Лекция: CSS Flexbox: Современная верстка и выравнивание

МДК 09.01. Проектирование и разработка веб-приложений

Цель: Сформировать у студентов понимание модуля Flexbox Layout, его свойств и областей применения для создания гибких и адаптивных макетов веб-страниц.

Задачи:

- Изучить базовые концепции и терминологию Flexbox.
- Освоить свойства контейнера (flex-container) для управления направлением, переносом и выравниванием элементов.
- Освоить свойства элементов (flex-items) для управления их порядком, гибкостью и выравниванием.
- Научиться применять Flexbox для решения типовых задач верстки.

План лекции:

1. **Введение в Flexbox.**
    - Проблемы традиционной верстки (float, inline-block).
    - Что такое Flexbox? Основные преимущества.
    - Ключевые термины: Flex Container, Flex Items, Main Axis, Cross Axis.
2. **Свойства Flex-контейнера.**
    - `display: flex | inline-flex`
    - `flex-direction` (направление главной оси).
    - `flex-wrap` (перенос элементов).
    - `justify-content` (выравнивание по главной оси).
    - `align-items` (выравнивание по поперечной оси для одной строки).
    - `align-content` (выравнивание по поперечной оси для многострочного контейнера).
    - `gap` (расстояние между элементами).
3. **Свойства Flex-элементов.**
    - `order` (порядок отображения).
    - `flex-grow` (коэффициент расширения).
    - `flex-shrink` (коэффициент сжатия).
    - `flex-basis` (базовый размер).
    - `flex` (сокращенная запись: `grow`, `shrink`, `basis`).
    - `align-self` (индивидуальное выравнивание элемента).
4. **Практические примеры и паттерны верстки.**
    - Центрирование элемента по горизонтали и вертикали.

    - Создание "липкого" футера (footer).
    - Верстка карточек товара или новостей.
    - Создание адаптивной навигационной панели.

5. **Резюме. Когда и зачем использовать Flexbox?**

Рассмотрение пунктов плана

1. Введение в Flexbox

Проблемы традиционной верстки: До появления Flexbox верстальщики использовали свойства `float` и `display: inline-block`. Это приводило к ряду сложностей:

- Сложное вертикальное выравнивание.
- Необходимость использования "костылей" для очистки обтекания (clearfix).
- Хрупкость макета, особенно в адаптивном дизайне.
- Невозможность простого перераспределения свободного пространства между элементами.

Что такое Flexbox? Flexbox (Flexible Box Layout) — это CSS-модуль, предназначенный для эффективного расположения, выравнивания и распределения пространства между элементами в контейнере, **даже когда их размер неизвестен или динамичен**.

Основные преимущества:

- Простое и предсказуемое выравнивание по осям.
- Возможность легко менять порядок элементов.
- Элементы могут "гибко" растягиваться и сжиматься, заполняя доступное пространство.
- Упрощение создания адаптивных макетов.

Ключевые термины:

- **Flex Container (флекс-контейнер)** — элемент, у которого свойство `display` установлено как `flex` или `inline-flex`. Он является родителем для flex-элементов.
- **Flex Items (флекс-элементы)** — прямые потомки флекс-контейнера.
- **Main Axis (главная ось)** — основная ось, вдоль которой располагаются флекс-элементы. Ее направление задается свойством `flex-direction` (по умолчанию — горизонтально, слева направо).
- **Cross Axis (поперечная ось)** — ось, перпендикулярная главной.

> **Визуализация:** Представьте себе коробку (контейнер) с карандашами (элементами). Вы можете решить, класть карандаши вдоль коробки (главная ось — горизонтальная) или поперек (главная ось — вертикальная). Выравнивать их можно как по длине коробки (главная ось), так и по высоте (поперечная ось).

---

2. Свойства Flex-контейнера

Все эти свойства применяются к родительскому элементу (контейнеру).

- `display: flex | inline-flex`

- `flex`: контейнер становится блочным элементом.
- `inline-flex`: контейнер становится строчно-блочным элементом.
- **Пример:**

```
.container {
  display: flex; /* Теперь все дочерние элементы стали flex-items */
  background-color: #f0f0f0;
}
```

- `flex-direction` — определяет направление главной оси.
  - `row` (по умолчанию): слева направо.
  - `row-reverse`: справа налево.
  - `column`: сверху вниз.
  - `column-reverse`: снизу вверх.
  - **Пример:**

```
.container {
  display: flex;
  flex-direction: column; /* Элементы выстроятся в колонку */
}
```

- `flex-wrap` — определяет, могут ли элементы переноситься на новую строку.
  - `nowrap` (по умолчанию): все элементы помещаются в одну строку.
  - `wrap`: элементы переносятся на следующую строку при нехватке места.
  - `wrap-reverse`: элементы переносятся на предыдущую строку.
  - **Пример:**

```
.container {
  display: flex;
  flex-wrap: wrap; /* Элементы будут красиво переноситься на узких
экранах */
}
```

- `justify-content` — выравнивает элементы по **главной оси**.
  - `flex-start` (по умолчанию): к началу оси.
  - `flex-end`: к концу оси.
  - `center`: по центру оси.
  - `space-between`: первый элемент в начале, последний в конце, остальное пространство распределено между ними.
  - `space-around`: пространство распределяется вокруг каждого элемента.
  - `space-evenly`: пространство распределяется равномерно между всеми элементами.
  - **Пример (центрирование по горизонтали):**

```
.container {
  display: flex;
  justify-content: center; /* Элементы встанут по центру горизонтали */
}
```

- `align-items` — выравнивает элементы по **поперечной оси** для **одной строки**.
  - `stretch` (по умолчанию): элементы растягиваются на всю высоту контейнера.
  - `flex-start`: к началу поперечной оси.
  - `flex-end`: к концу поперечной оси.
  - `center`: по центру поперечной оси.
  - `baseline`: выравнивание по базовой линии текста.
  - **Пример (центрирование по вертикали):**

```
.container {
  display: flex;
  align-items: center; /* Элементы встанут по центру вертикали */
  height: 300px; /* Важно задать высоту контейнера! */
}
```

- `align-content` — выравнивает **строки** элементов по поперечной оси в **многострочном** контейнере (работает только если `flex-wrap: wrap`).
  - Принимает те же значения, что и `justify-content`.
  - **Пример:**

```
.container {
  display: flex;
  flex-wrap: wrap;
  align-content: space-between; /* Строки распределятся по высоте
контейнера */
}
```

- `gap` — задает отступы между элементами.
  - `gap: 10px;` — отступ 10px по обеим осям.
  - `gap: 10px 20px;` — рядный отступ 10px, колоночный 20px.
  - **Пример:**

```
.container {
  display: flex;
  gap: 15px; /* Между всеми элементами появится отступ в 15px */
}
```

3. Свойства Flex-элементов

Эти свойства применяются к дочерним элементам (flex-items).

- `order` — определяет порядок отображения элемента. По умолчанию 0.
  - Элементы располагаются в порядке возрастания `order`.
  - **Пример:** Сделаем так, чтобы второй элемент отображался первым.

```
.item:nth-child(2) {
  order: -1; /* Меньше, чем у остальных (0), поэтому будет первым */
}
```

- `flex-grow` — определяет **коэффициент расширения** элемента. По умолчанию 0.
  - Показывает, какую долю свободного пространства в контейнере может занять элемент.
  - **Пример:** Два элемента. Первый должен занимать оставшееся пространство.

```
.item:first-child {
  flex-grow: 1; /* Займет всю доступную ширину */
}
.item:last-child {
  /* flex-grow: 0; по умолчанию, не будет расширяться */
}
```

- `flex-shrink` — определяет **коэффициент сжатия** элемента. По умолчанию 1.
  - Показывает, насколько элемент будет сжиматься относительно других, если не хватает места.
  - **Пример:** Запретим первому элементу сжиматься.

```
.item:first-child {
  flex-shrink: 0; /* Не будет сжиматься, сохранит свой базовый размер
*/
}
```

- `flex-basis` — определяет **базовый размер** элемента до распределения свободного пространства.
  - Может быть в px, %, em, auto. По умолчанию `auto`.
  - Аналогичен `width`/`height` (в зависимости от `flex-direction`).
- `flex` — сокращенное свойство: `flex-grow flex-shrink flex-basis`.
  - **Рекомендуется использовать именно его.**
  - **Часто используемые значения:**

    - `flex: 1` -> `1 1 0` (элемент гибкий, растет и сжимается).
    - `flex: 0 0 200px` -> элемент негибкий, фиксированной ширины 200px.
    - `flex: auto` -> `1 1 auto`.

- `align-self` — позволяет переопределить выравнивание по поперечной оси для отдельного элемента.
    - Принимает те же значения, что и `align-items`.
    - **Пример:** Один элемент в конце, а остальные по центру.

```
.container {
  display: flex;
  align-items: center;
}
.item:last-child {
  align-self: flex-end; /* Этот элемент "отобьется" от общей группы */
}
```

---

4. Практические примеры и паттерны верстки

Пример 1: Идеальное центрирование по горизонтали и вертикали

```
.centered-container {
  display: flex;
  justify-content: center; /* Центр по горизонтали */
  align-items: center;     /* Центр по вертикали */
  height: 100vh;           /* На всю высоту окна просмотра */
}
```

Пример 2: "Липкий" футер (всегда прижат к низу, даже если контента мало)

```
<body style="display: flex; flex-direction: column; min-height: 100vh;">
  <header>...</header>
  <main style="flex: 1;"> <!-- Основной контент растянется, вытолкнет футер -->
    ...
  </main>
  <footer>...</footer>
</body>
```

Пример 3: Верстка карточек товара

```
.products-grid {
  display: flex;
  flex-wrap: wrap;
```

```
  gap: 20px;
  justify-content: space-around; /* Карточки равномерно распределятся */
}

.product-card {
  flex: 0 1 300px; /* Карточка не растет, может сжиматься, базовая ширина 300px */
  /* Благодаря flex-wrap и flex-basis, карточки будут красиво переноситься */
}
```

Пример 4: Адаптивная навигационная панель

```
.navbar {
  display: flex;
  justify-content: space-between; /* Лого слева, меню справа */
  align-items: center;
  padding: 1rem;
}

.nav-links {
  display: flex;
  gap: 1.5rem;
  list-style: none;
}

/* На мобильных устройствах превращаем меню в колонку */
@media (max-width: 768px) {
  .navbar {
    flex-direction: column;
  }
}
```

---

5. Резюме. Когда и зачем использовать Flexbox?

Итоги:

- Flexbox — это мощный одномерный инструмент компоновки. Он управляет расположением либо по **горизонтали**, либо по **вертикали** в один момент времени.
- Он решает проблемы выравнивания, распределения пространства и порядка элементов, которые были крайне сложны с помощью старых методов.
- Ключ к пониманию — мысленное представление **главной** и **поперечной** осей.
- Свойства контейнера (`justify-content`, `align-items`) управляют всеми элементами сразу.
- Свойства элементов (`order`, `flex`, `align-self`) позволяют точечно управлять отдельными элементами.

Когда использовать Flexbox?

- Для компонентов интерфейса: навигационные панели, карточки, кнопки, формы.
- Для выравнивания содержимого внутри блока (например, центрирование иконки и текста).

- Для создания простых одномерных макетов (например, ряд секций или список с иконками).
- Когда вам нужно динамически распределять доступное пространство между элементами.

Ограничения:

- Flexbox — **одномерный**. Для сложных двумерных макетов (сеток одновременно по строкам и столбцам) лучше использовать **CSS Grid Layout**.

Заключение: Flexbox — это фундаментальный инструмент в арсенале современного фронтенд-разработчика. Его освоение значительно ускоряет процесс верстки и делает код более чистым и поддерживаемым. Используйте его для компоновки небольших компонентов и одномерных макетов, и комбинируйте с CSS Grid для построения общих структур страниц.