Тема 15. Пользовательская интерактивность .

хекслет колледж

Цель занятия:

Сформировать у студентов понимание принципов пользовательской интерактивности и научить создавать простые интерактивные элементы интерфейса с использованием JavaScript и DOM.

Учебные вопросы:

1. Понятие пользовательской интерактивности
2. Обратная связь для пользователя
3. Управление состоянием элементов интерфейса
4. Интерактивные элементы страницы
5. Динамическое изменение интерфейса
6. Простые интерактивные сценарии

1. Понятие пользовательской интерактивности

Пользовательская интерактивность — это способность веб-страницы реагировать на действия пользователя и изменять своё поведение или внешний вид в ответ на эти действия.
Проще говоря, интерактивность — это диалог между пользователем и сайтом.

Зачем нужна интерактивность

Без интерактивности веб-страница является статической: пользователь может только просматривать информацию.
Интерактивность позволяет:

- управлять элементами страницы (кнопками, формами, списками);
- получать обратную связь на действия пользователя;
- делать интерфейс удобным и понятным;
- снижать количество ошибок при вводе данных.

Роль JavaScript

JavaScript является основным инструментом создания интерактивности.
С его помощью можно:

- обрабатывать события (клики, ввод текста, отправку форм);
- изменять элементы DOM;
- обновлять содержимое страницы без её перезагрузки;
- управлять состоянием интерфейса.

Понятие

Пример простой интерактивности:

```
const button = document.getElementById("btn");
const text = document.getElementById("txt")

button.addEventListener('click', function() {
  text.textContent = 'Кнопка нажата';
});
```

Примеры пользовательской интерактивности

К базовым элементам интерактивного интерфейса относятся:

- кнопки, реагирующие на клики;
- поля ввода с проверкой данных;
- выпадающие списки;
- сообщения об ошибках и подсказки;
- переключение состояния элементов (показать / скрыть).

Интерактивность и пользовательский опыт

Интерактивность напрямую влияет на удобство использования сайта (User Experience, UX).
Хорошая интерактивность:

- понятна пользователю;
- даёт своевременную обратную связь;
- не перегружает интерфейс.

Плохая интерактивность:

- не объясняет, что произошло;
- вызывает ошибки или непредсказуемое поведение;
- снижает удобство работы с сайтом.

Вывод:

- Пользовательская интерактивность — это реакция сайта на действия пользователя.
- Она реализуется с помощью JavaScript и DOM.
- Интерактивность делает веб-страницы удобными, понятными и функциональными.
- Грамотно реализованная интерактивность улучшает пользовательский опыт и качество интерфейса.

2. Обратная связь для пользователя

Обратная связь — это реакция интерфейса на действия пользователя, которая сообщает, что действие было выполнено, принято или отклонено.
Она помогает пользователю понять, что произошло и что делать дальше.

Зачем нужна обратная связь?

Обратная связь позволяет:

- подтвердить действие пользователя;
- предупредить об ошибке;
- направить пользователя;
- сделать работу с интерфейсом понятной и предсказуемой.

Без обратной связи пользователь не понимает:

- было ли выполнено действие;
- правильно ли он что-то ввёл;
- работает ли сайт вообще.

Виды обратной связи

1. Визуальная обратная связь

Проявляется через изменение внешнего вида элементов:

- изменение текста;
- изменение цвета;
- добавление или удаление CSS-классов;
- появление или скрытие элементов.

Пример:

```
const button = document.getElementById("btn");
const message = document.getElementById("msg")

button.addEventListener('click', function() {
  message.textContent = 'Данные сохранены';
  message.classList.add('success');
});
```

2. Текстовая обратная связь

Сообщения, объясняющие результат действия:

- сообщения об ошибках;
- подсказки;
- подтверждение выполнения действия.

Пример:

```
if (input.value === '') {
  error.textContent = 'Поле обязательно для заполнения';
}
```

3. Поведенческая обратная связь

Изменение поведения интерфейса:

- блокировка кнопки;
- отключение поля ввода;
- запрет повторного действия.

Пример:

button.disabled = true;

Когда нужно давать обратную связь

Обратная связь необходима, если:

- действие пользователя приводит к изменениям;
- есть вероятность ошибки;
- операция занимает время;
- пользователь ожидает подтверждения.

Примеры из практики:

- Подсветка неверно заполненного поля.
- Сообщение «Пароль слишком короткий».
- Изменение надписи на кнопке после нажатия.
- Блокировка кнопки «Отправить», пока форма не заполнена.

Основные принципы хорошей обратной связи:

- Ясность — сообщение должно быть понятным.
- Своевременность — появляться сразу после действия.
- Ненавязчивость — не перегружать интерфейс.
- Соответствие действию — реакция должна быть логичной.

Вывод:

- Обратная связь — ключевой элемент пользовательской интерактивности.
- Она помогает пользователю понимать действия интерфейса.
- Реализуется через изменения DOM, текста, стилей и свойств элементов.
- Грамотно реализованная обратная связь делает интерфейс удобным и понятным.

3. Управление состоянием элементов интерфейса

Состояние элемента интерфейса — это текущее положение или режим работы элемента в данный момент времени. Состояние отражает, как элемент выглядит и как он себя ведёт в зависимости от действий пользователя.

Понятие состояния

Примеры состояний элементов:

- кнопка: активна / неактивна;
- блок: показан / скрыт;
- форма: доступна / заблокирована;
- поле ввода: заполнено / пустое / с ошибкой.

Состояние может изменяться динамически в ходе работы пользователя с интерфейсом.

Зачем управлять состоянием?

Управление состоянием позволяет:

- делать интерфейс понятным;
- предотвращать ошибки пользователя;
- визуально отражать текущий статус элементов;
- связывать поведение элементов между собой.

Способы управления состоянием

1. Через CSS-классы (основной способ)
Состояние описывается классом, который управляет внешним видом элемента.

```css
.hidden {
  display: none;
}
```

```js
const button = document.getElementById("btn");
const box = document.getElementById("box")

button.addEventListener('click', function() {
  box.classList.toggle('hidden');
});
```

2. Через свойства элементов

Некоторые состояния задаются напрямую через свойства:

- disabled
- checked
- value
- hidden

Пример:

```
button.disabled = true;
```

3. Через изменение текста и атрибутов

Состояние можно показывать текстом или атрибутами:

```
button.textContent = 'Отключено';
```

Связь состояния и поведения

Состояние влияет на поведение элемента:

- отключённая кнопка не реагирует на клики;
- скрытый элемент недоступен пользователю;
- поле с ошибкой требует исправления.

Пример:

```
input.addEventListener('input', function() {
  submitButton.disabled = input.value === '';
});
```

Типичные примеры управления состоянием

- «Показать / скрыть» дополнительную информацию;
- блокировка кнопки отправки формы до заполнения полей;
- подсветка активного элемента меню;
- переключение состояния «включено / выключено».

Важные рекомендации:

- Логику состояния хранить в JavaScript, оформление — в CSS.
- Не менять стили напрямую, если можно использовать классы.
- Поддерживать понятные и логичные названия состояний (active, hidden, error).

Вывод:

- Состояние — это текущий режим элемента интерфейса.
- Управление состоянием осуществляется через классы, свойства и текст.
- Чёткое управление состоянием улучшает удобство и предсказуемость интерфейса.
- Это основа для создания интерактивных и удобных веб-страниц.

4. Интерактивные элементы страницы

Интерактивные элементы — это части веб-страницы, с которыми пользователь может взаимодействовать, а страница реагирует на эти действия.

Основные интерактивные элементы:

1. Кнопки (<button>)

- Основной способ инициировать действия.
- Примеры: отправка формы, показ/скрытие информации, выбор опции.

2. Переключатели и чекбоксы (<input type="checkbox">)

- Позволяют включать/выключать опции.
- Часто используют для фильтров или согласия с условиями.

3. Поля ввода (<input>, <textarea>)

- Пользователь может вводить данные.
- JavaScript может проверять и реагировать на ввод в реальном времени.

4. Списки и элементы выбора (<select>, <option>, <li>)

- Можно выбирать один или несколько вариантов.
- Например: меню, вкладки, карточки продуктов.

5. Ссылки (<a>)

- Помимо перехода на другую страницу, на них можно назначать обработчики событий, чтобы выполнять действия без перехода.

6. Блоки и контейнеры (, <section>)

- Используются как интерактивные области: кликабельные карточки, выпадающие меню, табы.

Как элементы делают страницу интерактивной?

1. Обработка событий

- Любой элемент, с которым можно взаимодействовать, реагирует на события (click, input, change, mouseover).

2. Изменение состояния элемента

- CSS-классы (active, hidden, completed) или свойства (disabled, checked) отражают текущее состояние.

3. Динамическое изменение содержимого и внешнего вида

- Изменение текста, HTML, стилей, атрибутов.
- Создание и удаление элементов через JavaScript.

Примеры использования:

- Кнопка «Показать/скрыть» меняет видимость блока.
- Чекбокс «Согласен с правилами» активирует кнопку отправки формы.
- Вкладки (карточек) подсвечиваются при выборе, меняется видимый контент.
- Поле ввода с обратной связью показывает подсказку «Поле заполнено» или «Поле пустое».

Вывод:

- Интерактивные элементы — это средства взаимодействия пользователя с веб-страницей.
- Любой элемент, на который можно нажать, ввести данные или выбрать опцию, может быть интерактивным.
- JavaScript в сочетании с CSS позволяет управлять состоянием, изменять содержимое и визуально реагировать на действия пользователя.

5. Динамическое изменение интерфейса

Динамическое изменение интерфейса — это изменение содержимого, структуры или внешнего вида страницы в ответ на действия пользователя без перезагрузки страницы. Именно динамика делает страницу «живой» и интерактивной.

Что можно изменять динамически?

С помощью JavaScript можно изменять:

- текст элементов;
- HTML-содержимое;
- атрибуты элементов;
- CSS-классы и стили;
- наличие элементов на странице (создание и удаление).

Изменение текста и содержимого

Изменение текста.
title.textContent = 'Новый заголовок';

- Меняет текст внутри элемента.
- Без вставки HTML.

Изменение HTML-содержимого
box.innerHTML = 'Важное сообщение';

- Позволяет вставлять HTML-разметку.
- Требует аккуратности (безопасность и структура).

Управление отображением элементов

Показ и скрытие
Через классы:

```
element.classList.add('hidden');
element.classList.remove('hidden');
```

Через переключение:

```
element.classList.toggle('hidden');
```

Изменение атрибутов элементов

```
input.setAttribute('placeholder', 'Введите имя');
button.disabled = true;
```

Используется для:

- блокировки элементов;
- подсказок;
- изменения поведения.

Динамическое создание элементов

```
const item = document.createElement('li');
item.textContent = 'Новый элемент списка';
list.appendChild(item);
```

- Создаёт новые элементы «на лету».
- Часто используется для списков, уведомлений, карточек.

Удаление элементов

```
item.remove();
```

- Полностью удаляет элемент из DOM.

Примеры пользовательских сценариев:

- Добавление комментария без перезагрузки страницы;
- Появление сообщения об ошибке;
- Показ дополнительной информации по кнопке;
- Обновление счётчика или текста при действии пользователя.

Важные рекомендации:

- Не изменять интерфейс без необходимости.
- Стараться использовать CSS-классы, а не inline-стили.
- Делать изменения логичными и предсказуемыми для пользователя.

Вывод:

- Динамическое изменение интерфейса — основа интерактивных веб-страниц.
- JavaScript позволяет изменять содержимое, внешний вид и структуру DOM.
- Такой подход улучшает удобство и скорость взаимодействия пользователя с сайтом.

6. Простые интерактивные сценарии

Интерактивный сценарий — это последовательность действий пользователя и реакций интерфейса на эти действия. Простые сценарии показывают, как с помощью JavaScript можно связать события, состояние и динамические изменения интерфейса.

Сценарий 1. Кнопка «Показать / скрыть»

HTML:

```html
<button id="toggleBtn">Показать</button>
<div id="text" class="hidden">Дополнительная информация</div>
```

CSS:

```css
.hidden {
  display: none;
}
```

JavaScript:

```
const button = document.getElementById('toggleBtn');
const text = document.getElementById('text');

button.addEventListener('click', function() {
  text.classList.toggle('hidden');
  button.textContent = text.classList.contains('hidden')
    ? 'Показать'
    : 'Скрыть';
});
```

Сценарий 2. Проверка заполнения поля ввода

HTML:

```
<input type="text" id="nameInput">
<p id="message"></p>
```

JavaScript:

```
const input = document.getElementById('nameInput');
const message = document.getElementById('message');

input.addEventListener('input', function() {
  if (input.value === '') {
    message.textContent = 'Поле не заполнено';
  } else {
    message.textContent = 'Поле заполнено';
  }
});
```

Сценарий 3. Блокировка кнопки

```html
<input type="input" id="nameInput">
<button id="button">send</button>
```

```js
const button = document.querySelector('button');
const input = document.querySelector('input');

input.addEventListener('input', function() {
  button.disabled = input.value === '';
});
```

Кнопка активируется только при наличии текста.

Сценарий 4. Изменение стиля по клику

Выбор активного элемента (карточка, кнопка, пункт меню)

HTML:

```
<div class="card">Тариф «Базовый»</div>
<div class="card">Тариф «Стандарт»</div>
<div class="card">Тариф «Премиум»</div>
```

CSS:

```css
.card {
  padding: 12px;
  border: 1px solid #ccc;
  margin-bottom: 8px;
  cursor: pointer;
}

.card.active {
  background-color: lightgreen;
  border-color: green;
}
```

JavaScript:

```
cards.forEach(function(card) {
  card.addEventListener('click', function() {
    cards.forEach(function(item) {
      item.classList.remove('active');
    });
    card.classList.add('active');
  });
});
```

Как это работает (пошагово):

1. Получаем элементы;
2. Назначаем обработчики;
3. Меняем состояние при клике:

- Сначала убираем active у всех элементов.
- Затем добавляем active только выбранному.
- Если класс active есть → он удаляется.
- Если нет → добавляется.

Вывод:

- Простые интерактивные сценарии — основа пользовательских интерфейсов.
- Они объединяют события, состояние и динамику DOM.
- Практика таких сценариев формирует понимание интерактивной логики.

Итоги лекции:

- Пользовательская интерактивность — это способность интерфейса реагировать на действия пользователя.
- Интерактивность делает сайт живым, удобным и понятным.
- Обратная связь важна, чтобы пользователь понимал результат своих действий.
- Состояние элементов интерфейса отражает их текущий режим (активен / неактивен, видим / скрыт и т.д.)
- Динамическое изменение интерфейса позволяет изменять DOM: текст, стили, атрибуты, создавать и удалять элементы.
- Простые интерактивные сценарии объединяют события, состояние и динамику DOM.
- Часто применяются: кнопки «Показать / скрыть», переключатели, проверка ввода, выбор активного элемента.
- Использование CSS-классов + JavaScript — лучший способ управлять состояниями и визуальной обратной связью.

Контрольные вопросы:

- Что такое пользовательская интерактивность?
- Зачем нужна обратная связь для пользователя?
- Какие виды обратной связи вы знаете?
- Что такое состояние элемента интерфейса?
- Как управлять состоянием элемента с помощью CSS и JavaScript?
- Какие изменения можно делать динамически через JavaScript?
- Что такое простой интерактивный сценарий?
- Приведите пример сценария «Показать / скрыть» с кнопкой и текстом.
- Как реализовать выделение активного элемента на странице?
- Почему рекомендуется использовать classList вместо прямого изменения стилей через

Домашнее задание:

https://ru.hexlet.io/courses/js-basics

хекслет колледж

@HEXLY.KZ