Подсказки, лайфхаки и JavaScript

Студенты факультета фронтенда подготовили для вашего дипломного проекта HTML-шаблоны и JS-код. Вам нужно будет адаптировать шаблоны для Django (фронтендеры мало что знают о бэкенде, не забывайте об этом!).

Чтобы сберечь нервы — сначала полистайте код шаблонов и JS в репозитории, и только потом читайте этот урок.

Основные принципы

На протяжении всего курса запросы к серверу вы отправляли из адресной строки браузера или через формы. После этого страница перезагружалась — и отрисовывалась новая страница, которую сервер вернул в ответ на запрос.

JavaScript (JS) может творить чудеса: отправлять запросы без перезагрузки страницы, получать ответ сервера и на основе этого ответа перерисовывать страницу, изменяя HTML-код в браузере клиента. **JavaScript** — это язык сценариев, которые исполняются прямо в браузере, на стороне клиента.

Код JS обычно подключается к HTML-странице ссылкой в теге `<head>`. JS-скрипты могут отслеживать действия пользователя на странице и при каких-то событиях («пользователь нажал кнопку *Добавить рецепт в избранное*») выполнять определённые действия («отправить POST-запрос на определённый URL, в запросе передать ID добавленного рецепта»).

Запросы, которые JS отправляет к серверу, собраны в файле `js/api/Api.js`. Проверьте и подготовьте этот файл для работы в проекте:

- Проверьте и, при необходимости, измените URL, на который отправляются запросы.

- Обрабатывать запросы можно обычными средствами Django, а можно подключить Django Rest Framework.
- Если вы не используете Django Rest Framework — в заголовки запросов `POST` и `DELETE` нужно включить csrf-токен.

  Для этого в любое место HTML-шаблона, из которого JS отправляет запрос, добавьте токен: `{% csrf_token %}`. Шаблонизатор заменит этот тег на скрытое `hidden` поле: `<input type="hidden" name="csrfmiddlewaretoken" value="здесь_строка_токена">`.

  Теперь JS сможет найти токен в коде страницы по имени (по аргументу `name`): `document.getElementsByName('csrfmiddlewaretoken')[0].value`
  Токен нужно включить в заголовок JS-запроса: `'X-CSRFToken': document.getElementsByName('csrfmiddlewaretoken')[0].value`.
- Не стесняйтесь редактировать запросы, если посчитаете это необходимым.

При добавлении/удалении рецептов в список покупок и в избранное на сервер должен отправляться ID рецепта.

Этот ID прописывается в аргументе `data-id` элемента с классом `card`. Например `<div class="card" data-id="1">`. При нажатии на кнопку *Добавить в список* или *Добавить в избранное* JS отправит запрос, в котором передаст ID выбранного рецепта.

Счётчик списка покупок расположен в HTML-элементе с `id="counter"`: `<span class="badge badge_style_blue nav__badge" id="counter">4</span>`.

После добавления/удаления рецепта отправляется запрос на сервер, в ответе сервера ожидается `{"success": true/false}` в зависимости от того, успешно ли обработан запрос. Если всё прошло успешно — JS заменит значение счётчика.

При загрузке страницы не забудьте установить актуальное значение счётчика.

Кнопка добавления в избранное, «звёздочка», описана в HTML-коде элементом с классом `icon-favorite`.

В коде неактивной кнопки должен стоять атрибут `data-out`, а если кнопка активна — этот атрибут удаляется JS-скриптом. По наличию этого атрибута JS понимает, надо ли отправить запрос на добавление в избранное или на удаление. Дополнительно для активной кнопки добавляется класс `icon-favorite_active` (он отвечает за внешний вид звёздочки):

- Код звёздочки на рецепте, который **не добавлен** в избранное:

  ```
  <button class="button button_style_none" name="favorites" data-out><span class="icon-favorite"></span></button>
  ```

- Код звёздочки на рецепте, который **добавлен** в избранное:

  ```
  <button class="button button_style_none" name="favorites"><span class="icon-favorite icon-favorite_active"></span></button>
  ```

Кнопки фильтрации по тегам — это ссылки, для них в атрибуте `href` можно прописать нужный URL запроса — например, с добавлением GET-параметра. В результате клик по фильтру отправит пользователя на страницу с отфильтрованными рецептами. В этой части никакого JS, обычный GET-запрос.

Не забудьте, что на странице с отфильтрованными рецептами должны быть выделены кнопки тех фильтров, по которым проведена фильтрация.

Кнопка добавления/удаления товара в список покупок:

- Если рецепт **не добавлен** в список покупок: `<button class="button button_style_light-blue" name="purchases" data-out><span class="icon-plus button__icon"></span>Добавить в покупки</button>`. Обратите внимание на классы и атрибут `data-out`
- Если рецепт **добавлен** в список покупок: `<button class="button button_style_light-blue-outline" name="purchases"><span class="icon-check button__icon"></span> Рецепт добавлен</button>`. Относительно недобавленного в список рецепта изменены классы у кнопки и значка, отсутствует `data-out` и изменен текст кнопки
- Не забудьте про `data-out`: он должен быть только на неактивной кнопке.

Логика запросов JavaScript написана так, что в ответе сервера ожидается `{"success": true/false}` для определения успешности запроса. Если запрос выполнен успешно — сработают JS-скрипты, которые изменят HTML-

страницу: перекрасят звёздочку *Добавить в избранное* или заменят кнопку *Добавить в список покупок* на *Удалить из списка покупок*.

В форме создания рецепта при вводе в поле «Название ингредиента» на сервер отправляется запрос с введённой в форму частью названия продукта. В ответ ожидается список из словарей с ключами `title` и `dimension`. Например, на сервер отправляется строка `гре`, в ответ должен придти такой список: `[{"title": "гренадин", "dimension": "г"}, {"title": "гренки", "dimension": "г"}]`.

Не забудьте указать HTML-шаблонах в формах атрибут `method` и добавить csrf-токены.

Если вы не будете настраивать работу со списком покупок для неавторизованных пользователей (это необязательное задание) — не забудьте скрыть от них ссылку меню *Список покупок* и кнопку *Добавить в покупки*.

В архиве с шаблонами нет страницы с информацией об успешной регистрации. Как поступить — решайте самостоятельно:

- Можно добавить новый шаблон с текстом «Вы зарегистрированы и вошли в систему. Спасибо за регистрацию и приятного аппетита!» и после регистрации перенаправлять пользователя на эту страницу.
- Можно просто перенаправлять пользователя на главную страницу, авторизовав его. Но в такой ситуации невнимательный пользователь не сразу поймёт, что он авторизован.

Некоторые шаблоны повторяют друг друга с минимальными отличиями, их можно уверено изменять, объединять и структурировать так, как вам будет удобно. Здесь будет уместно вспомнить про инструкции `{% include %}` и `{% extends %}`.

Не забывайте подключать к HTML нужные js-файлы: для большинства HTML-шаблонов подготовлен одноимённый файл с расширением *.js*.

Обратите внимание, что к разным шаблонам могут быть подключены разные css- и js-файлы.

Не забывайте выделять нужный пункт меню в шапке сайта в зависимости от страницы, на которой находится пользователь.

Какие сложности могут возникнуть

Одна из первых сложностей может возникнуть в списке рецептов при взаимодействии пагинатора с тегами. Если фильтрацию по тегам вы будете настраивать через GET-параметры, не забывайте передавать их при переходе на следующую страницу пагинатора: при этом должны передаваться два GET-параметра — теги и номер страницы.

При отправке данных из формы создания рецепта:

- Теги передаются из формы на сервер как `"lunch": "on"`, если тег выбран и не передаются вовсе — если тег не выбран.
- Ингредиенты и их количество передаются в нумерованном виде, на сервер отправляется не ID ингредиента, а его название. Если при создании рецепта пользователь сперва добавил ингредиент, а затем удалил его — нумерация будет сбита, будет пропущен один номер.

  Например: `{"nameIngredient_1": "Картошка", "valueIngredient_1": "100", "nameIngredient_3": "Курица", "valueIngredient_3": "400",}`, здесь пропущена пара с номером 2, так тоже может быть.

Вам понадобится отдать пользователю файл со списком ингредиентов. Такого вы не делали, но найти решение не составит большого труда. View-функции умеют возвращать через *HttpResponse* не только текст, но и файлы.