.
Your current username is: Pesho. Would you like to change it? (Y/N)
Your current email is: pesho@gmail.com. Would you like to change it? (Y/N)
Your current phone number is: 90-60-90. Would you like to change it? (Y/N)
```

Solution: Form Filler

```
function fillForm(username, email, phone, data) {
  data.forEach(line => {
    line = line.replace(/<![a-zA-Z]+!>/g, username);
    line = line.replace(/<@[a-zA-Z]+@>/g, email);
    line = line.replace(/<\+[a-zA-Z]+\+>/g, phone);
    console.log(line);
  });
}
```

```
fillForm('pit', 'pit@pit.com', '032746',
['I am <!user!>, my email is <@email@>, my phone is <+p+>.'])
```

Check your solution here: https://judge.softuni.bg/Contests/312

Problem: Match Multiplication

- Write a JS function to multiply numbers in a text
  - Replace `{num1} * {num2}` by their product

```
My bill: 2*2.50 (beer); 2* 1.20 (kepab); -2  * 0.5 (deposit).
```

```
My bill: 5 (beer); 2.4 (kepab); -1 (deposit).
```

```
function performMultiplications(text) {
  text = text.replace(/(-?\d+)\s*\*\s*(-?\d+(\.\d+)?)/g,
    (match, num1, num2) => Number(num1) * Number(num2));
  console.log(text);
}
```

```
performMultiplications('My bill: 2*2.50 (beer)')
```

Check your solution here: https://judge.softuni.bg/Contests/312

RegExp in JavaScript Recap

SoftUni Foundation

- To create a **RegExp** with an object **literal**:

```
let regex = /ab+c/ig;
```

- To create a **RegExp** with a **constructor**:

```
let regex = new RegExp(/ab+c/, 'ig');
let regex = new RegExp('ab+c', 'ig');
```

- To get the matched text and **sub groups** (use in a **loop**):

```
regex.exec(str);
// [match, group1, group2, …]
```

RegExp in JavaScript Recap (2)

SoftUni Foundation

- To get all **matched** strings (use with **global** flag):

```
str.match(regex);
// [match1, match2, …]
```

- To **validate** string (use with **anchors**):

```
regex.test(str); // true | false
```

- Functional replace:

```
str.replace(regex, function);
// params: match, [p1, p2, …] offset, string
```

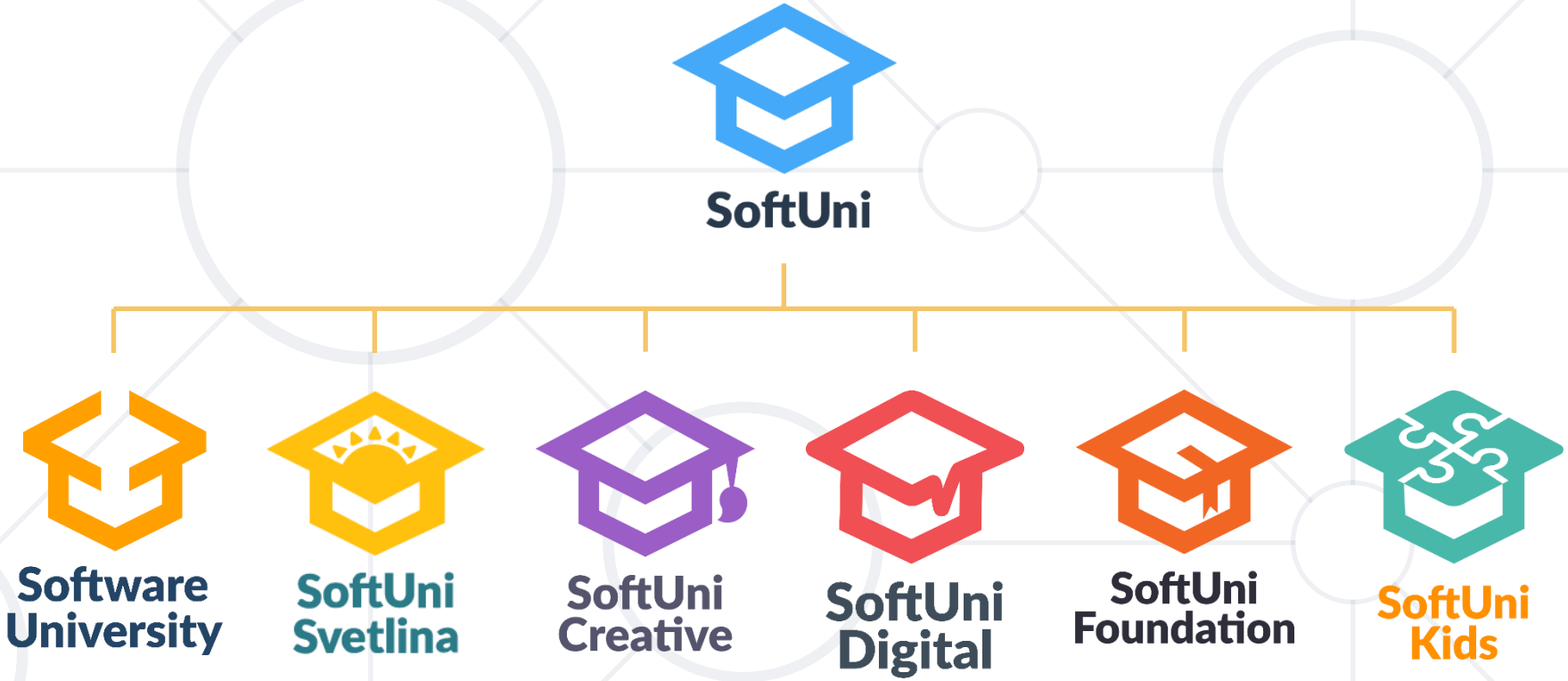
List of groups

Live Exercises

Summary

- String hold Unicode text
  - Have `length` and access by index `[]`
- String operations: `split()`, `substring()`, `indexOf()`, `trim()`, `replace()`, …
- Regular expressions
  - Patterns, groups, literals …

Questions?



https://softuni.bg/trainings/2080/js-fundamentals-september-2018

SoftUni Diamond Partners

SoftUni Foundation

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

codexio

SoftUni Organizational Partners

SoftUni Foundation

ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

WORLD OF MYTHS

Lukanet.com

codexio

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