1. **Фиксированная верстка веб-страницы** — это подход к созданию веб-дизайна, при котором размеры элементов страницы задаются в фиксированных единицах измерения (например, пикселях). Это означает, что при изменении размера окна браузера элементы остаются неизменными по размеру, что может привести к проблемам с отображением на устройствах с различными разрешениями.
2. **Резиновый макет веб-страницы** — это подход, при котором элементы страницы адаптируются к размеру окна браузера. Размеры элементов задаются в относительных единицах (например, процентах), что позволяет им изменять свои размеры в зависимости от ширины экрана, обеспечивая более гибкое отображение на различных устройствах.
3. **Адаптивный дизайн** — это метод веб-дизайна, при котором страницы создаются с учетом различных устройств и разрешений экрана. Адаптивные сайты используют несколько фиксированных макетов, которые автоматически подстраиваются под размеры экрана, обеспечивая оптимальное отображение на всех устройствах.
4. **Медиазапросы** — это CSS-технология, позволяющая применять стили в зависимости от характеристик устройства, таких как ширина экрана, высота, разрешение и ориентация. Медиазапросы подключаются с помощью правила @media, например:

css

Копировать код

@media (max-width: 600px) {

*/\* Стили для экранов шириной до 600px \*/*

}

1. **Правило @import** используется для подключения внешних CSS-файлов в другой CSS-файл. Это позволяет организовать стили и разделить их на несколько файлов для удобства.
2. **Правило @media** применяется для определения условий, при которых определенные стили должны быть применены. Это позволяет создавать адаптивные и отзывчивые дизайны.
3. **Свойство flex** используется для управления поведением flex-элементов внутри flex-контейнера. Оно позволяет задавать размеры, порядок и распределение пространства между элементами.
4. **@media (min-width: 560px)** означает, что стили внутри этого медиазапроса будут применяться только на устройствах с шириной экрана 560 пикселей и более.
5. **@media (max-width: 960px)** означает, что стили внутри этого медиазапроса будут применяться на устройствах с шириной экрана 960 пикселей и менее.
6. **flex: 1 2 200px;** расшифровывается как:
   * 1 — значение flex-grow, указывающее, что элемент может расти, занимая доступное пространство.
   * 2 — значение flex-shrink, указывающее, что элемент может сжиматься в два раза быстрее, чем элементы с меньшим значением.
   * 200px — значение flex-basis, задающее начальный размер элемента.
7. **Значение wrap** используется для указания, что flex-элементы могут переноситься на новую строку, если не хватает места в контейнере.
8. **Свойство flex-flow** комбинирует свойства flex-direction и flex-wrap, позволяя задавать направление и поведение переноса flex-элементов.
9. **Свойство flex-direction** определяет направление, в котором располагаются flex-элементы внутри flex-контейнера (например, по горизонтали или вертикали).
10. **Свойства align-items и align-content** используются для выравнивания flex-элементов по вертикали. align-items выравнивает элементы внутри строки, а align-content управляет пространством между строками.
11. **Свойство align-self** позволяет переопределить выравнивание конкретного flex-элемента, отличное от выравнивания, заданного для всего контейнера.
12. **Свойства flex-контейнера** включают: display, flex-direction, flex-wrap, flex-flow, justify-content, align-items, align-content.
13. **Свойства flex-элементов** включают: flex, flex-grow, flex-shrink, flex-basis, align-self.
14. **Свойство justify-content** используется для распределения свободного пространства между flex-элементами по основной оси (горизонтально или вертикально).
15. **Чтобы вставить видео**, можно использовать тег <video>:
16. **Свойство align-items** применяется ко всему flex-контейнеру и определяет выравнивание всех flex-элементов по вертикали, тогда как **свойство align-self** применяется к отдельному flex-элементу и позволяет переопределить его выравнивание независимо от остальных элементов.
17. **Чтобы создать адаптивное меню**, можно использовать комбинацию HTML и CSS с медиазапросами. Например, можно скрыть элементы меню на мобильных устройствах и заменить их на кнопку, которая будет открывать выпадающее меню. Пример:

html

Копировать код

<nav>

<ul class="menu">

<li><a href="#">Главная</a></li>

<li><a href="#">О нас</a></li>

<li><a href="#">Услуги</a></li>

<li><a href="#">Контакты</a></li>

</ul>

<button class="menu-toggle">Меню</button>

</nav>

css

Копировать код

.menu {

display: flex;

list-style: none;

}

.menu-toggle {

display: none; */\* Скрыть кнопку по умолчанию \*/*

}

@media (max-width: 768px) {

.menu {

display: none; */\* Скрыть меню на мобильных устройствах \*/*

}

.menu-toggle {

display: block; */\* Показать кнопку на мобильных устройствах \*/*

}

}

1. **Документ, в котором цвет абзаца изменяется на красный при изменении ширины до 768 px**:

html

Копировать код

*<!DOCTYPE html>*

<html lang="ru">

<head>

<meta charset="UTF-8">

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

<title>Изменение цвета абзаца</title>

<style>

p {

color: black; */\* Цвет по умолчанию \*/*

}

@media (max-width: 768px) {

p {

color: red; */\* Цвет при ширине до 768px \*/*

}

}

</style>

</head>

<body>

<p>Этот текст изменит свой цвет на красный, если ширина экрана станет меньше 768 пикселей.</p>

</body>

</html>

Этот код создает веб-страницу, на которой текст абзаца будет черным по умолчанию, но изменится на красный, если ширина окна браузера станет меньше 768 пикселей.