Домашнее задание к лекции 2.6 «Регулярные выражения»

Перед началом работы

1. Активируйте строгий режим соответствия.

Задача № 1. Купон-палиндром.

В нашем интернет-магазине акция! Мы решили дать скидку 50 % на все заказы с купоном, код которого является палиндромом. Палиндром — число, буквосочетание, слово или текст, одинаково читающееся в обоих направлениях.

Нужна функция `checkCoupon`, которая будет проверять код купона и возвращать `true`, если код является палиндромом, без учета регистра символов, и `false`, если нет, либо если длина кода меньше `10` символов.

Описание функции

Функцию checkCoupon принимает:

1. Код купона, строка.

Функция должна откинуть всё, что не является буквой латинского алфавита или цифрой, проверить размер строки и вернуть `true`, если после отбрасывания ненужных символов строка имеет длину `10` или более символов и при этом является палиндромом, и `false` в остальных случаях.

Пример использования функции

```
let codes = [
  'Madam, I’m Adam',
  'A man, a plan, a canal. Panama',
  '----<-------Eve------->-----',
  '[__777-x-44-x-777__]',
  '1234564321',
  'Olson in Oslo'
];

for (let code of codes) {
  let result = checkCoupon(code) ? 'подходит' : 'не подходит';
  console.log(`Код «${code}» ${result}`);
}
```

Если функция checkCoupon реализована верно, то получим такой вывод в консоль:

```
Код «Madam, I’m Adam» подходит
Код «A man, a plan, a canal. Panama» подходит
```

```
Код «----<-------Eve------->-----» не подходит
Код «[__777-x-44-x-777__]» подходит
Код «1234564321» не подходит
Код «Olson in Oslo» подходит
```

Процесс реализации

1. Создайте функцию `checkCoupon`, принимающую нужное количество аргументов.
2. Приведите код купона к нижнему регистру и отбросьте ненужные символы.
3. Проверьте, что код купона имеет требуемое количество символов.
4. Любым удобным способом проверьте, что код купона является палиндромом.
5. Верните результат проверки.

После создания функции проверьте работу кода из примера использования, а также свои собственные варианты купонов.

Задача № 2. Вырезаем теги.

Наш маркетолог начал проходить курс по основам HTML и CSS и теперь размечает тегами каждое слово в тексте для сайта. Может поэтому ему все еще не выдали диплом.

Для защиты от верстальщика-энтузиаста нам требуется реализовать функцию `stripTags`, которая будет удалять все HTML-теги из текста. На данном этапе необязательно реализовывать удаление тегов с атрибутами, потому что наш маркетолог пока не знает об их существовании.

Описание функции

Функция `stripTags` должна принимать один аргумент: текст с разметкой, строка. Функция должна удалить из текста все HTML-теги и вернуть текст без них.

Пример использования

```
const texts = [
  '<strong>Наши</strong> <em>ховерборды</em> лучшие в <u>мире</u>!',
  '<EM>Световой меч</EM> в <strong>каждый</strong> дом!'
];

for (let text of texts) {
  console.log(stripTags(text));
}
```

Если все правильно сделать, то получим такой вывод в консоль:

```
Наши ховерборды лучшие в мире!
```

```
Световой меч в каждый дом!
```

Процесс реализации

1. Создайте функцию `stripTags`, принимающую нужное количество аргументов.
2. Удалите все теги из текста, переданного в функцию, любым удобным способом.
3. Верните строку без тегов, полученную после их удаления.

После создания функции убедитесь, что пример использования работает без ошибок и даёт правильный результат. После чего проверьте работу функции, используя свои варианты размеченного текста.

Задача № 3. Валидатор форм.

У нас на сайте — множество форм заявок. И наши клиенты часто заполняют их всякой белибердой, что причиняет боль и страдания отделу продаж.

Реализовать функцию `validate`, которая будет проверять правильность заполнения любой формы и возвращать `true`, если форма заполнена правильно, иначе – `false`.

Описание функции

Функция `validate` должна принимать следующие аргументы:

1. Данные формы, объект, имена свойств которого соответствуют полям формы, а значения — введенным в них данным;
2. Требования к данным формы, массив объектов, каждый имеет следующие свойства:
   `name`— название поля формы, строка;
   `rule`— проверка значения, строка или регулярное выражение.
   Функция должна проверить каждое поле из массива требований к данным формы.
   Если свойство `rule` — строка, то использовать следующие условия:
   `email` — в поле должен быть правильный адрес электронной почты;
   `phone` — в поле должен быть правильный полный номер телефона в России, начинающийся с +7.

Если поле `rule` — регулярное выражение, то просто проверить поле на соответствие этому выражению.

Пример использования функции

```
const fields = [
  { name: 'name', rule: /^[a-z ]{5,}$/i },
  { name: 'email', rule: 'email' },
  { name: 'phone', rule: 'phone' },
];

const forms = [
  { name: 'Ivan Ivanov', email: 'ivan@test.co', phone: '+79212753690' },
```

```
  { name: 'III', email: 'ivan@test', phone: '11111' }
];

for (let form of forms) {
  console.log(form);
  if (validate(form, fields)) {
    console.log('Ошибок нет');
  } else {
    console.log('Форма заполнена неверно');
  }
}
```

Если все правильно сделать, то вывод в консоль будет таким:

```
{ name: 'Ivan Ivanov',
  email: 'ivan@test.co',
  phone: '+79212753690' }
Ошибок нет
{ name: 'III', email: 'ivan@test', phone: '11111' }
Форма заполнена неверно
```

Процесс выполнения

1. Создайте функцию `validate`, принимающую нужное количество аргументов.
2. Для каждого элемента из массива требований сделайте следующее:
    1. Получите значение, введенное в соответствующее поле формы, взяв имя поля из свойства `name`.
    2. Получите правило проверки значения поля из свойства `rule`.
    3. Если правило является строкой, преобразуйте его в подходящее регулярное выражение из описания функции.
    4. Проверьте значение поля формы на соответствие регулярному выражению.
3. Если все поля формы соответствуют требованиям, верните `true`.
4. Если хотя бы одно из полей не соответствует, верните `false`. После создания функции убедитесь, что пример использования работает без ошибок и даёт правильный результат. После чего проверьте работу функции, используя свои варианты форм и правил проверки.