Приднестровский государственный университет им. Т.Г. Шевченко
Физико-технический институт

ОСНОВЫ ВЕБ-РАЗРАБОТКИ С ПРИМЕНЕНИЕМ ИИ-АССИСТЕНТОВ

Лабораторный практикум

Разработал:
ст. преподаватель
кафедры ИТ
Бричаг Д.В.

г. Тирасполь
2025 г.

Лабораторная работа №2

Семантическая вёрстка, Flex/Grid, адаптивность с медиазапросами.

Цель работы:

- Освоить семантические HTML5-теги для структурирования страницы.
- Научиться работать с flexbox и grid-сетками.
- Познакомиться с принципами адаптивного дизайна (responsive design).
- Научиться использовать медиазапросы для адаптивности.
- Закрепить практику работы с ИИ при генерации и модификации кода.

Теоретическая справка

CSS (Cascading Style Sheets) — язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам. Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL. Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на него никакого влияния.

Объявление стиля состоит из двух частей: селектора и объявления. Объявление состоит из двух частей: имя свойства (например, color) и значение свойства (grey). Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (код в фигурных скобках) перечисляются форматирующие команды — свойства и их значения.

Основные концепции:

1. Каскадность

   Стиль определяется сразу из нескольких источников (браузер, разработчик, внешние и внутренние таблицы стилей).

   Применяется тот стиль, который **побеждает в каскаде**:

   - на основе источника (inline > внешние стили > стили браузера),
   - веса (specificity),
   - порядка следования (последний объявленный стиль при равных условиях).

2. Наследование

   Некоторые свойства (например, color, font-size, line-height) наследуются от родителя, другие — нет (например, margin, padding, border).

3. Блочная модель (Box Model)

   Каждый элемент — прямоугольный блок:

   - content (содержимое),
   - padding (отступ внутри блока),
   - border (рамка),
   - margin (отступ снаружи блока).

4. Единицы измерения

   - Абсолютные: px, cm, mm.
   - Относительные: %, em, rem, vh, vw.
   - em зависит от размера шрифта родителя, rem — от корня (html).

Специфичность (вес селекторов)

Это правило, определяющее *какой стиль важнее*, если одно свойство задано в нескольких местах.

Вес выражается как 4-значное число (A, B, C, D):

- A — встроенные стили (inline) style="..."

- B — количество ID-селекторов
- C — количество классов, атрибутов и псевдоклассов
- D — количество тегов и псевдоэлементов

Примеры**:**

- #header → 0100 (ID = сильный)
- .menu .item:hover → 0022 (2 класса + 1 псевдокласс)
- div p → 0002 (2 тега)
- style="color:red;" → 1000 (inline всегда почти выше всего)
- * (универсальный селектор) → 0000 (минимум)

Правила:

1. Сравниваются веса, выигрывает больший.

2. Если веса равны — побеждает тот стиль, который объявлен позже в коде.

3. !important перебивает обычные правила, но проигрывает более специфичному !important.

Приоритетность

1. **!important** (самый сильный, но использовать лучше редко).

2. Inline-стили (style="...").

3. Стили из внешних/внутренних таблиц (сравниваются по весу селектора).

4. Наследование.

5. Стиль по умолчанию браузера.

Хорошие практики

- Минимизировать использование !important.
- Не злоупотреблять ID-селекторами (лучше классы → гибче).
- Стараться держать специфичность **низкой**, чтобы стили легко переопределялись.
- Использовать rem для масштабируемости и адаптивности.

- Всегда проверять результат в каскаде через **DevTools (F12)**.

Семантические теги HTML5

- <header> — шапка страницы
- <nav> — навигация
- <main> — основное содержимое
- <section> — раздел контента
- <article> — статья или запись блога
- <aside> — боковая колонка
- <footer> — подвал

Семантика помогает поисковым системам и скринридерам понимать структуру документа.

Flexbox

Flexbox (Flexible Box Layout) — это модель раскладки в CSS, которая облегчает выравнивание и распределение элементов в одном измерении — по строке или по колонке.

- Родителю задаём display: flex;.
- Главная ось (row или column) и поперечная ось.
- Управление распределением:
  - justify-content — выравнивание по главной оси,
  - align-items — выравнивание по поперечной оси,
  - flex-wrap — перенос строк.
- Для элементов:
  - flex-grow, flex-shrink, flex-basis — управляют размером и "растяжением".

Flexbox отлично подходит для линейных компоновок (меню, карточки, кнопки), а Grid — для построения полноценной сетки.

Пример размещения элементов по горизонтали:

```
.container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

CSS Grid

Grid — это система раскладки, которая позволяет строить сетку из строк и столбцов и удобно размещать в ней элементы. Используется для адаптивных макетов, сложных сеток и упрощения верстки, где раньше приходилось использовать float или flexbox-хаки.

- Родителю задаём display: grid;
- Определяем колонки и строки;
- Дочерним элементам можно указывать, сколько колонок/строк они занимают;

Пример сетки 3x2:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr; /* три равные колонки */
  grid-template-rows: auto auto;     /* две строки */
  gap: 10px;                         /* отступы между ячейками */
}
```

Медиазапросы (адаптивность)

Медиа-запросы (media queries) — это механизм CSS для применения разных стилей в зависимости от характеристик устройства или окна браузера (ширины, высоты, ориентации экрана, типа носителя и т. д.).

Базовый синтаксис медиа-запроса выглядит следующим образом:

```
@media (условие) {
  /* стили */
}
```

Пример изменения стиля для экранов меньше 768px:

```
@media (max-width: 768px) {
  .container {
    grid-template-columns: 1fr;
  }
}
```

Пример изменения стиля для экранов больше 769px:

```
@media (min-width: 769px) {
  .container {
    grid-template-columns: 2fr;
  }
}
```

Практическая часть

Аналогично предыдущей работе необходимо создать папку проекта и файлы. Для реализации текущей лабораторной работы будет достаточно простой структуры, состоящей из папки lab2 и файлов index.html и style.css, а также файла README.md

Использование ИИ для генерации шаблона

Примеры промптов:

```
Сделай HTML-страницу блога с семантическими тегами: header, nav, main, aside, footer.
В main должно быть 2 статьи. Используй CSS Grid для сетки. Стили в отдельном файле.
```

```
Сгенерируй макет сайта с шапкой, навигацией, контентной областью и подвалом.
Сделай расположение элементов через Flexbox. Добавь медиазапросы, чтобы на мобильных всё отображалось в одну колонку.
```

```
Напиши HTML+CSS для страницы с левой колонкой (меню), центральным контентом и правой колонкой (реклама).
Сделай макет адаптивным: на мобильных пусть колонки складываются в один столбец.
```

```
Сделай HTML-разметку новостной ленты: блок новостей в grid-сетке 3 колонки.
Добавь адаптивность: при ширине экрана меньше 768px сделать 1 колонку.
```

```
Создай простую адаптивную страницу с header, секцией контента и footer.
Стили реализуй через flexbox и медиазапросы.
```

Редактирование вручную

- Замените текст статей/контента на свой.
- Подстройте цвета и шрифты.
- Проверьте работу сетки при изменении ширины окна браузера.

Проверка адаптивности

- Откройте сайт в браузере.
- Сожмите окно до ширины смартфона (~400px).
- Проверьте, что элементы перестраиваются корректно.

Работа с git и GitHub

1. Инициализируйте репозиторий:

```
git init
git add .
git commit -m "Lab2: initial commit"
```

2. Создайте новый репозиторий на GitHub.
3. Свяжите локальный проект с GitHub:

```
git remote add origin https://github.com/<ваш_логин>/<имя_репозитория>.git
git branch -M main
git push -u origin main
```

Отчёт (в README.md)

В README.md добавить:

- Заголовок: *Лабораторная работа №2*
- Скриншоты: версия для компьютера и мобильной ширины.
- Ответы на вопросы:

- Какие семантические теги вы использовали?
- Где использовали flexbox, а где grid?
- Какой медиазапрос добавили и что он меняет?
- Где помог ИИ, а что пришлось дорабатывать вручную?

Результаты работы

В итоге у вас должно быть:

1. Семантическая страница с grid/flex-сеткой.
2. Реализованная адаптивность через медиазапросы.
3. Репозиторий на GitHub с проектом.
4. README.md с отчётом и скриншотами.

Критерии оценки

- Использованы семантические теги (2 балла).
- Макет собран на flex/grid (2 балла).
- Реализована адаптивность через медиазапросы (2 балла).
- Код выложен на GitHub (2 балла).
- README.md с отчётом и скриншотами (2 балла).