Домашнее задание

💎 Вопросы

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

Flexbox (гибкий контейнер) - это мощный инструмент для создания упорядоченных и адаптивных макетов в веб-дизайне. Он предоставляет простой и эффективный способ организации и выравнивания элементов в контейнере, и в большинстве случаев предпочтителен из-за следующих преимуществ:

Простота использования: Flexbox легко освоить, даже для начинающих разработчиков. Его концепции и свойства понятны и интуитивны.

Упорядочивание элементов: С помощью flexbox вы можете управлять порядком и расположением элементов в контейнере без необходимости изменения порядка элементов в HTML-разметке.

Адаптивность: Flexbox позволяет легко создавать адаптивные макеты, которые могут меняться в зависимости от размера экрана и устройства.

Эффективное выравнивание: Выравнивание элементов как по горизонтали, так и по вертикали происходит с помощью небольшого количества свойств, что делает код более чистым и понятным.

Управление местоположением: Flexbox позволяет легко управлять разделением доступного пространства между элементами, создавая адаптивные и равномерно распределенные макеты.

Поддержка браузерами: Flexbox хорошо поддерживается большинством современных браузеров, что делает его надежным выбором для веб-разработки.

Сеточные макеты: В сочетании с CSS Grid Layout, flexbox позволяет создавать сложные и гибкие сеточные макеты.

Несмотря на преимущества, есть ситуации, когда более сложные макеты могут потребовать комбинации различных методов позиционирования, включая flexbox, CSS Grid, и позиционирование с помощью плавающих элементов или абсолютного позиционирования. Выбор метода зависит от конкретных требований проекта и легко может изменяться в зависимости от конкретных задач.

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

Позиционирование через display: table основывается на использовании CSS-свойств, которые обычно ассоциируются с таблицами HTML. Этот метод позволяет создавать макеты, которые ведут себя подобно таблицам, но без необходимости использовать HTML-таблицы. Вот несколько основных применений для этого метода:

Создание равномерных сеток: display: table может использоваться для создания равномерных сеток с фиксированными или автоматическими ширинами колонок и строк. Это особенно полезно, если вам нужно разделить контейнер на несколько равных частей.

Выравнивание по вертикали и горизонтали: Этот метод обеспечивает простой способ выравнивания элементов как по вертикали, так и по горизонтали. Это может быть полезно при создании центрированных элементов или контроля выравнивания внутри ячеек таблицы.

Работа с переменным контентом: display: table может легко адаптироваться к изменяющемуся контенту. Например, если вы имеете разные по высоте элементы в одной строке, они будут автоматически выровнены по высоте.

Поддержка старых браузеров: Этот метод хорошо поддерживается старыми версиями браузеров, включая Internet Explorer 8 и более ранние версии. Это может быть важным фактором, если вам нужно обеспечить совместимость с устаревшими браузерами.

Однако следует отметить, что display: table имеет свои ограничения и не всегда является лучшим выбором для создания сложных макетов. В некоторых случаях современные методы позиционирования, такие как Flexbox и CSS Grid, могут предоставить более гибкие и эффективные решения.

Таким образом, display: table может быть полезным для простых сеточных макетов и выравнивания элементов, особенно если важна совместимость с устаревшими браузерами.

3. **Какие есть оси во флекс-верстке и как задается их направление?**

Во флекс-верстке (Flexbox), есть две оси: главная ось (main axis) и поперечная ось (cross axis).

Главная ось (Main Axis): Это основная направленность флекс-контейнера. Эта ось определяет, как флекс-элементы (дочерние элементы флекс-контейнера) будут расположены вдоль нее. Главная ось может быть горизонтальной или вертикальной, и ее направление задается свойством flex-direction. Значениями этого свойства могут быть:

row: Главная ось идет слева направо (горизонтальное направление по умолчанию).

row-reverse: Главная ось идет справа налево (горизонтальное направление в обратном порядке).

column: Главная ось идет сверху вниз (вертикальное направление).

column-reverse: Главная ось идет снизу вверх (вертикальное направление в обратном порядке).

Поперечная ось (Cross Axis): Это ось, перпендикулярная главной оси. Поперечная ось используется для управления выравниванием элементов вдоль нее. Например, если главная ось горизонтальная (flex-direction: row), то поперечная ось будет вертикальной и наоборот. Выравнивание элементов вдоль поперечной оси контролируется свойствами align-items и align-content.

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

Свойство margin: auto во флекс-верстке используется для центрирования флекс-элементов вдоль главной оси (main axis) флекс-контейнера. Когда margin: auto применяется к одному или нескольким сторонам флекс-элемента, он автоматически будет центрирован вдоль главной оси контейнера.

Пример использования margin: auto для центрирования элемента по горизонтали внутри горизонтального флекс-контейнера:

```
<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8">

    <title>Центрирование элемента во флекс-контейнере</title>

    <style>

      .container {

        display: flex;
```

```
      justify-content: center; /* Центрирование по горизонтали */
      align-items: center; /* Центрирование по вертикали */
      height: 200px; /* Высота контейнера для демонстрации */
    }

    .item {
      width: 100px;
      height: 100px;
      background-color: #3498db;
      margin: auto; /* Центрирование по горизонтали и вертикали */
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="item"></div>
  </div>
</body>
</html>
```

5. **В чем преимущества `box-sizing`?**

box-sizing - это CSS свойство, которое позволяет управлять тем, как размеры элемента (ширина и высота) вычисляются и как они взаимодействуют с внутренними отступами (padding) и границами (border) элемента. Основные значения этого свойства:

content-box (значение по умолчанию): Размеры элемента вычисляются на основе его контента, игнорируя внутренние отступы и границы. Это означает, что если вы установите

ширину элемента в 100 пикселей, а затем добавите 10 пикселей внутренних отступов и 2 пикселя границы, реальная ширина элемента будет 100 + 10 + 2 = 112 пикселей.

padding-box: Размеры элемента вычисляются на основе его контента и внутренних отступов, но не включают границы. Таким образом, ширина элемента будет равна ширине контента плюс внутренние отступы, но без учета границ.

border-box: Размеры элемента вычисляются на основе его контента, внутренних отступов и границ. Это означает, что если вы установите ширину элемента в 100 пикселей, внутренние отступы в 10 пикселей и границы в 2 пикселя, реальная ширина элемента будет 100 пикселей, и границы и отступы будут включены в эту ширину.

Преимущества box-sizing: border-box:

Поведение более интуитивно: Это более предсказуемый способ задания размеров элементов, так как внутренние отступы и границы включаются в общую ширину элемента. Это позволяет более точно управлять размерами элементов, не переживая "переполнения" контейнера.

Упрощение реакции на изменение размеров: При использовании box-sizing: border-box, изменение размера элемента, например, при ресайзе окна браузера или при разработке адаптивных дизайнов, менее проблематично, так как внутренние отступы и границы не добавляются к общей ширине элемента.

Уменьшение необходимости расчетов: Вам не нужно вычитать внутренние отступы и границы из общей ширины элемента вручную при определении размеров элементов. Это снижает вероятность ошибок и упрощает структуру CSS правил.

В целом, использование box-sizing: border-box может существенно упростить и улучшить управление размерами элементов в CSS, особенно в современной веб-разработке, где адаптивный и отзывчивый дизайн являются важными требованиями.

6. **Чем отличается `flex-grow` от `flex-shrink`?**

flex-grow и flex-shrink - это два разных свойства в CSS, используемые для управления поведением элементов внутри контейнера с использованием flexbox.

flex-grow: Это свойство определяет, насколько элемент может увеличивать свой размер внутри flex-контейнера, чтобы заполнить доступное пространство. Оно принимает числовое значение, которое представляет относительный вес элемента в сравнении с другими элементами в контейнере. Если установить flex-grow для элемента равным 1, он будет занимать все доступное пространство, если оно доступно. Если установить flex-grow для одного элемента равным 2, а для другого равным 1, первый элемент будет расти вдвое быстрее, чем второй.

flex-shrink: Это свойство определяет, насколько элемент может уменьшать свой размер, чтобы уместиться в контейнере, если доступное пространство становится меньше, чем сумма размеров элементов. Оно также принимает числовое значение, которое представляет относительный вес уменьшения размера элемента в сравнении с другими элементами. Если установить flex-shrink для элемента равным 1, он будет уменьшаться в размере так же, как и другие элементы. Если установить flex-shrink для одного элемента равным 2, а для другого равным 1, первый элемент будет уменьшаться вдвое быстрее, чем второй.

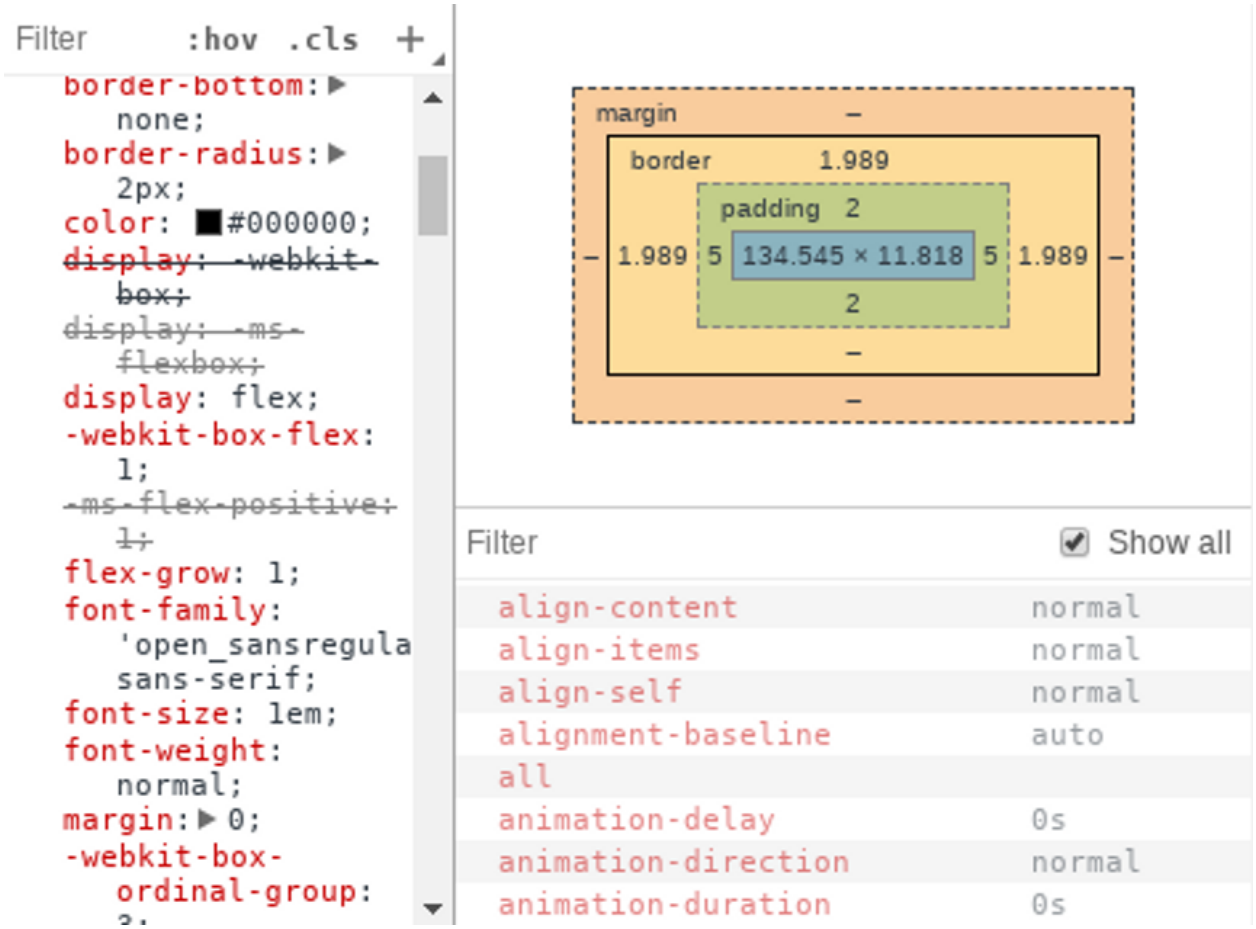
Таким образом, основное различие между flex-grow и flex-shrink заключается в том, как элементы реагируют на доступное пространство. flex-grow управляет ростом элементов, когда есть лишнее место, а flex-shrink управляет уменьшением элементов, когда место ограничено.

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

display: flex;

justify-content: space-between;

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



Размер элемента составляет около 148.523 пикселей в ширину и 17.807 пикселей в высоту

9. Самостоятельно разберись, зачем нужно свойство `order`?

Свойство order используется в CSS флексбоксе для управления порядком элементов в контейнере без изменения структуры HTML. Это свойство задает порядковый номер элемента в последовательности отображения в контейнере. Значение по умолчанию для всех элементов - 0.

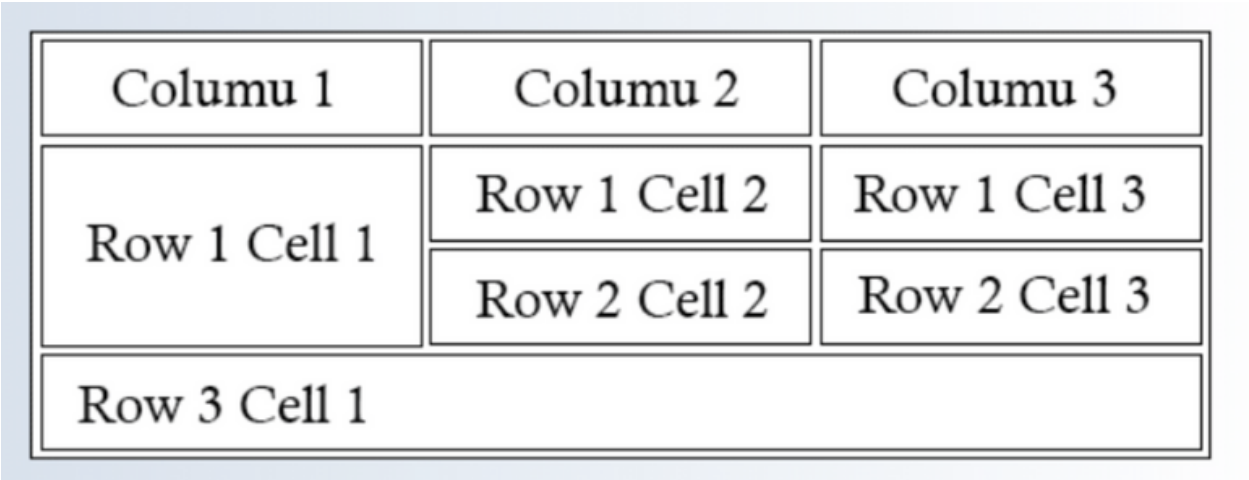
Применение свойства order может быть полезным в следующих ситуациях:

Переупорядочивание элементов: Вы можете изменить порядок отображения элементов без необходимости изменения их расположения в HTML-коде. Это полезно для адаптивного дизайна и изменения порядка отображения на разных устройствах.

Создание сложных макетов: Свойство order позволяет создавать сложные макеты, перегруппировывая элементы внутри флекс-контейнера так, как вам нужно для достижения желаемого дизайна.

Управление порядком анимации: Если вы анимируете элементы и хотите управлять их порядком анимации, order может быть полезным для изменения этого порядка на лету.

10.Каким кодом можно сделать такую таблицу?




```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <table border="1">
```

```
        <tr>
          <td>Column 1</td>
          <td>Column 2</td>
          <td>Column 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>
</body>
</html>
```

11. Как рассчитывается размер flex-контейнера?

Размер flex-контейнера в флексбокс-верстке определяется на основе его содержимого и свойств, которые вы устанавливаете. Размер контейнера складывается из следующих факторов:

Размер содержимого элементов: Размер контейнера зависит от размеров его дочерних элементов (flex-элементов). Это включает в себя ширину и высоту дочерних элементов, а также их внутренние отступы (padding) и границы (border).

Свойства flex-grow и flex-shrink: Если установлены значения flex-grow или flex-shrink, то размер контейнера будет регулироваться исходя из этих параметров. Например, если у элементов установлено flex-grow: 1, то они будут равномерно распределены внутри контейнера, изменяя его размер.

Свойство flex-basis: Свойство flex-basis задает начальный размер элементов до учета их растяжения или сжатия. Если flex-basis установлен, это влияет на исходный размер контейнера.

Размер контейнера по умолчанию: Если ни один из вышеперечисленных факторов не устанавливает размер контейнера, то размер контейнера будет минимальным, необходимым для размещения его содержимого.

Свойство flex-wrap: Если контейнер имеет множество элементов и не умещает их по горизонтали, то свойство flex-wrap определяет, будет ли происходить перенос элементов на следующую строку. Это также влияет на размер контейнера.

Свойство align-content: Если контейнер имеет несколько строк, свойство align-content определяет, как строки располагаются в контейнере, и может влиять на высоту контейнера.

Размеры flex-элементов и параметры контейнера влияют на то, как будет распределено пространство внутри контейнера. Флексбокс позволяет гибко управлять распределением и размерами элементов внутри контейнера, что делает его мощным инструментом для создания адаптивных и удобных макетов.