PHASE 1

WEEK 1

DAY 4

ELBRUS / BOOTCAMP

Strings

строки

ELBRUS/BOOTCAMP

Strings

- Строковые литералы (" !== ")

```
"Это строка";

'Это тоже строка';
```

- Специальные символы и экранирование

```
'Hello, \n\t world!';
'Hello \'my\' world!';
'Это \u2014\x20 просто строка';

'Это строка с \'кавычками\'';
```

ELBRUS / BOOT CAMP

Strings

- JS-код в строках (в апострофах!!!)

```
const names = [‘Oleg’, ‘Анна’, ‘Mike’];
let i = names.length - 1;
while (i) {
  console.log(`Hello, ${names[i]}!`)
  i--;
}
// "Hello Mike!"
// "Hello Анна!"
// "Hello Oleg"
```

Strings

- Получение символа

```
const str = ‘Строка’;
str[1]; // "т"
str.charAt(2); // "р"

str.charCodeAt(2); // 1088 - номер символа Юникода
```

- Приведение строк к одному регистру

```
str.toLowerCase(); // "строка"
str.toUpperCase(); // "СТРОКА"
```

indexOf / lastIndexOf

```
const str = "Widget with id";
```

- indexOf() возвращает индекс первого подходящего вхождения, начиная с указанного индекса (-1 - при отсутствии совпадения)

```
str.indexOf("id"); // 1 -> "Widget with id"
str.indexOf("id", 6); // 12 -> "Widget with id" (начало поиска с 6го символа)
```

- lastIndexOf() - аналогично indexOf(), поиск по строке ведётся от конца к началу, начиная с указанного индекса

```
str.lastIndexOf("id"); // 12 -> "Widget with id"
```

ELBRUS/BOOTCAMP

slice / substring

```
const str = "Просто пример";
```

- slice() извлекает все символы до указанного индекса (второй аргумент), не включая сам этот индекс.

```
const s1 = str.slice(7);    // "пример"
const s2=  str.slice(-6, -3); // "при"
```

- substring() очень похож на slice.

```
const s3 = str.substring(7);    // "пример"
const s4 = str.substring(3, 6); // "сто"
```

ELBRUS/BOOTCAMP

replace / replaceAll

```
const str = "JavaScript такой такой простой!";
```

- replace() возвращает новую строку с некоторыми или всеми сопоставлениями с шаблоном, замененными на заменитель. Шаблон может быть строкой или регулярным выражением.

```
str.replace("такой", "не");
// "JavaScript не такой простой!"

str.replace(/такой/g, "не");
// "JavaScript не не простой!"
```

Вторым параметром replace() может принимать строку или функцию для замены.

Regexp

Регулярные выражения

Regexp

У вас есть проблема.

Вы решили использовать регулярные выражения, чтобы её решить.

Теперь у вас две проблемы

Unknown Author

ELBRUS/BOOTCAMP

Форма записи

```
const regexp1 = new RegExp("шаблон", "флаги");
const regexp2 = /шаблон/g;
```

Пример использования:

```
const str = "Я люблю JavaScript!";
const regexp = /лю/;
str.search(regexp); // 2
str.search(/лю/);   // 2
```

RegExp: методы match(), test()

`match` — строковый метод, возвращает результат сопоставления строки с регулярным выражением.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/match

`test` — метод регулярного выражения, возвращает булевое значение, совпала ли строка с выражением.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/test

ELBRUS/BOOTCAMP

Флаги

i (ignoreCase)

регистронезависимость. A и a - одно и то же

g (global)

ищет все вхождения, иначе только первое

m (multiline)

работает с несколькими строками

Наборы символов

[123] - вхождение любого символа из перечисленных

[0-9] - вхождение любого символа из диапазона

[^0-9] – всё, кроме цифр

/[взкд]ол/g // сработает на слова: вол, кол, дол, зол

. (точка) – любой символ

\d = [0-9] \D = [^0-9]

\w = [a-zA-Z0-9_] \W = [^a-zA-Z0-9_]

\s = [\t\n\v\f\r] \S = [^\t\n\v\f\r]

ELBRUS / BOOT CAMP

Специальные символы

Символы, которые используются в правилах регулярок, надо экранировать, если мы хотим искать их как обычные символы.

К спецсимволам относятся:

[\ ^ $. | ? * + ( )

/\d\.\d/ // найдёт: 5.1, 7.4

/\d.\d/ // найдёт: 5Щ1 551 и другие

ELBRUS BOOTCAMP

Границы строки

^ - начало строки

$ - конец строки

/+7\d/ // найдёт: 89+743

/^+7\d/ // найдёт: +75765

/+7\d$ // найдёт: 64+75

Квантификаторы

{n} - ровно n вхождений

\d{6} - срабатывает на 6 цифр подряд. [0-9]{6} = \d\d\d\d\d\d

{2,4} - от 2 до 4 вхождений

{3,} - 3 и более вхождения

+ = {1,}

* = {0,}

? = {0,1}

/ка*[пй]?от/ // сработает на: капот, кот, кайот

Группы

- Скобки

  /(ко-)+/i

  // Ко-ко-ко-ко-ко-кот, ко-кот, ко-ко-кот

- Один из вариантов

  Правильно:

  /(Работать надо)((плохо)|(хорошо))/g

  Неправильно:

  /Работать надо (хорошо)|(плохо)/g

Группы порядок вывода

Чтобы поменять местами слова (символы) в строке, использует обозначения $1 и $2 для обозначения первого, второго и т.д. совпадения скобочного выражения

```
const date = "2022.05.19";
const myReg = /(\d{4})\.(\d{2})\.(\d{2})/gmi

const dateCopy = date.replace(myReg, "$3-$2-$1")
```

ELBRUS/BOOTCAMP

Регулярка для телефона

+7(903 123-45-67

(8|\+7)\(?(9\d{2})\)?((\d{7})|(\d{3}-\d{2}-\d{2}))

/\+7\(9\d{2}\)\d{3}-\d{2}-\d{2}/g

/\+7\(9\d{2}\)\d{3}(-\d{2}){2}/g

Добавим начало +7 или 8

/((\+7)|8)\(9\d{2}\)\d{3}(-\d{2}){2}/g

Скобки и дэши сделаем опциональными

/((\+7)|8)\(?9\d{2}\)?\d{3}(-?\d{2}){2}/g

ELBRUS / BOOTCAMP