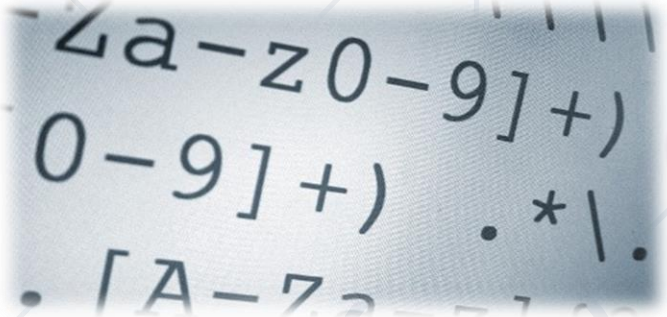
Regular Expressions (RegExp)

Regular Expressions Language Syntax



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Table of Contents

Have a Question?

Software University

sli.do

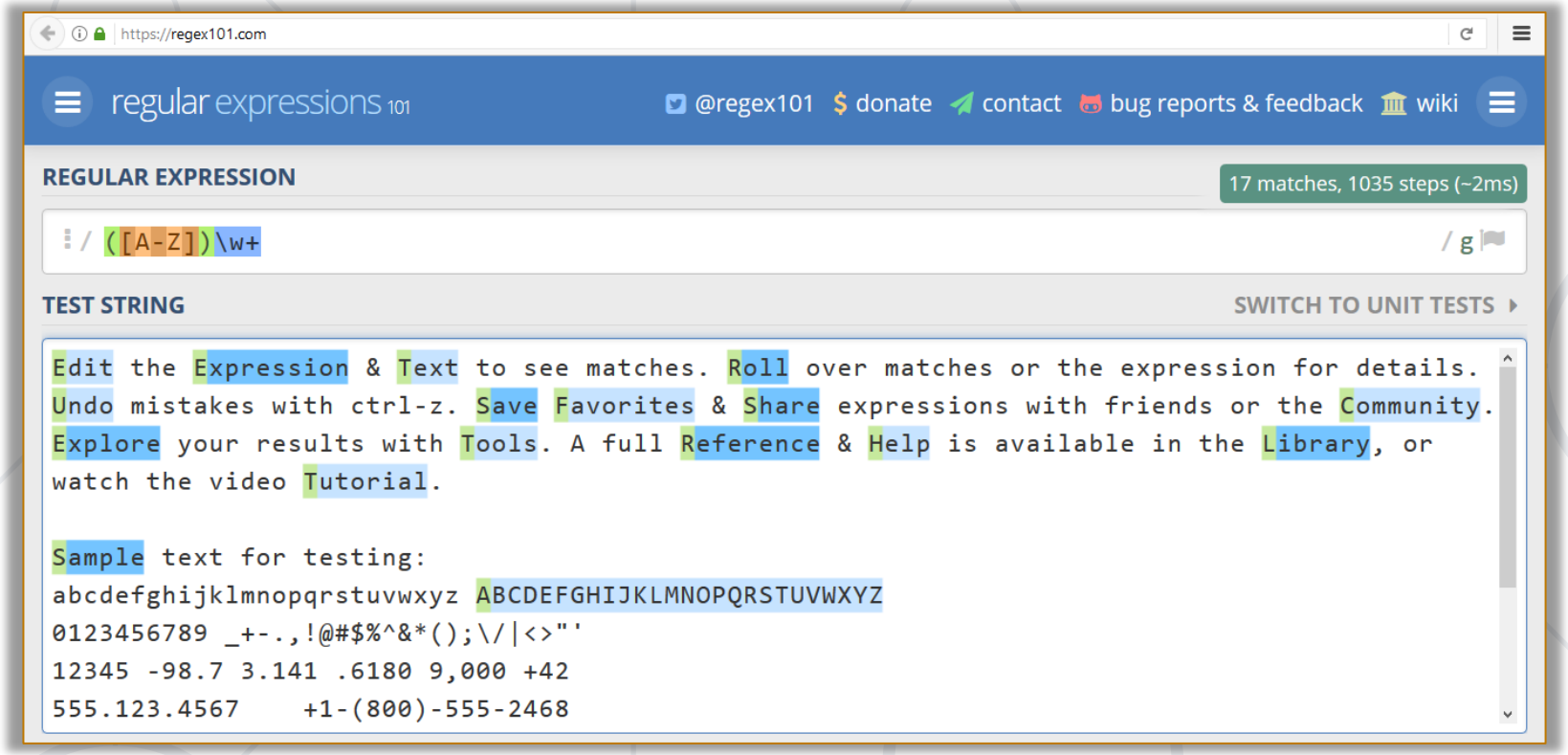
#fund-js

Regular Expressions

Definition and Classes

What Are Regular Expressions?

- Regular expressions (RegExp)
  - 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
- Play with regexp live at: regexr.com, regex101.com



www.regex101.com

Live Demo

Regular Expression Pattern – Example

- Regular expressions (**RegExp**) describe a search pattern
- Used to find / extract / replace / split data from text by **pattern**

```
[A-Z][a-z]+ [A-Z][a-z]+
```

```
John Smith
```

```
Linda Davis
```

```
Contact: Alex Scott
```

Character Classes: Ranges

- `[nvj]+` matches any character that is either **n**, **v** or **j**

```
node.js v0.12.2
```

- `[^abc]` – matches any character that is **not** **a**, **b** or **c**

```
Abraham
```

- `[0-9]` – character range: matches any digit from **0** to **9**

```
John is 8 years old.
```

Predefined Classes

- \w – matches any **word character** (a-z, A-Z, 0-9, _)
- \W – matches any **non-word character** (the opposite of \w)
- \s – matches any **white-space** character
- \S – matches any **non-white-space** character (opposite of \s)
- \d – matches any **decimal digit** (0-9)
- \D – matches any **non-decimal character** (the opposite of \d)

Quantifiers

Grouping

Quantifiers

- `*` – matches the previous element zero or more times

```
\+\d*
```

```
+359885976002 a+b
```

- `+` – matches the previous element one or more times

```
\+\d+
```

```
+359885976002 a+b
```

- `?` – matches the previous element zero or one time

```
\+\d?
```

```
+359885976002 a+b
```

- `{3}` – matches the previous element exactly 3 times

```
\+\d{3}
```

```
+359885976002 a+b
```

Grouping Constructs

- `(subexpression)` – captures the matched subexpression as numbered group

```
\d{2}-(\w{3})-\d{4}
```

```
22-Jan-2015
```

- `(?:subexpression)` – defines a non-capturing group

```
^(?:Hi|hello),\s*(\w+)$
```

```
Hi, Peter
```

- `(?<name>subexpression)` – defines a named capturing group

```
(?<day>\d{2})-(?<month>\w{3})-
(?<year>\d{4})
```

```
22-Jan-2015
```

Problem: Match All Words

- Write a regular expression in www.regex101.com that extracts all word char sequences from given text

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

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

Problem: Match Dates

- Write a regular expression that extracts **dates** from text
  - 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.
```

Problem: Email Validation

- Write a regular expression 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`

Backreferences

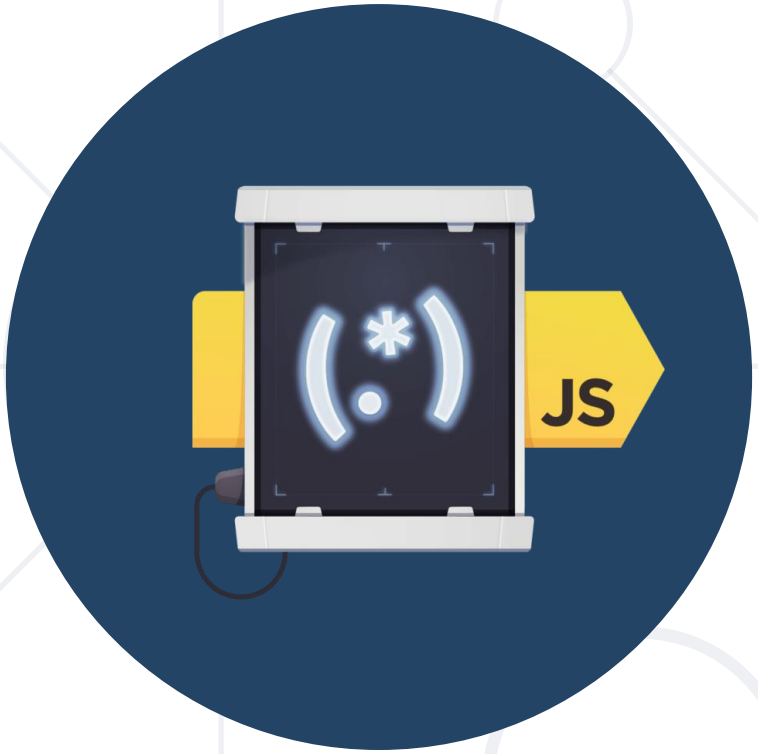
Numbered Capturing Group

Backreferences Match Previous Groups

- \number – matches the value of a numbered capture group

```
<(\w+)[^>]*>.*?<\/\1>
```

```
<b>Regular Expressions</b> are cool!
<p>I am a paragraph</p> … some text after
Hello, <div>I am a<code>DIV</code></div>!
<span>Hello, I am Span</span>
<a href="https://softuni.bg/">SoftUni</a>
```



Regular Expressions in JS

RegExp in JS

- In JS you construct a regular expression in one of two ways:
  - Regular Expression Literal
  - The constructor function **RegExp**

```
// Provides compilation when the script is loaded
let regLiteral = /[A-Za-z]+/g

// Provides runtime compilation
// Used when the pattern is from another source
let regExp = new RegExp('[A-Za-z]+', 'g');
```

Validating String by Pattern

- The method `test(string)`
  - Determines whether there is a match

```
let text = 'Today is 2015-05-11';
let regexp = /\d{4}-\d{2}-\d{2}/g;

let containsValidDate = regexp.test(text);
console.log(containsValidDate); // true
```

Checking for Matches

- The method **match(regexp)**
  - Returns an **array** of all matches (strings)

```
let text = 'Peter: 123 Mark: 456';
let regexp = /([A-Z][a-z]+): (\d+)/g;
let matches = text.match(regexp);


console.log(matches.length); // 2
console.log(matches[0]); // Peter: 123
console.log(matches[1]); // Mark: 456
```

Using the Exec() Method

Software University

- The method **exec(string, text)**
  - Works with a pointer & returns the **groups**

```
let text = 'Peter: 123 Mark: 456';
let regexp = /([A-Z][a-z]+): (\d+)/g;
let firstMatch = regexp.exec(text);
let secondMatch = regexp.exec(text);
console.log(firstMatch[0]) // Peter: 123
console.log(firstMatch[1]); // Peter
console.log(firstMatch[2]); // 123
```

Replacing with RegExp

- The method `replace(regexp, stringReplacement)`
  - Replaces all strings that **match the pattern** with the provided replacement

```
let text = 'Peter: 123 Mark: 456';
let replacement = '999';
let regexp = /\d{3}/g;
let result = text.replace(regexp, replacement);
// Peter: 999 Mark: 999
```

MatchAll

- The method **matchAll(regexp)**
  - returns an iterator of all results matching a string against a **regular expression**, including **capturing groups**

```
const regexp = /t(e)(st(\d?))/g;
const str = 'test1test2';
const array = [...str.matchAll(regexp)];
console.log(array[0]);
// ['test1', 'e', 'st1', '1', index: 0, input:'test1test2', length: 4]
```

Splitting with RegExp

- The method `split(regexp)`
  - Splits the text by the pattern
  - Returns an array of strings

```
let text = '1   2 3      4';
let regexp = /\s+/g;
let result = text.split(regexp);
console.log(result) // ['1', '2', '3', '4'];
```

Live Exercises

Problem: Match Full Name

- You are given a list of names
  - Match all full names

```
Ivan Ivanov, Ivan ivanov, ivan Ivanov, IVan Ivanov, Test Testov, Ivan   Ivanov
```

```
Ivan Ivanov Test Testov
```

Solution: Match Full Name

```
function solve(input) {
  let pattern = /\b[A-Z][a-z]+[ ][A-Z][a-z]+\b/g;
  let validNames = [];
  let validName = null;
  while((validName = pattern.exec(input)) !== null){
    validNames.push(validName[0]);
  }
  console.log(validNames.join(' '));
}
```

Problem: Match Phone Number

- Match a **valid phone number** from **Sofia**. After you find all **valid phones**, **print** them on the console, separated by **", "**
- A valid number has the following characteristics:
  - Starts with "**+359**"
  - Followed by the area code (always **2**)
  - Followed by the **number** itself, which consists of **7 digits** (separated into **two groups** of **3** and **4 digits** respectively)
  - The different **parts** are **separated** by either a **space** or a **hyphen** ('**-**')

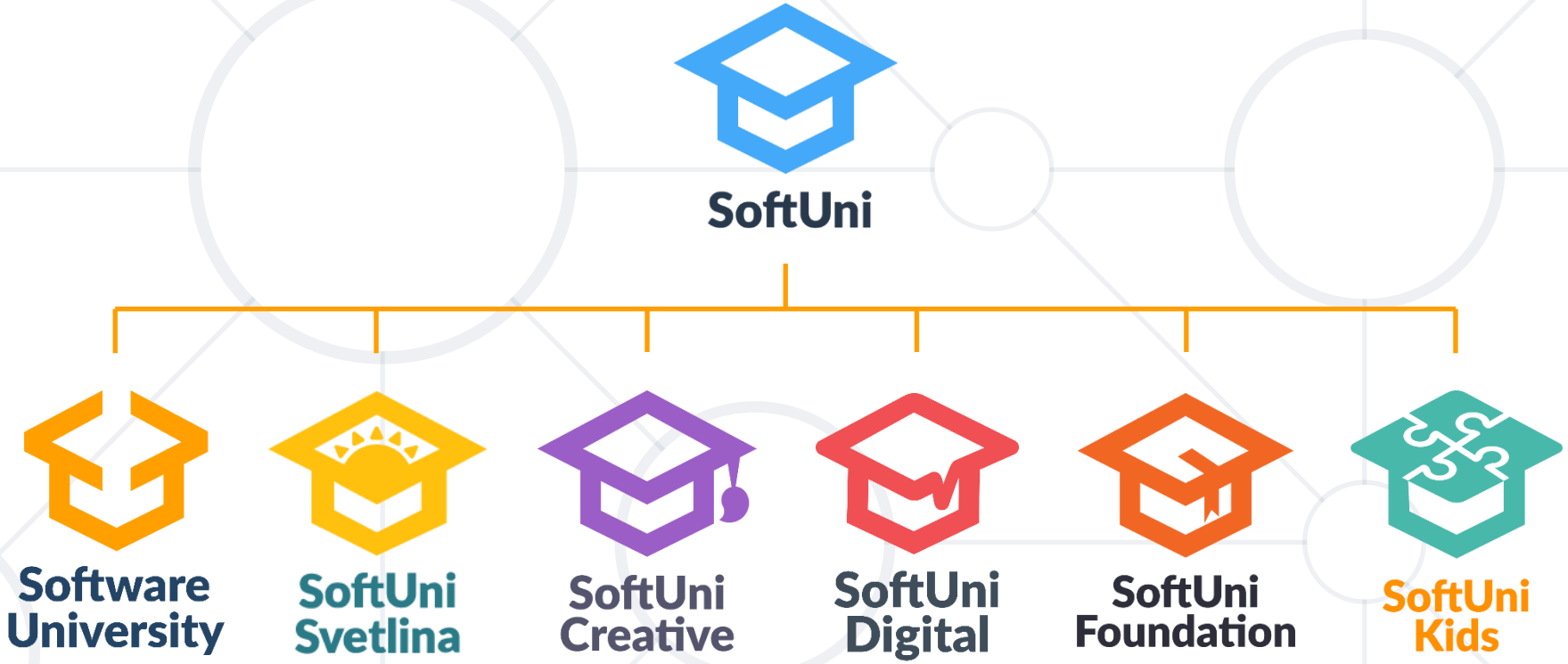
Solution: Match Phone Number

```
function regExPhones(input) {
  let validNames = [];
  let pattern = /(?<!\d)[+]359([ -])2\1\d{3}\1\d{4}\b/g;
  while ((validName = pattern.exec(input)) !== null) {
    validNames.push(validName[0]);
  }
  console.log(validNames.join(', '));
}
```

Summary

- **Regular expressions** describe **patterns** for searching through text.
- Define **special characters**, **operators** and **constructs** for building complex pattern.
- Can utilize **character classes**, **groups**, **quantifiers** and more.

Questions?

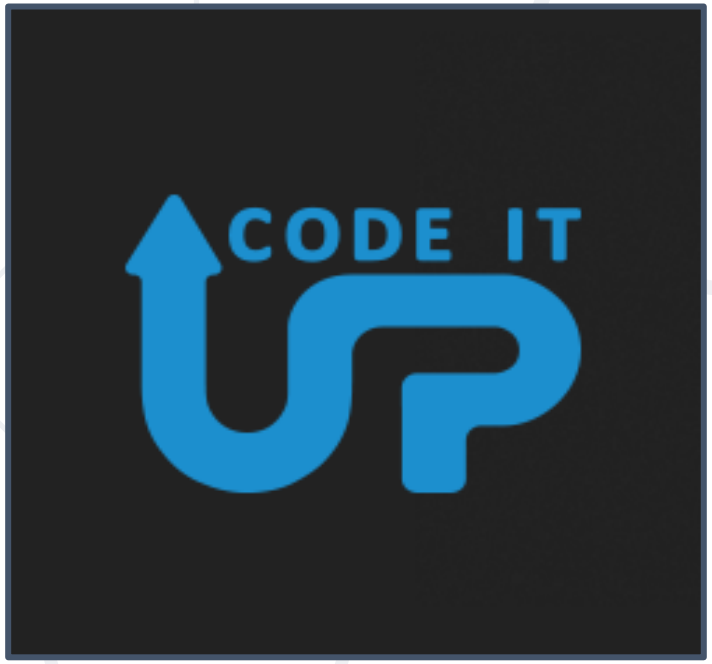


SoftUni Diamond Partners

Software University

Educational Partners

Software University



VIRTUAL RACING SCHOOL

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
  - Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg