

**1. Создаем функцию `getStorageUsers`, которая запрашивает/создает в `localStorage` свойство `users`, в котором находится массив зарегистрированных юзеров.**

1. Если в `localStorage` данного свойства нет, то создаем его и записываем в него исходный массив `USERS`.
2. Функция `getStorageUsers` возвращает массив зарегистрированных юзеров.
3. При заходе на страницу вызываем ее и полученный массив сохраняем в глобальную переменную `const storageUsers = getStorageUsers();`

**Для страницы `login.html` пишем логику логина/регистрации пользователя.**

**2. Login form**

1. Достаемся к форме `const LoginForm = document.querySelector('#LoginForm');` и навешиваем обработчик на событие `submit`.
2. Сохраняем введенные юзером значения полей:
  1. `let email = e.target.querySelector('input[data-name=«email»]').value;`
  2. `let password = e.target.querySelector('input[data-name=«password»]').value;`
3. В массиве `storageUsers` ищем пользователя у которого email равен значению с формы.
4. Если юзер с таким email в `localStorage` НЕ найден, то в форме логине достаемся к `div.error`, добавляем ему класс `active` и устанавливаем `innerHTML = 'Invalid email'`.
5. Если юзер с таким email в `localStorage` найден, то далее проверяем, равны ли поля `password`. Если поля НЕ совпадают, то в форме логине достаемся к `div.error`, добавляем ему класс `active` и устанавливаем `innerHTML = 'Invalid password'`.
6. Если юзер с таким email в `localStorage` найден и пароли совпадают, то данному юзеру, которого нашли в `localStorage`, меняем свойство `status` на `true` и в `localStorage` обратно отправляем свойство `users`.  
`localStorage.setItem('users', JSON.stringify(storageUsers));` После этого редиректим юзера на страницу его аккаунта. `document.location.href = 'account.html';`
7. Так же, при сабмите формы нужно убирать для `div.error` класс `active`, если он ранее был добавлен.

**3. Register form**

1. Достаемся к форме `const RegistrationForm = document.querySelector('#RegistrationForm');` и навешиваем обработчик на событие `submit`.
2. Сохраняем введенные юзером значения полей:
  1. `let name = e.target.querySelector('input[data-name=«name»]').value;`
  2. `let email = e.target.querySelector('input[data-name=«email»]').value;`
  3. `let password = e.target.querySelector('input[data-name=«password»]').value;`
  4. `let passwordVerify = e.target.querySelector('input[data-name=«passwordVerify»]').value;`
3. Если значения полей `password` и `passwordVerify` не равны, то в форме регистрации достаемся к `div.error`, добавляем ему класс `active` и устанавливаем `innerHTML = 'Password not matches!'`.
4. Если значения полей `password` и `passwordVerify` равны, то:
  1. В массиве `storageUsers` ищем пользователя у которого email равен значению с формы.

2. Если юзер с таким email в localStorage найден, то в форме регистрации достаемся к div.error, добавляем ему класс active и устанавливаем innerHTML = `User with email \${email} already exist!`.
3. Если юзер с таким email в localStorage НЕ найден, то сохраняем все введенные в форме значения (кроме passwordVerify) в объект user. Добавляем свойства status: true, orders: [], favourites: [], shoppingCart: [].  
let user = { name: name, email: email, password: password, status: true, orders: [], favourites: [], shoppingCart: [] } Запускаем его в массив storageUsers. Обратно отправляем в localStorage свойство users.  
localStorage.setItem(`users`, JSON.stringify(storageUsers)); После этого редиректим юзера на страницу его аккаунта. document.location.href = `account.html`;
4. Так же, при сабмите формы нужно убирать для div.error класс active, если он ранее был добавлен.

Делаем глобальную переменную **userInSession**, в которой будет храниться объект залогиненного пользователя.

Для этого: в массиве **storageUsers** ищем юзера, у которого свойство **status === true**.

**const userInSession = storageUsers.find(user => user.status === true);**

#### 4. Работа с Шапкой сайта

1. Достаемся к блоку где выводится либо Log in, либо Имя залогиненного юзера. **const headerUser = document.querySelector(`#headerUser`);**
2. Достаемся к блоку где выводится сердечко (количество избранных товаров залогиненного юзера). **const headerFavourites = document.querySelector(`#headerFavourites`);**
3. Достаемся к блоку где выводится корзина (количество добавленных в корзину товаров залогиненного юзера). **const headerShoppingCart = document.querySelector(`#headerShoppingCart`);**
4. Достаемся к кнопке Log out. **const headerLogout = document.querySelector(`#headerLogout`);**
5. Если **userInSession** существует, то:
  1. В блоке **headerUser**:
    1. Вместо текста Log in выводим имя залогиненного юзера  
headerUser.innerHTML = userInSession.name;
    2. Вместо ссылки на login.html пишем ссылку на страницу account.html.  
headerUser.href = `account.html`;
  2. В блоке **headerFavourites**:
    1. Вместо ссылки на login.html пишем ссылку на страницу favourites.html.  
headerFavourites.href = `favourites.html`;
    2. В блок **const headerFavouritesCount = document.querySelector(`#headerFavouritesCount`);** (количество избранных товаров залогиненного юзера) выводим длину массива свойства favourites. headerFavouritesCount.innerHTML = userInSession.favourites.length;
  3. В блоке **headerShoppingCart**:
    1. Вместо ссылки на login.html пишем ссылку на страницу shoppingCart.html. headerShoppingCart.href = `shoppingCart.html`;
    2. В блок **const headerShoppingCartCount = document.querySelector(`#headerShoppingCartCount`);** (количество добавленных в корзину товаров залогиненного юзера) выводим длину



массива свойства shoppingCart. headerShoppingCartCount.innerHTML =  
userInSession.shoppingCart.length;

4. Кнопке **Log out** добавляем класс active для отображения.  
headerLogout.classList.add('active');

#### 5. Навешиваем обработчик на кнопку Log out.

1. Навешиваем обработчик на событие click по кнопке **Log out**:
  1. В массиве **storageUsers** ЗАНООВО ищем юзера, у которого status === true.  
**const userInSession = storageUsers.find(user => user.status === true);**
  2. Для найденного юзера меняем свойство status на false. userInSession.status = false;
  3. В localStorage обратно отправляем свойство users.  
localStorage.setItem('users', JSON.stringify(storageUsers));
  4. После этого редиректим юзера на главную страницу index.html.  
document.location.href = 'index.html';

#### 6. Для страницы index логика вывода товаров.

1. Достаемся к блоку вывода всех товаров. **const categoriesContainer = document.querySelector('#categoriesContainer');**
2. Запускаем цикл forEach по исходному массиву PRODUCTS.
3. При работе с каждым элементом массива PRODUCTS определяем в какую section-катеорию его выводить. Если на странице еще не создана section для данной категории, то создам ее и выводим в **categoriesContainer**. После этого рендерим блок с товаром и выводим во все нужные section-категории.
4. Вывод каждого товара:
  1. Кнопка Сердечка 
    1. Если **userInSession** и товар в favourites, то картинка сердечка **images/product\_\_favourite—true.png**
    2. Если **userInSession** и товар не в favourites ИЛИ если юзер НЕ залогинен, то картинка сердечка **images/product\_\_favourite.png**
    3. При нажатии на сердечко:
      1. Если пользователь НЕ залогинен, то редиректим его на страницу login.html
      2. Если пользователь залогинен и товар ранее не был добавлен в favourites, то:
        1. Меняем картинку сердечка на images/product\_\_favourite—true.png
        2. Добавляем id товара в массив favourites в localStorage для данного юзера.
        3. В шапке сайта в блоке с выводом количества избранных товаров меняем значение.
      3. Если пользователь залогинен и товар ранее был добавлен в favourites, то:
        1. Меняем картинку сердечка на images/product\_\_favourite.png
        2. Удаляем id товара с массива favourites в localStorage для данного юзера.
        3. В шапке сайта в блоке с выводом количества избранных товаров меняем значение.
  2. Кнопка Корзины 
    1. Если пользователь не залогинен, то код кнопки будет таким: `<button class="product__cart"></button>`. При нажатии редиректим пользователя на страницу login.html

2. Если **userInSession** и товар ранее не был добавлен в массив shoppingCart, то:
  1. Код кнопки: `<button class="product__cart"></button>`.
  2. При нажатии:
    1. Добавляем объект товара в массив shoppingCart в localStorage для данного юзера. Вид объекта товара: {id: 1, count: 1}, где id – id товара, count – количество товара.
    2. В шапке сайта в блоке с выводом количества добавленных в корзину товаров меняем значение.
3. Если **userInSession** и товар ранее был добавлен в массив shoppingCart, то:
  1. Код кнопки: `<button class="product__cart product__cart--in"></button>`.
  2. При нажатии:
    1. Удаляем объект товара с массива shoppingCart в localStorage для данного юзера.
    2. В шапке сайта в блоке с выводом количества добавленных в корзину товаров меняем значение.
4. Если **userInSession** и товар ранее был добавлен в массив orders, то:
  1. Код кнопки: `<button class="product__cart product__cart--ordered"></button>`.
  2. При нажатии ничего не происходит.

Доступ к страницам [account.html](#), [favourites.html](#), [shoppingCart.html](#) возможен ТОЛЬКО если юзер ЗАЛОГИНЕН. Все данные для юзера берем с **userInSession**.

#### 7. Для страницы [account.html](#) логика работы с блоком **My Info**.

1. Достаемся к input где выводится имя залогиненного юзера. **const userInfoName = document.querySelector(`#userInfoName`);** Меняем value поля `userInfoName.value = userInSession.name;`
2. Достаемся к td где выводится email залогиненного юзера. **const userInfoEmail = document.querySelector(`#userInfoEmail`);** Меняем innerHTML поля `userInfoEmail.innerHTML = userInSession.email;`

#### 8. Редактирование имени юзера в блоке **My Info**.

1. Достаемся к форме userInfo. **const userInfo = document.querySelector(`#userInfo`);**
2. При submit формы сохраняем значение поля **let name = userInfoName.value;**
3. В массиве **storageUsers** ЗАНОВО ищем юзера, у которого `status === true`. **const userInSession = storageUsers.find(user => user.status === true);**
4. Для найденного юзера меняем свойство name на значение с формы `userInSession.name = name;`
5. В localStorage обратно отправляем свойство users. `localStorage.setItem(`users`, JSON.stringify(storageUsers));`
6. После перезагружаем страничку `location.reload();`

#### 9. Удаление юзера в блоке **My Info**.



1. Достаемся к кнопке deleteAcc. **const deleteAcc = document.querySelector(`#deleteAcc`);**
2. Навешиваем обработчик на событие click. При клике на кнопку:
3. В массиве **storageUsers** ищем ИНДЕКС юзера, у которого `status === true`. **const userInSessionIndex = storageUsers.findIndex(user => user.status === true);**

4. Имея порядковый номер залогиненного юзера в массиве `storageUsers` с помощью метода `splice()` вырезаем юзера.  
`storageUsers.splice(userInSessionIndex, 1);`
5. В `localStorage` обратно отправляем свойство `users`.  
`localStorage.setItem(`users`, JSON.stringify(storageUsers));`
6. После этого редиректим на главную страницу `index.html`.  
`document.location.href = `index.html`;`

#### 10. Для страницы `account.html` логика работы с блоком `Ordered Items`.


1. Достаемся к таблице заказов юзера `const orderTable = document.querySelector(`#orderTable`);`
2. Если у залогиненного юзера в свойстве `orders` находится НЕ пустой массив, то проходимся по нему и каждый товар выводим в таблицу `orderTable` в виде `tr`. Пример объекта в массиве `orders`: `{id: 10, count: 2}`, где `id` - соответствует `id` товара в исходном массиве `PRODUCTS`, и `count` - количество единиц товара, которое приобрёл данный юзер.
3. **Общая сумма за каждый товар считается исходя из трех параметров: цена за единицу товара, скидка на товар и количество приобретенных единиц товара.** Например, если юзер приобрел товар стоимостью \$200, скидка на товар -25% и количество приобретенных единиц 2, то общая сумма будет высчитываться по формуле:  $(200 - (200*25)/100) * 2 = 300$ .

#### 11. Для страницы `favourites.html` логика работы с блоком `Favourite Items`.

1. Если `userInSession` существует, то:
  1. Достаемся к таблице избранных товаров юзера `const favouriteTable = document.querySelector(`#favouriteTable`);`
  2. Если у залогиненного юзера в свойстве `favourites` находится НЕ пустой массив, то проходимся по нему и каждый товар выводим в таблицу `favouriteTable` в виде `tr`. Пример данных в массиве `favourites`: `[10, 12, 1]`, где каждый элемент массива соответствует `id` товара в исходном массиве `PRODUCTS`.
3. Кнопка Сердечка  При нажатии на сердечко:
  1. Удаляем `id` товара с массива `favourites` в `localStorage` для данного юзера.
  2. В шапке сайта в блоке с выводом количества избранных товаров меняем значение.
  3. Удаляем `tr` товара
4. Кнопка Корзины 
  1. Если товар ранее НЕ был добавлен в массив `shoppingCart`, то:
    1. Код кнопки: `<button class="product__cart"></button>`
    2. При нажатии:
      1. Добавляем объект товара в массив `shoppingCart` в `localStorage` для данного юзера. Вид объекта товара: `{id: 1, count: 1}`, где `id` - `id` товара, `count` - количество товара.
      2. В шапке сайта в блоке с выводом количества добавленных в корзину товаров меняем значение.
  2. Если товар ранее был добавлен в массив `shoppingCart`, то:
    1. Код кнопки: `<button class="product__cart product__cart--in"></button>`
    2. При нажатии:

1. Удаляем объект товара с массива shoppingCart в localStorage для данного юзера.
2. В шапке сайта в блоке с выводом количества добавленных в корзину товаров меняем значение.

## 12. Для страницы shoppingCart.html логика работы с блоком Items in Shopping Cart.

1. Если **userInSession** существует, то:
  1. Достаемся к таблице товаров добавленных в корзину **const shoppingCartTable = document.querySelector(`#shoppingCartTable`);**
  2. Если у залогиненного юзера в свойстве shoppingCart находится НЕ пустой массив, то проходимся по нему и каждый товар выводим в таблицу **shoppingCartTable** в виде tr. Пример объекта в массиве shoppingCart: {id: 10, count: 2}, где id - соответствует id товара **в исходном массиве PRODUCTS**, и count - количество единиц товара, которое выбрал данный юзер.
  3. При изменении единиц товара:
    1. Пересчитываем поле Total для данного товара.
    2. Пересчитываем поле Order total в блоке My Order Summary.
    3. В localStorage для данного юзера в массиве shoppingCart для объекта данного товара меняем свойство count.
4. Кнопка Корзины  При нажатии:
  1. Удаляем объект товара с массива shoppingCart в localStorage для данного юзера.
  2. Пересчитываем поле Order total в блоке My Order Summary.
  3. В шапке сайта в блоке с выводом количества добавленных в корзину товаров меняем значение.

## 13. Для страницы shoppingCart.html логика работы с блоком My Order Summary.

1. Достаемся к форме **const orderSummary = document.querySelector(`#orderSummary`);** При submit формы:
  1. В массиве **storageUsers** ищем объект юзера, у которого status === true. **const userInSession = storageUsers.find(user => user.status === true);**
  2. У найденного юзера проходимся по массиву shoppingCart и каждый объект с массива shoppingCart:
    1. Запушиваем в массив orders.
    2. Индекс текущего товара удаляем с массива favourites.
  3. Очищаем массив shoppingCart. **userInSession.shoppingCart = [];**
  4. В localStorage обратно отправляем свойство users. **localStorage.setItem(`users`, JSON.stringify(storageUsers));**
  5. После этого редиректим на страницу аккаунта orders.html. **document.location.href = `orders.html`;**