# Домашние задание #3

## #11 Пройти курс «JavaScript в браузере»

Для того, чтобы выполнить задание, нужно полностью пройти курс.
https://htmlacademy.ru/courses/219

## #12 Учебный проект: нас орда

**Подготовка**

В форке учебного проекта создайте ветку `module3-task1` и в этой ветке выполните следующие шаги:

- Создайте файл `js/setup.js` в вашем учебном проекте. Это файл, в котором вы будете вести работу со всплывающим окном настройки персонажа.
- В файле `index.html` подключите ваш файл при помощи тега `script`.

**Задача**

В файле `setup.js`

1. Покажите блок `.setup`, убрав в JS-коде у него класс `.hidden`.
2. Создайте массив, состоящий из 4 сгенерированных JS объектов, которые будут описывать похожих персонажей. Объекты должны содержать следующие поля:
    - `name`**, строка** — случайно сгенерированное имя персонажа. Имя генерируется из массивов имен и фамилий: нужно случайным образом выбрать из массива имен имя, а из массива фамилий фамилию и сложить их. При желании имя и фамилию можно в случайном порядке менять местами:)

      **Имена**
        - Иван
        - Хуан Себастьян
        - Мария
        - Кристоф
        - Виктор
        - Юлия
        - Люпита
        - Вашингтон

      **Фамилии**
        - да Марья
        - Верон
        - Мирабелла
        - Вальц
        - Онопко
        - Топольницкая
        - Нионго
        - Ирвинг

    - `coatColor`**, строка** — случайный цвет мантии на выбор из следующих:

        - rgb(101, 137, 164)
        - rgb(241, 43, 107)
        - rgb(146, 100, 161)
        - rgb(56, 159, 117)
        - rgb(215, 210, 55)
        - rgb(0, 0, 0)

    - `eyesColor`**, строка** — случайный цвет глаз персонажа на выбор из следующих:
        - black
        - red
        - blue
        - yellow
        - green

3. На основе данных, созданных в предыдущем пункте и шаблона `#similar-wizard-template` создайте DOM-элементы, соответствующие случайно сгенерированным волшебникам и заполните их данными из массива:
    - Имя персонажа `name` запишите как текст в блок `.setup-similar-label`;
    - Цвет мантии `coatColor` задайте как цвет заливки `fill` в стилях элемента `.wizard-coat`;
    - Цвет глаз `eyesColor` задайте как цвет заливки `fill` в стилях элемента `.wizard-eyes`.

4. Отрисуйте сгенерированные DOM-элементы в блок `.setup-similar-list`. Для вставки элементов используйте `DocumentFragment`.
5. Покажите блок `.setup-similar`, удалив у него CSS-класс `hidden`.

**Требования к коду**

Код должен быть разделен на отдельные функции. Стоит отдельно объявить функцию генерации случайных данных, функцию создания DOM-элемента на основе JS-объекта, функцию заполнения блока DOM-элементами на основе массива JS-объектов. Пункты задания примерно соответствуют функциям, которые вы должны создать.

## #13 Личный проект: пока все дома

В этом задании мы начнём выполнение части ТЗ, в которой говорится об отображении похожих объявлений в Кексобукинге и фотографий других пользователей в Кекстаграме.

Пока для отрисовки нет настоящих данных, которые будут получены с сервера позже, данные нужно будет сгенерировать самостоятельно и воспользоваться шаблонизацией для их отрисовки в разметку страницы.

В тексте задания в шаблонах вы будете встречать текст в фигурных скобках, выделенный жирным начертанием. Такой текст будет означать, что на месте этого текста должно появиться значение, которое вы возьмете из данных.

Например, в шаблоне может быть написано `<div>{{x}}</div>`, и это будет значить, что `{{x}}` нужно заменить на значение переменной `x`. Если она будет равна `100`, то разметка должна выглядеть как `<div>100</div>`. Фигурные скобки

в этой записи ничего не значат, они просто показывают, что закончилась разметка и в этом месте будут стоять данные. Сами фигурные скобки переносить в разметку не нужно.

**Подготовка**
В форке личного проекта создайте ветку `module3-task1` и в этой ветке выполните следующие шаги:

Кексобукинг

Создайте файл `js/map.js` в вашем личном проекте. Это файл, в котором вы будете вести работу с похожими объявлениями на карте.
**Имена файлов, функций и пр. в заданиях имеют рекомендательный характер**.
При выполнении задания необязательно создавать файлы, названия которых указаны в названии. Вы можете самостоятельно формировать любую структуру проекта по своему усмотрению, главное, чтобы проект выполнял ТЗ и соответствовал критериям.
В файле `index.html` подключите ваш файл при помощи тега `script`.
**Задача**
В файле `map.js`:

1. Создайте массив, состоящий из 8 сгенерированных JS объектов, которые будут описывать похожие объявления неподалёку. Структура объектов должна быть следующей:

```
{
  "author": {
    "avatar": строка, адрес изображения вида img/avatars/user{{xx}}.png, где {{xx}} это число от 1 до 8 с ведущим нулём. Например, 01, 02 и т. д. Адреса изображений не повторяются
  },

  "offer": {
    "title": строка, заголовок предложения, одно из фиксированных значений "Большая уютная квартира", "Маленькая неуютная квартира", "Огромный прекрасный дворец", "Маленький ужасный дворец", "Красивый гостевой домик", "Некрасивый негостеприимный домик", "Уютное бунгало далеко от моря", "Неуютное бунгало по колено в воде". Значения не должны повторяться.
    "address": строка, адрес предложения, представляет собой запись вида "{{location.x}}, {{location.y}}", например, "600, 350"
    "price": число, случайная цена от 1000 до 1 000 000
    "type": строка с одним из четырёх фиксированных значений: palace, flat, house или bungalo
    "rooms": число, случайное количество комнат от 1 до 5
    "guests": число, случайное количество гостей, которое можно разместить
    "checkin": строка с одним из трёх фиксированных значений: 12:00, 13:00 или 14:00,
    "checkout": строка с одним из трёх фиксированных значений: 12:00, 13:00 или 14:00
    "features": массив строк случайной длины из ниже предложенных: "wifi", "dishwasher", "parking", "washer", "elevator", "conditioner",
    "description": пустая строка,
    "photos": массив из строк "http://o0.github.io/assets/images/tokyo/hotel1.jpg", "http://o0.github.io/assets/images/tokyo/hotel2.jpg" и "http://o0.github.io/assets/images/tokyo/hotel3.jpg" расположенных в произвольном порядке
  },

```

```
21.   "location": {
22.     «x»: случайное число, координата x метки на карте. Значение ограничено раз
    мерами блока, в котором перетаскивается метка.
23.     «y»: случайное число, координата y метки на карте от 130 до 630.
24.   }
25. }
```

26. У блока `.map` уберите класс `.map--faded`.

> Это временное решение, этот класс переключает карту из неактивного состояния в активное. В последующих заданиях, в соответствии с ТЗ вы будете переключать режимы страницы: неактивный, в котором карта и форма заблокированы и активный режим, в котором производится ввод данных и просмотр похожих объявлений. Сейчас, для тестирования функции генерации похожих объявлений мы временно сымитируем активный режим, а в последующих разделах запрограммируем его полностью.

27. На основе данных, созданных в первом пункте, создайте DOM-элементы, соответствующие меткам на карте, и заполните их данными из массива. Итоговую разметку метки `.map__pin` можно взять из шаблона `.map__card`.
    - У метки должны быть следующие данные:
    - Координаты:`style="left: {{location.x}}px; top: {{location.y}}px;"`
    - `src="{{author.avatar}}"`
    - `alt="{{заголовок объявления}}"`

> **Обратите внимание**
> Координаты X и Y, которые вы вставите в разметку, это не координаты левого верхнего угла блока метки, а координаты, на которые указывает метка своим острым концом. Чтобы найти эту координату нужно учесть размеры элемента с меткой.

28. Отрисуйте сгенерированные DOM-элементы в блок `.map__pins`. Для вставки элементов используйте `DocumentFragment`.
29. На основе первого по порядку элемента из сгенерированного массива и шаблона `.map__card` создайте DOM-элемент объявления, заполните его данными из объекта и вставьте полученный DOM-элемент в блок `.map` перед блоком`.map__filters-container`:
    - Выведите заголовок объявления `offer.title` в заголовок `.popup__title`.
    - Выведите адрес `offer.address` в блок `.popup__text--address`.
    - Выведите цену `offer.price` в блок `.popup__text--price` строкой вида `{{offer.price}}₽/ночь`. Например, 5200₽/ночь.
    - В блок `.popup__type` выведите тип жилья `offer.type`: `Квартира` для `flat`, `Бунгало` для `bungalo`, `Дом` для `house`, `Дворец` для `palace`.
    - Выведите количество гостей и комнат `offer.rooms` и `offer.guests` в блок `.popup__text--capacity`строкой вида `{{offer.rooms}} комнаты для {{offer.guests}} гостей`. Например, 2 комнаты для 3 гостей.
    - Время заезда и выезда `offer.checkin` и `offer.checkout` в блок `.popup__text--time` строкой вида `Заезд после {{offer.checkin}}, выезд до {{offer.checkout}}`. Например, заезд после 14:00, выезд до 12:00.
    - В список `.popup__features` выведите все доступные удобства в объявлении.
    - В блок `.popup__description` выведите описание объекта недвижимости `offer.description`.

- В блок `.popup__photos` выведите все фотографии из списка `offer.photos`. Каждая из строк массива `photos` должна записываться как `src` соответствующего изображения.

Замените `src` у аватарки пользователя — изображения, которое записано в `.popup__avatar` — на значения поля `author.avatar` отрисовываемого объекта.

Пример того, как выглядит страница до, в процессе и в итоге выполнения задания:


---

## Кекстаграм

Создайте файл `js/pictures.js` в вашем личном проекте. Это файл, в котором вы будете вести работу с фотографиями, размещенными другими участниками. В файле `index.html` подключите ваш файл при помощи тега `script`.

**Задача**

В файле `pictures.js`:

1. Создайте массив, состоящий из 25 сгенерированных JS объектов, которые будут описывать фотографии, размещённые другими пользователями:
   - `url`**, строка** — адрес картинки вида `photos/{{i}}.jpg`, где `{{i}}` это число от 1 до 25. Адреса картинок не должны повторяться.
   - `likes`**, число** — количество лайков, поставленных фотографии. Случайное число от 15 до 200
   - `comments`**, массив строк** — список комментариев, оставленных другими пользователями к этой фотографии. Комментарий должен генерироваться случайным образом. Для каждого комментария нужно взять одно или два случайных предложений из предложенных ниже:
     - Всё отлично!
     - В целом всё неплохо. Но не всё.
     - Когда вы делаете фотографию, хорошо бы убирать палец из кадра. В конце концов это просто непрофессионально.
     - Моя бабушка случайно чихнула с фотоаппаратом в руках и у неё получилась фотография лучше.
     - Я поскользнулся на банановой кожуре и уронил фотоаппарат на кота и у меня получилась фотография лучше.
     - Лица у людей на фотке перекошены, как будто их избивают. Как можно было поймать такой неудачный момент?!
   - `description`**, строка**, в которой есть одно из следующих предложений:
     - Тестим новую камеру!
     - Затусили с друзьями на море
     - Как же круто тут кормят
     - Отдыхаем...
     - Цените каждое мгновенье. Цените тех, кто рядом с вами и отгоняйте все сомненья. Не обижайте всех словами.....
     - Вот это тачка!

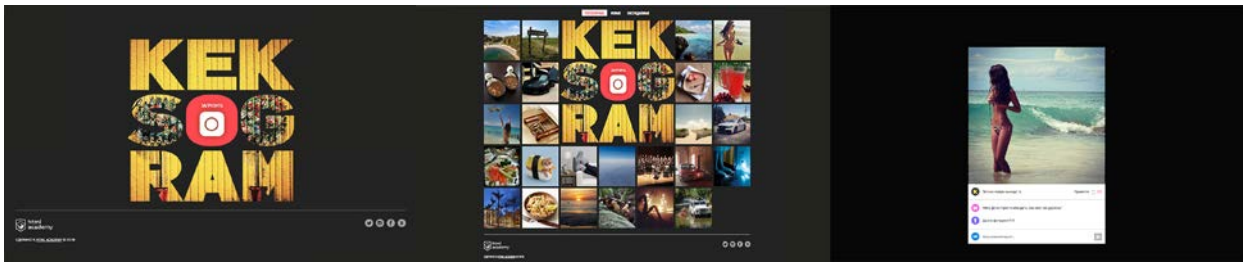
2. На основе данных, созданных в предыдущем пункте и шаблона `#picture` создайте DOM-элементы, соответствующие фотографиям и заполните их данными из массива:
    - Адрес изображения `url` подставьте как `src` изображения.
    - Количество лайков `likes` подставьте как текстовое содержание элемента `.picture__stat--likes`.
    - Количество комментариев `comments` подставьте как текстовое содержание элемента `.picture__stat--comments`.

3. Отрисуйте сгенерированные DOM-элементы в блок `.pictures`. Для вставки элементов используйте `DocumentFragment`.
4. Покажите элемент `.big-picture`, удалив у него класс `.hidden` и заполните его данными из первого элемента сгенерированного вами массива:
    - Адрес изображения `url` подставьте как `src` изображения внутри блока`.big-picture__img`.
    - Количество лайков `likes` подставьте как текстовое содержание элемента `.likes-count`.
    - Количество комментариев `comments` подставьте как текстовое содержание элемента `.comments-count`.
    - Список комментариев под фотографией: коментарии должны вставляться в блок `.social__comments`. Разметка каждого комментария должна выглядеть так:

```
<li class="social__comment social__comment--text">
  <img class="social__picture" src="img/avatar-
    {{случайное число от 1 до 6}}.svg"
    alt="Аватар комментатора фотографии"
    width="35" height="35">
    <p class="social__text">{{текст комментария}}</p>
      </li>
```

    - Описание фотографии `description` вставьте строкой в блок `.social__caption`.
5. Спрячьте блоки счётчика комментариев `.social__comment-count` и загрузки новых комментариев `.social__loadmore`, добавив им класс `.visually-hidden`.

Пример того, как выглядит страница до, в процессе и в итоге выполнения задания:



---

**Требования к коду**

Код должен быть разделён на отдельные функции. Стоит отдельно объявить функцию генерации случайных данных, функцию создания DOM-элемента на основе JS-объекта, функцию заполнения блока DOM-элементами на основе массива JS-объектов. Пункты задания примерно соответствуют функциям, которые вы должны создать.