**Flexbox — гибкая компоновка элементов в CSS**

**7.1. Что такое Flexbox и зачем он нужен?**

Flexbox (Flexible Box Layout) — это современный CSS-инструмент для управления расположением элементов внутри контейнера. Позволяет создавать сложные адаптивные и отзывчивые (responsive) макеты без костылей.

**Зачем нужен Flexbox:**

* Удобно выравнивать элементы по горизонтали и вертикали.
* Легко изменять порядок и направление элементов.
* Позволяет просто создавать адаптивные дизайны.

**7.2. Основы использования Flexbox**

**Шаг 1. Создание flex-контейнера**

css

КопироватьРедактировать

.container {

display: flex;

}

**Шаг 2. Настройка направления (flex-direction)**

Определяет, как расположены элементы (по горизонтали или вертикали):

css

КопироватьРедактировать

.container {

display: flex;

flex-direction: row; /\* горизонтально, по умолчанию \*/

/\* flex-direction: column; вертикально \*/

}

**Шаг 3. Выравнивание элементов (justify-content и align-items)**

* justify-content выравнивает элементы по основной оси.
* align-items выравнивает элементы по перпендикулярной оси.

css

КопироватьРедактировать

.container {

justify-content: center; /\* по центру по горизонтали \*/

align-items: center; /\* по центру по вертикали \*/

}

**7.3. Гибкость элементов — свойства flex-grow, flex-shrink, flex-basis**

**flex-grow (растягивание элемента)**

css

КопироватьРедактировать

.item {

flex-grow: 1; /\* занимает всё доступное пространство \*/

}

**flex-shrink (сжатие элемента)**

css

КопироватьРедактировать

.item {

flex-shrink: 0; /\* не сжимается при уменьшении размера контейнера \*/

}

**flex-basis (базовый размер элемента)**

css

КопироватьРедактировать

.item {

flex-basis: 200px; /\* начальный размер элемента \*/

}

**Сокращённая запись**:

css

КопироватьРедактировать

.item {

flex: 1 0 200px; /\* grow shrink basis \*/

}

**7.4. Порядок и перенос элементов (order и flex-wrap)**

**order (порядок элементов)**

css

КопироватьРедактировать

.item-1 { order: 2; }

.item-2 { order: 1; }

Элементы могут отображаться не в порядке HTML-разметки, а в любом заданном.

**flex-wrap (перенос элементов)**

css

КопироватьРедактировать

.container {

flex-wrap: wrap; /\* элементы переносятся на новую строку \*/

}

**7.5. Расстояния между элементами (gap)**

gap позволяет легко задавать одинаковые отступы между элементами:

css

КопироватьРедактировать

.container {

gap: 20px;

}

**7.6. Примеры и реальные кейсы применения Flexbox**

**Горизонтальное меню**

css

КопироватьРедактировать

.menu {

display: flex;

justify-content: space-between;

}

**Карточки товаров**

css

КопироватьРедактировать

.cards {

display: flex;

flex-wrap: wrap;

gap: 15px;

}

.card {

flex: 1 1 300px; /\* растягивается, но не меньше 300px \*/

}

**Выравнивание текста и изображения по центру блока**

css

КопироватьРедактировать

.container {

display: flex;

justify-content: center;

align-items: center;

}

**7.7. Современные подходы и трюки Flexbox**

* **Sticky Footer** (подвал всегда внизу страницы):

css

КопироватьРедактировать

body {

display: flex;

flex-direction: column;

min-height: 100vh;

}

footer {

margin-top: auto; /\* footer прижимается вниз \*/

}

* **Идеально центрированный блок** (по центру вертикально и горизонтально):

css

КопироватьРедактировать

.container {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

**7.8. Частые ошибки и анти-паттерны Flexbox**

* Неиспользование gap для отступов (приводит к хаосу и сложным margin).
* Злоупотребление абсолютными размерами элементов (ломает адаптив).
* Избыточное применение flex-direction: column (лучше для ряда элементов использовать row + flex-wrap).

**7.9. Best Practices работы с Flexbox**

* Используй gap вместо margin для простых отступов между элементами.
* Применяй flex-wrap для адаптивности.
* Используй относительные единицы (rem, %) для размеров элементов.
* Учитывай особенности браузеров и проверяй поддержку через CanIUse.com.

**7.10. Flexbox vs Grid — когда что использовать?**

* **Flexbox** идеален для одномерных макетов (меню, строки, колонки).
* **Grid** лучше для двумерных (сложные сетки, адаптивные галереи).

Используй Flexbox для простых выравниваний и небольших наборов элементов, Grid — для больших и сложных layout'ов.

**7.11. Глоссарий Flexbox**

| **Термин** | **Определение** |
| --- | --- |
| flex-direction | Направление элементов |
| justify-content | Выравнивание по основной оси |
| align-items | Выравнивание по перпендикулярной оси |
| flex-wrap | Перенос элементов |
| flex-grow | Растягивание элементов |
| flex-shrink | Сжатие элементов |
| flex-basis | Базовый размер элемента |
| order | Порядок элементов |
| gap | Отступы между элементами |

**7.12. Вопросы для самопроверки**

1. Какие основные преимущества использования Flexbox?
2. Чем отличается justify-content от align-items?
3. Как работает flex-wrap и для чего он нужен?
4. Почему gap лучше margin при работе с Flexbox?
5. В каких случаях Grid лучше Flexbox и наоборот?