**1-урок**

1. **Основы веб-разработки и внедрения HTML, CSS и JavaScript.**

* **HTML (язык гипертекстовой разметки)** :
* HTML — это язык разметки, основа для создания структуры и содержимого веб-страниц.
* Он состоит из тегов, которые определяют различные элементы страницы, такие как заголовки, абзацы, изображения и ссылки.
* HTML задает скелет страницы, определяя, как элементы будут размещены на странице.
* **CSS (каскадные таблицы стилей)** :
* CSS — это стильный язык, который используется для оформления веб-страниц.
* С помощью CSS можно задавать цвета, шрифты, размеры и многое другое для элементов, созданных с помощью HTML.
* Он позволяет разработчикам делать страницы более красочными и привлекательными для пользователей.
* **JavaScript** :
* JavaScript — это язык программирования, который используется для добавления интерактивности и динамического поведения на веб-страницах.
* С его помощью можно создавать функции, которые реагируют на действия пользователей, а также выполняют операции над данными на клиентской стороне (в браузере).
* JavaScript также позволяет получать данные с сервера без перезагрузки страницы (Ajax).

1. **Основы HTML: структура документа, теги, атрибуты.**

HTML (язык гипертекстовой разметки) — это язык разметки, созданный для веб-страниц. Вот основы HTML:

* Структура документа: HTML-документ обычно начинается с тега **<html>**, который определяет начало и конец документа. Он содержит два основных раздела:
* **<head>**: Этот раздел содержит метаданные, такие как заголовок документа, ссылки на внешние ресурсы (стили, скрипты) и другие настройки.
* **<body>**: Здесь размещаются веб-сайты контента, такие как текст, изображения и другие элементы.
* Теги:
* **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>**, **<h6>**: Заголовки разного уровня.
* **<p>**: Параграф.
* **<a>**: Гиперссылка.
* **<img>**: Изображение.
* **<ul>**, **<ol>**, **<li>**: Списки.
* **<table>**, **<tr>**, **<td>**: Таблицы.
* **<div>**: Блочный элемент для группировки настроек.
* **<span>**: Инлайн-элемент для стилизации текста.
* Атрибуты:
* **src**: Атрибут **<img>**для указания источника изображения.
* **href**: Атрибут **<a>**для определения URL-адреса ссылки.
* **alt**: Атрибут **<img>**для текстового описания.
* **class**и **id**: атрибуты для стилизации и идентификации элементов с помощью CSS и JavaScript.

1. **Работа с текстом, изображениями и ссылками в HTML.**

* **Текст** : Для отображения текста используется тег **<p>**для абзацев или **<h1>**, **<h2>**, **<h3>**и так далее для заголовков разного уровня. Пример:
* HTMLСкопировать код
* <p>Это обычный текст.</p>  
  <h1>Заголовок первого уровня</h1>
* **Изображения** : Для вставки изображений используется тег **<img>**. Вы должны указать атрибут **src**, который содержит путь к изображению. Пример:
* HTMLСкопировать код
* <img src="путь\_к\_изображению.jpg" alt="Описание изображения">
* **Ссылки** : Для создания гиперссылок используется тег **<a>**. Атрибут **href**определяет URL, по которому будет переход по клику. Пример:
* HTMLСкопировать код
* <a href="https://www.example.com">Это ссылка на example.com</a>

Вы также можете структурировать вашу страницу с помощью различных контейнеров, таких как и **<div>**, **<span>**а также форматировать текст с помощью тегов и других. HTML предоставляет множество инструментов для создания и стилизации контента на веб-страницах.**<strong><em>**

**2) Глубокое погружение в HTML**

**4)** **Таблицы и формы в HTML.**

HTML (язык гипертекстовой разметки) — это стандартный язык разметки для создания веб-страниц. Таблицы и формы - это два важных элемента HTML, которые позволяют создавать интерактивные и организованные веб-страницы. Давайте более подробно рассмотрим каждое из них:

**Таблицы в HTML:**

Таблицы в формате HTML для организации данных в виде строк и столбцов. Они основаны на следующих основных элементах:

* **<table>**: Этот элемент определяет начало таблицы.
* **<tr>**: Элемент строки (строка таблицы). Он определяет каждый символ в таблице.
* **<td>**: Элемент ячейки (данные таблицы). Он определяет каждую ячейку данных в строке.
* **<th>**: Элемент заголовка (заголовок таблицы). Он определяет заголовки столбцов или строки таблицы.

Пример таблицы:

HTMLСкопировать код

**<table border="1">**  
 **<tr>**  
 **<th>Заголовок 1</th>**  
 **<th>Заголовок 2</th>**  
 **</tr>**  
 **<tr>**  
 **<td>Ячейка 1,1</td>**  
 **<td>Ячейка 1,2</td>**  
 **</tr>**  
 **<tr>**  
 **<td>Ячейка 2,1</td>**  
 **<td>Ячейка 2,2</td>**  
 **</tr>**  
**</table>**

**Формы в HTML:**

Формы в HTML позволяют пользователям отправлять данные на сервер. Они содержат элементы ввода, такие как текстовые поля, радиокнопки, флажки, кнопки отправки сообщений и другие. Основные элементы формы включают в себя:

* **<form>**: Этот элемент определяет форму и указывает, отправьте данные после их заполнения.
* **<input>**: Элемент ввода, который позволяет пользователю вводить данные. Это может быть текстовый полем, флажком, радиокнопкой и т. д. д.
* **<textarea>**: Элемент для ввода многострочного текста.
* **<select>**: Элемент для создания выпадающего списка.
* **<button>**: Элемент для создания кнопок формы или других действий.

Пример формы:

HTMLСкопировать код

<form action="/submit" method="post">  
 <label for="username">Имя пользователя:</label>  
 <input type="text" id="username" name="username" required>  
  
 <label for="email">Email:</label>  
 <input type="email" id="email" name="email" required>  
  
 <input type="submit" value="Отправить">  
</form>

Эти базовые формы таблиц позволяют разработчикам создавать структурированные и интерактивные веб-сайты, которые могут собирать и обрабатывать данные от пользователей.

**2) урок**

**1) Проверка форм и обработка ввода пользователя.**

Проверка форм и обработка ввода пользователя в HTML — важная часть создания интерактивных веб-страниц. Вот основные шаги и инструменты для этого процесса:

* **HTML-формы** : приступим к созданию HTML-форм, используя теги **<form>**, **<input>**, **<textarea>**, и другие элементы. Например:

HTML

<form id="myForm">use  
 <label for="username">Имя пользователя:</label>  
 <input type="text" id="username" name="username" required>  
 <input type="submit" value="Отправить">  
</form>

* **Проверка на стороне клиента** : используйте атрибуты HTML5, такие как **required**, **pattern**, и **minlength**для определения правил проверки на стороне клиента. Например:

HTML

<input type="email" id="email" name="email" required>  
<input type="password" id="password" name="password" minlength="6">

* **JavaScript** : Для более быстрой проверки и обработки форм используйте JavaScript. Напишите функции, которые будут сохраняться при отправке формы. Например:

javascript

document.getElementById("myForm").addEventListener("submit", function(event) {  
 const username = document.getElementById("username").value;  
 if (username.length < 3) {  
 alert("Имя пользователя должно содержать минимум 3 символа.");  
 event.preventDefault();  
 }  
});

* **Обработка на расстоянии** : всегда предполагайте, что данные могут быть подделаны на стороне клиента, поэтому важно также проводить проверку на стороне сервера при получении данных от клиента.
* **Отправка данных** : используйте методы HTTP, такие как POST или GET, для отправки данных на сервер. Например:

HTMLС

<form method="post" action="обработчик.php">  
 <!-- поля формы -->  
</form>

* **Обратная связь** : включите механизмы обратной связи для пользователя, чтобы он знал, что форма отправлена успешно или есть ошибка в введенных данных.
* **Библиотеки и фреймворки** : если вы работаете с более сложными формами, рассмотрите использование библиотек и платформеров, таких как jQuery, React или Angular, для упрощения валидации и обработки ввода.

Обработка ввода пользователя и проверка форм — это важная часть разработки веб-приложений, которая помогает обеспечить безопасность и удобство пользователей.

**3) Урок**

**2) Семантическая разметка: верхний, нижний колонтитул, навигация и другие.**

Семантическая разметка HTML Позволяет структурировать веб-страницу, указывая смысл каждого элемента. Вот несколько основных элементов семантической разметки:

* **<header>**: Этот элемент обычно используется для отображения верхней части страницы, заголовка, логотипа, навигации и других важных элементов.
* **<footer>**: **<footer>**служители для размещения нижней части страницы, часто создают контактную информацию, копирайт и ссылки в социальных сетях.
* **<nav>**: **<nav>**используется для обозначения блока с навигацией, например меню сайта. Это помогает браузерам и системе поиска понимать, что это именно навигация.
* **<article>**: Этот элемент используется для выделения независимых статей, новостей, постов блога и другого самостоятельного контента.
* **<section>**: **<section>**Позволяет группировать контент по тематическим разделам. каждая секция может иметь свой собственный заголовок.
* **<aside>**: Этот элемент предназначен для выделения вспомогательных сведений, которые могут быть связаны с основным содержанием страницы, но не являются обязательными для его понимания.
* **<main>**: **<main>**означает общее содержание страницы. Обычно на странице должен быть только один элемент **<main>**.

Используя эти семантические элементы, вы можете создавать более понятные и доступные веб-страницы, улучшать индексацию вашего контента с помощью функции поиска и помогать пользователям с ограниченными возможностями лучше понимать структуру вашего сайта.

**3) Основы CSS**

**3)** **Введение в CSS, селекторы и стили.**

Введение в CSS (каскадные таблицы стилей) представляет собой всю часть веб-разработки, которая позволяет управлять стилями и внешним видом веб-страницы. Вот краткое описание основных концепций CSS:

* **Селекторы (Selectors)** : Селекторы — это шаблоны, которые определяют элементы на веб-странице, к которым применяются стили. Самый простой селектор, например - это имя тега, **p**для абзацев. Также существуют классовые селекторы ( **.класс**), идентификаторы ( **#идентификатор**), селекторы атрибутов ( **[атрибут="значение"]**) и множество других.
* **Свойства (Properties)** : Свойства определяют, какие стили будут применяться к выбранным элементам. Например, **color**укажите цвет текста, а **font-size**- размер шрифта.
* **Значения (значения)** : Значения характеризуют определенные стили. Например, **color**значение может быть **red**, **#FF0000**, или **rgb(255, 0, 0)**.
* **Объединение селекторов (Объединение селекторов)** : Вы можете комбинировать селекторы, чтобы применять стили к различным элементам одновременно. Например, **p, h1**применит стили к абзацам и заголовкам первого уровня.
* **Каскадность (каскадирование)** : если несколько правил CSS применяются к одному элементу, они могут анализировать. В этом случае CSS использует каскадность системы, чтобы определить, какие стили следует применить. Это можно контролировать с помощью специфичности селекторов и порядка следования правил.
* **Внешние и внутренние стили (Внешние и внутренние стили)** : Стили могут быть выбраны непосредственно в HTML с использованием тега **<style>**, внутри тегов HTML с использованием атрибута **style** или во внешних CSS-файлах, которые подключаются к HTML-документу.
* **Комментарии (Комментарии)** : Вы можете добавлять комментарии в CSS с помощью **/\* ваш комментарий \*/**.

Это основы CSS, которые помогут вам начать работу со стилями в веб-разработке. CSS предоставляет бесконечные возможности для оформления и манипулирования внешним видом ваших веб-страниц.

1. **Оформление текста, цвета и фоны.**

Оформление текста, цвета и фонов веб-страницы можно легко настроить с помощью CSS (каскадных таблиц стилей). Вот основные свойства CSS для этой цели:

* **Оформление текста** :
* **color**: Задает цвет текста. Можно использовать названия цветов (например, «красный» или «синий») или коды цветов (#FF0000 для красного).
* **font-family**: Определяет шрифт текста.
* **font-size**: Устанавливает размер шрифта.
* **font-weight**: Определяет толщину шрифта (например, «жирный» для жирного шрифта).

Пример:

p {  
 color: blue;  
 font-family: Arial, sans-serif;  
 font-size: 16px;  
 font-weight: bold;  
}

* **Фоны** :
* **background-color**: Устанавливает цвет элемента фонаря.
* **background-image**: Установить изображение в качестве фона.
* **background-repeat**: Определяет, как будет повторяться фон (например, «повторить» для повторения по горизонтали и вертикали).
* **background-size**: Определяет размер фонового изображения.

Пример:

body {  
 background-color: #f0f0f0;  
 background-image: url('background.jpg');  
 background-repeat: no-repeat;  
 background-size: cover;  
}

Это лишь небольшая часть возможностей CSS для оформления текста, цветов и фонов на веб-страницах. CSS также предоставляет множество других свойств и способов настройки стилей, чтобы сделать вашу страницу более креативной и уникальной.

**Позиционирование и макетирование элементов с помощью CSS.**

Позиционирование и макетирование элементов с помощью CSS — это фундаментальные аспекты веб-разработки, позволяющие размещать и стилизовать элементы на веб-странице. Давайте подробно рассмотрим каждый из этих аспектов.

* **Позиционирование элементов с помощью CSS:**
* **Статическое позиционирование:** Это значение по умолчанию. Элементы размещаются в потоке документа и друг напротив друга сверху вниз.
* **Относительное позиционирование:** элементы также отображаются в потоке документа, но вы можете использовать свойства **top**, **bottom**, **left**и **right**, чтобы сместить их к нормальному состоянию.
* **Абсолютное позиционирование:** элементы передаются из потока документа и располагаются относительно ближайшего предка с заданным позиционированием (обычно **relative**или **absolute**). Это позволяет точно контролировать свое местоположение.
* **Фиксированное позиционирование:** элементы остаются на одном и том же месте экрана, даже при прокрутке страницы.
* **Макетирование элементов с помощью CSS:**
* **Блочная модель:** Элементы могут быть блочными или стрингами. Блочные элементы занимают все доступные фрагменты и начинают новый символ, в то время как строчные элементы занимают только то место, которое им необходимо, и продолжают друг друга за другим в тексте.
* **Flexbox:** это подвесной инструмент для создания гибких макетов. Он позволяет легко управлять пространством между элементами в контейнере и изменять их размеры.
* **Сетка:** CSS Grid Layout позволяет создавать двумерные макеты, разбивая страницы на строки и столбцы, что представляет собой сложную сложную сетку макетов.
* **Позиционирование:** уже упомянутое абсолютное и относительное позиционирование позволяет точно сохранять положение элементов.

Представьте, что у вас есть веб-страница, и вы хотите связать логотип, меню навигации и текстовое содержание. Вы можете использовать вышеупомянутое оборудование для создания желаемого макета и цитировать элементы на странице. Например, вы можете использовать Flexbox для спортивных элементов в тексте, а затем абсолютно точное позиционирование для точного размещения логотипа в верхней части страницы. Это позволяет создать красивый и удобный пользовательский интерфейс для вашего веб-сайта.