

# ОСНОВЫ JS

## (RegExp)

# RegExp введение

- Регулярные выражения – очень мощное средство для поиска, валидации и замены текстовых данных
- `var regexpr = new RegExp("шаблон", "флаги");`
- `var regexpr = /шаблон/gmi; // с флагами (Предпочтительнее!)`

# RegExp введение 2

```
var sea=/ava/
```

```
var str='Я изучаю RegExp JavaScript'
```

```
alert(str.search(sea));//17
```

```
alert(str.search(/чаю/));//5
```

```
alert(str.search(/script/));//-1(если нет)
```

# Сторона и флаги

- **i** - ищет независимо от регистра, то есть не различает большие и маленькие буквы
- **g** - ищет все совпадения, иначе — только первое.
- **m** - многострочный режим.
- `alert(str.search(/script/i));`//20 нашло!

# "Инструменты" или методы

- **str.search(reg)** - Он возвращает позицию первого совпадения или -1, если ничего не найдено
- **str.test(reg)** - Метод test проверяет, есть ли хоть одно совпадение в строке str. Возвращает true/false. **[str.search(reg) != -1]**
- **str.split(reg|substr,limit)** - Разбивает строку в массив по разделителю – регулярному выражению regexr или подстроке substr

\*\*\* `str.match(reg)` без флага `g` и с ним

- Находит только одно, первое совпадение
- `result[0]` – совпадение
- `result.index` – индекс вхождения
- `result.input` – входная строка
- При наличии флага `g`, вызов `match` возвращает обычный массив из всех совпадений.
- Или `null`, если ничего не найдено

\* `str.replace(reg, str|func)`

Заменить дефис на двоеточие

```
alert('12-34-56'.replace("-", ":")) // 12:34-56
```

\* `str.replace(reg, str|func)`

Заменить дефис на двоеточие

```
alert('12-34-56'.replace("-", ":")) // 12:34-56
```

// Заменить дефис на двоеточие(все!)

```
alert( '12-34-56'.replace( /-/g, ":" ) ) // 12:34:56
```

Попробуйте сами:

- Заменить точки на восклицательные знаки
- Сделать каждое слово с новой строки



# Важнейшие классы

**\d** – цифра от 0 до 9

"Стандарт CSS4 - это здорово".match(**\d/**) //4

**\s** - Пробел, включая табы, переводы строки

**\w** - буква латинского алфавита или цифра  
или подчёркивание '\_'

**\b** – граница между словами

alert( "Я люблю HTML5!".match(**\s\w\w\w\w\d/**)  
); // 'HTML5'

Каждому классу соответствует один символ!

## \*\* Обратные классы

**\D** – Что угодно, кроме цифр от 0 до 9

**\S** - Не пробел, например цифра или буква

**\W** - не латинница, не подчёркивание, не цифра, например **русские буквы**

**\B** – Анти \b !

## \* Набор

[acid]

"искать любой символ из указанных в [...]".

"In his garden".match(/[acid]/g) //["i", "a", "d"]

// найти [г или т], а затем "оп"

alert( "Гоп-стоп".match(/[гт]оп/gi) ); // "Гоп",  
"топ"

## \* Диапазон

[A-Z] – Любой символ от A до Z.

[a-zA-Z] – Любой символ.

[a-Z] – НЕДОПУСТИМО (\*\*Почему?)

[0-5] – любая из цифр от 0 до 5

"искать любой символ из указанных в [...]".

\d – то же самое, что [0-9],

\w – то же самое, что [a-zA-Z0-9\_],

\s – то же самое, что [\t\n\v\f\r ] плюс несколько юникодных пробельных символов.

# Квантификаторы +, \*, ? и {n}

{n} – где n – число повторений

{4} – 4 повторения например год

{2,4} – от двух до четырех цифр

{2,} – как минимум 2

Сокращения: (Напишите самостоятельно!)

- + – один или более
- ? – ноль или один
- \* – ноль или больше

## \*\* Скобочные группы

- Позволяет применять квантификатор сразу ко всей скобке, а не всего лишь к одному символу:

```
var str = "#abccabc и #ars";
```

```
alert( str.match(/#([\w\d]{3}){1,2}/g) );
```

\*\*\*\*\* Обратные ссылки: \n и \$n изучаем на свой страх и риск (сложный материал)

| сдавайся | тебе \$ ! Сейчас я ^  
править!

| - или "альтернатива"

/html|php|css|java(script)?/gi – найдет или html  
или php или css или java или javascript

В отличие от диапазонов берет выражения  
ЦЕЛИКОМ. Выражением может быть RegExr

Знак каретки '^' и доллара '\$' являются так  
называемыми «якорями» для валидации

Каретка ^ совпадает в начале текста, а  
доллар \$ – в конце.

# Практика

|, ^, \$

1. Проверить ip адрес. (Формат – четыре числа от 0 до 255 отделенные ".") )
2. Написать регулярное выражение, которое проверит условие if видов:

if (a>=b) { ;}

if (a===b) { ;}

if (a!==b) { ;}

if (a==b) { ;}

if (a!=b) { ;}

if (das<vas) {;}



# Задачи 1

Эх, пссс... Стоит начать с 3-ей задачи! =]

- 1. Почта

Проверить адрес почты, если он может содержать только латинские символы, \_ и цифры. При этом должен располагаться на домене mail.ru или gmail.com

\*\* Располагаться на любом домене, кроме "mailforspam"

- Например : [hi\\_@my.perfect.mail.site.ru.com.de](mailto:hi_@my.perfect.mail.site.ru.com.de)

# Задачи 2

- Взять номера телефонов в произвольном формате и привести к образцу

+7 (9XX) XXX – XX – XX

- Номер состоит из 9 или 10 цифр и разделителей
- \*\*\* Разобрать **входные данные**:

3547868      89211233242      8125532142

+7952-2382978    \*7\*911\*111\*11\*1\*1

И прочие. *Подробнее о номерах телефонов  
почитайте в интернете.*

# Задачи 3

Распарсить ФИО на составляющие части.  
Учесть, что не у всех людей есть отчество.  
Написать Фамилию, Имя и Отчество с  
больших букв, если они с маленьких.

## \* Задача список покупок

- Необходимо написать парсер, который из естественной записи формата "одно наименование на строку" составит формализованный список продуктов.

Пример:

Из

Сыр 200 г

Помидоры 3 кило =>

Сырки – 3 шт.

Получить

Сыр (0,2 кг.)

Помидоры (3 кг.)

Сырки (3 шт.)

# Задача 5

## Анализатор пароля

По введенному паролю проверить использование только допустимых символов в пароле (список на Ваш выбор) и оценить его устойчивость.

Устойчивость зависит от длины, используемых символов, цифр, РеГиСтрА.