**Вопросы 💎**

1. **Почему в большинстве ситуаций стоит использовать flexbox-позиционирование?**

Flexbox позволяет создать адаптивный семантичный современный дизайн намного проще и понятнее, чем с использованием других видов позиционирования.

Блоки становятся как бы «резиновыми»- могут сжиматься и растягиваться по необходимым правилам, занимая нужное пространство.

1. **Самостоятельно изучите способ позиционирования через display: table и ответьте на вопрос, для каких ситуаций оно лучше всего подходит?**

display: table  делает вывод группы элементов подобно таблице <table>, но с ограничением – объединения ячеек colspan и rowspan не поддерживаются.

**Этот способ хорош для семантической вёрстки и позволяет избавиться от лишних тегов (не нужно прописывать каждую строку и столбец в table).**

**Самое главное, нужно использовать таблицы только тогда, когда реально нужны таблицы на сайте, а не верстать таким образом весь сайт (устаревший вариант табличной верстки).**

1. **Какими способами можно сделать горизонтальное выравнивание по центру? Минимум 3 варианта, можно больше**
   1. P { text-align: center }
   2. Свойство помогает отцентровать элемент по вертикали и горизонтали так, чтобы центровка сохранялась при изменении размеров элемента или контейнера. Необходимо задать контейнеру раскладку флексбокса, а дочернему флекс-элементу margin: auto. – пример в вопросе 5.
   3. justify-content: center -  браузер распределяет пространство по центру flex контейнера
2. **Какие есть оси во флекс-верстке и как задается их направление?**

По умолчанию:

Главная- по горизонтали слева направо

Поперечная- по вертикали сверху вниз

Оси можно менять при помощи flex-direction (По умолчанию flex-direction: row;)

1. **Разберитесь, как работает свойство margin: auto во флекс-верстке, приведите пример использования**

Свойство помогает отцентровать элемент по вертикали и горизонтали так, чтобы центровка сохранялась при изменении размеров элемента или контейнера. Необходимо задать контейнеру раскладку флексбокса, а дочернему флекс-элементу margin: auto.

В этом случае флекс-элемент уменьшит свой размер под содержимое и отцентруется по вертикали и горизонтали.

Пример:

Html:

 <footer class="flex-container-footer">

        <section class="footer">

            <h2>Секция футера 1</h2>

        </section>

    </footer>

Css:

.flex-container-footer {

    display: flex;

}

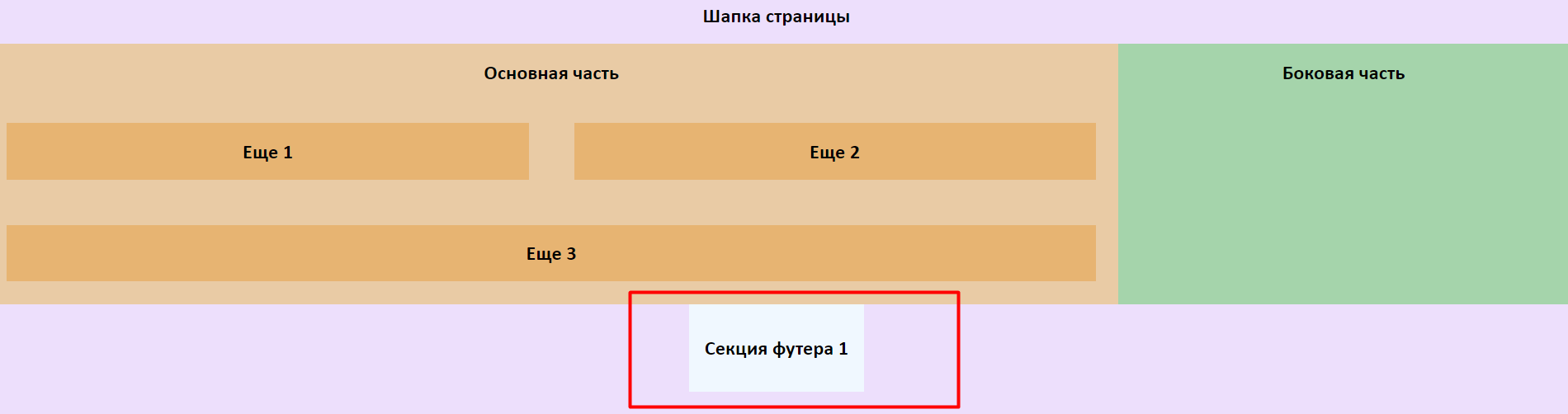
.footer {

    background-color: aliceblue;

    padding: 15px;

    margin: auto;

}



1. **В чем преимущества box-sizing?**

Используется при использовании блочной модели.

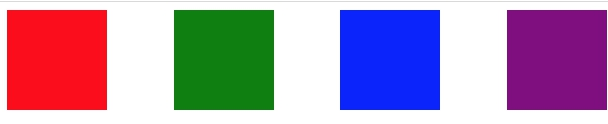
Преимущество в том, что задав элементу размер, можно быть уверенным что этот размер и будет фигурировать. box-sizing, позволяет нам точно задавать размер элемента (иначе реальный размер задается путем сложения width, padding и border)

1. **Чем отличается flex-grow от flex-shrink?**

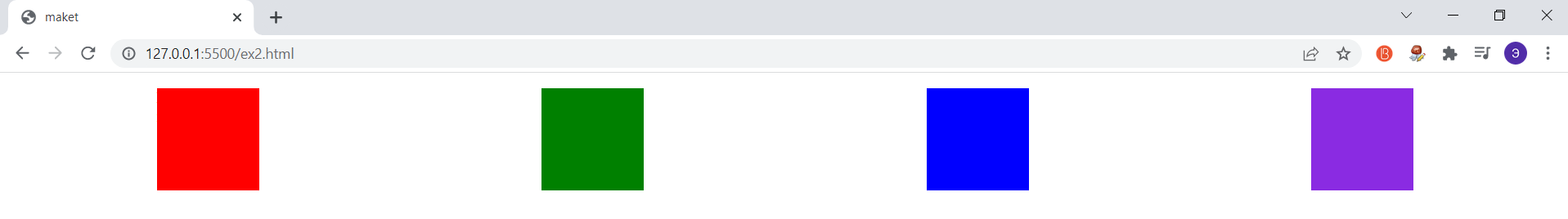
flex-grow - свойство определяет, может ли начальная ширина flex-элемента увеличиваться (расти)

flex-shrink - определяет, может ли ширина flex-элемента уменьшиться. Flex-элементы будут заполнять контейнер в зависимости от значения flex-shrink, когда стандартная ширина flex-элементов шире, чем flex-контейнер. + Подробнее в вопросе 14

1. **Как можно добиться следующего позиционирования элементов:**



При помощи flex, у меня получилось вот так:



Html:

div class="flex-container">

        <div class="red">

            Красный квадрат

        </div>

        <div class="green">

            Зеленый квадрат

        </div>

        <div class="blue">

            Синий квадрат

        </div>

        <div class="fiolet">

            Фиолетовый квадрат

        </div>

    </div>

Css:

.flex-container{

    display: flex;

    justify-content: space-around;

    padding: 15px;

}

.red{

    background-color: red;

    color: red;

    width: 100px;

    height: 100px;

}

.green{

    background-color: green;

    color: green;

    width: 100px;

    height: 100px;

}

.blue{

    background-color: blue;

    color: blue;

    width: 100px;

    height: 100px;

}

.fiolet {

    background-color:blueviolet;

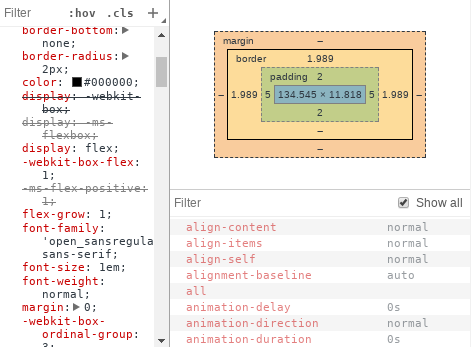
    color: blueviolet;

    width: 100px;

    height: 100px;

}

1. **Какой в итоге будет размер у элемента (можно округлить)?**



148,523 х 17,807

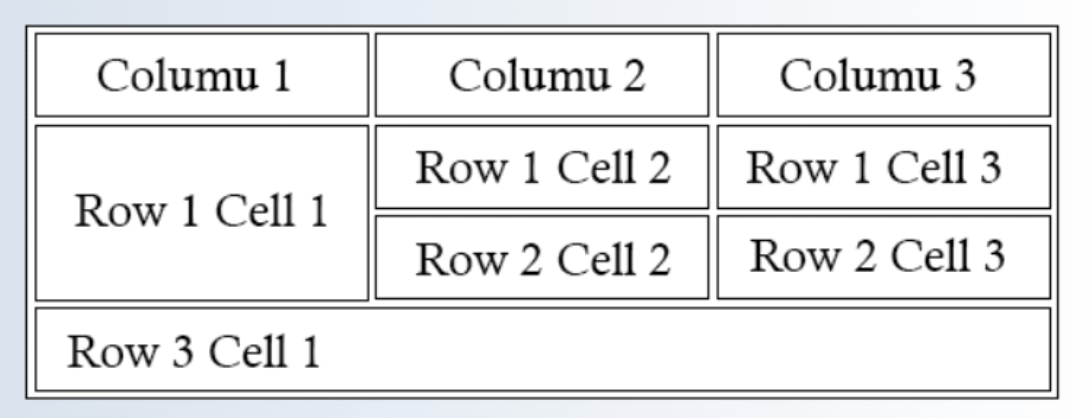
1. **Самостоятельно разберитесь, зачем нужно свойство order?**

Order отвечает за порядковый номер флекс-элемента (задаётся целым числом, положительным или отрицательным).

По умолчанию- 0. Если задать -1, то этот элемент будет первым в контейнере, потом 0,1 и т д.

Элементы с одинаковым значением order располагаются в том порядке, в каком они находятся в исходном коде.

1. **Каким кодом можно сделать такую таблицу?**



<table border="1" style="width:50%">

        <tr>

            <td>Columu 1</td>

            <td>Columu 2</td>

             <td>Columu 3</td>

        </tr>

        <tr>

            <td rowspan="2">Row 1 Cell 1</td>

            <td>Row 1 Cell 2</td>

            <td>Row 1 Cell 3</td>

        </tr>

        <tr>

            <td>Row 2 Cell 2</td>

            <td>Row 2 Cell 3</td>

        </tr>

        <tr>

            <td colspan="3">Row 3 Cell 1</td>

        </tr>

    </table>

1. **Изучите материалы и найдите, для каких ситуаций подходит position: fixed?**

Position- Устанавливает способ позиционирования элемента относительно окна браузера или других объектов на веб-странице.

fixed : привязывается к указанной свойствами left, top, right и bottom точке на экране. **Когда страницу прокручивают, фиксированный элемент остаётся на своём месте и не прокручивается вместе со страницей.**

Подходит, к примеру, для ситуаций, когда есть много элементов и текста. Привязав кнопку «наверх», которая всегда находится на экране, можно быстро попасть в начало страницы. Или для рекламной страницы товара, где всегда на экране у пользователя есть возможность купить товар при помощи кнопки «Купить», которая всегда рядом.

1. **Каким способом лучше всего верстать большие блоки текста?**

Не стоит использовать flexbox для верстки больших блоков текста. В этом случае лучше верстать с помощью Блочных элементов (например, для заголовков и структурных элементов)

1. **Как рассчитывается размер flex-контейнера?**

# При помощи свойств flex-grow, flex-shrink, flex-basis.

**flex-basis-** для установления начальной ширины flex-элементу (посредством различных единиц измерения, таких как px, %, em и др). Конечная ширина flex-элемента будет определяться в зависимости от значений CSS-свойств flex-grow и flex-shrink, которые установлены не только для этого элемента, но и для других flex-элементов этого flex-контейнера.

**flex-grow** – «Жадность»- Это свойство определяет, может ли начальная ширина flex-элемента увеличиваться (расти),  как много свободного пространства во flex-контейнере должно быть назначено текущему элементу. Указывается целое число.

Если элементы имеют одинаковый коэффициент flex-grow, то все они получат одинаковую долю свободного пространства. Иначе- оно распределяется в соответствии с соотношением, определённым различными коэффициентами flex-grow.

Например flex-grow: 2; - ширина элемента будет в 2 раза больше, чем ширина других элементов в контейнере

**flex-shrink**- определяет, может ли ширина flex-элемента уменьшиться, определяет фактор сжатия flex-элемента. Flex-элементы будут заполнять контейнер в зависимости от значения flex-shrink, когда стандартная ширина flex-элементов шире, чем flex-контейнер.

Например flex-shrink: 0; - элементы будут выходить за пределы флекс контейнера

flex-shrink: 1; - элементы не будут выходить за пределы флекс контейнера, размер элементов одинаковый

flex-shrink: 2; - элементы не будут выходить за пределы флекс контейнера

По умолчанию е значения flex-свойств выглядят следующим образом:

* flex-grow: 0; flex-shrink: 1; flex-basis: auto.

Для удобной установки flex-grow, flex-shrink и flex-basis можно использовать CSS свойство flex, например flex: 0 1 auto;