Регулярные выражения в JavaScript

Поиск и замена строк с применением символьных паттернов

Обзор регулярных выражений

- Регулярное выражение **набор символов, которые определяют шаблон используемый для проверки соответствия комбинаций символов в строках**
  - Очень **мощьные для поиска и/или замены** тип операций
- Некоторые мощные примеры применения:
  - Найти и извлечь данные из документа
    - Извлечь изображения из HTML, извлечь исключения / ошибки из логов
  - Проверка текстовых данных:
    - Пароли, электронная почта, номера телефонов, интернет адреса

Regex синтаксис

- Регулярные выражения экстремально мощьный инструмент которым снабжены множество языков программирования
- Регулярные выражения используют свой собственный синтаксис (одинаковый в большинстве языков) для выполнения поставленных задачь
  - Трудно запоминаемый если использовать нечасто
  - MDN Regex
- Могут быть протестированы по следующим ресурсам:
  - http://www.regexr.com/
  - https://regex101.com/
- Регулярные выражения **встроены в JavaScript**
  - Могут быть созданы с использованием **regex литерала** или **функции конструктора**
  - Regex литерал удобно применять для статических выражений
  - Констуктор позволяет играть выражением в зависимости от параметров
- Следующее выражение найдет 'В**е**б программировани**е**', 'Программировани**е**', 'Вп**е**р**е**д', 'Что-то **е**щ**е**'

```
// literal syntax
const literalRegex = /e/g;
```

- Следующее выражение будет соответствовать '**В**еб программирование', '**В**есенний'

```
// function constructor syntax
const constructorRegex = new RegExp('^B', 'g');
```

Regex методы и свойства

- Полный лист свойств и методов MDN
- `RegExp.test` – Ищет совпадение в строке. Возвращает `true` или `false`
- `RegExp.exec` - ищет все совпадения в строке
  - Возвращает массив содержащий все найденные совпадения `null`.
- `String.match` – ищет совпадение в строке
  - Возвращает массив с информацией или `null` если совпадение не найдено
- `String.replace` – заменяет все совпадение подстроки в строке на указанную строку
  - Возвращает новую строку
- `String.split` – разбивает строку на массив подстрок, используя регулярное выражение или подстроку
  - Возвращает массив
- `String.search` – тестирует наличие совпадения в строке
  - Возвращает индекс совпадения, или -1 если совпадения не обнаружено

Флаги регулярных выржений

- Регулярное выражение имеют необязательные флаги, которые позволяют устанавливать глобальный поиск и поиск без учета регистра
  - Эти флаги могут использоваться как по отдельности, так и совместно в одном выражении
    - **g** - глобальное сопоставление
    - **i** - игнорирование регистра при сопоставлении
    - **m** - сопоставление по нескольким строкам; символы начала и конца (^ и $) начинают работать по нескольким строкам (то есть, происходит сопоставление с началом или концом каждой строки (строки разделяются символами \n или \r), а не только с началом или концом всей вводимой строки)

- **y** - «липкий» поиск; сопоставление в целевой строке начинается с индекса, на который указывает свойство lastIndex этого регулярного выражения (и не пытается сопоставиться с любого более позднего индекса).

Используем Regex in JavaScript

- Учесть все совпадения об упоминании о **Курс веб-программирования**
  - `RegExp.test` , `String.match` , `RegExp.exec`

```
let academyRegex = /Курсах\s(веб\s)?программирования/gi;

let text = 'Женя изучает JavaScript На курсах веб программирования ',
          'пока Саня, то же учится на курсах программирования.';

// должен содержать массив совпадений или null
let matches = text.match(academyRegex);

// get matches and matched groups one by one
let currentMatch;
while(currentMatch = academyRegex.exec(text)) {
    console.log(currentMatch);
}
```

- Заменить все пробелы, табуляцию и переводы строки из текста
  - `String.replace`

```
let text = 'text    with    lots of       spaces\n' +
           '      and lots of tabulations    ';
console.log(text.replace(/\s\s+/g, ' '));
```

- Разбить JavaScript выражения для получения операндов
  - `String.split`

```
let expression = '4+5*count-initialCount+1';
let operands = expression.split(/\+|\*|-/);
console.log(operands);
```

- Поиск первого соответствия выражению
  - `String.search`

```
let text = 'JavaScript крутой!';
let index = text.search(/крутой/);
console.log(index);
```

Специальные символы в Regex

- Регулярные выражения имеют несколько специальных символов которые позволяют менять поведение:
  - Символ для поиска нескольких символов
  - Символ для поиска пробела
  - Символ для поиска цифр
  - Символ для поиска букв
  - и.т.д.
- Полный перечень специальных символов доступен по ссылке здесь
  - `*` – Представляет символ или группу в выражении 0 или больше раз
  - `+` – То же что и `*` - только предтавляет 1 или больше раз
  - `?` – представляет символ или группу в выражении 0 or 1 раз
  - `.` (dot) – представляет один любой символ исключая перевод строки
  - `|` – Совпадение одного или другого паттерна
  - `[xyz]` – Набор символов - Совпадения указанных символов
  - `[x-z]` – Набор символов - Совпадения символов, вхоящих в указанный диапазон
  - `[^xyz]` – Инвертированый набор символов - совпадения других символов
  - `{N}` – точно соответствует `N` вхождений элемента или группы
  - `{N, M}` – соответствует от `N` до `M` вхождений элемента или группы
  - `^` - совпадения от начала строки
  - `$` совпадения до конца строки
  - `\s` – совпадение одного пробельного символа, включает пробел, табуляция, form feed, line feed
  - `\S` – Совпадение любого символа кроме пробельного
  - `\d` – Совпадение цифры
    - Еквивалент `[0-9]`
  - `\D` – совпадение не цифрового символа
    - Еквивалент `[^0-9]`
  - `\w` – соответствует любому алфавитно цифровому символу, включая нижнее подчеркивание
  - `\W` – любые совпадения кроме алфавитно-цифровых символов или нижнего подчеркивания

Валидация имени пользователя

- имя пользователя:
  - может содержать латинские буквы верхнего и нижнего регистра, цифры и нижнее подчеркивание `_`
  - его длина может быть между `4` и `15`, искючительно
  - Должен начинаться с заглавной буквы

- Протестировать регулярные выражения на следующем массиве:

```
['Chris11', '', 'Joe', 'Peter_356', '123george',
 '__proto__', 'ImATooLongUsername15', 'J0hn_', '<h1>scripter</h1>']
```

Извлечь все ссылки на изображения

- Извлечь все ссылки на изображения в HTML разметке
- Протестируйте ваше выражение на unn.ru